# CS 340 README

## About the Project/Project Title

This project is about learning how to implement CRUD functionality using MongoDB and Python. This is a two-part project where you will learn how to implement create, read, update, and delete functions in Python so that it can communicate with a database in MongoDB. For the second portion of this project Grazioso Salvare is seeking a software application that can work with existing data from the animal shelters to identify and categorize available dogs. We will be completing the development of this project by coding the dashboard and the database interface logic. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and training time.

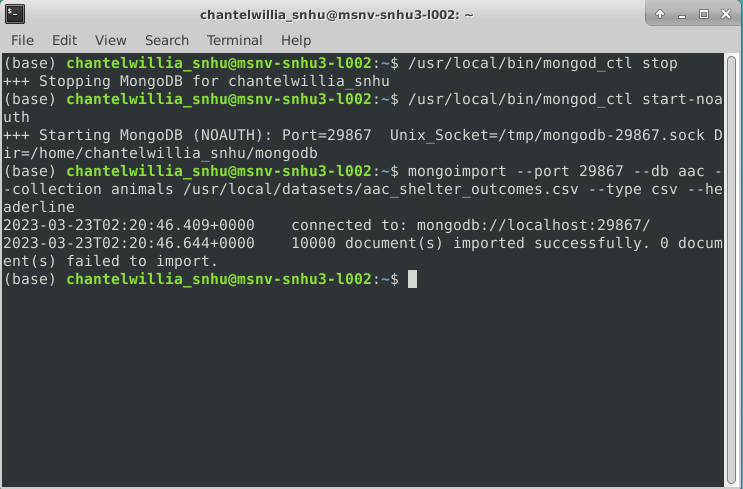
## Motivation

This project gives students an opportunity to apply concepts learned from Mastering MongoDB 4.X (Second Edition) by Alex Giamas. By doing this project students will be able to further understand the nuances of working with a database while earning practical experience. Additionally, this project will help students become more knowledgeable with integrating CRUD functionality with client facing web applications.

## Getting Started

Importing CSV file

You should first Sign into MongoDB and start your connection while making sure you have your port number as you will need it later. You will then import your AAC database. Once imported you will see a conformation message letting you know that your documents have been imported successfully.



Authentication

With your admin account we will create an **“aacuser”** and set the authorization to be able to use the database.

## 

## Once the account has been created you will be able to use the username and password that was just set up to login.

## Installation

When you are enrolled in this class you will be given access to the Virtual lab through Jupyter Notebook where these tools have been pre-installed. But for reference or if you would like to install these on your own these are the following tools that were used.

* Virtual lab access
  + This provides students with a virtual machine in case they don’t have the proper environment to complete this assignment at home.
* Jupyter Notebook
  + This provides students access to the server provided by SNHU so that you can connect your database with your Python script.
* Python Version 3.6
  + Python is used to send commands to MongoDB and can later be used when creating websites that need access to the database.
  + <https://www.python.org/downloads/release/python-360/>
* Command line Terminal
  + This is necessary to start MongoDB and import data.

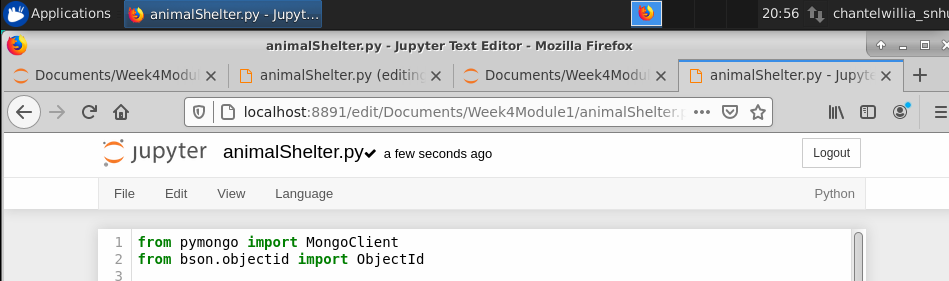
## Usage

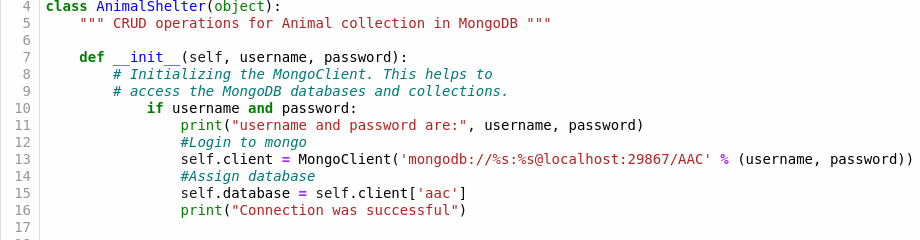
Below are all the steps used to write and test your python class.

**Part One setting up CRUD functionality.**

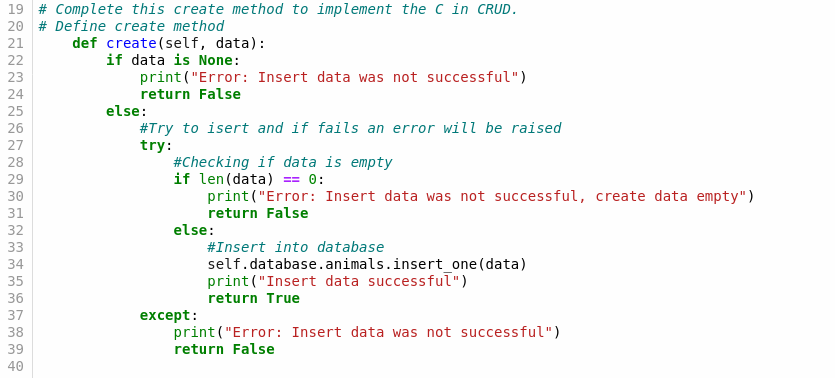
### Step by Step Code Example

Create your Python file in Jupyter Notebook and then import the needed libraries.

