# CS 340 README Nick Franklin

## About the Project/Project Title

This project is about creating an interface to interact with a pet database using MongoDB. It is intended to be user friendly and encompass all the necessary and expected functions to interact with the pet database using the proper CRUD methods which are create, read, update, and delete. It will include a dashboard for viewing the animal information in a grid view. The user can select up to three animals and their positions will be marked on the map below. Beside the map a chart will display information on the breeds currently on display on the grid view.

## Motivation

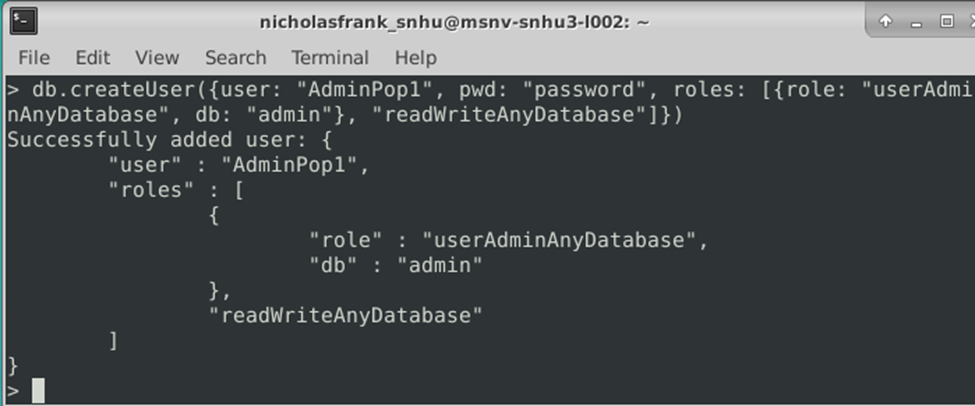
The motivation behind this project is to make an easy way to manage a pet database that is user friendly and encompasses all of the necessary tools. It is intended to practice and test my skills with databases and python.

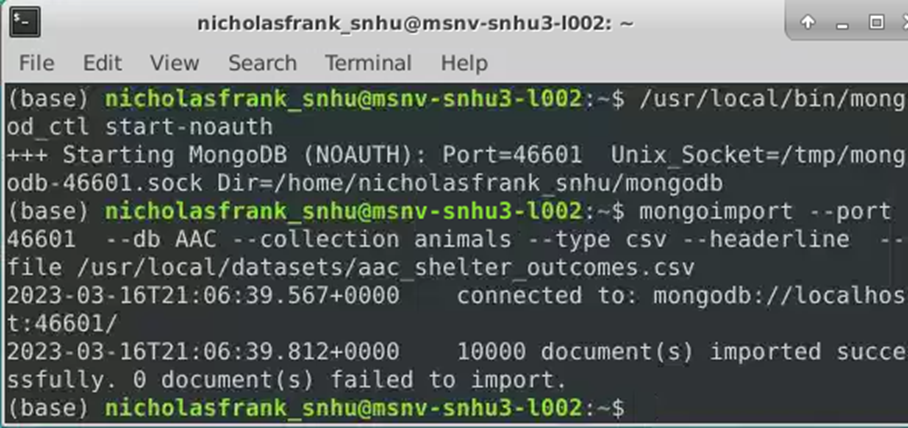
MongoDB was used for this program because it is quick and fairly intuitive to setup with a csv file and works well with Python. Python is compatible with other database tools such as SQL but the syntax is more of a headache and the CRUD operations are going to be more complex than they are when teaming MongoDB with Python.

Dash is a react JavaScript tool that uses html tags to control outputs in segments and then updates the specific ones from the callbacks from the Python module. Ultimately, it is because Dash is a dynamic and responsive framework that we chose to use it for this project.

