# CS 340 README

## About the Project

The Animal Shelter Database Management System is designed to manage animal records for animal shelters. This system provides a streamlined way to perform Create, Read, Update, and Delete (CRUD) operations on animal data stored within a MongoDB database. The project uses Python and MongoDB, making it easier for shelters to maintain accurate and up-to-date records of their animals, including details such as name, type, breed, and age, etcetera. On top of this a dashboard was made where the shelter data can be viewed and filtered using a Dash framework. Dash allows developers to focus on the core logic of the application while achieving highly interactive and responsive user interfaces. The combination of MongoDB for the backend and Dash for the frontend provides a robust solution for developing web applications that require real-time data processing and visualization.

## Motivation

The motivation behind this project was to create an efficient and user-friendly system to help animal shelters manage their records with ease. Recognizing the challenges shelters face in keeping track of numerous animals, this system aims to simplify data management tasks, thereby allowing staff to focus more on animal care. Python was chosen for syntax readability and how easily code can be written in the language. Python code is also easy to run with the built in complier, especially when using a program like Jupyter Notebook.

## Getting Started

To get a local copy up and running, follow these simple example steps:

## Installation

Install Python and MongoDB.

Before running the application, install the necessary Python libraries in your command prompt:

pip install pymongo

pip install bson

## Usage

The system allows shelters to create, read, update, and delete animal records. Here's how to use it:

Creating a Record:

from animal\_shelter import AnimalShelter

shelter = AnimalShelter()

animal\_data = {"name": "Odin", "type": "Dog", "breed": "German Shepherd", "age": 1}

shelter.create(animal\_data)

Reading a Record:

results = shelter.read({"type": "Dog"})

print(results)

Updating a Record:

animal\_data = {"name": "Odin", "type": "Dog", "breed": "German Shepherd", "age": 1}

new\_animal\_data = {"name": "Odin", "type": "Dog", "breed": "German Shepherd Mix", "age": 2}

shelter.update(animal\_data, new\_animal\_data)

Delete a Record:

new\_animal\_data = {"name": "Odin", "type": "Dog", "breed": "German Shepherd Mix", "age": 2}

shelter.delete(new\_animal\_data)

### Code Example

Here's an example of how to instantiate the AnimalShelter class and create, update and delete a new animal record:

shelter = AnimalShelter()

shelter.create({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": 3})

shelter.update({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": **3**}, {"name": "Bella", "type": "Cat", "breed": "Siamese", "age":**4**})

shelter.delete({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": 4})

### Tests

To test the functionality of the CRUD operations, you can run the following script:

shelter = AnimalShelter()

shelter.create({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": 3})

print(shelter.read({"name": "Bella"}))