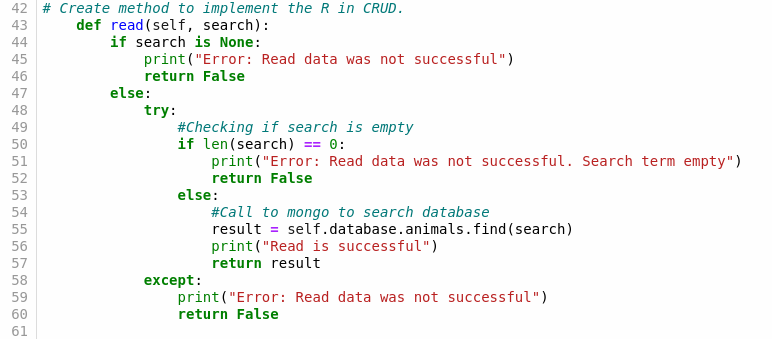
Next you will need to define your class and the initialize function and be sure to update the port number.



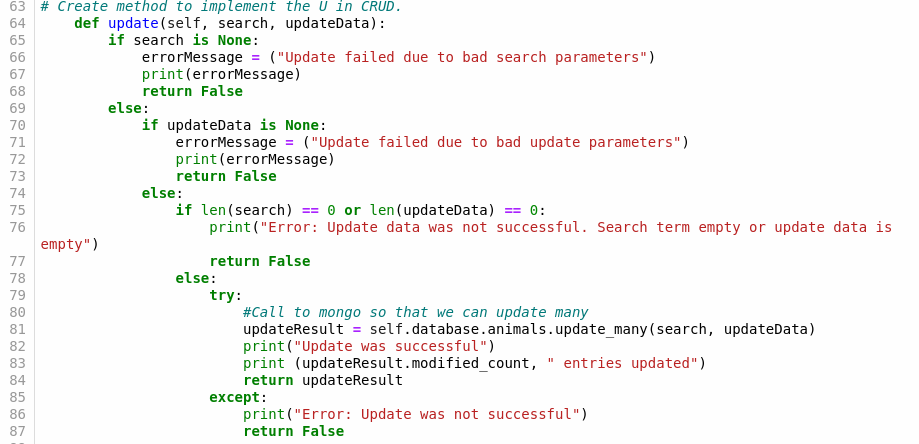
You will then need to define your **create** method. *As you develop your code, be sure to use industry standard best practices such as proper naming conventions, exception handling, and in-line comments. This will ensure that your code is easy to read and reusable for future projects.*



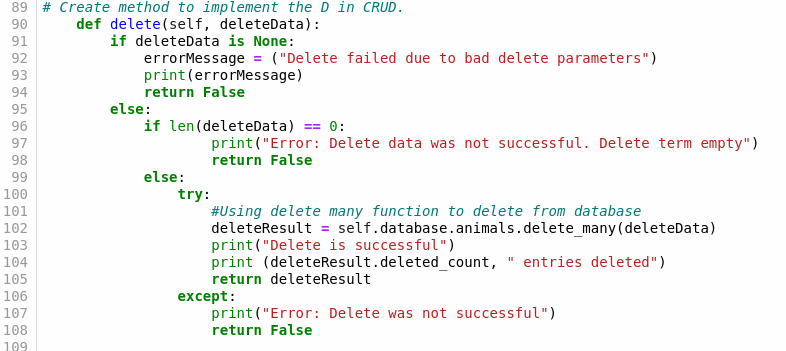
You then need to define your **read** method.



You then need to define your **update** method.

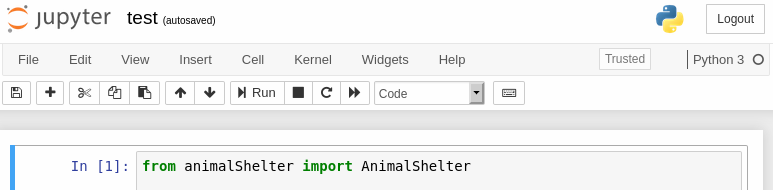


### And once completed you can then create your Delete method.



### Tests

Create your Test file in Jupyter Notebook and then import your AnimalShelter Python class.

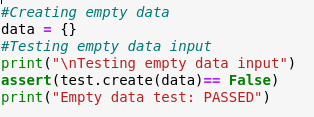
We will then test our authentication by passing your username and password.



When your run this test you should see the following result if the test was successful.



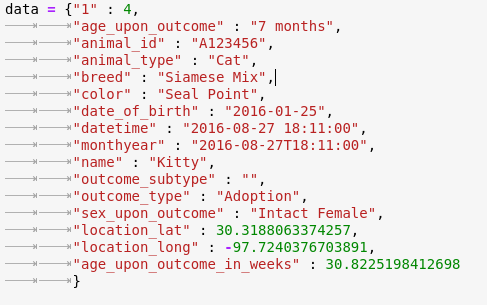
Then we will test our create function by first creating empty data.



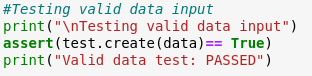
When you run this test, you should see the following result if the test was successful.



