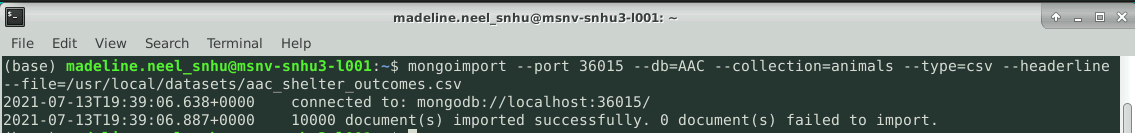
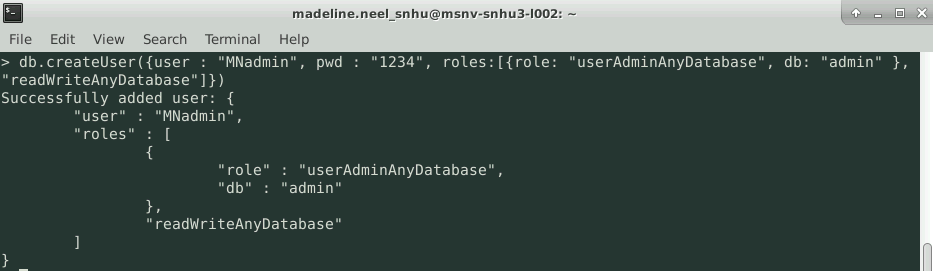
**About the Project**

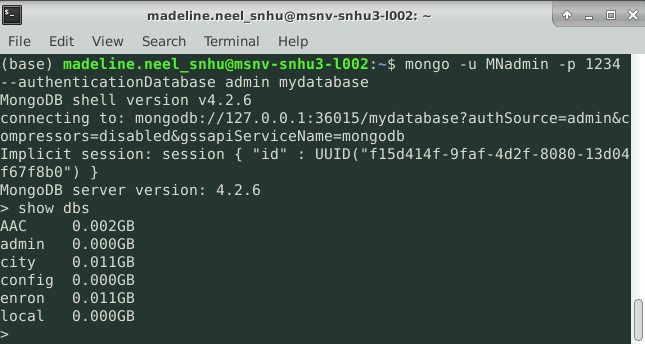
Grazioso Salvare, a rescue-animal training company, hired Global rain to create a software application that will be able to categorize available dogs. Grazioso Salvare hopes to use this application to help find certain dogs that are good candidates for search and rescue training.

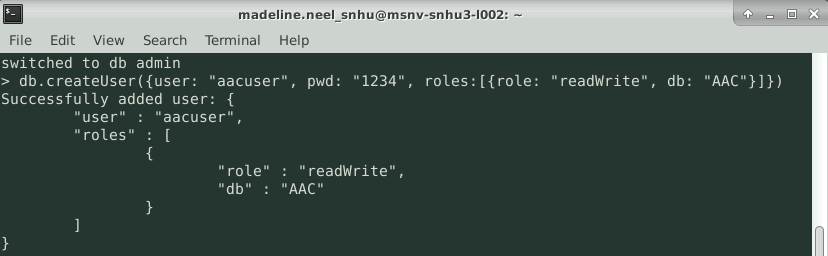
**Getting Started/Installation**

You must have the Mongo Shell and Jupyter Notebook. The Mongo shell will be used to import the given CSV file and to create a user account for that file. To import *Austin Animal Center Outcomes* dataset, you will first open the mongo shell and type in the import phrase: 

Since you are already in the Mongo shell, this would be a good time to create your administrator and user accounts.

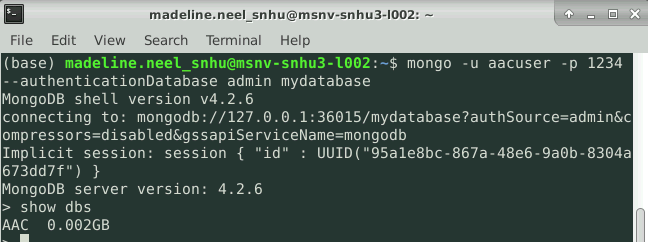
To create the Administrator account: ****

Checking to make sure the Administrator account (MNadmin) was created: 

To create the user account: ****

To make sure the user account (aacuser) was created:

\*This account will only be used for the AAC database (*Austin Animal Center Outcomes)*



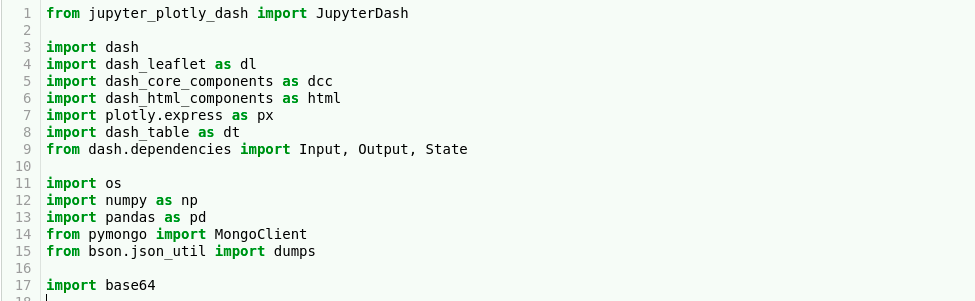
Keep track of your username and password of both accounts as it will be used in the Jupyter Notebook files. Jupyter Notebook will be used to create both a .py file for the CRUD functions and .ipynb to work with the given dataset.

The libraries used for the *animals.py* (Python File) are:

\*The Python file will be created in Jupyter Notebook but will be made with the “text file”



The libraries used for the *ProjectTwoDashboard.ipynb* (Jupyter Notebook) are:



**Usage**

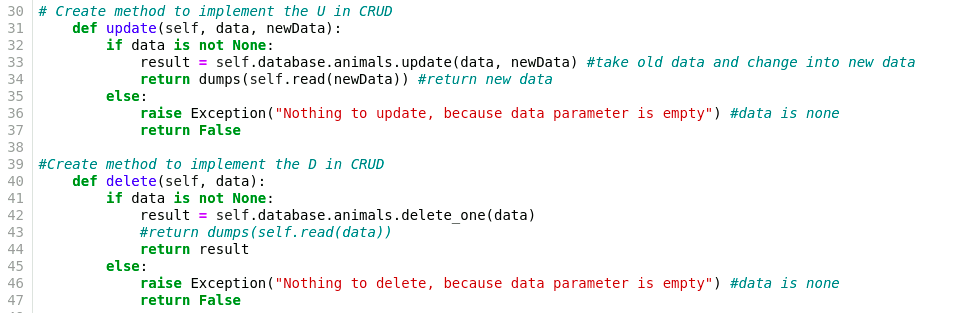
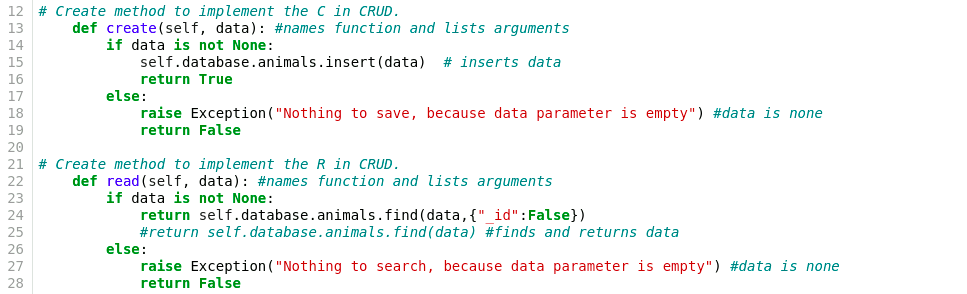
**Code Example**

Python modules are beneficial to create because of their reusability. Due to that, I will show you how to create a reusable .py file for our software.

