# CS 340 README Template

## About the Project/Project Title

This is the Pet Profiler Project (P3). It was created for the international rescue-animal training company, Grazioso Salvare.

## Motivation

The purpose of P3 is to accumulate data from non-profit animal shelters and use that data to identify good candidates for rescue dog training. By allowing the user to quickly and easily filter based on a variety of characteristics, such as breed, age, and current status, identifying potential trainees will become faster and more accurate.

## Getting Started

In order to use P3, the client dashboard (in development) can be used to connect to and authenticate with the database. The Python middleware interacts directly with the database and provides a simple, easy-to-use API for the dashboard. This allows the execution of CRUD functions from the dashboard without in-depth knowledge of MongoDB syntax and operation.

## Installation

MongoDB is needed to access the database directly.

Python is needed to access the database via the Python middleware API.

## Usage

Python is used to interface with the Mongo Database via the pymongo library. A connection to the database is established and a user account is authenticated which provides access to the animal shelter database. CRUD commands are implemented as methods.

### Code Example

The following is the section of the CRUD Python module handling the creation of documents:

Text

Description automatically generated

The following handles the reading of documents:

A screenshot of a computer

Description automatically generated with medium confidence

### Tests

Testing of the modules was conducted a Jupyter Notebook. After importing the AnimalShelter library and authenticating (Step 1), the CRUD functionality can be tested.

To test the read method, a dictionary was passed which initiated a query, in this case a search for all documents that have a particular breed (Step 2).

If the function was called and no data was passed, an exception would be thrown. If the function finds no documents matching the parameter, a value of False is returned (Step 3).

Graphical user interface, text, application, email

Description automatically generated

To test the create function, a dictionary with several key value pairs was passed as an argument. Upon successful creation of a new document, the method returns a value of True (Step 4).

The read method can be used to confirm that the document was added to the database (Step 5).

Passing an invalid parameter to the create function (in this case an empty dictionary), will cause the method to return a value of False, indicating that no document was created (Step 6). The create method requires a minimum of an animal\_id to be passed in order to create a document.

Graphical user interface, text, application, email

Description automatically generated

The update method requires two arguments to be passed, a dictionary used to query results of documents to be updated, and a dictionary used to hold the key-value pairs to update (Step 7). A successful update will pass back the results from the Mongo database. An update will be applied to any document which matches the query; therefore, care must be taken to make queries specific and avoid changing documents unintentionally.

The updated values can be verified by using the read method on the document that the update was applied to. In this example, the name of the animal was changed (Step 8).

If the query of the update method does not match any documents, nothing will be updated, and the method will return a value of False (Step 9).

Text

Description automatically generated

The final CRUD function is the delete method. To test this function, a dictionary was passed which initiated a query, in this case a search for all documents that have a particular breed (Step 10). All documents that match this query are deleted from the database.

The results can be verified by searching for the same dictionary. In this example, no results are found with a read method, indicating that all results have been deleted from the database (Step 11).

If the function was called and no data was passed, an exception would be thrown. If the function finds no documents matching the parameter, a value of False is returned (Step 12).