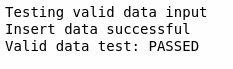
Then we will test our create function by first creating new data.



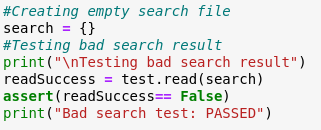
Then you’re going to pass this data into your create function and print the results.



When your run this test you should see the following result if the test was successful.



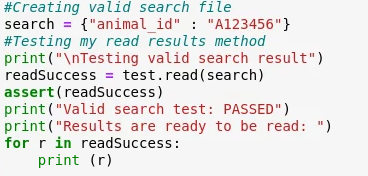
We will then create an invalid data parameter and perform error testing.



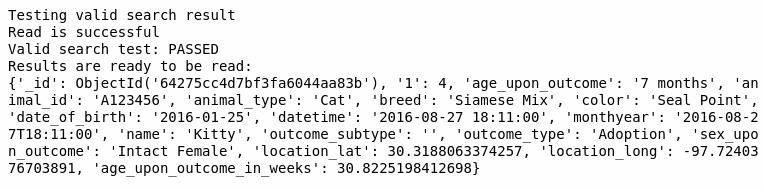
When the test passes, you will see these results.



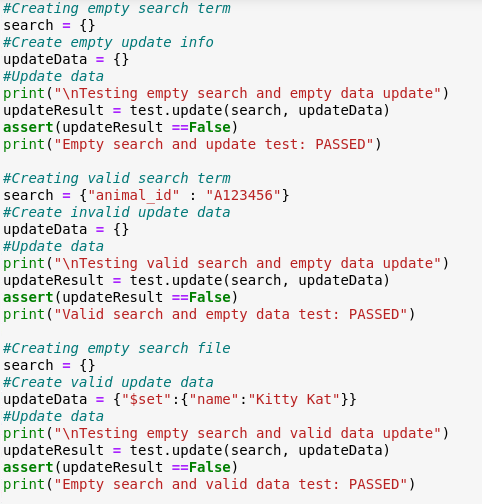
We will then create a valid search parameter and perform testing on the read function.



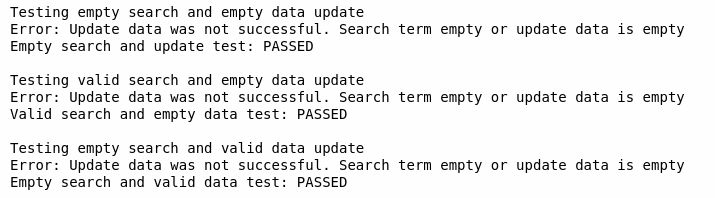
When you run this test, you should see the following result if the test was successful.



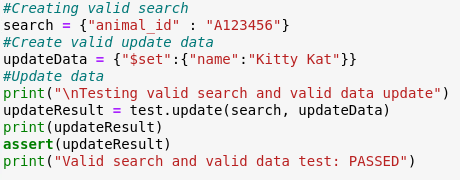
We will create multiple error testing for the update function (for each case where the search and update data are invalid)



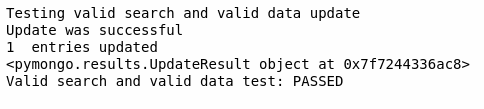
After running this test, you should see the following result.



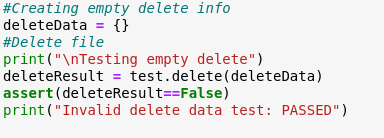
To test the **update** method, we will create valid search and update parameters. We will then pass these two parameters into the update method.



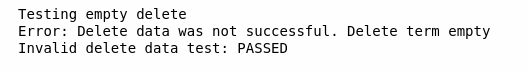
After running this test, you should see the following result.



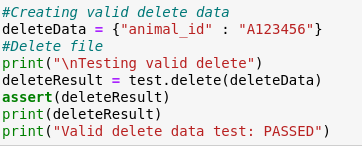
We will then incorporate error testing on the delete method by creating an empty delete data and passing it into the delete function.



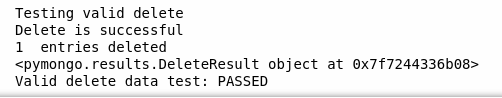
After running this test, you should see the following result.



Finally, we will test the delete method. We will need to crate a key/value lookup pair for the entry we want to delete. We will pass this as a parameter into our delete method.

