# CS 340 – 7-2 Final Project

## About the Project/Project Title

Global Rain’s Animal Shelter Organizer project is designed to provide companies like our client, Grazioso Salvare, to streamline their search for eligible canine search-and-rescue candidates. It sources data from local animal shelters that are tasked with keeping track of the animals they are currently housing. This project, complete with a detailed data table, geolocation chart, and pie chart, provides authorized clients with a simplified and user-friendly method for scouting out their new trainees.

## Motivation

Here at Global Rain, we know that finding the right canine candidates for search-and-rescue training can be a meticulous task. This project serves to streamline this process so that trainers can focus on training. Global Rain’s Animal Shelter Organizer provides local animal shelter information using a dynamic data table, geolocation chart, and pie chart. Companies like our client, Grazioso Salvare, will no longer have to manually comb through large datasets to find eligible candidates for their training services.

## Getting Started

To start using the Animal Shelter Organizer, you must sign up at our website, purchase access to the Global Rain premium suite, and provide us with the necessary client information, such as your location.

## Installation

To begin using the Animal Shelter Organizer project, simply navigate to the Global Rain website, login with your credentials, and head to the Animal Shelter Organizer page.

## Usage

### Code Example

The backend of the Animal Shelter Organizer project includes a Python class named AnimalShelter that stores the necessary attributes and methods to connect to the appropriate MongoDB database and collection using an authorized username and password, provides a method to insert provided information for a newly admitted animal to the database in the appropriate fields, a separate method to retrieve the information for an existing animal’s document, a third method to update the field of an existing animal’s document, and a fourth method to delete an existing animal's document.

# Necessary imports

class AnimalShelter(object):

def \_\_init\_\_(self):

# Username, password, host, port, database, and collection

# information used to connect to the appropriate MongoDB

# database and collection.

def create(self, data):

# If some data was provided:

try:

self.collection.insert\_one(data) # data should be  
 # dictionary

return True #The data insert was successful

except Exception as e:

print(f"Failed to insert data: {e}")

return False # The data insert was not successful

# If no data was provided:

raise Exception("Nothing to save, because data

parameter is empty")

While writing the create() method, the main challenge I encountered mainly had to do with user authorization. At first, I spent some time troubleshooting how to properly include try/except blocks in this method as well as how to properly utilize the self keyword. While testing this method, I spent significantly more time troubleshooting the module’s access to the database collection, as it turned out I had given the user authorization for a non-existent database. In this way, I learned that the database collection title is case-sensitive.

def read(self, lookup\_dictionary):

try:

my\_key\_value\_pair = {key: value}

result = list(self.collection.find(lookup\_dictionary))

return result # Print the result if the read command is successful

except Exception as e:

print(f"Failed to read data: {e}")

return [] # Return an empty list if the read command is unsuccessful

After my learning experience while writing the create() method, my process to write the read() method was much more straightforward. It took much less time to troubleshoot this try/except block and to find the best feed the provided key:value pair to the find() method.

def update(self, lookup\_data, update\_key, update\_value):

try:

result = self.collection.update\_many(lookup\_data, {"$set": { update\_key : update\_value }})

return result.modified\_count # Return the number of modified objects if successful

except Exception as e:

print(f"Failed to read data: {e}")

The main challenge I encountered while writing the update() method was working with Jupyter Notebook. Occasionally the test script would not recognize that changes had been made to the AnimalShelter.py module and vice versa, so I ended up having to force stop and restart the Jupyter Notebook process often to have any saved changes work as intended.

def delete(self, lookup\_key, lookup\_value):

try:

result = self.collection.delete\_many({ lookup\_key : lookup\_value })

return result.deleted\_count # Return the number of deleted objects if successful

except Exception as e:

print(f("Failed to delete data with lookup key/value pair: " + lookup\_key + " : " +

lookup\_value + "."))

Once I learned what I did while writing the update() method, writing the delete() method went very smoothly.

