# **README for Grazioso Salvare Project**

## **About the Project**

This project, developed by Global Rain for Grazioso Salvare, aims to identify and categorize dogs suitable for search-and-rescue training using data from animal shelters. The project's goal is to create a database in MongoDB that interacts with client-side code, supporting full stack development.

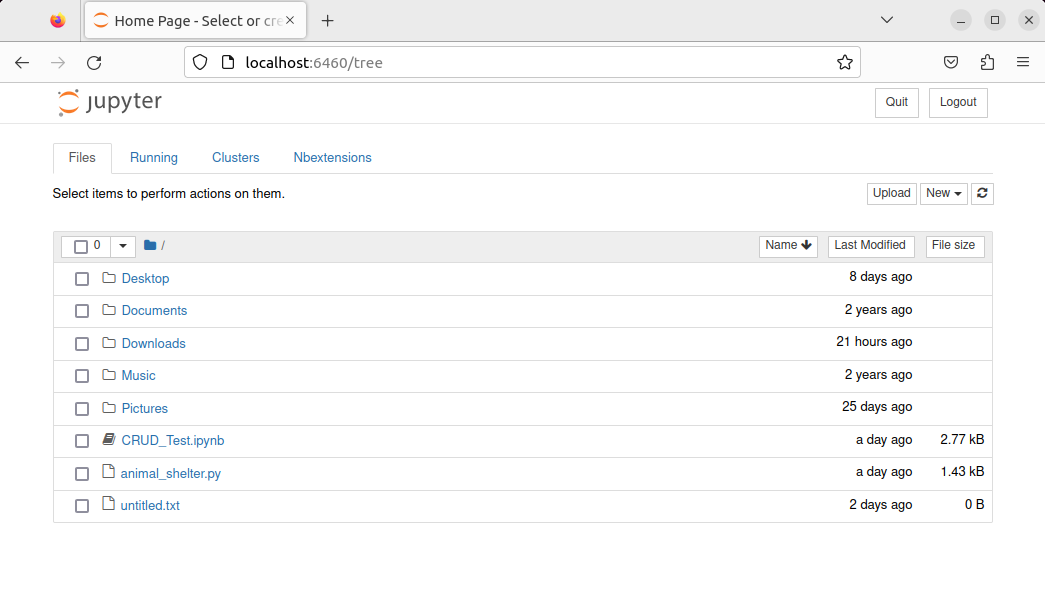
## **Motivation**

The project aims to enhance skills in creating reusable code for MongoDB and develop a web application to perform CRUD operations, supporting search-and-rescue training efforts.

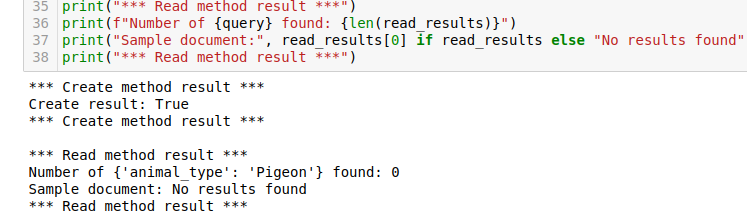
**Getting Started**

To run this program locally, you will need Jupyter Notebook or a Jupyter extension installed on your chosen IDE(not explained here due to wide range of variations).

1. Place the ‘CRUD\_Test.ipynb’ and ‘animal\_shelter.py’ files to your Jupyter notebook directory.



1. Open the ‘CRUD\_Test.ipynb’ file and run it. You will see the results of the create and read methods in the output.



## **Installation**

You will need Python and Jupyter Notebook to run this program.

1. Python: Download from here - <https://www.python.org/downloads/>
2. Jupyter Notebook: Available in the Python Package Index. To install with pip from a terminal prompt:

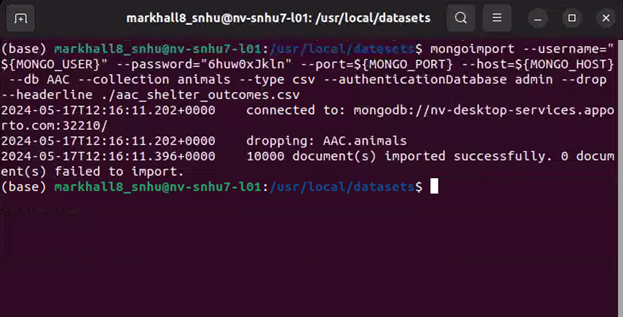


To run the notebook:



1. MongoDB: Import the dataset using the mongoimport tool.

mongoimport --username="${MONGO\_USER}" --password="6huw0xJkln" --port=${MONGO\_PORT} --host=${MONGO\_HOST} --db AAC --collection animals --type csv --authenticationDatabase admin --drop --headerline ./aac\_shelter\_outcomes.csv

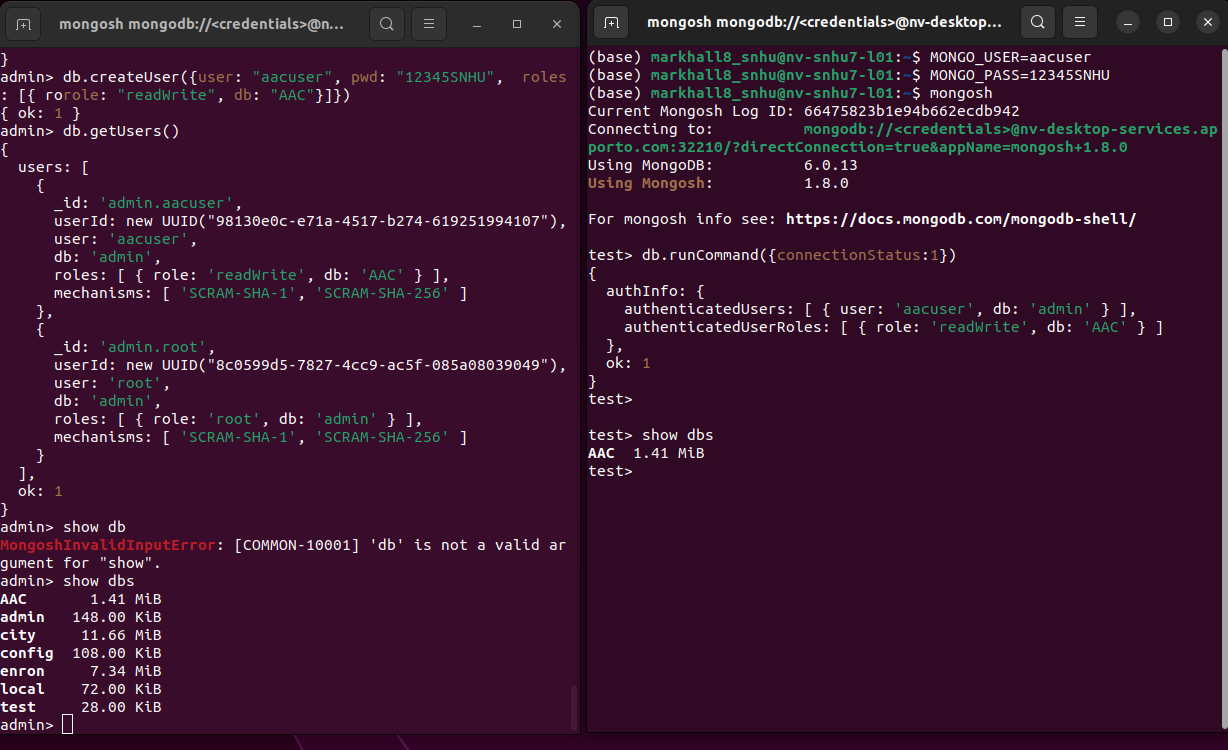


4. Setting up user authentication:

1. Create a user account called “aacuser” in the mongo shell.

Use admin

db.createUser({user: "aacuser", pwd: "YOURPASSWORD", roles: [{ role: "readWrite", db: "AAC"}]})

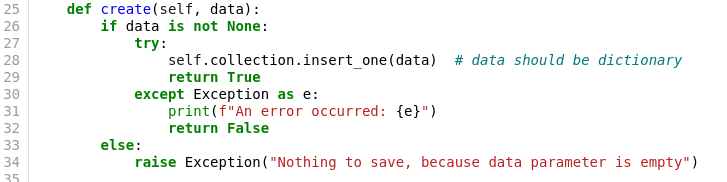


## **Usage**

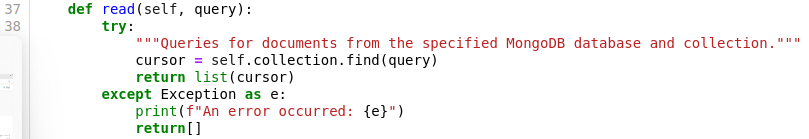
This project includes a Python module that provides CRUD functionality for the MongoDB database.

