# CS 340 README – Grazioso Salvare

## About the Project/Grazioso Salvare - Rescue Animal Training

Identifying dogs as candidates for search and rescue training is the primary goal of the Grazioso Salvare Rescue Animal Training software application being developed by Global Rain. The important work these canines will perform includes finding and helping to rescue humans and other animals in life-threatening conditions. The software will assist in identifying suitable dogs for training from data collected by a non-profit agency operating five animal shelters in Austin, Texas.

## Motivation

It is of dire importance to find dogs that are well suited to perform the life-saving actions rescue training requires. Therefore, it is critical to find dogs with very specific characteristics. For example, dogs that are no more than 2 years old, certain breeds of dogs are more proficient in specialized types of rescue settings such as, water, mountain, or wilderness rescue, scent tracking, or locating natural disaster victims. The Gracioso Salvare software will provide this information to ease the selection process.

## Getting Started

1. To get started, start your Apporto Environment through the BrightSpace *Virtual Lab Access* link.
2. Open a terminal window and start mongo with the following command:

/usr/local/bin/mongod\_ctl start-noauth

1. Navigate to the proper directory:

cd /usr/local/datasets

1. Import the “aac\_shelter\_outcomes.csv” file with command below:

mongoimport --port 53168 --db AAC --collection animals --type csv --headerline ./aac\_shelter\_outcomes.csv

1. Finally, start mongo:

#start mongo without authentication  
/usr/local/bin/mongod\_ctl start-noauth

(If authentication and other user accounts are needed, please follow the screenshots section for more information on how to enable access control).

Obstacles encountered:

When creating a password, try to avoid using certain symbols such as “@”. I had a problem authenticating and accessing the database due to the weird interaction caused by this symbol in the password.

## Installation and Tools

Apport Linux Environment – SNHU’s virtual computer lab in which all necessary applications for this project are installed, including Jupyter Notebook, Python 3.6 compiler, and Pymongo.

Jypyter Notebook – Server-client platform in which our python code will run to make changes to the MongoDB database.

Python 3.6 Compiler – Installed to add python functionality to our database since Jupyter Notebook does not come with Python included.

MongoDB – Database platform with flexible tools necessary to implement the CRUD functions and the client-server database requirements for this project.

Pymongo – A Python distribution that allows communication with a MongoDB server, including creating, reading, updating, and deleting data.

## Usage

*Methods:*

* create(<dictionary> data): Inserts data of dictionary type and returns True if successful, else returns False.
* read(<dictionaty> data): Finds documents that include the value/pair searchValue and returns its cursor, else returns an error.
* update(<dictionary> data) Finds value/pair searchValue and modifies to the updateValue and returns it in JSON format, else return an error.
* delete(<dictionary> data) Deletes all value/pair deleteValue found, else returns error message.

*Dash:*

Using the CRUD methods in the AnimalShelter class, we can filter through results in the MongDB database with the options below:

- Water Rescue.

- Mountain Wilderness Rescue.

- Disaster Rescue and Individual Tracking.

- Reset.

Each option displays dogs suited for a specific rescue mission in accordance with the parameters determined by Grazioso Salvare. Additional queries can be added for added functionality and interaction with the database.

### Code Example

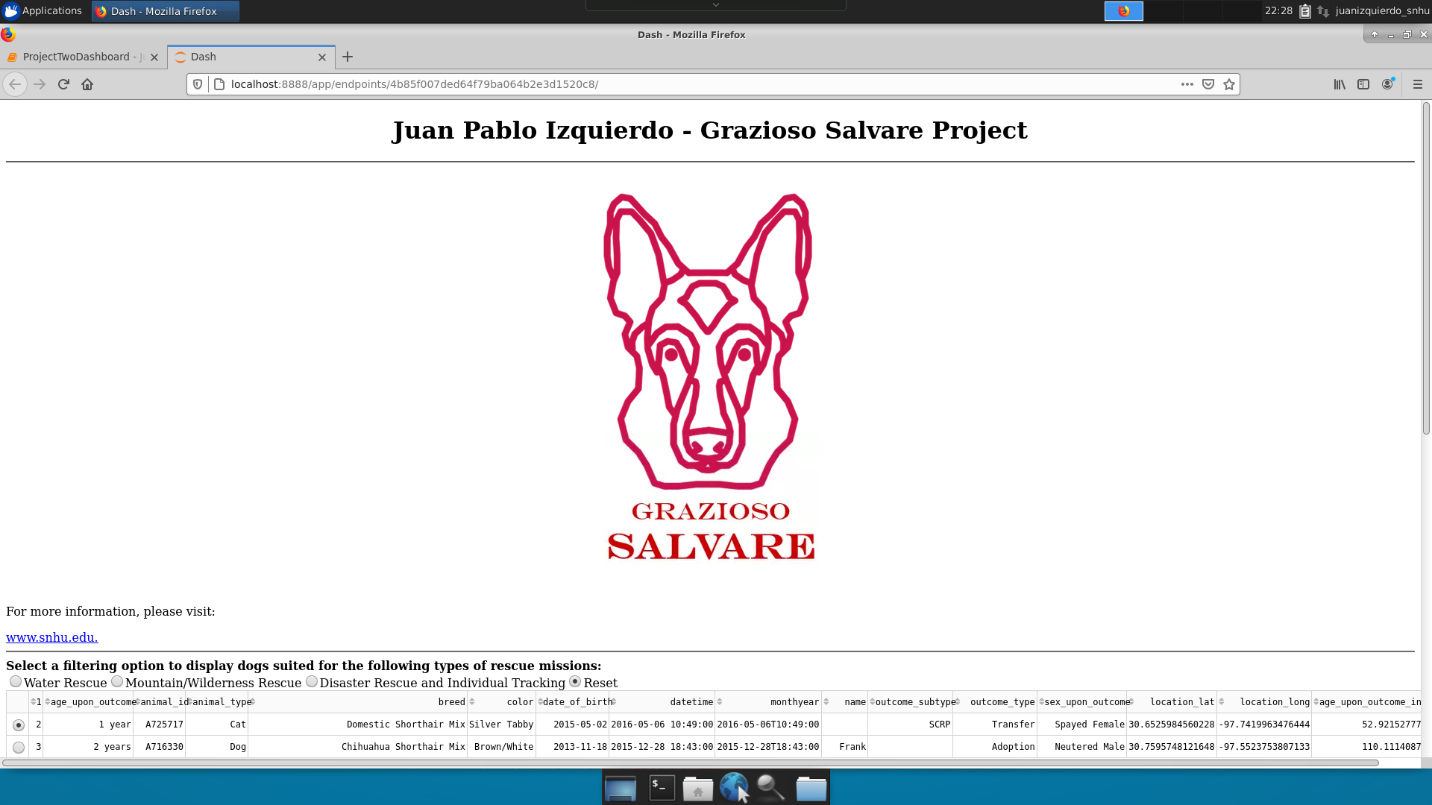
*animalShelter.py* code below:

from pymongo import MongoClient  
from bson.objectid import ObjectId  
from bson.json\_util import dumps  
  
  
class AnimalShelter(object):  
 *""" CRUD operations for Animal collection in MongoDB """* def \_\_init\_\_(self, username, password):  
 # Initializing the MongoClient. This helps to access the MongoDB databases and collections.  
 if username and password:  
 print("username and password are: ", username, password)  
 self.client = MongoClient('mongodb://%s:%s@localhost:53168/AAC' % (username, password))  
 # where xxxx is your unique port number  
 self.database = self.client['AAC']  
 print("Connection was successful\n")  
 else:  
 print("username and/or password are empty or null.")  
  
 # Create method to implement the C in CRUD.  
 # Inserts data of dictionary type and returns True if successful, else False.  
 def create(self, data):  
 if data is not None and dict:  
 self.database.animals.insert\_one(data) # data should be dictionary  
 print("New data inserted: \n", data, "\n")  
 return "True"  
 else:  
 return "False"  
  
 # Read method to implement the R in CRUD.  
 # Finds documents that include the value/pair searchValue and returns its cursor, else returns an error.  
 def read(self, searchValue):  
 if searchValue is not None and dict:  
 searchResult = self.database.animals.find(searchValue)  
 if len(list(searchResult)) > 0:  
 searchResult = self.database.animals.find(searchValue, {"\_id": False})  
 for data in searchResult:  
 print("Read found search result ", searchValue, ": \n", data, "\n")  
 return searchResult  
 else:  
 print("ERROR: Read value ", searchValue, " document not found.\n")  
 else:  
 raise Exception("Error: No valid search value entered.\n")  
  
 # Update method to implement the U in CRUD.  
 # Finds value/pair searchValue and modifies to the updateValue and returns it in JSON format, else return an error.  
 def update(self, searchValue, updateValue):  
 if searchValue and updateValue is not None and dict:  
 # Stores the UpdateResult  
 updated = self.database.animals.update\_many(searchValue, {"$set": updateValue})  
 # Stores the cursor of the updated documents  
 updated\_cursor = self.database.animals.find(updateValue, {"\_id": False})  
 #  
 if updated.modified\_count > 0:  
 print("Items modified: ", updated.modified\_count, "\n")  
 print("Returning result in JSON format: \n", dumps(updated\_cursor), "\n")  
 return dumps(updated\_cursor)  
 else:  
 print("Update value ", searchValue, " not found.\n")  
 else:  
 raiseException("Error: searchValue or updateValue invalid.\n")  
  
 # Delete method to implement the D in CRUD.  
 # Deletes all value/pair deleteValue found, else returns error message.  
 def delete(self, deleteValue):  
 if deleteValue is not None and dict:  
 deleted = self.database.animals.delete\_many(deleteValue)  
 if deleted.deleted\_count > 0:  
 print("Items ", deleteValue, " deleted: ", deleted.deleted\_count, "\n")  
 else:  
 print("DeleteValue ", deleteValue, " not found.\n")  
 else:  
 raise Exception("Error: deleteValue is invalid.\n")