### Tests

Here is an example of what an animal shelter attendant can write in the test file to create a document for a newly admitted animal. The terminal will print “True” if the insert of the new data was successful, and “False” if the insert failed. Note that the new data must be entered in dictionary format, with the appropriate column titles, as seen below:

from AnimalShelter import AnimalShelter

my\_shelter = AnimalShelter()

data = {

"rec\_num": "99999",

"age\_upon\_outcome": "1 year",

"animal\_id": "A991234",

"animal\_type": "Cat",

"breed": "Russian Blue",

"color": "Gray",

"date\_of\_birth": "2009-02-25",

"datetime": "2010-02-25",

"monthyear": "2010-02-25T14:00:00",

"name": "Frankie",

"outcome\_subtype": "SCRP",

"outcome\_type": "Transfer",

"sex\_upon\_outcome": "Neutered Male",

"location\_lat": 41.909500,

"location\_long": -72.470802,

"age\_upon\_outcome\_in\_weeks": 52

}

insert\_result = my\_shelter.create(data)

print(insert\_result)

Here is an example of an animal shelter attendant pulling up the document of an existing animal at the shelter:

from AnimalShelter import AnimalShelter

my\_shelter = AnimalShelter()

result = my\_shelter.read("rec\_num", "99999")

print(result)

This example shows an animal shelter attendant updating the existing animal’s document that was created above:

from AnimalShelter import AnimalShelter

my\_shelter = AnimalShelter("aacuser", "Auser607")

my\_lookup\_pair = {"rec\_num" : "99999"}

my\_update\_key = "datetime"

my\_update\_value = "2023-11-29"

my\_shelter.update(my\_lookup\_pair, my\_update\_key, my\_update\_value)

Finally, here is an example of an animal shelter attendant deleting an existing animal’s document from the database:

from AnimalShelter import AnimalShelter

my\_shelter = AnimalShelter("aacuser", "Auser607")

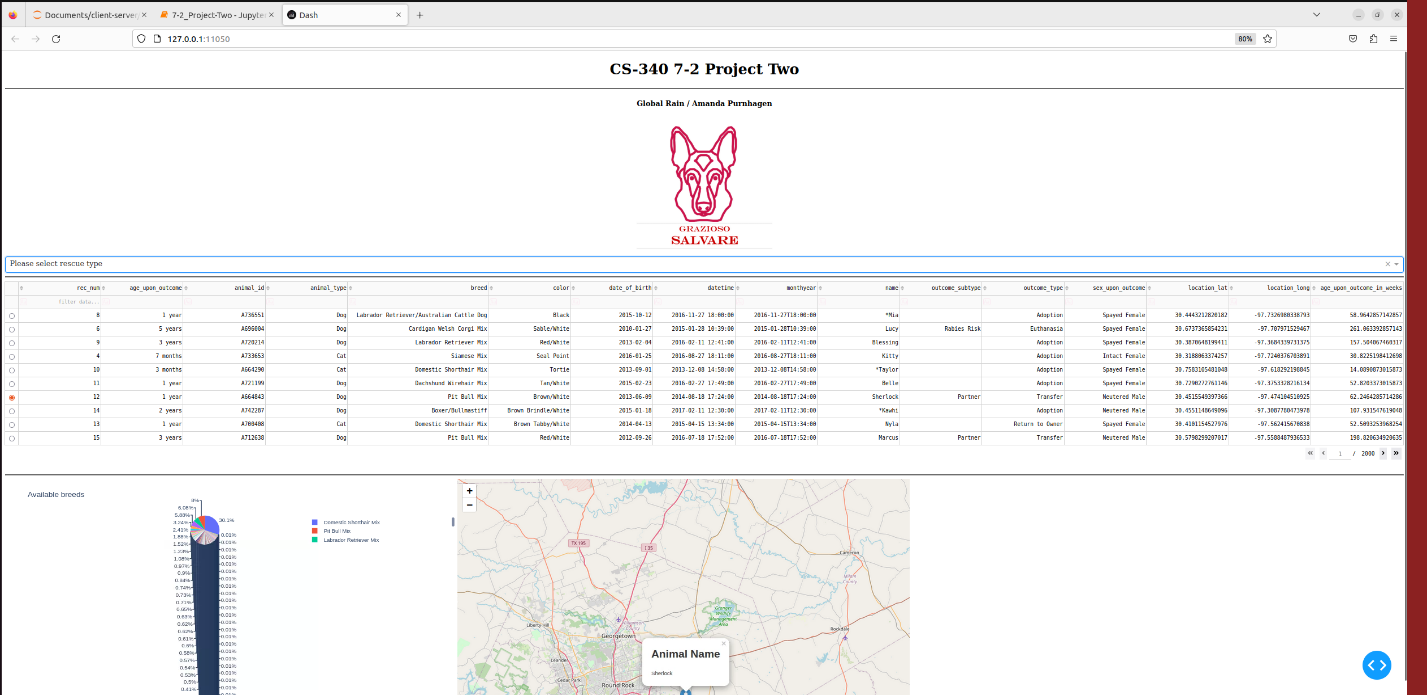
my\_lookup\_key = "name"

my\_lookup\_value = "Frankie"

