# CS 340 README Project Two

## About the Project

Grazioso Salvare, an innovative international rescue-animal training company, has partnered with several animal shelters to identify dogs that are good candidates for search-and-rescue training. Grazioso Salvare is seeking a software application, one that will include a database and a client-facing web application dashboard, that can work with existing data from the animal shelters to identify and categorize available dogs. The user will be able to filter the data, through the use of radio buttons, to specify which category of dog they’d like to view, Water Rescue, Mountain or Wilderness Rescue or Disaster or Individual Tracking. When filtering by type or rescue animal, a graph chart will display, showing the percentages of each breed for each category or rescue type. Additionally, by selecting a single row within the data table, a geo-location map containing the latitude/longitude data for that particular animal will display.

As the lead developer on the project, we are to establish a connection from a client-side user interface to a database, through the use of a portable Python module that enables CRUD functionality for the data connection. The CRUD functionality will allow specific users to create, read, update and delete data.

## Motivation

The purpose of the project is to develop a web application that connects a client-side user interface to a database, in the form of a dashboard, available to users at Grazioso Salvare. The project will help streamline the process of identifying and categorizing ideal candidate dogs found in the animal shelters that would make for excellent search-and-rescue training.

## Getting Started

Upload the Austin Animal Center (AAC) Outcomes data set into MongoDB by **importing a CSV file using the appropriate MongoDB import tool**.

**Develop** a python CRUD class within the IDE of your choice, that when instantiated, provides the following functionality:

* 1. **A *Create* method that inserts a document into a specified MongoDB database and collection**
     1. Input argument to function will be a set of key/value pairs in the data type acceptable to the MongoDB driver insert API call
     2. Return “True” if successful insert, else “False”
  2. **A *Read* method that queries for documents from a specified MongoDB database and collection**
     1. Input arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call
     2. Return result in a list if the command is successful, else an empty list.
  3. **An *Update* method that queries for and changes document(s) from a specified MongoDB database and specified collection**
     1. Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver **Find API** call. The last argument to function will be a set of key/value pairs in the data type acceptable to the MongoDB driver update\_one() or update\_many() API call.
     2. Return -> The number of objects modified in the collection.
  4. **A *Delete* method that queries for and removes document(s) from a specified MongoDB database and specified collection**
     1. Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.
     2. Return -> The number of objects removed from the collection.

Finally, create a Python testing script in Jupyter Notebooks that imports your CRUD Python module to call and test the CRUD functionality. Additionally, the testing script will contain the necessary infrastructure for a radio-button filtering function, for each rescue type, a single-select data table, which displays the geo-location of the selected animal, as well as a pie chart graph that showcases the percentages by breed for each type of rescue animal chosen by way of the radio buttons.

## Installation

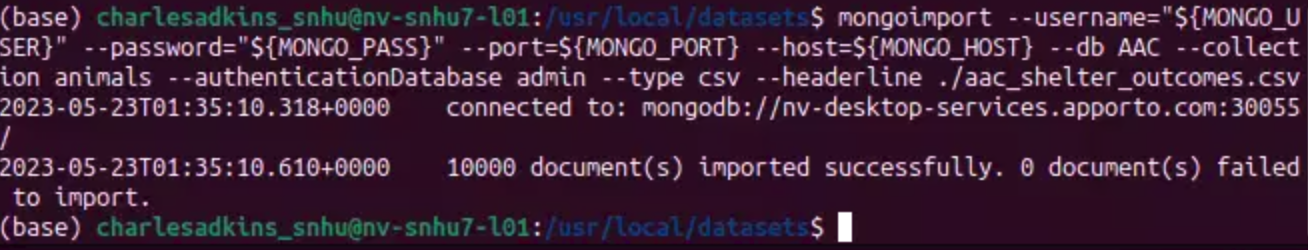
The terminal will be the environment by which you implement the MongoDB scripts. “MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas” (Gillis, 2023). We first create a database in MongoDB by importing the data, Austin Animal Center (AAC) Outcomes data, through the use of the MongoDB import tool (**Figure 1**). We then create both a simple (**Figure 2)** and a compound index to optimize queries (**Figure 3**). Then we create a new user, with authentication, which users at Grazioso Salvare will access the database through (**Figure 4**). We can verify the newly created user and the connection credentials through updating our Mongo environment variables (**Figure 5**).

Python code will be the means by which we develop our CRUD functionality for the database. “Python is a dynamically typed programming language. Python can be used on a server to create web applications and supports multiple programming paradigms, including structured, object-oriented and functional programming” (W3schools, n.d).

