**README FILE**

**Attn: G. Salvare**

**About the Project**

MongoDB is a tool we will be utilizing to provide an easy-to-use interface for managing the creation, reading, updating, and deleting of data for the AAC Shelter. This project includes a module for CRUD testing, a database, and the addition of Dash to incorporate multiple apps including: a table, a geolocator, and a pie graph.

**Motivation for the Utilization of MongoDB & Python**

MongoDB and Python go together incredibly well and with the ease of set up it was an easy choice to implement MongoDB over other options. A major highlight for MongoDB is the navigability of the databases compared to competitors, such as Blazer or other SQL based database IDEs. Because of the reality of utsing CRUD, having a language such as Python to use - powerful enough to run those intense SQL level queries but digestible and user-friendly for less involved usage - is an asset for the longevity of the projects application.

**Motivation for the Utilization of DASH**

Dash is a tool that is incredibly dynamic and easy to work with, especially in the case of needing a lightweight and easy to use tool for data visualization. HTML tags can be used to update visuals of apps, easily adjust labelings, change filters, and so much more. We can also use callbacks with Python which means each tool we have selected integrates well together!

**Getting Started**

To set up a local copy there are quite a few steps and it is imperative that they are followed closely to reduce system breakage or invalid database information. These steps can be further broken into groupings to allow time for testing and ensuring a correct set up.

1. **Mongo Database Creation:**
   1. **Set Up** - install the initial MongoDB environment following the instructions at mongodb.com - the MongoDV interface and system itself will be how you interact with the data once set up is finalized. This will also store the future dbs (databases) and collections (further parsed files).
   2. **Create a DB** - Create a db (database) for the AAC Shelter
   3. **Import the CSV** - this is the file containing the required data for the database. The file that was provided to us was called “aac\_shelter\_outcomes.csv” however this may differ depending on your requirements for this database.
   4. **Verify the Import** - Ensure that the file has uploaded appropriately by testing a few MongoDB commands and practicing navigating through the database, finding a file, or even creating one (db.collections.find() is an example of how to locate files.)
   5. **Create Authentication** - Once you have ahad an opportunity to understand MongoDB a little bit better, it is imperative that authentication is set up so that unauthorized users can’t access your system. You can create administrative users and users that can just read and write - depending on what you need.
2. **Python CRUD Module:**
   1. **CRUD Module Creation -** This is an application that will reacts to the dynamic state of the database and update as necessary to align with the database itself.
      1. **CRUD** stands for **C**reate, **R**ead, **U**pdate, **D**elete
   2. **Coding** - The code we have developed can be provided and should be formatted as a Python Module
   3. **Testing** - Once the module has been successfully compiled the next step is to test the module by adding the code to Jupyter Notebook. The file should either create unique data each time it is run, however there may be cases where deleting the created file will suit better in order to verify the testing has run successfully.
3. **DASH:**
   1. **App Creation - Dash utilizes Plotly and requires being imported into the module for the great applications:**
      1. **Map -** The map app can be created by providing a call back within the file and allows the user to utilize a table or other data format to select, filter, and mark coordinates for locating and/or labeling data within the physical world.
      2. **Interactive Graphing -** While there are many options for data visualization, based off the data we were utilizing in this project a pie chart was our best option due to it’s high visibility for data and quick imagery for comparisons. You can find more on interactive graphing here: https://dash.plotly.com/interactive-graphing
      3. **Data Table -** Lastly, a tried a true method of data visualization that is easy to filter through is the data table. Through appropriate nomenclature and sectioning of data this table is the main tool that many other apps can be built off of. You can find more here: https://dash.plotly.com/datatable
   2. **Callbacks -** Callbacks are the code that allow the apps to be called and visualized, have rules attributed, and filtering options further elaborated on in order to better visualize data. You can find more here: https://dash.plotly.com/advanced-callbacks
   3. **Search/Filtering Queries -** Queries to read and filter through data can be a heavy lift in some situations depending on how much information you need, but through the usage of appropriate syntax and tools like “radio buttons” or “dropdowns” you can more effectively gather the data you require without combing through immense amounts of information over unreasonable amounts of time. You can read more here: https://dash.plotly.com/datatable/filtering
   4. **More Information -** 
      1. **Plotly** - For more instructionals and walkthroughs you can review this website as the information available is extensive and self-paced. https://dash.plotly.com/introduction

**Installation**

There are a few tools that are necessary to run this project successfully. They are listed below, along with a few key high-level details, and installation links.

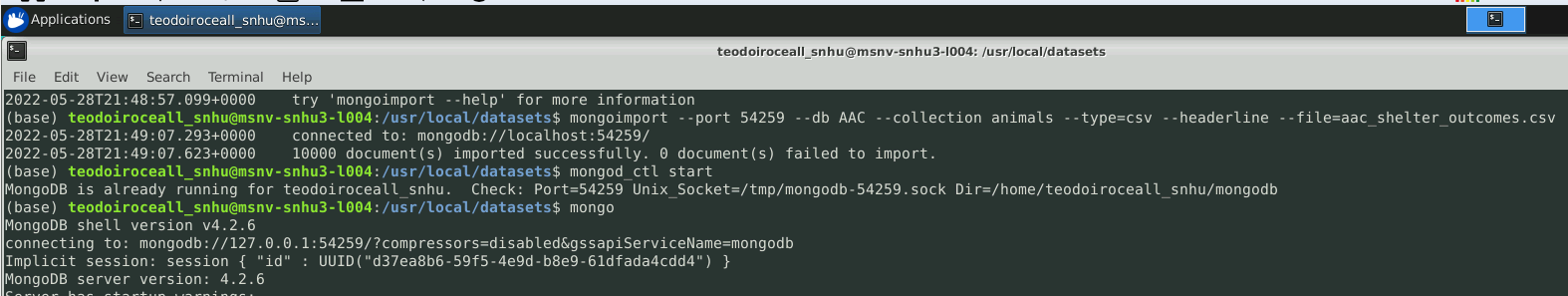
1. **MongoDB:**
   1. MongoDB has both an Enterprise and a Community edition and as a business I would recommend the Enterprise level as your business grows, however for now Community will work great.
   2. Installation Link: https://docs.mongodb.com/manual/installation/
2. **Python:**
   1. Python is an easy-to-use and incredibly versatile coding language that will be used to test. It can be accessed via your operating system’s terminal.
   2. Installation Link: https://realpython.com/installing-python/
3. **Jupyter Notebooks:**
   1. Jupyter is a tool that is used alongside Python to create modules that will be utilized for testing.
   2. Installation Link: <https://jupyter.org/install>
4. **DASH:**
   1. Dash is a framework that can be importedand installed for creating interactive and visually appealing dashboards with minimal experience.
   2. Installation Link: https://pypi.org/project/dash/