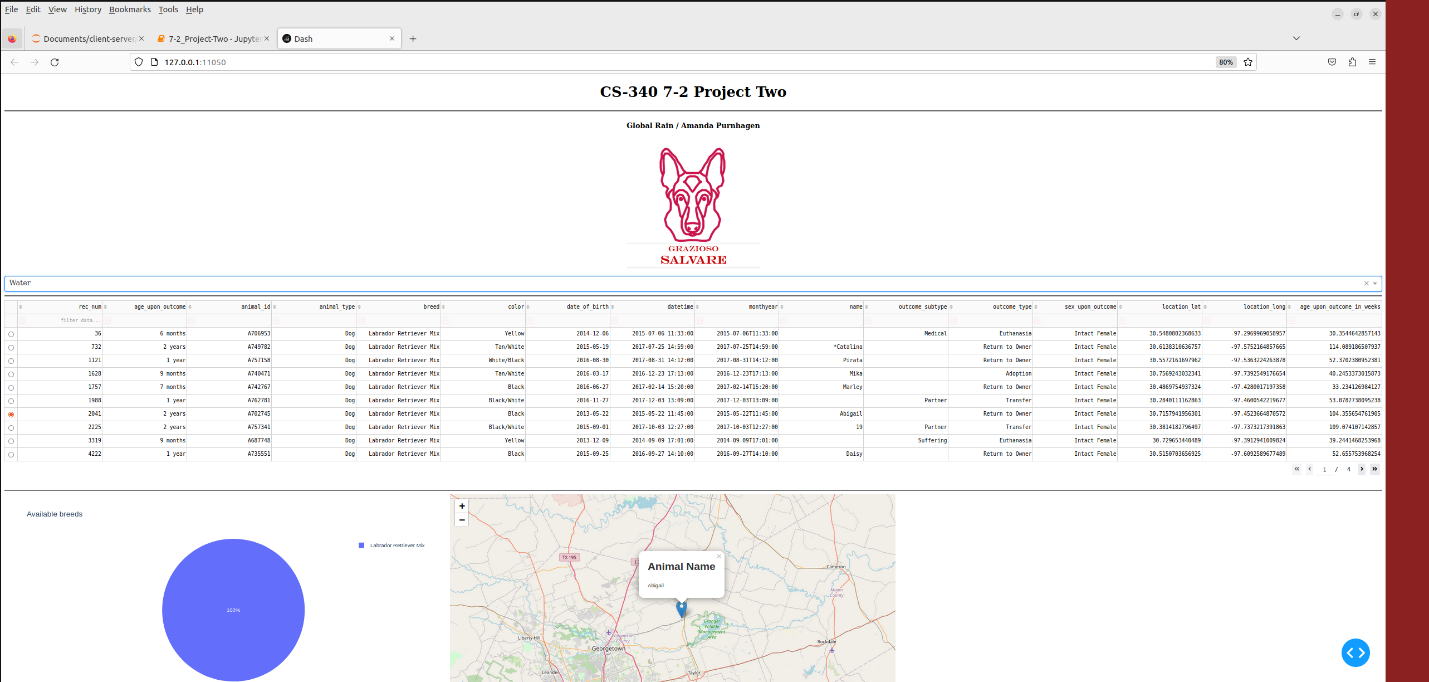
my\_shelter.delete(my\_lookup\_key, my\_lookup\_value)