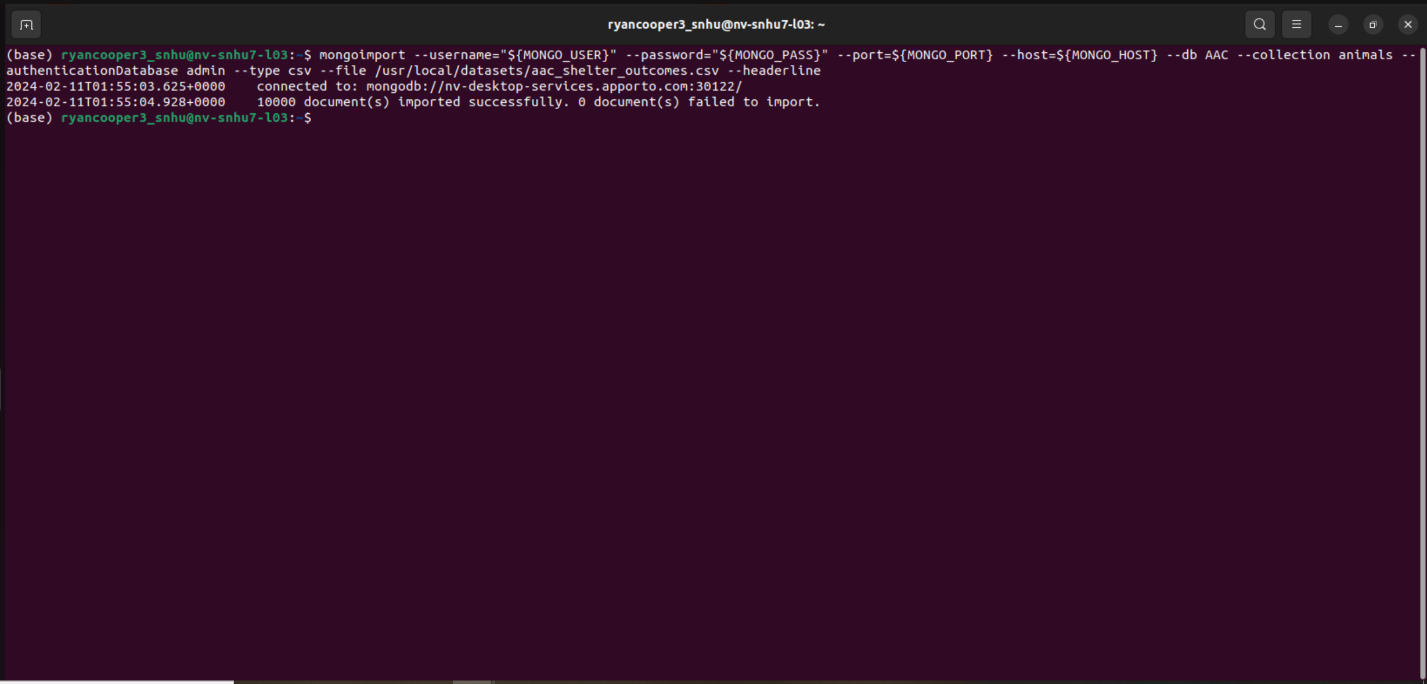
shelter.update({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": **3**}, {"name": "Bella", "type": "Cat", "breed": "Siamese", "age":**4**})

print(shelter.read({"name": "Bella", "type": "Cat", "breed": "Siamese", "age":**4**}))

shelter.delete({"name": "Bella", "type": "Cat", "breed": "Siamese", "age": 4})

print(shelter.read({"name": "Bella", "type": "Cat", "breed": "Siamese", "age":**4**}))

