# CS 340 Project Two README

## About the Project

The purpose of this project is to help Grazioso Salvare more efficiently categorize potential search-and-rescue dogs available within the area. Development has already been underway for the main CRUD functionality for the database. The application has been expanded to include a dashboard with a friendly user interface that will ideally reduce user error and training time.

## Getting Started

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

1. Make sure Python and MongoDB are installed and the csv file is located in the datasets folder
2. Enter the following commands in the terminal:
   1. cd /usr/local/datasets
   2. /usr/local/bin/mongod\_ctl start\_noauth
   3. mongoimport –port<YOUR-PORT> --db=AAC –collection=animals –type=csv –headerline –file=aac\_shelter\_outcomes.csv
   4. mongo
   5. show dbs
   6. use aac
3. This should import the AAC database and initialize it so it’s ready to query and manipulate. Next step is to create users for the database we’ll start with creating an admin account. Enter the following commands in the terminal:
   1. Inside the Mongo shell: use admin
   2. db.createUser({user:”admin”, pwd: passwordPrompt(), roles:[{role: “userAdminAnyDatabase”, db: “admin”}, “readWriteAnyDatabase”]})
   3. Enter password for admin account
   4. Exit Mongo shell: exit()
   5. /usr/local/bin/mongod\_ctl stop
   6. /usr/local/bin/mongod\_ctl start
   7. mongo –authenticationDatabase “admin” -u “admin” -p
4. This should prompt for the password again and complete creation of the admin account. From this account we can create more users like the “aacuser” we’ll be using for the examples below

## Installation

* Database: MongoDB - <https://www.mongodb.com/docs/manual/installation/>
  + MongoDB was chosen as the model component due to its flexibility, beginner-friendliness, and the speed at which you can develop queries and schemas is helpful due to the timeframe.
* GUI: Dash - <https://dash.plotly.com/>
  + Dash was chosen for the GUI to provide the view and controller structure for the application. It’s built-in UI components helps simplify creating data-driven dashboards with the way it integrates modern UI elements directly to the analytical python code.
* Scripting: Python3 - <https://www.python.org/downloads/>
* Integration: PyMongo - <https://pypi.org/project/pymongo/>

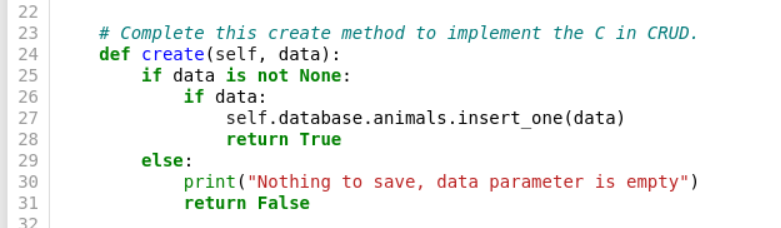
## Usage

### Code Example

This is a CRUD application so the main functionality is based around manipulating the data in the database using the four data methods, Create, Read, Update and Delete. To make the application user-friendly a dashboard has been added as well with interactive components to help categorize the data.

*CRUD API:*

To create a new document, there is a Create method that takes in an object and returns a Boolean based on if it successfully inserted the data into the database

**

### To query for a document with specific parameters, there a Read method that takes in an object and returns an array with Cursor objects that can be looped over to see which documents matched the query

Graphical user interface, text, application, email

Description automatically generated

To update an existing document with new data or replace data, there is an Update method that takes in two objects, one a query for the existing document to update, the other has the new data to add/replace on the queried document. This will return an “UpdateResult” object with a field to track the count of objects recently modified

Graphical user interface, text, application

Description automatically generated

To delete an existing document, there is a Delete method that takes in an object, a query for the existing document, and returns a “DeleteResult” object with a field to track the count of objects recently deleted

Graphical user interface, text, application

Description automatically generated

*Dashboard UI:*

The Dashboard includes a radio group to allow the user to filter the Data Table for dogs that fit within a certain Rescue Type category.

Text

Description automatically generated with low confidence

Mountain or Wilderness Rescue will filter the Data Table for Intact Male Dogs between 26-156 weeks with any of these breeds: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler.

