# CS 340 README

## Animal Shelter CRUD Project

Grazioso Salvare assigned Global Rain the present project. They had requested that our group create an application to help locate and train the appropriate canines for the appropriate jobs. The customer had requested, among other things, that we make the code available as open source on GitHub. This implies that all code must be publicly available and reusable. This will enable a user to search for certain qualities by looking through the data in the databases of various shelters. To read a Python file, the system makes use of the PyMongo driver included in MongoDB. The customer requested that the code be reusable, hence the PyMongo driver was used. All the methods required for the functionality are included in a Py file that will be read in. The Py file contains a class that allows creating, reading, updating, and deleting from the database. These 4 methods are the core to CRUD.

The software will have a dashboard to make it easier to use and determine which dogs are the most visually suitable. A table with details about the pets in the database makes up the dashboard. You may use a drop-down menu to filter this table based on which jobs need dogs. For example, it will filter canines that meet the requirements for "Water Rescue" if that is the type of task that is needed. A geolocation map and a graph are also accessible beneath the table. The filtered dogs are displayed in the graph according to breed. Based on the latitude and longitude information for the specified dog, a pinned location indicates the position of the highlighted row on the geolocation map. Regarding the contents included inside this application to work we had to integrate a few libraries.

This project took a long time to complete. which have its share of difficulties. The creation of the database and the creation of an authorized user for it came first. I'm learning databases relatively recently; thus everything is from scratch. It took some time to figure out how everything fits together. I generated a Python file that contained the CRUD functions for the database's functionality after creating the database. I've only used Python once in my whole computing career, so remembering the syntax to write it correctly took some time. That was the hardest aspect of developing the Python file. The dashboard was the next goal after the database functionality was finished.

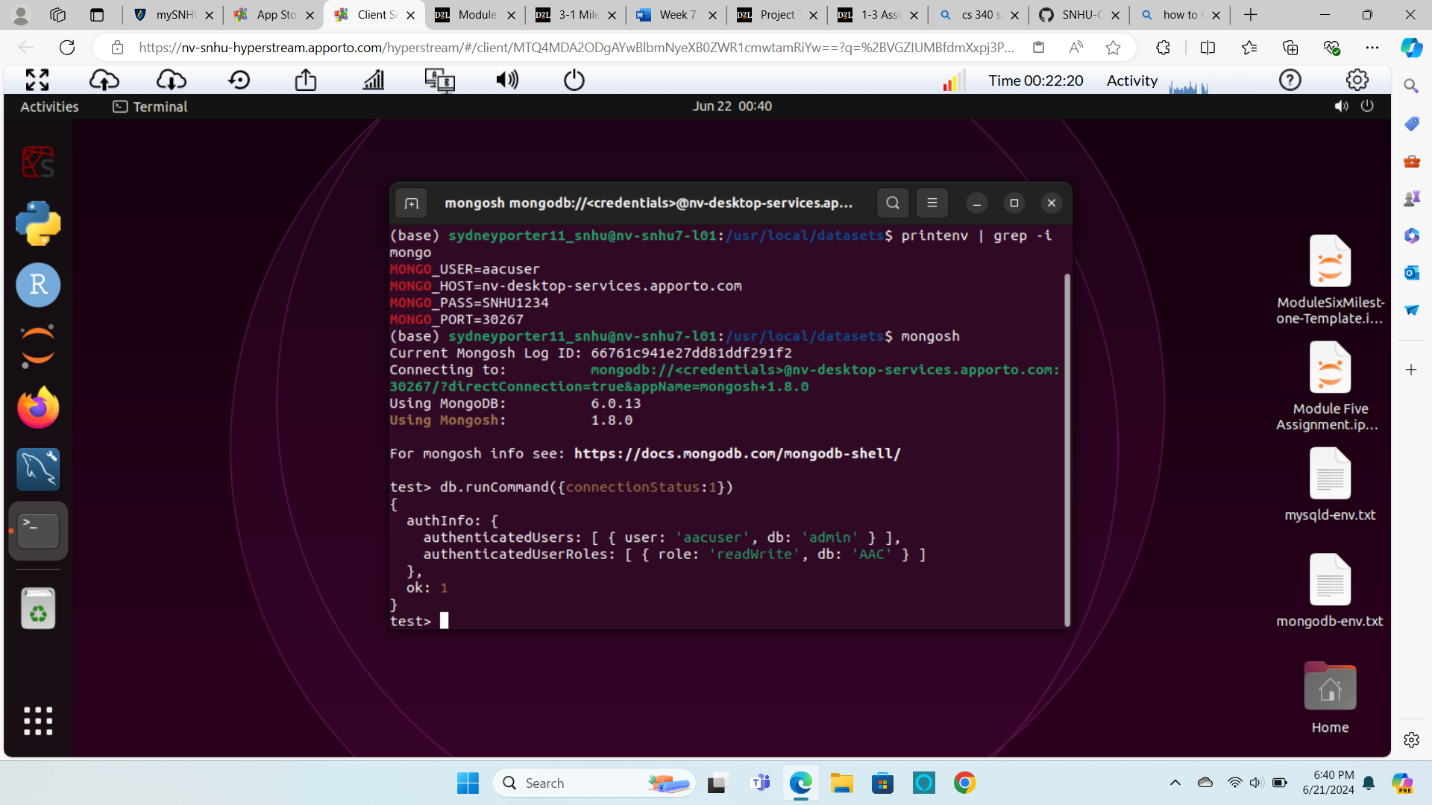
The application's dashboard was created primarily for the client's visual usage. It presents the database's contents in an easily readable visual detail according to the client's demands. I encountered several difficulties with the dashboard. The main issue I had was getting my callbacks to function correctly. I battled with it for days until I was able to get the filtering to function properly, which eliminated the callbacks. The customer can select from three distinct sorts of rescues using the drop-down menu. Based on the qualities the customer initially agreed to work with, it filters the database to display only dogs that are suitable for the rescue breed.

