**CS 340 README**

**About the Project**

**Author:** Alexander Wagner

**Project Title:** CRUD Operations and Dashboard for Animal Shelter Database

This project is a full-stack web application that facilitates **CRUD (Create, Read, Update, Delete)** operations on a **MongoDB** database managing animal shelter data. It also provides an interactive dashboard for users to interact with the data, visualize information, and filter records based on specific criteria. The dashboard is built using the **Dash** framework, and the CRUD operations are handled using **PyMongo** for MongoDB integration.

The application was developed for the **CS-340 Client/Server Development** course as part of a project for **Grazioso Salvare**, a company specializing in rescue animal training. The primary goal is to manage animal shelter data efficiently while providing meaningful visual insights to the user.

**Motivation**

The aim of the project is to develop a robust and user-friendly interface for managing animal shelter data, providing users with the ability to easily add, retrieve, update, and delete records. Additionally, the interactive dashboard allows users to explore the shelter data visually, making it easier to identify trends and gain insights.

The project provides hands-on experience with integrating Python, MongoDB, and Dash, which makes it a practical exercise in full-stack development.

**Getting Started**

To begin using this project, set up the development environment by ensuring that **Python 3.x** and **MongoDB** are installed on your machine. Python is required to script the CRUD operations, while MongoDB is needed to manage the database where animal records are stored.

First, clone the repository to your local machine. After cloning, confirm that MongoDB is running locally. You will need to create a database named AAC and a collection called animals. The animal records should be imported into the `animals` collection using the aac\_shelter\_outcomes.csv file. This can be achieved with the following mongoimport command:

mongoimport --host localhost:27017 --username aacuser --password YourPassword123 --authenticationDatabase admin --db AAC --collection animals --type csv --file "C:\Your\Path\aac\_shelter\_outcomes.csv" --headerline

After configuring the database, install the required Python dependencies. The project relies primarily on the **PyMongo** library to interact with MongoDB. You will also need **Jupyter Notebook** to run the CRUD operations and test the project. Install the required libraries using the following command:

pip install pymongo dash dash-leaflet plotly pandas notebook

**Run Jupyter Notebook:**

1. Launch Jupyter Notebook by running the following command in your terminal or command prompt:

jupyter notebook

1. Open the notebook for your project and import the animal\_shelter.py module.

Once the environment is set up and Jupyter Notebook is running, proceed to the **Usage** section to begin interacting with the MongoDB database and running the CRUD operations.

**Usage**

Once your environment is set up and MongoDB is running, you can interact with the animal shelter database using the CRUD operations provided in the `animal\_shelter.py` module. I’ve also added in-app CRUD functionality.

Here are some examples of how to use the module:

**Instantiate the `AnimalShelter` class:**

from animal\_shelter import AnimalShelter

shelter = AnimalShelter(username="aacuser", password="SNHU1234")

**Create a new record:**

new\_animal = {

"animal\_id": "A123457",

"animal\_type": "Dog",

"breed": "Beagle",

"color": "Brown",

"outcome\_type": "Adoption",

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

}

create\_result = shelter.create(new\_animal)

print("Create operation successful:", create\_result)

**Read from the database:**

search\_criteria = {"animal\_id": "A123457"}

read\_result = shelter.read(search\_criteria)

print("Read operation result:", read\_result)

**Update an existing record:**

update\_result = shelter.update({"animal\_id": "A123457"}, {"color": "White"})

print("Update operation modified count:", update\_result)

**Delete a record from the database:**

delete\_result = shelter.delete({"animal\_id": "A123457"})

print("Delete operation removed count:", delete\_result)

**Running the Dashboard**

To launch and interact with the Dash app, you will need to use the provided IPYNB file from the repository.

1. Open the `ProjectTwoDashboard.ipynb` file from the repository in Jupyter Notebook.

2. Ensure that MongoDB is running and you have imported the necessary dependencies.

3. Once the setup is complete, run the cell with the code from the IPYNB file.

This will launch the Dash app directly within the notebook. You can interact with the dashboard from within the notebook interface.

**Functionality**

This project includes both CRUD operations and an interactive dashboard for data visualization and interaction.

The CRUD operations are implemented in the `AnimalShelter` class, which allows users to create, read, update, and delete animal records from the MongoDB database. The application integrates these functionalities into an interactive web interface where users can manage animal data effectively.

The dashboard features a login system that ensures only authorized users can access the shelter data. Once authenticated, users are presented with an interactive dashboard where they can:

* Search and filter animal records by animal type, breed, outcome type, and rescue type (e.g., Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking).
* View an interactive data table that updates based on the applied search filters. The table allows for sorting and pagination to make large datasets easier to navigate.
* Use a geolocation map to visualize the location of rescue animals. When an animal is selected from the table, the map updates to show the animal’s location, along with additional details such as name, breed, and animal ID.
* Analyze a pie chart that displays the distribution of the top 10 most common breeds within the shelter, providing quick insight into breed trends in the data.
* Added an export button that allows users to export the current filtered data as a CSV file.
* Newly added CRUD buttons allow users to add a new record or delete an existing record directly from the dashboard.

