# CS 340 README

## About the Project/Project Title

This project built on work with MongoDB to create CRUD functionality in the Python Middle-ware layer of a stacked development. The primary purpose of the project was to create a Create, Read, Update and Delete (C, R, U and D) functionality for the Grazioso Salvare database and then further build a dashboard that will enable Grazioso to efficiently identify dogs that are good candidates for search-and-rescue training. This project enabled me to develop the “glue” of reusable code that would hold the development together while also laying the foundation for events beyond the current horizon, where the code could interface with additional web applications or other programs.

## Motivation

The primary motivation of this project is to expand my knowledge of a stacked development project while making use of reusable and adaptable code that will enable projects to interface with other web applications and programs. By further developing my use of reusable code and developing the python middle-ware, or “glue”, of the project, I am developing my project to have a smoother transition to a different base or server level. I further expanded my knowledge by applying my stacked development to a web application and created a basic dashboard.

## Getting Started

In order to get started with this project, you will need access to MongoDB. For this project, I was provided the Austin Animal Center (AAC) Outcomes data and I imported the data into MongoDB. Upon opening the mongo shell, the user can switch to the newly imported database and can perform queries on the data present.

## Installation

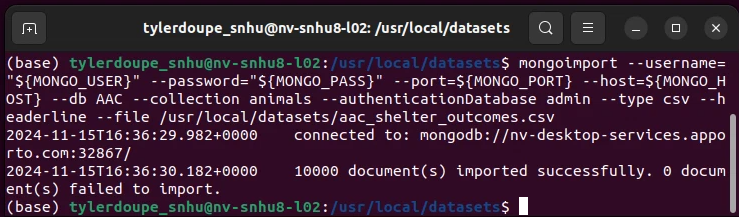
You will need the following software in order to complete this project. I have also included the software versions that I have used to produce the results displayed later.

* MongoDB 6.0.13 – A NoSQL database used for its flexibility and schema-less design
* Python 3.9.12 – Used as the primary programming language to support CRUD operations and for testing the newly created CRUD operations
* Jupyter Notebook 6.4.8 – A Python friendly IDE used to write Python code
* PyMongo – A Python library used for MongoDB interaction that offers efficient database operations
* Dash Framework – Provided the frontend view and backend logic for a web application to be built. Enabled the use of tables, charts and filters in the dashboard along with dynamic callbacks and smooth data handling of large information sets.

