# CS 340 README

# by: Steven Rodas

## About the Project/Project Title

*CRUD Python*

This project was developed to create functionality to create and read data, entered as queries, into a specified database and collection. This project uses Python and MongoDB to simplify the process of creating, reading, updating, and deleting (CRUD) data queries.

## Motivation

The motivation for this project was to simplify the process of creating, reading, updating, and deleting (CRUD) data queries. In an increasingly expanding world, simplifying data management is an integral part of developing secure and scalable software.

## Getting Started

*This is an example of how you may give instructions on setting up your project locally: “To get a local copy up and running, follow these simple example steps.”*

In order to get this program running on your machine, you must first install Python and MongoDB to your device if you have not already. Here is the step-by-step guide to getting started:

1. After installing the latest versions of Python and MongoDB to your machine, open the terminal.
2. In the terminal, access the /usr/local/datasets directory using “cd”.
3. Once in the directory, use “mongoimport” to import the aac\_shelter\_outcomes.csv file. Be sure to include the correct authentication variables needed to access the databases.
4. Run module4.py

## Installation

*Tools needed:*

1. Python
2. MongoDB
3. Linux Terminal
4. pymongo
5. MongoClient

## Usage

*Use this space to show useful examples of how your project works and how it can be used. Be sure to include examples of your code, tests, and screenshots.*

### Code Example

*mainCRUD.py*

The following will be code snippets that showcase how exactly the create and read methods work underneath the hood.

The following methods are the create and read methods

def create(self, data):

if data is not None:

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

else:

raise Exception(“Nothing to save, because data parameter is empty”)

def read(self, data):

if data is not None:

self.database.animals.find(data)

else:

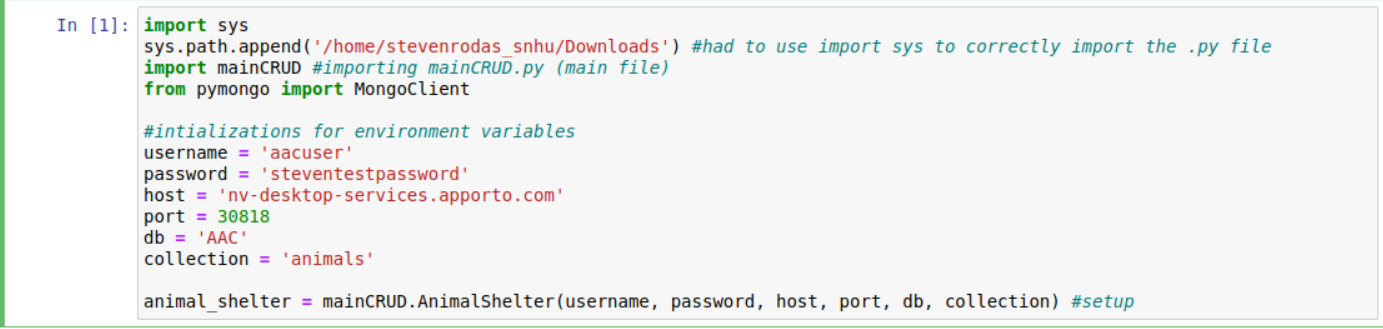
raise Exception(“Nothing to read, data is empty”)

