# Project Two README – Nathan Anglin

## About the Project

This project incorporates Mongo DB with python to create an easier way to Create, Read, Update and Delete Items for a database. In this case, we are using an animal shelter’s database for animals. In this project, we are developing a web interface for Grazioso Salvare, which identifies dogs that are good candidates for search-and-rescue training.

## Purpose

In Project Two, our company, Global Rain, creates an interface to search through rescue animal databases to identify dogs that are good candidates for search-and-rescue training. Grazioso Salvare wishes to have a web interface designed to read through databases of animals in animal shelters to find dogs that are good candidates for search-and-rescue training. This interface should allow the company not only to view all animals available, but filter the results to the specific breeds, sex, and ages they are looking for different types of search-and-rescue training.

## Tools Used

In this application, I will be using MongoDB as our database. MongoDB is a good choice for this type of database manipulation, and we can upload the information given by the non-profit to populate our database with all available animals. Python and the Dash framework will be used to process this data and look through available animals in a way that is more intuitive and easier for the customer.

MongoDB is a database tool and is accessed through the command line. To use this tool but make the use of the final program more intuitive and easier to use, we are using the Dash framework, which allows us to write Python Code and display the results in a web page format. We are also using Jupyter Notebook to run this code, which can run both the core Python file, and the program file. This allows us to Read, Write, Update and Delete information held on the secure MongoDB server. This allows use and filtering of the data in a more useful way. The Dash framework then lets us display these results in HTML format, both in pure text and maps, tables, and graphs. In this implementation, we will have a datatable, map and pie chart showing the data, and a dropdown box to filter the results by the type of rescue dog. More filtering can be done by typing in a value above a row, this will narrow down the searches more. The company wishes for this project to be open source, so the steps will be provided on how to set up this environment on your computer.

## Installation

To run this program, you need to have MongoDB, Jupyter Notebook, and Python 3.6 installed. The following are the steps needed to set up the environment and run the application.

* **First, we need to import the datafile containing the animals available from local shelters. This is done by importing the csv file into MongoDB.**

Text

Description automatically generated

* **Next, we need to set up an admin user, and create a user named aacuser that will be our main login for this project.**

Text

Description automatically generated

Next, after getting the PythonCRUD.py and ProjectTwoDashboard.ipynb files provided, these are put into the same folder, and we can then access our program.

* **First open the terminal and start MongoDb. In my case, it was already running, but you will need to start this.**

Graphical user interface, text

Description automatically generated

* **Open Jupyter Notebook. This is where the PythonCRUD.py and ProjectTwoDashboard.ipynb files are.**

Graphical user interface, text, application

Description automatically generated

* **Open the ProjectTwoDashboard.ipynb file. This is our main program. At this point we can begin to explore the final program and its functionality.**

The home page shows the Grazioso Salvare Logo, followed by the author’s name. Below it is a dropdown box to select filters for searching results. In this first case, this is all animals in the system. You can see there is a datatable with the results, a pie chart showing the percentages of different animal breeds and a geolocation chart that shows the location of the animal. The datatable shows 10 animals at a time, and you can search left to right through the pages to view more matches.

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, table

Description automatically generatedMap

Description automatically generated with medium confidence

Using the next filter, Water Rescue, we filter the animals to only those dogs that are suitable for training using the requirements given to us by Grazioso Salvare. By choosing this option from our dropdown menu, you can now see these results reflected in the datatable, pie chart geolocation tables.

*Graphical user interface, text, application, table, email

Description automatically generated*

*Graphical user interface, application

Description automatically generated*

Using the next filter, Mountain / Wilderness, we again filter the animals to only those dogs that are suitable for training using the requirements. By choosing this option from our dropdown menu, you can again see these results reflected in the datatable, pie chart, and geolocation tables.

Graphical user interface, application, table

Description automatically generated

Graphical user interface, application

Description automatically generated

Finally, the filter for Disaster / Individual Tracking shows the results for this query. In this result, you can see that when you hover over an item in the pie chart, you get a tooltip giving you more information. This also applies to the map.

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

**Conclusion**

**Steps Taken to Complete the Project**

In this project, we went through several smaller steps to work up to the final project. First, we imported the data into MongoDB, then created custom index searches. These searches keep common searches available for faster processing of the data. After this, we created admin and user accounts.

Using this new user account, I programed a python file that could access the database without using the terminal window. If MongoDB was running, this allowed me to perform Create, Read, Update and Delete functions and use them outside the shell.

Once this was completed, I ran tests to make sure that everything was functioning properly, and that all functions can be verified working.

Finally, I put all this together by writing code in Python and using the Dash Framework to both display the data in an interesting and intuitive way, and to be able to search through the results to find what the customer needed. I used the criteria given by the customer to set up custom filters to the breeds, age, and other statistics to find dogs suitable for each type of training.

**Challenges encountered**

Along the way, I ran into several hurdles. The first was that it was hard to get a grasp on the callback functions used in the Dash framework. My Python code knowledge was limited, but after figuring out how these worked, I was able to start displaying data.

Finding bugs in the code was very difficult, because within a callback function, if the code is broken and has errors, none of these are displayed when the code is run. The result is that you just get a blank page. I was able to pull pieces of code out of the callback function to the top of the code and run it there. At this point I got errors and could work with the code. This was probably my biggest ah-ha moment, as I could move forward and test the code in smaller pieces.

In the end I still had one issue with Jupyter Notebook, where my program would display correctly, but when I opened it in a new tab, the pie chart wouldn’t show up and the map would only show up after the filters were updated. I still am not sure why this is happening, but the code works in the preview window, and I think this may just be an issue with Jupyter Notebook.

Overall, I am very proud of finishing this assignment, it was one of the hardest I have had so far in a class, and I am proud I got it working.

**Contact**

Nathan Anglin

CS-340

**nathan.anglin@snhu.edu**