## Motivation

Not every dog possesses the ideal qualities for every task. It is necessary to have a method in place to identify the precise characteristics that a dog should possess in order to teach it. After the data is analyzed, this system will output a list of the dogs that match the proper profile. The software's ability to efficiently employ the functions and query via the database made it the first choice.

## Getting Started

To get a local copy up and running, follow these simple steps:

* Importing the database is required. Data from a CSV file can be used for this. It will include all characteristics and information about the various products in the database. We imported the aac\_shelter\_outcomes.csv file in Linu for this specific project in the following manner:
* 
* We must sort through each item in the database in order to read through it. By making an index, we do this. The findOne() function in the Mongo shell served as one example for us to comb through for our project.
  + db.animals.findOne()
* In order to grant access to the functionality that the system will employ, a user other than the administrator must be established. In this assignment, we applied the CRUD "C" and "R" concepts. (Write, Read)   
    
  completed in Mongo Shell

db.createUser({ user: "aacuser", pwd: "SNHU1234", roles: [{ role: "readWrite", db: "AAC"}]})

After that, it has to be verified.

MONGO\_USER=aacuser

MONGO\_PASS=SNHU1234

printenv | grep -i mongo

**Installation**

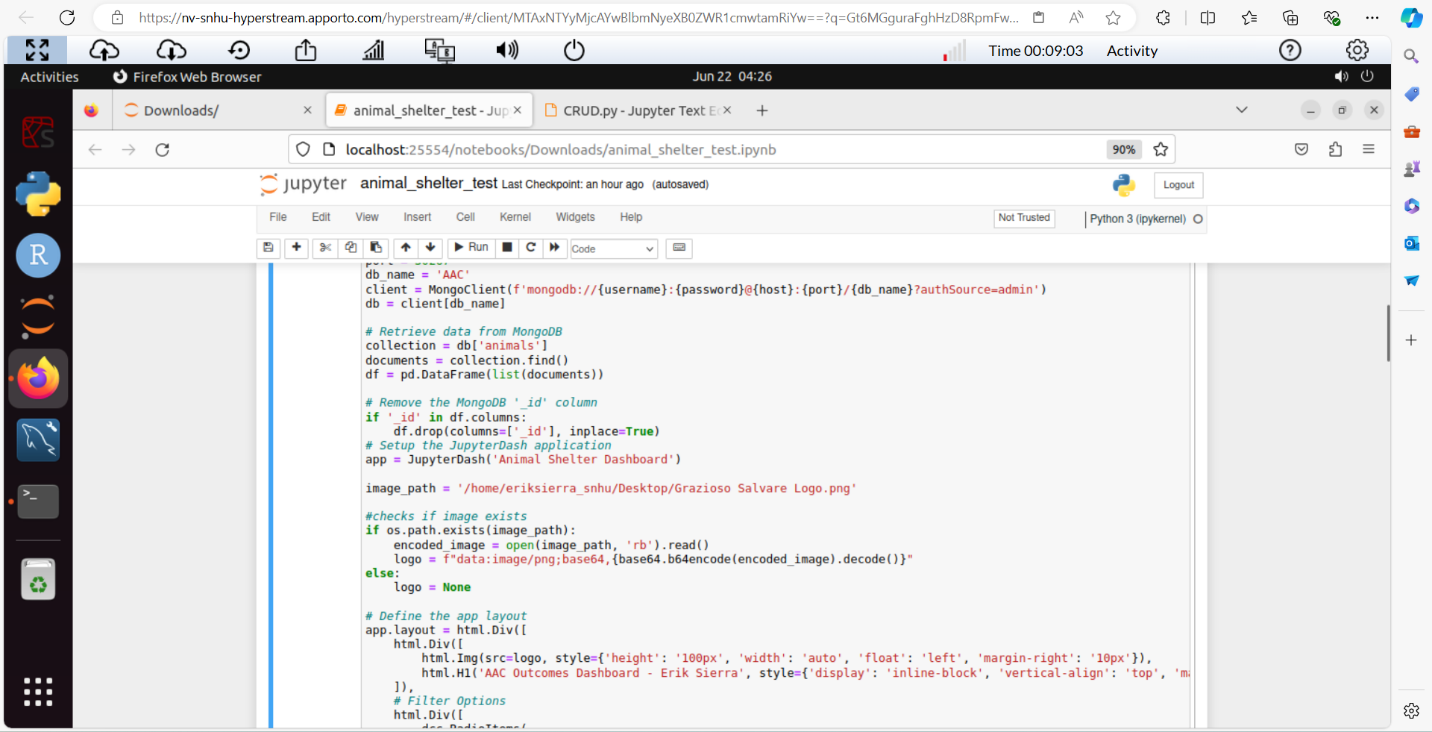
Install Python: Ensure Python is installed on your system. You can download it from python.org.

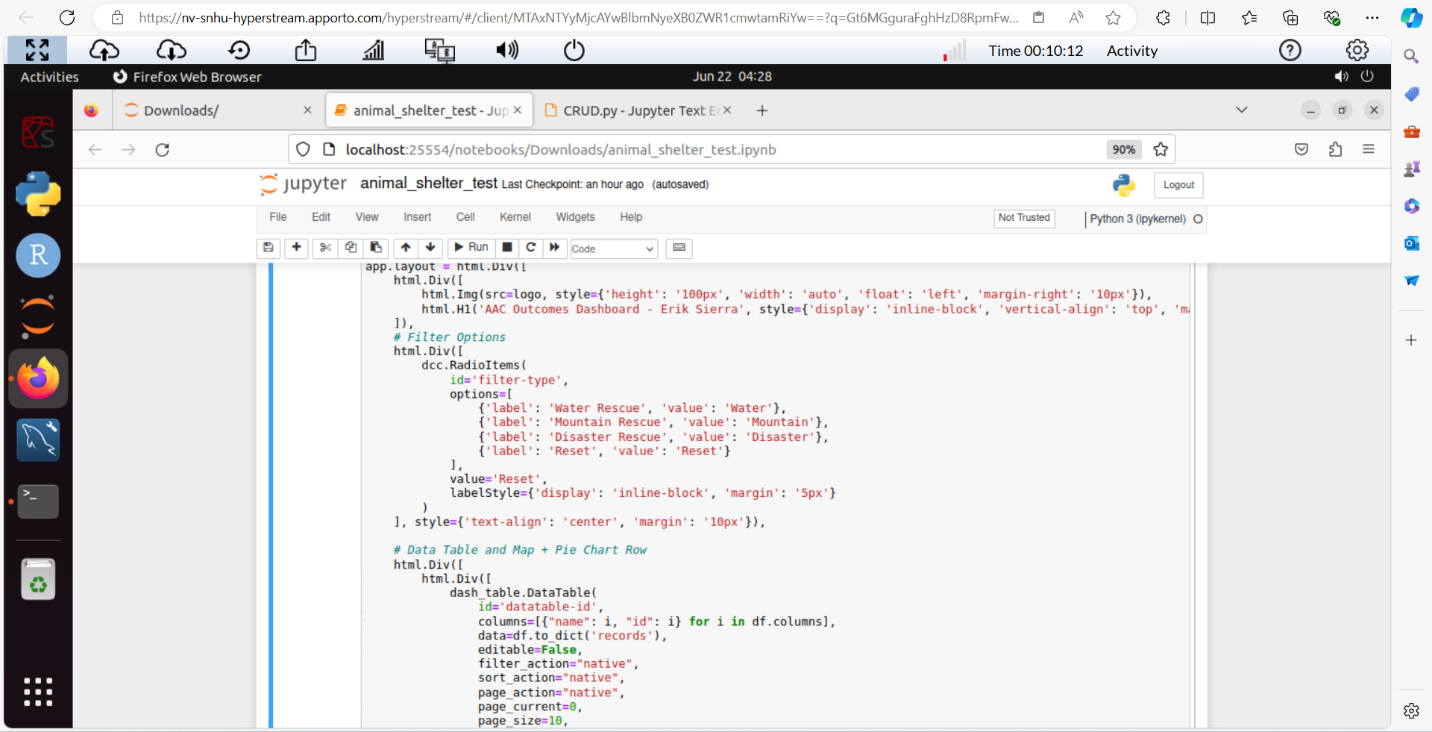
Install MongoDB: Follow the instructions on the MongoDB website to install MongoDB on your machine.