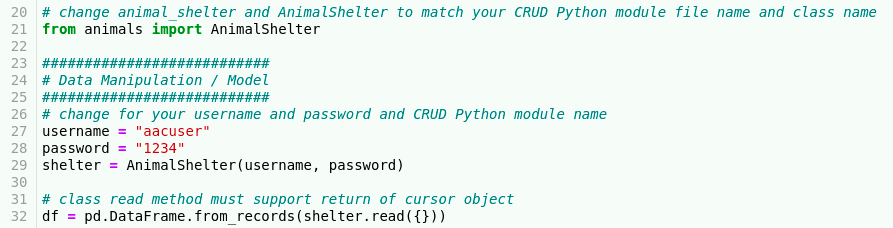
animals.py

After the libraries are imported in the *animal.py* file (shown above), you will want to initialize the class and describe the purpose of the module. Following creating the class, you will want to initialize MongoClient to access the MongoDB databases and collections. These steps are shown by: 

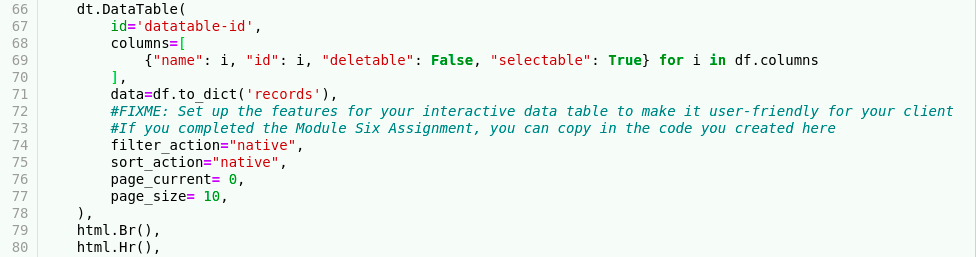
Next, you will want to create the create, read, update, and delete method:

\*The main method we’ll be using for this dashboard is *def read*

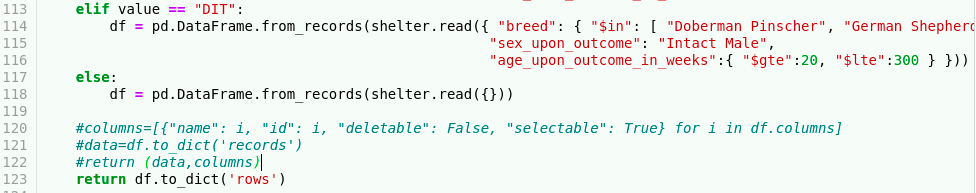
ProjectTwoDashboard.ipynb

After the libraries are imported in the ProjectTwoDashboard.ipynb (shown above), it is time to call your CRUD python module file that we just created. You will, also, hardcode in your username and password that we created in the Getting Started section and attach it to your imported class. Next we will be calling our Read method from the python module to return a cursor object. These steps will be done by:

If you choose to add in the Grazioso Salvare Logo, you can access it [here](https://learn.snhu.edu/content/enforced/795519-CS-340-H6993-OL-TRAD-UG.21EW6/course_documents/Grazioso%20Salvare%20Logo.png?_&d2lSessionVal=n5ZStD4PM58bAtpIsNB4UMkKM&ou=795519). Once you have downloaded it, upload it into the same folder that has your .py and .ipynb files. Once that is completed, you will get it ready to place by: 

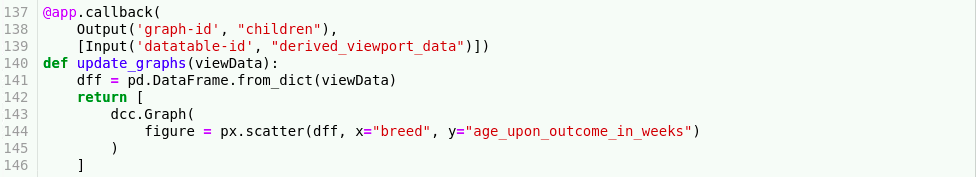
The following screenshot will set up the dashboard to have each widget placed in the correct areas. Line 51 is a personal, unique identifier. Line 52 is placing Grazioso’s logo. Line 56 to 64 is creating the buttons that will be used to filter the dataset. Line 66 to 78 is creating the data table and including the filters for it to be interactive. Line 82 to 92 is setting up the dashboard so the chart is on the left and the geolocation chart is on the right. More information on data tables can be found [here](https://dash.plotly.com/datatable). 

To make the radio buttons work, you will want to input the radio-id and establish what the radio values will be searching for:

\*These radio items will call the read method from our python file.