The combination of these features makes the project a complete tool for managing and exploring animal shelter data in an intuitive way.

**Tests**

The CRUD module was tested by running all four operations in Jupyter Notebook. The **Create** method was tested by inserting a new record and verifying its existence in the database. The **Read** method was used to retrieve records, including those created during testing. The **Update** method was tested by modifying a field in an existing record and verifying the changes. Finally, the **Delete** method was tested by removing a record and confirming that it was no longer in the database.

The Jupyter Notebook environment allows for easy testing of the `AnimalShelter` class. To run the tests, follow these steps:

1. Import the animal\_shelter.py module into your Jupyter Notebook.
2. Create cells that instantiate the `AnimalShelter` class, then test the create, read, update, and delete methods (provided in the Usage section).
3. Run the cells by pressing **Shift + Enter**. The output will display whether the operations were successful, as demonstrated by the **Create operation successful** and **Read operation result** outputs in the notebook.

**Why MongoDB Was Used**

**MongoDB** was selected as the database for this project because it is a **NoSQL database** that allows for flexibility in data structure, making it ideal for handling unstructured or semi-structured data like animal shelter records. MongoDB’s schema-less nature means that fields can vary from one document to the next, which is useful when storing different types of animal data (e.g., breed, age, medical history).

For this project, MongoDB interfaces seamlessly with Python through the **PyMongo** library, which allows for simple CRUD operations using Python scripts. MongoDB's capability to handle large datasets and perform complex queries with fast read/write operations ensures that the dashboard remains responsive even with a significant amount of data.

**Dash Framework and Application Structure**

The **Dash** framework was used to build the web application’s dashboard, providing both the **view** and **controller** structure. Dash is a Python framework built on top of Flask, Plotly.js, and React.js. It allows developers to create interactive web applications entirely in Python. The view in Dash is made up of various components such as input fields, data tables, charts, and maps that render dynamic content based on user interaction.

The **controller** is defined through **callbacks** in Dash, which link the input components to the output components (like a data table or map), updating them based on user input.

Dash makes it simple to tie together the backend CRUD functionality (MongoDB queries) with an intuitive user interface, making it a natural choice for building this application.

**Links to Resources**

The following resources and software applications were accessed and used during the project:

* **MongoDB**(https://www.mongodb.com/): Used as the NoSQL database to store and manage animal shelter records.
* **PyMongo**(https://pymongo.readthedocs.io/en/stable/): The official MongoDB driver for Python, used for CRUD operations.
* **Dash**(https://dash.plotly.com/): The framework used to build the web application’s dashboard.
* **Dash Leaflet** (https://dash-leaflet.herokuapp.com/): A Dash extension used for creating the interactive maps in the dashboard.
* **Plotly** (https://plotly.com/python/): The graphing library used to create the pie charts in the dashboard.
* **Pandas** (https://pandas.pydata.org/): Used for data manipulation, especially when processing the data retrieved from MongoDB.
* **Jupyter Notebook** (https://jupyter.org/): Used for testing and developing the Python code interactively.

**Steps Taken to Complete the Project**

The project began by developing the CRUD functionality in the `AnimalShelter` class, allowing for the basic operations of adding, retrieving, updating, and deleting animal records in the MongoDB database. Once this foundation was laid, the focus shifted to developing the interactive dashboard using Dash.

The dashboard was designed to provide a simple, intuitive interface where users could interact with the data through search and filter functions. The geo-location map and pie chart were integrated to enhance data visualization, making it easier for users to analyze trends and specific records within the dataset. Additionally, an authentication system was added to secure access to the dashboard, ensuring that only authorized users can interact with the data.

Throughout the project, testing and debugging were carried out to ensure that all functionalities, including data filtering, CRUD operations, and visualizations, worked seamlessly together.