### Tests

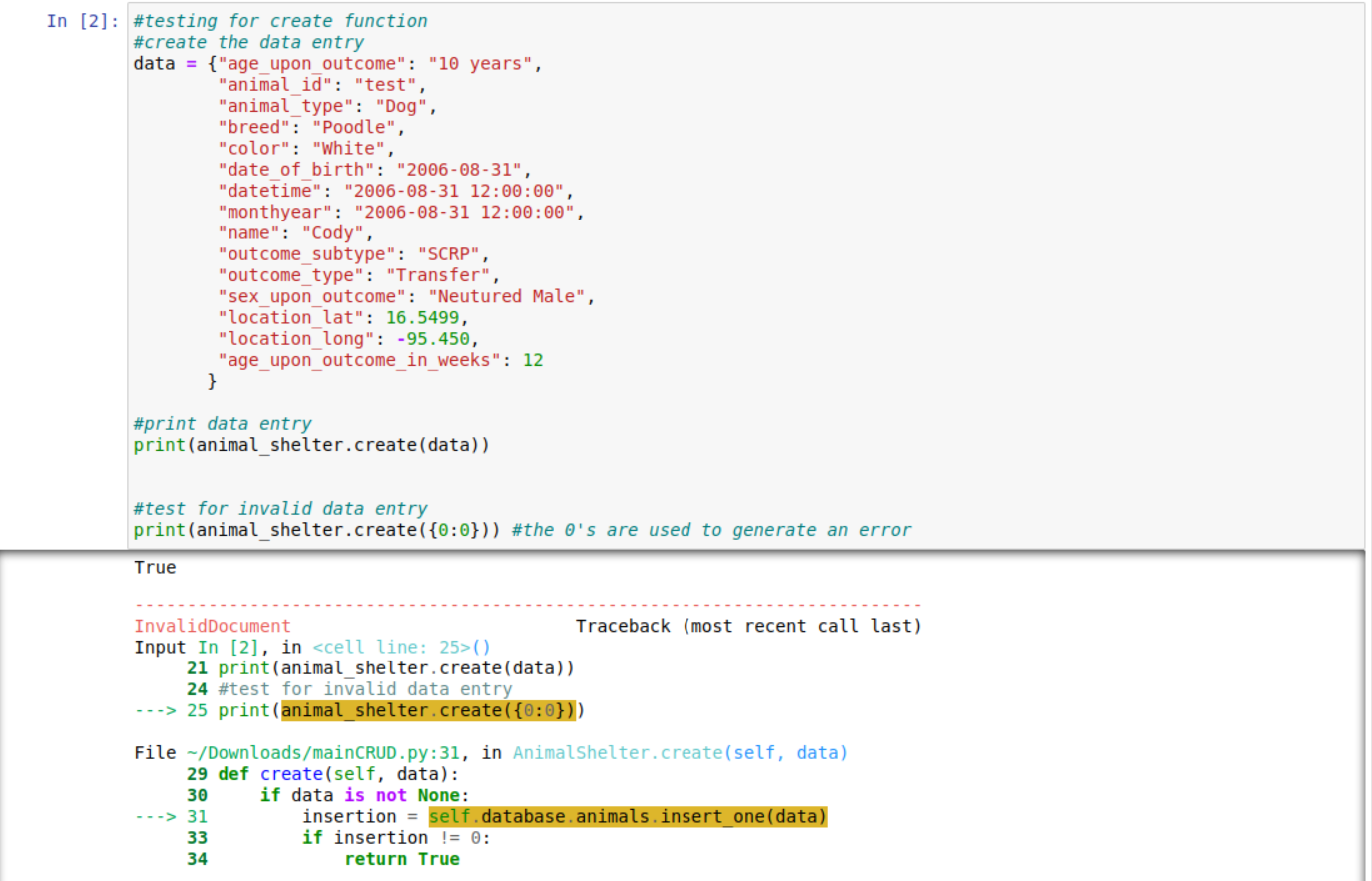
For testing, I tested every CRUD function in the main Python file. To ensure the functions were working correctly, I created a testing module that tested each in every CRUD function, such as create, read, update, and delete. The following will be screenshots and examples.

