# CS 340 README

## About the Project/Project Title

CRUD module. This python module implements CRUD (create, read, update, delete) functionality for a MongoDB database using the PyMongo library.

## Motivation

The primary purpose of this project is to enable basic functionality (create, read, update, and delete) between a user interface and an associated MongoDB database.

## Getting Started

First, download the python module file and save it in your specified directory. Import the module into your project and it’s ready to use. The initialization method requires a username and password to access the database. It can be changed to access the localhost without a username and password. Creating the read and create portions of the codebase was mainly an exercise in reading the PyMongo documentation and implementing the associated functions.

## Installation

No tools are necessary to use this module other than a working computer and text editor. Two libraries are utilized by the codebase – pymongo and bson.objectid.

## Project Tools

The MongoDB database program was utilized as the backend for this project. In terms of the model, view, controller architecture its primary function was to serve as the model. The model serves as the back-end data structure of the architecture. Therefore, MongoDB is perfect for this application. Dash was utilized for the view and controller components of the application due to its ability to easily construct a user interface and its compatibility with python. Dash also contains useful functions to display and manipulate data.

**Steps to Complete the Project**

1. Set up back-end database using MongoDB

2. Set up the create, read, update, and delete methods in a python class

3. Connect the backend database to the frontend view using the Dash framework

**Challenges**

The main challenge of this project was integrating the various frameworks. Finding and parsing through the documentation was difficult at times. The most difficult documentation to locate was for the dash Map. Ultimately, it took a combination of students to find it and post it on the class forum.

## Usage

### Code Examples

The library consists of one class: 

This class consists of a python init method and a create and read method. See below for more details.

**Init method:**

def \_\_init\_\_(self):

self.client = MongoClient('mongodb://aacuser:apple@localhost:37032/?authSource=AAC')

self.database = self.client['AAC']

**Create method: create new documents for the database**

def create\_one\_document(self, data):

if data is not None:

self.database.animals.insert\_one(data)

return True

else:

raise Exception ("False. Nothing to save, because data parameter is empty.")

**Read method: read documents from the database**

def read(self, data=None):

if data is None:

return self.database.animals.find()

elif data is not None:

return self.database.animals.find(data)

else:

raise Exception("False. Invalid entry.")

**Update method: update one or many documents in the database based on a key value pair**

def update (self, filter\_data, new\_data, isSingle):

if filter\_data and new\_data is not None:

if isSingle:

self.database.animals.update\_one(filter\_data, new\_data)

return True

elif isSingle is False:

self.database.animals.update\_many(filter\_data, new\_data)

return True

else:

raise Exception ("Invalid input. Please try again.")

**Delete method: delete one or many documents from the database based on a key value pair**

def delete (self, data, isSingle=True):

if data is not None:

if isSingle:

self.database.animals.delete\_one(data)

return True

elif not isSingle:

self.database.animals.delete\_many(data)

return True

else:

raise Exception ("Value does not exist.")

**See below for an example of the data is set up in the database:**

**Document schema:**

{'1': , 'age\_upon\_outcome': , 'animal\_id': , 'animal\_type': , 'breed': , 'color': , 'date\_of\_birth': , 'datetime': ,

'monthyear': , 'name': '', 'outcome\_subtype': , 'outcome\_type': , 'sex\_upon\_outcome': , 'location\_lat': ,

'location\_long': , 'age\_upon\_outcome\_in\_weeks': }

**Sample data:**

{'\_id': ObjectId('63c5d4a946805ca89a0c9e8e'), '1': 1, 'age\_upon\_outcome': '3 years', 'animal\_id': 'A746874',