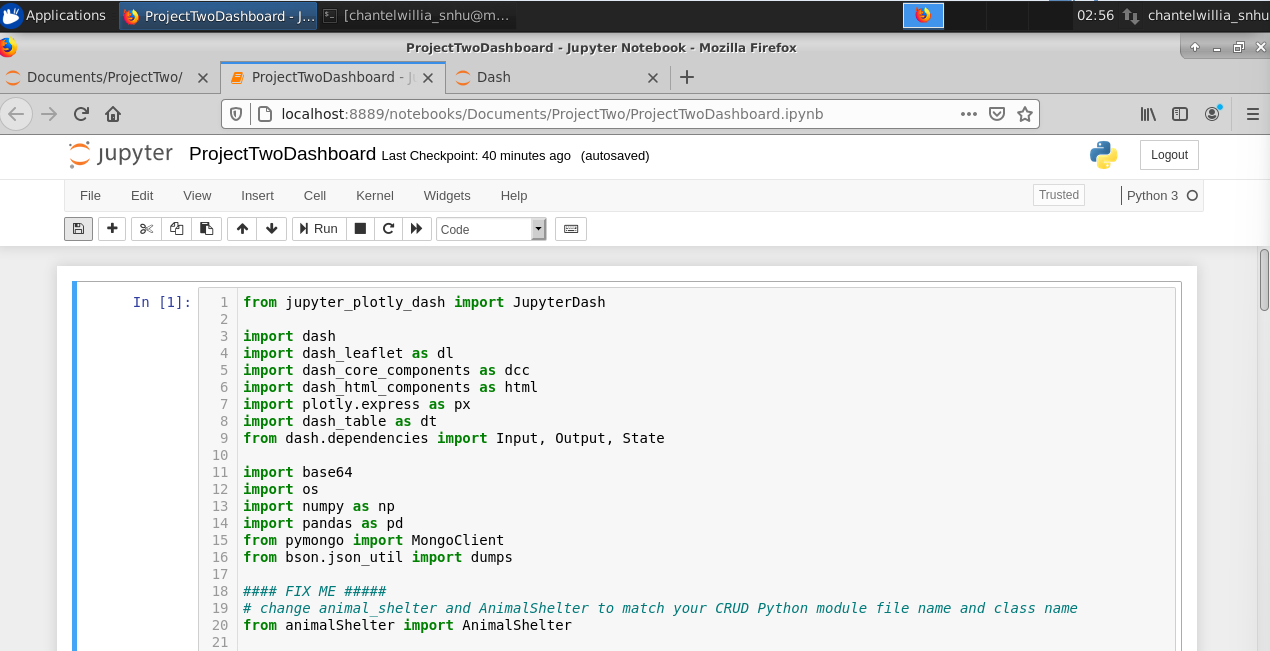


After running this test, you should see the following results.



**Part Two dashboard and database interface**

**Step by Step Code Example**

Create your Python file in Jupyter Notebook and then import the needed libraries.

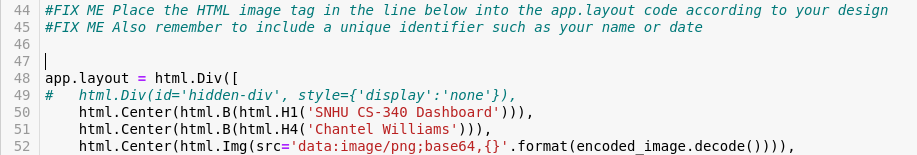
Next, we will import our AnimalShelter Python class and implement our authentication by passing our username and password.

Graphical user interface, text, application, email

Description automatically generated

Once this is completed, we will then need to check the status of MongoDB to see if it is already running and if not, we must connect to the server.



We will then follow Grazioso Salvare’s branding requirements by incorporating the company logo and our unique identifier. 

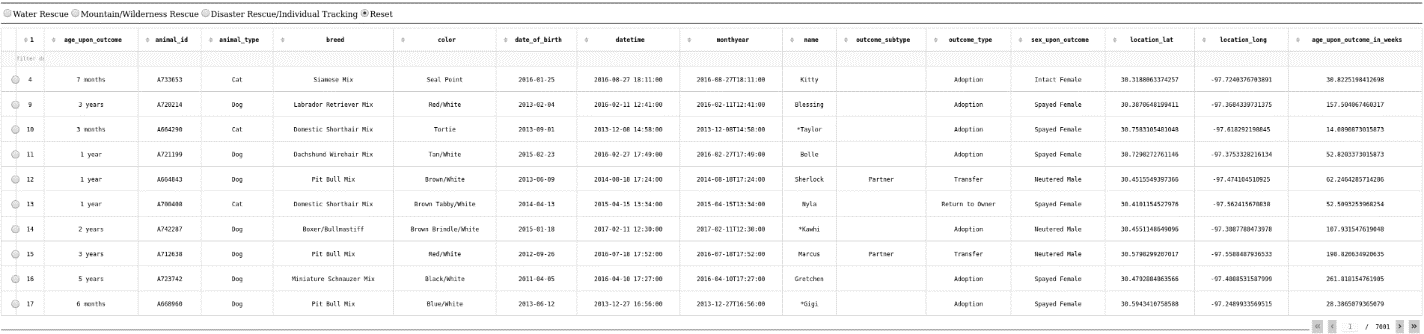
Once this code is implemented you should see the following results.



We then created our data table on the dashboard which shows an unfiltered view of the Austin Animal Center Outcomes data set. We added other settings such as padding and centering to improve the readability to the user.



Once this code is implemented you should see the following results.



Next, we will continue to follow Grazioso Salvare’s branding requirements by incorporating the required dashboard requirements. We have decided to incorporate radio items as our interactive filtering option. The first portion of this code will allow us to see our filtering options at the top of our data table.

Once this code is implemented you should see the following results.