**Usage**

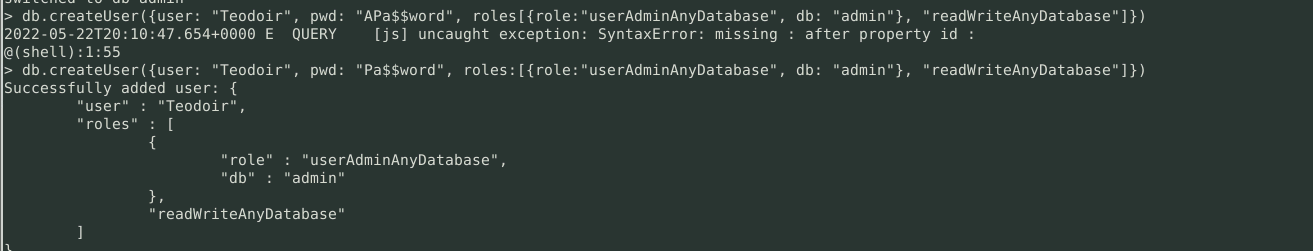
The CRUD Module was created with Python and tesed to ensure that the appropriate responses are being elicited from the database. With MongoDB itself there are ample ways that the terminal can be used to amend the database based on what the team needs. Additionally, the implementation of Dash with Plotly has allowed us to implement some great visuals with relative ease. I have shared some great examples in the “Code Example” section below. The CRUD Module itself is being utilized to test and showcase the ability to create, read, update, and delete files through a system with a better GUI. Further down you will see the additions of the dashboard and apps. This series of screenshots can be utilized as additional support during the set up process to ensure you aer achieving the correct results!

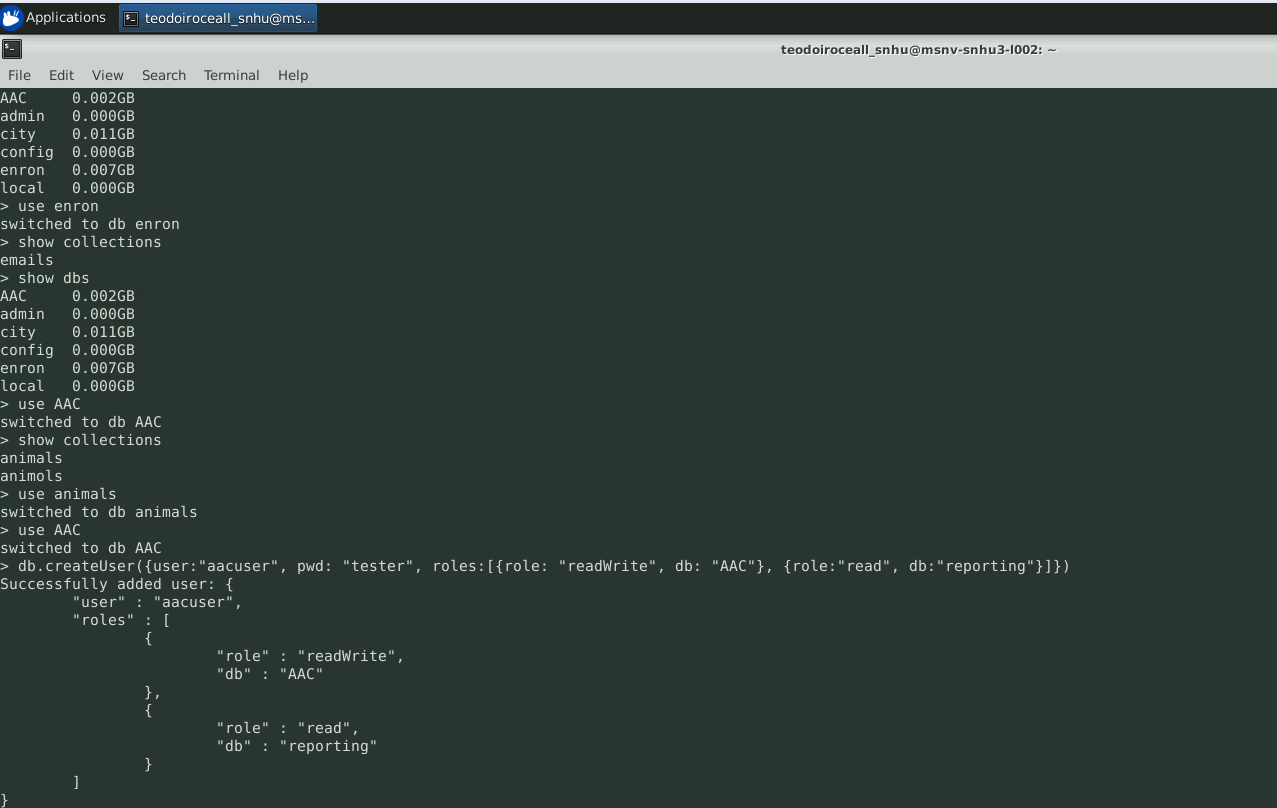
**Code Examples**

* **MongoDB:**
  + **Importing a File:**

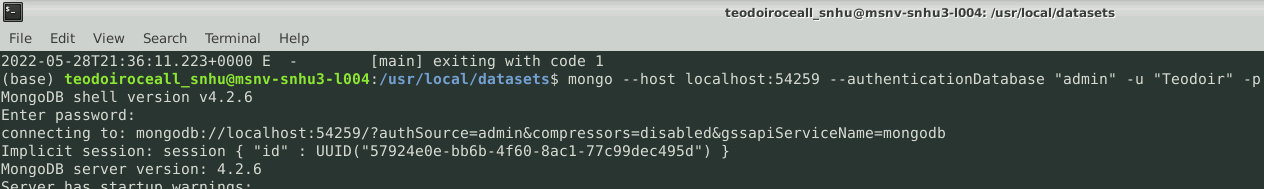
****

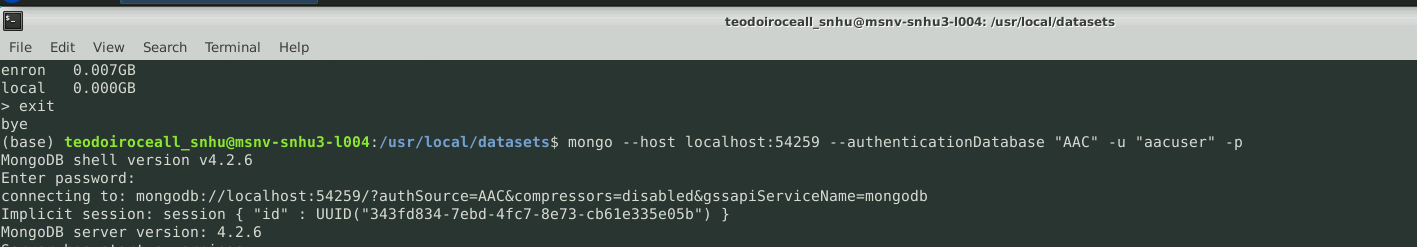
* + **User Authentication Set Up - Admin and Read/Write User:**