### Screenshots

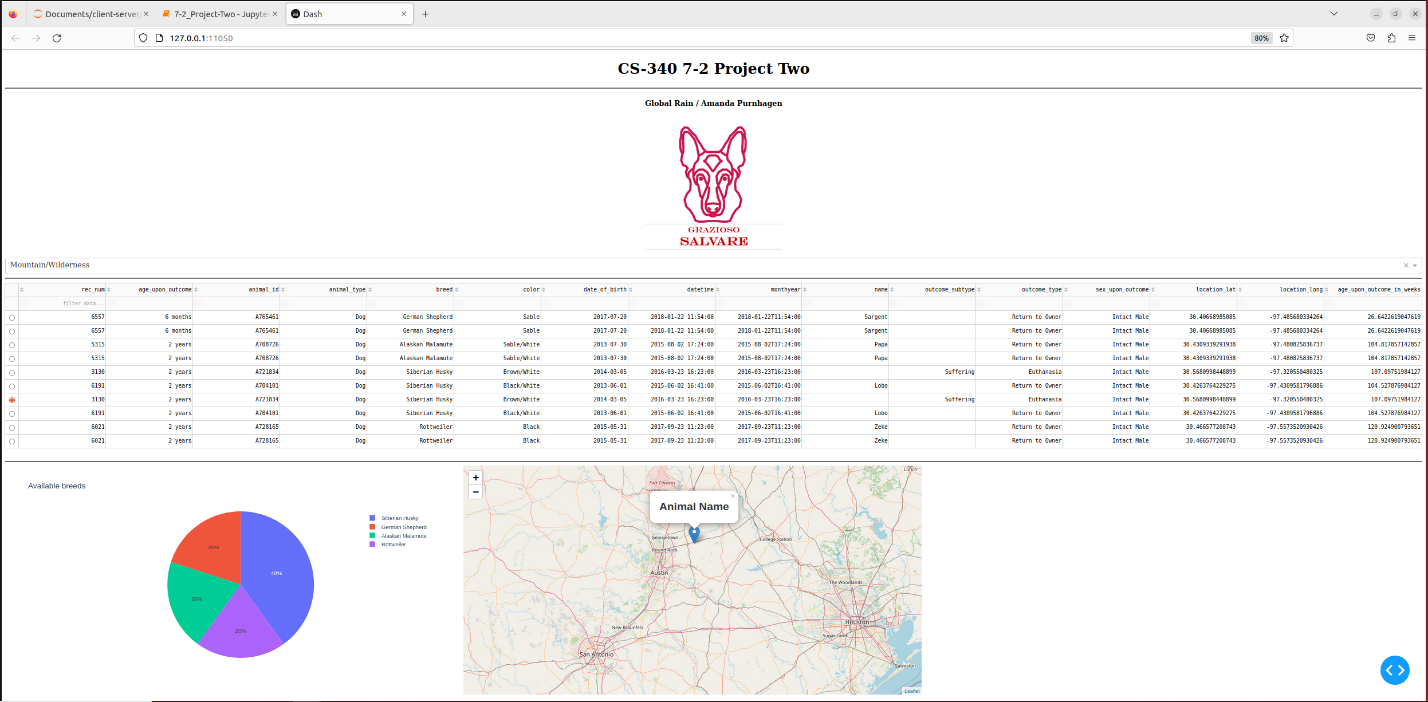
Starting state of dashboard:



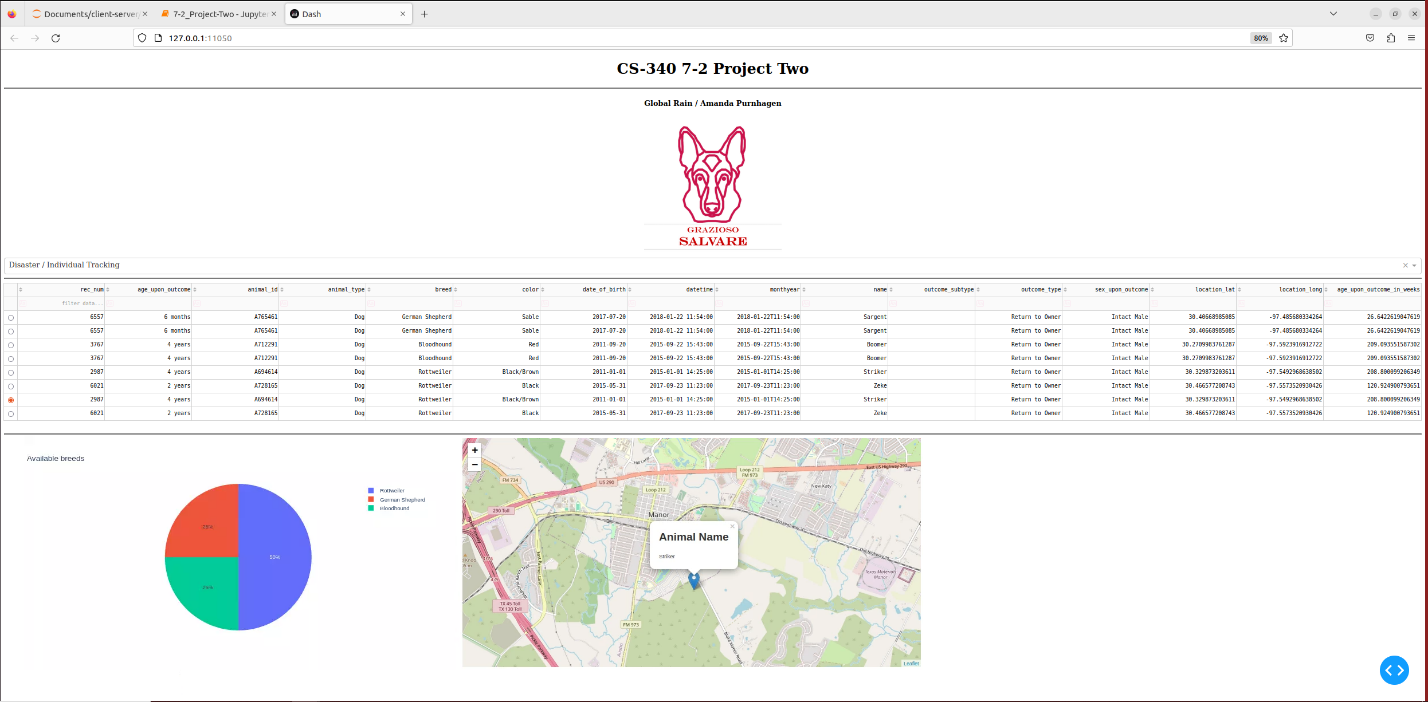
Dashboard after the **water rescue** filter has been applied:



Dashboard after the **mountain or wilderness rescue** filter has been applied:



Dashboard after the **disaster or individual tracking rescue** filter has been applied:



To see that the widgets return to their original, unfiltered state when the filter is removed, please refer to the provided screencast.

### Tools Used and Rationale

### The tools used in this project include MongoDB, Python, and the Dash framework.

In terms of database management systems, MongoDB is particularly equipped to handle projects like the Animal Shelter Organizer project. It is optimized to structure data for deployment through web apps or software due to its compatibility with many programming languages (Giamas, 2022, p. 11). MongoDB supports various types of queries and indexing, allowing developers to have more freedom when optimizing search capabilities (p. 11). It is schema-less, meaning the developer is not limited by as many strict rules regarding how data may be stored and retrieved (p. 11). MongoDB is built with server uptime, load balancing, horizontal (and vertical) scaling, and security in mind (Giamas, 2022, p. 12), creating a service that is both powerful and reliable.

Additionally, MongoDB is prepared to interface well with many programming languages. This includes the language the Animal Shelter Organizer project is written in, Python. In this case, Python acts as the “glue” that adheres the server side, MongoDB, to the client side, the Dash framework. Developers can create methods to manipulate data in a MongoDB database with ease using the Python driver. This includes the methods mentioned above in the Code Example and Tests sections. It allows for quick and effective troubleshooting since Python has simple syntax while being quite powerful.

The Dash framework is what the client sees. It provides filtering options, data tables, and visually appealing graphical representations of data. This particular Dash framework uses pandas, the Python data analysis library, to process the data used for visualization.

I encountered some challenges while creating the Animal Shelter Organizer project, including how to properly organize code within a project in Python that utilizes MongoDB and the Dash framework. It took me a while to wrap my head around how to feed the appropriate data to the various Dash widgets and then manipulate that data to appropriately display the data in a responsive manner. Creating the appropriate queries for the rescue types also proved to be challenging. In the end, it was properly formatting these queries with white space that actually helped me have clarity as to what was happening. I would have liked to further customize the layout of the web app, but I am pleased with the responsiveness and accuracy of the widget data. It was very satisfying to see all of these concepts coalesce into this final project.

## Contact

Amanda Purnhagen

amanda.purnhagen@snhu.edu

**References**

Giamas, A. (2022). *Mastering mongodb 6.X: Expert techniques to run high-volume and fault-tolerant database solutions using mongodb 6.X*. Packt Publishing.