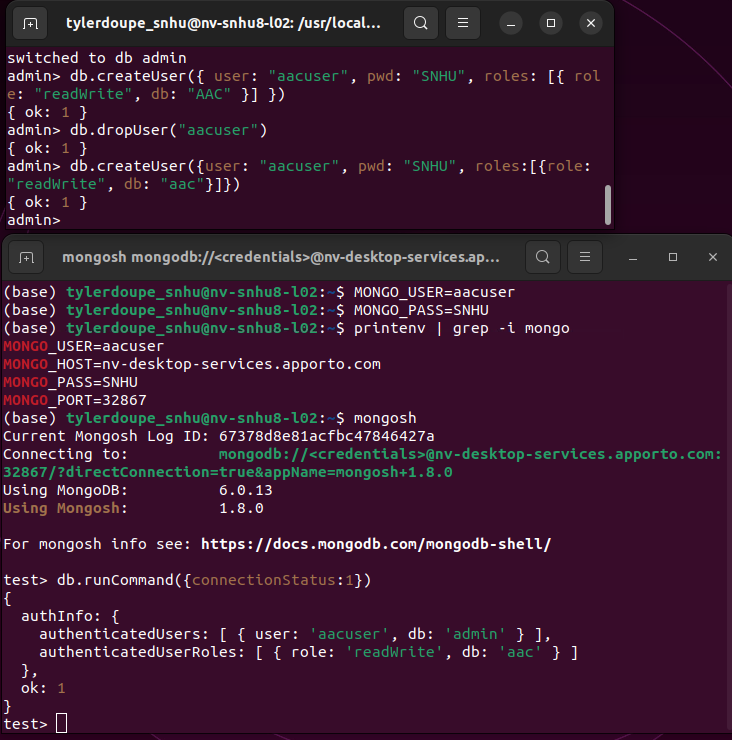
## Usage

### Code Example

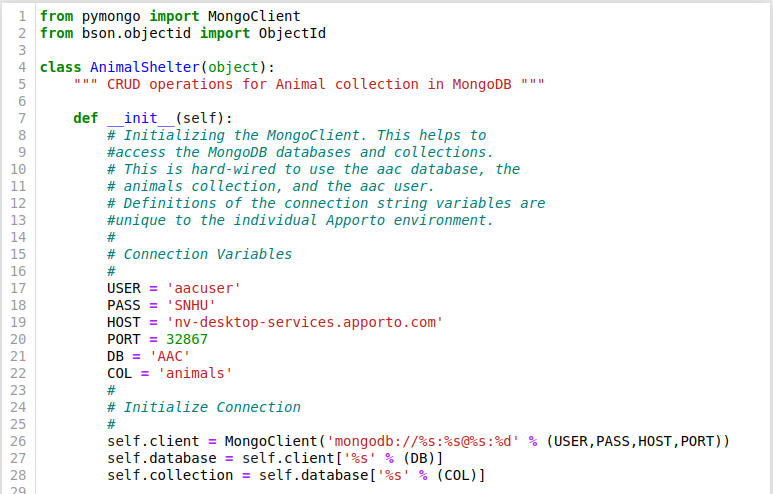
After successful download and set up of Python, Jupyter Notebook and MongoDB, you can begin to import the database into MongoDB:



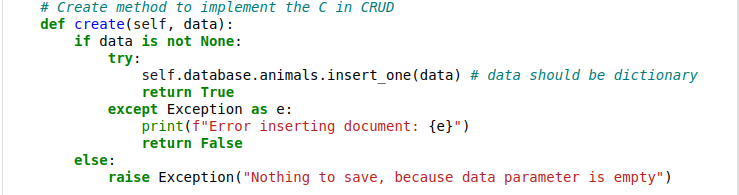
Once the database has been imported into MongoDB, you can enter the mongo shell, switch to the admin account and create an “aacuser” account that has read/write permissions within the database. Following successful creating of the “aacuser” account, in a second terminal window, the authentication of the new account can be tested to see if the new account can access the database:



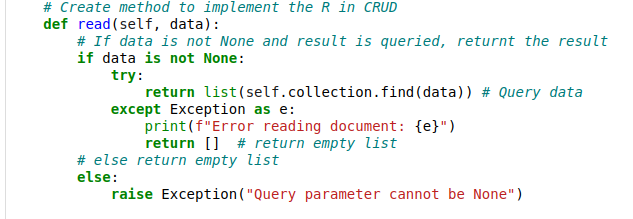
The next steps for this project include creating a Python file in order to create the Create and Read methods (C and R). To start, this involves importing the MongoClient and importing ObjectId, followed by creating an object class called AnimalShelter. Following the creation the AnimalShelter class object, the MongoClient will need to be initialized in order to access the MongoDB databases and collections using the “aacuser” connection criteria that you created previously:



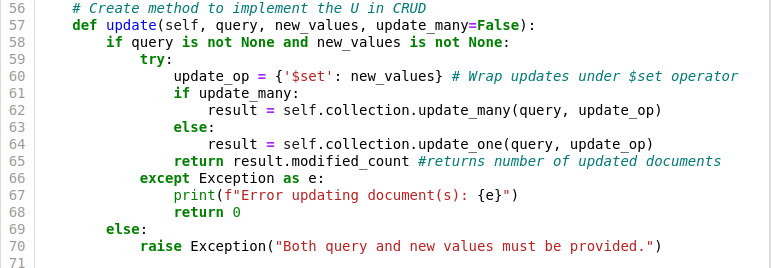
Now that the class has been created, we can create a method to implement the C in CRUD. In this project, we made it so upon a successful creation, the results would return True and unsuccessful creation would return False. We also implemented exceptions if the provided parameter is empty.



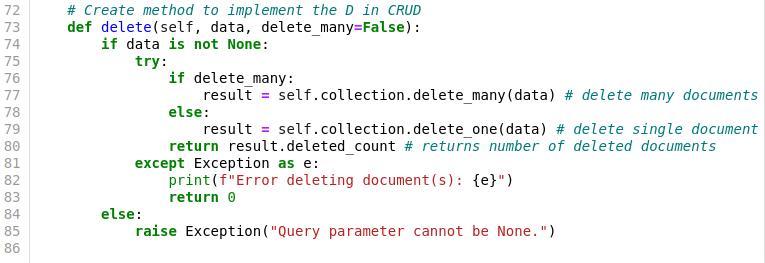
Next the R in CRUD was created so a successful read/query would produce the searched object and an unsuccessful search would provide an empty list. Exception cases were also implemented in the case that the query parameter was None:



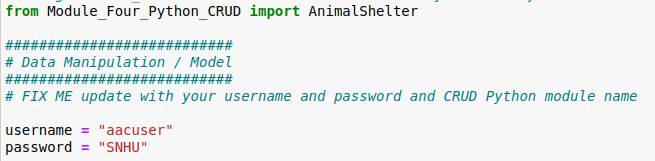
Next the U in CRUD was created so a successful update would query for document(s) with a matching key/value pair and update the document(s) accordingly. Following the update operation, the number of objects modified will be displayed. Exception cases were also implemented in the case that the query parameter was None:



Next the D in CRUD was created in order to delete a document based on a matching key/value pair. Following the deletion operation, the number of objects deleted would be displayed. Exception cases were also implemented in the case that the query parameter was None:



To begin creating the dashboard, I was provided a ProjectTwoDashboard.ipynb file that contained starter code for cre4ating the dashboard. My work to create this project started with importing my CRUD functionality and updating the username and password according to my CRUD module:



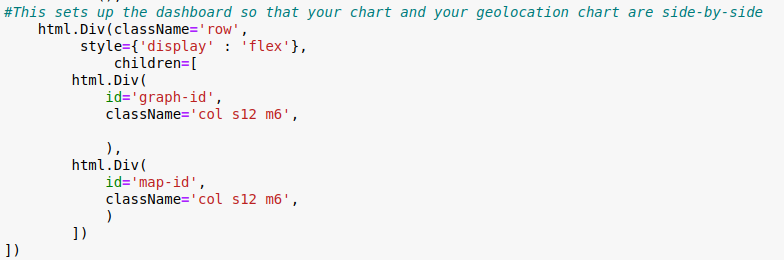
Next I imported the Grazioso Salvare Logo and constructed a header with the logo present in the upper left of the dashboard. Grazioso requested that the logo includes a URLanchor tag to their client’s home page of [www.snhu.edu](http://www.snhu.edu). The y also asked that I include a text stating my name as the author to give me credit for creating the dashboard.



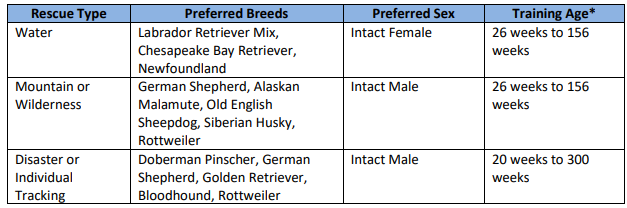
I next was asked to add filters based on rescue type in which I decided to integrate using radial buttons and I then created a user-friendly table for the data to be presented in:



I also formatted my dashboard to have a pie chart and a geolocation map side-by-side with each other below the table:

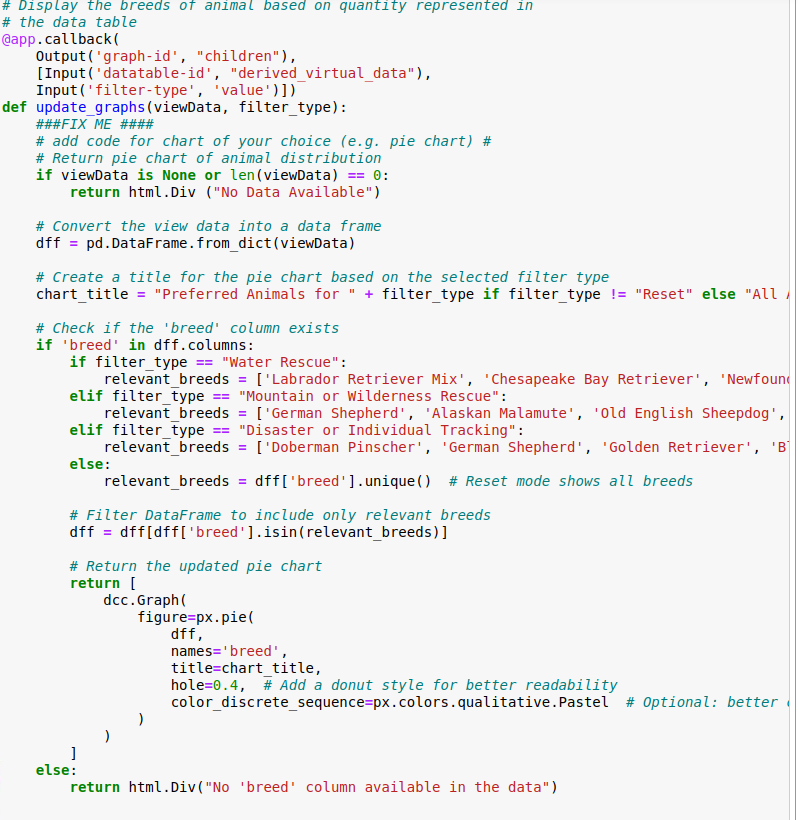


Next I defined my dashboard for what the various filters would actually include. Each filtered recue type would provide different results based off of Grazioso’s requested criteria for each rescue type:

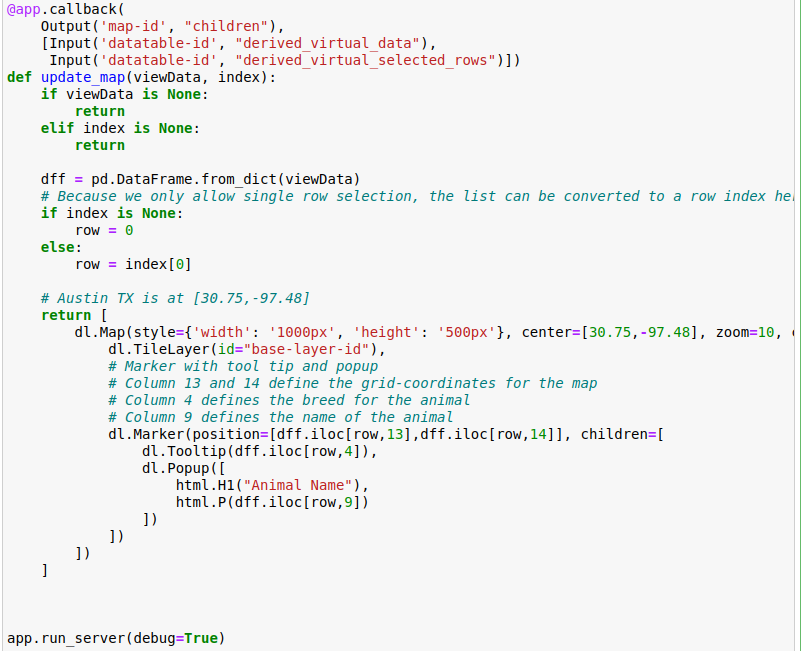




The updated filtered results would also update the respective pie chart to depict the results:

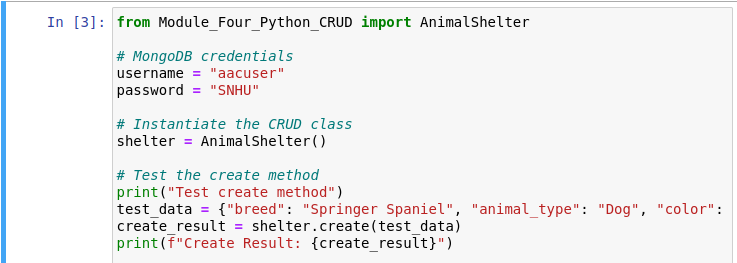


I also included the geolocation that was created in Module 6:



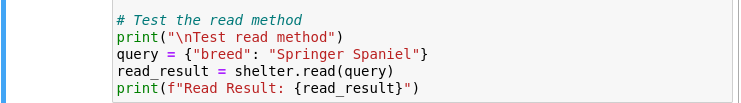
### Tests

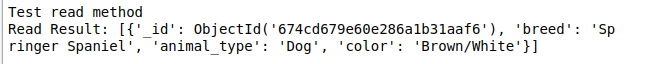
The newly created CRUD methods can now be tested with a Python test script (ipykernel). In order to do this, the newly created AnimalShelter class from the Python file is imported into the test script and two tests are created for the C and R methods. The CRUD class was instantiated with the variable shelter, a variable called test\_data was defined to consist of a breed, animal type and color. The results were then created and upon successful creation, we would see the result of True below. If the creation was unsuccessful, then we would see the result of False below.