****

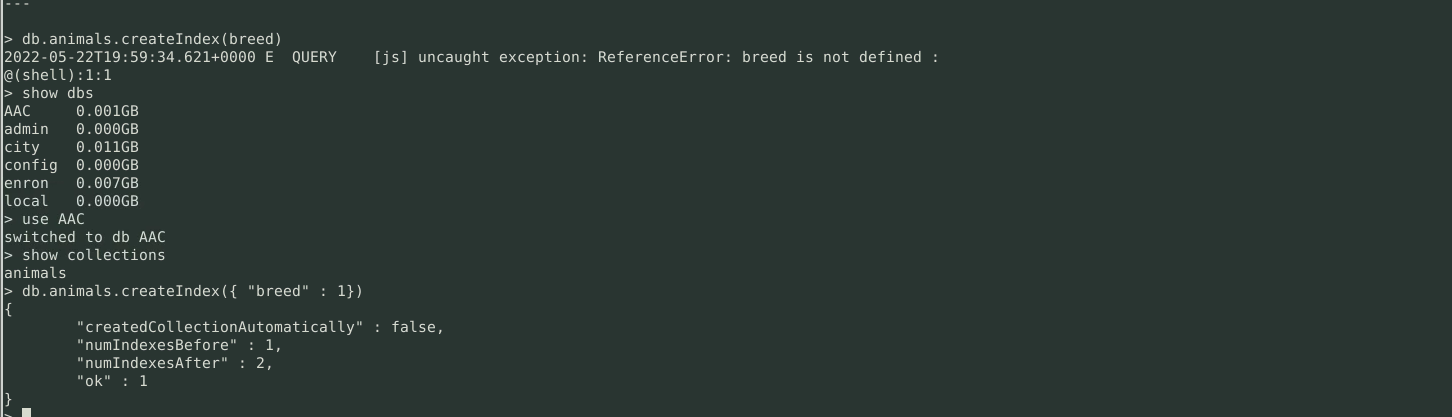
****

* + **User Authentication Log In - Admin and Read/Write User:**

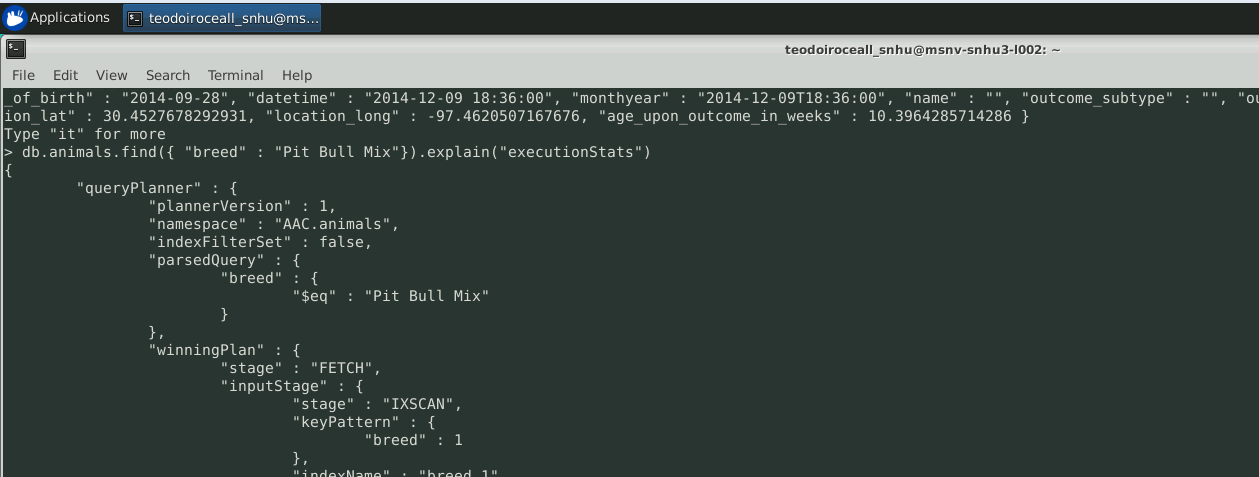
****

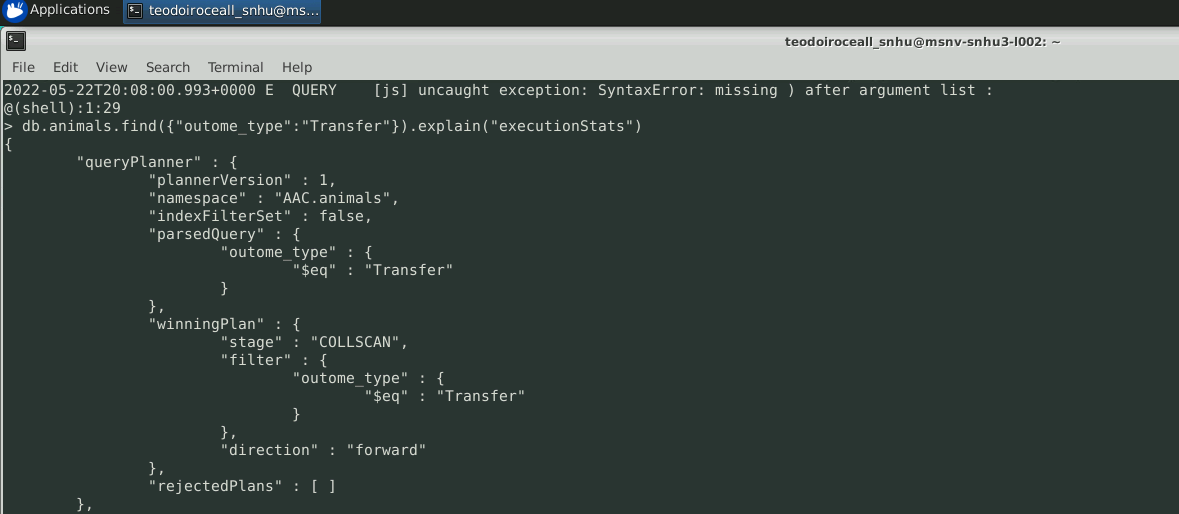
****

* + **DBS & Collection Navigation:**

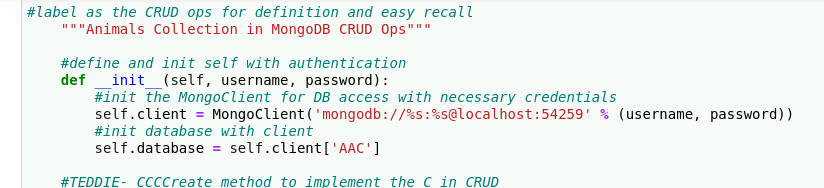
****

* + **Information Navigation:**

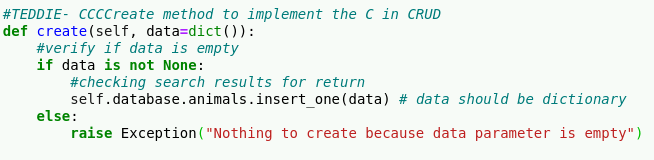
****

****

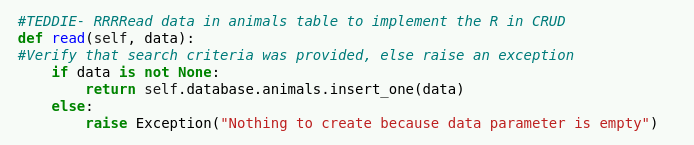
* **Python CRUD Module:**
  + **CRUD INIT:**