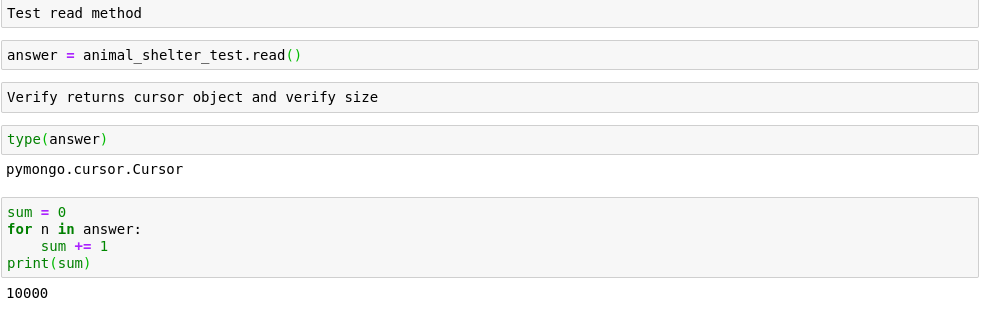
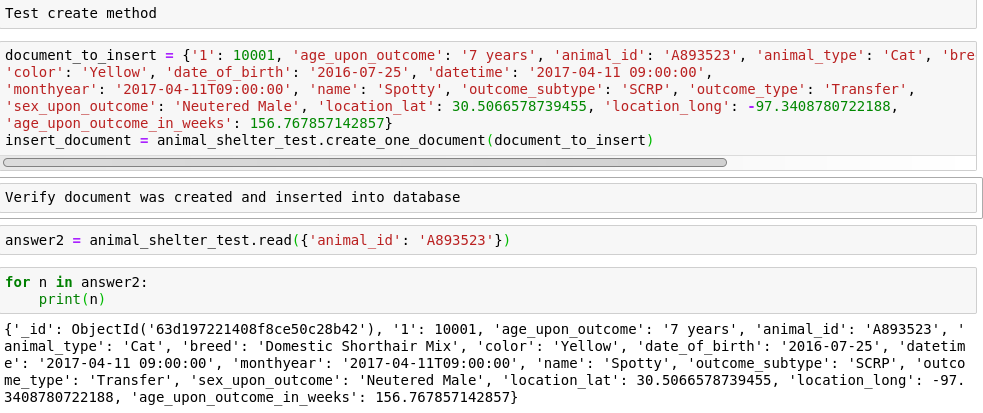
'animal\_type': 'Cat', 'breed': 'Domestic Shorthair Mix', 'color': 'Black/White', 'date\_of\_birth': '2014-04-10',

'datetime': '2017-04-11 09:00:00', 'monthyear': '2017-04-11T09:00:00', 'name': '', 'outcome\_subtype': 'SCRP',

'outcome\_type': 'Transfer', 'sex\_upon\_outcome': 'Neutered Male', 'location\_lat': 30.5066578739455,

'location\_long': -97.3408780722188, 'age\_upon\_outcome\_in\_weeks': 156.767857142857}