This will create a graph that is dependent on the radio button selected:

\*You can find more information on scatter graphs [here](https://plotly.com/python/line-and-scatter/) as well as find other types of graphs [here](https://dash.plotly.com/dash-core-components/graph).

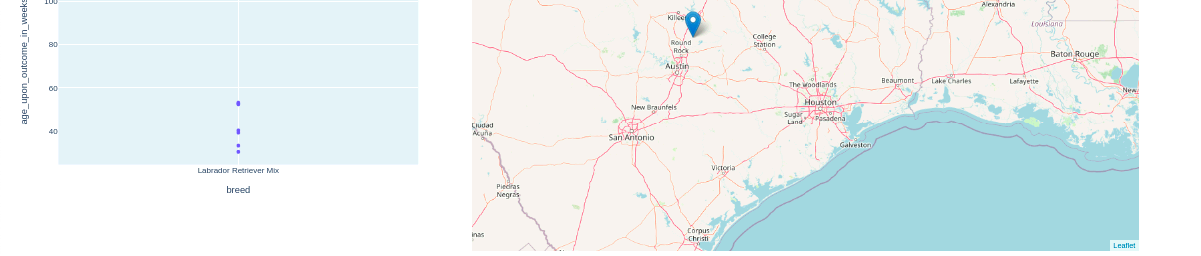
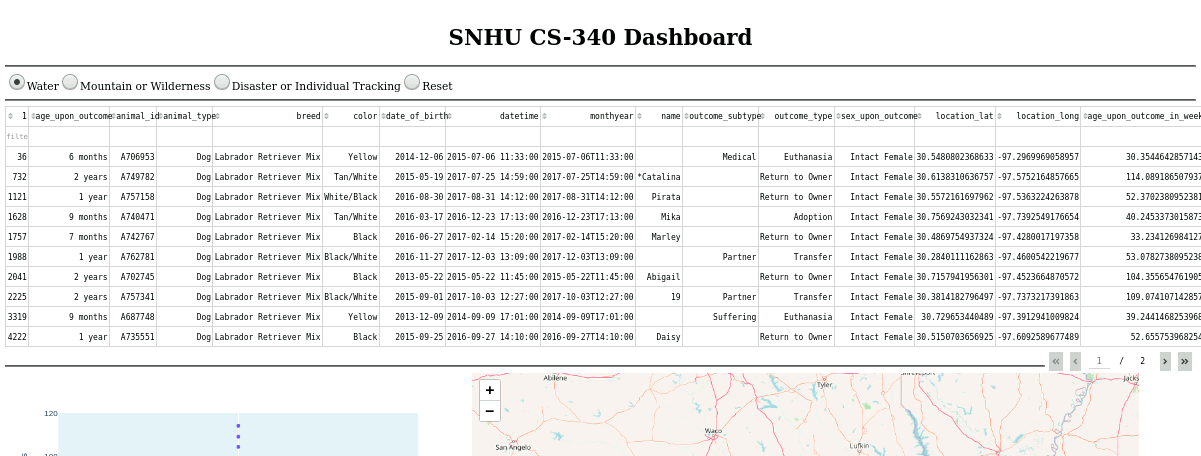
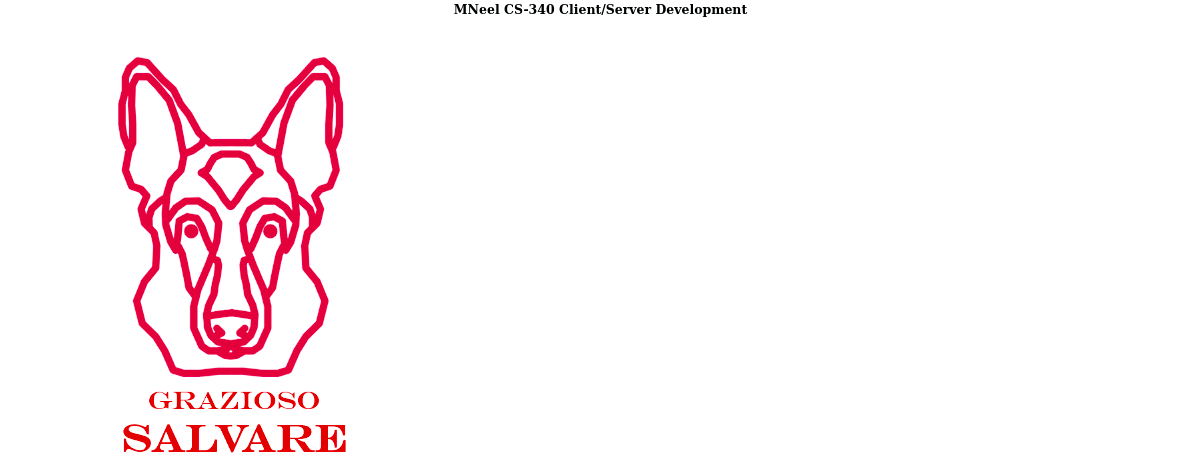


