Jacob Perry

5/27/2024

Assignment 4-2: README

**About the Project/Project Title**

This project is being created to make finding adequate candidates for rescue-animal training more seamless and efficient. Our client has partnered with a nonprofit agency who runs five animals shelters around the Austin, Texas region. Our hope is for this project to allow our client to streamline their search for capable animals to train by creating a seamless user interface that interacts with a MongoDB database in the background using Python scripting to do so. The freedom of using a NoSQL database versus other SQL databases allows much more versatility for applications like this. We are using the Dash framework for many reasons, one of which being how efficiently it can handle large datasets.

**Motivation**

This project will help our client and directly help in events like disaster relief efforts. Creating a way to help our client, Grazioso Salvare, more quickly identify dogs that are suitable for search and rescue training will allow more animals from these shelters to be adopted. Not only will they be getting adopted from the shelters, but they will also play an immensely important role in saving countless lives. They will be trained for rescue efforts in water, mountains, wilderness, and even post-disaster events.

**Getting Started**

To get this project running locally on your machine, you must first create an environment where you can run Python 3 and MongoDB. Python is free to download and can run on your machine in minutes. MongoDB has paid subscriptions for enterprise level usage, but the free MongoDB Atlas should suffice for the needs of his project. You will also need a Python compatible application to create scripts for the modules to interact with the user interface. In this example, we have been using Jupyter Notebook.

Our database is created based on data supplied by the animal shelters. This data has all the relevant information we need for our searches, such as breed and age. We will set up user authentication for any users who plan to use this project. We can create database admins, read only roles, read and write roles, and many more configurations. All roles will require authentication through username and password protection.

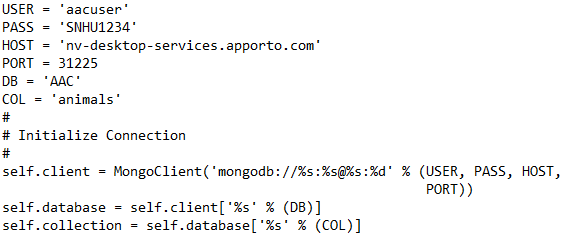
Our Python module uses CRUD functionality. Meaning users can create, read, update, and delete from our database. To create a database entry, you must utilize our create() method. This method takes a dictionary as a parameter and enters it into the database. You can pass many key value pairs within your dictionary, such as animal\_id, breed, color, age, and many more relevant tags.

The read() method will allow you to query all entries within the database. You must simply pass in a key value pair and you will receive all entries that match the parameters passed into read(). An example of this could be looking up an animal via animal\_id.

The update() method allows you to query and existing entry with a key value pair and update it to contain new information that is passed as a second key value pair parameter. You can also update multiple entries by setting a third parameter to True.

The delete() method allows you to delete one or many entries by querying the corresponding key value pair that will find the entry or entries to delete. This method also makes use of a parameter to indicate if you want to delete one or many entries.

You will also need to pass in your correct information to the Python module to be logged in as the correct user. You will need information like username, password, port number, your host, and the proper database and collection names.



**Installation**

The latest version of Python can simply be downloaded from their website.

<https://www.python.org/downloads/>

All of the various MongoDB plans caa be explored on their website. The free version will work for our needs within this project.

<https://www.mongodb.com/atlas>

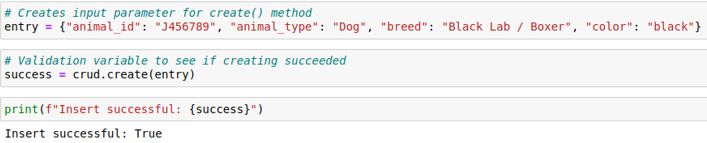
All of the steps and necessary downloads to run Jupyter Notebook can be found directly on their website. It can be installed directly from the Python shell, using the ‘pip’ command.

<https://jupyter.org/>

You will also need access to the raw data from the animal shelters. This data will be supplied by the shelters, and we will import them into MongoDB as a CSV file, using the mongoimport tool.

**Usage**

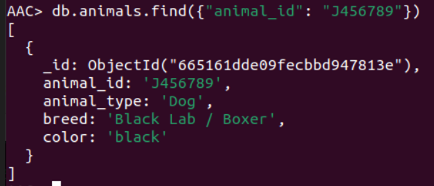
Here is an example of a Python script created in Jupyter Notebook creating an entry in our animals database.



We can double check the success of our creation by querying for it in the same notebook.