**Challenges and Solutions**

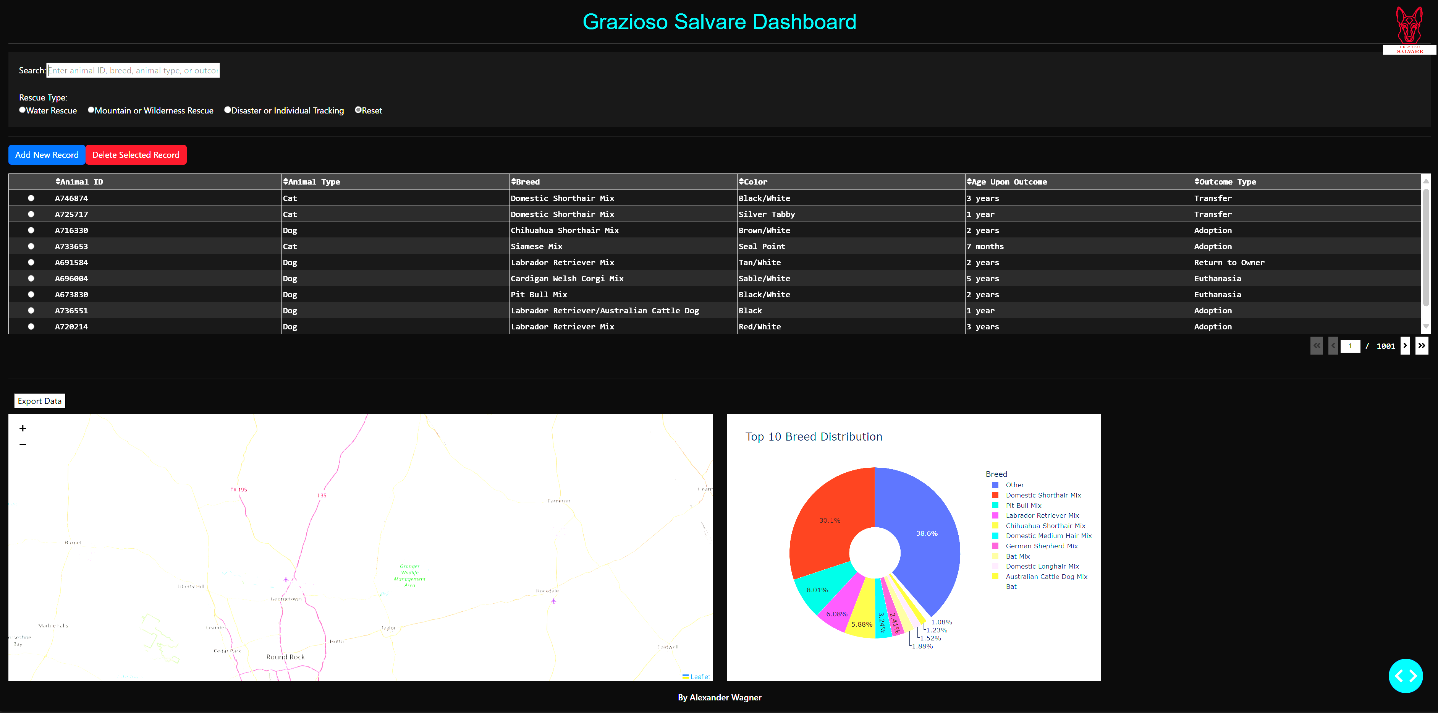
One challenge encountered was ensuring the real-time interaction between the MongoDB data and the Dash components. This was resolved by integrating PyMongo with Pandas, allowing for efficient manipulation of the data retrieved from MongoDB and seamless updates in the dashboard.

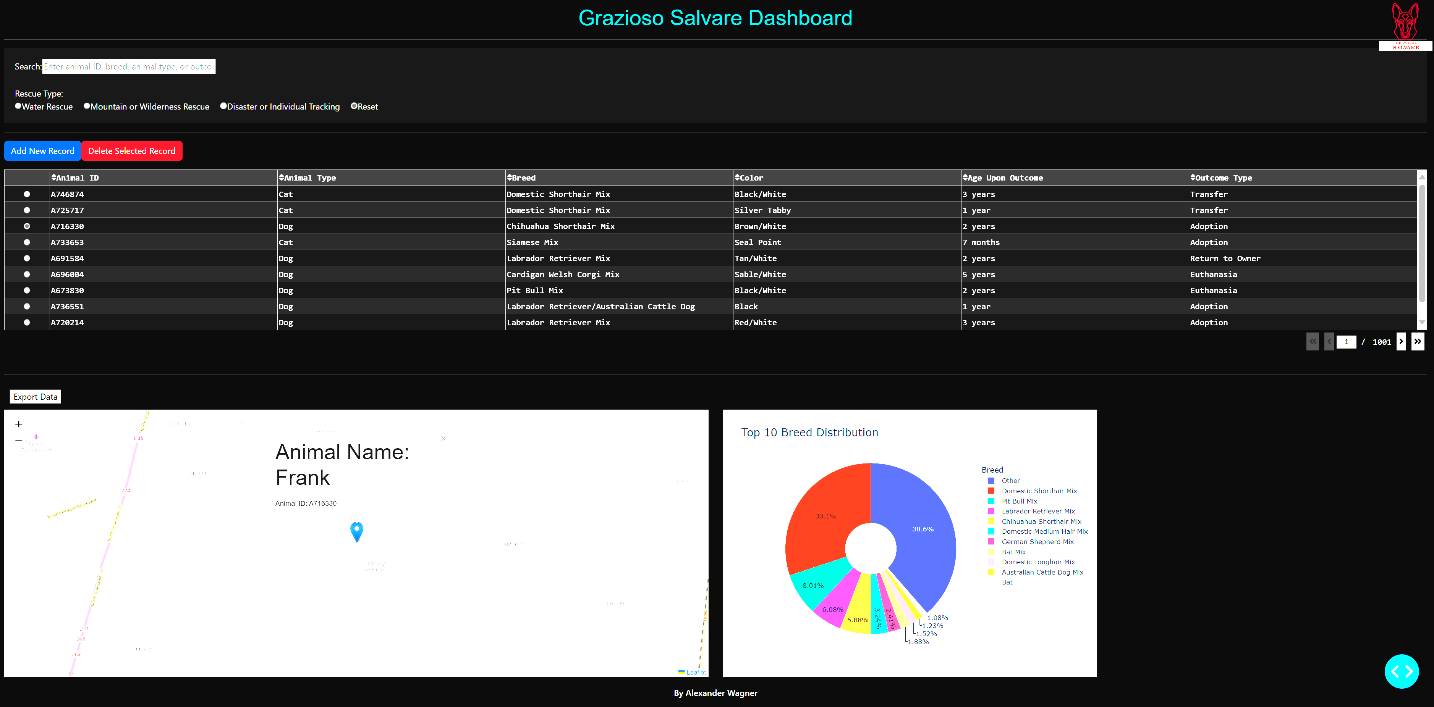
Another challenge was implementing the geo-location map. The issue was resolved using Dash Leaflet, which provided a flexible and user-friendly way to create dynamic maps that responded to user inputs.