****

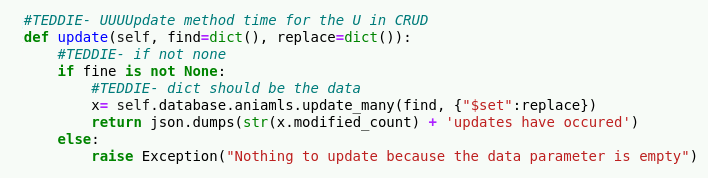
* + **C - Create:**

****

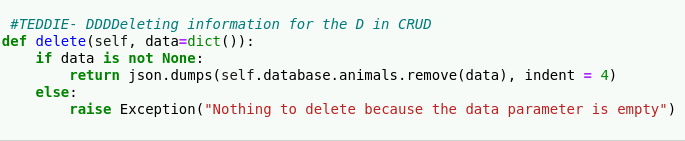
* + **R - Read:**

****

* + **U - Update:**

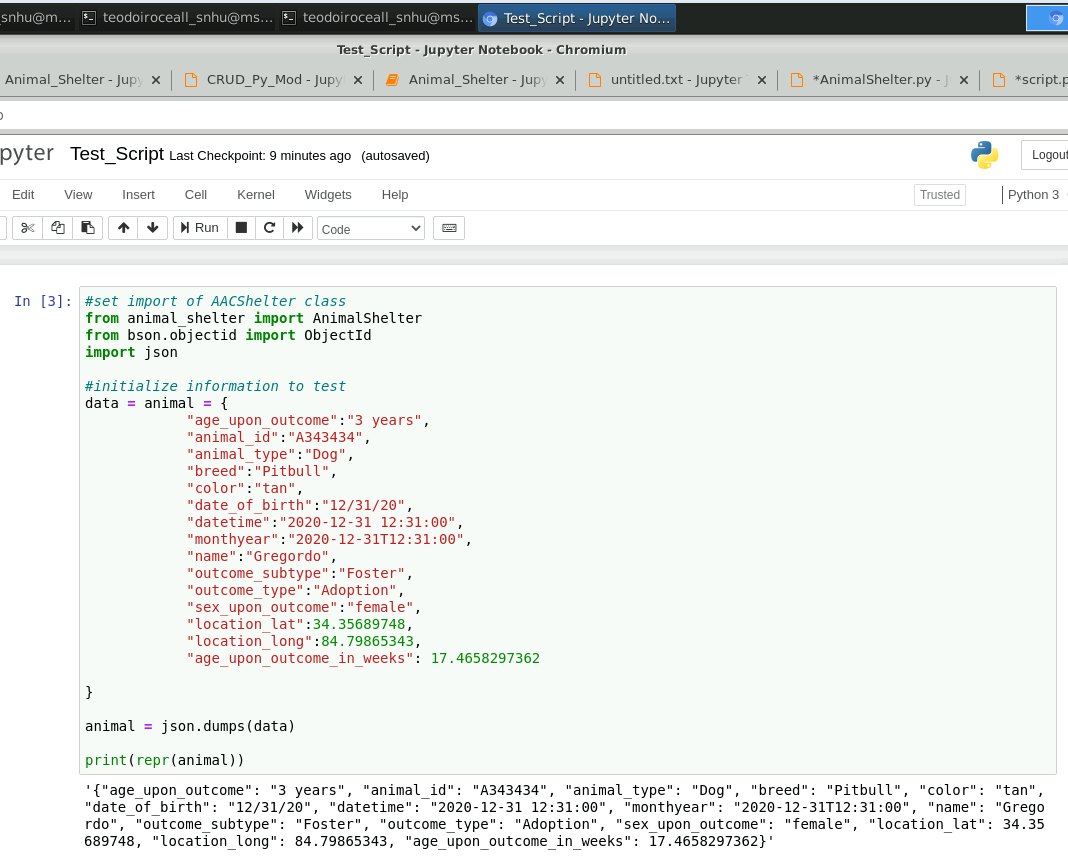
****

* + **D - Delete:**

****

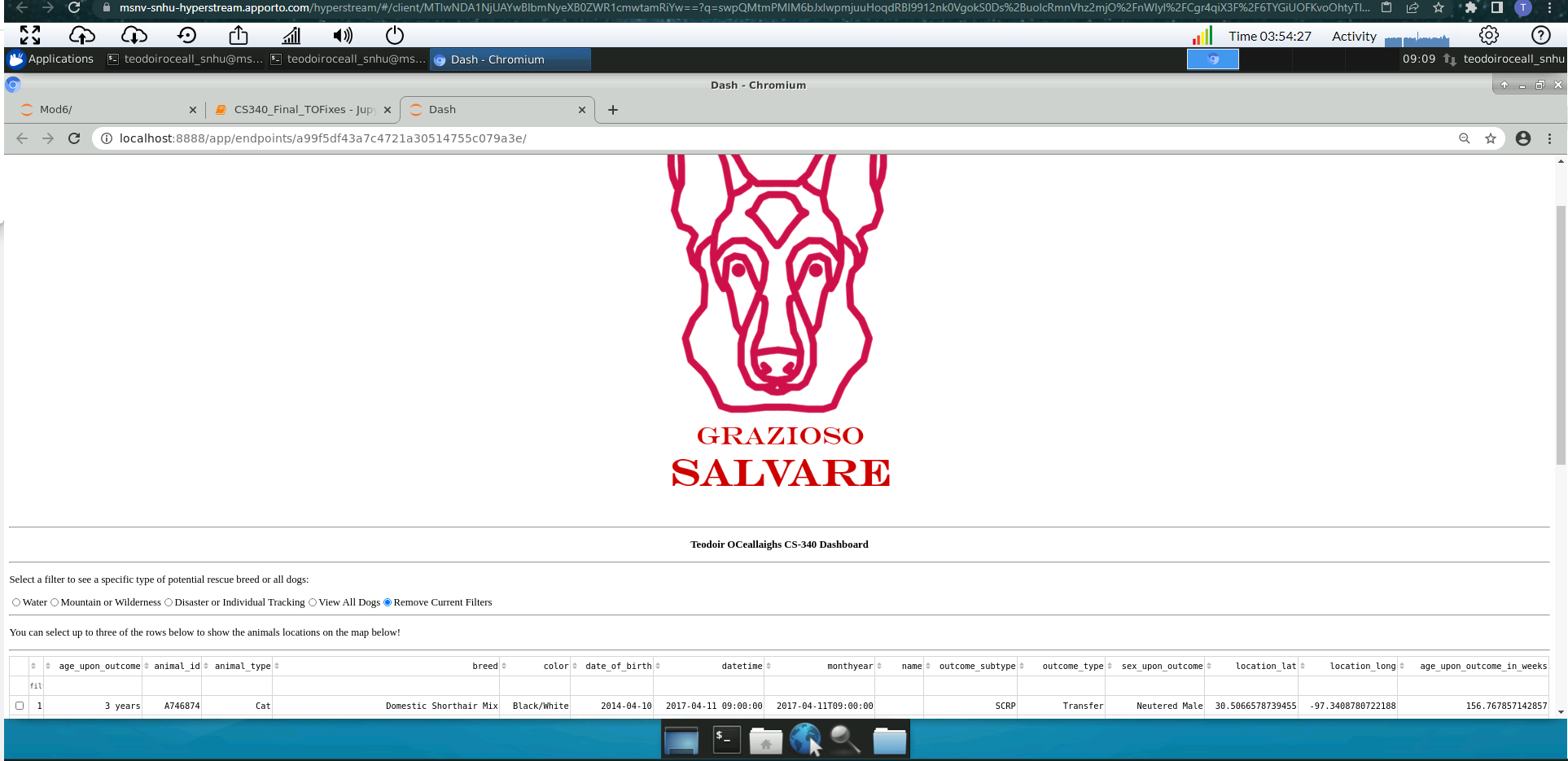
**Tests:**

* + **Test Code:**