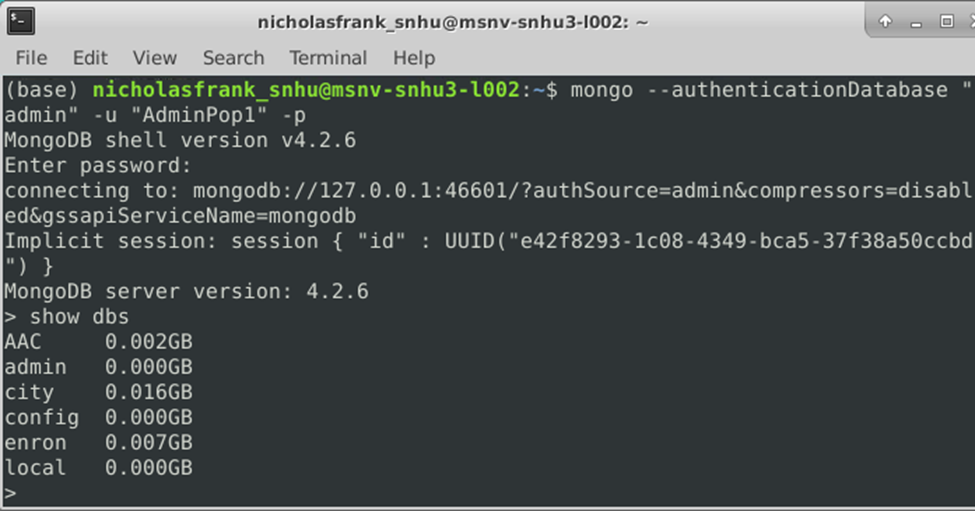
## Getting Started

To get started you need to first create a Mongo Database. You then need to create a username and password for the database. You can then use the python code I have supplied for the CRUD module to interact with the database. Lastly, use the Dash web app to view and search the data easily with a graph and map view. In order to easily get started, I have listed it in steps below:

1. Create a database using MongoDB called AAC.
2. Create an admin account and then a user that has read and write privileges for the AAC database.
3. Import the csv file ‘aac\_shelter\_outcomes.csv’ file.



1. Make sure you can login as admin or user with privileges.



1. Install Python and open up Jupyter Notebook.
2. Update the username, and user password in the ‘AnimalShelter.py’ code.

A picture containing text, screenshot, font

Description automatically generated

1. Type in the address for the dash and connect to the dashboard.

**Steps Taken to Complete the Project**

Beginning after the steps above such as creating an admin account for the database and uploading the csv file, I went about completing this project as follows:

1. Created the crud operations in Python.
2. Created a new Dash web application dashboard.
3. Created an app callback to get the data that would populate the data frame.
4. Created radial options for filtering the data in the desired ways.
5. Created a callback to update the map with the selected animals’ locations.
6. Created a chart that represents the number of animals per breed on the current page of the data.

## Installation

Tools you will need for this project include Python, Jupyter Notebook, and MongoDB. Plotly, Dash, and Pandas were also used. I have supplied links below for download.

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

**Jupyter Notebook**: <https://jupyter.org/install>

**MongoDB**: <https://www.mongodb.com/try/download/shell>

**Plotly:** Getting started guide - <https://plotly.com/python/getting-started/>

**Dash:** Installation guide - <https://dash.plotly.com/installation>

**Pandas:** Installation guide - <https://pandas.pydata.org/docs/getting_started/install.html>

## Usage

The application has been tested to perform all of the CRUD operations as it includes a create method, a read method, an update method, and a delete method as shown in the screenshots below:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The Dashboard can be used to display the database information in a spreadsheet sort of view. It can then be sorted based upon the different characteristics. The user may select multiple animals and up to three will have their location displayed on the map below the animal data sheet. Beside the map is a graph that displays the amount of animals per breed that are on the current page of the data sheet view above. The animals can also be filtered by selecting one of the four options above the data view. These were filters specified by Grazioso Salvare and include a filter to display dogs ideal for Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking. A fourth option is also present that simply resets the displayed data to have no filter.

Because of the page setup it is difficult to get everything included in one readable screenshot. So I will begin with three closeups that represent nearly the whole dashboard.