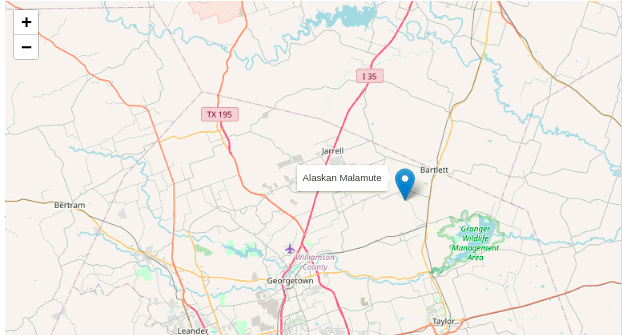
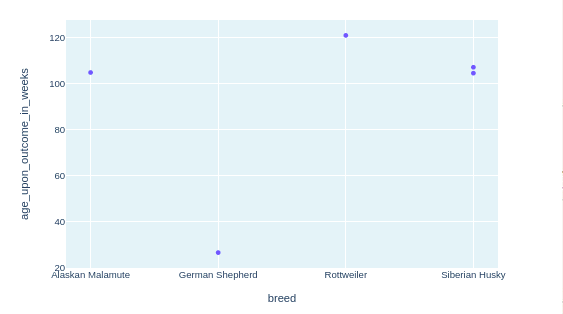
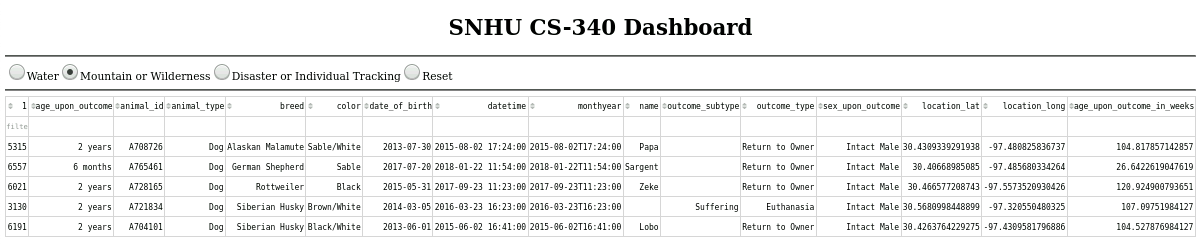
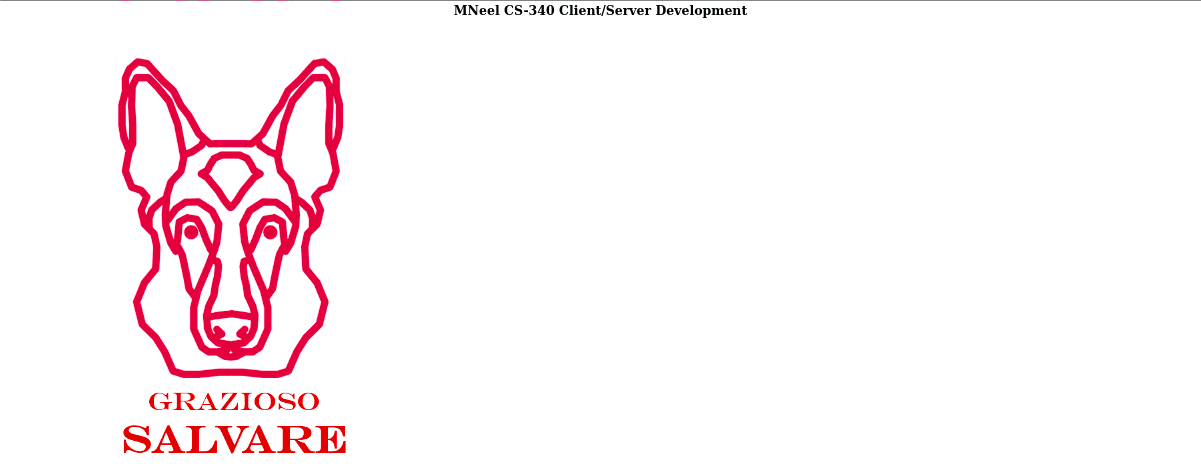
This will create a geolocation on the selected animal in the data table. More information on geolocation maps can be found [here](https://dash-leaflet.herokuapp.com/).

