# CS 340 README Template

## About the Project/Project Title

This project is about demonstrating and implementing the ability to manipulate a Mongo database from outside of the shell using Python. It has all of the CRUD functions, create, read, update, and delete. The CRUD functions determine how the user interacts with the Mongo database.

## Motivation

The reason for creating this project is to practice and learn how to use the Pymongo API in order to interact with a Mongo database.

**Why MongoDB?**

The reason for using MongoDB is because of the flexibility it offers as a non-relational database that uses storage similar to JSON. MongoDB allows for a more customizable storage than other databases like SQL do. MongoDB also has multiple API integrations that are utilized further in this project.

**Why Dash?**

The Dash framework was chosen for this app because of it’s ability to one, interact with Python and MognoDB, and two, because of its ability to present data in a format that is friendly, specifically from an analytical standpoint. Dash offers the ability to interact with plotly express which can be used to create graphs that best display the data trying to be conveyed.

## Getting Started

1. Install Mongo DB and create a local database.
2. Populate the database with data. (you can use the provided animal\_shelter\_outcomes.csv if you would like)
3. Create a user with read and write permissions to the database you imported the collection into.
4. Download and install the current version of Python if you haven’t already. <https://www.python.org/>
5. Download and install the current version of Pymongo if you haven’t already. <https://pymongo.readthedocs.io/en/stable/installation.html>
6. Use the following template to begin creating the CRUD portion of the application.  
   Graphical user interface, text, application

   Description automatically generated
7. After completing the code then test the code with Jupyter Notebook. <https://jupyter.org/install>
8. Next create the dashboard hmtl and css layout using the Dash web application
9. Connect data table callback to either radio buttons or a drop down menu, callback should use queries to match the data to be provided based on the filter option
10. Create a graph using plotly express, connect its callback to the data table. Extract data from the data table and pass that data to the plotly express graph object
11. Create a map and give it a callback. Connect the callback to the data table and it’s selected rows. Use the selected rows to populate the data needed to create map markers.

## Installation

Jupyter Notebook - <https://jupyter.org/install>  
Python - <https://www.python.org/>  
Pymongo - <https://pymongo.readthedocs.io/en/stable/installation.html>  
Mongo DB - <https://www.mongodb.com/try/download/shell>

Plotly - <https://dash.plotly.com/installation>

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

Pandas - <https://pandas.pydata.org/getting_started.html>

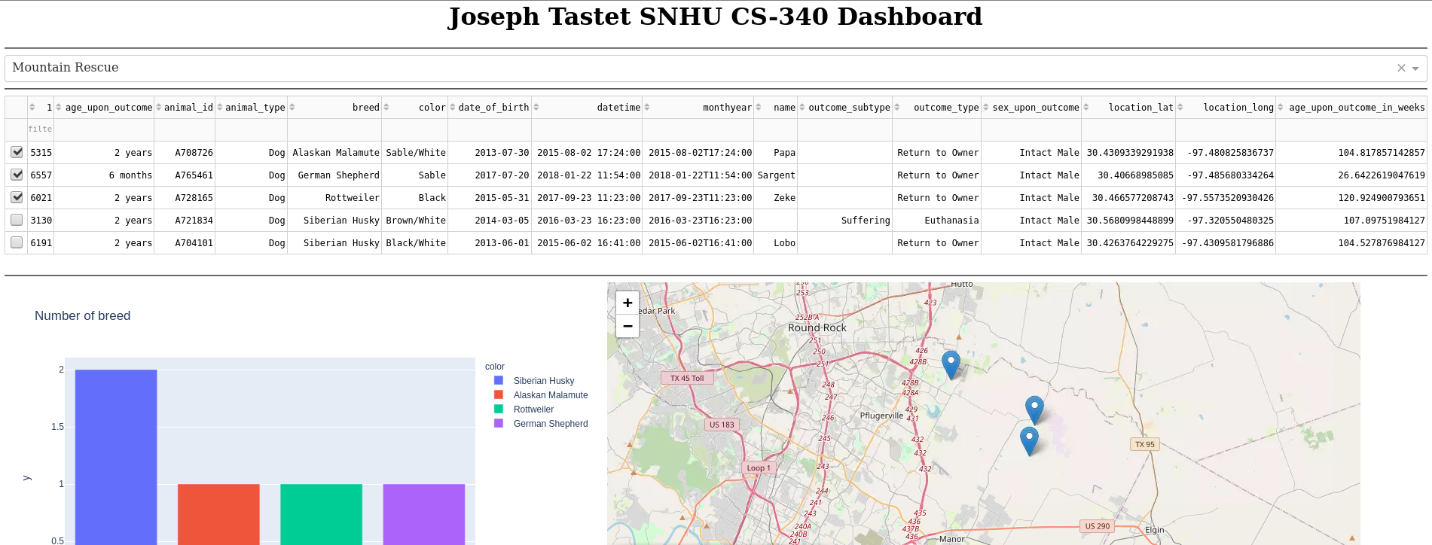
Any text editor (Visual Studio Code preferred) - <https://code.visualstudio.com/download>