**Screenshots**

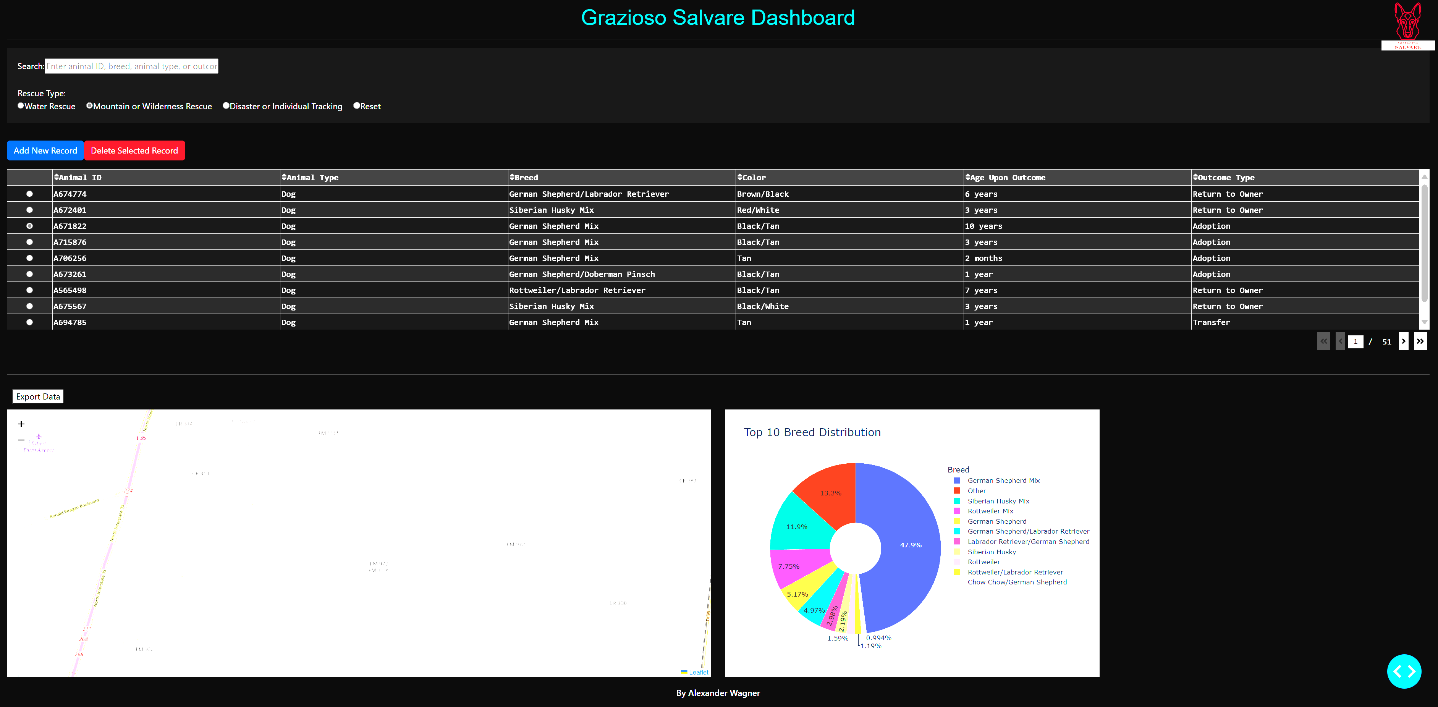
**Login Screen**

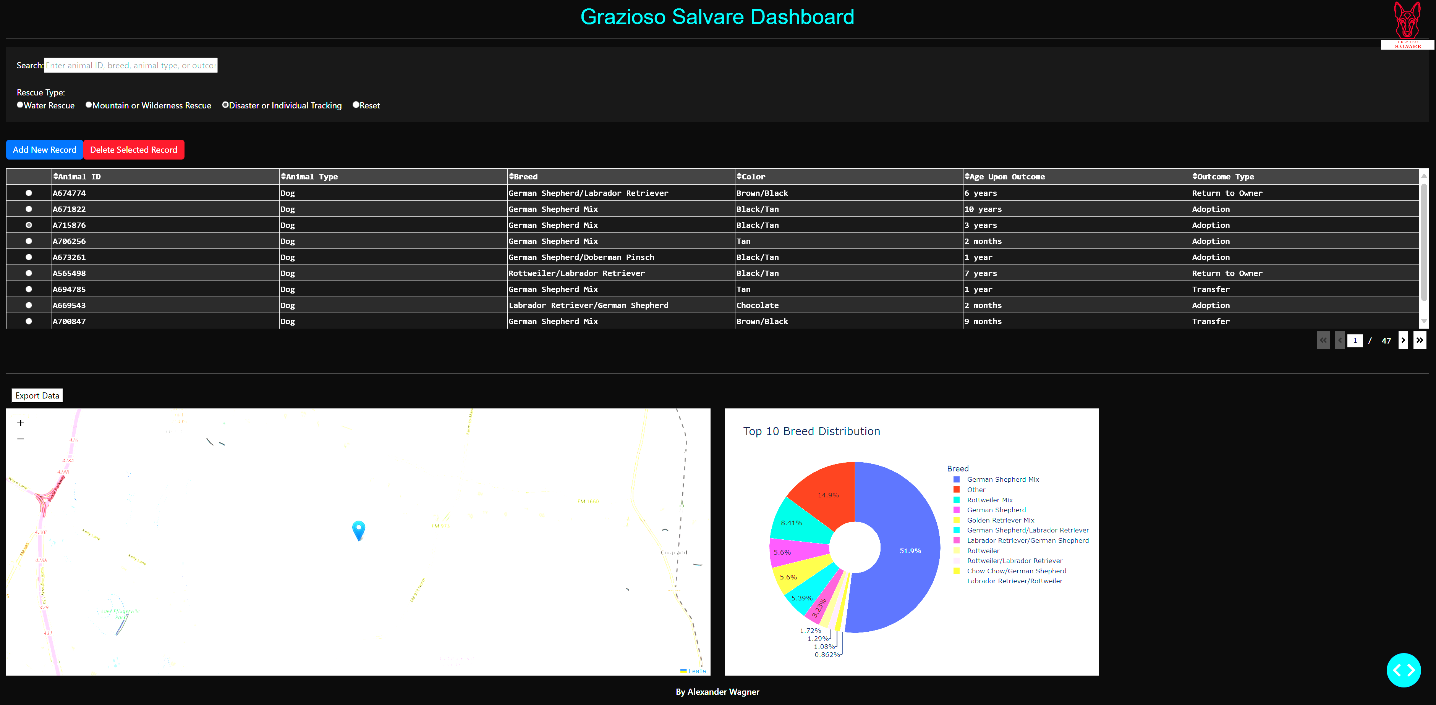
**Default State**

****