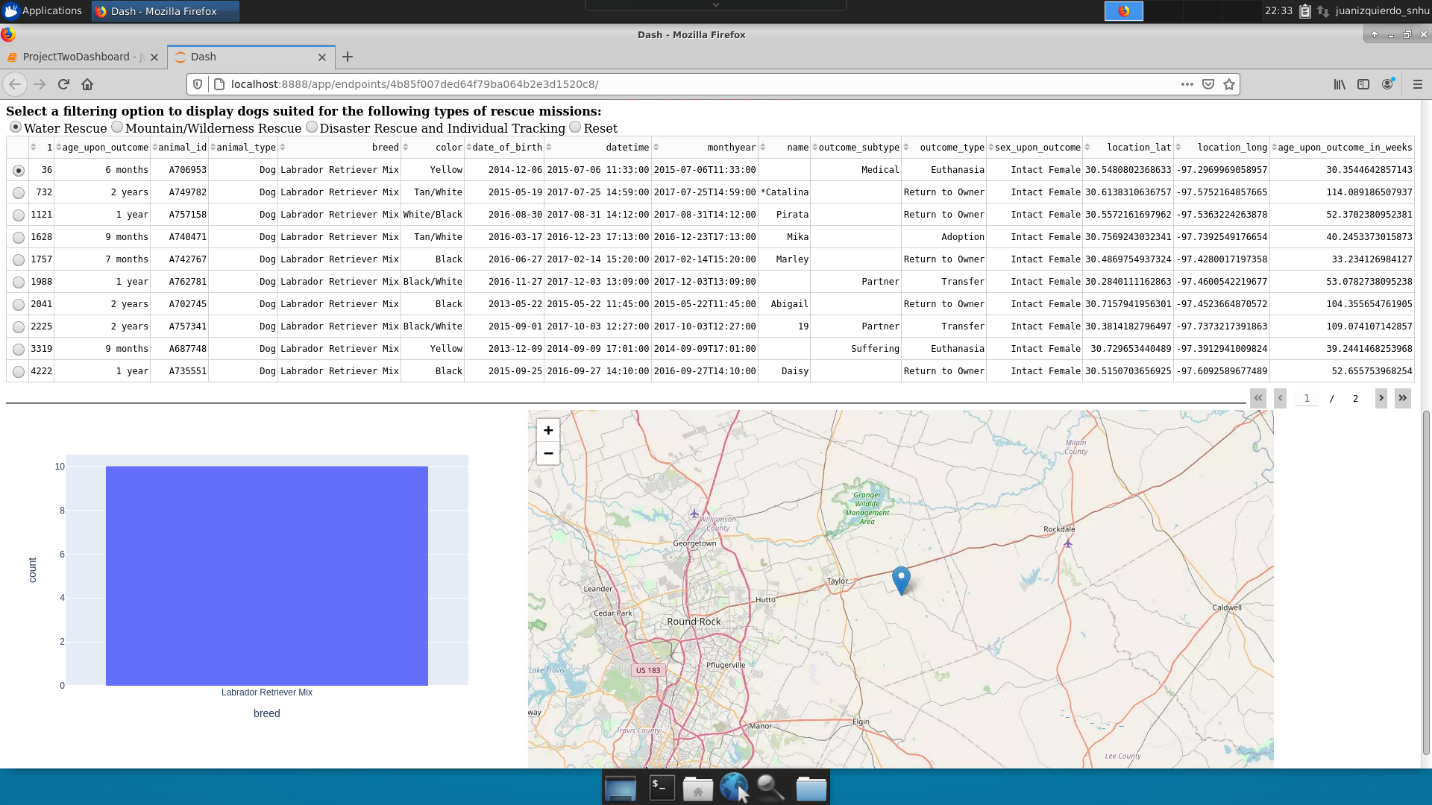
test.ipynb code below:

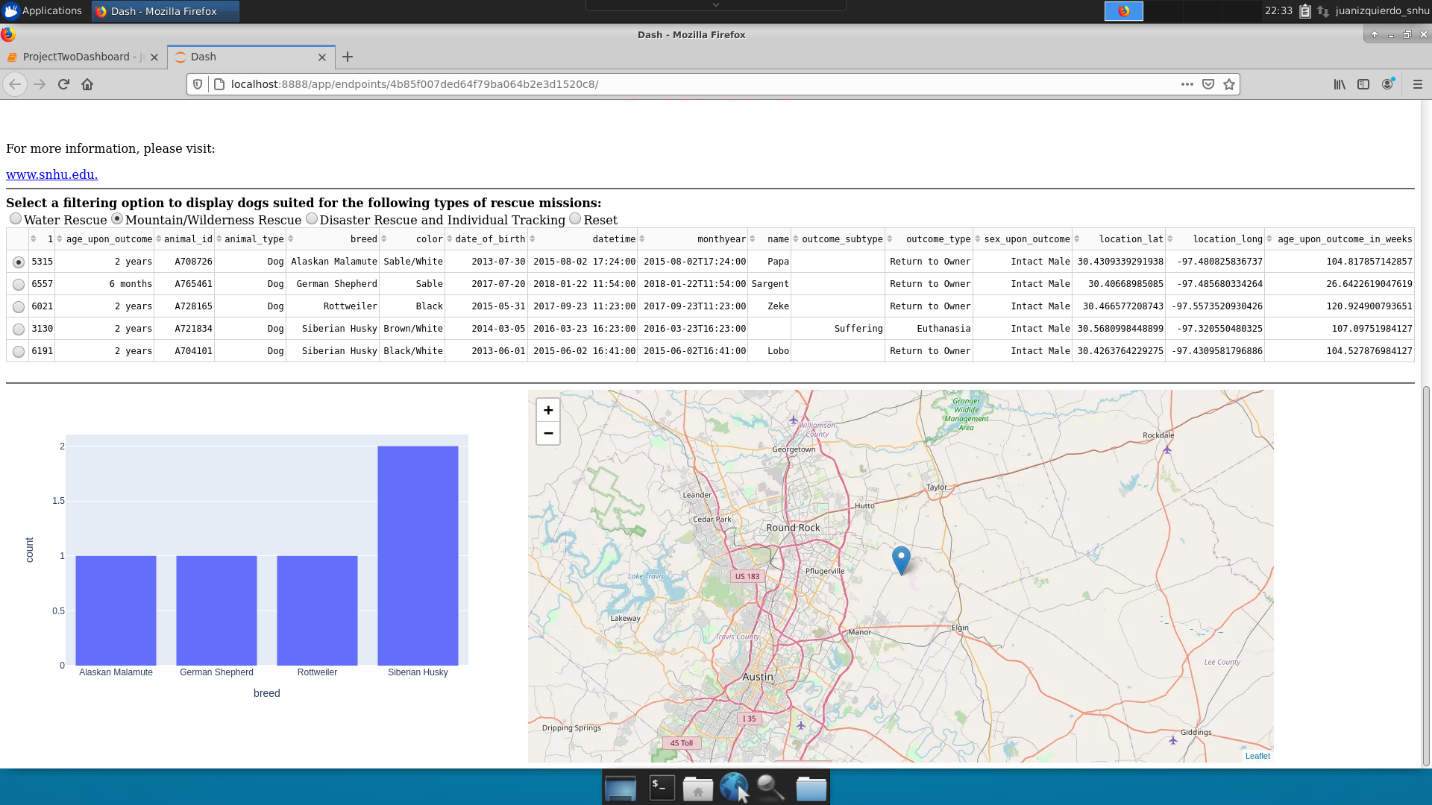
from animalShelter import AnimalShelter  
  
print("Testing\n")  
  