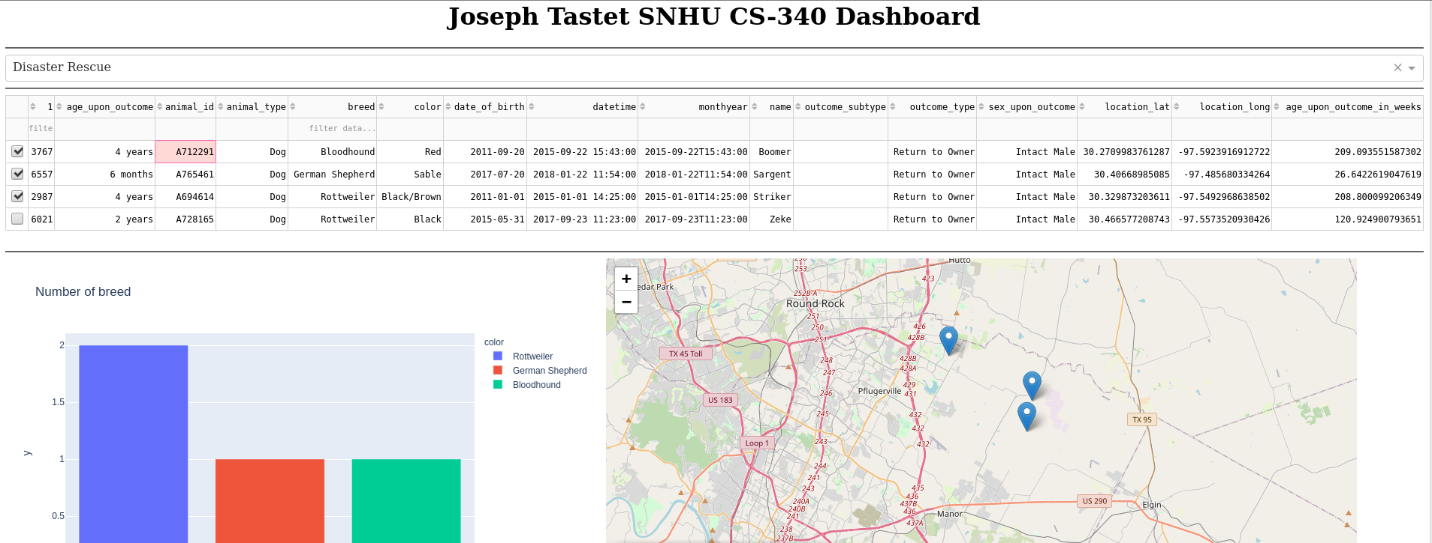
## Usage

This app is used in order to filter data for the Grazioso Salvare animal shelters. Users can sort the data based on 3 options from the dropdown menu including Water Rescue, Mountain Rescue, and Disaster Rescue. Users can also select rows in order to show those animals on the map. The app also shows the current selection’s breed information in the form of a bar graph. The data table, bar graph, and the map all update dynamically.

