**Grazioso Salvare**

## About Grazioso Salvare Dashboard

The Grazioso Salvare Dashboard uses python and needs to be able to Create new entries into a Mongo database for the AAC database, Read and access files based on search criteria, Updates files for each of the animals in the database as needed, and Delete files when we do not need them anymore. The program should be able to pull up a table with all the animals in the database and be able to choose certain types of rescue dogs to look at. There will be a pie chart to show how many of each breed is contained in the table and the geolocation map included in the dashboard.

## Motivation

We need a way to easily use a Mongo database without the knowledge of getting to it. When a program is created that uses a database, it is user friendly, so people do not have the need to learn the “behind-the-scenes.” This will also allow the user to easily find a certain type of the dog they need for rescue missions.

## Getting Started

The Mongo database needs to have a user added to the list of users, which I created for this program. This screenshot shows the user we are using have the ability to view the correct database the Pymongo needs to run.

A screenshot of a computer program

Description automatically generated

The Mongo username and password needs to be entered into the terminal before you open mongosh. The system will still pull the correct host and port that it was using beforehand. Once you have your username and password set up, you will have access to read and write in the ACC database. The Python program has a mongo user, password, host and port already programed in for easy access.

The Pymongo coding needs to have the database user information added into the code. We called the user information “self” which contains the username, password, host, and the port for easy access to the database. “Self” is added to all the other defined functions to use. The function Create imports new animals, once the insert portion of the function is complete, the function will call for the Object ID to confirm the import was successful. The Reading function will look for any queries that are needed in the database and list the animal’s information once the function is completed. Updating a file for an animal will need the animal identification number and the column with the updated information to continue; once updated, the function will call that animal identification number in a query to ensure that the update was successful. Deleting an animal identification number will delete the whole animal out of the system. To confirm that it is done, we will call that number to ensure it fails to print.

When it comes to using Dash in the Juypter Notebook to create data tables, everything relies on the Pymongo code that connects to the MongoDB. If it does not correctly connect, it will not work. My program still has issues with connecting however I think I have the main idea of it.

A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

After it is connected, you need to set up the layout of the dashboard; including the titles/headlines, radio buttons, dropdown lists, and the table with all the bells and whistles you wish to include. Depending on your setup, you can also include code to have different graphs next to each other or above and below as well. Once that is done, you need to add functionality to your “bells and whistles”, in my case just the radio buttons. I called for the database to pull up specific dogs for each of the rescue groups.

A screen shot of a computer code

Description automatically generated

When the functionality is completed, that is when you set up your pie chart and the geolocation map and you are finished.

My biggest challenge is testing portions. I am not familiar with Juypter and I am trying to learn how it works. I have not yet overcome this challenge.

## Installation

I used Spyder Version 5 with Python 3.9 to create the program. <https://www.spyder-ide.org/>

I used Jupyter Notebook App with extension configured to 0.6.1 for testing. <https://jupyter.org/>

To set up the Mongo Database, I just used a terminal Windows app store. (You can download a terminal app on windows.)

## Usage

The way it works is when you have another class accepting input, this class would pull the input and connect to the database. Each section would have a specific task to complete.

### Code Example

Simple code is a better way to go when you can.

These show each part of the code is used for the program.

To gain access to the Mongo Database

A screen shot of a computer code

Description automatically generated

Create a new string in the database.A screen shot of a computer code

Description automatically generated

Read the database looking for specific animals to match the criteria.

A screen shot of a computer code

Description automatically generated

To Update an attribute for a specific animal.

A screen shot of a computer code

Description automatically generated

To delete an animal out of the database using the animal id.

A screen shot of a computer code

Description automatically generated

Layout set up start.

A close-up of a computer code

Description automatically generated

Radio Button setup



### Tests

To use Jupyter Notebook, you should be able to import the python program as a library in the app and add testing lines into it. As long as the python program is uploaded to the Jupyter filesystem, type “from fileName import className”

However, my tests are not successful due to unknown reasons. I do not believe my program is being read correctly. I have very little experience with Jupyter.

## Contact

Samantha Lasseigne

Samantha.Lasseigne@SNHU.edu