# Creating a test object from the AnimalShelter class  
test = AnimalShelter('aacuser', '1qaz!QAZ')  
  
# Testing Create method by inserting dog named ODEN.  
print("TESTING CREATE by inserting two entries named ODEN AND MOMO.\n")  
test.create({  
 "age\_upon\_outcome" : "1 years",  
 "animal\_id" : "AAAAA",  
 "animal\_type" : "Dog",  
 "breed" : "Pug",  
 "color" : "Green",  
 "date\_of\_birth" : "2005-03-24",  
 "datetime" : "2016-03-24 18:34:00",  
 "monthyear" : "2016-03-24T18:34:00",  
 "name" : "ODEN",  
 "outcome\_subtype" : "",  
 "outcome\_type" : "Return to Owner",  
 "sex\_upon\_outcome" : "Intact Male",  
 "location\_lat" : 30.6131087527131,  
 "location\_long" : -97.3189539659419,  
 "age\_upon\_outcome\_in\_weeks" : 574.110515873016  
})  
  
# Testing Create method by inserting dog named MOMO.  
test.create({  
 "age\_upon\_outcome" : "2 years",  
 "animal\_id" : "BBBBB",  
 "animal\_type" : "Dog",  
 "breed" : "Pug",  
 "color" : "Green",  
 "date\_of\_birth" : "2005-03-24",  
 "datetime" : "2016-03-24 18:34:00",  
 "monthyear" : "2016-03-24T18:34:00",  
 "name" : "MOMO",  
 "outcome\_subtype" : "",  
 "outcome\_type" : "Return to Owner",  
 "sex\_upon\_outcome" : "Intact Male",  
 "location\_lat" : 30.6131087527131,  
 "location\_long" : -97.3189539659419,  
 "age\_upon\_outcome\_in\_weeks" : 574.110515873016  
})  
  
# Testing Read method by searching for new dogs named ODEN and MOMO.  
print("\*\*\* TESTING READ by searching for two dogs named ODEN and MOMO. \*\*\*\n")  
test.read({"name": "ODEN"})  
test.read({"name": "MOMO"})  
  
# Testing Update method by updating document with dog named ODEN.  
print("\*\*\* TESTING UPDATE by updating ODEN entry's name to MOMO, totalling two MOMO and zero ODEN entries. \*\*\*\n")  
test.update({"name": "ODEN"}, {"name": "MOMO"})  
  
# Checking that update method updated document by searching for both ODEN And MOMO named dogs.  
test.read({"name": "ODEN"})  
test.read({"name": "MOMO"})  
  
# Testing Delete method. This should only delete the one object, the newly updated dog MOMO.  
print("\*\*\* TESTING DELETE by erasing the two existing MOMO entries and finding zero ODEN entries. \*\*\*\n")  
test.delete({"name": "ODEN"})  
test.delete({"name": "MOMO"})  
  
# Checking that delete method deleted documents by searching for both ODEN And MOMO named dogs.  
print("\*\*\* Checking that all entries have been deleted. \*\*\*\n")  
test.read({"name": "ODEN"})  
test.read({"name": "MOMO"})