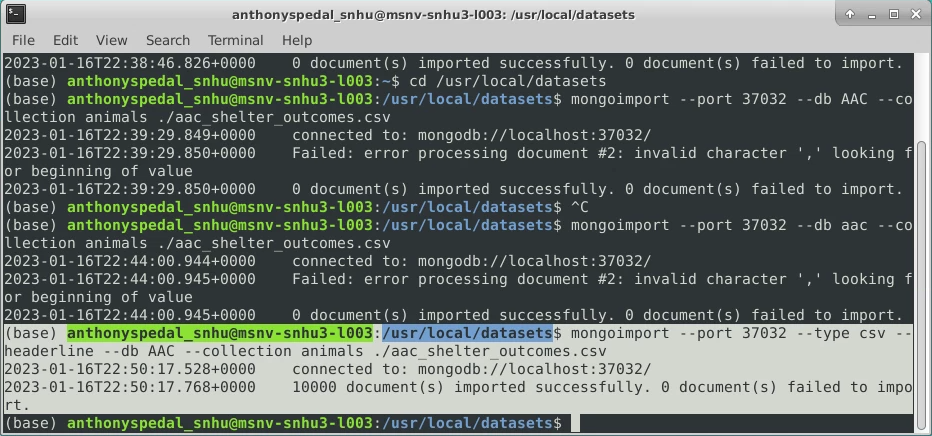
### Tests and Screenshots



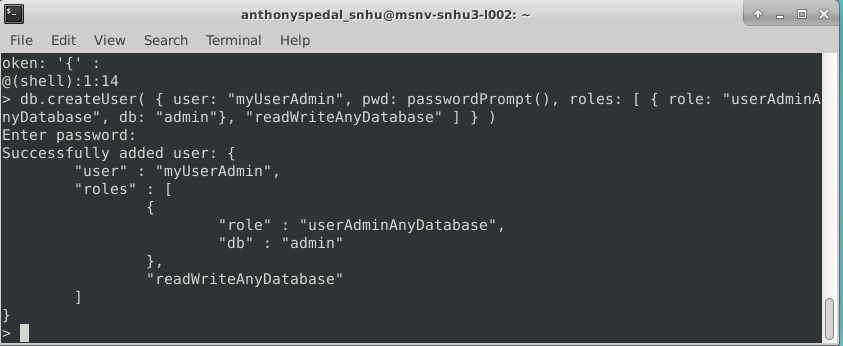
**Import dataset**

**Prompt 1 – Import dataset command and