# CS 340 README

## About the Project

The purpose of this project was to create an interactive dashboard for Grazioso Salvare to be able to look at animals that are sheltered at the Austin Animal Center.

## Motivation

The motivation for this project was to make an easy way for Grazioso Salvare to search for dogs to be able to train for search and rescue missions through an interactive web application.

## Installation and Getting Started

**Tools needed:**

* Python version 3.6: <https://www.python.org/downloads/>
* MongoDB version 4.2.6: <https://www.mongodb.com/docs/v4.2/>
* Pymongo version 4.2: <https://pypi.org/project/pymongo/4.2.0/>
* Dash: https://lyz-code.github.io/blue-book/coding/python/dash\_leaflet/
* Plotly: https://plotly.com/
* An account to use for accessing the AAC database with read/write permissions.
* The Austin Animal Center CSV

To get started, the user will need to have Python, MongoDB, and Pymongo installed. Dash and Plotly can be imported using import commands.

Then the user will need download the Austin Animal Center CSV and upload it on the MongoDB server with the mongoimport command, example of how to do this is in the “Screenshots” section of this README. The user will also need an account to access the database that has read and write permissions, and this is also explained in the “Screenshots” section.

**Dash and Plotly Frameworks Explanation:**

Dash is a framework used by developers to show data. Dash helped me to create a visual dashboard representing data, which was the animals at Austin Animal Center. Plotly is a Python library that can be utilized to create interactive graphs and maps. In this case, plotly helped me to create a pie graph and the geolocation map for this project.

### Testing Explanation

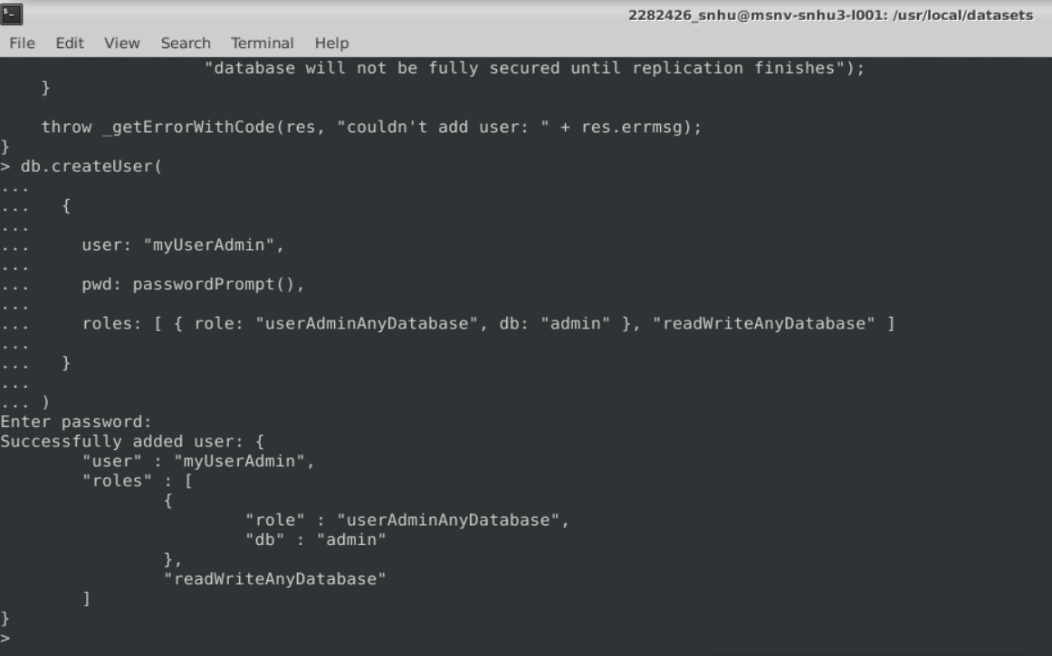
When I ran my tests, I used Jupyter Notebook and imported the pieces of my code into it. I tested the create, read, update, delete, invalid documents, valid and invalid queries functions. When it came to testing, this is where I struggled a bit. I overcame this issue by looking up guides on how to use Jupyter Notebook. I also had to test the dashboard to make sure it worked and the only problems I ran into were with indentations being incorrect.