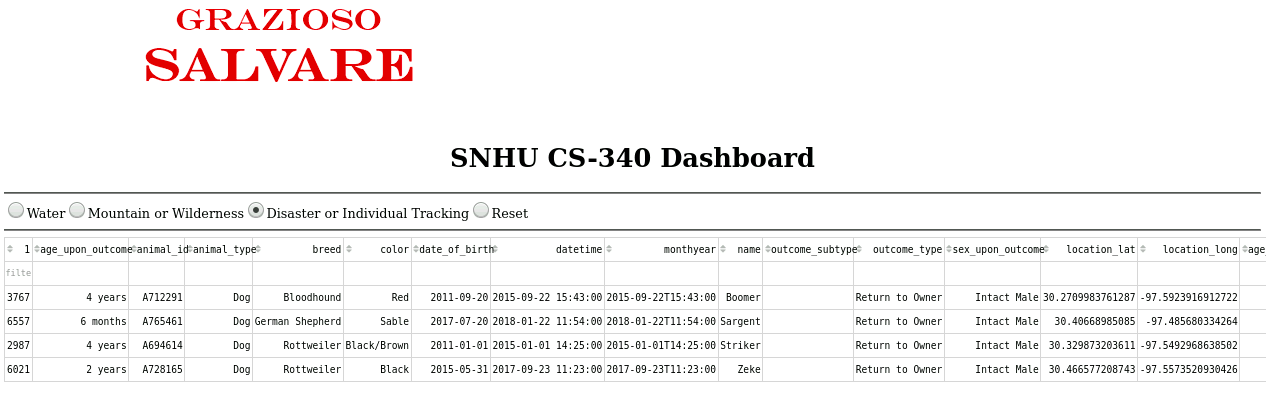
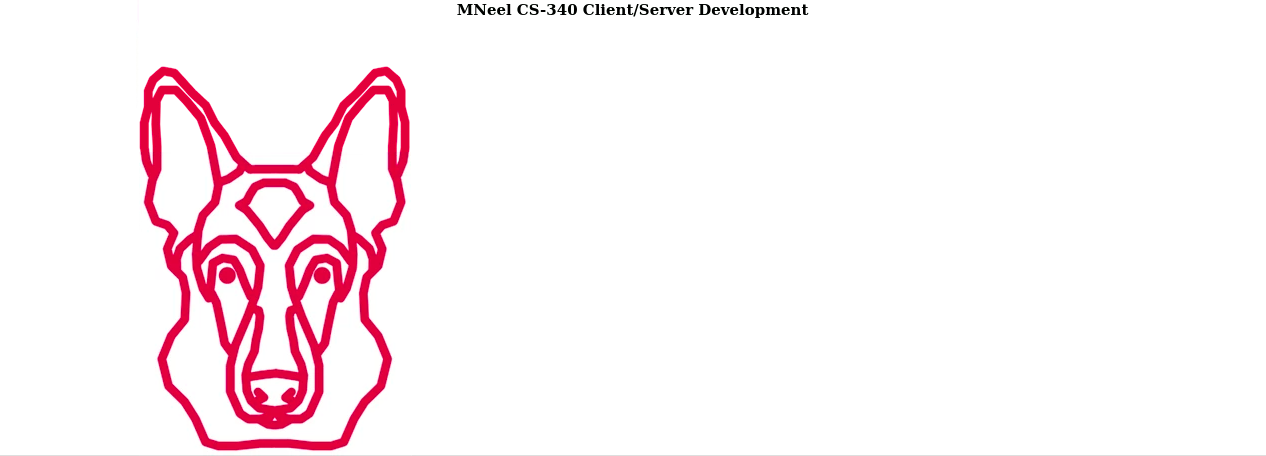


