# CS 340 PROJECT README

## About the Project/Project Title

The Grazioso Salvare company has contracted Global Rain for a full stack development of an animal database with a client-face web application dashboard to catalog the animals of five animal shelters in the region around Austin, Texas that are working with them. Grazioso Salvare identifies dogs that are good candidates for search and rescue training. The Austin Animal Center (AAC) Database holds information on the animals of the five shelters that have partnered with Grazioso Salvare and the Grazioso Salvare Dashboard allows users to view the animals’ information and filter dogs that meet the specific criteria suitable for rescue training.

## Motivation

As mentioned, the Grazioso Salvare company identifies dogs that are good candidates for search and rescue training. Search and rescue dogs can aid in rescuing people and/or other animals in life threatening conditions. By having accessible collection of information for animals at the shelters, Grazioso Salvare will be able assess potential candidates more easily.

## Getting Started

The project requires MongoDB and PyMongo to be installed on your system. (Please see the “Installation” section for links and more information.)

To access the AAC Database locally, you will first need to import the CSV file into MongoDB using the MongoDB import tool. Once the CSV file has been imported, you will be able to interface with the AAC Database using the Python Module file.

Type the code below into the terminal window to import the aac\_shelter\_outcomes.csv file onto your system:

mongoimport --username=”${MONGO\_USER}” \

--password=”${MONGO\_PASS}” --port=${MONGO\_PORT} \

--host=${MONGO\_HOST} –db AAC --collection animals \

--authenticationDatabase admin --type csv --headline --file /usr/local/datasets/aac\_shelter\_outcomes.csv

Sample:

A screenshot of a computer program

Description automatically generated

Once the AAC Database has successfully imported locally, you will need Jupyter Notebook to run the .ipynb file and access the web-based dashboard.

## Installation

Below are the programs needed to install the AAC Database locally. Please visit the links for instructions on installing the programs onto your system.

MongoDB: <https://www.mongodb.com/docs/v2.4/installation/>

PyMongo: <https://pymongo.readthedocs.io/en/stable/tutorial.html>

Jupyter: <https://jupyter.org/install>

Using Jupyter, users will be able to run the .ipynb and access the web-based dashboard. The project will interface with the MongoDB through application programing interface (API) calls initiated through the dashboard.

## Usage

The Dashboard interacts with the AAC Database through API calls established in the Create, Read, Update, and Delete (CRUD) Python file. The Grazioso Salvare Dashboard allows its users to see the animals currently cataloged in the AAC Database and filter the animals to dogs that meet the specific criteria for different rescue training. The Dashboard also show the location of the animal through a geo-location chart. Users will be able to select different animals on the interactive data table to see their location in the Austin, TX area. Users will also be able to see the percentage of dog breeds that fit the desired traits for the rescue training in a pie chart on the dashboard. This pie chart will update live with new filter selections.

### Code Example

The project uses a Python driver to interface with MongoDB using application programing interface (API) calls. Below is the Create, Read, Update, and Delete (CRUD) Python module. There were difficulties with translating MongoDB queries to Python script, but after some research I was able to figure out the correct syntax. I struggled with how to use the MongoDB cursor position to print out documents, but again I was able to research it and find the solution. As the name suggests, the code can create, read, update, and delete documents in the AAC Database. After user authentication, the user passes a query through one of the four methods to the AAC Database and a result is returned.

When creating a new animal document, the query is made with the necessary attributes and passed to the AAC Database. That information is then inserted as a document to the AAC Database. On a successful insert, the method returns “True”. For the read method, the query is passed to the AAC Database and all documents matching the passed query will be printed to the screen. If no documents match the passed query, the user will be informed. The update method passes a search query and the update information. All documents that match the passed query will have their information updated with the passed update information. Once completed, the number of documents that were updated is returned. Lastly, the delete method passes the query to the AAC Database and deletes all documents that match the passed query. The method then returns the number of documents deleted.

CRUD Code:

*A screenshot of a computer program

Description automatically generated*A computer code with text

Description automatically generated with medium confidence

The Dashboard required a lot of research as it was the first time working with the Dash framework. Database authentication is hard coded in for ease of access while using the Dashboard. The code can be simplified to two portions: layout and callbacks. With in the .ipynb file I setup the layout of the Dashboard with the specified information requested from Grazioso Salvare. This includes the charts, maps, and buttons that are going to be interacted with and displayed. I must define each object I am planning on implementing as well as set how and where it will be displayed in relation to the other objects. Each object that will require an “-id” to callback to and interact with data.

A screenshot of a computer code

Description automatically generated

The radio buttons, when clicked, will pass their respective value to the update\_dashboard callback function. Here, depending on the value that is passed, the callback function will send a specific query to the AAC database through the CRUD python module file. Each radio button is coded with a specific query to search for the AAC database for the animals that meet the designated criteria.

A screenshot of a computer screen

Description automatically generated

The Breed Pie Chart and Geo-location Chart use the data table information and update dynamically based on the filters used. As mentioned earlier, the data table was established in the layout portion of the code. The breed pie chart and the geo-location chart are executed in the update\_graphs and update\_map callback functions respectively. They utilize the “datatable-id” I assigned to the data table object in layout portion. As the data table changes and is interacted with, the chart and map will update with the data available to provide the user with the information they need.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

### Tests

The picture below shows the test sample calls made to the Python module and their results. A new animal document is created, which prints “True” to the screen. The test module then queries the created document and prints the results to the screen. The animal type for “Sherlock Bones” is then updated to “Cat” and the number of documents modified is shown. This is confirmed by another query for the created animal showing that the animal type has been updated. The animal is then deleted resulting in the number of deleted documents being printed to the screen. This is confirmed with a final query which prints that the animal was “not found”.

*A screenshot of a computer program

Description automatically generated*

Below are screenshots of the Dashboard’s functionality. The Dashboard opens with the Grazioso Salvare logo front and center. As the developer, my name is showcased at the bottom of the screen with the copyright date. The data table shows all the animals cataloged in the AAC database. For manageability, the animals are listed in pages ten at a time. At the bottom left side of the screen is the breed percentage pie chart. This pie chart will show the percentage of animal breeds listed on the data table and will update dynamically as filters are applied. Lastly, is the geo-location map of the selected animal shown at the bottom center of the page. By default, the map will show the first animal of the data table. And like the breed pie chart, the map will update dynamically as new filters are applied or selections are made.

A screenshot of a computer

Description automatically generated

The screen shots below show how the filters being applied and the pie chart and map updating.

Water Rescue filter:

A screenshot of a computer

Description automatically generated

Wilderness Rescue filter:

A screenshot of a computer

Description automatically generated

Disaster Rescue filter:

A screenshot of a computer

Description automatically generated

Reset filter:

A screenshot of a computer

Description automatically generated

## Difficulties

As mentioned earlier, this was my first time working with the Dash Framework, so it took some time to fully understand it. There is still a wealth of information that I have not struck, but I was able to get by with what I learned. My first impression of it are positive though. It simplifies implementing objects that would be more tedious and complicated in another format. While I am adequate with python, I am more versed in C++ and Java. The Dash Framework utilized more complicated python techniques I am not familiar with, but I find it very intriguing. One of my major struggles delt with variable passing. Reading the information on the dash website was fairly straightforward, but implementing it was difficult. Especially with the template code used to start the project, they would define or implement variables that I didn’t understand where they got. Still, I got through it, but it took a lot of time.

## Contact

Your name: Raphael Coloma