### Screenshots

**Importing the AAC database:**

Here is an example of how I imported the database using the “mongoimport” command. With the ”mongoimport” command, I was able to upload a CSV file of each animal in the collection from the AAC database.

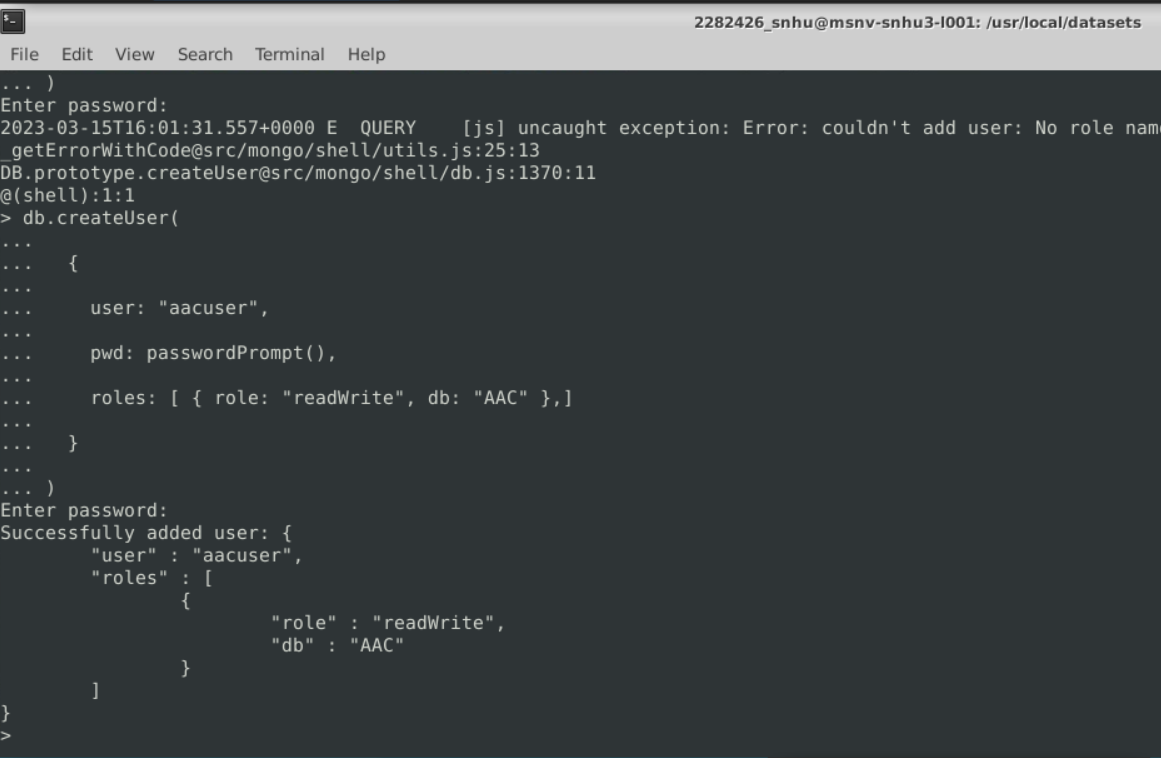
**Adding and Testing User Authentication:**



Here is where I was able to create an admin user called “myUserAdmin” using “createUser” for authentication purposes. I am now able to perform admin action and not have to worry about someone getting into my account since I have a password set-up now.

Now, with the “mongo --authenticationDatabase” command, I can now login to my admin account.

**Adding and Testing User Authentication for the AAC Database:**



This is how I created another user using the “createUser” command as I did previously with admin but now, I made an admin account for the AAC database. This account is only allowed to read and write to the AAC database.

Here is where we test the sign-in for the AAC admin account with “mongo --authenticationDatabase” command.