### **Code Example**

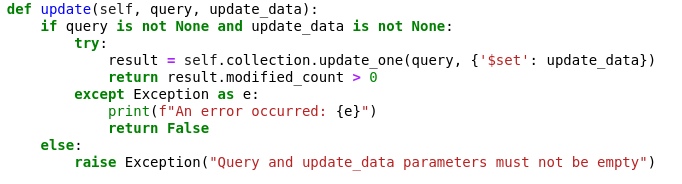
Create method: The create method attempts to insert a document. If successful, it returns True and False if unsuccessful. Error handling ensures if it fails it does so gracefully.



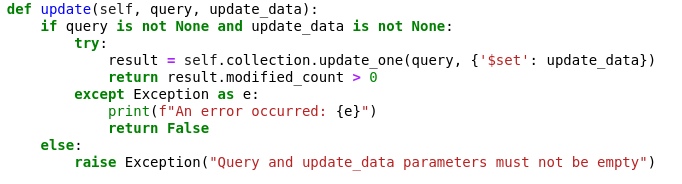
Read method: Similarly structured, the read method attempts to find a document and returns the document or an empty list.



Update method: This method allows updating of documents. It includes error handling.



Delete method: Lastly, the delete method allows deletion of a record.



### **Tests**

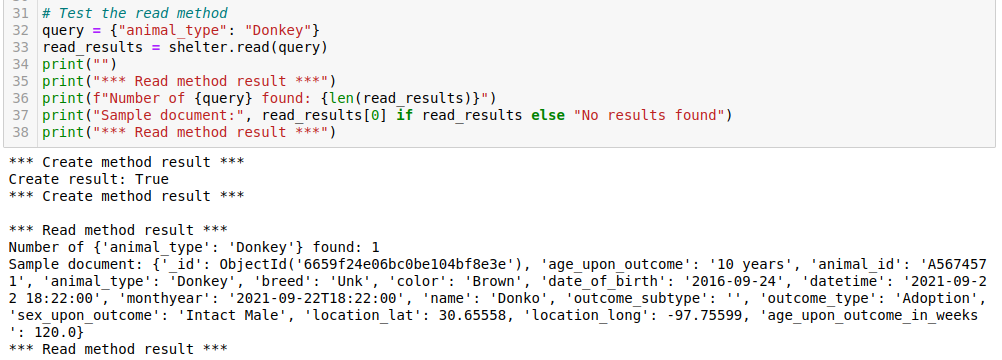
***Create functionality*:** In the test create method section, edit some parameters of new\_animal and run the program again to see if it was successful.



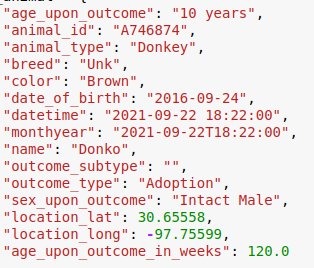


***Read functionality*:** Additionally, you can search for animals by any field to test the read method.

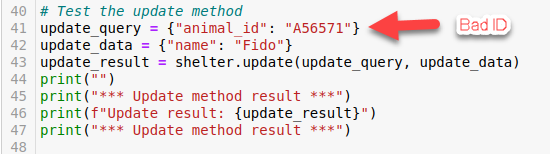
***Example:*** Enter an animal type and run it to see the results.

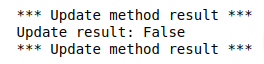
****

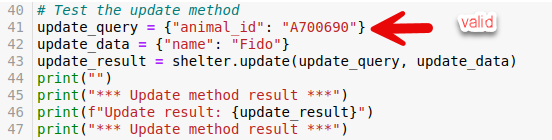
To test read functionality on another parameter change query to any other field and enter the search term, for example: query = {“breed”: “Siamese”}. The available fields are all listed in the create method test ↓

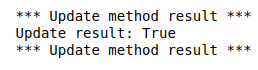


***Update* *functionality*:** The update method can be tested by changing the animal\_id to a valid id. This can be used to update the name for testing purposes. If the id is not found it will return false. If found, it will return true.