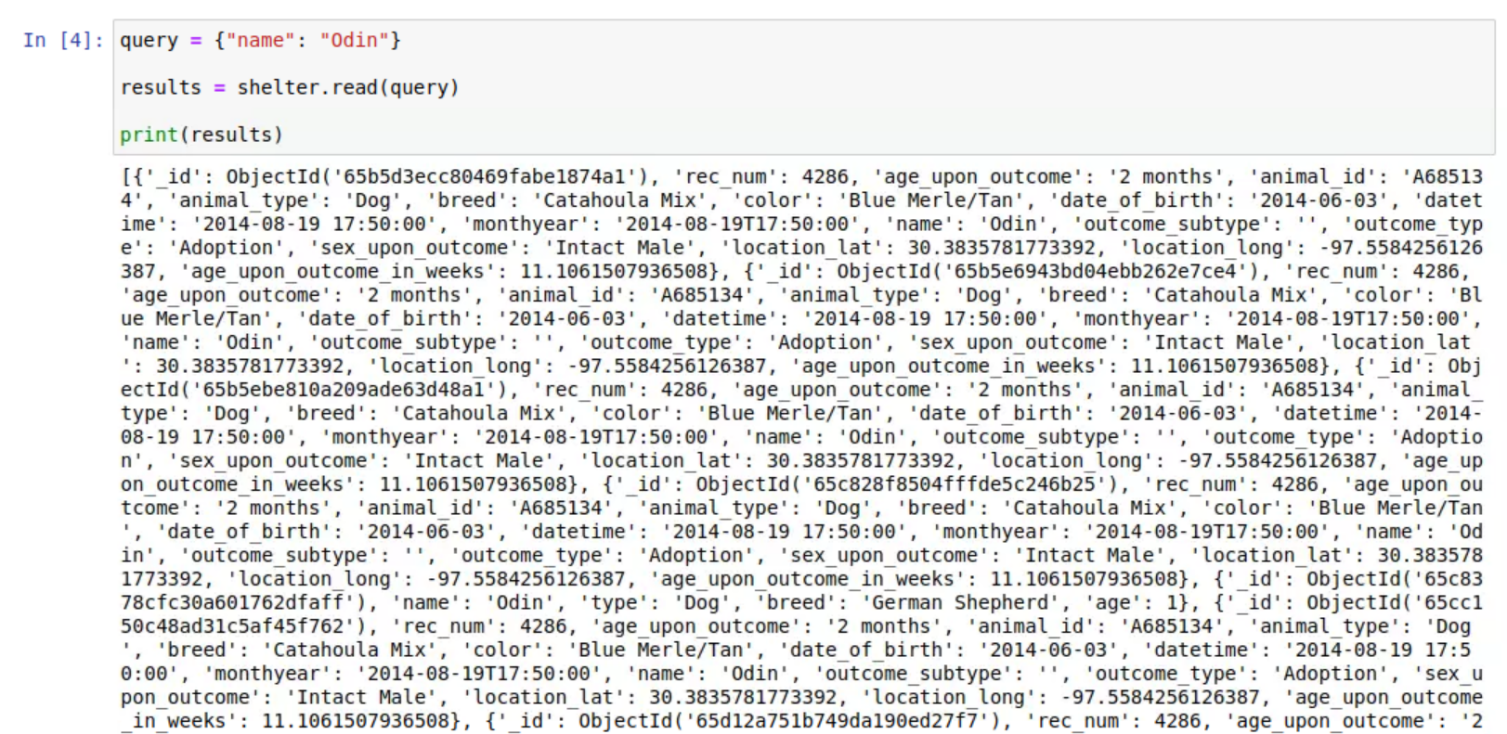
### Screenshots

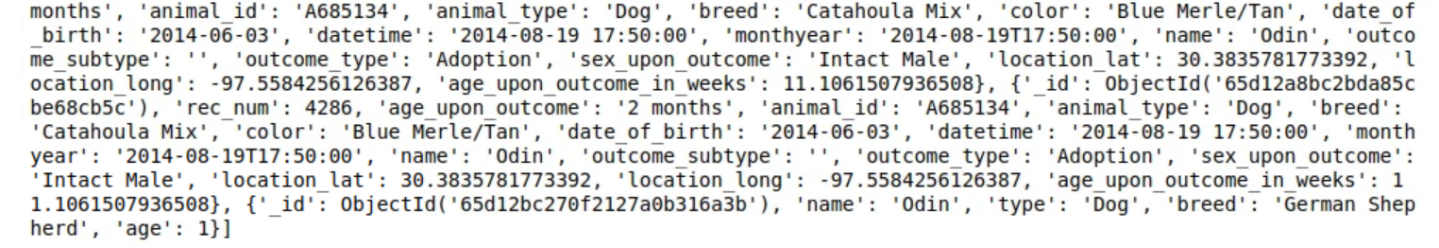


This imports the MongoDB database of the animal shelter data. The important part is to import the CSV file of your animal shelter data.



Here shows an example of importing the AnimalShelter class then making an instance of the AnimalShelter class. Then new animal data is set to the animal\_data variable which is used create a record in the MongoDB database. Since the animal data was added, True is returned. Otherwise if an error occurs False is returned so the users knows the animal data was not added to the database.





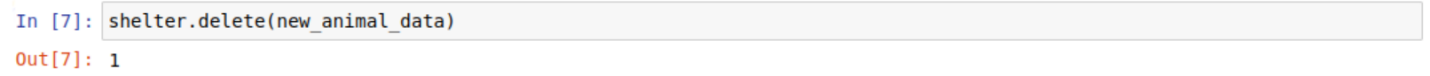
This shows a query for all the animals in the database that have the name Odin. The last line shows the same record that was created above.



This shows the code for updating a record. If a record is updated then the number of records that were updated are returned, in this case it is one.