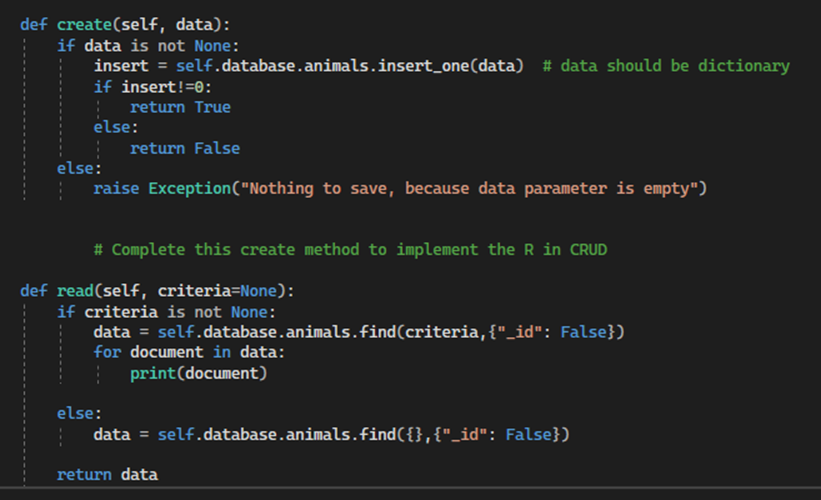
**Code Sample for Sign-In:**

Text

Description automatically generated

For the sign-in function, the user must enter their username and password along with their port number to access the database.

**Code Sample for Create and Read functions and Tests:**



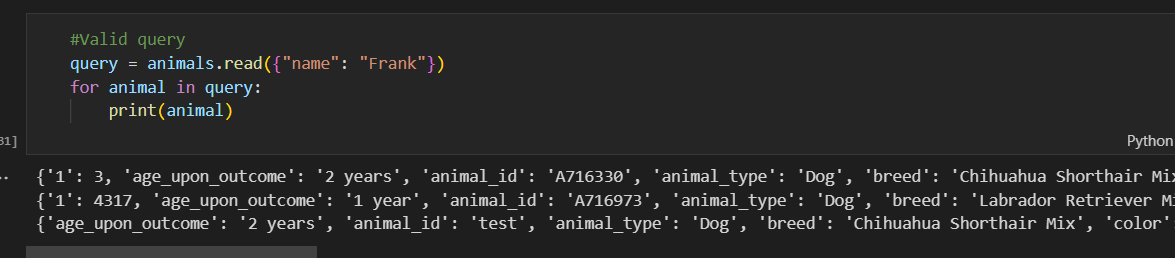
The next snippet is for the create and read functions which allow the user to create an entry and to also read that entry. When using the create function, if there is no data in the system that matches what the user has added, then the entry is accepted. If it is already there though, an error will be thrown. I struggled to understand how exactly I would go about making a read and create function, so to overcome this, I reread parts of the MongoDB documentation and textbook to help grasp better how a create and read function would work.

**Testing Create function by creating “Frank”:**



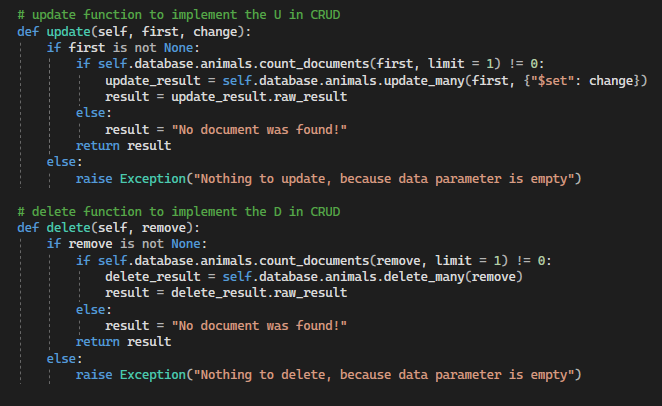
In Jupyter Notebook, I tested the create function with an animal already in the database after using the “find\_one()” function to find a random animal and it showed me “Frank.” So, I added my own new entry of “Frank” by changing up the “animal\_id” since they cannot have the same id in the AAC database. The “True” outcome at the bottom means that the function worked and added the entry.