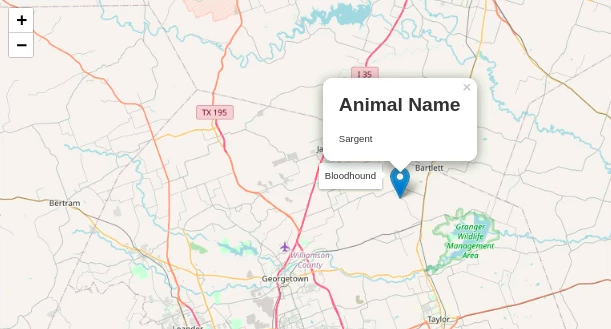
**Results**

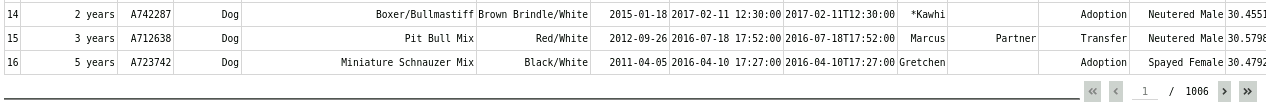
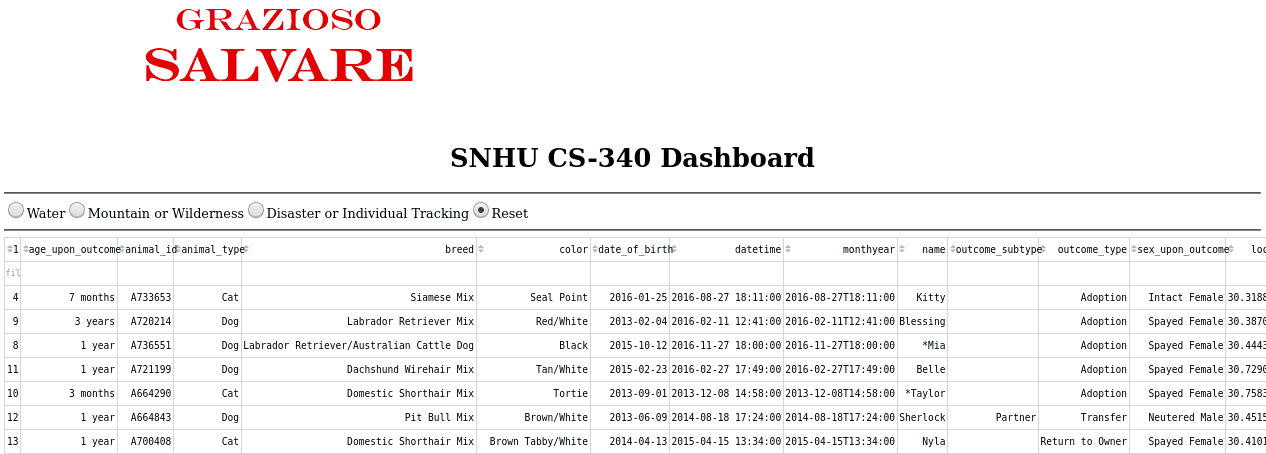
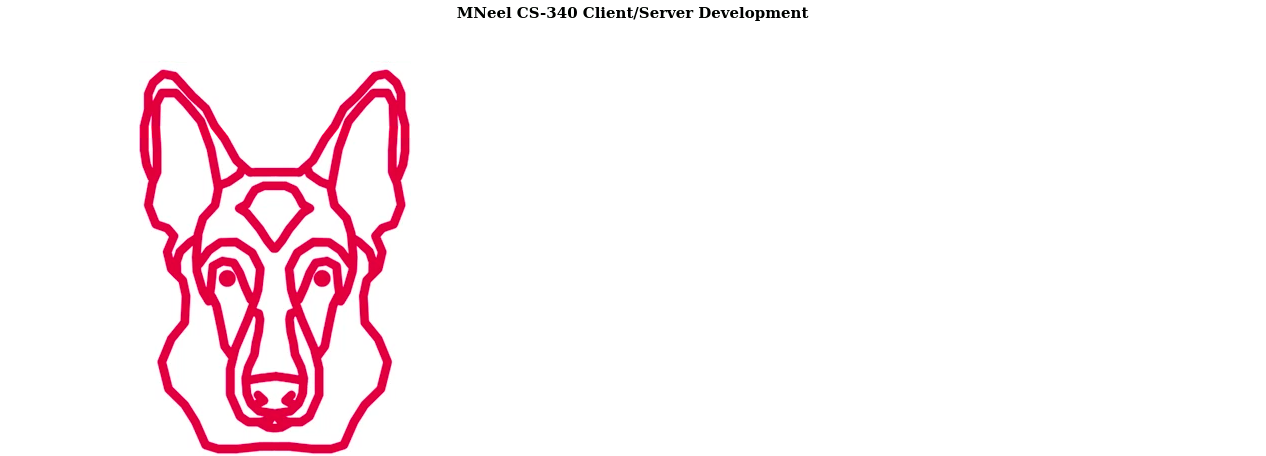
Water