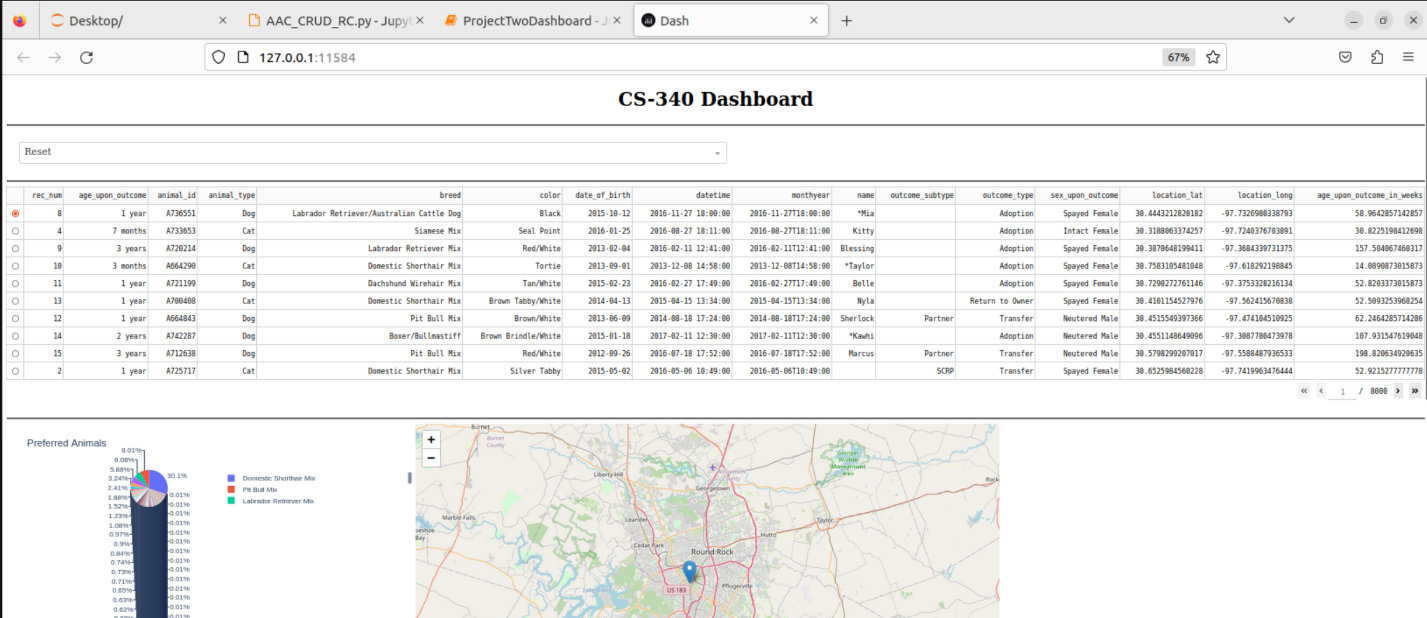
Here is a query that shows the update did take effect.