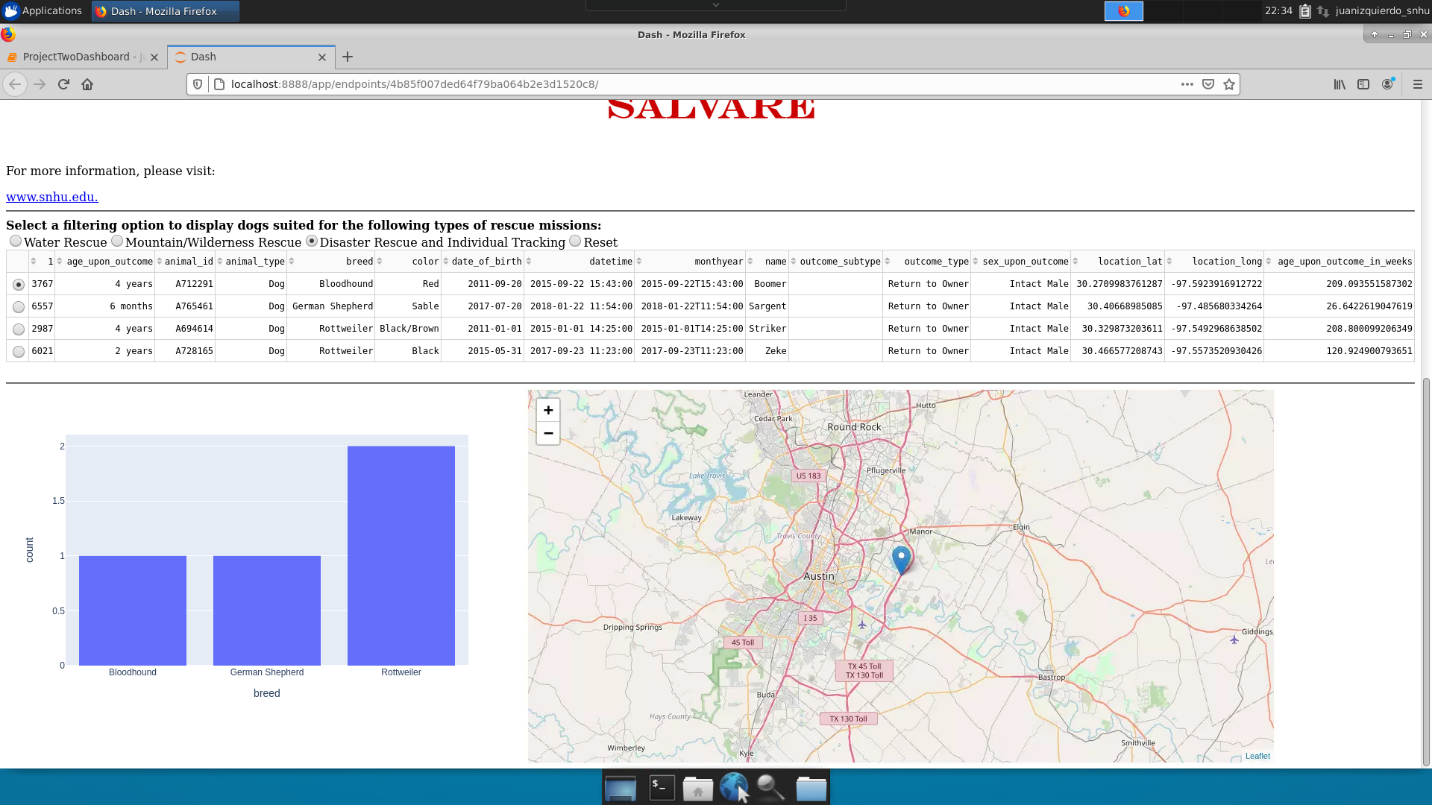
### Tests

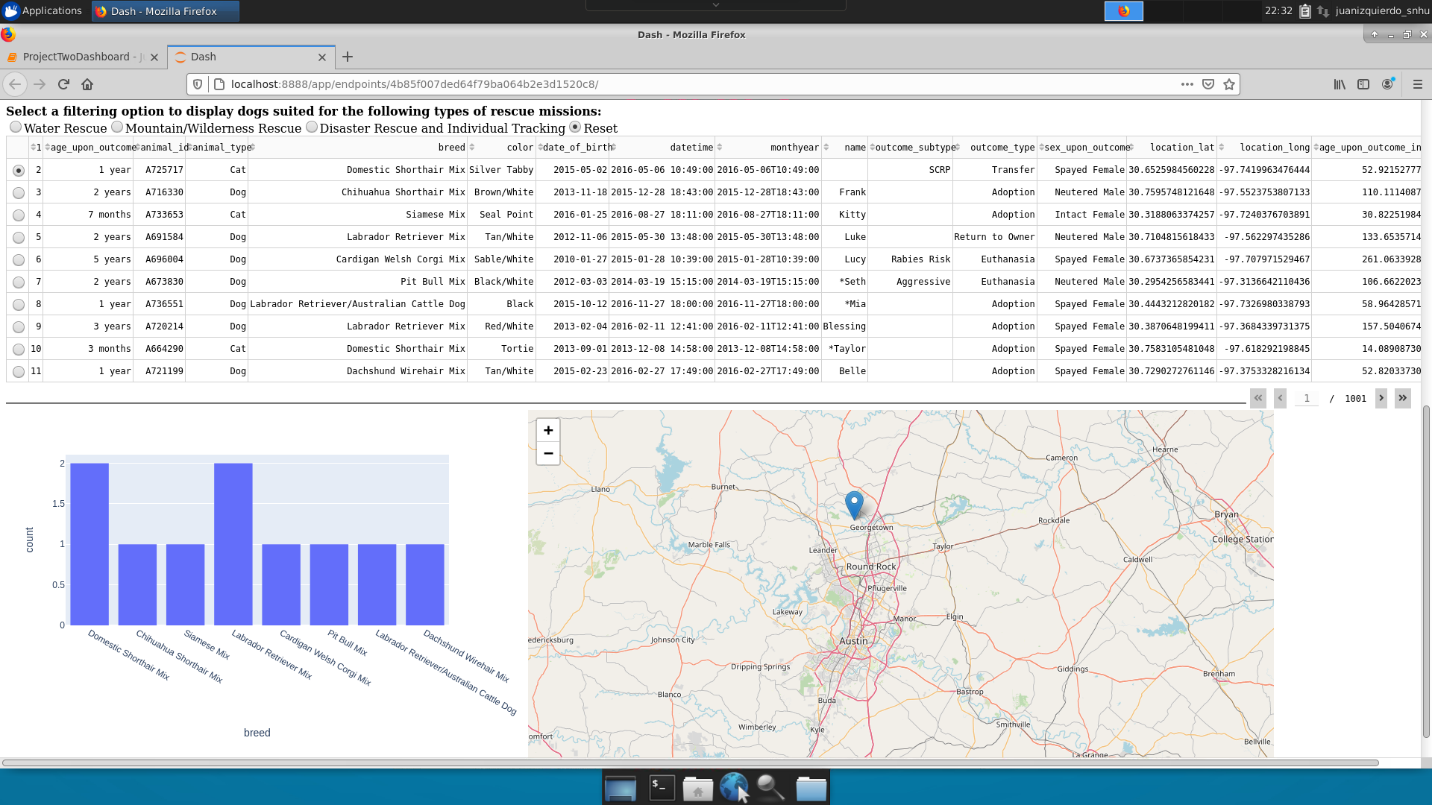


CS-340 M7 Identifier, logo, and SNHU link.

CS-340 M7 Water Rescue filtering displayed.