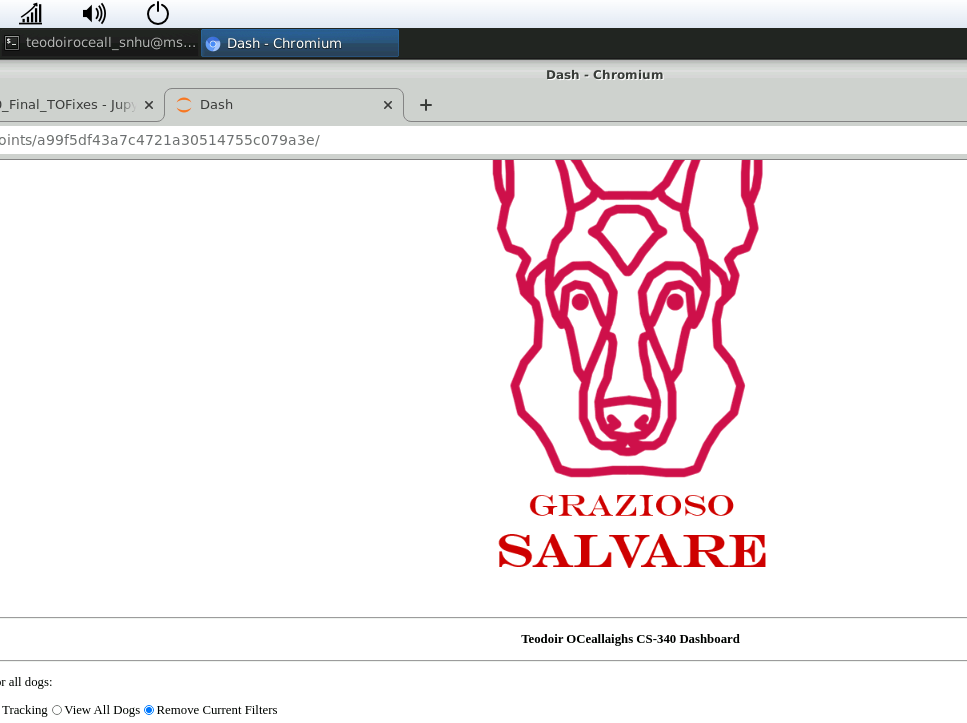
****

**Dashboard Visualization:**

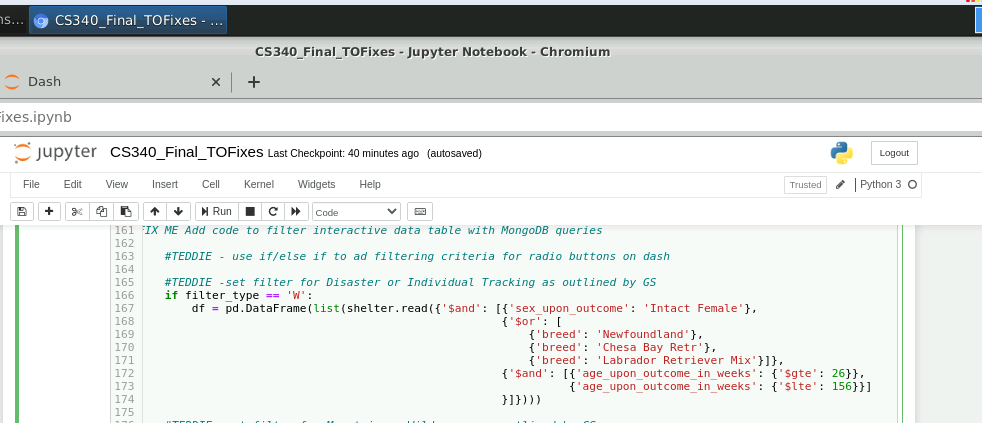
1. **Company Logo Present:**

****

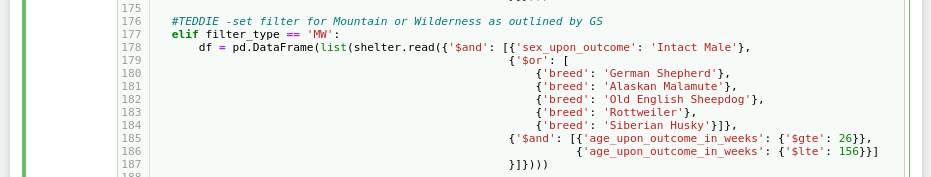
1. **Unique Identifier:**

****

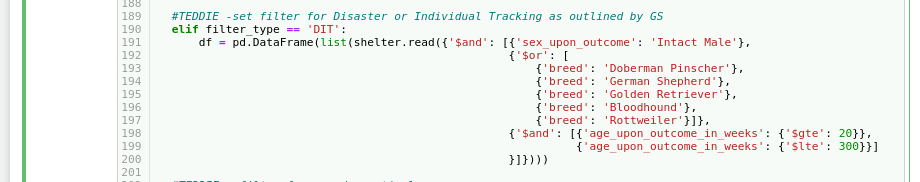
1. **Filter Options:**
   1. **Water Rescue Filter:**