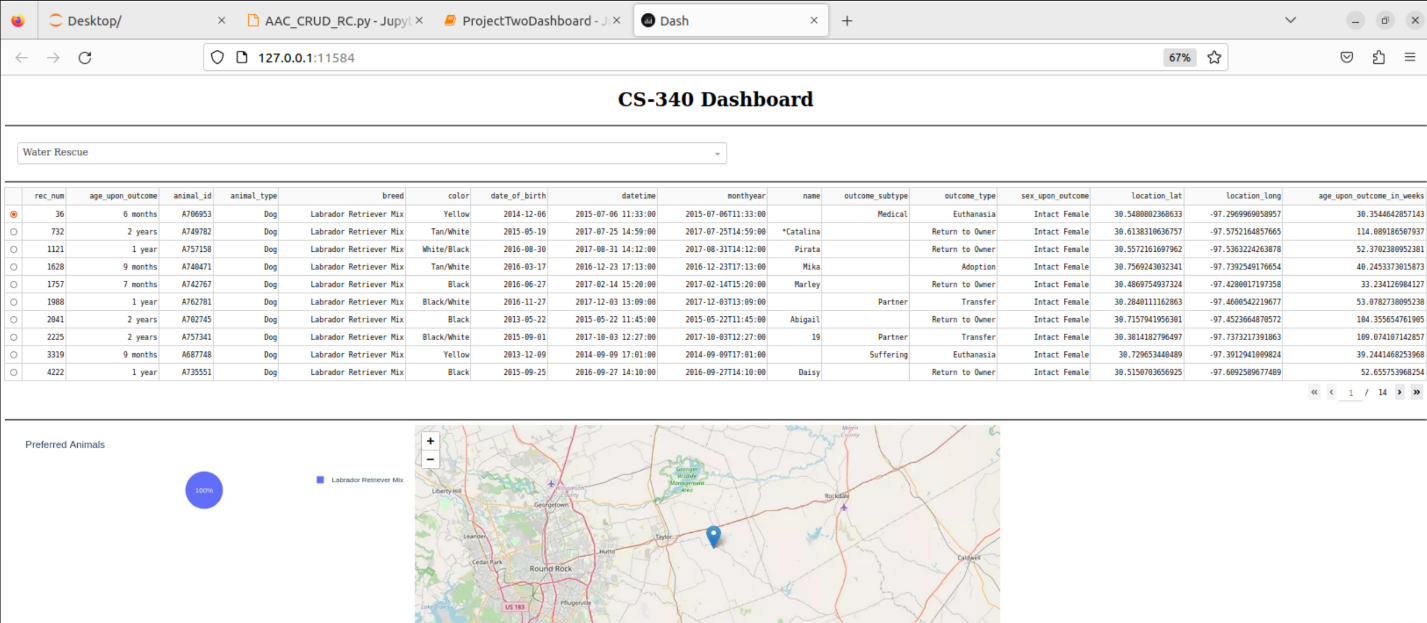
This shows the code for deleting a record. If a record is deleted then the number of records that were deleted are returned, in this case it is one.



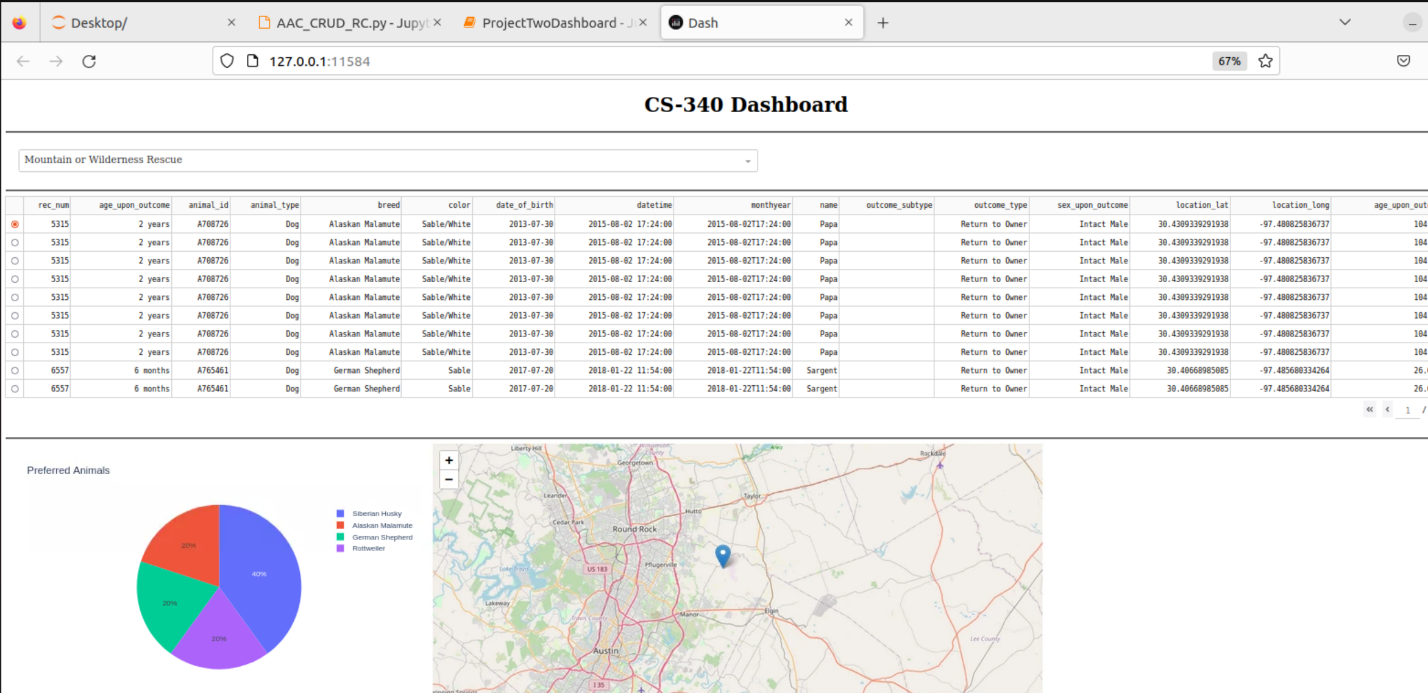
This is the query again that shows the record that we had made is deleted from the database.