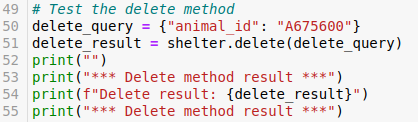


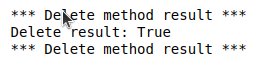




Like the read method, you can change the field you want to update to any of the available fields. Example: Instead of changing the name to Fido in the above example, you could change the color to brown by changing that field to update\_data = {“color”: “brown”}

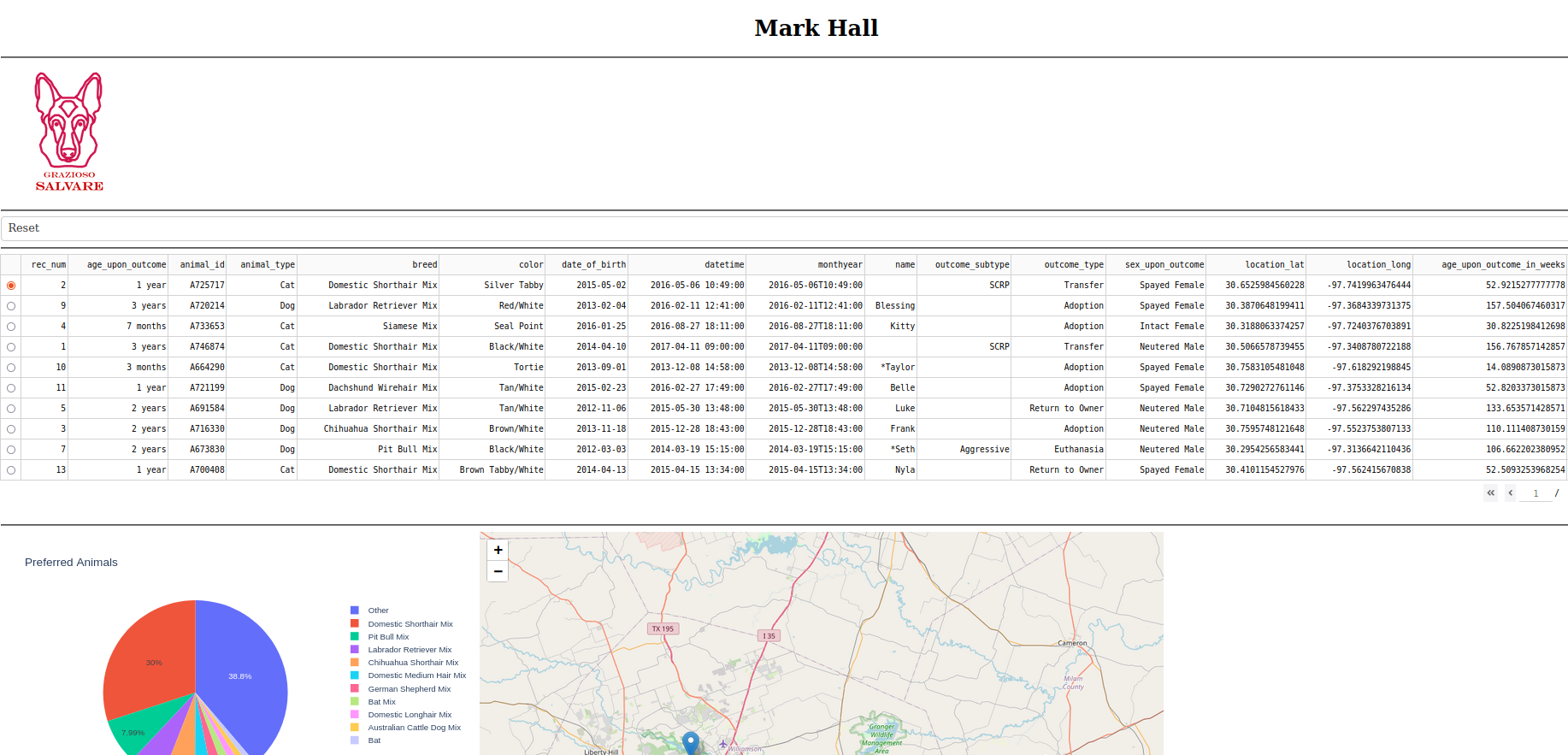
***Delete* *functionality*:** To test deletion of a document, enter a valid ID and run. Similar to the update test you will see true if successful, false if not.