execution**

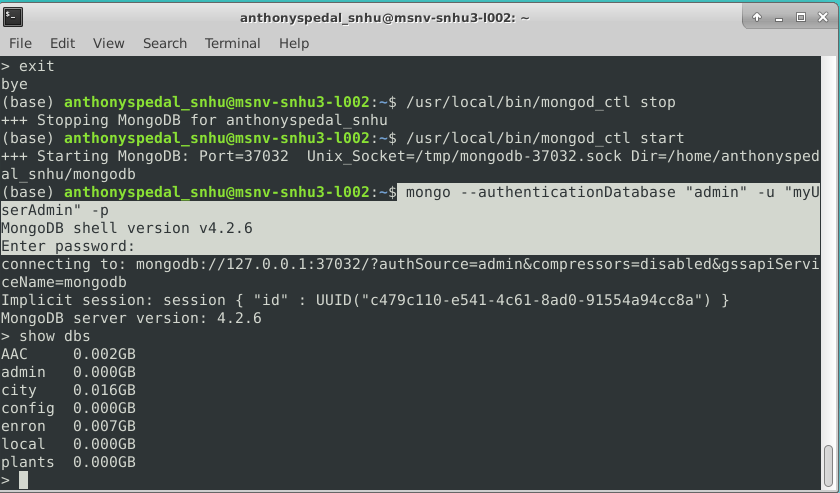


**Admin and User account creation screenshots**

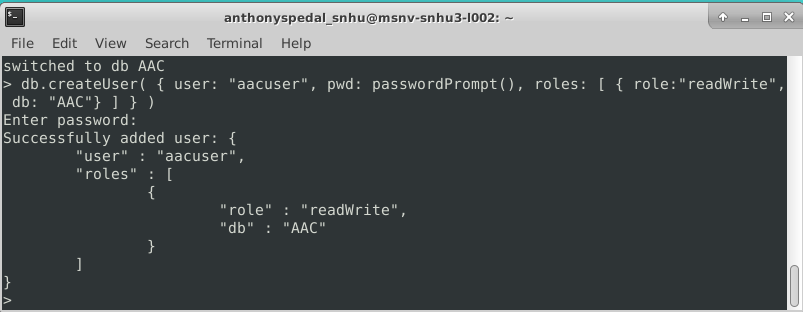
**Prompt 1 – Create admin account**