****

* 1. **Mountain/Wilderness Rescue:**

****

* 1. **Disaster or Individual Tracking Rescue:**

****

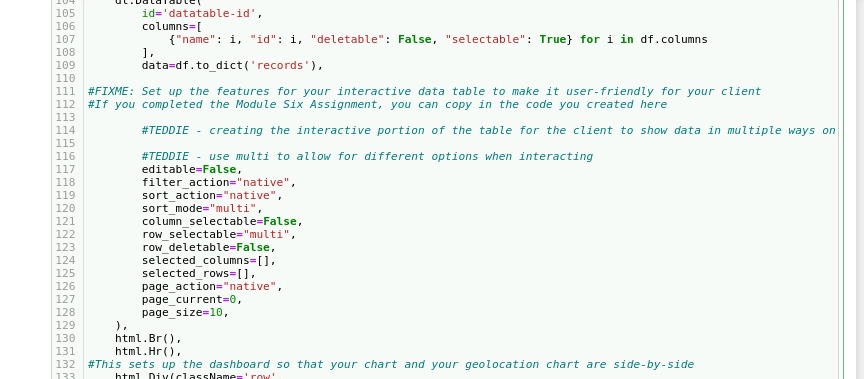
* 1. **Option for all dogs:**

****

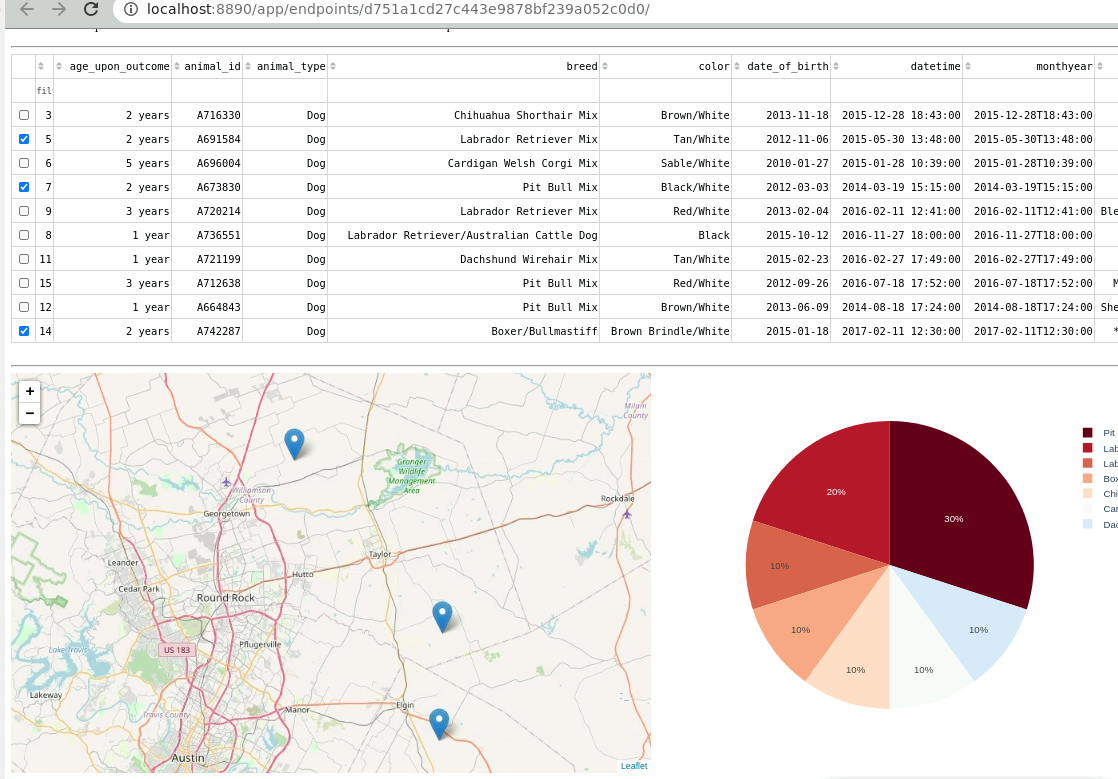
* 1. **Reset Filters:**

****

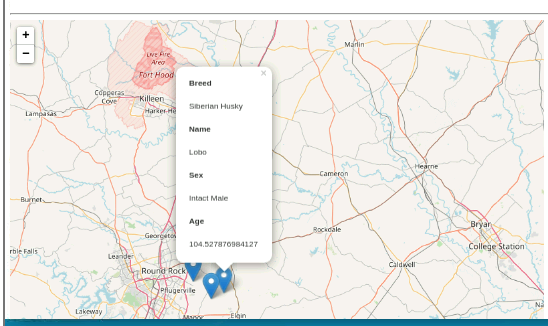
1. **Table View:**
   1. **Multiple Row Selection:**

****

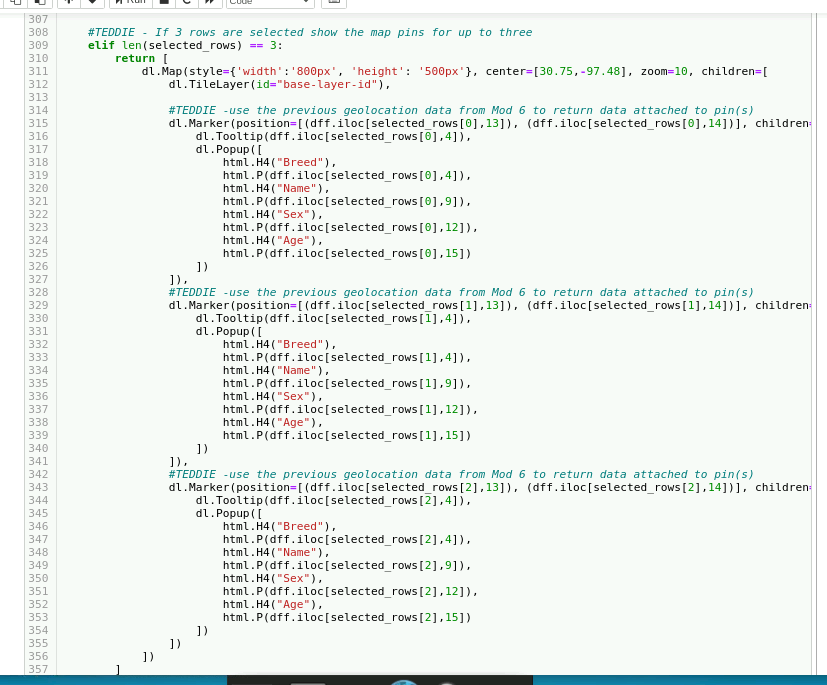
* 1. **Multiple Row/Map Marker Coordination:**

****

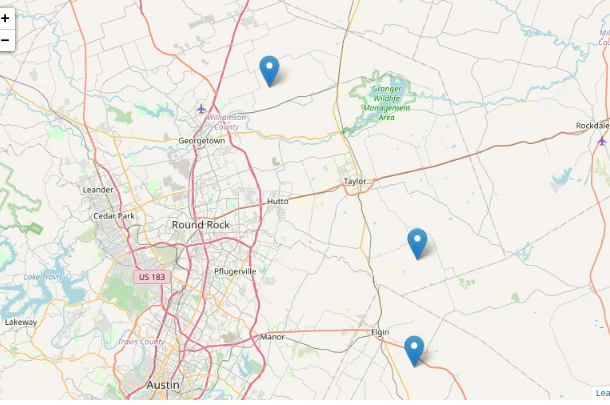
1. **Geolocation/Map App:**
   1. **Label:**