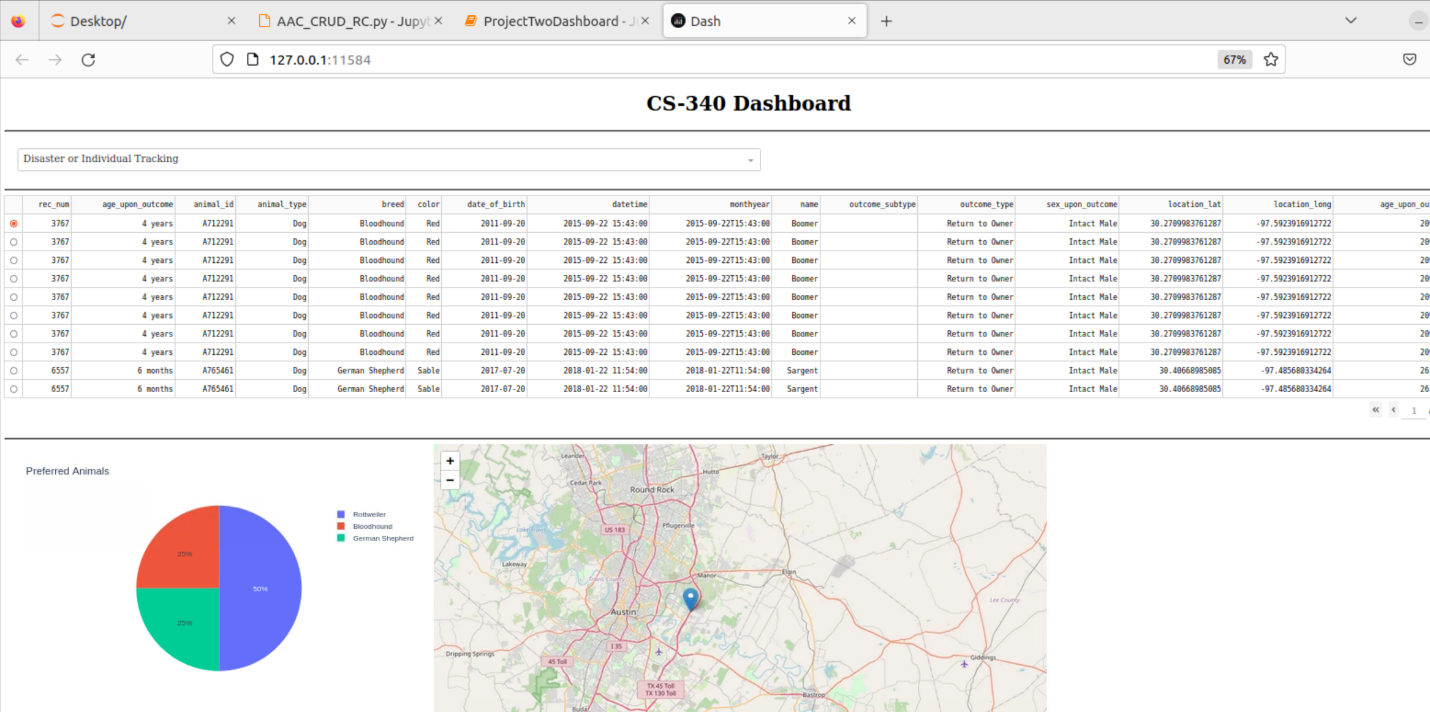
Here is a Dashboard that uses the CRUD Mongo DB python program and the Dash framework.