Mountain Wilderness Rescue filtering displayed.

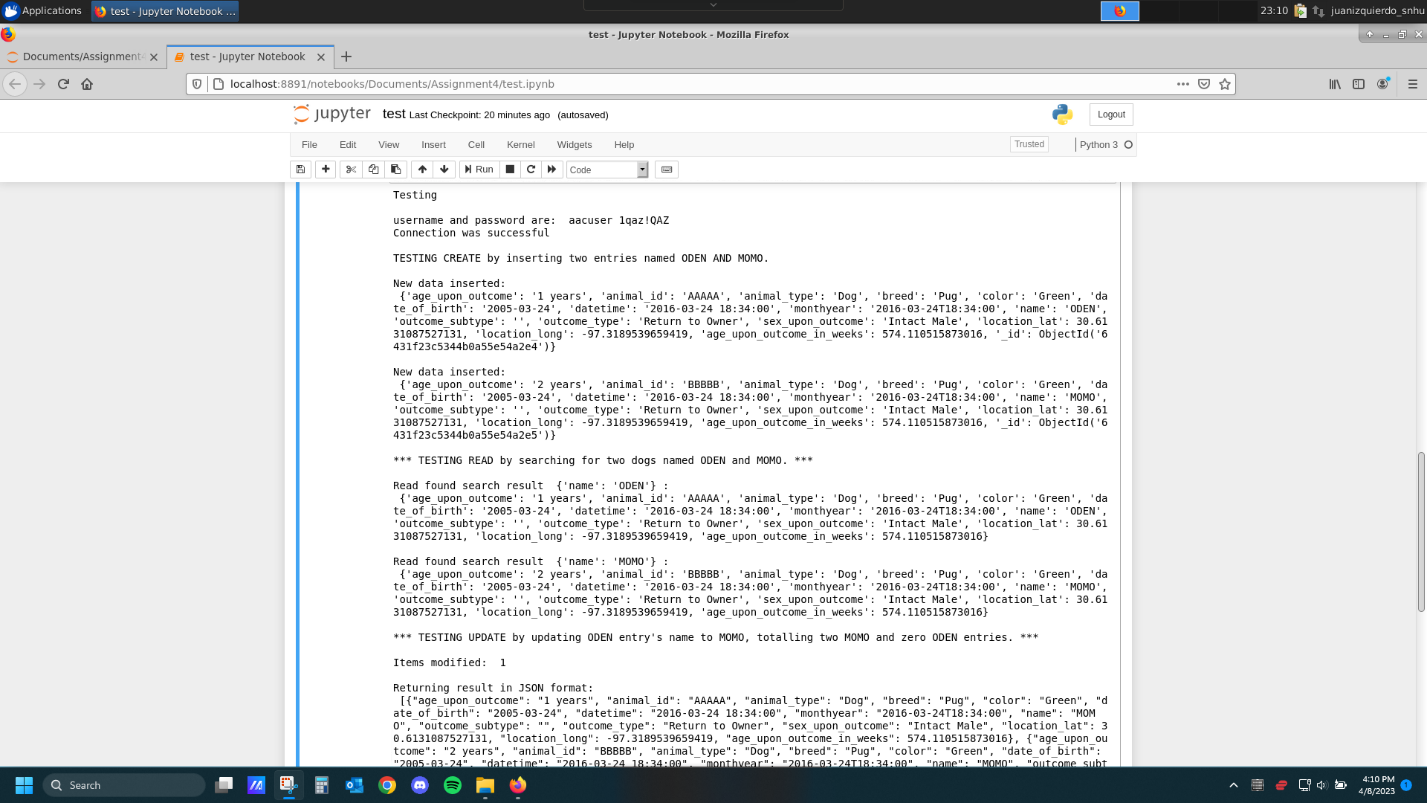
Disaster Rescue filtering displayed.

Reset filtering displayed.