****

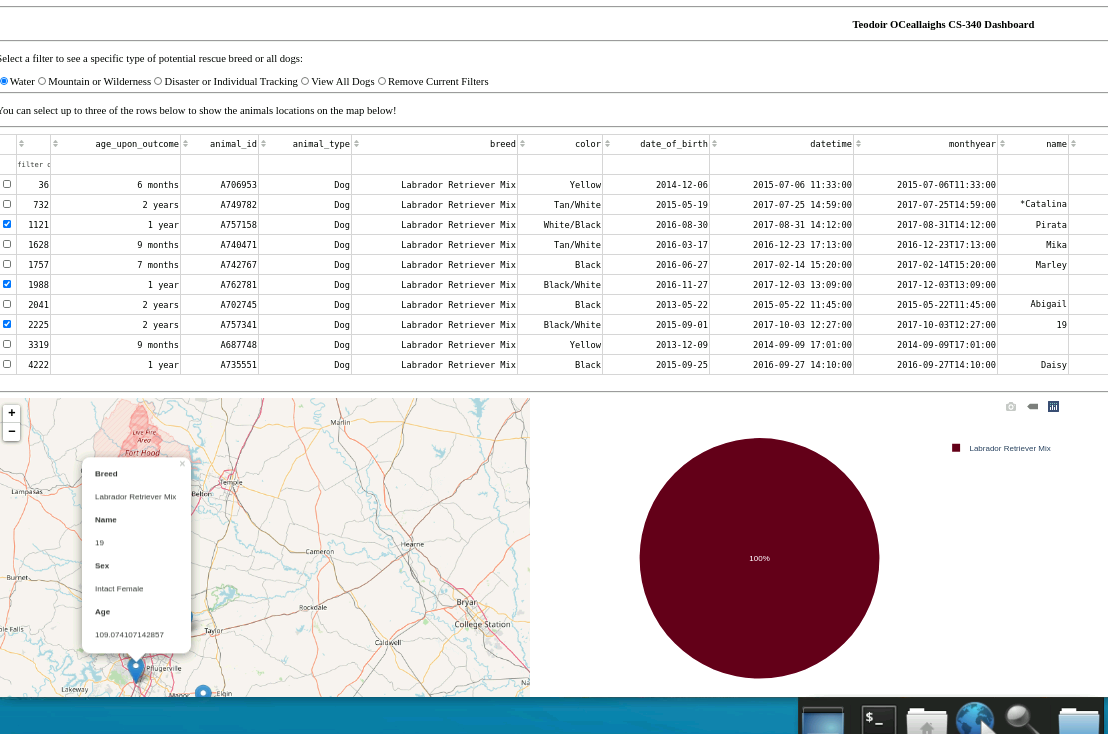
* 1. **Multi-Selection Code:**

****

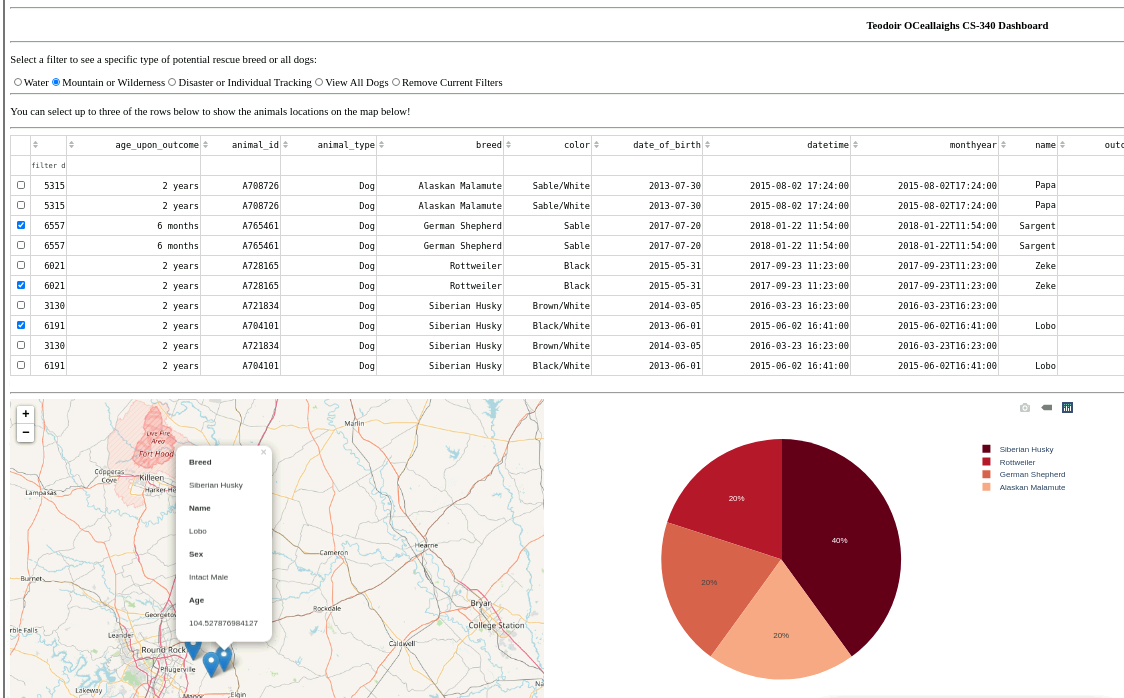
* 1. **Multi-Selection Visual:**

****

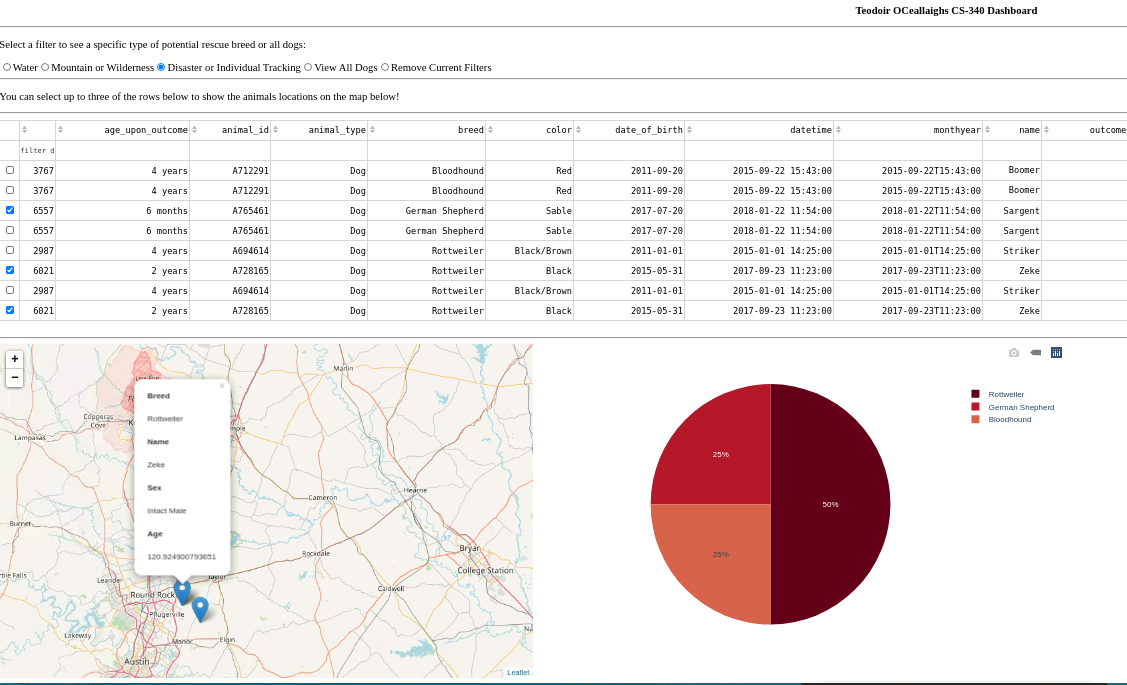
1. **Pie Chart:**
   1. **Water Rescue Filter:**