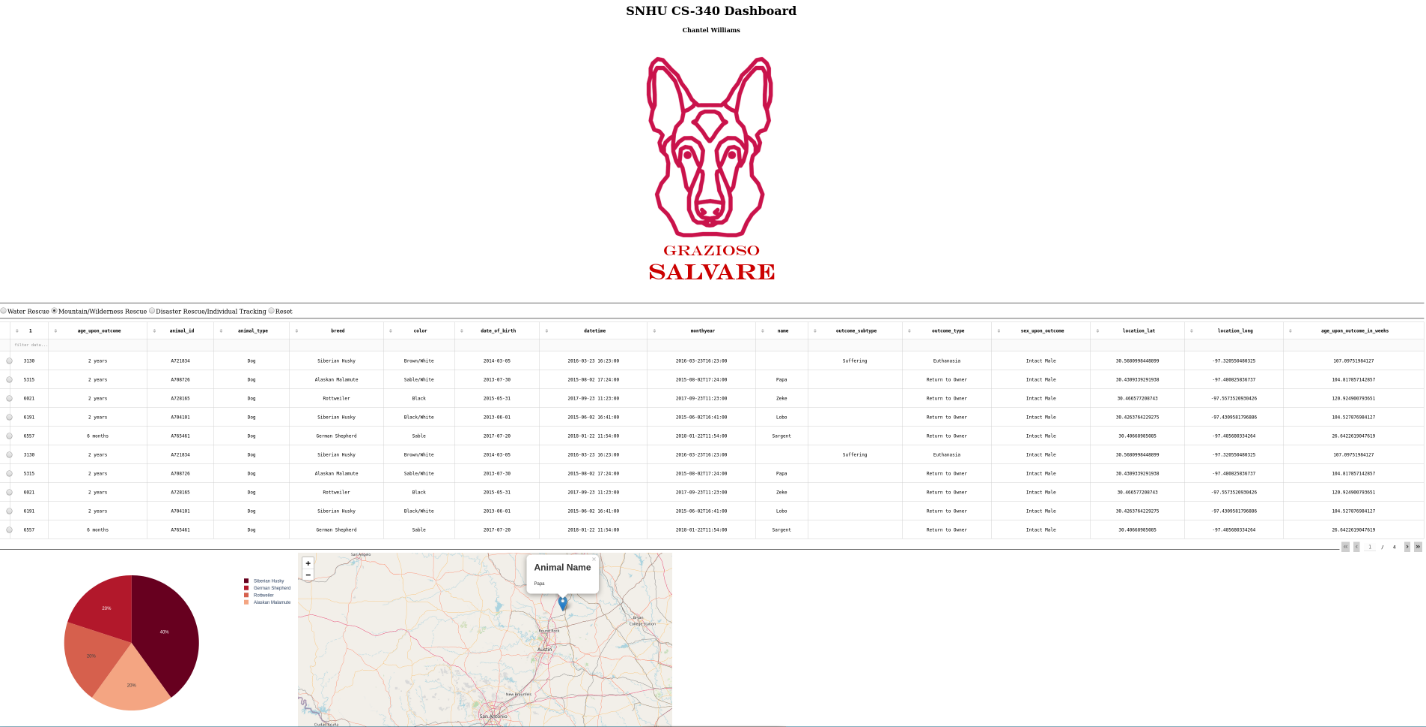


Graphical user interface, text

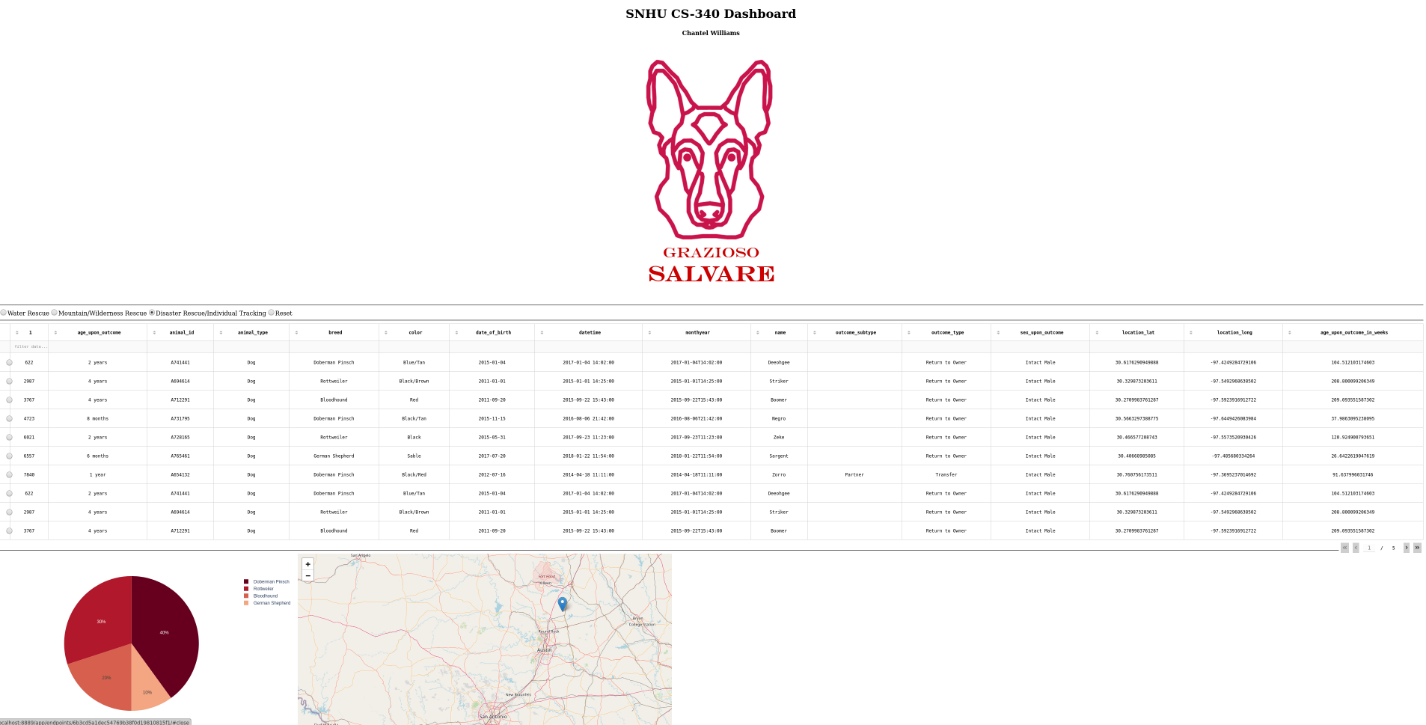
Description automatically generated with medium confidenceText, timeline

Description automatically generated with medium confidenceWe will now write queries for the interactive option functionality based on what Grazioso Salvare has provided us with. This is an example of how the CRUD functionality can be used, specifically the **Read** function where we submit specific queries to filter the specific information needed.