**Testing Read function:**



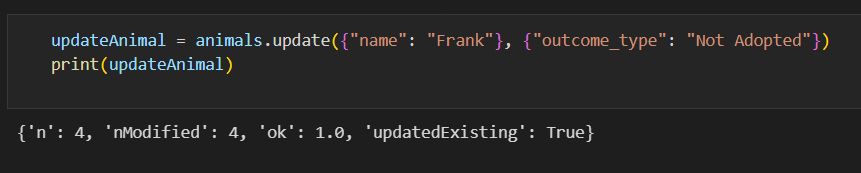
For the read function, I had the query find animals with the name “Frank” and down below is the result with every animal named “Frank.”

**Code Sample for Update and Delete functions and Tests:**



For the update function, this function changes a data entry depending on what the user chooses. The delete function removes an entry or entries completely.

**Testing the Update function and Changing Adopted Status to “Not Adopted”:**

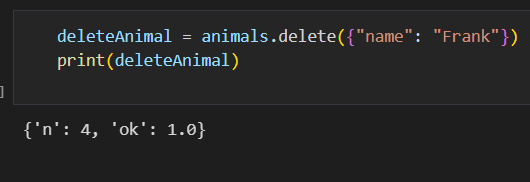


I tested the update function still with “Frank”, but I updated their “outcome\_types”, which is whether they were adopted or not to, to “Not Adopted.” So, each animal named “Frank” is now “Not Adopted.”

**After Update is Executed:**

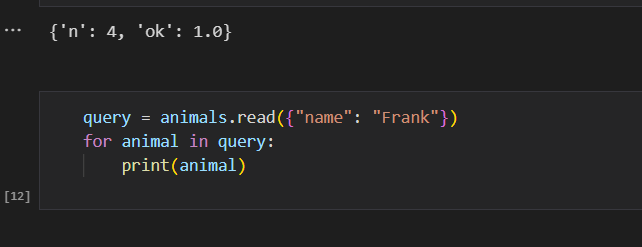
I used the read function to check if the animals named “Frank” had been updated from “Adopted” to “Not Adopted.” Which, as you can see, worked!

**Testing the Delete function and Removing any Animal with the Name “Frank”:**



Next is the delete function and this is where I deleted every entry with the name “Frank.”

**After Delete is Executed:**



When I used the read function again for “Frank”, we can see that there are now no animals named “Frank.” This is what we want since we need the delete function to remove entries.

**Dashboard Explanation:**

Grazioso Salvare requested a few things they wanted for their dashboard:

* Their personalized logo
* A unique identifier with my name (the creator’s name)
* Interactive filtering options such as buttons and drop-downs to be able to sort through Austin Animal Center’s dogs. These include filtering for “Water Rescue,” “Mountain or Wilderness Rescue,” Disaster Rescue or Individual Tracking,” and a “Reset” option.
* A data table that can change dynamically when filtering options are pressed.
* A geolocation map of the exact location of the animal and chart (I chose a pie graph) that shows the number of a certain breed there is.

**Screenshot of Dashboard and Logo:**

## A picture containing text Description automatically generated

**Screenshot of Pie Graph:**

**Chart, pie chart

Description automatically generated**

**Screenshot of Map:**

**Graphical user interface, map

Description automatically generated**

## Screenshot of Reset Option with Pie Graph and Map:

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

So, the “reset” option resets the page and clears any previous searches.

**Screenshot of Water Rescue Option with Pie Graph and Map:**

## Table Description automatically generated

Graphical user interface, application

Description automatically generated

The “water rescue” option will filter out any dogs that aren’t good for water rescue missions. The “Labrador Retriever Mix” was one of the breeds I typed in to filter out the dogs that would not perform well.

**Screenshot of Mountain/Wilderness Rescue with Pie Graph and Map:**

**Table

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

For the “mountain/wilderness” rescue, I typed “German” to see if it would filter out anything that wasn’t a “German Shepard”, and it did just that. But of course, other breeds that were good for these types of missions were kept in the results too.

**Screenshot of Disaster Rescue and Individual Tracking with Pie Graph and Map:**

**Table

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

Lastly, for the “disaster rescue and individual tracking” filter, I tested for “Doberman Pinchers” to see if the results fit, and they did.

## Contact

Alexis Indick