### Screenshots

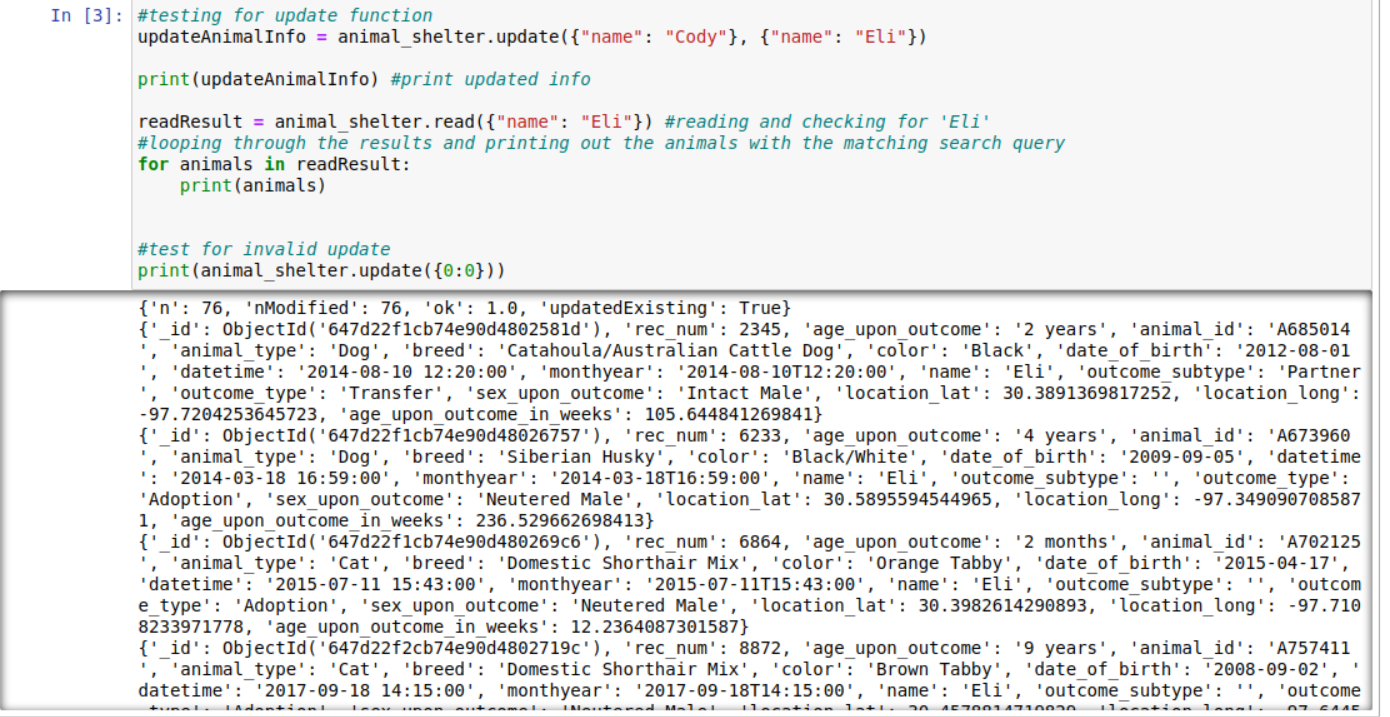
*module4testing.ipynb:*

**

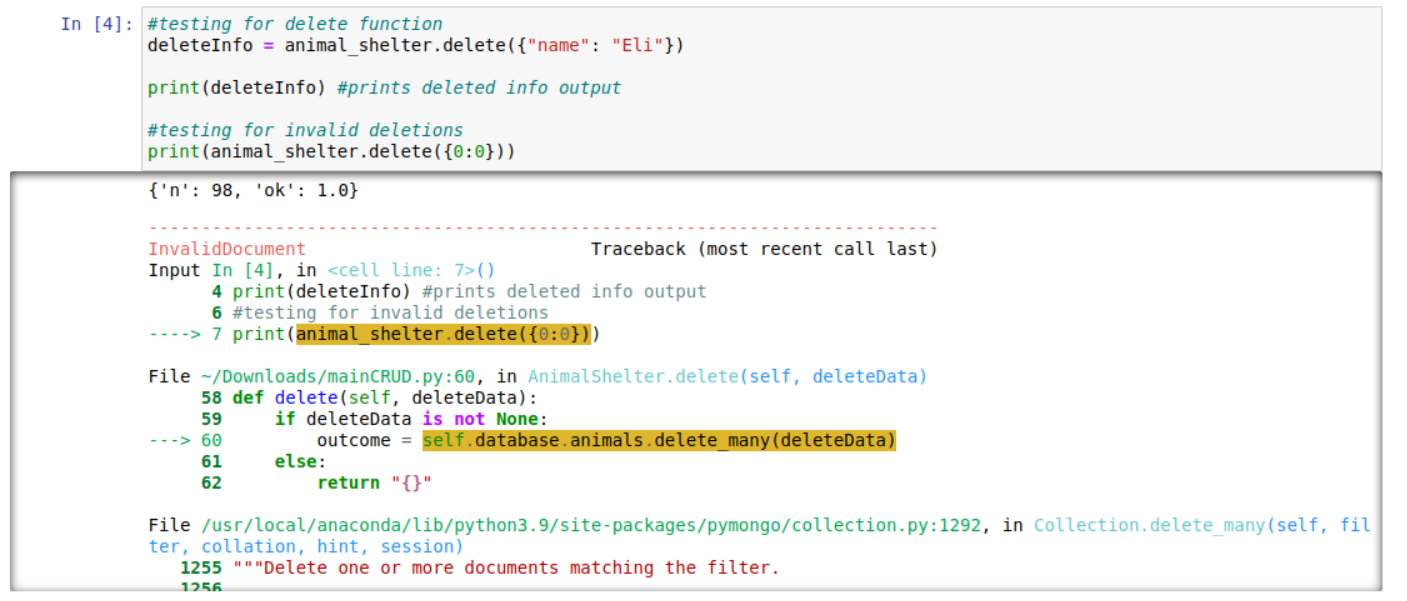
In the module4testing.ipynb module, I began by setting up the environment variables to ensure the tests ran under the correct user, permissions, collection, and database. I had initial problems importing the file normally so I found a workaround using “import sys” to append the correct path for the mainCRUD.py folder. After that, I initialized all the environment variables to their intended values and initialized “animal\_shelter” with those parameters.