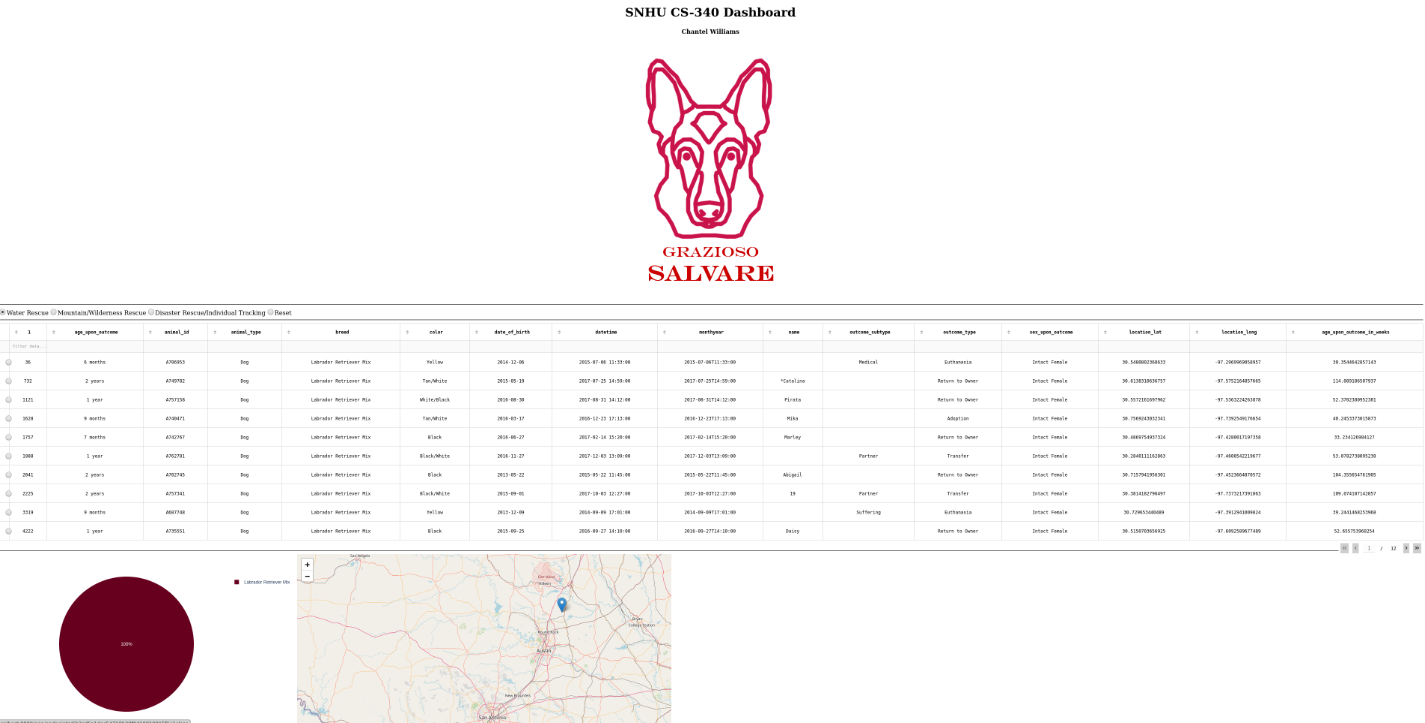
Once this code is implemented you should see the following results. For the below screenshot I will be showing how the **Mountain/Wilderness Rescue** interactive button can filter the information.