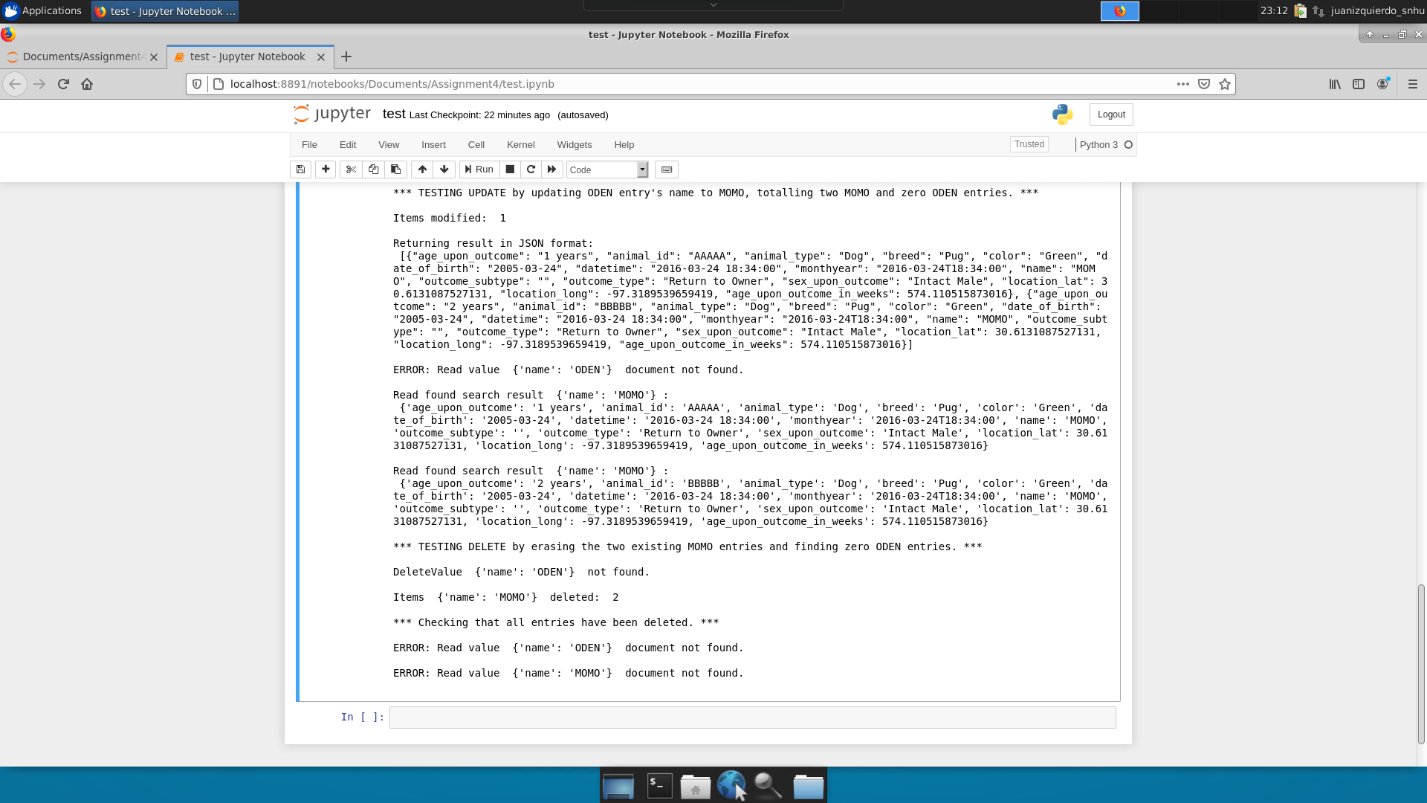
Graphical user interface, text

Description automatically generated

Example of authentication and execution of the Create and Read methods.



CRUD functionality test execution part 1



CRUD functionality test execution part 2

### Screenshots

Graphical user interface, text

Description automatically generated

Import CSV file.

Graphical user interface, application

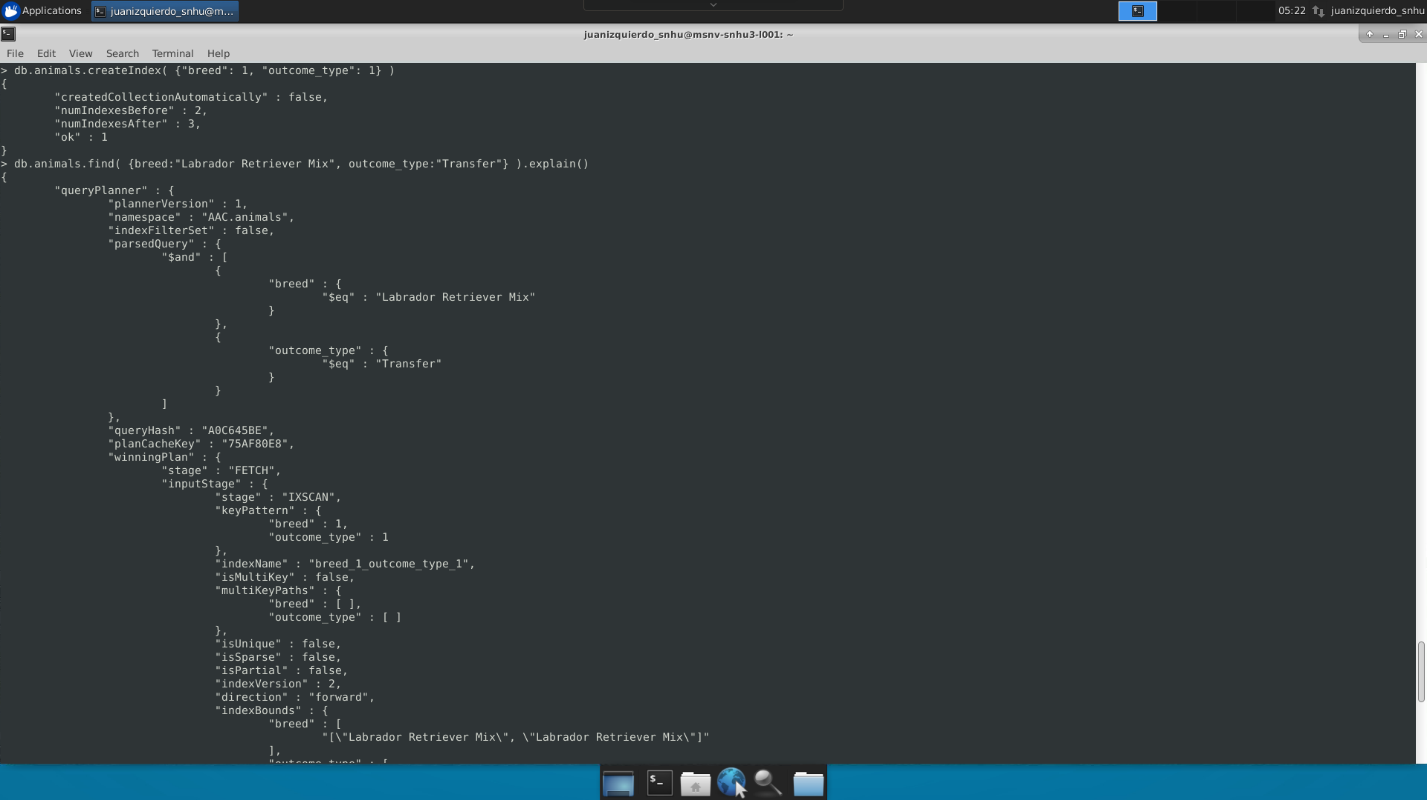
Description automatically generated

Create a simple index named “breed”.