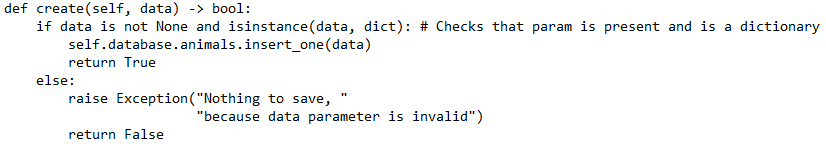


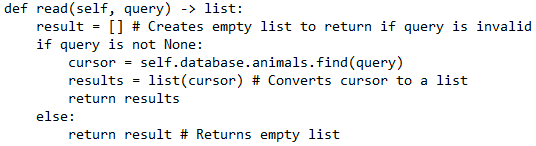
We can also query for the same entry directly in our Mongo shell. This check fully proves the functionality of our create and read methods.



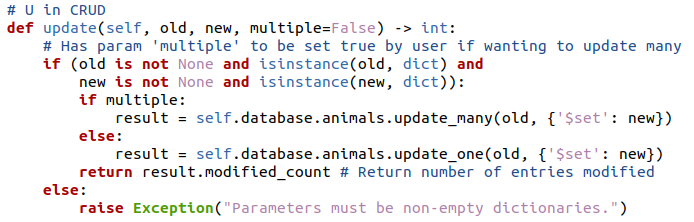
**Code Example**

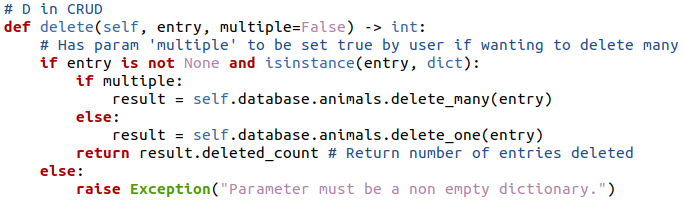
Our create() method inserts a document into our MongoDB database, given a valid dictionary is passed as a parameter. Our method will return True if it was successful, and False if it failed to create.

Our read() method allows users to query the database using as key value pair. We use the find() function, rather than the findOne() function to allow users to see all matches to their query, rather than the first one. If successful, a list populated with all matches is return. If the query fails, an empty list is returned.



Our update() method allows users to query the database for an existing document(s) and allows a second key value parameter to be passed in to update the initial entry parameter. We can also pass a third parameter in which is ‘multiple=False’ by default. We can set this to true if we wish to use the update\_many() pymongo function. This method returns the number of entries that were modified.

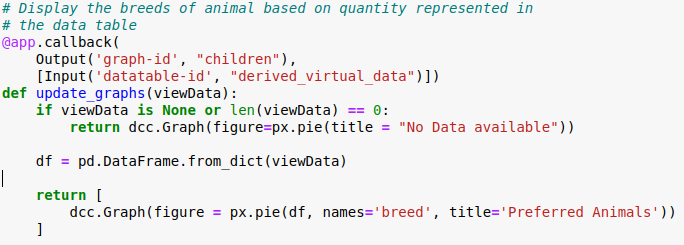
Our delete() method allows users to pass in a key value pair that they wish to delete from the collection. In the same manner as the update() method, users can also set the ’multiple’ parameter to True if they wish to delete\_many instead of delete\_one. This method returns the number of entries that were successfully deleted.



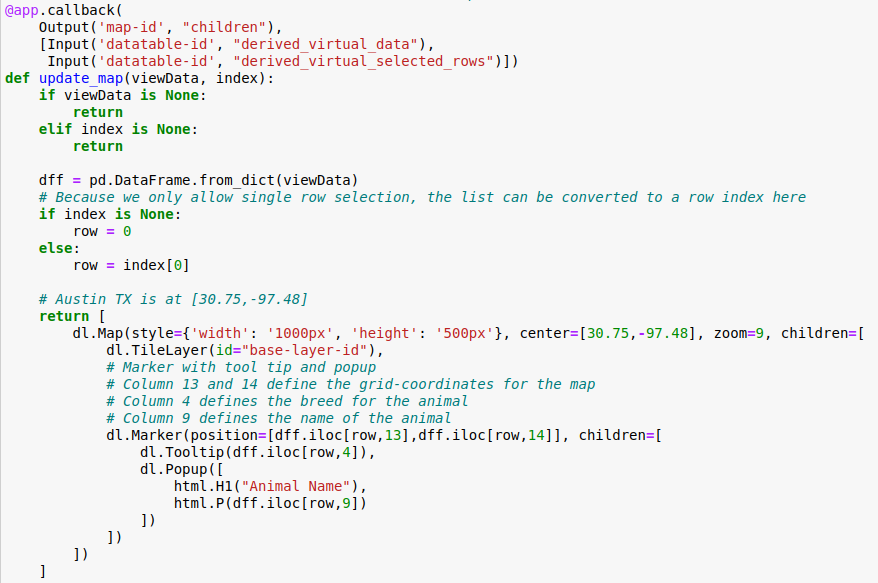
We make use of the read() function in our dashboard by passing in queries to help us filter out the different dogs suitable for various types of rescue training.