**

The testing above is for testing the “create()” function in mainCRUD.py. After initializing “data” with the information we want to use to create a new database entry. After that, I use a print statement to show if the creation process was completed using a boolean value. As shown above, the boolean returned true. Underneath, I used a print statement with 0’s as the parameter to .create() to generate a user error. In the bottom part of the screenshot we can see the ‘InvalidDocument’ traceback error displaying meaning the create function testing has succeeded.

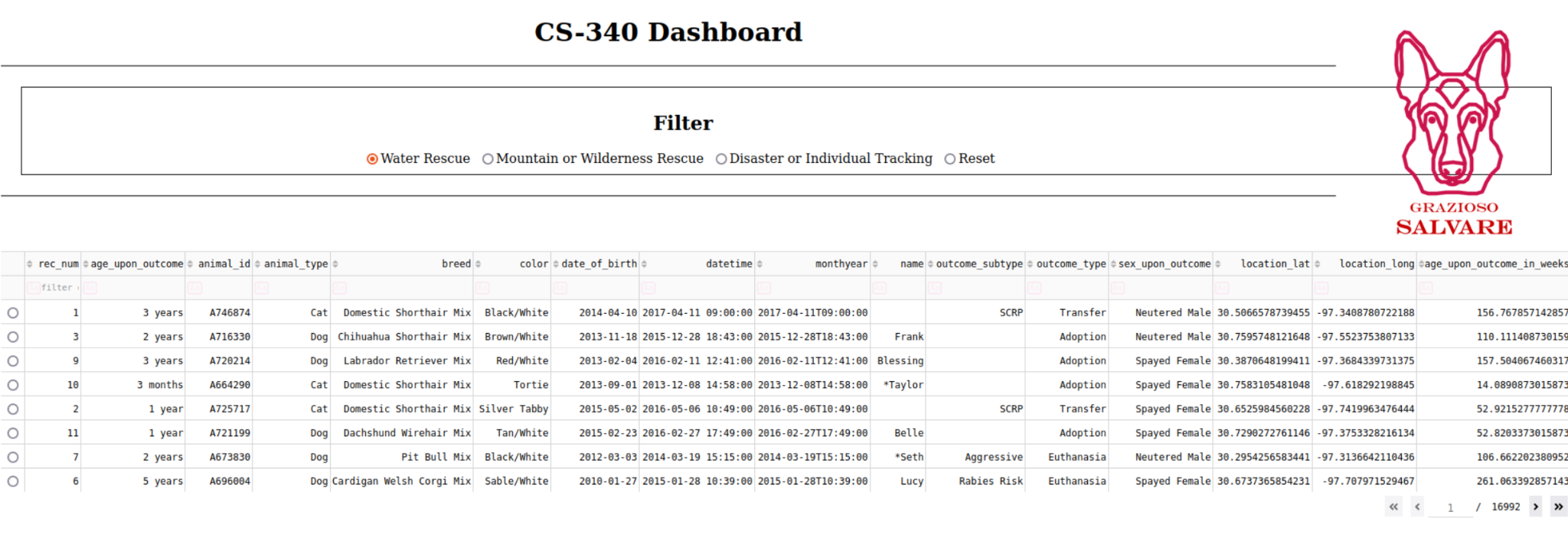


In order to test the read() and update() functions, I initialized “updateAnimalInfo” with the name we want to replace within the database. In this instance, we searched for the name “Cody” and want to update the name to “Eli.” At the same time, I test the read() function by reading through the information, finding the search query, and then looping through the results to print out all the animals in the “readResult”.

