# CS 340 Grazioso Salvare Dashboard Program README

## About the Project/Project Title

Application that allows permitted users of the Grazioso Salvare company to access a database that contains data given by the Austin Animal Center through a CSV file. Users will be able to look through the database and refine the detail of what animal they are looking for in particular. Furthermore, once a filter has been applied, there is a geolocator for the animals the user selects as well as a histogram to show the user how many breeds there are for the current filter selected.

## Motivation

As a company that specializes in custom software design and development, we were more than happy to accept the request from Grazioso Salvare about creating a database with C.R.U.D. capabilities that utilizes a python program to help manipulate the data within. Not only that, but also be able to display the data in a visual manner for the user to be able to see. Accomplishing a test as such will surely display the skill and abilities of our team.

## Getting Started

For full functionality, the following steps must be completed:

1. Import the CSV file (aac\_shelter\_outcome.csv) into the Mongo Database.
2. Within the database, make sure that there is a simple and complex index so that the data within the document can be parsed through.
3. Ensure that there is an administrative account as well as a standard user account that is authorized to access and make changes to the document in the database.
4. Ensure that the device that is accessing the database has a supported version (preferably the most current version) of Python so that the python files can run properly.
5. Run the Program through the Jupyter Notebook application.

## Installation

A supported version (preferably the most current version) of Python to be able to run the CRUD.py and Dashboard.ipynb files necessary for the program to work correctly.

MongoDB will also need to be installed since this will be the database that the program will be operating with.

**Usage:**

**CRUD:**

***Create –*** The purpose of “Create” is to be able to create data and be able to input it into the database. In order to do so, the user would have to run a command such as “print(animals.create({StringId : String})” in which it will respond on whether or not the task was successful or not (see *Tests* below for further explanation).

***Read –*** The purpose of “Read” is for the user to be able to pull out a data entry for further examination. The user would have to write a command such as “animals.read({StringId:String}) with the string id and string being the information that the user is trying to find. For more a more detailed search a user can enter multiple “stringId:String” entries with a comma separating each field.

***Update –*** The purpose of the “Update” tool in this program is to grant the user the ability to alter the information of an already existing data entry within the database. In order to accomplish this task, the user will need to run a command such as “animals.update({StringId:String}, {StringId:String})”. The first set of parameters in the bracket is used to find the entry while the second parameter is what will be updated.

***Delete –*** The delete function is used to delete an existing data entry that exists within the database. For a user to accomplish this, a command such as “animals.delete({StringId:String})” in order to delete a data entry. Similar to the Read function, a search can be further refined by adding more parameters in the parenthesis.

**Dashboard:**

The dashboard will allow the user to see a more visually appealing representation of the data that exists within the database. Furthermore the user has abilities to select from one of the four major filter options (filtered by suitable animals for various rescue types) or choose to filter out an option on the top of the data column.

## Tests:

**CRUD:**

***Create*** – The first test that was done was creating a small sample of data to import into the database, which was a success. A successful implementation of the data will result in the command returning “True” whereas as a “False” return means that the data was not able to be put into the database.

***Read*** – A test was conducted with a name that is very common (and most likely in the list) while another test was conducted with a bizarre name to see what would happen if no entry could be found.

***Update*** – Testing this portion of the program involved having the program first look for a data entry based on the first selection, in this case it’s the name of the animal which is “Rex”. Then, we choose the field that needs to be updated and then input the information (in the form of a string) that we want to update the information to. In this test, we updated the “outcome\_type” so that all dog’s named Rex were set to “Return to Owner”. Another test was conducted to attempt to update a data entry that doesn’t exist. Having deleted the animals named “Doug” earlier, attempting to update to the information will return with a “False” result. The first test with “Rex” returned “True” since there were 175 entries of Rex.

***Delete*** – The delete function was tested by deleting an entry by looking for a name that did exist in the database. Using the name “Horatio” we were able to find and delete a total of 18 animals successfully. When attempting to delete a data entry not in the database, like the name “Nebuchadnezzar”, it shows that no entries were found but will still send the “ok” message as if it had actually deleted the data that was requested.

**Dashboard:**

Each of the four major filter options were selected and then examined to make sure the proper breed, animal sex, age, and animal type were displayed based on the specifications document.

***Screenshots:***

***MongoDB CSV import execution:***

A screenshot of a computer

Description automatically generated

***A screenshot of a computer program

Description automatically generatedUser(s) authentication:***

**A screenshot of a computer screen

Description automatically generated**

***Create Method:***

***A screenshot of a computer

Description automatically generated***A computer screen shot of a program code

Description automatically generated

***A white rectangular object with a black border

Description automatically generated***

***Read Method:***

A screenshot of a computer

Description automatically generatedA computer screen shot of a program code

Description automatically generated

***A screenshot of a computer code

Description automatically generated***A screen shot of a computer program

Description automatically generated***Update Method:***

***Delete Method:***

A screen shot of a computer program

Description automatically generated

***A screenshot of a computer

Description automatically generated***

***Dashboard:***

***A screenshot of a computer

Description automatically generated***

***Dashboard Water Rescue:***

***A screenshot of a computer

Description automatically generated***

***Dashboard Mountain Rescue:***

***A screenshot of a computer

Description automatically generated***

***Dashboard Disaster Rescue:***

***A screenshot of a computer

Description automatically generated***

***Dashboard Reset:***

***A screenshot of a computer

Description automatically generated***

***Challenges:***

One of the largest challenges that I encountered was having the map and graph data display correctly, however I found that this was mostly due to syntax errors and having improper callback methods for each of the sections. Another challenge was having the data table correctly apply the major filters onto them.

## *Contact:*

Douglas Buckley ([douglas.buckley1@snhu.edu](mailto:douglas.buckley1@snhu.edu))