A picture containing calendar

Description automatically generated





A picture containing calendar

Description automatically generated

### Code Example

animalShelter = AnimalShelter(username, password)

animalShelter.create({data})

animalShelter.read({criteria})

HTML layout of the dashboard

*Text

Description automatically generated*

Dynamic map updating using a for loop to populate an array of map markers

Text

Description automatically generated

### Tests

Using the below snippet of code should create the “test” animal document in the database and return true.

print(animalShelter.create({

'age\_upon\_outcome': "2 years",

'animal\_id': "test",

'animal\_type': "Cat",

'breed': "Domestic Shorthair Mix",

'color': "Blue",

'date\_of\_birth': "2021-04-06",

'datetime': "2023-04-06 15:14:00",

'monthyear': "2023-04-06T15:14:00",

'name': "Test Cat",

'outcome\_subtype': "",

'outcome\_type': "Transfer",

'sex\_upon\_outcome': "Spayed Male",

'location\_lat': 30.5066578739455,

'location\_long': -97.3408780722188,

'age\_upon\_outcome\_in\_weeks': 156.767857142857

}))

Using the next line of code should return the animal record just created.

read = animalShelter.read({"name": "Test Cat"})

for animal in read:

print(animal)

### Screenshots

Importing the animal\_shelter\_outcomes.csv into Mongo

Graphical user interface, text

Description automatically generated

Creating “aacuser” for authentication to AAC db

Text

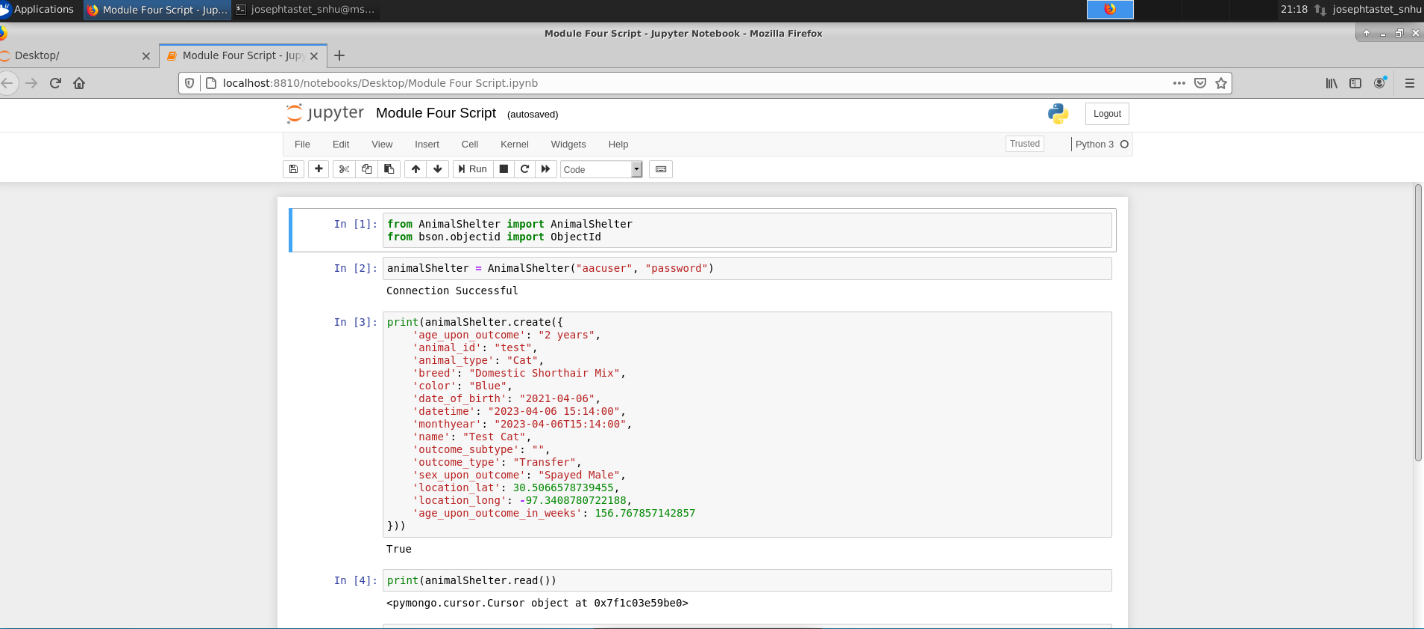
Description automatically generated

Confirming “aacuser” has permissions to the AAC db

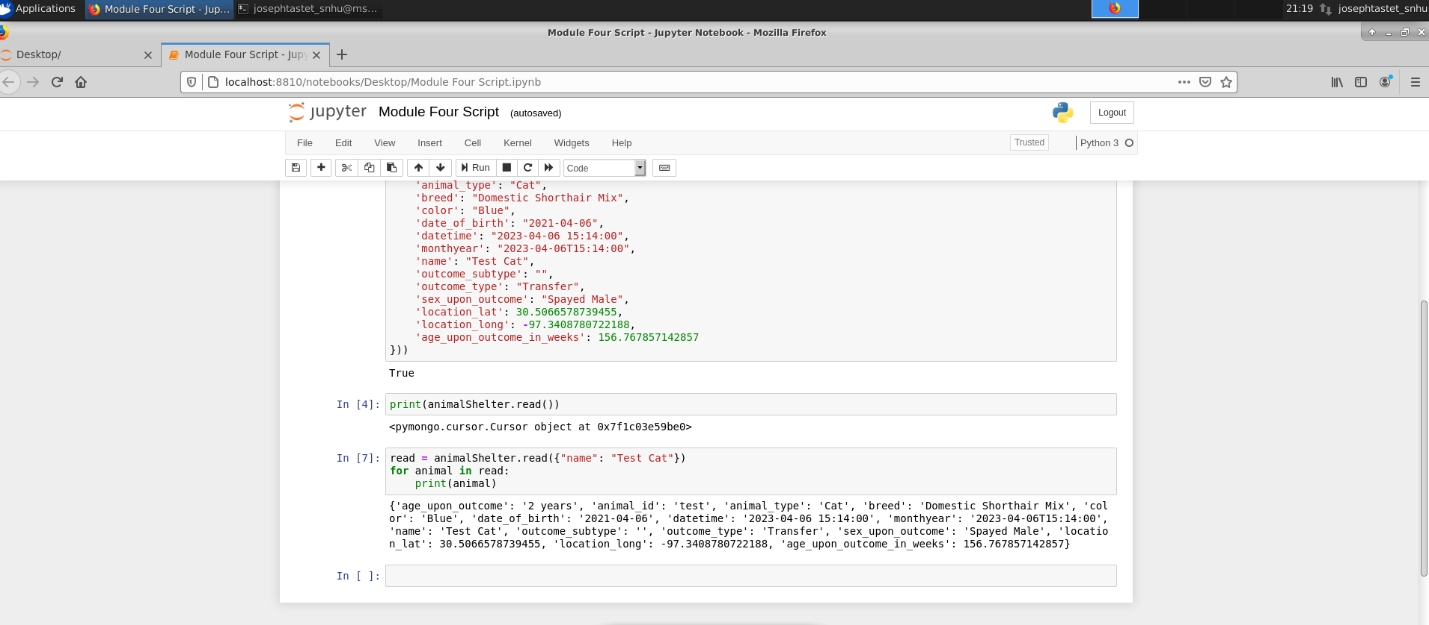
Text

Description automatically generated

In Cell 3, example of adding a record to a database. Returns the value True that it succeeded.

**

In cell 7 we can see that the record was created and read.



In cell 9 we update “Test Cat” name to “Test Cat 2”

Graphical user interface, text, application, email

Description automatically generated

In cell 11 and 12 we confirm it was updated, we then delete it in cell 15, and confirm it was deleted in cell 16

Graphical user interface, text, application, email

Description automatically generated

## Contact

Name: Joseph Tastet

Email: joseph.tastet@snhu.edu