****

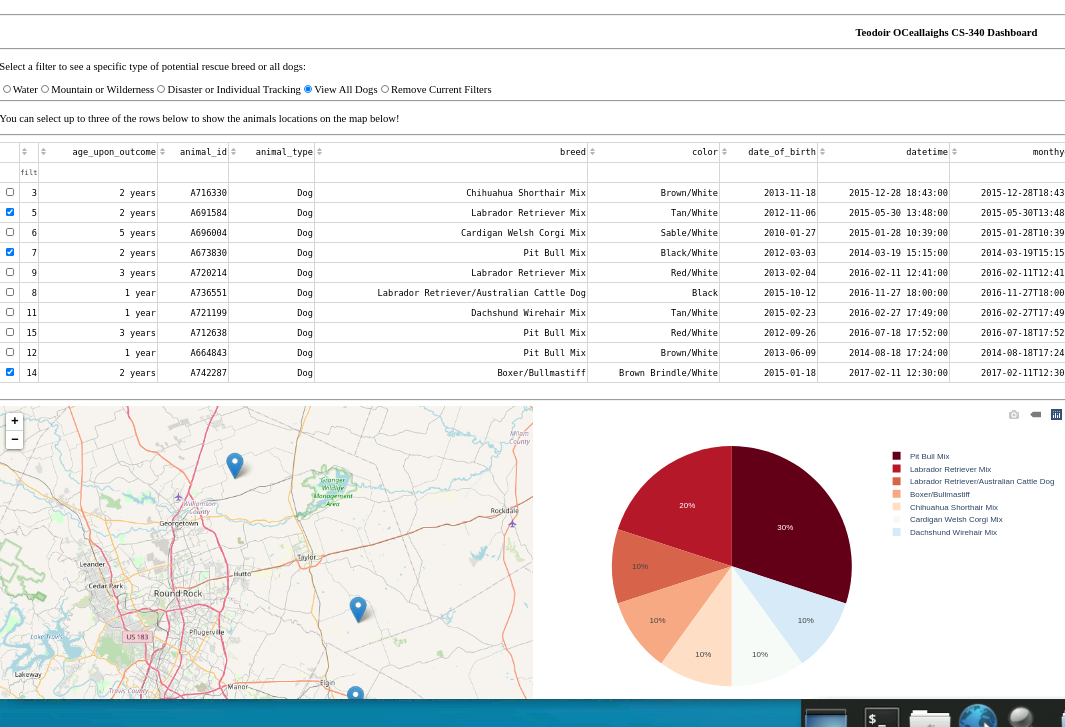
* 1. **Mountain/Wilderness Rescue:**

****

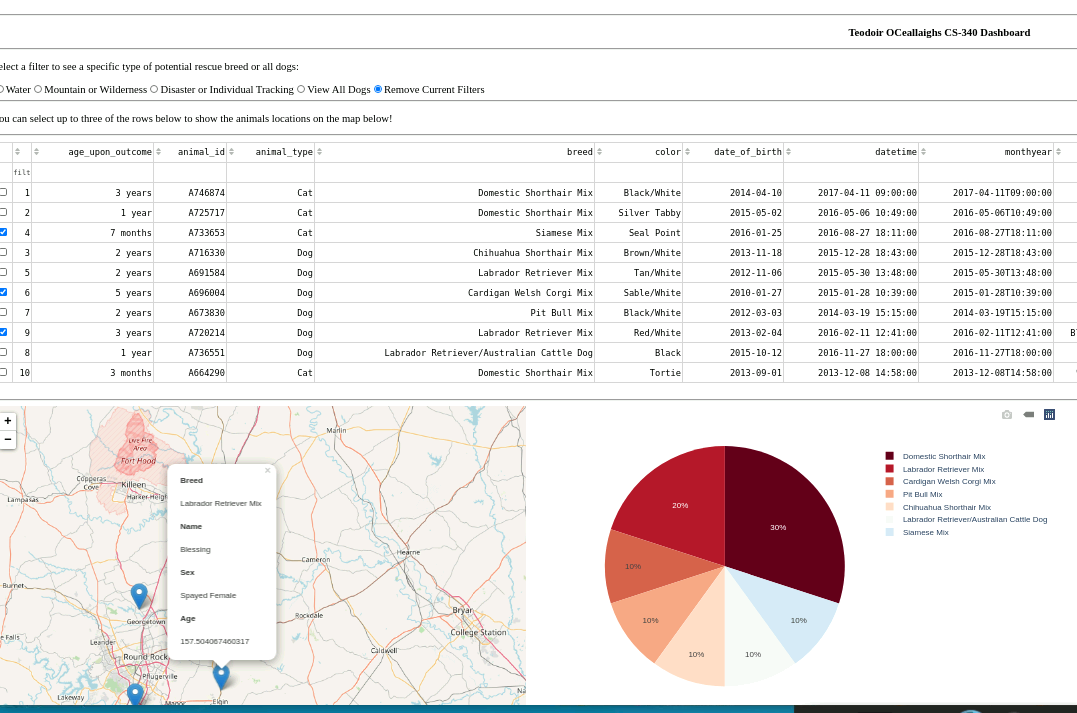
* 1. **Disaster or Individual Tracking Rescue:**

****

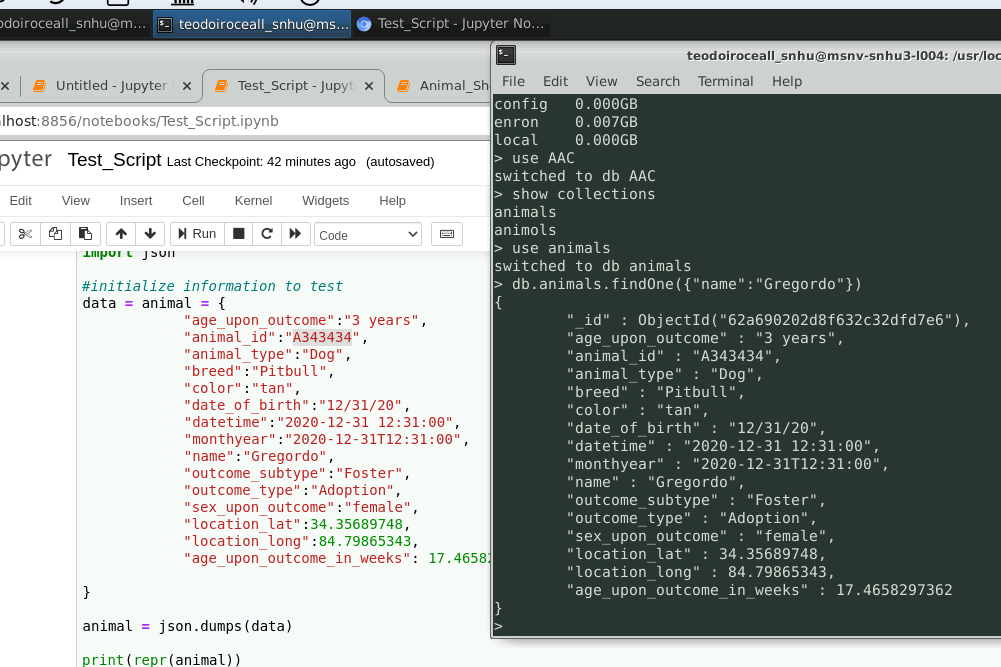
* 1. **Option for all dogs:**

****

* 1. **Reset Filters:**

****

**Extra Visuals:**



**Roadmap/Features:**

From past iterations I found it limiting to only view one animal at a time on the map considering the sheer size of Austin. I added functionality for 3 rows to be selected and viewed at the same time if travel is an important consideration. In the future the goal will be to add filters for age, for more options with mixed breeds, and improved structure for the table!

**Contact: teodoir.oceallaigh@snhu.edu**

**Your name: Teodoir O’Ceallaigh**