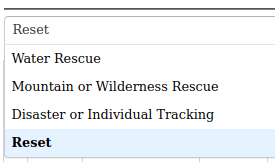


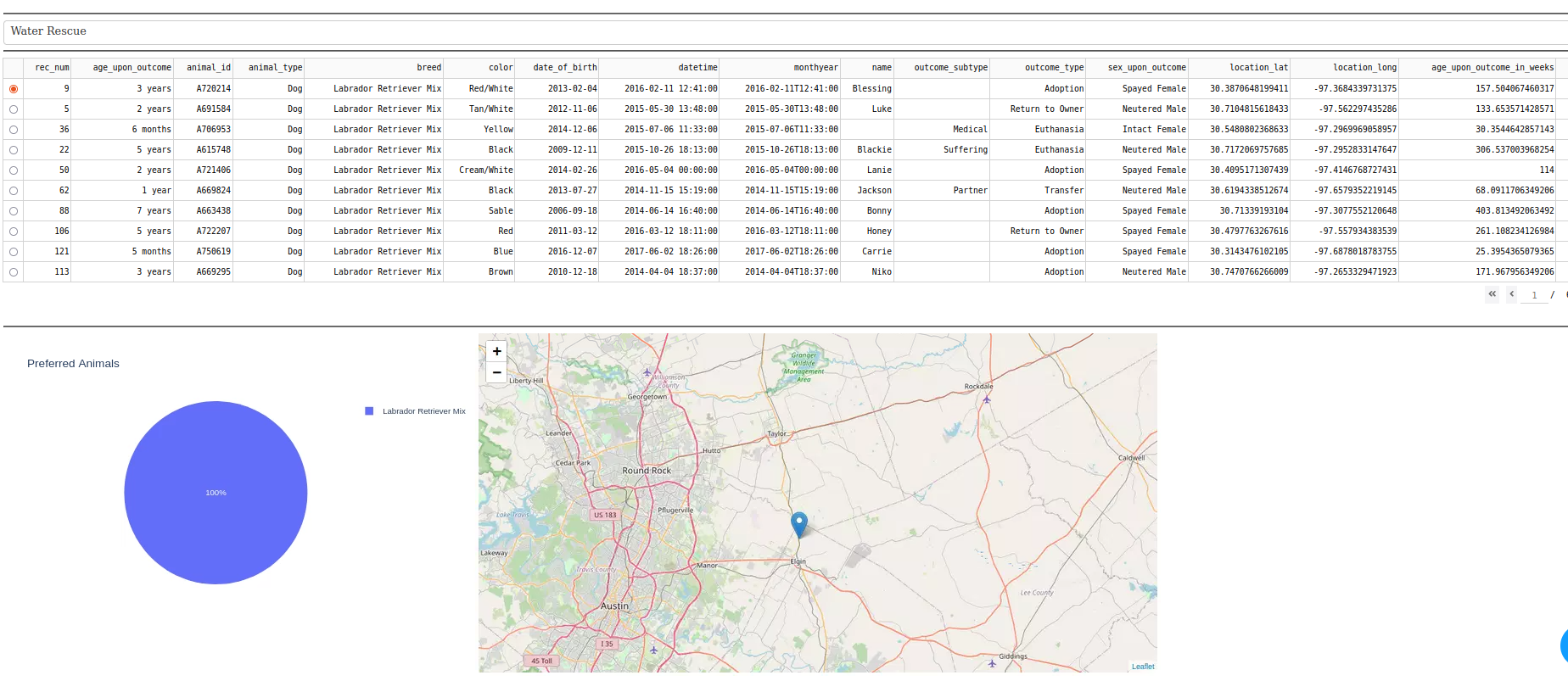
**Navigating the site**

This screenshot shows the initial state of the Grazioso Salvare dashboard. It includes the interactive filtering options (dropdown), the interactive data table, and the charts, all set to their default states.