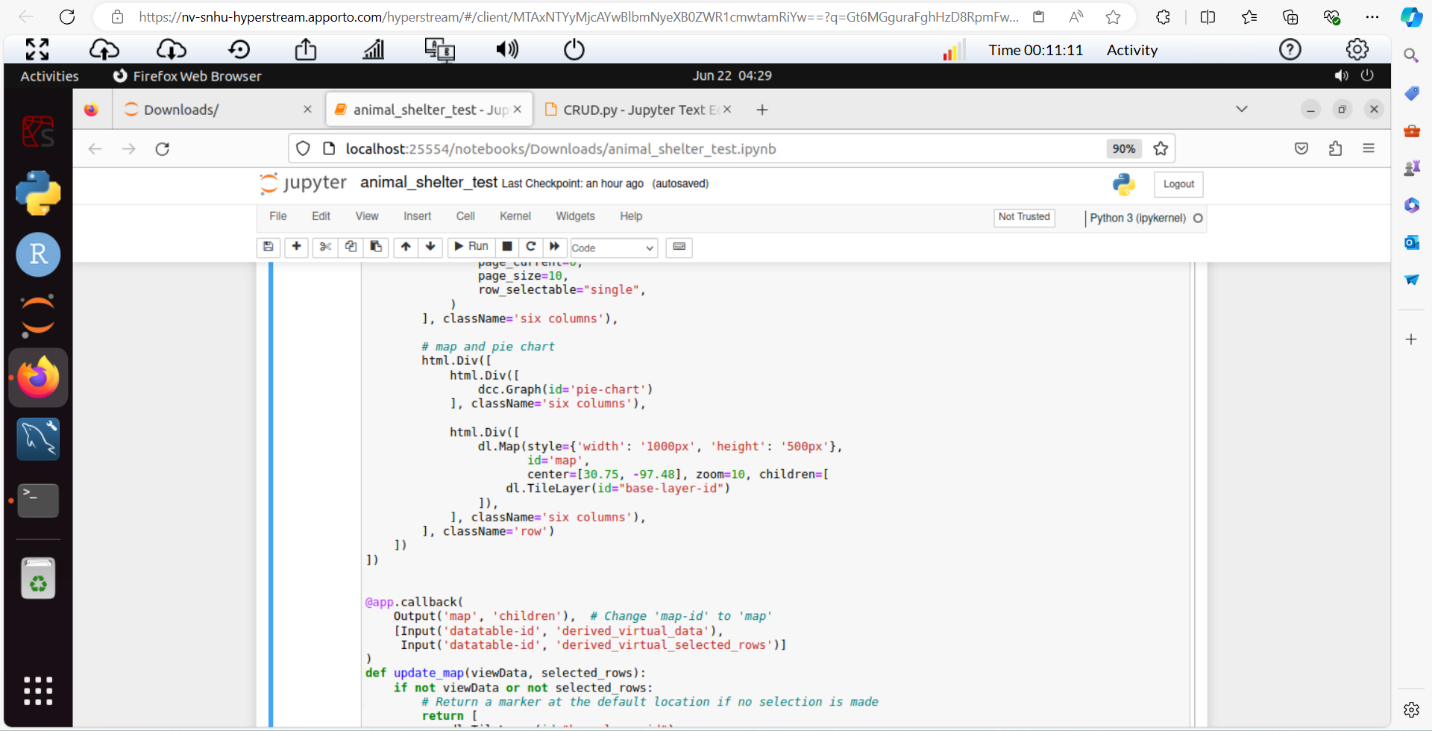
Install pymongo: Install the pymongo library to interact with MongoDB from Python:

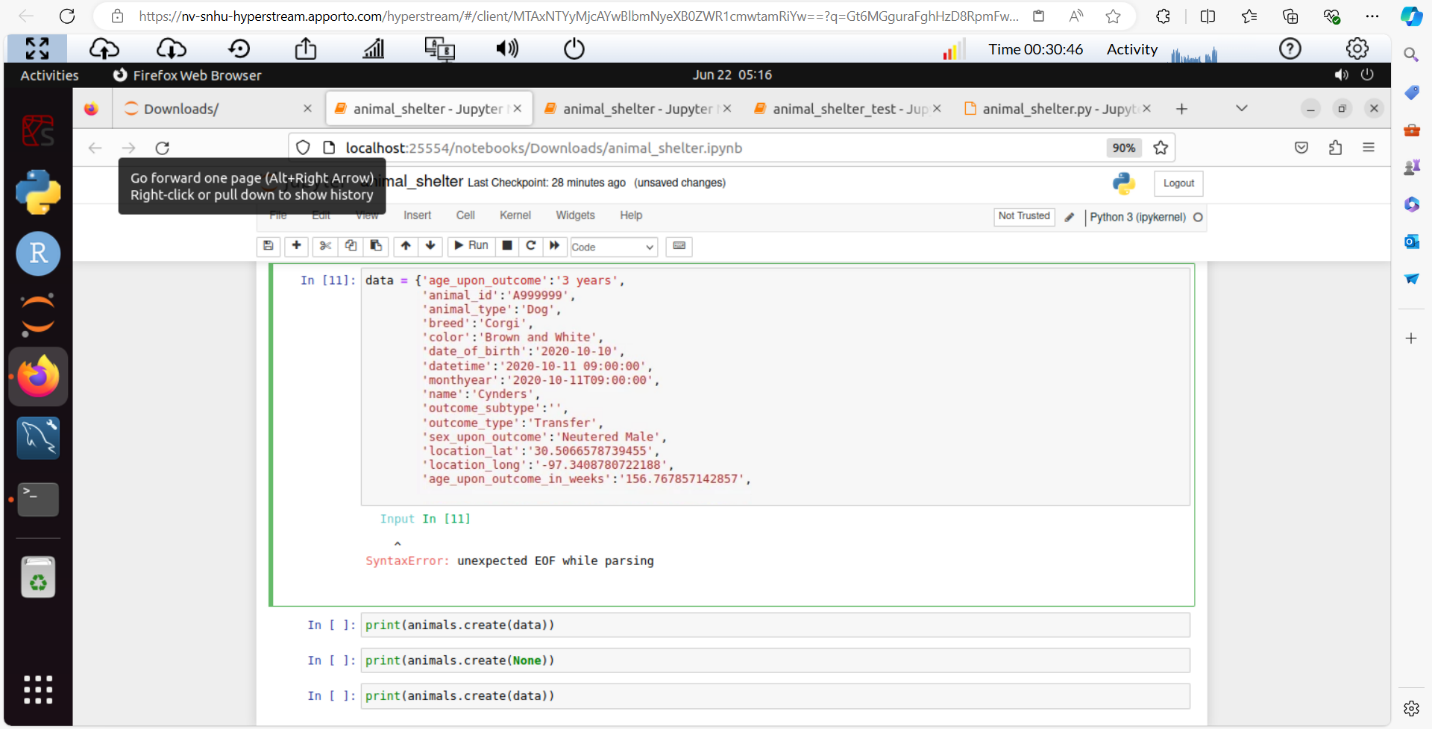
## Usage

### Code Example



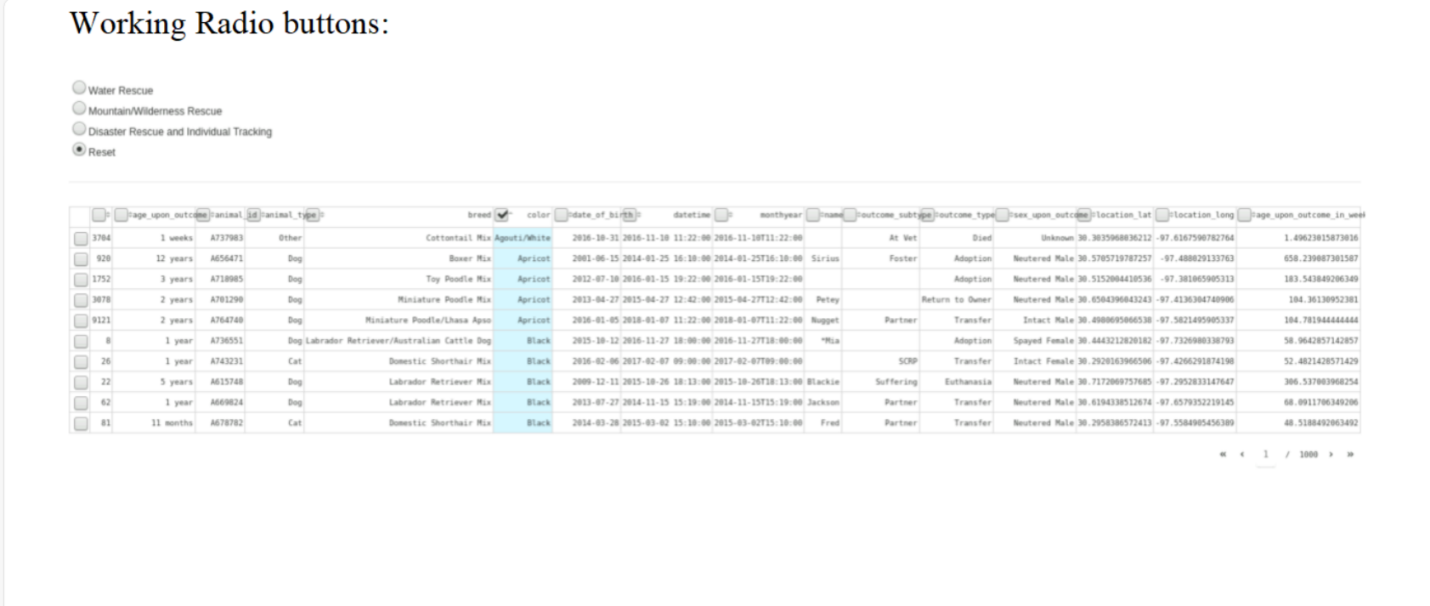


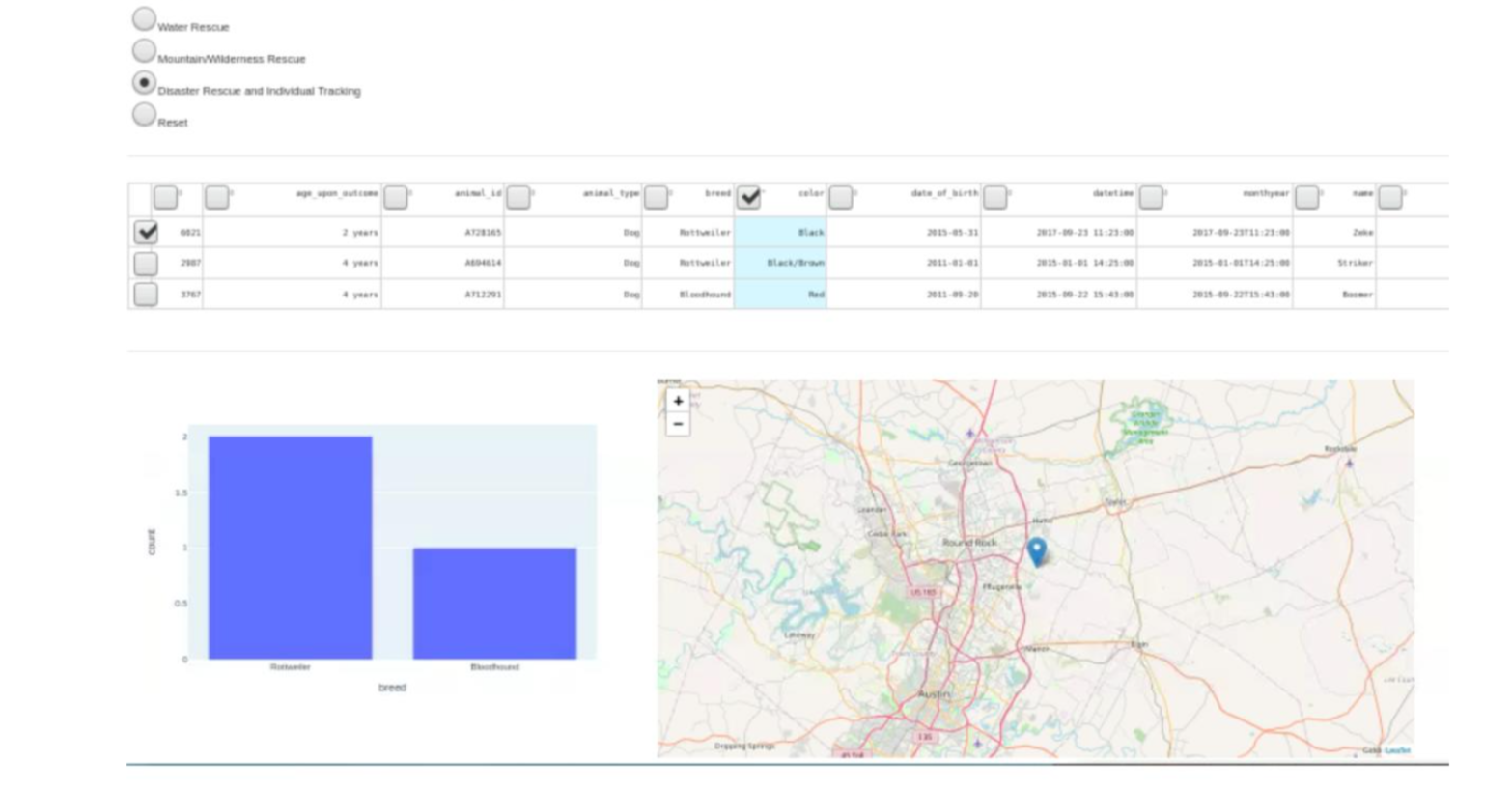




A CRUD class that can add, edit, remove, and read documents from the database is part of the project. The following lists the usage and testing examples for this class, which may all be utilized in the dashboard as needed:

* The constructor for the CRUD Class Initializes the MongoDB server.
* Create() adds a new record to the database. need one parameter in order to add data. It needs to be a dictionary.
* Read() all of the search's documents are located and printed. uses a single argument as its search parameter. Can be as lengthy as necessary, however for queries, it needs to be in the proper Mongo format.