Then, utilizing an IDE of your choice, develop the python code that establishes the database connection (**Figure 6**) and utilizes the CRUD methods that have the capability of creating (**Figure 7**), reading (**Figure 8**), updating (**Figure 9**) and deleting (**Figure 10**) documents from the specified collection. Then we create a testing file to validate our database connection and CRUD methods, which can be developed through a Jupyter notebook. The Jupyter Notebook is the original web application for creating and sharing computational documents, which supports over 40 programming languages including Python, R, Julia and Scala (Jupyter, n.d.). The file created in Jupyter Notebook will import our CRUD Python module, import necessary libraries, pass user credentials and call the read function to establish our dataset (**Figure 11**). We then filter the dataset based upon the criteria outlined by the client, which details the characteristics for each rescue type, including, breed, sex, and age (**Figure 12**). Then we set the layout and format of our dashboard, to include title, unique identifier, logo and radio-buttons which enables the filtering functionality (**Figure 13**). We then setup the data table, making it single selection and establishing space for the graph and map (**Figure 14**). Then we provided functionality for updating the dashboard based upon the rescue type radio-button selection made (**Figure 15**). We added some interactive elements to the data table, which changes the color of the table cell and selection button to red when chosen (**Figure 16**). We added functionality to the graph, pie chart, as well. When the user selects a rescue type dog, the pie chart displays the percentage and count of each breed per rescue type (**Figure 17**). Lastly, we added the functionality for the geo-location map, which updates based upon the selection made within the data table, ultimately displaying the location of that specific dog (**Figure 18**). The following screenshots display the results of choosing each rescue type, ‘Reset’, ‘Water Rescue’, ‘Mountain or Wilderness Rescue’, ‘Disaster or Individual Tracking’, (**Figure 19-22**), respectively.

Challenges for this particular work include understanding and implementing several different functions within several different areas of requirements, specifically, MongoDB, python, Jupyter Notebooks. Understanding the syntax, formatting, functionality and how it relates to the needs of the client, can take considerable time if the developer is not familiar with the platforms. Consulting official platform and language documentation is always the best way to navigate such challenges.

## Usage



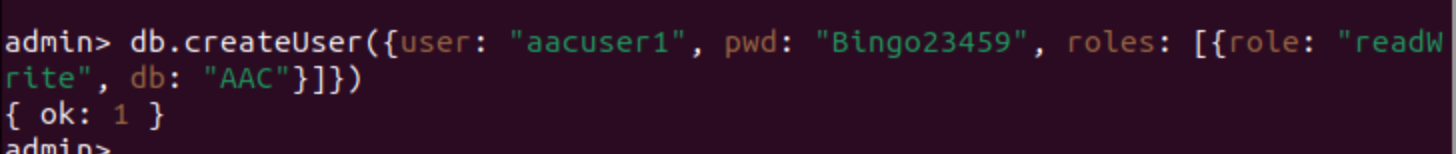
**Figure 1: Importing the AAC database with confirmation of 1000 documents imported.**



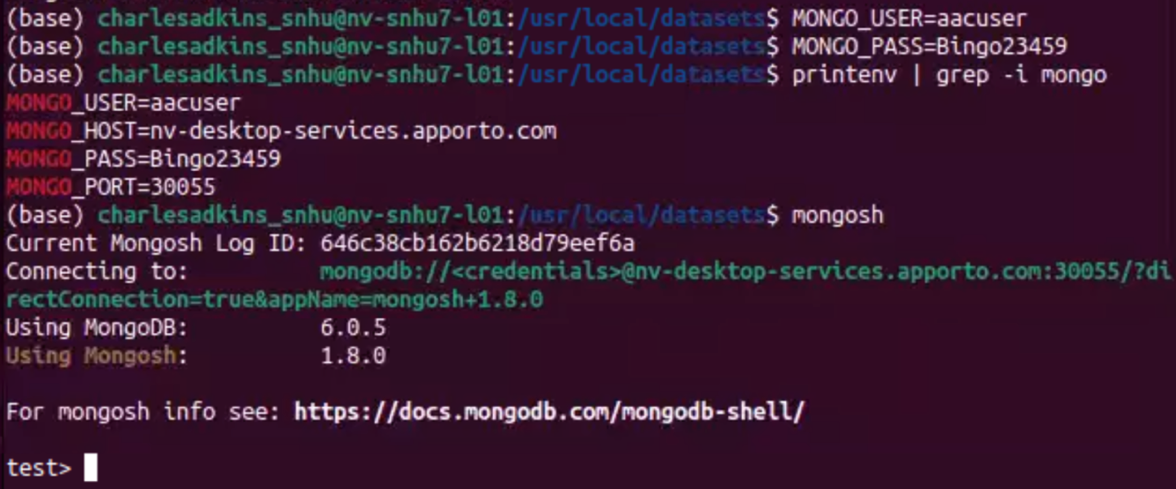
**Figure 2: Simple Index - Creating index for the key “breed”.**