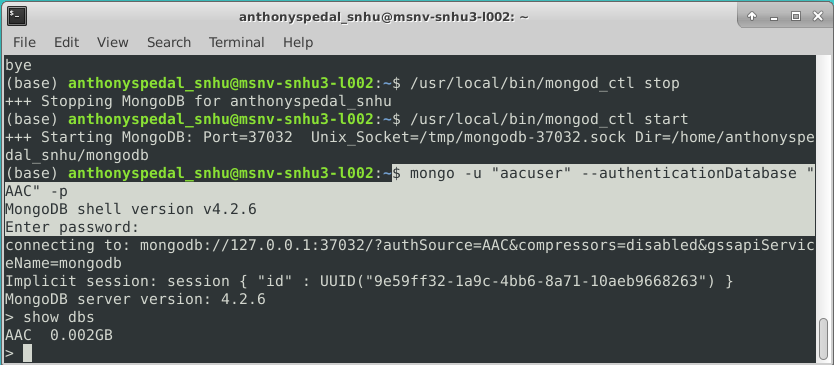


**Prompt 2 – verify admin account set up**

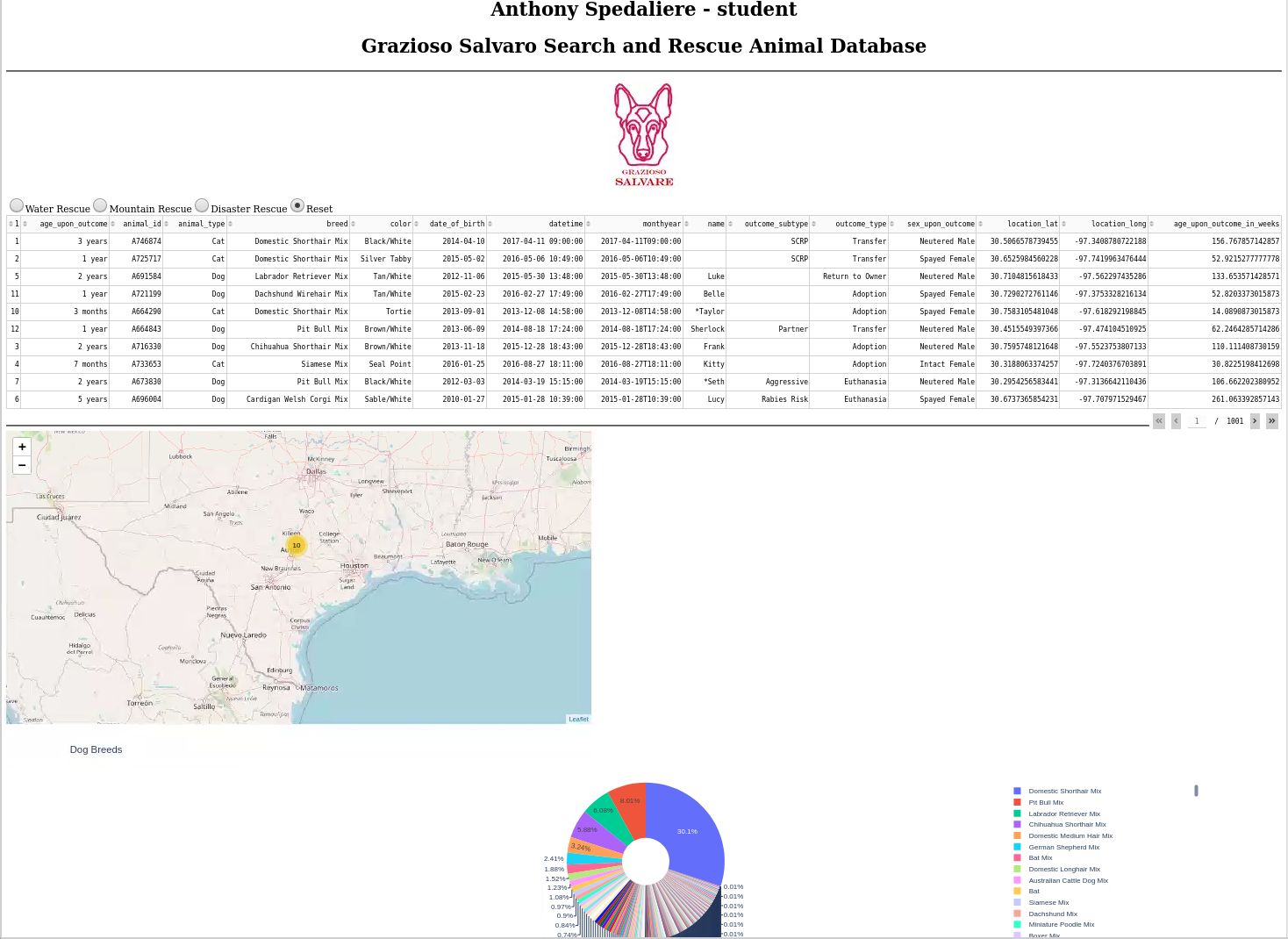


**Prompt 3a – create a new user account called “aacuser”**

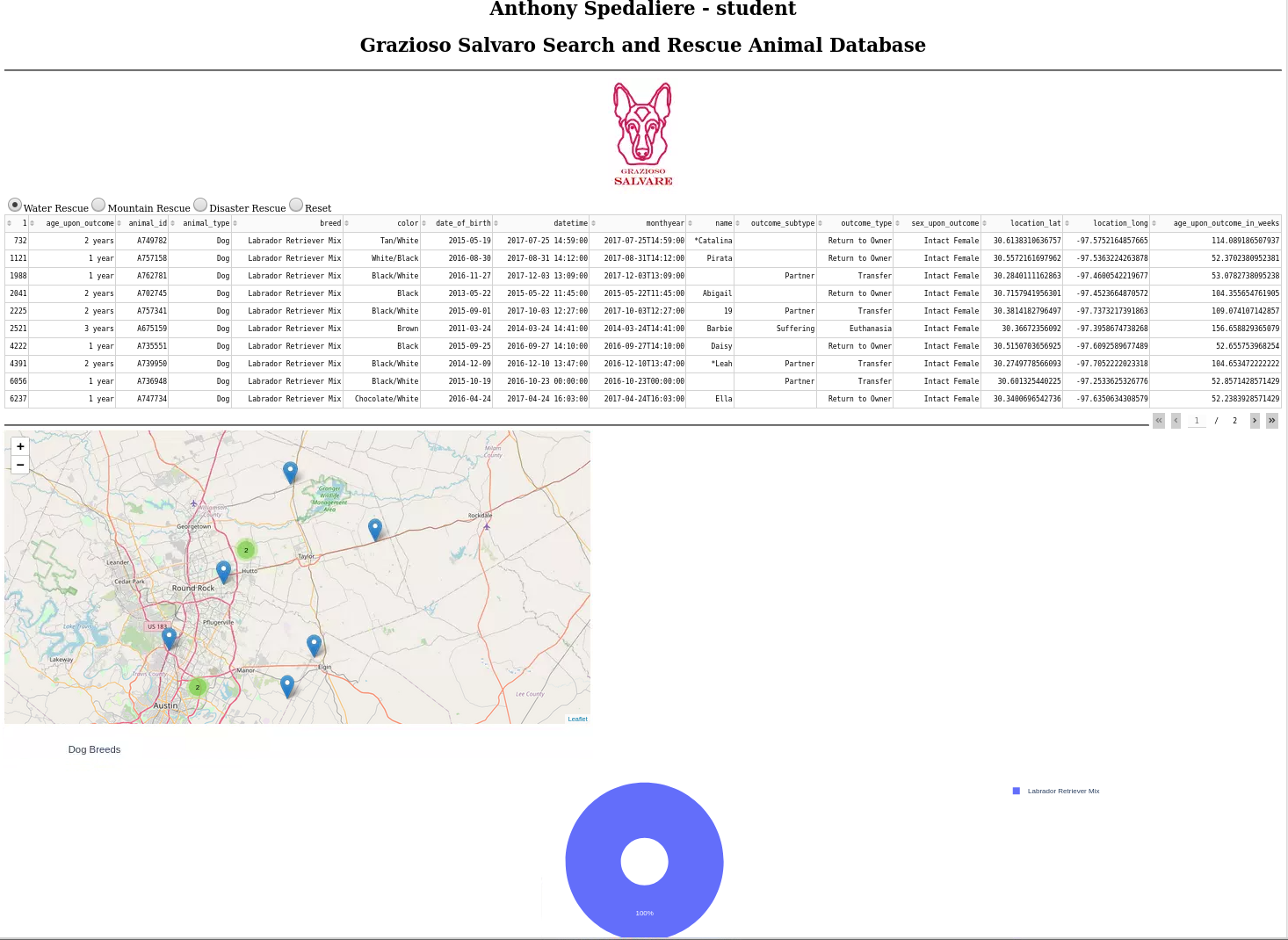


**Prompt 3b – login with new user account and verify database shows** 

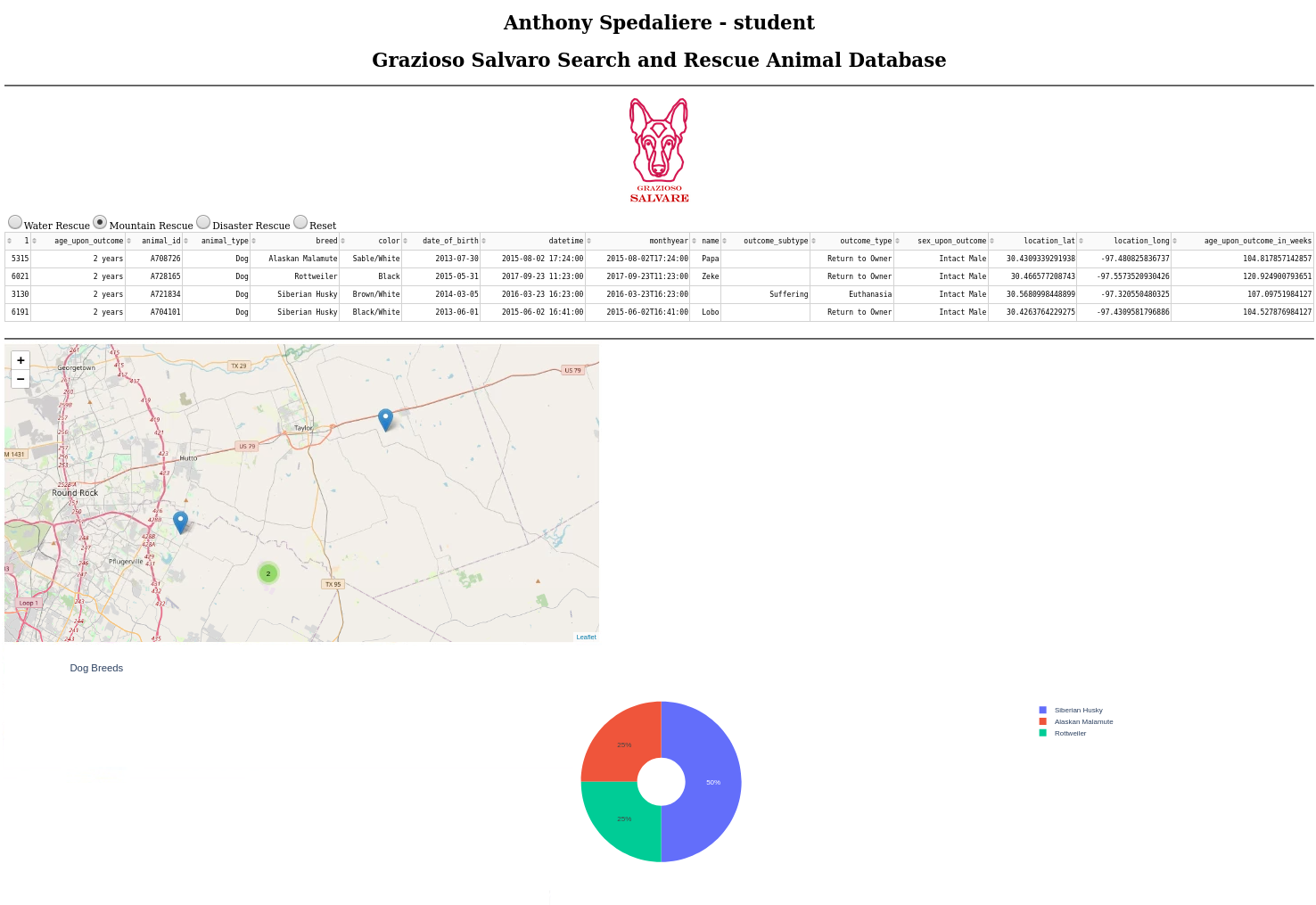