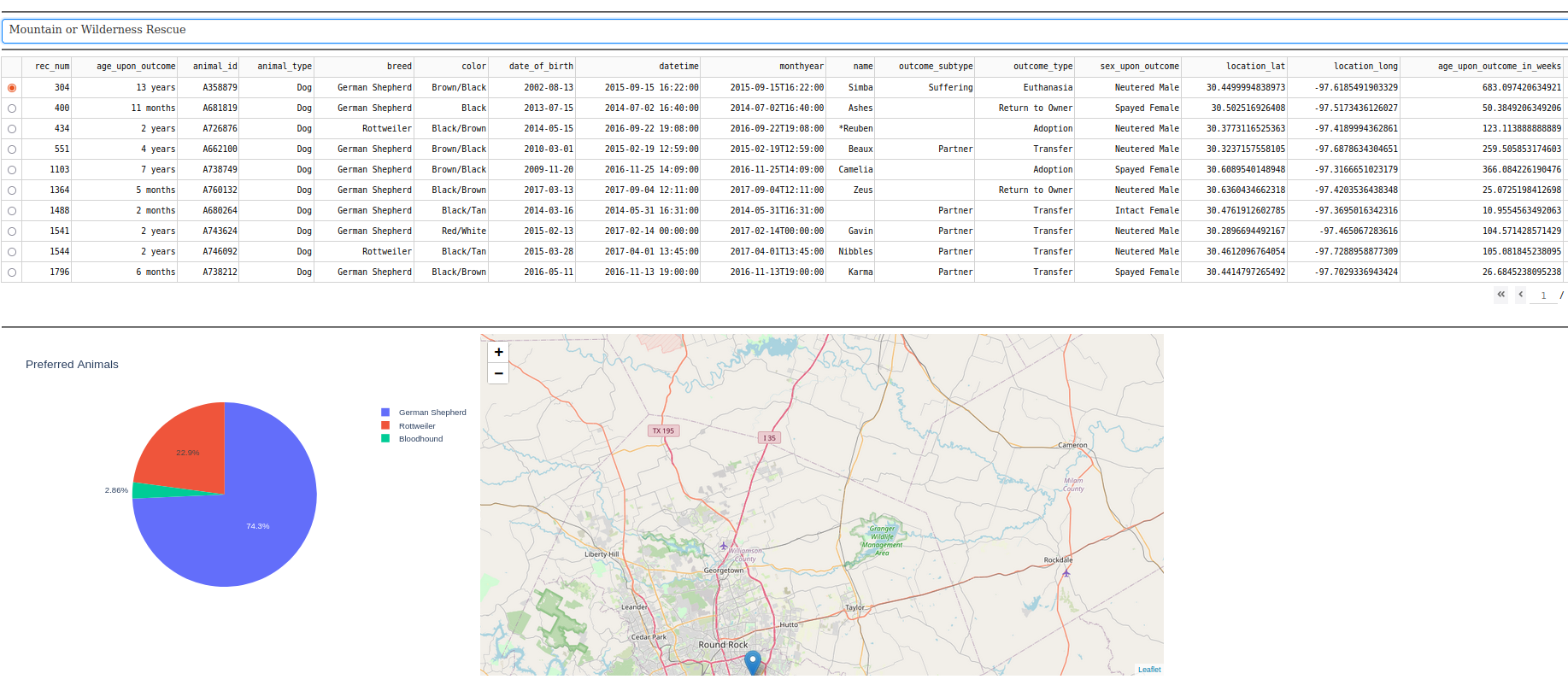
****

Available filter options

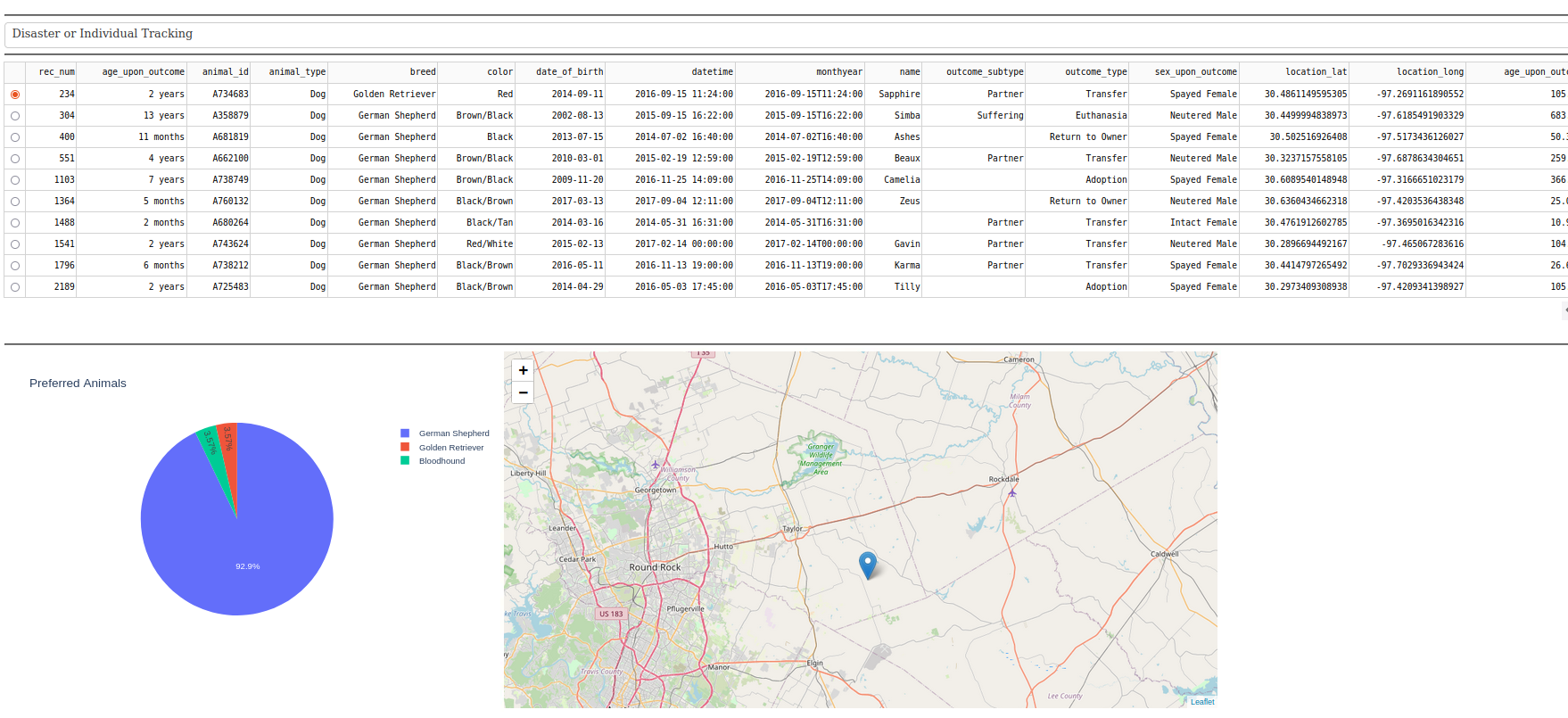


This screenshot demonstrates the dashboard after applying the "Water Rescue" filter. The data table updates to show only the animals suitable for water rescue, and the charts reflect the filtered data.