**

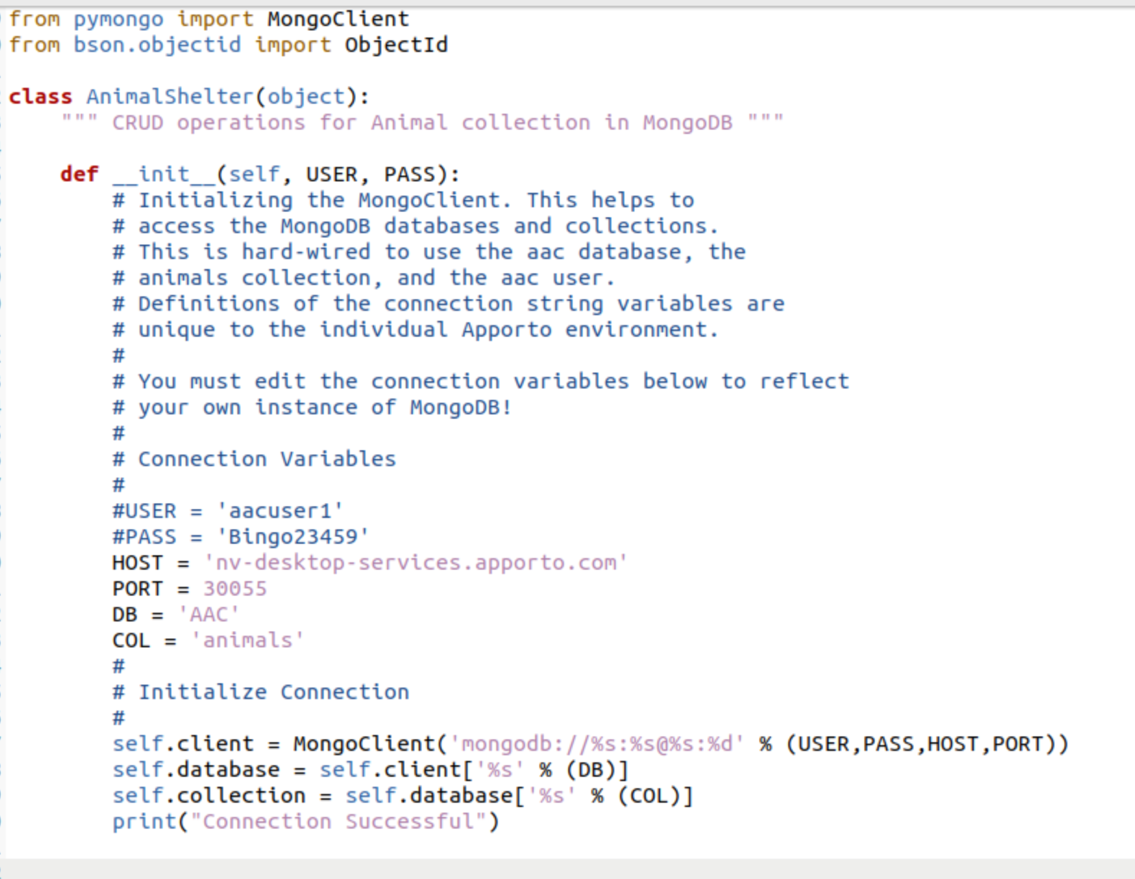
**Figure 3: Compound Index - Finding results for specific breed and outcome\_type, while also utilizing the explain() function.**

****

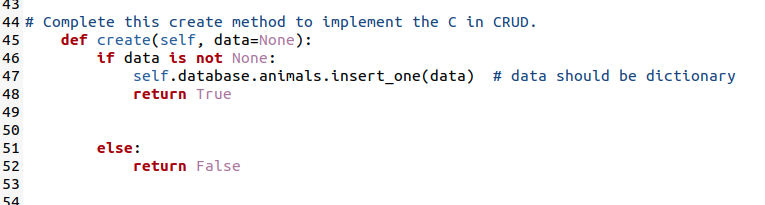
**Figure 4: Creating a new user called “aacuser1” with “readWrite” permission on the “AAC” database.**



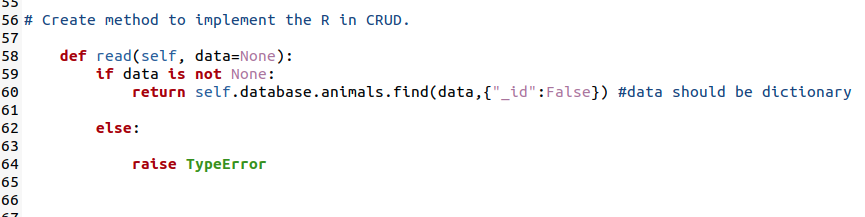
**Figure 5: Updating environment variable values to verify the new users’ credentials.**