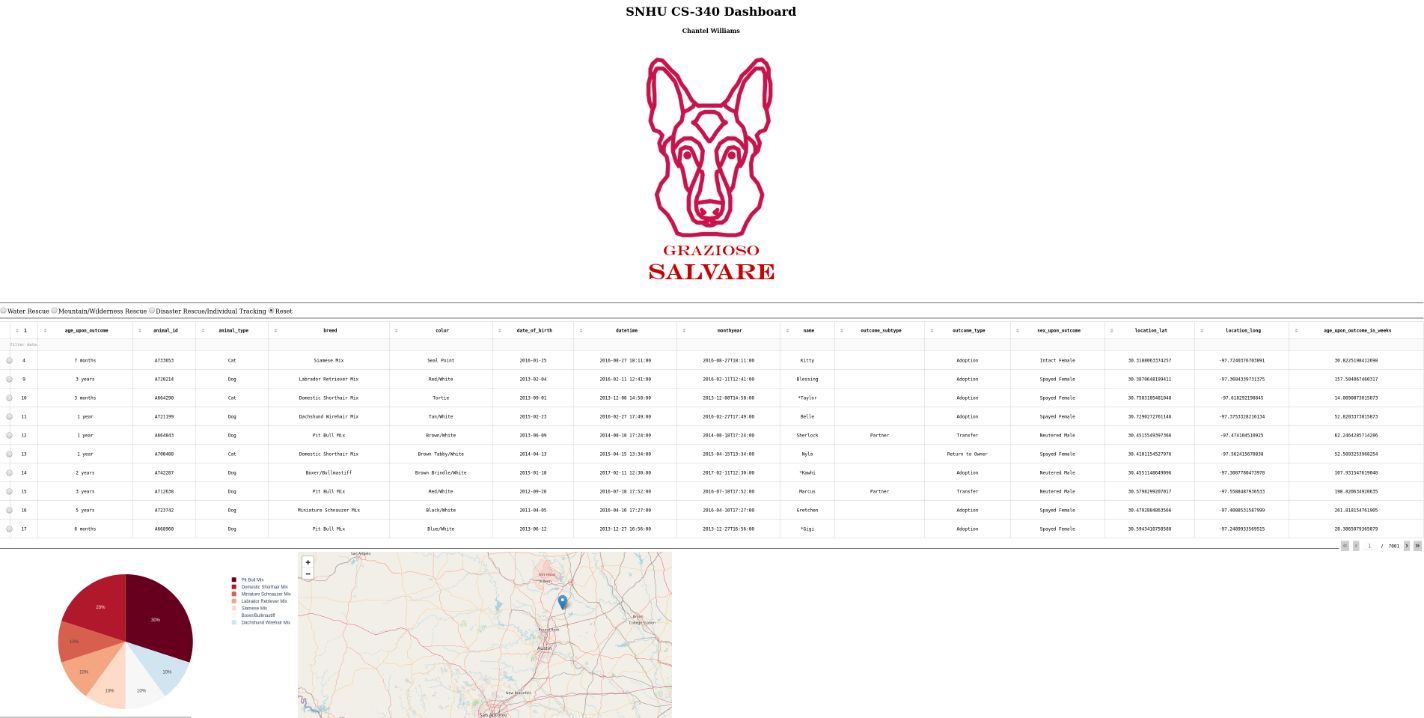
## Here you can see how the Disaster Rescue/Individual Tracking interactive button can filter the information.



Next you can see how the **Water Rescue** interactive button can filter the information.



And finally, you can see how the **Reset** button returns all widgets to their original unfiltered state.



## We will then set up the dashboard so that our data table and geolocation chart are side-by-side.

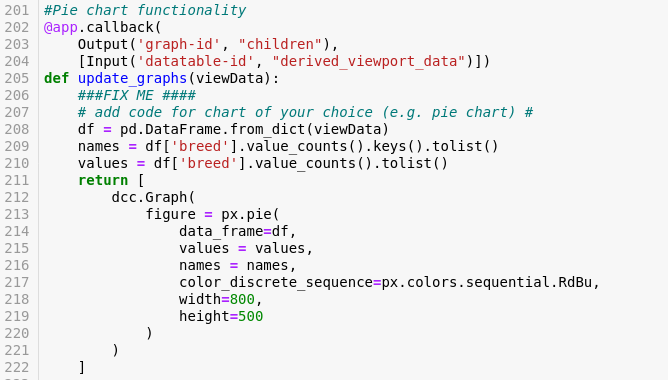
****

## Next, we will continue to follow Grazioso Salvare’s branding requirements by incorporating a geolocation chart.

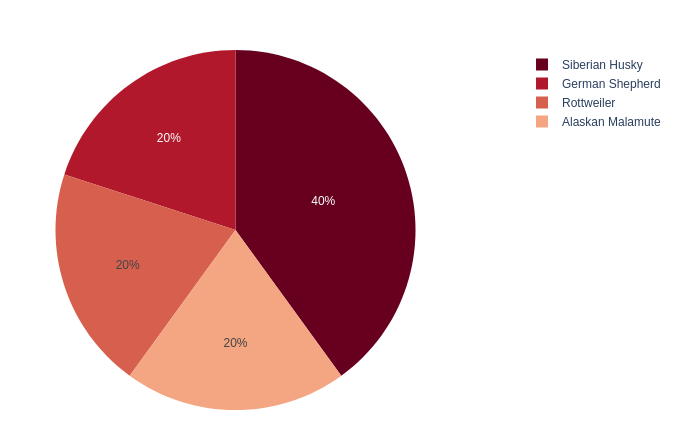
## Once this code is implemented you should see the following results.

