**AAC Jupyter Dashboard**

The AAC Dashboard is a web-based application written in Python for reading data stored in the AAC database. It is and implements a GUI to abstract the user away from the database to interact with an intuitive table to sort through its data.

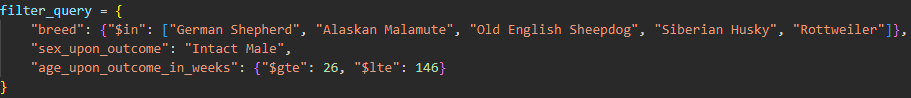
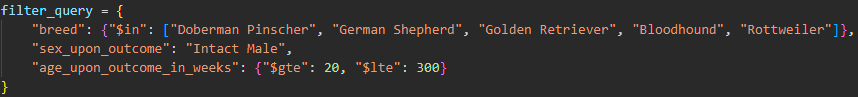
MongoDB was chosen as the backend database for its intuitive web API, making it an optimal choice for modern web applications.

**Dependencies**

1. Pymongo – Pymongo is used to interface with the MongoDB Database.
   1. https://pypi.org/project/pymongo/
2. JupyterDash – Used to create the table and geolocation functionality.
   1. https://pypi.org/project/jupyter-dash/
3. Plotly – Required for the chart functionality.
   1. https://plotly.com/python/getting-started/

**Table Functionality**

* **Table**
  + Radio item selection for preferred rescue animals. The Reset option will revert the table to its unfiltered state.
    - Water Rescue Query: **A black background with text

      Description automatically generated**
    - Mountain Rescue Query: 
    - Disaster Rescue Query: 
* **Pie Chart**
  + Displays the ratio of breeds shown in the table

A colorful circle with black text

Description automatically generated

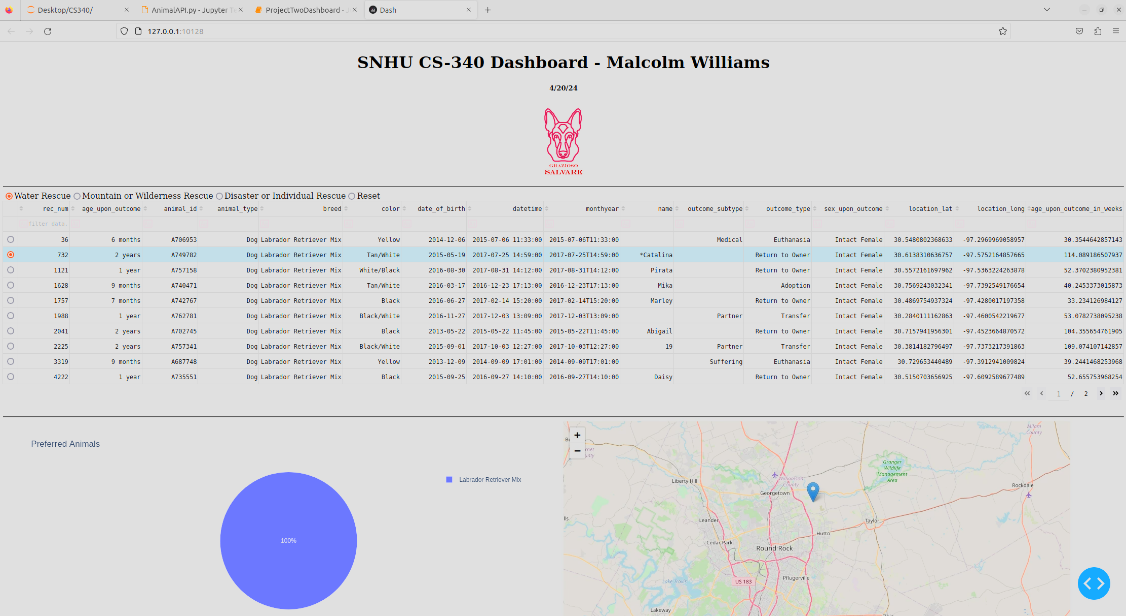
* **Geolocation**
  + Displays the latitude and longitude coordinates on a map of the animal in the selected row.