**

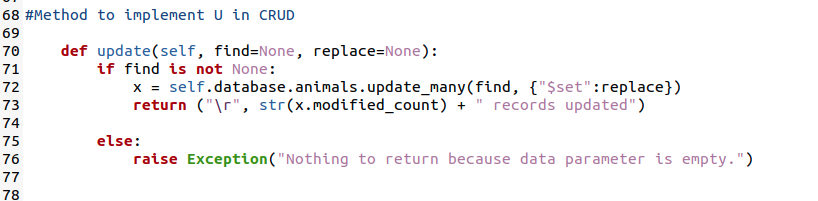
**Figure 6: The Mongoclient ties python to MongoDB and the ‘init’ function provides the credentials and the connection details for the AAC database. The username and password parameters can either be passed into the init function, or established in the “connection variables” section, of which are commented out below, to show the various possibilities. The ‘Host’, ‘Port’, ‘DB’ and ‘Col’ variables are their respective values will depend on the information provided back to you when you verify the new users’ credentials, shown in Figure 5 above.**

**

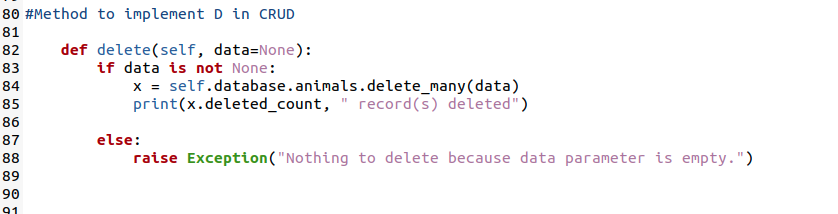
**Figure 7: The create method takes in the parameter self and data and inserts it into the animals collection.**

****

**Figure 8: The read method utilizes the find() function to locate and return all of the matched objects, in a list form, that which is the data parameter.**

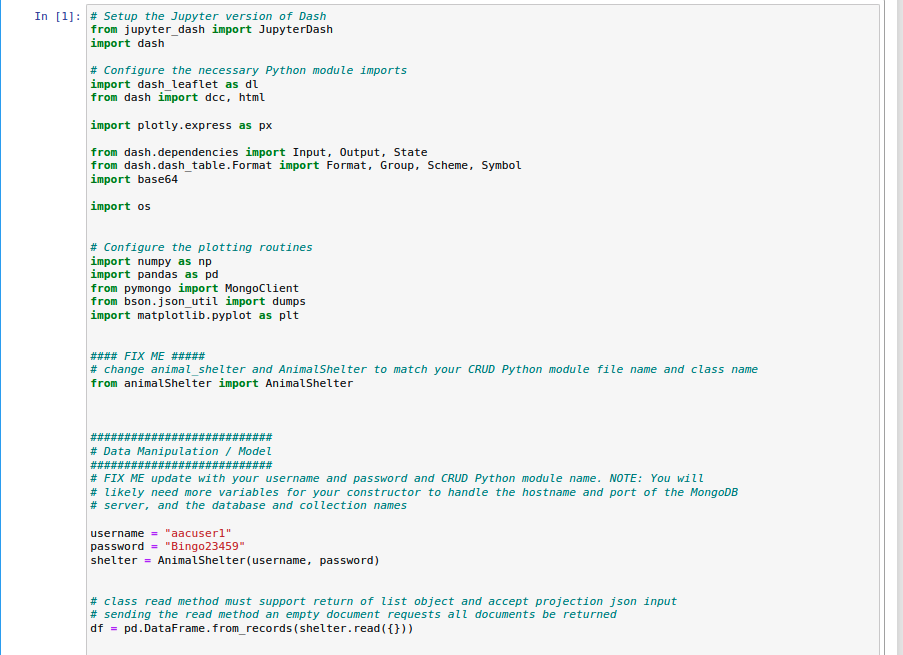


**Figure 9: The update method takes in two parameters, one to find the object and one to replace it, which utilizes the find() function and returns a statement that displays the count of records updated.**



**Figure 10: The delete method takes in a parameter and deletes the number of occurrences found and returns a statement that displays the count of records deleted.**