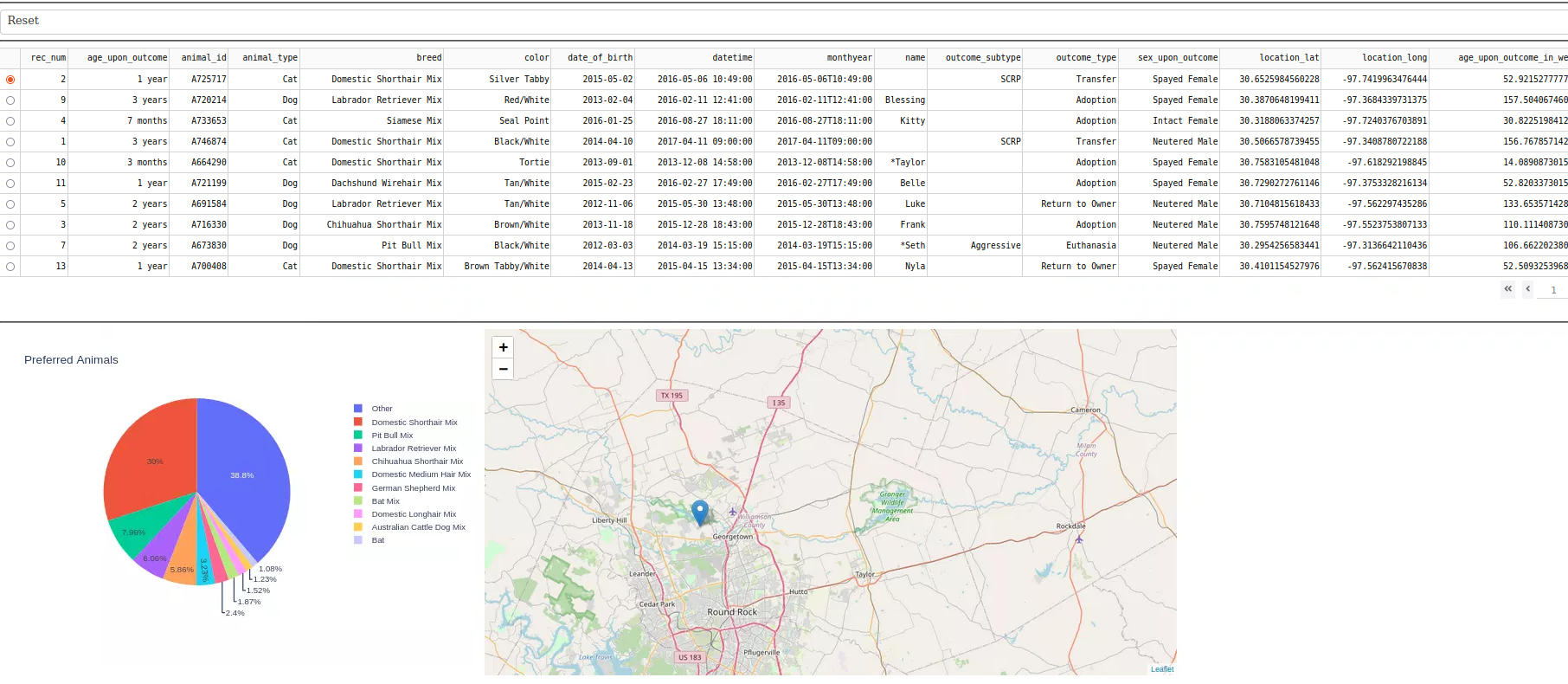
Here you can see the dashboard with the "Mountain or Wilderness Rescue" filter applied. The data table and charts update to display only the animals appropriate for mountain or wilderness rescue.



This screenshot highlights the dashboard after selecting the "Disaster or Individual Tracking" filter. The displayed data and visualizations are now tailored to show animals suitable for disaster or individual tracking.



Finally, the reset option can be selected to return the dashboard to its initial state.



**Tools and Rationale**

**MongoDB -** MongoDB was chosen as the model component for its flexibility and scalability. It allows for easy storage and retrieval of complex data structures, which is ideal for managing the lare dataset that makes up the animal shelter database. MongoDB's integration with Python via PyMongo simplifies the implementation of CRUD operations, making it a good choice for this project.

**Dash Framework -** The Dash framework provides a structure for building interactive web applications with Python. It seamlessly integrates the view and controller components, enabling the creation of responsive and dynamic user interfaces. Dash's compatibility with Plotly allows for advanced data visualization capabilities, enhancing the overall user experience of the dashboard.

**Project Steps and Associated Challenges**

**Setup -** The initial setup involved configuring the MongoDB database and integrating it with the Python module to handle CRUD operations. This provided a solid foundation for data management. Challenges included ensuring secure and efficient database connections, which were overcome by thorough persistence, testing and optimization.

**Dashboard Development -** Developing the dashboard involved creating various interactive components, such as the data table, pie chart and interactive map, using the Dash framework. Each component was designed to respond to user inputs and display relevant data. A key challenge was managing the large dataset, especially in the pie chart visualization. This was addressed by implementing data filtering techniques prior to displaying the chart.

**Filtering and Interactivity -** The implementation of interactive filtering options, such as dropdown filtering, allows users to easily filter the data based on specific criteria. This enhanced the usability and functionality of the dashboard. Ensuring real-time updates and responsiveness of the dashboard components required efficient callback functions and optimization of data queries.

**Testing -** Comprehensive testing was conducted to verify the functionality and performance of the dashboard. This was straightforward and ensured the final product met the requirements.

**Roadmap**

* Expand search functionality across more fields to allow wider range of search functionality.

## **Contact**

Your name: Mark Hall

Email: mark.hall8@snhu.edu