A map with a blue pin on it

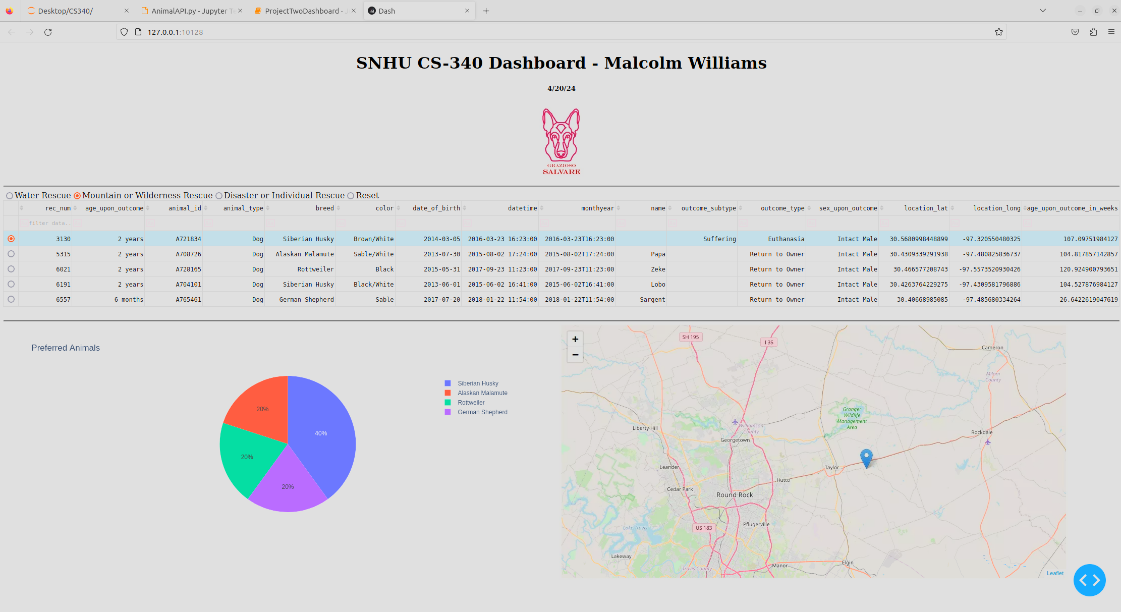
Description automatically generated

**Screenshots**

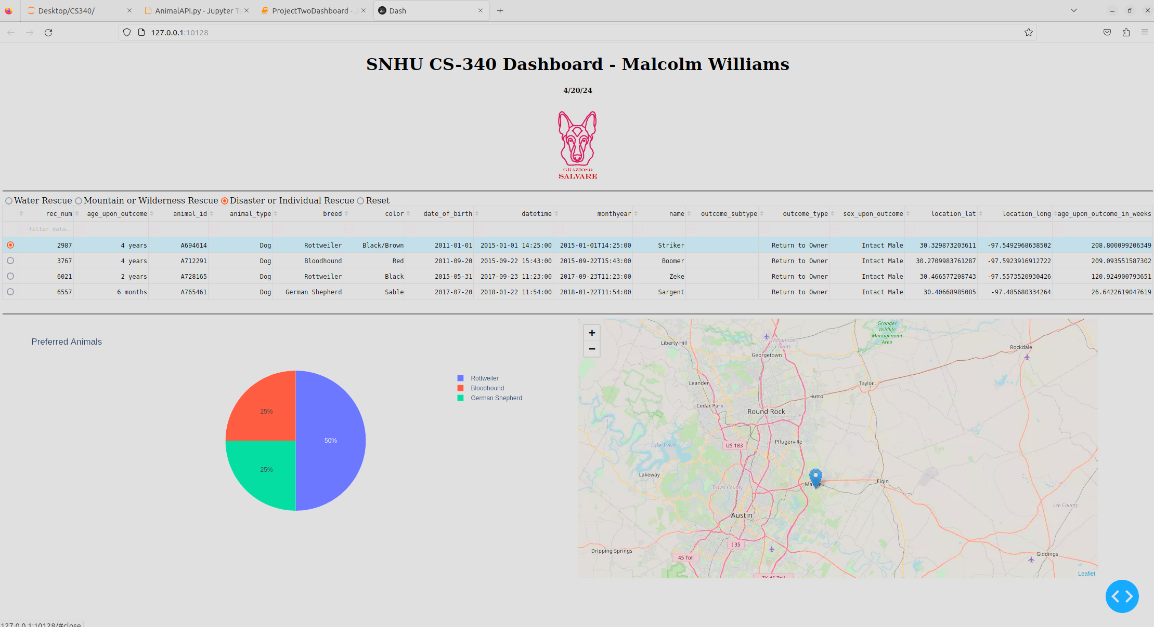
**Preferred Rescue – Water**

****

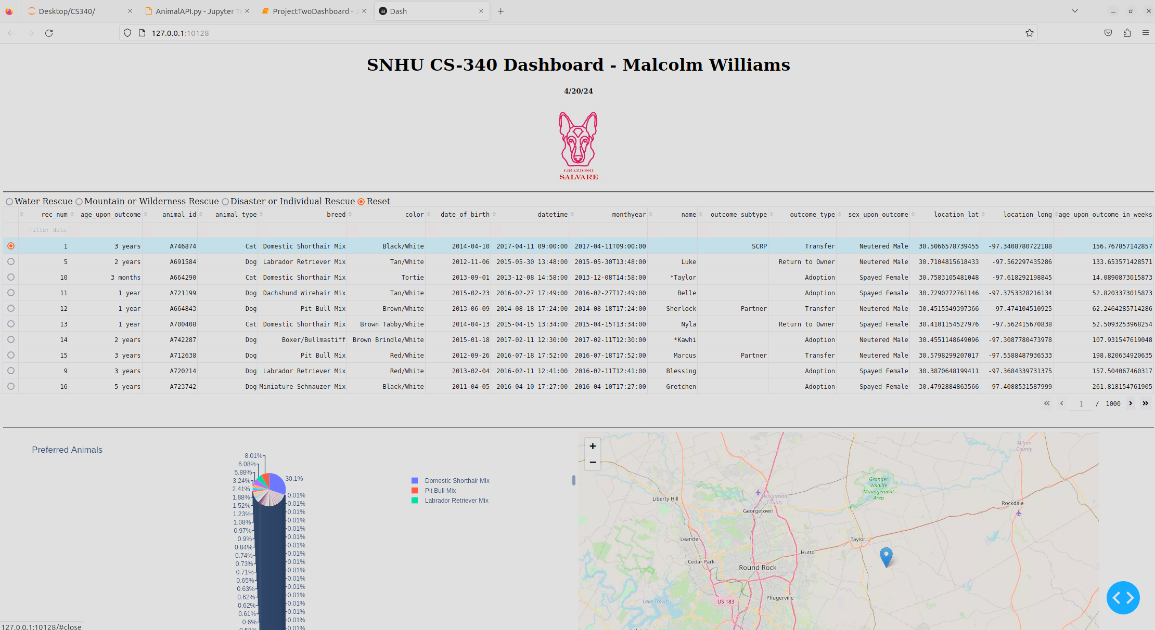
**Preferred Rescue – Mountain**

****

**Preferred Rescue – Disaster**

****

**Reset Table**

****

**AnimalAPI.py**

## Usage

### Code Example

Initialization:

The AnimalShelter object provides CRUD access to the Austin Animal Center database.

from AnimalAPI import AnimalShelter

shelter = AnimalShelter()

test\_data = {

"name": "Greenie Weenie",

"animal\_type": "Dog",

"breed": "Dachshund",

"color": "Green"

}

Create a document:

The create function accepts key/value pairs and returns a Boolean value, representing the success of the insertion of the document.

insert = shelter.create(test\_data)

Read documents:

The read function accepts a filter of key/value pairs and returns a list of all matching documents.

read = shelter.read({"name": "Greenie Weenie"})

Update documents:

The update function accepts a filter of key/value pairs, and data to write to all matching documents. It returns the number of documents that were modified.

update = shelter.update({"name": "Greenie Weenie"}, {"color": "Lime"})

Delete documents:

The delete function accepts a filter of key/value pairs and deletes all matching documents. It returns the number of documents that were deleted.

delete = shelter.delete({"name": "Greenie Weenie"})

### Tests

from AnimalAPI import AnimalShelter

shelter = AnimalShelter()

test\_data = {

"name": "Greenie Weenie",

"animal\_type": "Dog",

"breed": "Dachshund",

"color": "Green"

}

insert = shelter.create(test\_data)

print("Insert Result: {}".format(insert))

read = shelter.read({"name": "Greenie Weenie"})

print("Read Result: {}".format(read))

update = shelter.update({"name": "Greenie Weenie"}, {"color": "Lime"})

print("Documents Updated Count: {}".format(update))

read = shelter.read({"name": "Greenie Weenie"})

print("Read Result: {}".format(read))

delete = shelter.delete({"name": "Greenie Weenie"})

print("Documents Deleted Count: {}".format(delete))

## Contact

Malcolm Williams: Malcolm.williams4@snhu.edu