One way we display this data in the dashboard is with a pie chart using plotly express. This provides us with a great visualization of the breakdown of the different species suiting certain types of rescues.



We also make use of a geolocation chart to help the end user better visualize where these suitable dogs are located. We determined the best center point to be Austin, TX since all the rescues are based around this area.

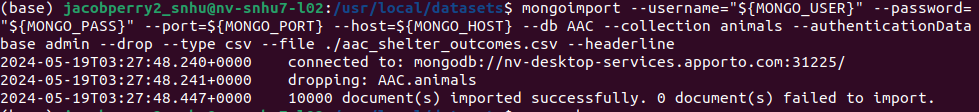


**Tests**

The “Usage” section of this README can also serve as a functionality for testing within this project. The “Usage” section demonstrates how to create an entry. It then depicts how to query for this new entry. It also shows how to verify that it went into the proper database by querying the new entry directly from the Mongo shell within the AAC database in the ‘animals’ collection.

**Screenshots**

This screenshot demonstrates importing the Animal Center data set into MongoDB as a CSV file using mongoimport.



This screenshot shows the login process for the user account ‘aacuser’ we created.

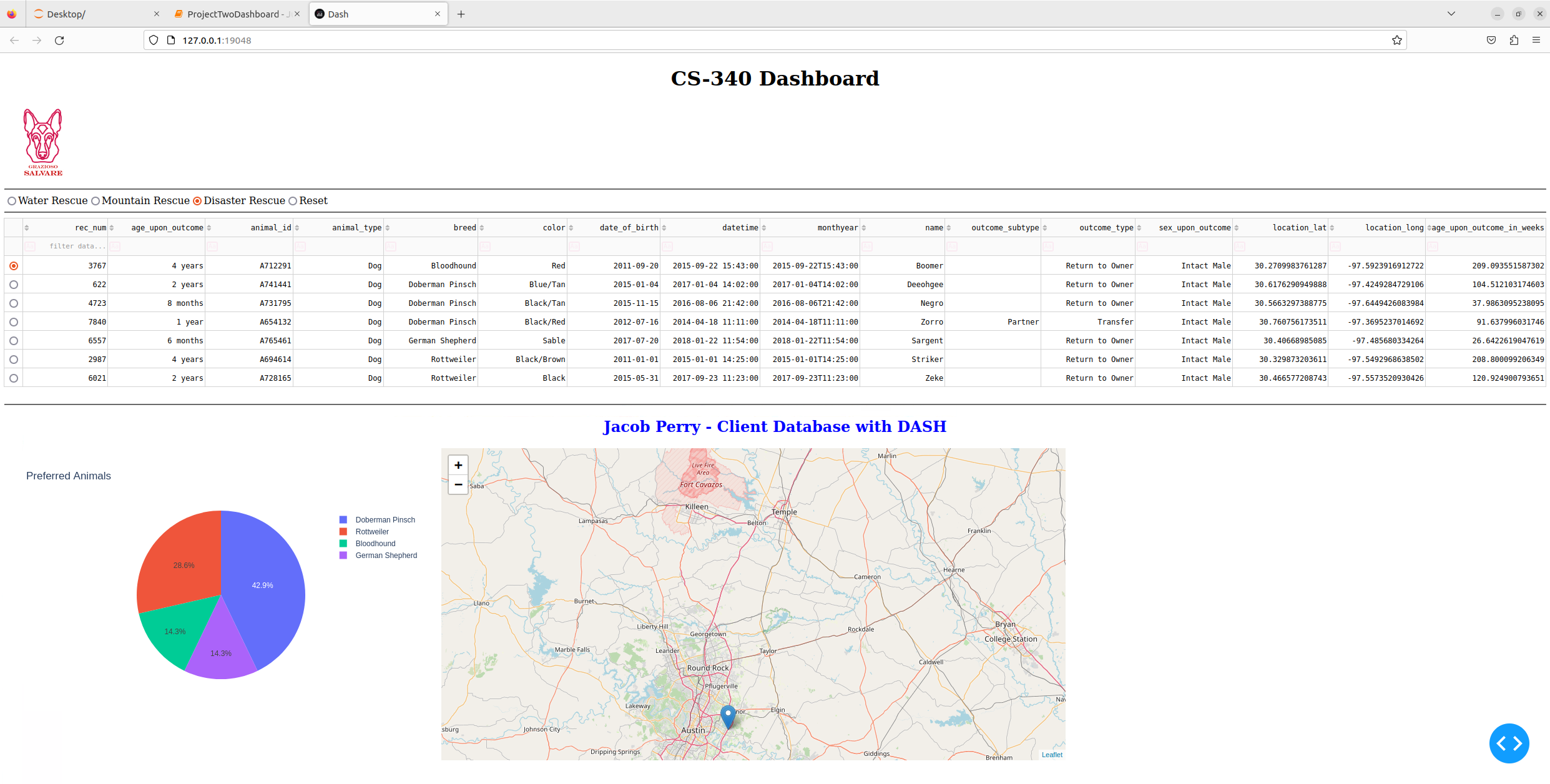


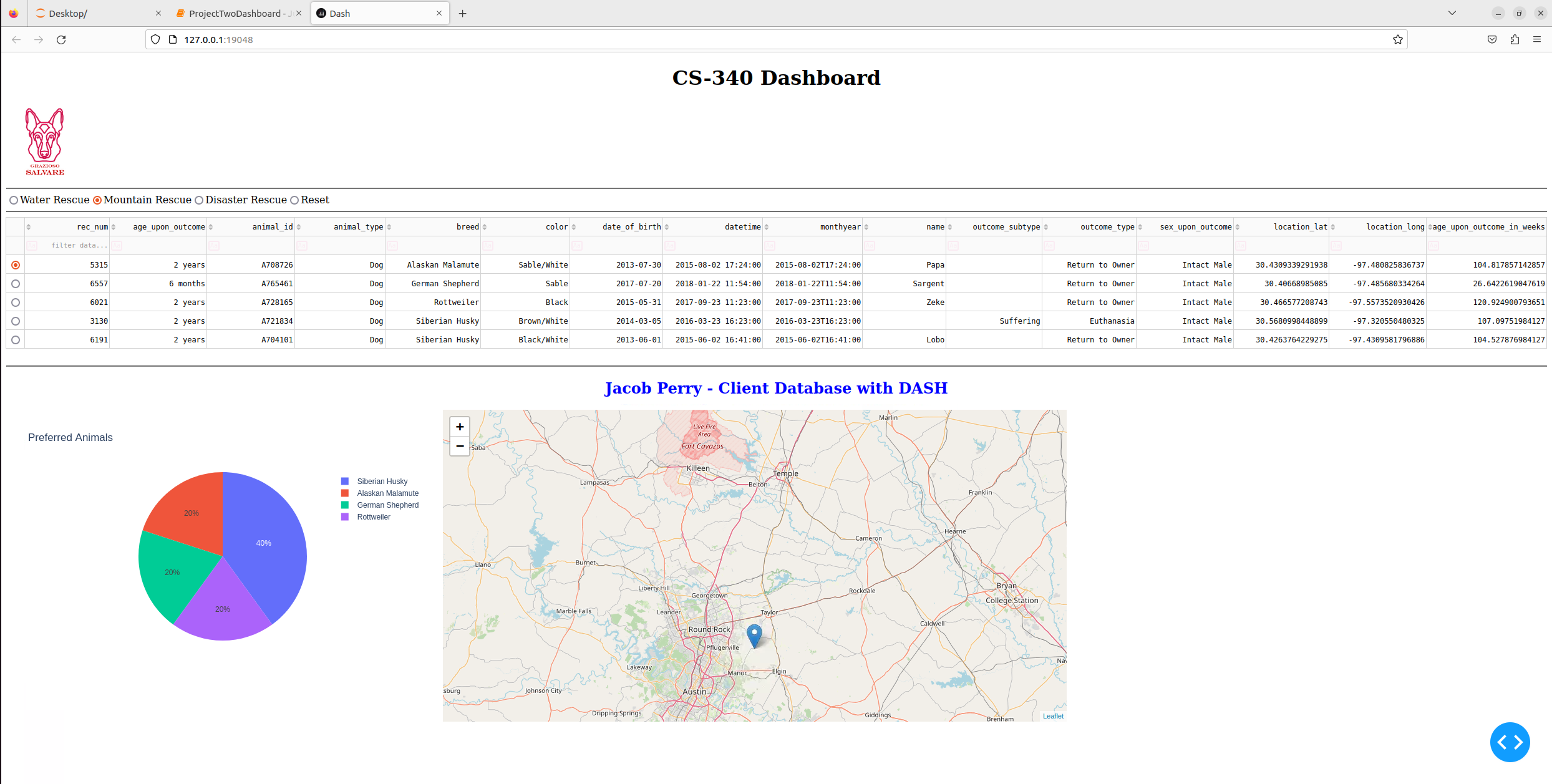
These screenshots provide our entire Jupyter Notebook script when we were getting ready to begin work on the dashboard. They contain the code for importing our CRUD module, creating an instance of our AnimalShelter object, and properly executing the create(), read(), update(), and delete() methods.