* Update() edits an existing document. accepts two inputs: the data that has to be changed and the search for the documents that need to be altered.
* Delete() removes the active document. assumes that the search query is one parameter. can be as lengthy as necessary, but for queries, it needs to be in the proper Mongo format.





**Tools Used in Project**

* The document database management system that was utilized was MongoDB. The data was simple to index and handle because to its documentation.
* I ran MongoDB locally and the virtual Linux server using Docker.
* Throughout the project, Python was utilized to interface with the Mongo database using PyMongo.
* A web-based tool called Jupyter Notebook allows users to create and share documents with text, equations, and code. This was a straightforward program that was used as an easy method to develop and run the Python code; however, it is not required to utilize any Python IDE.
* A Python framework called Plotly Dash is used to create interactive webpages. It does not require knowledge of HTML, CSS, or JavaScript because it simply utilizes Python to construct and format the website. This is the program that manages the data table, charts, and maps on the webpage.

**Challenges**

The hardest part of this project for me was getting the code to run. I had a lot of issues with Apporto and had nothing but issues with it.

## Roadmap/Features

Future development will focus on MongoDB initialization. Currently, the database must be hardcoded. In future releases this will be able to be passed to the constructor. Also, the constructor will try and initialize the database for each new object. A singleton will be added in future releases.

## Contact

Sydney Porter: Sydney.porter1@snhu.edu