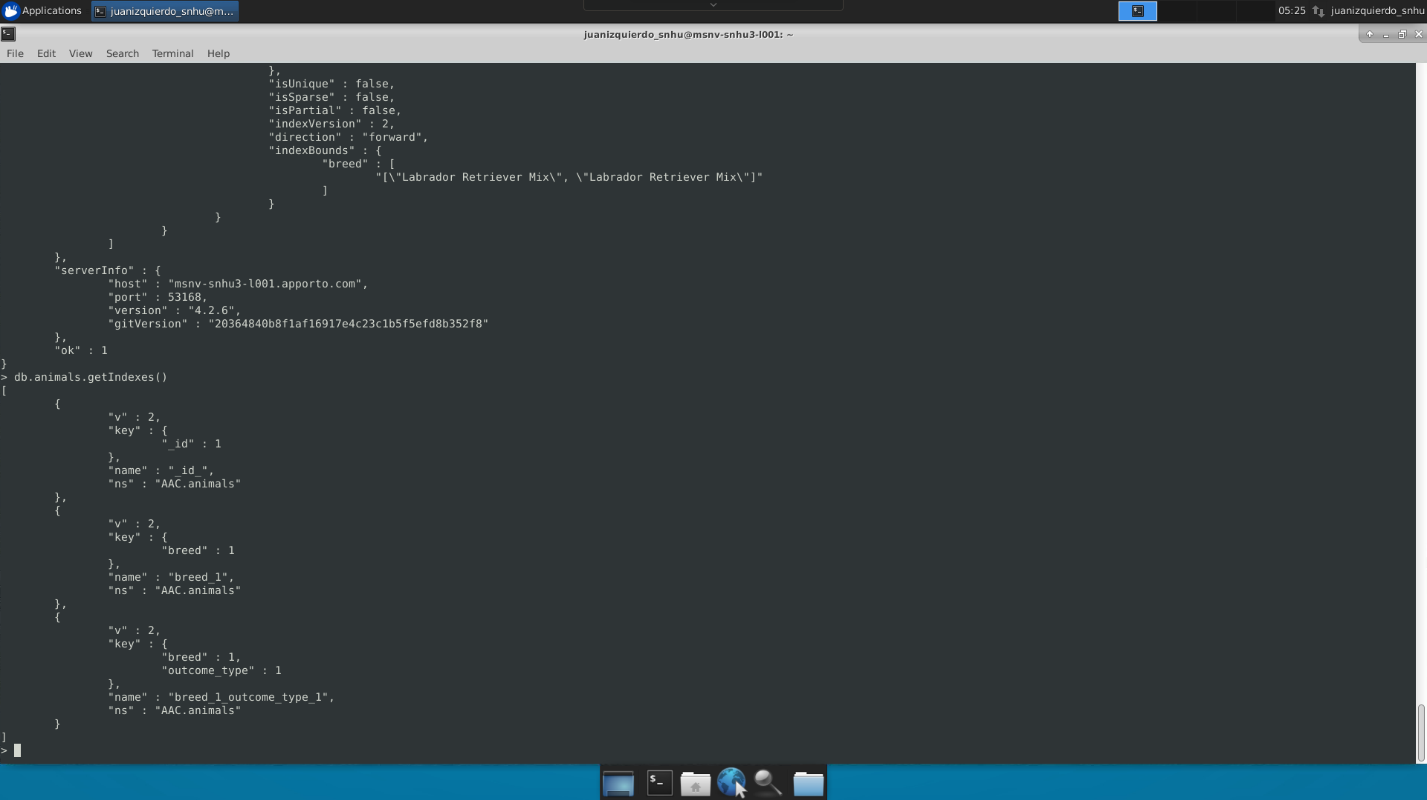
Graphical user interface, text

Description automatically generated

Sample query of “breed” index



Create compound index and show sample query.



List of indexes.

Text

Description automatically generated

Create “admin” account and verify access.

Text

Description automatically generated

Create “aacuser” account.

Text

Description automatically generated

Login process for both “admin” and “accuser” accounts.

## Contact

Juan Pablo Izquierdo: [juan.izquierdo@snhu.edu](mailto:juan.izquierdo@snhu.edu)

## References:

$in - MongoDB Manual. (n.d.). Retrieved April 13, 2023, from https://www.mongodb.com/docs/manual/reference/operator/query/in/

Ash15, Ash15Ash15 2777 bronze badges, & DaweoDaweo 30k33 gold badges1111 silver badges2323 bronze badges. (1969, May 1). *Python dash HTML element formatting*. Stack Overflow. Retrieved April 13, 2023, from https://stackoverflow.com/questions/73137305/python-dash-html-element-formatting

Barry, P. (n.d.). *Head First Python, 2nd edition*. O'Reilly Online Learning. Retrieved April 13, 2023, from https://learning.oreilly.com/library/view/head-first-python/9781491919521/?sso\_link=yes&sso\_link\_from=SNHU

*Dash HTML components*. Plotly. (n.d.). Retrieved April 13, 2023, from https://dash.plotly.com/dash-html-components

db.collection.distinct() - MongoDB Manual. (n.d.). Retrieved April 13, 2023, from https://www.mongodb.com/docs/manual/reference/method/db.collection.distinct/

*Html.link*. Plotly. (n.d.). Retrieved April 13, 2023, from https://dash.plotly.com/dash-html-components/link

*Python enhancement proposals*. PEP 8 – Style Guide for Python Code. (n.d.). Retrieved April 13, 2023, from https://peps.python.org/pep-0008/