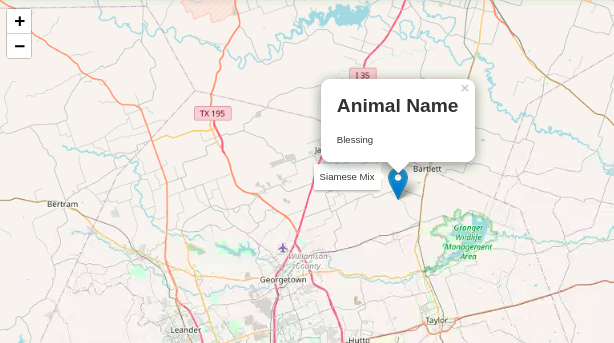
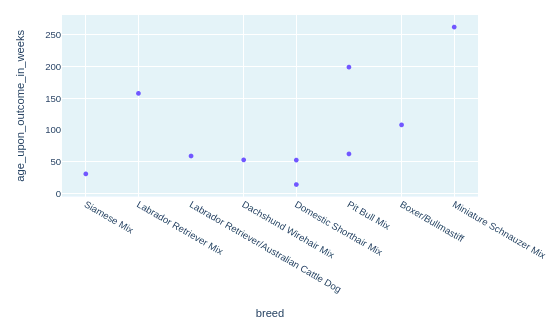


Mountain or Wilderness

Disaster or Individual Tracking

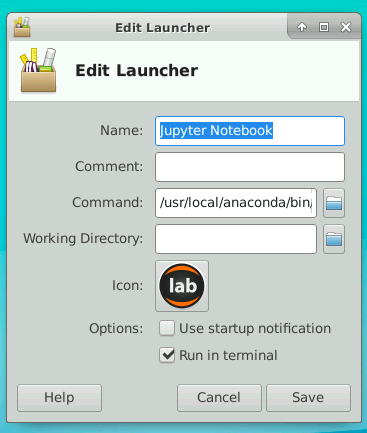
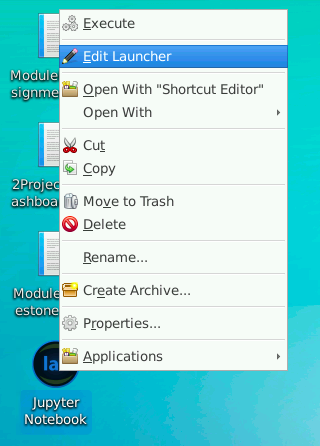


Reset



**Challenges**

Errors. Errors happen in every code, but it helps knowing how to read and identify them to be able to fix them. It is important to note that you must start the mongo shell with */usr/local/bin/mongod\_ctl start* before trying to run anything in Jupyter – if you do not, you will get a runtime error. A way to set up your Jupyter Notebook to show you the errors is by opening a terminal. First start by right clicking your application on the desktop, hitting edit launcher, and putting a check in the checkbox next to run in terminal.



**Contact**

Madeline Neel