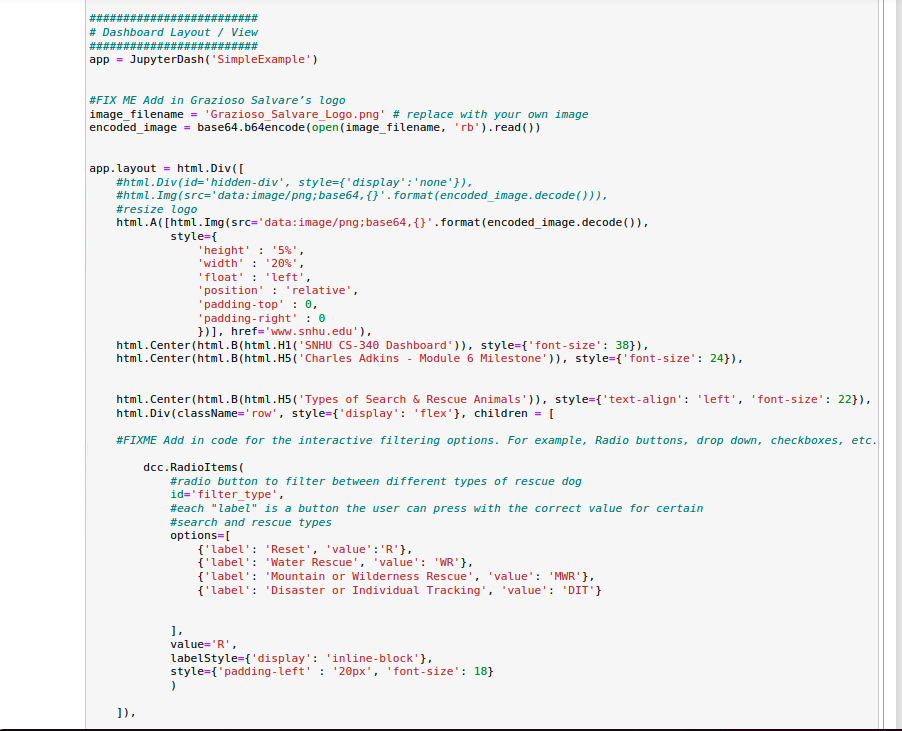
### Tests



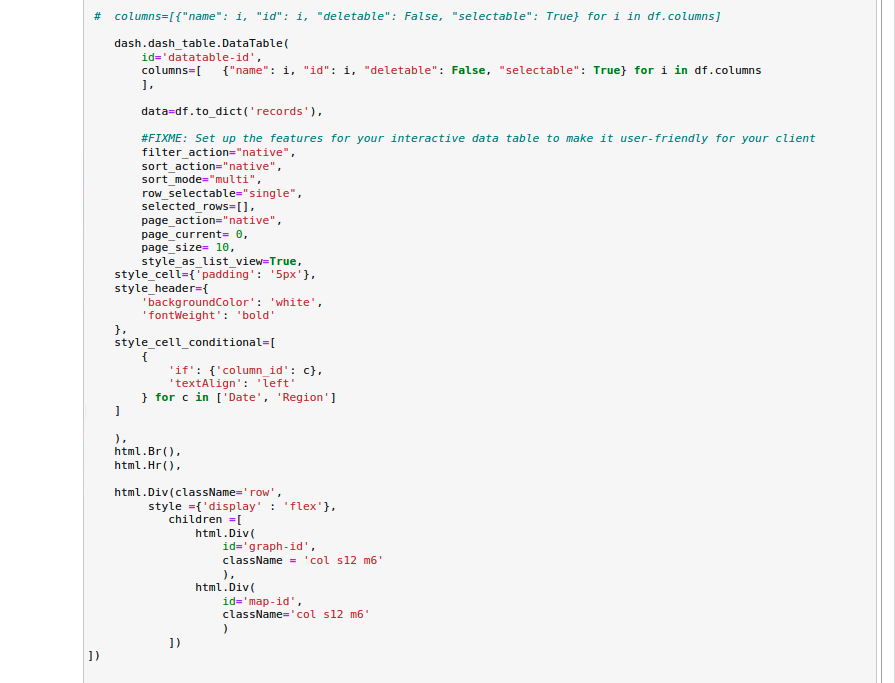
**Figure 11: Within Jupyter Notebooks, we import the necessary libraries, our CRUD module, pass the user credentials, and establish the data through the read() method.**

****

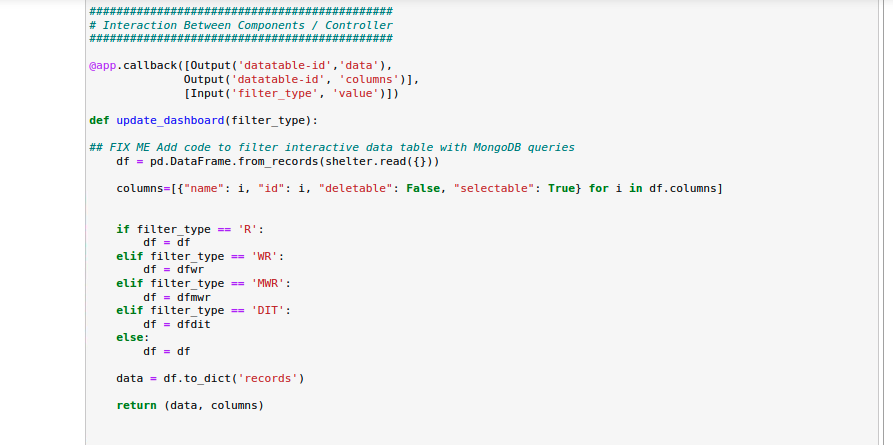
**Figure 12: Utilizing Mongo queries, we establish 4 total data frames, one as default (unfiltered data), 3 filtered data frames based upon specified requirements set by the client, which detail different conditions for various rescue style dogs.**