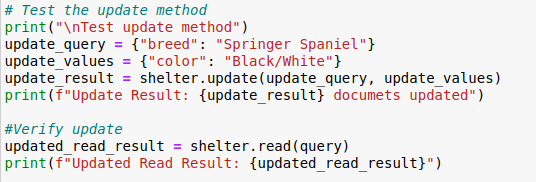


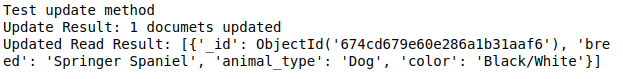
For the Read method (R in CRUD), a variable called query was constructed and then a variable called read\_result was created in order to search the database for the specified query.



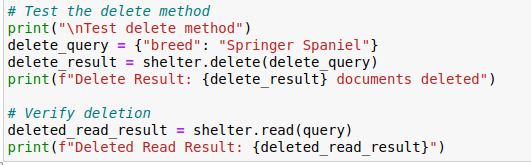


For the Update method (U in CRUD), a variable called update\_query was constructed, along with a variable called update\_values and update\_update\_result in order to search the database for the specified query, input the new desired values and to execute the update. Following the update, a statement regarding the number of document(s) updated is stated and a verify update test was implemented in order to check that the document for the update.





For the Delete method (D in CRUD), a variable called delete\_query was constructed, along with a delete\_result variable. The delete\_query variable was constructed in order to search the database for the specified query criteria and the delete\_result variable deletes the specified document. Following the deletion, an output statement regarding the number of document(s) deleted is displayed. A verify test was also implemented to show that the result was deleted. In the given picture, there are no instances of the query present thus it is displaying an empty list.