**Prompt 1 – Starting State**



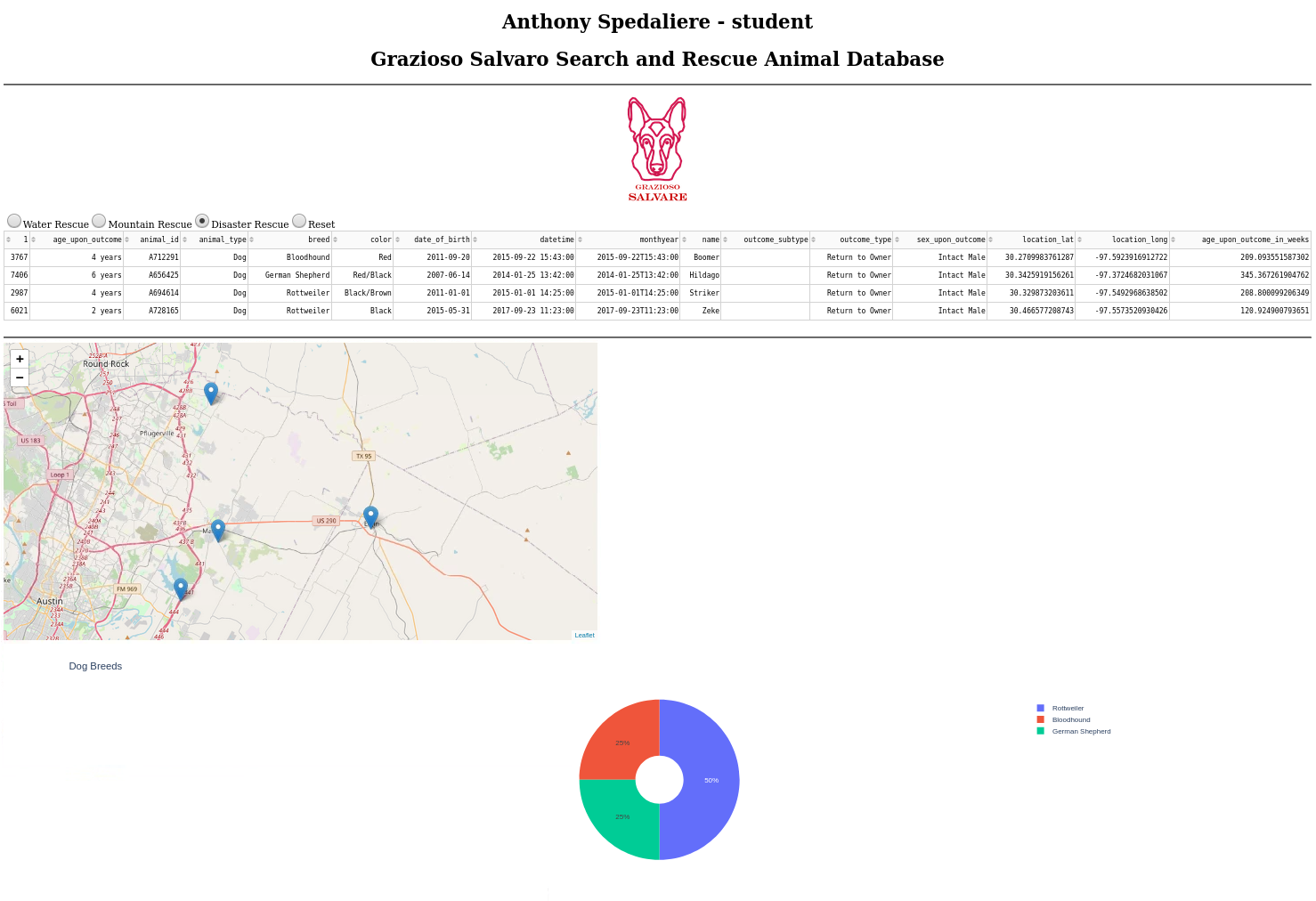
**Prompt 2 – Water Rescue data filter**



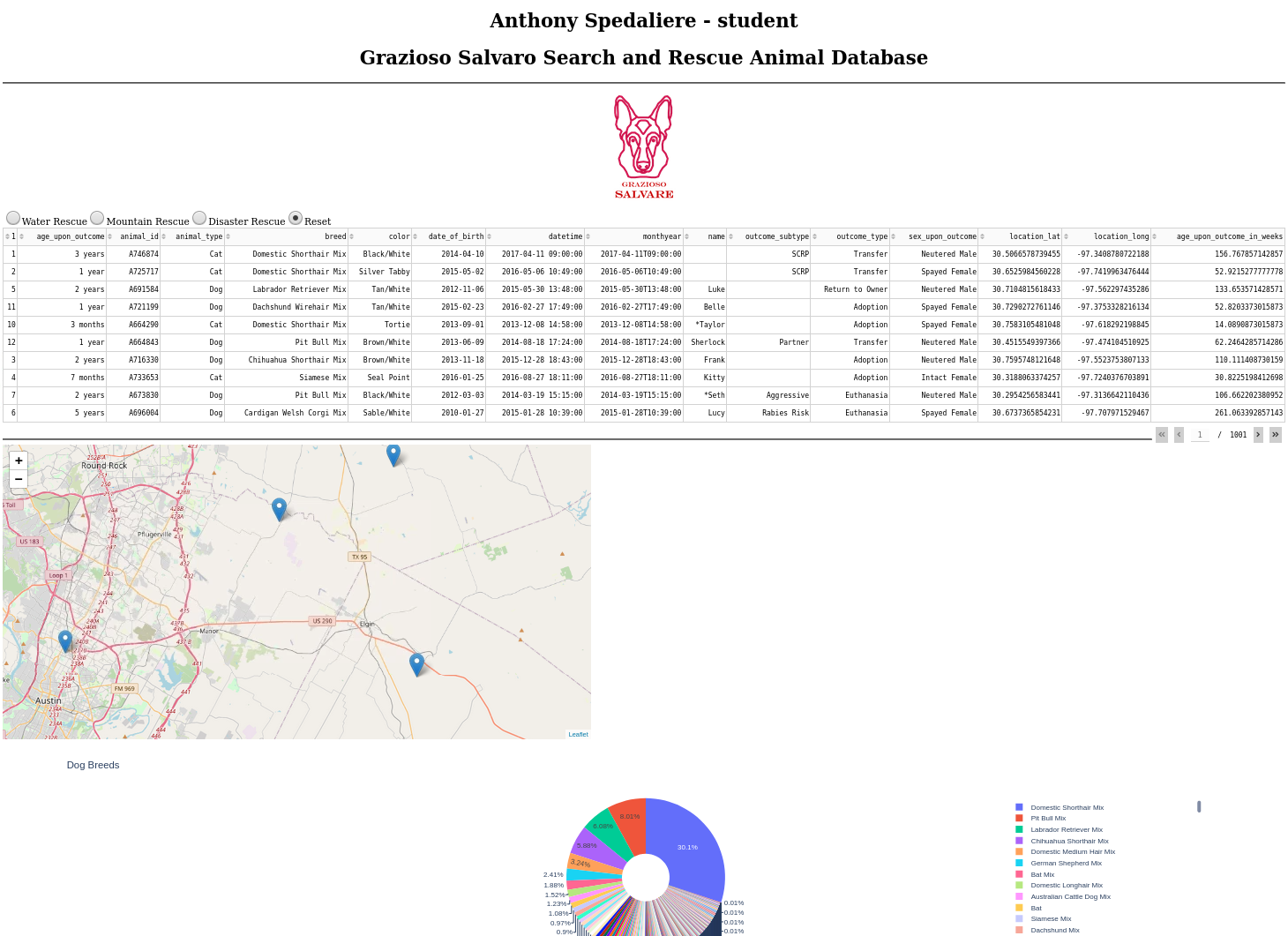
**Prompt 3 – Mountain Rescue data filter**



**Prompt 4 – Disaster Rescue data filter**



**Prompt 5 – Reset data filter**



## Contact

Your name: Anthony Spedaliere – anthony.spedaliere@snhu.edu