**

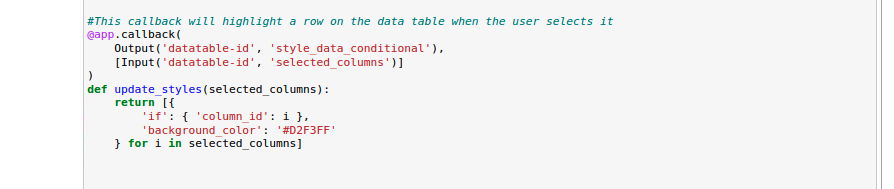
**Figure 13: Then the dashboard layout/view properties are set, including name, company logo, a unique identifier, and the radio buttons for data table filtering.**

****

**Figure 14: Then the properties for the data table are set, including the row selection functionality set to “single” and other formatting specifications. Additionally, space was reserved for both the graph chart and map, with two side-by-side divs.**

****

**Figure 15: The update\_dashboard() method incorporates the results of the radio-button selection and its respective filtered criteria to update the dashboard to reflect the users’ choice of rescue type dog.**

****

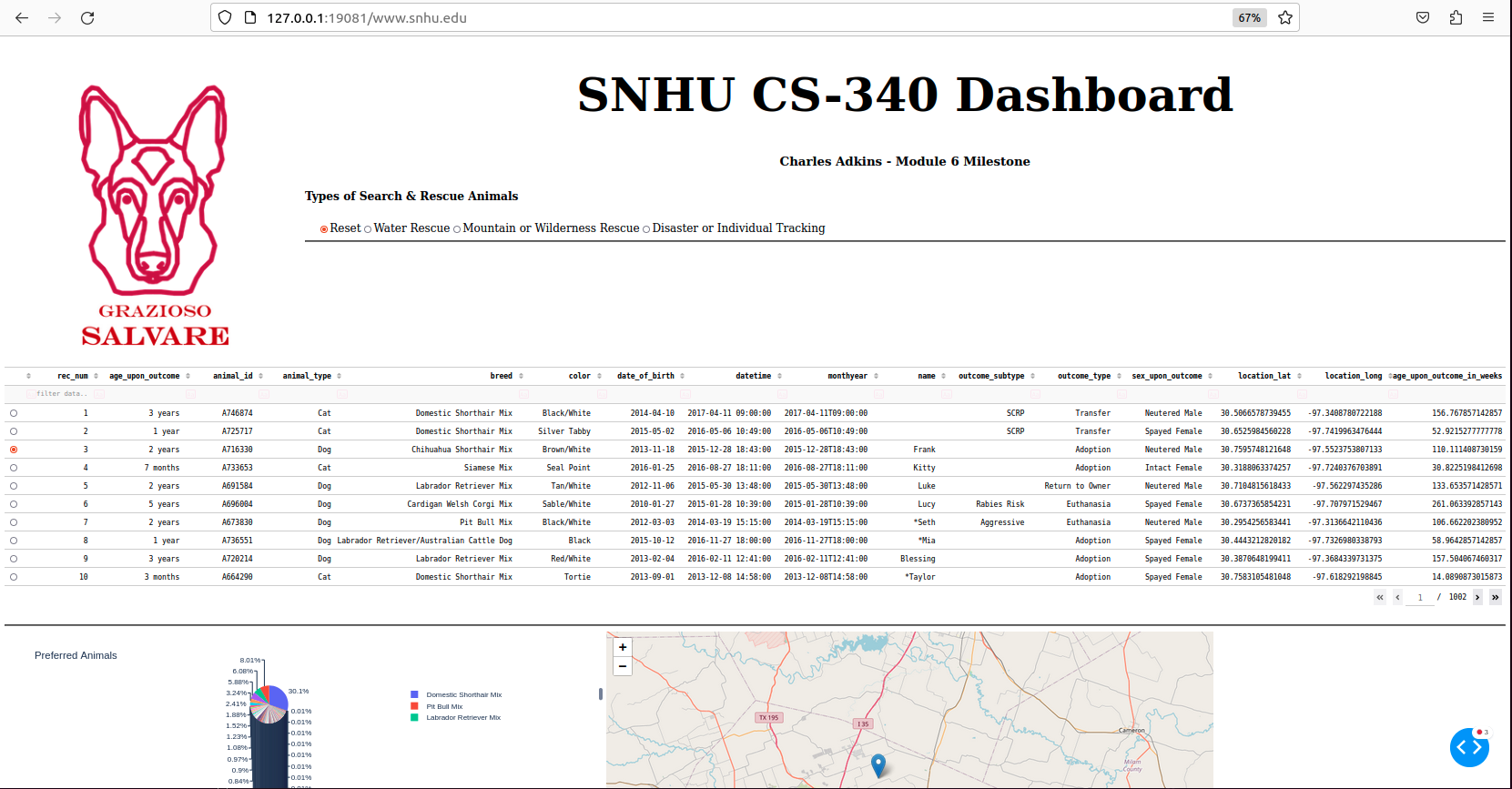
**Figure 16: The update\_styles() method adjusts the color of the data table if the user selects a choice, specifically, the row turns red, showcasing their selection.**