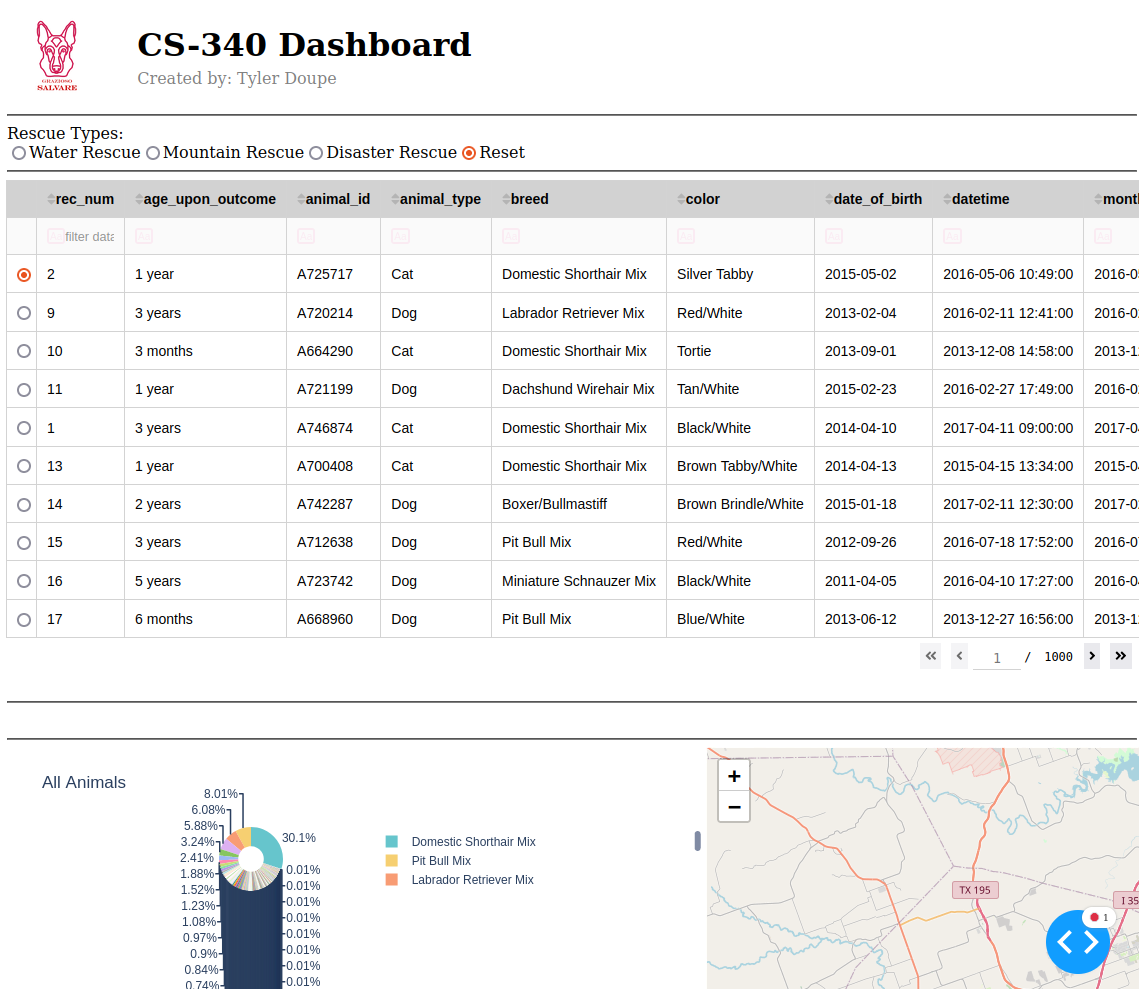




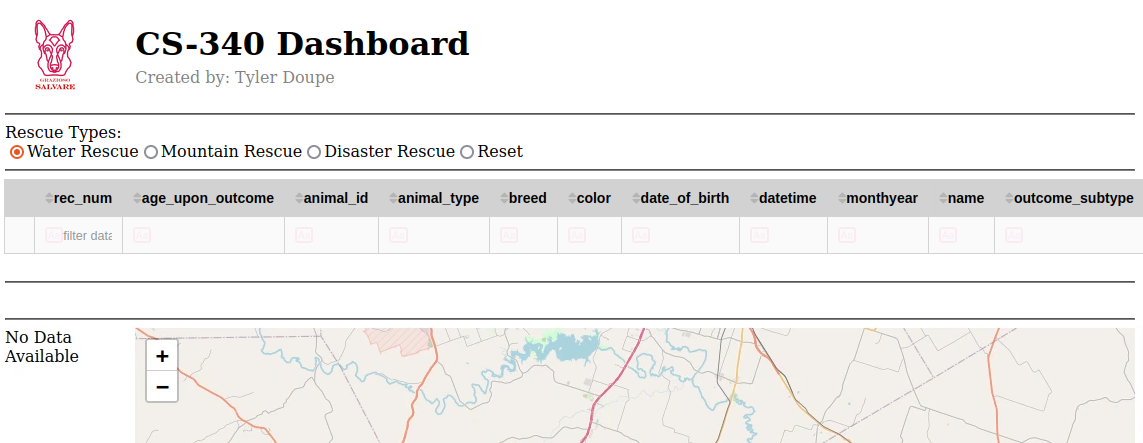
Now with the CRUD functionality integrated into the newly build Dashboard, we must test that the dashboard is working properly. The first test to ensure the dashboard is working correctly is to ensure that the code runs successfully:

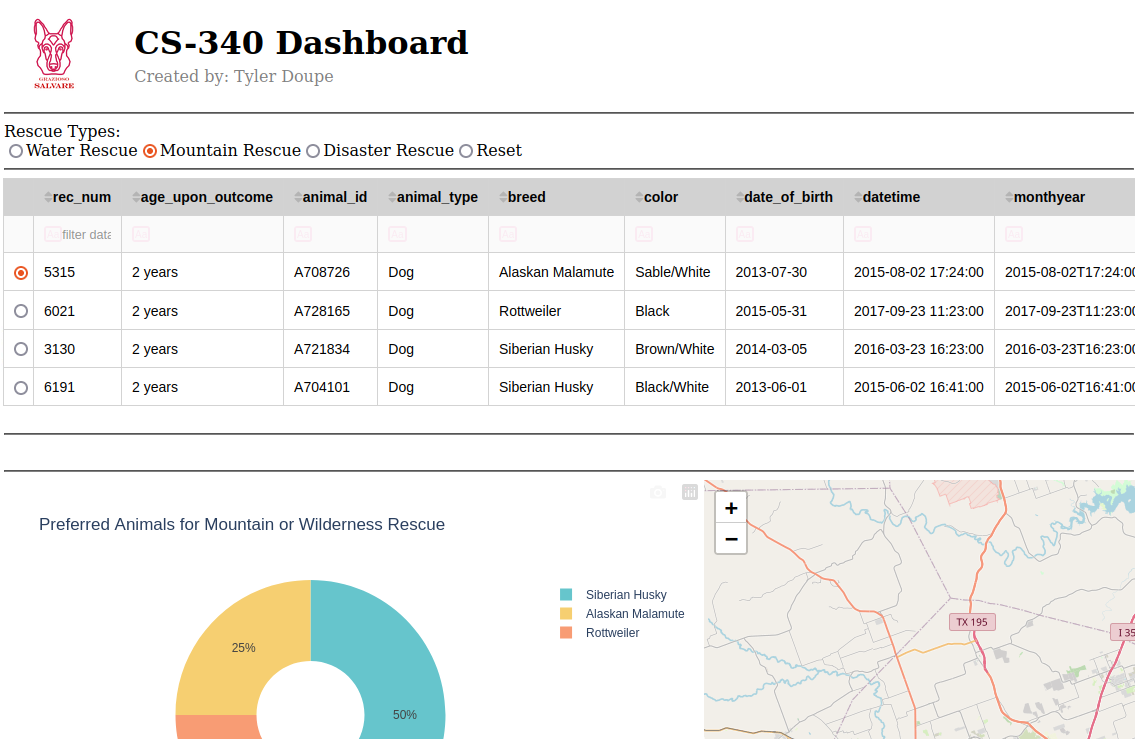
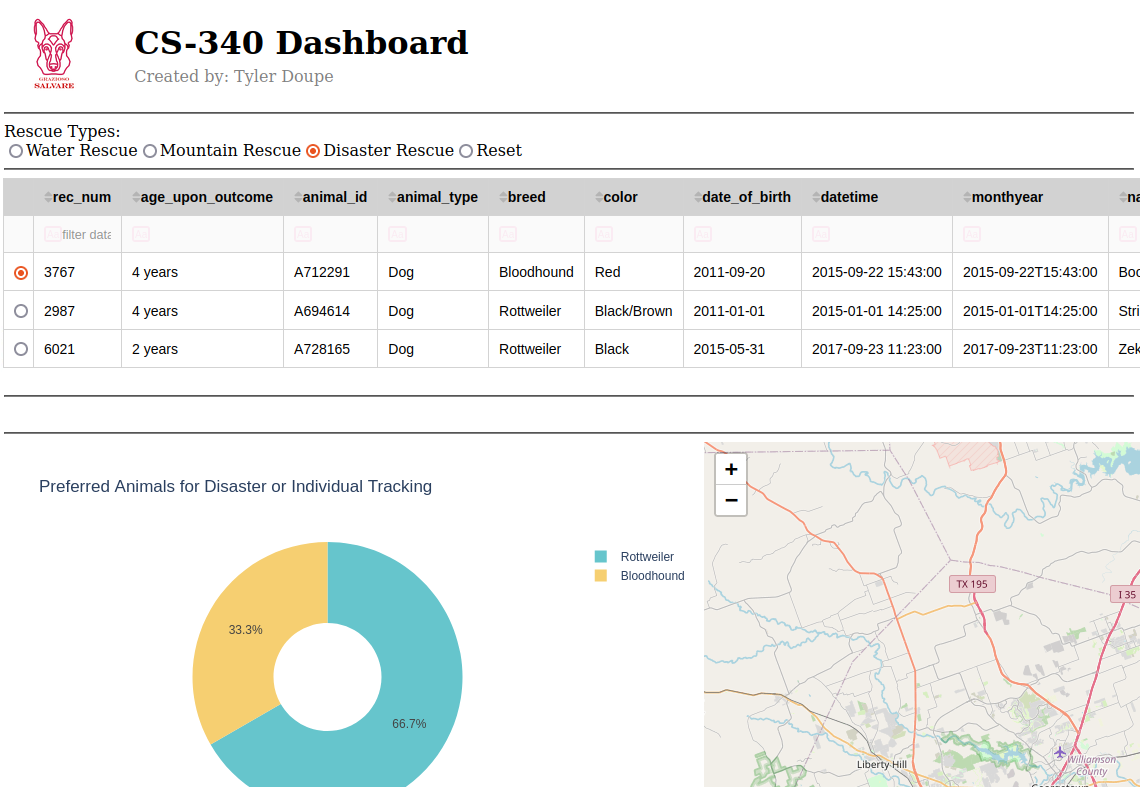