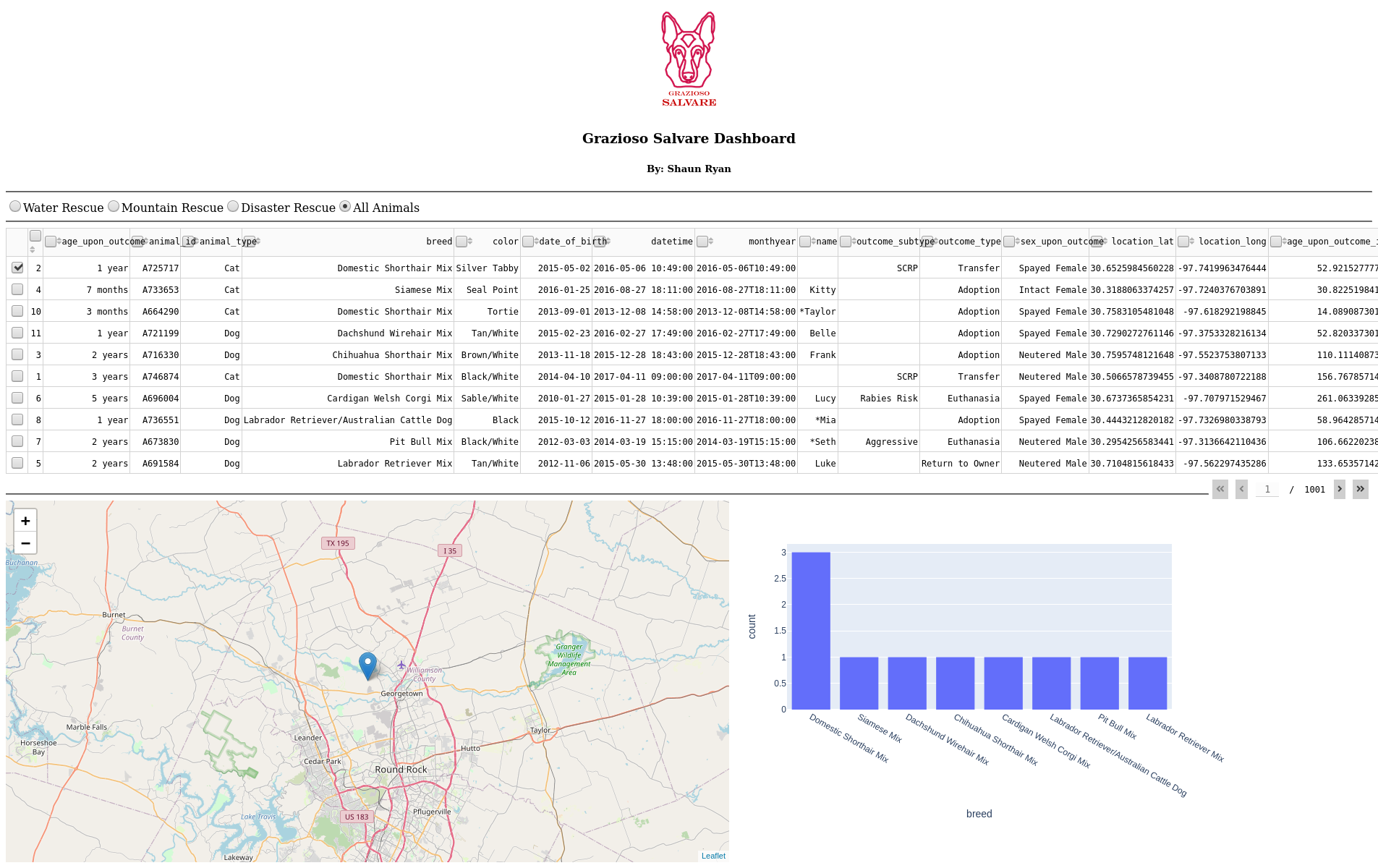
Text

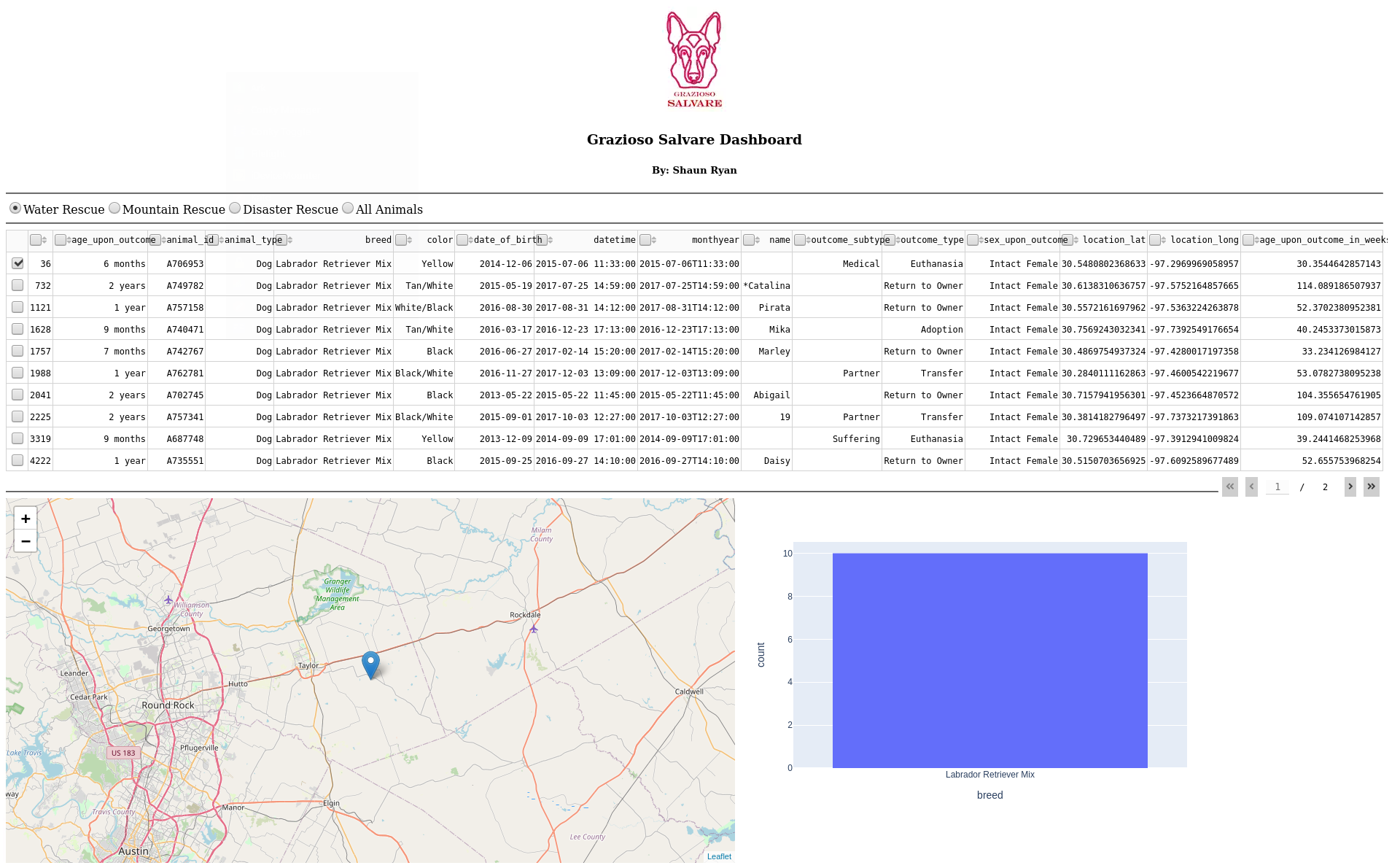
Description automatically generated

### Dashboard

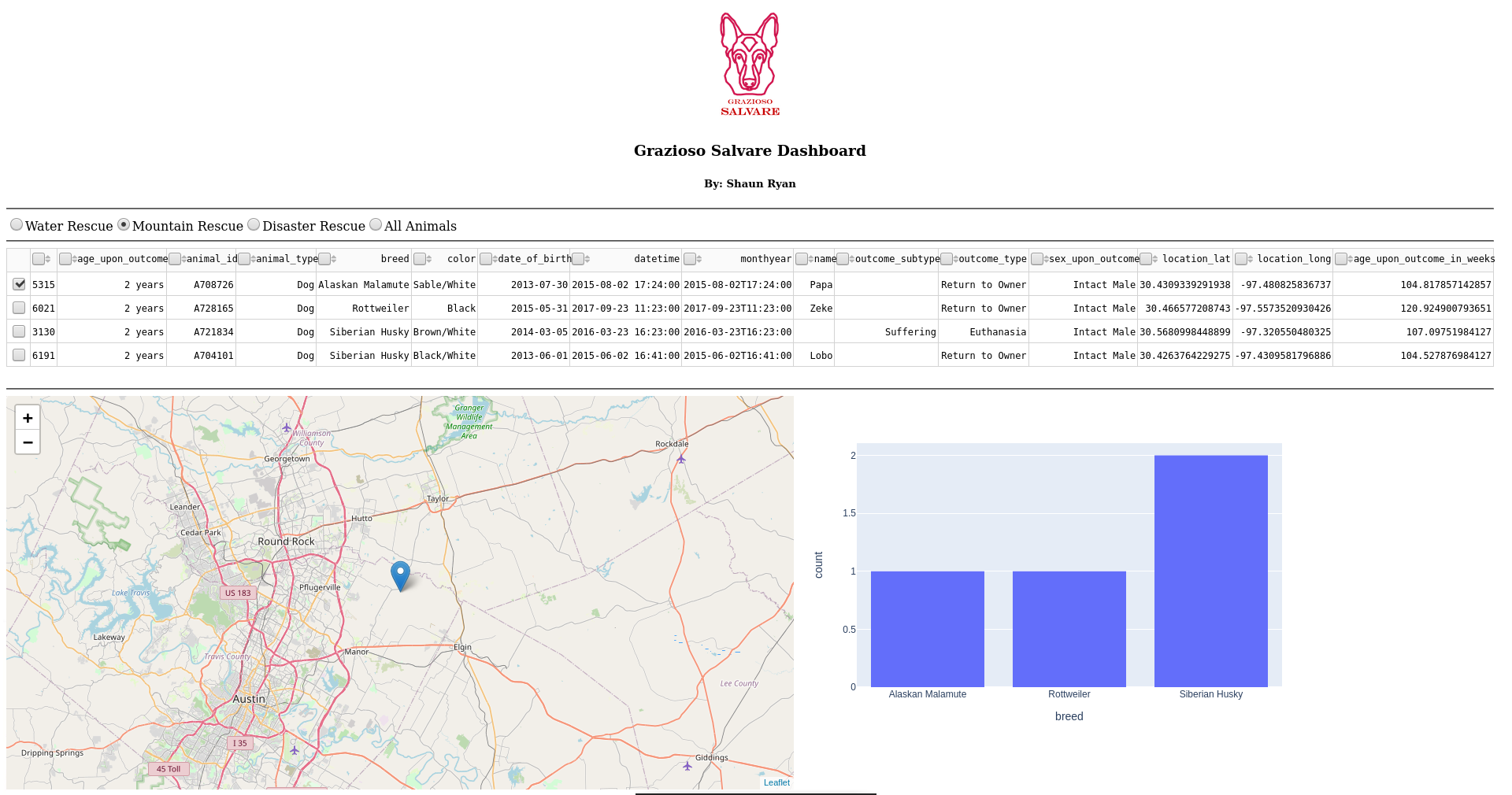
The dashboard was written in Python. A connection was established with the database using the previously developed CRUD module. Data in imported and manipulated using the Pandas software library. This allows data to be stored as a DataFrame object and be used and manipulated easily via a variety of tools. Data is displayed as a table using the Dash framework which allows sorting by column. Filters can be applied to the table in the form of radio buttons. These will run a query on the data frame for specific attributes (animal type, breed, sex, and age) which fit the criteria of the filter. The filters currently available are Water Rescue, Mountain Rescue, Disaster Rescue, and Reset, which removes all filters. In addition to data being displayed in a sort-able table, the Dash framework is used to display the data in two other graphs. The first is a map, which shows a pin on a graphical map for the geographic location of the selected dog. Clicking on the pin will give extra information about the dog, such as breed. The second graph is a histogram, which will show the breeds and the number of each breed returned by the query. This gives the end user an easy to digest idea of what types and numbers of dogs are available in each category. Examples of each filter can be seen here:

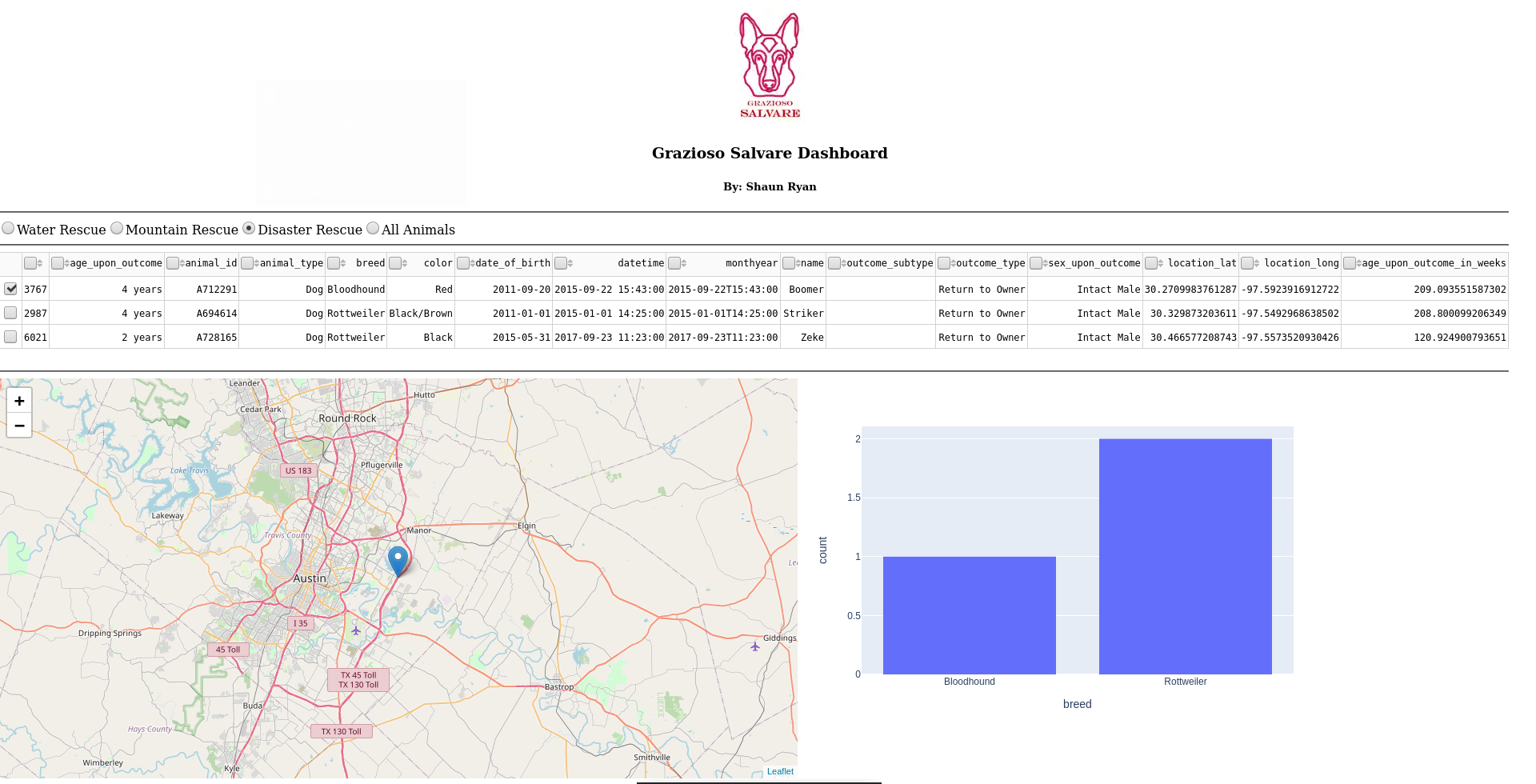
Default value/reset:

Water Rescue:



Mountain Rescue:

 Disaster Rescue:



### Screenshots

Data was initially imported to the database from a CSV file via the mongoimport utility:

Text

Description automatically generated

An administrator account was created to allow full access to the database.Text

Description automatically generatedText

Description automatically generated

A user account named “aacuser” was created which allows access to the AAC database. This user account is used for remote access to the database by the Python middleware.Text

Description automatically generatedText

Description automatically generated

## Contact

Your name:

Shaun Ryan

Shaun.ryan1@snhu.edu