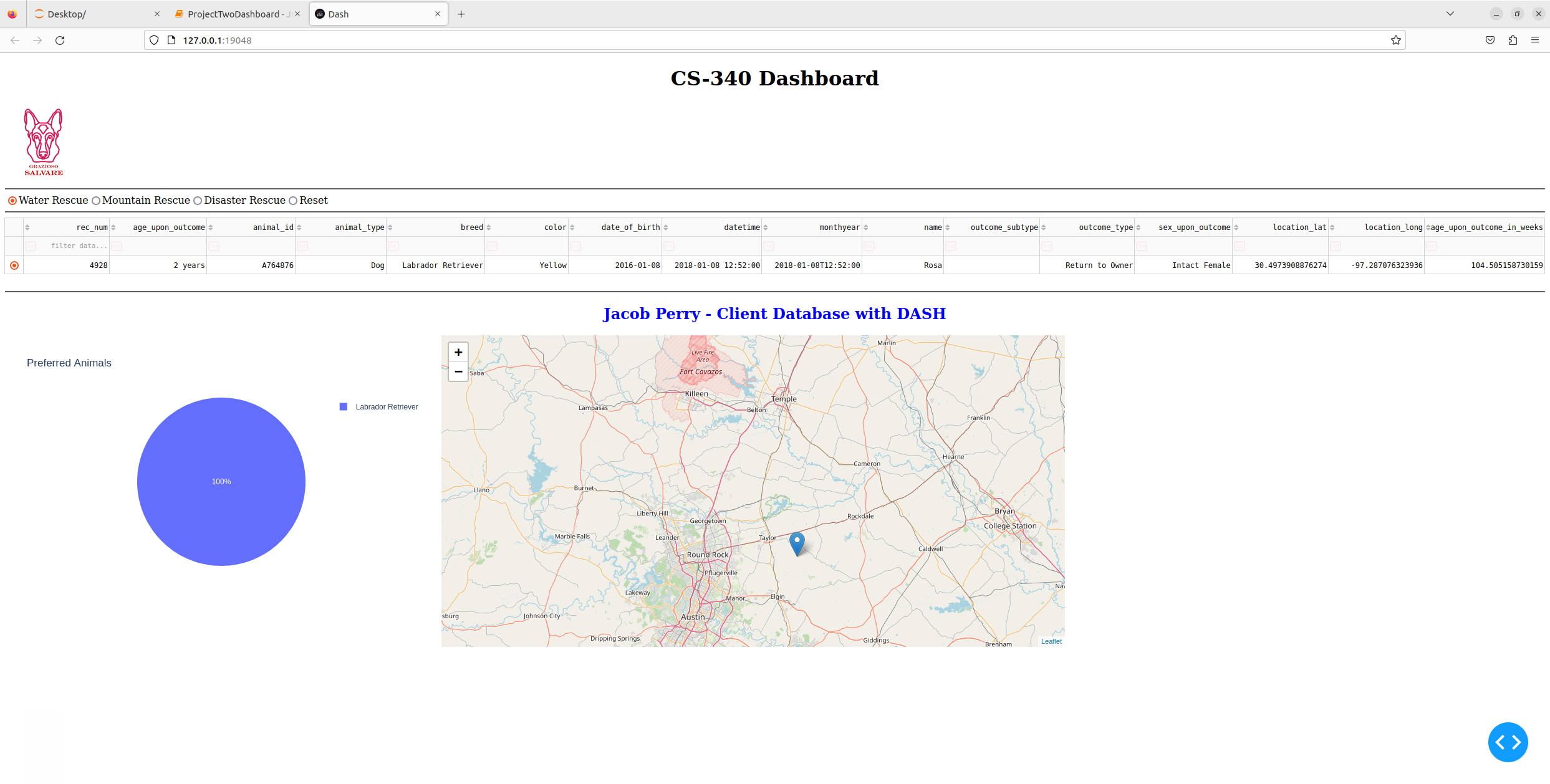


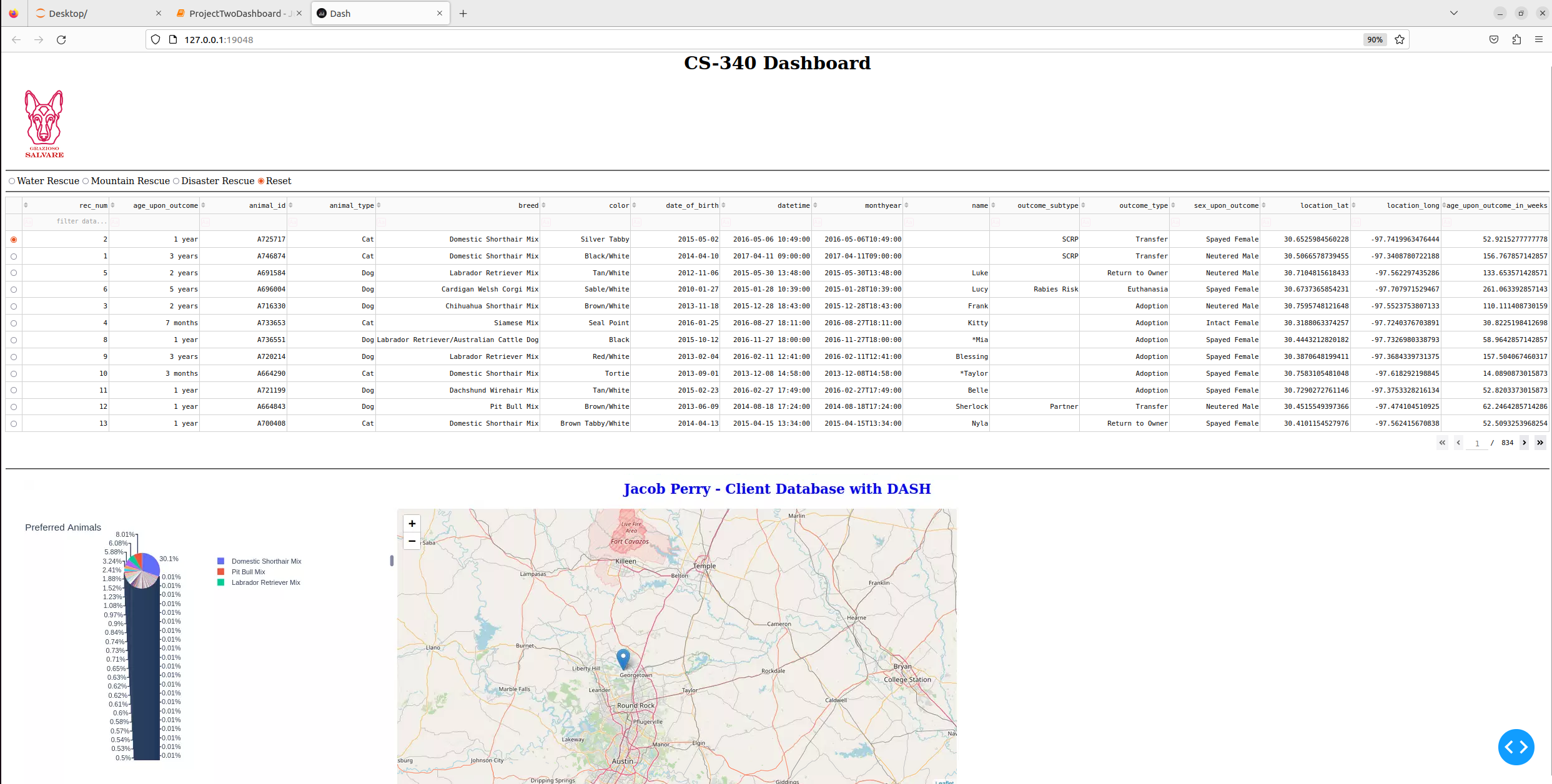


The following screenshots show the fully functional dashboard displaying the results from each of selected rescue type queries, along with the unfiltered view (Reset).









**Contact**

Jacob Perry

Jacob.perry2@snhu.edu