## 

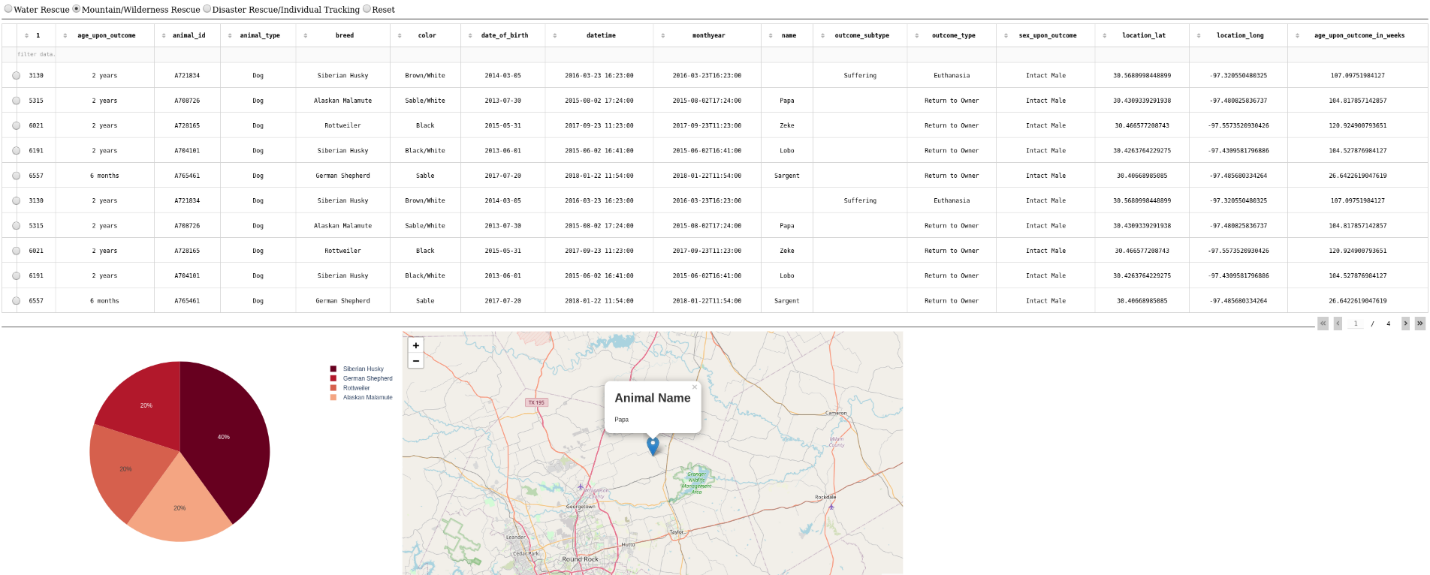
We were also asked to implement a second chart of your choice that dynamically responds to the filtering options. For the pie chart we need two sets of data, the first being the names of the breeds in that location and then the second being how many of each breed were in that location. To achieve this, we used a python package called Pandas which makes importing and analyzing data much easier. **df[‘breed’].value\_counts()** function returns the object containing counts of unique breeds.



Once this code is implemented you will see a pie chart that shows the breeds in each area with their corresponding percentages.



Once this code is implemented you should see the following results. For my below screenshot I will be selecting the **Mountain/Wilderness Rescue** interactive button, so you are able to see how the data dynamically responds to the filtering options.



**Grazioso Salvare has requested documentation to accompany the code for your dashboard.**

* **Describe the required** functionality of the project. Included above are the screenshots taken while testing and deploying the dashboard as proof that the required functionality has been achieved.
  + Grazioso Salvare requested that we create a UI/UX for our interactive database specifically to identify dogs that are good candidates for search-and-rescue training. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and

training time.

* + Grazioso Salvare is requiring the following widgets for the dashboard interface:

• Interactive filter options (buttons, drop-downs) to filter the Austin Animal Center Outcomes data set by:

○ Water Rescue

○ Mountain or Wilderness Rescue

○ Disaster Rescue or Individual Tracking

○ Reset (returns all widgets to their original, unfiltered state)

• A data table which dynamically responds to the filtering options

• A geolocation chart and a second chart of your choice (such as a pie chart) that dynamically respond to the filtering options.

**Describe the tools used to achieve this functionality and a rationale for why these tools were used.**

* Be sure to explain the Dash framework that provides the view and controller structure for the web application.
  + Controller – we developed the AnimalShelter class to have CRUD functionality using python that allows us to interact with the database (MongoDB). We then developed queries to filter out the information we needed from the database which we then used in the **@app.callback().**
  + View – We built interactive options through which users will activate through radio filters and then added a geo location and pie chart widgets for dynamic presentations of the retrieved data.

**Identify any challenges that were encountered and explain how those challenges were overcome.**

* Some challenges I faced will working on this project were becoming familiar with MongoDB (such as dealing with case sensitivity). During the second part of this project
* There were also multiple interruptions with virtual lab connectivity which was out of my control and caused me to have to stop and restart the virtual lab. Another challenge I encountered was learning error testing in Python because I have only learned how to unit test in Java.
* I also had to do a lot of research to use libraries such as Pandas to be able to use the dynamic widgets as well as conducting research on how to use Plotly to create the pie charts specifically for Dash.

**Previously requested information:**

* **An explanation of the purpose of the CRUD Python module**
  + The CRUD Python module is a component that provides a set of methods for performing basic database operations. The purpose of this module is to simplify the process of working with databases by simplifying the low-level language of MongoDB. By simplifying this it will save time and reduce errors while working with MongoDB. Lastly, it will make the code more readable and maintainable.
* **An explanation of how the module should be used, including:**
  + **A description of the Python driver for Mongo that was used and why it was chosen.**
    - Python is a mature and secure driver for web development because it has been around since the early 90’s. Python has many usable libraries and is used heavily within the industry.
  + **An explanation of the attributes and working functionality of the CRUD operations**
    - Create: This operation allows users to insert new records into a database
      * This create function takes the data to be inserted into the database as a parameter and checks to make sure that it isn’t empty or null.
      * This uses MongoDB insert\_one() functionality.
    - Read: This operation allows users to retrieve records from a database
      * This read function takes a key value pair as its search parameters and checks to make sure it is not empty.
      * It uses MongoDB find() method.
    - Update: This operation allows users to modify existing records in a database.
      * This update method takes two parameters. The first is the key/value pair for the search parameters and the second is another key/value pair for the data to be updated. The function checks to ensure that neither are empty or null.
      * This function uses the updateMany() function of MongoDB.
    - Delete: This operation allows users to remove records from a database.
      * This delete function takes in the key/value pair for the delete parameters and checks to ensure they are not empty or null.
      * This uses the deleteMany() functionalities for MongoDB.

## Contact

Chantel Williams