****

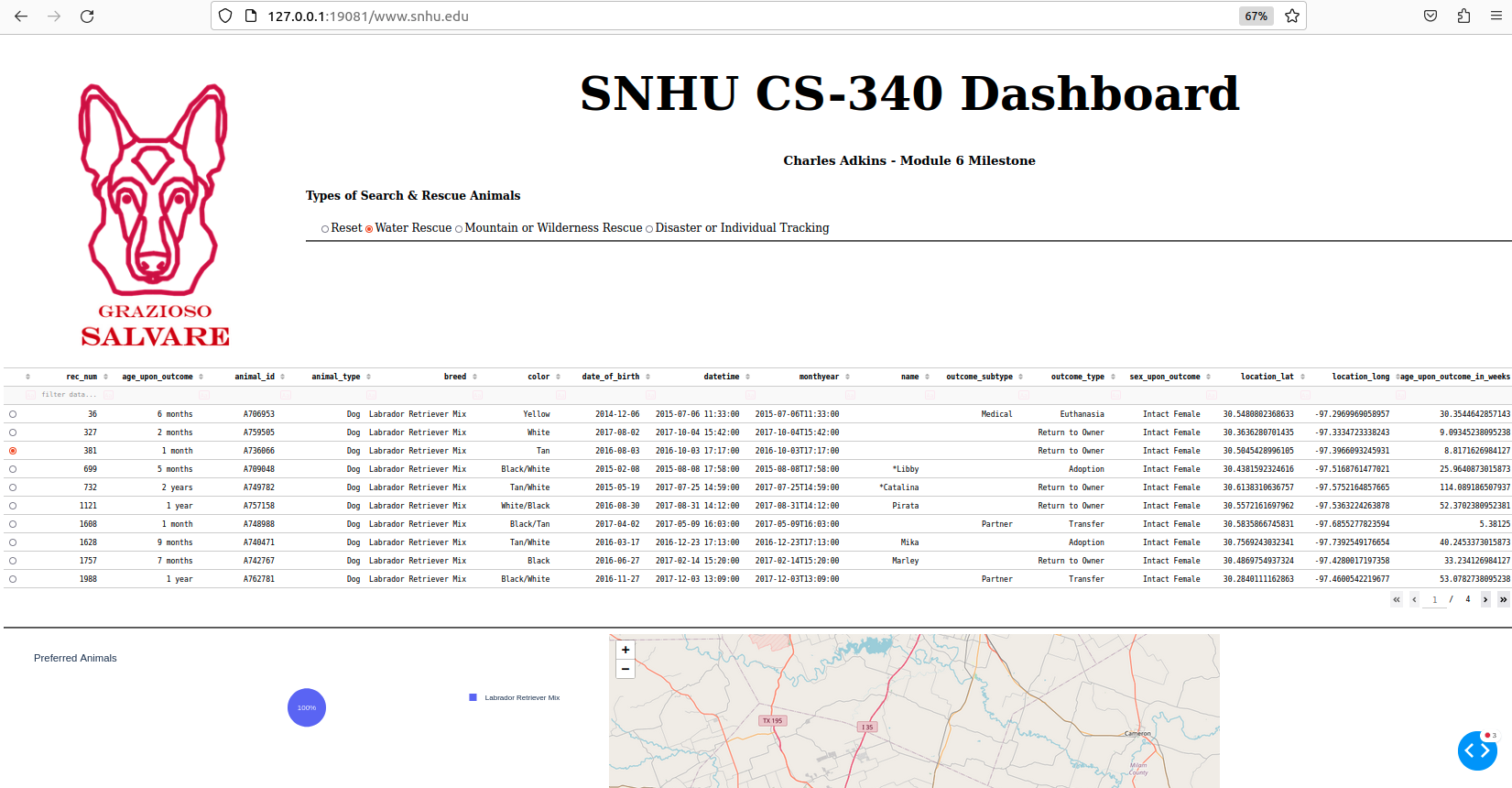
**Figure 17: The update\_graphs() method updates the pie chart, based upon the rescue dog type radio-button selection. It showcases the percentage of each breed for each rescue type, as well as the count of each.**



**Figure 18: The update\_map() method displays the map and the location of the animal selected within the data table, based their own unique latitude and longitude data.**



