**About the Project**

The primary goal of this project is to develop a full-stack application, including a MongoDB database and a client-facing web application dashboard. As the lead developer, I am tasked with creating a comprehensive software application to assist Grazioso Salvare in identifying and categorizing dogs for search-and-rescue training. This application will allow Grazioso Salvare to:

* Identify dogs suitable for search-and-rescue training based on specified criteria.
* Categorize dogs based on age, breed, and proficiency in various rescue scenarios.
* Access and manage data efficiently through a user-friendly web dashboard.

**Motivation**

This project aims to showcase proficiency in database management and manipulation of data using Python. By focusing on CRUD functionalities, the module provides a foundation for handling animal data within a shelter environment.

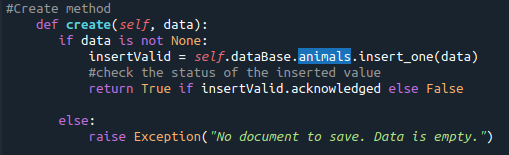
Grazioso Salvare is an international rescue-animal training company specializing in training dogs for search-and-rescue operations. These highly trained dogs play a crucial role in locating and rescuing humans and animals in challenging and life-threatening conditions.

To source potential candidates for training, Grazioso Salvare has partnered with a nonprofit agency operating five animal shelters in the Austin, Texas region. The nonprofit agency will provide data from their shelters to assist in identifying dogs with suitable profiles for search-and-rescue training.

**Code Example**

To add animals to the database, use the following code snippet, called through the accompanying class:

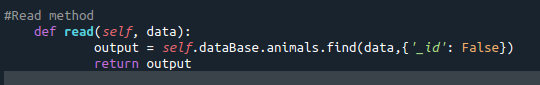
print(animals.create("STRING\_TYPE"))



This will add animals, and the program will return a boolean indicating success or an error if the addition fails.

To retrieve information about an animal, use the following query:

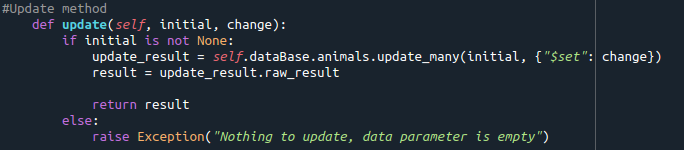
query = animals.read({"name": "NAME"})



Adjust the query parameters as needed for your specific use case.

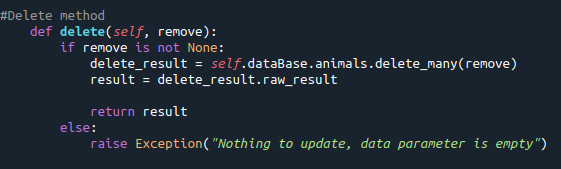
To update information about an animal, use the following snippet:

query = animals.update({"name": "NAME"})



To delete information about an animal, use the following snippet:

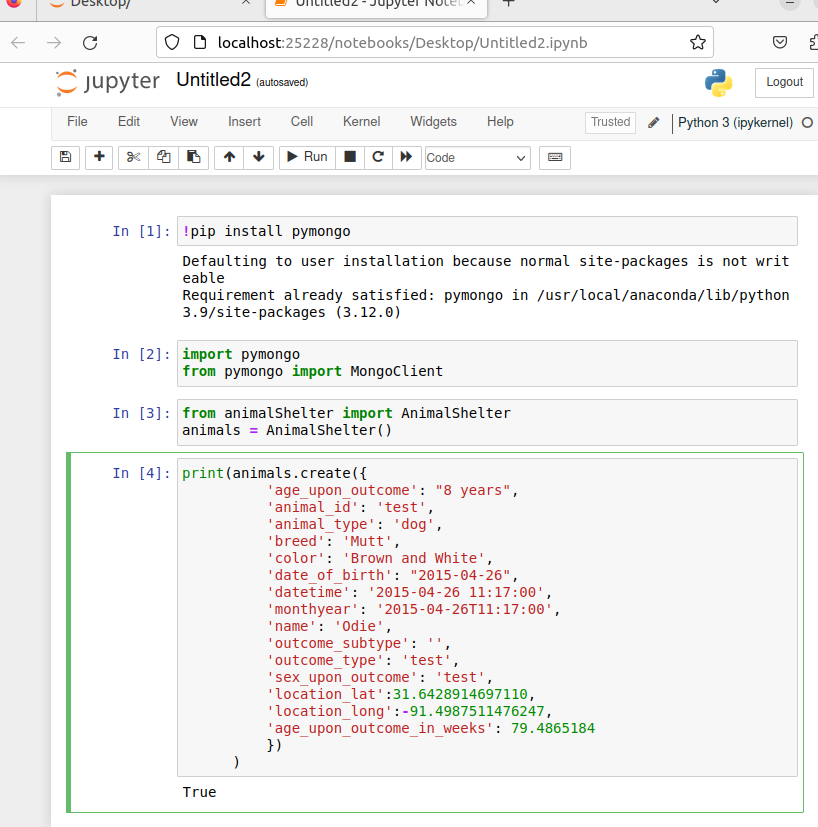
query = animals.delete({"name": "NAME"})