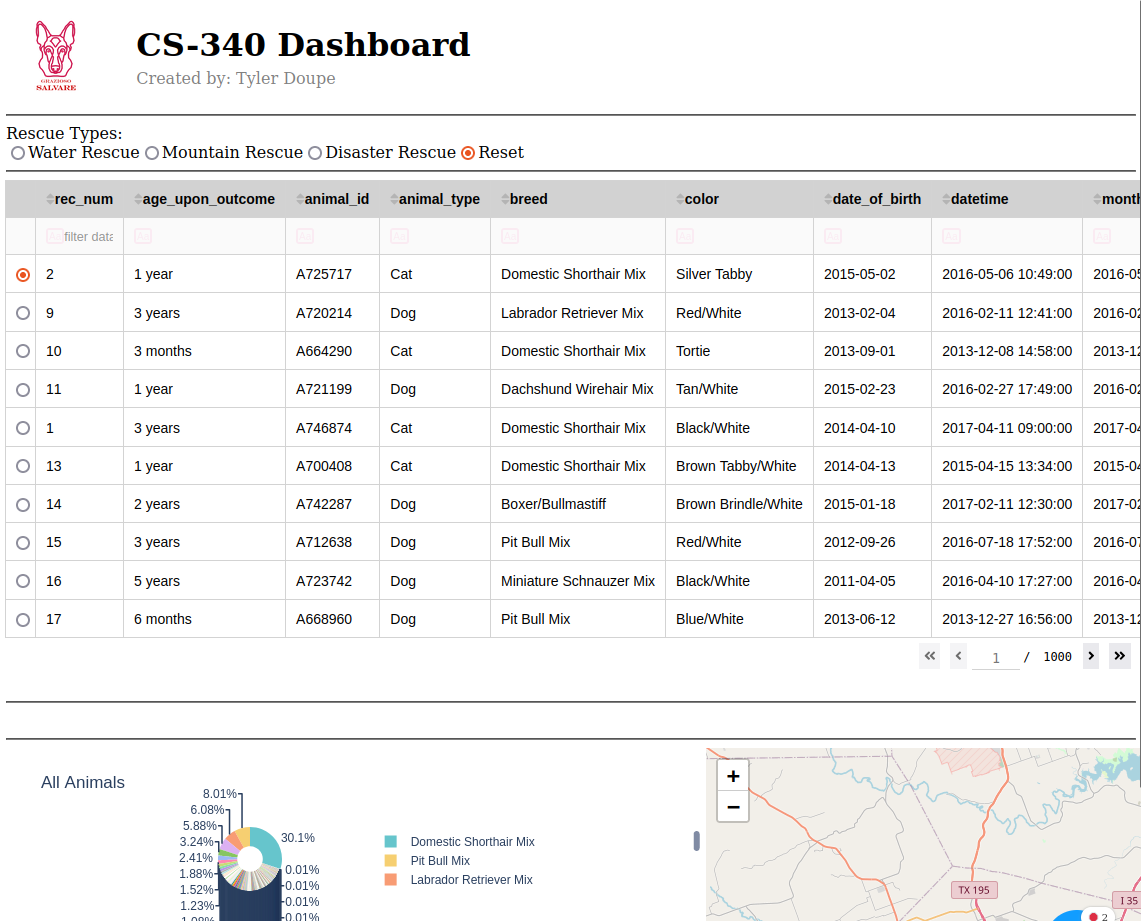
Followed by successful run of the code, we are going to the identified IP and Port number to view the dashboard where we are looking for successful load of the Grazioso logo, my unique indentifier, the various filters, the dynamic table, pie chart and geolocation widgets:



Next we must test that each filter provides different results and are consistent with Grazioso’s request for a filter applied. The following images are of each filter being applied:





**Resources**

MongoDB, Inc. (n.d.). Python with MongoDB. Retrieved December 14, 2024, from https://www.mongodb.com/resources/languages/python

Plotly Technologies Inc. Dash user guide. Retrieved December 14, 2024, from https://dash.plotly.com/

**Challenges Encountered**

As a growing programmer, I faced many challenges in building this project from start to finish. For starters, the work for this course was conducted on Linux which I had not been exposed to prior to this course. The course work also was performed through a VM and I faced some challenges early on figuring out how to appropriately upload and download my documents between the VM and my personal computer. These challenges were overcome but reviewing the course documentation provided, using YouTube videos and googling various user errors that I was performing. After figuring out the hurdles of performing the work for this course, my next challenges were a result of my limited knowledge on Python, MongoDB and building a dashboard with Python. In order to overcome the challenges, I faced as a new programmer, I had to frequently review the course documentation and readings, google any issues, errors, etc. in my code and hope to find resolutions online, and an enormous amount of trial and error. Overall, I found the challenges in the course difficult but extremely rewarding and helped build my knowledge of working with Python, databases and creating a front end user interface with the data.

## Contact

Your name: Tyler Doupe