Closeup 1:

A red line drawing of a dog

Description automatically generated with medium confidence

Closeup 2:

A screenshot of a computer

Description automatically generated with medium confidence

Closeup 3:

A screenshot of a computer

Description automatically generated

Screenshot of when the ‘Water Rescue’ filter is selected:

A screenshot of a computer

Description automatically generated with medium confidence

Screenshot of when the Mountain/Wilderness Rescue filter is selected:

A screenshot of a computer

Description automatically generated with medium confidence

Screenshot of when the Disaster/Individual Tracking filter is selected:

A screenshot of a computer

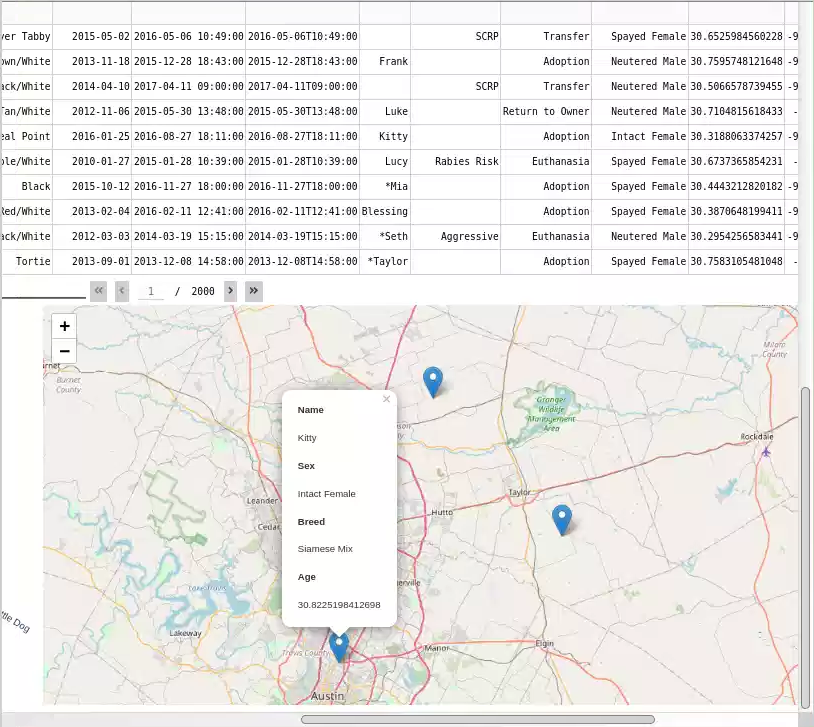
Description automatically generated with low confidence

Screenshot of when the ‘Reset’ filter is selected:

A screenshot of a computer

Description automatically generated with medium confidence

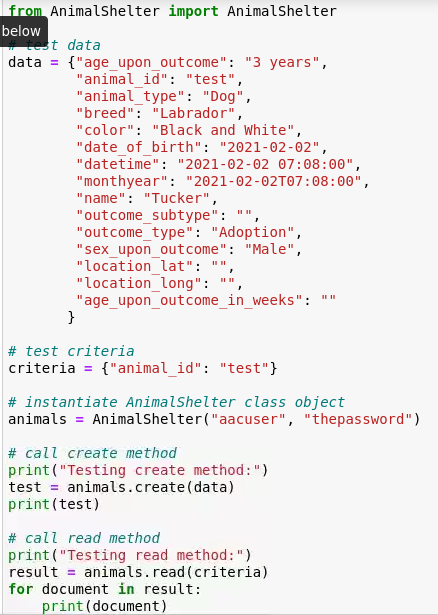
Screenshot of the map with three animals selected:



### Tests

The code can be tested using a script. I have set up a test for the create method that sends new pet info to the create method to add a new entry to the database. Another test provided is of the read method by passing criteria and testing to see whether it returns the correct results. I then added a test for the update method which edits the test entry we used for the create method. Next is a test for the delete method which deletes the test entry that we created originally to test the create method and updated.

### Screenshots

Code for testing the CRUD methods and the results:

Graphical user interface, text, application, email

Description automatically generated

## Contact

Nick franklin at nicholas.franklin@snhu.edu