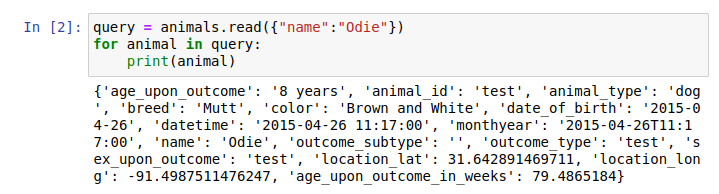


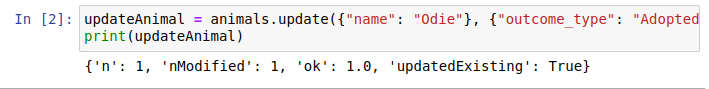
**Screenshots**

Include screenshots showcasing the module in action, demonstrating the creation and reading of animal records.

The following screenshots demonstrate the accompanying test script’s functionality.



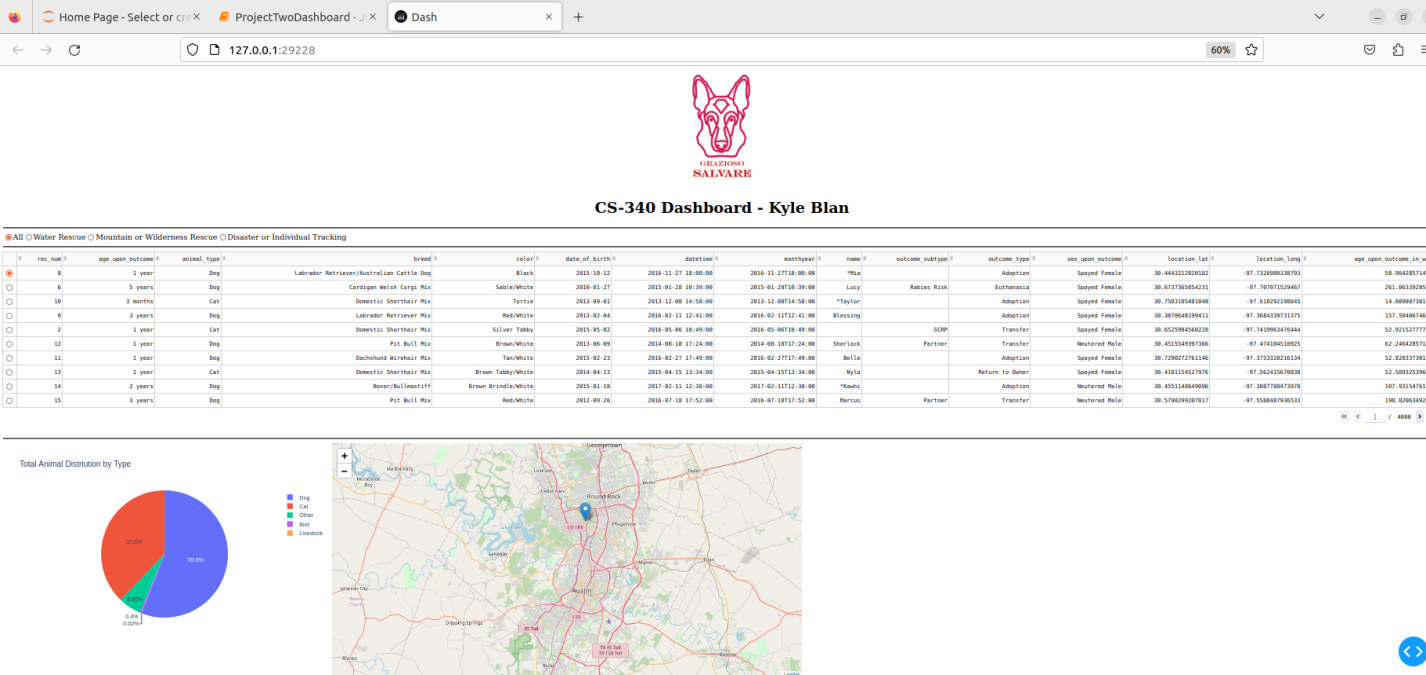




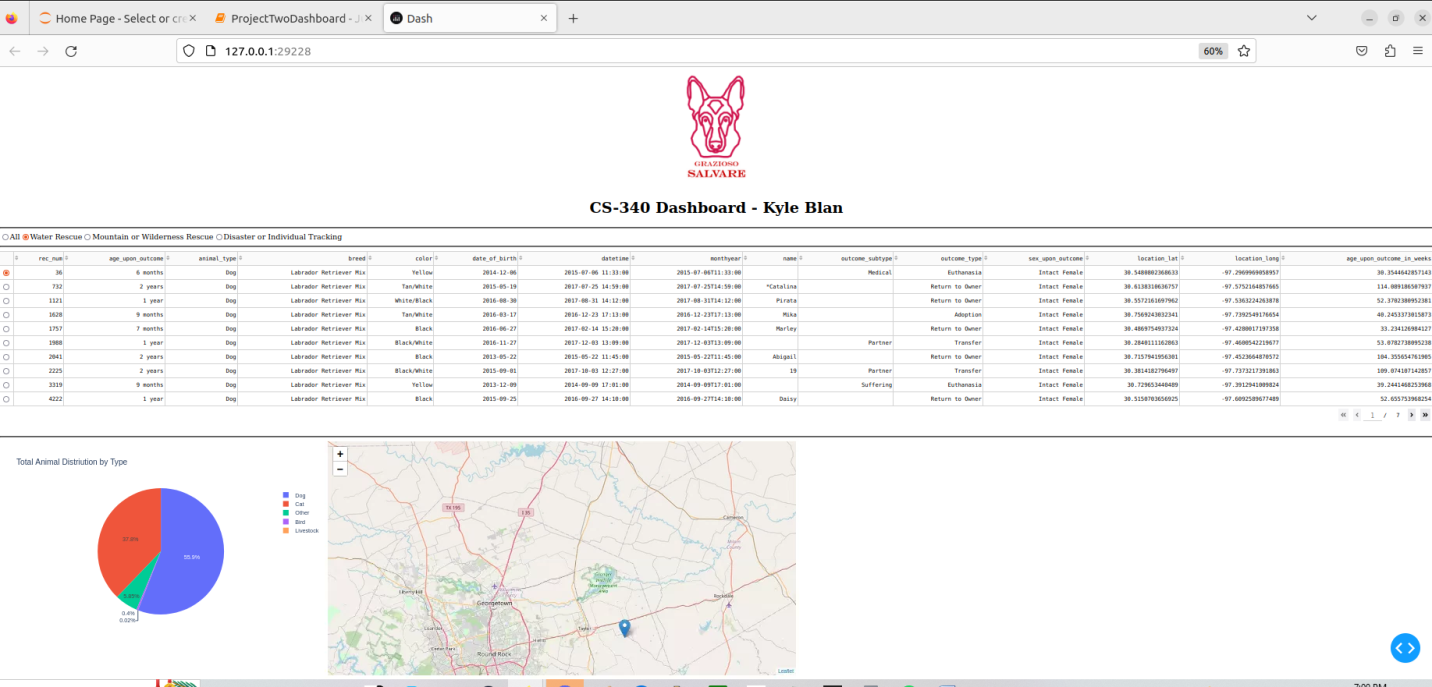


Continued, screenshots that show the Dashboard Functionality:

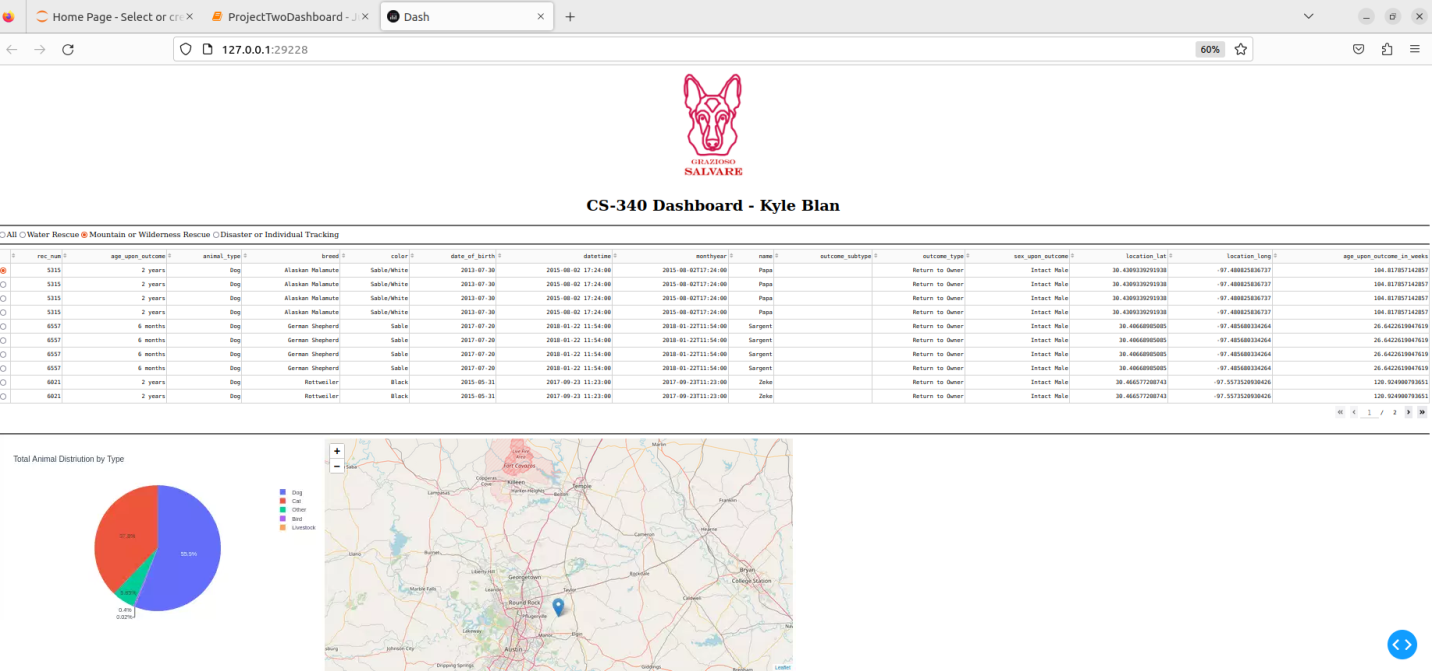
Default settings:



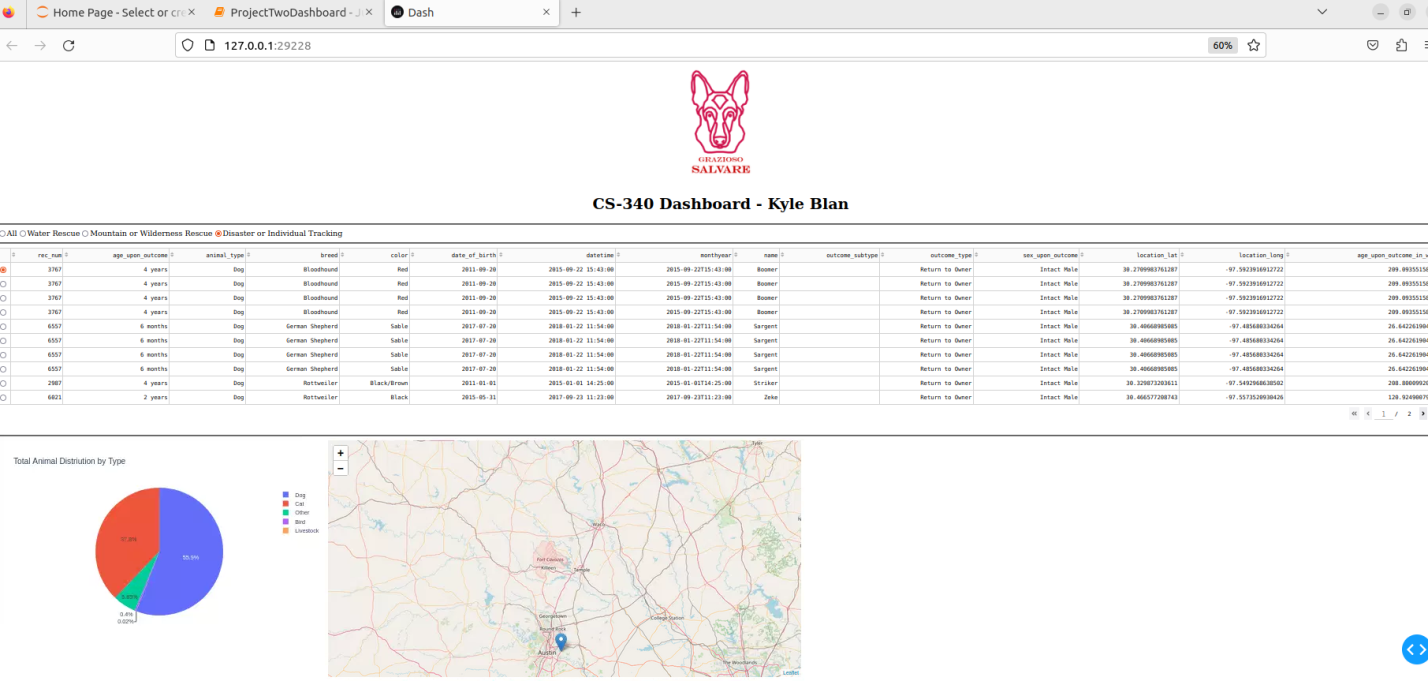
Filtered by Water Rescue selection:



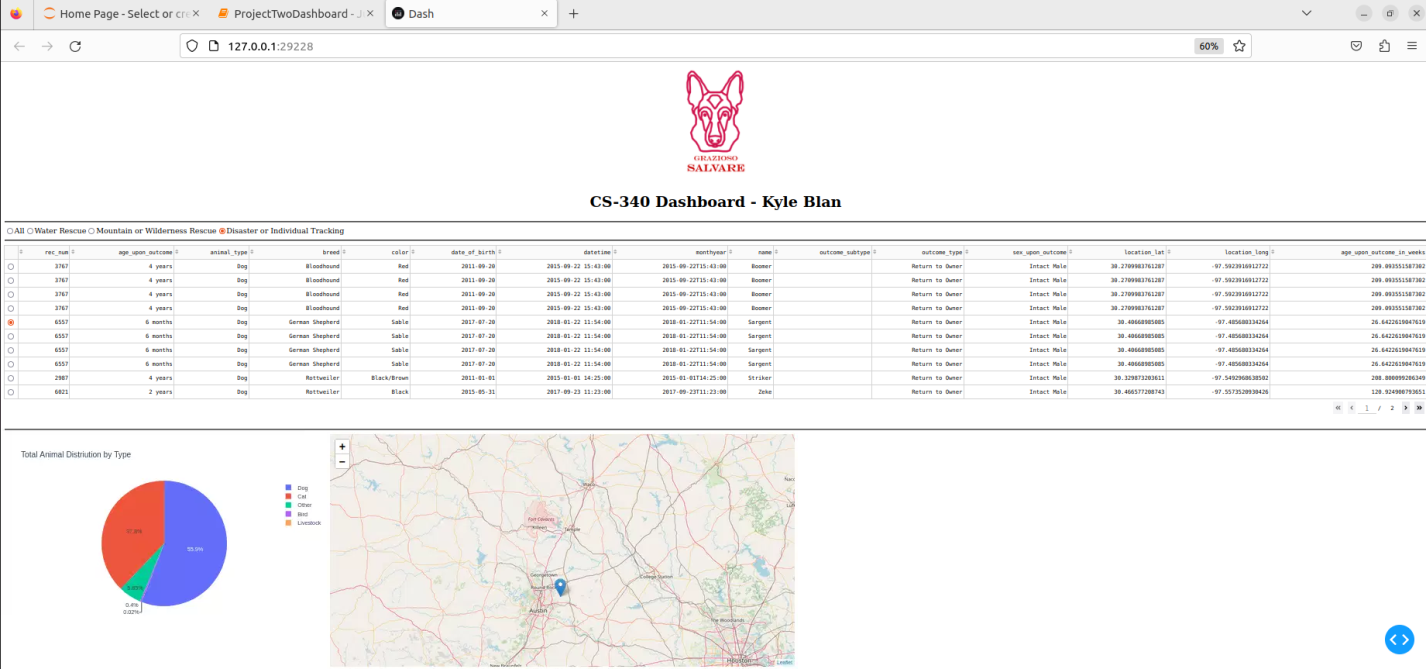
Filtered by Mountain or Wilderness Rescue selection:



Filtered by Disaster or Individual Tracking selection:



Note, the row selected will update the location specified on the corresponding map, with relation to the selected animal:



**Installation**

Ensure you have the following tools installed:

* Python (current version)
* MongoDB (to access the database)
* Jupyter Dash
* Plotly

**Getting Started**

The CRUD Python module is designed to facilitate basic Create, Read, Update, and Delete operations on a MongoDB database. The module serves as a toolkit for interacting with a database of animals, allowing users to manage and retrieve information about these animals efficiently.

The dashboard can be launched from the Jupyter script provided.

**Step 1: Import Your Dataset Using mongoimport**

1. Open a Terminal: Open a terminal on your computer.
2. Navigate to Dataset Directory: Use the cd command to go to the directory where your dataset file is located.
3. Run mongoimport Command: Execute the following mongoimport command to import your dataset into MongoDB:

mongoimport --db your\_database --collection your\_collection --file your\_dataset.json

1. Replace your\_database with your MongoDB database name, your\_collection with the collection name, and your\_dataset.json with your dataset file.

**Step 2: Manage User Permissions**

1. Access MongoDB Shell: Connect to your MongoDB server using the MongoDB shell.
2. Switch to Relevant Database: Switch to the database where you imported your dataset.
3. Create User and Assign Permissions: Create a user with appropriate permissions. For example:

db.createUser({

user: "your\_user",

pwd: "your\_password",

roles: ["readWrite", "dbAdmin"]

})

1. Adjust roles based on your specific requirements.

**Step 3: Verify User Authentication**

1. Access MongoDB Shell: Connect to your MongoDB server using the MongoDB shell.
2. Authenticate User: Authenticate the user you created in Step 2:

use your\_database

db.auth("your\_user", "your\_password")

1. :If authentication is successful, you'll see a message confirming authentication.

**Step 4: Use the Test Script for CRUD Functions**

1. Ensure Python and Dependencies Are Installed: Make sure you have Python installed on your system along with any required dependencies. You can use a virtual environment for isolation.
2. Execute the provided testing script to perform CRUD operations on your dataset.

**Step 5: Use the Jupyter script to launch the Jupyter Dashboard.**

1. Ensure all dependencies are properly installed. Make sure you have all related dashboard files in the same directory.
2. Execute the Dashboard script.
3. Toggle selections to apply specified data filters.