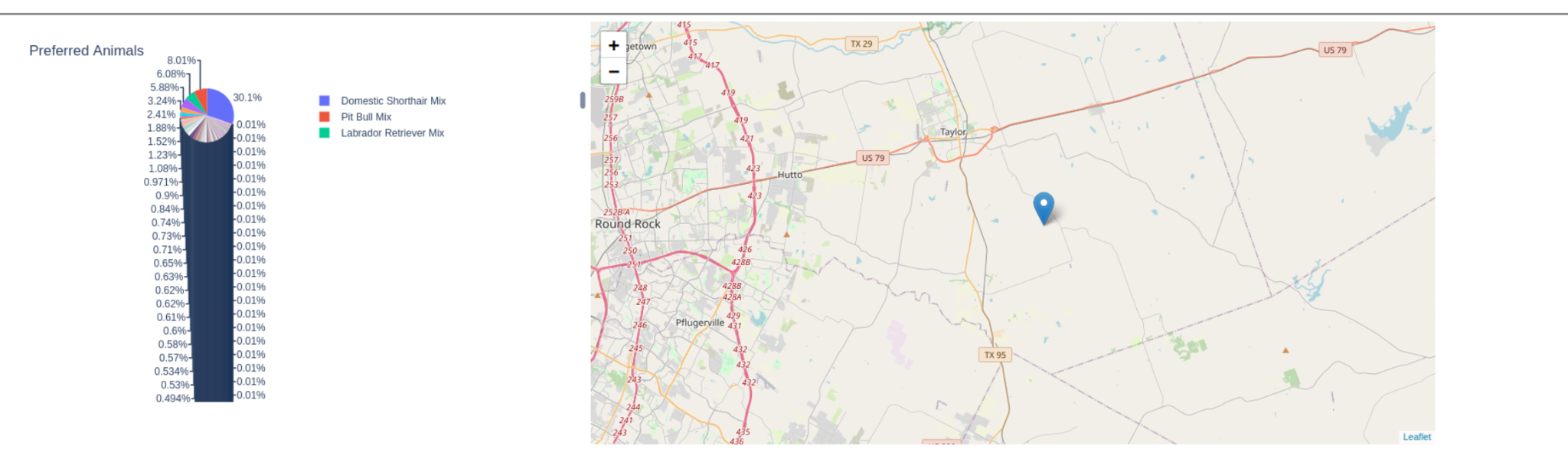


And finally, testing the delete() function. To do this, I used .delete() to delete any query with the matching name of “Eli”. After this, I print out the result confirming that the query was indeed deleted from the database. In the end, I check once again for an invalid deletion and display the results.

Final results from Grazioso’s dashboard:



As seen above, this is the Grazioso Salvare dashboard. This is the UI that enables Grazioso Salvare to filter through the database and look for animals that fit their search query, such as water rescue animals, mountain/wilderness rescue, disaster rescue, or more. Above the data table, there is a filter with radio buttons that enables the user to filter through the database based on their specific search criteria.



Once filtered, the dashboard will display valuable information, such as a pie chart and a map of the animals available based on their specific search query.

## Contact

Your name: Steven Rodas

E-mail: [steven.rodas@snhu.edu](mailto:steven.rodas@snhu.edu)