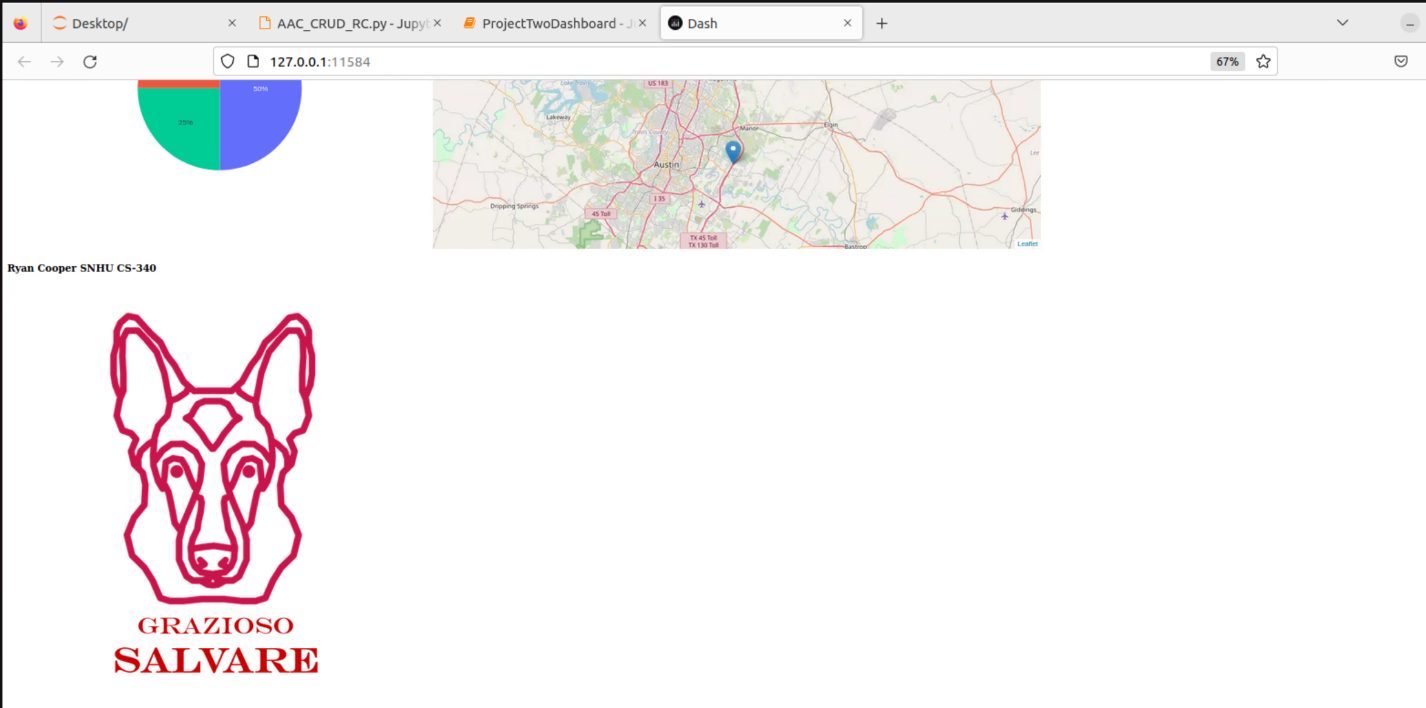
This shows the results of Water Rescue animals, as you can see the data table, pie chart and geolocation map all update as the filter is changed.



This shows the results of Mountain or Wilderness Rescue animals.



This shows the results of Disaster or Individual Tracking Rescue animals.



Here is the bottom of the page with the Grazioso Salvare logo and my name.

## Roadmap/Features

* Adding support for more complex queries.

## Contact

Ryan Cooper