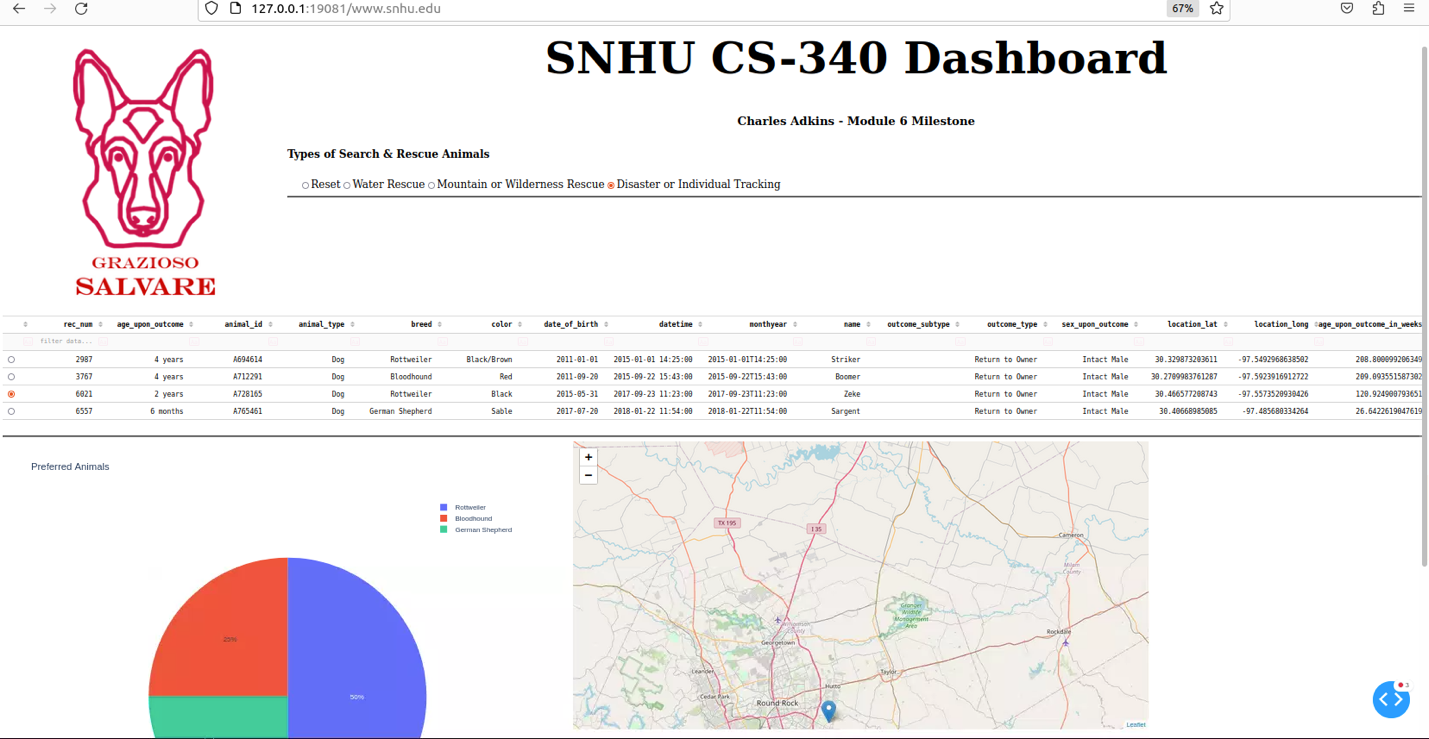
**Figure 19: The default display of the dashboard, when ‘Reset’ is the type of rescue animal, displays the data table, unfiltered, the totality of the animals by pie chart and the geo-location of the animal chosen from the table. The map will only display once a row item is chosen from the data table.**

****

**Figure 20: When ‘Water Rescue’ type is selected, the data table updates to display on the dogs that meet the internal criteria, outlined within the script, the pie chart updates with percentages and counts, and the map updates to display the selected location of the chosen animal.**



**Figure 21: When ‘Mountain or Wilderness Rescue’ is selected, the data table updates to display on the dogs that meet the internal criteria, outlined within the script, the pie chart updates with percentages and counts, and the map updates to display the selected location of the chosen animal.**

****

**Figure 22: When ‘Disaster or Individual Tracking’ is selected, the data table updates to display on the dogs that meet the internal criteria, outlined within the script, the pie chart updates with percentages and counts, and the map updates to display the selected location of the chosen animal.**

## Contact

Charles Adkins

**References**

Gillis, A. (2023). MongoDB. TechTarget. Retrieved June 1, 2023 from <https://www.techtarget.com/searchdatamanagement/definition/MongoDB>

W3schools. (n.d.). Python Tutorial. Retrieved May 30, 2023 from <https://www.w3schools.com/python/>

Jupyter (n.d.). Jupyter Notebook: The Classic Notebook Interface. Retrieved May 30, 2023 from https://jupyter.org/