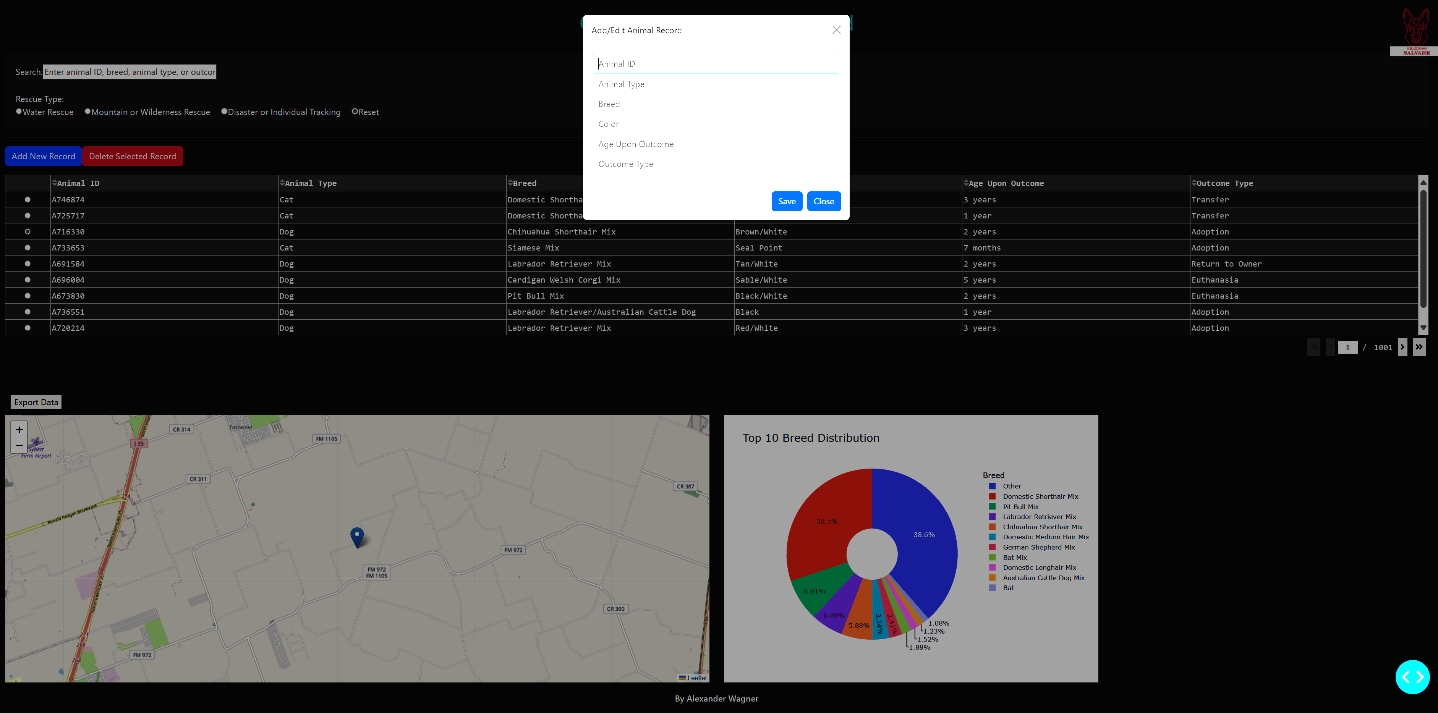
**Row Selection**

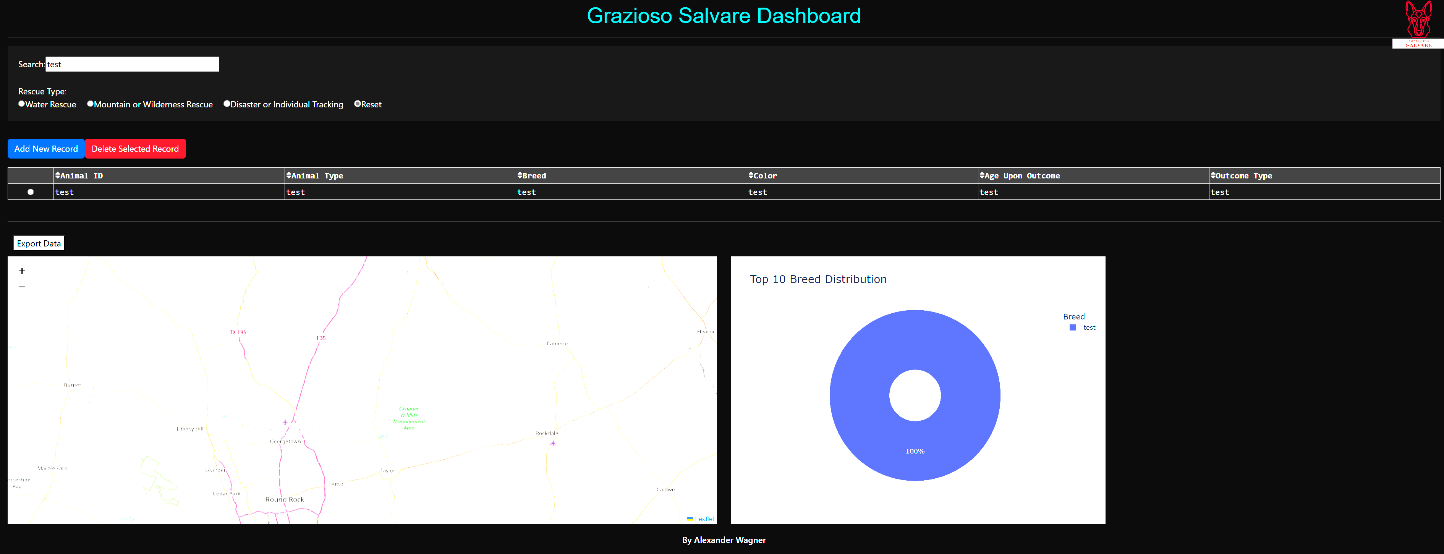
**Mountain/Wilderness Filter**

****