Graphical user interface, table

Description automatically generated

Disaster Rescue or Individual Tracking will filter the Data Table for Intact Male Dogs between 20-300 weeks with any of these breeds: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler

Graphical user interface, table

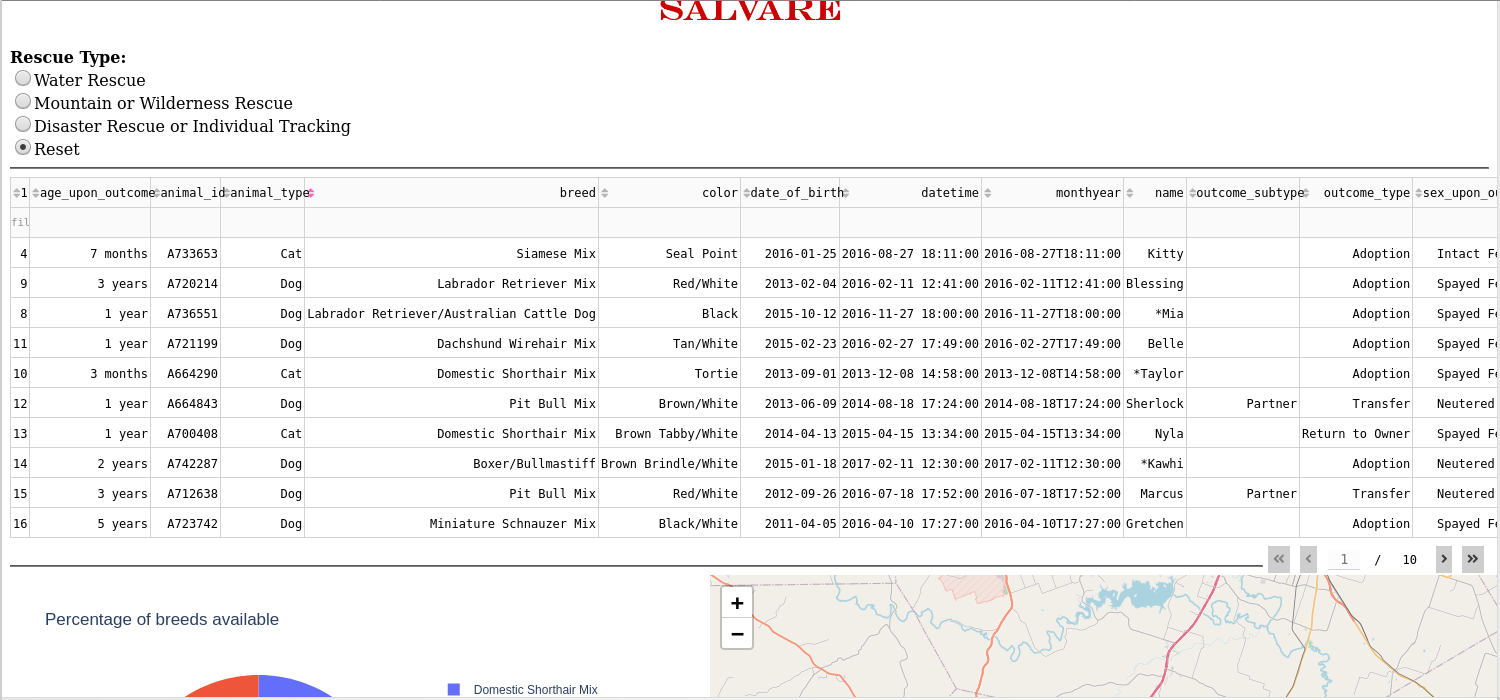
Description automatically generated

Water Rescue will filter the Data Table for Intact Female Dogs between 26-156 weeks with any of these breeds: Labrador Retriever Mix, Chesapeake Bay Retriever, Newfoundland