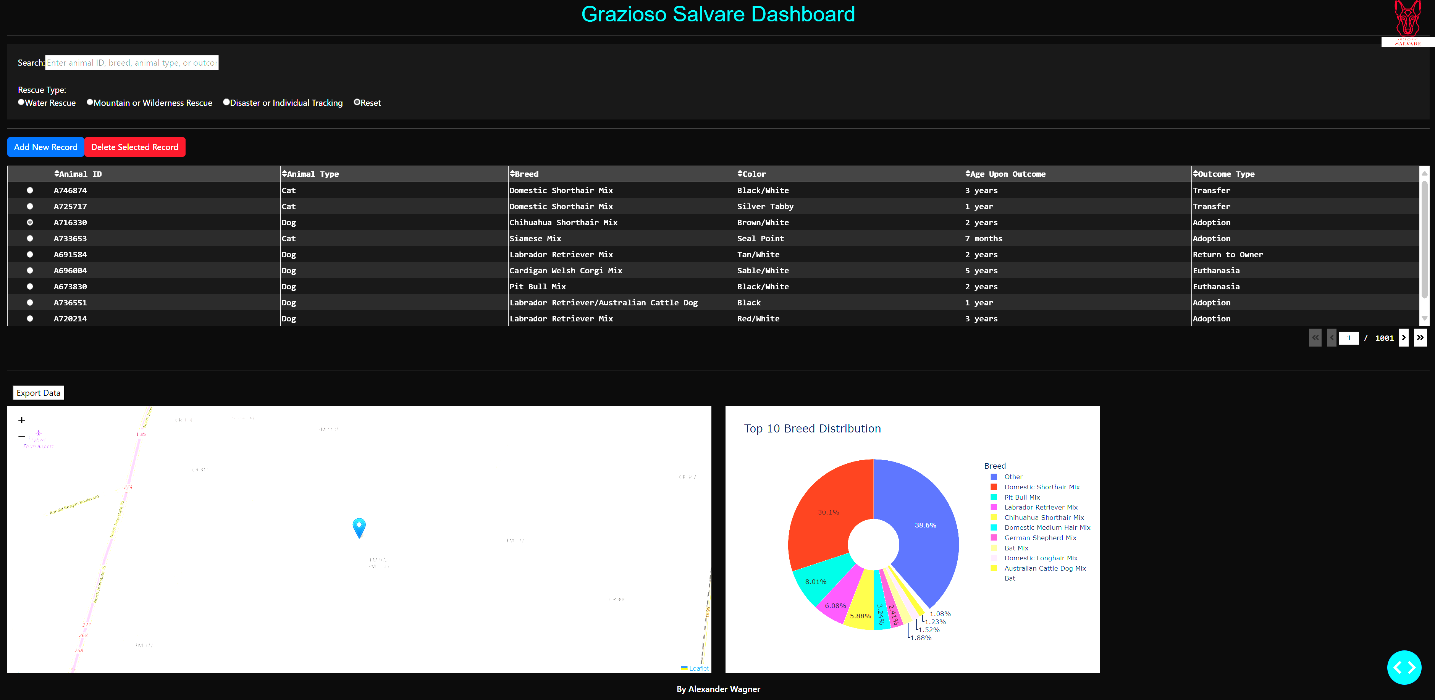
**Disaster/Individual Filter**

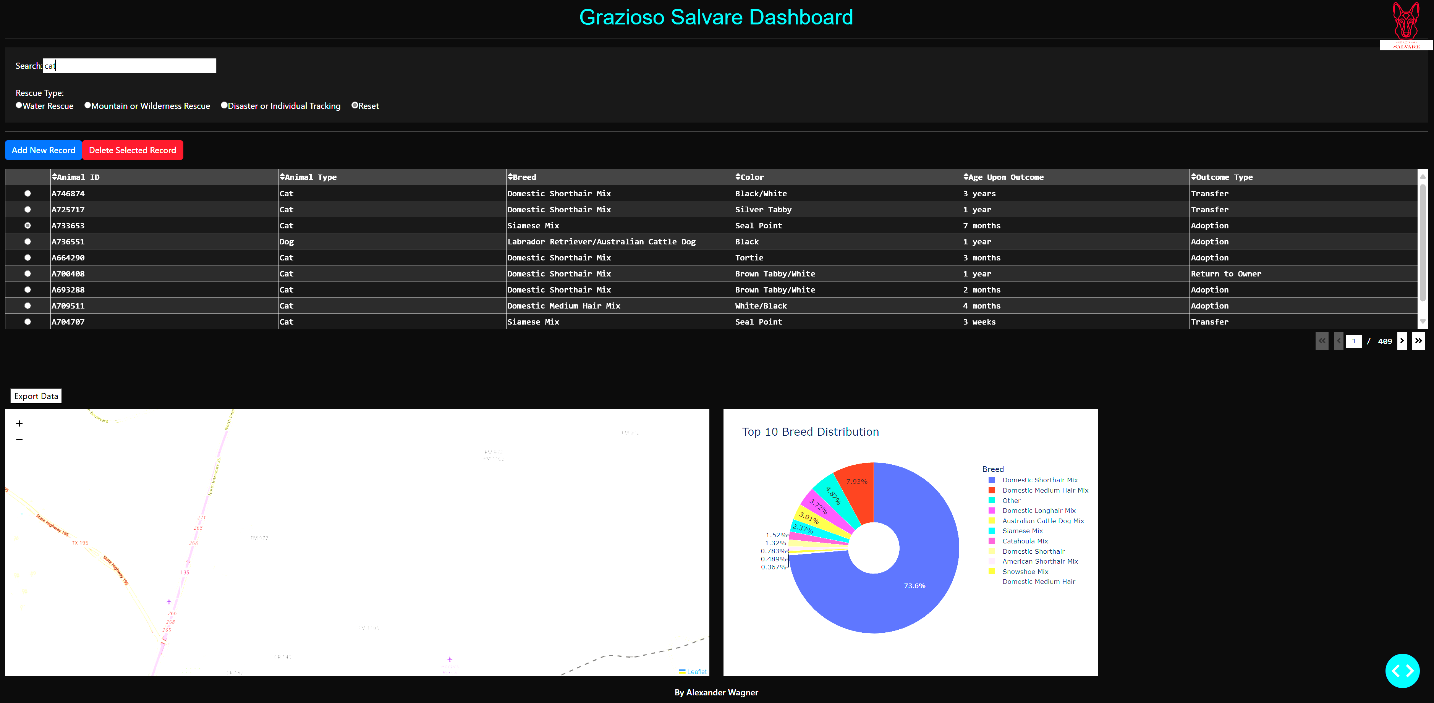
**Add New Record via CRUD**

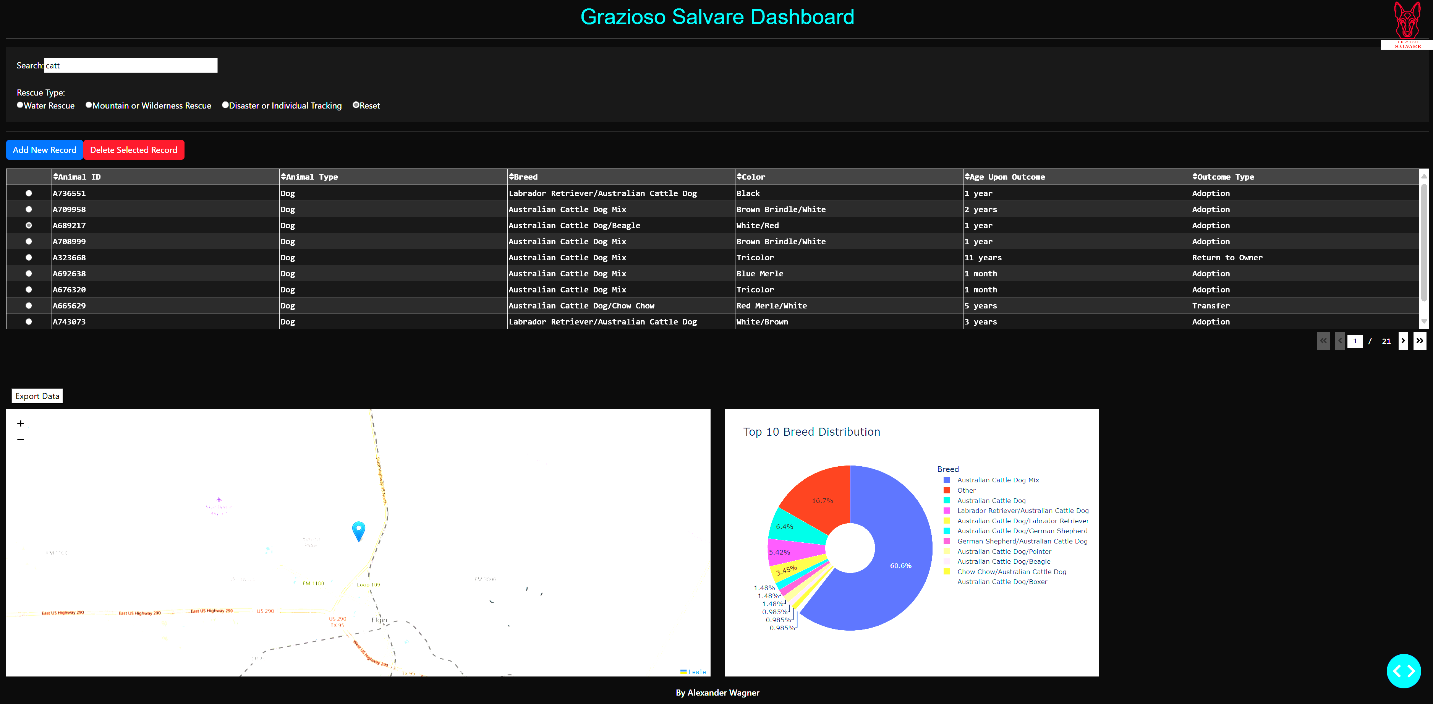
****

**Verify Record Added**

**Reset Filters**

****

**Search Bar Updating Table/Chart**

****

**Contact**

For any questions or feedback regarding the project, feel free to contact:

**Alexander Wagner**

Email: alexander.wagner@snhu.edu

GitHub: github.com/AlexSWagner