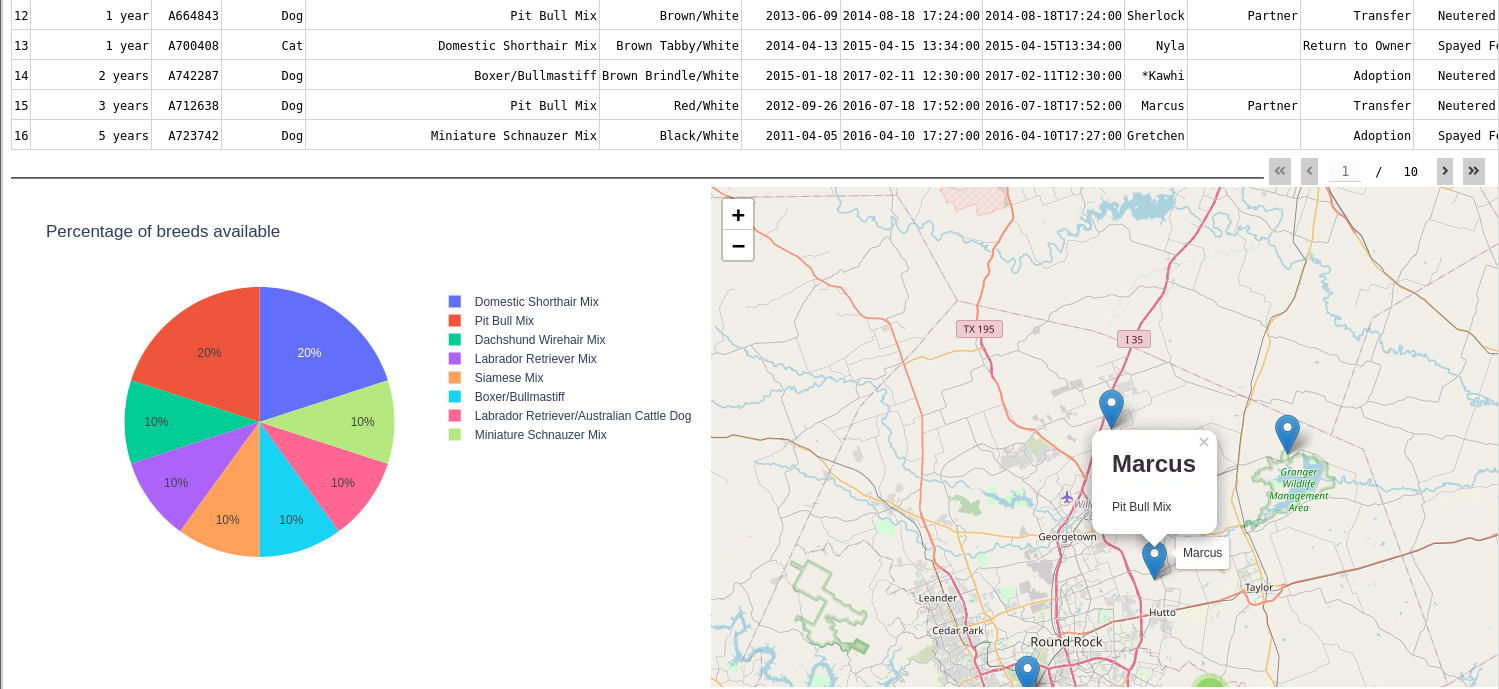
Table

Description automatically generated

Reset will be selected by default, if it is selected all animals in the database will be in the data table no sorting/filtering applied



There will also be a pie chart representing the amount of different breeds available from the data and a map with the locations of each animal from the data. These will update if one of the aforementioned filters is selected.



**Reflection**

There were definitely some challenges when it came to understanding how to get this application to function appropriately. As mentioned the CRUD functionality was implemented first and it took some time to get used to the way MongoDB queries work as well as implementing it in a way that would return the proper information needed. One of the trickiest parts was authentication, which was the next part of the implementation. The command line sometimes isn’t the most beginner-friendly so it took some trial and error to figure out the appropriate commands to get the proper imports and properly send the data needed to authenticate the server. The Dashboard was the last part of the implementation and perhaps the most overwhelming considering the amount of tools at your disposal with the Python Dash library. I had to reel myself back to get the base functionality first before getting fancy with all of the built-in components Dash gives you. I also work in Web Development so I’m familiar with HTML, CSS and JavaScript so though I recognized the structuring for the web page, it took a while to figure out how to implement the HTML and CSS the way Dash wants you to which I found interesting. All in all, I think I was able to overcome most of my obstacles and come out with a viable product that meets the clients needs and expectations.

## Contact

Your name: Monticia Dunn