# CS 340 README AnimalShelter

# Aaron McDonald

## About the Project/Project Title

Grazioso Salvare has asked that we create a full stack environment that utilizes a database from several nearby animal shelters to allow them to easily sort and filter specific dog breeds. Specifically, these dogs must meet certain criteria for specific types of training for rescue missions. This project is the overall results of this objective.

## Motivation

The motivation for this project was the need from Grazioso Salvare, to need a user-friendly way to look at and sort the information gathered. Since there are thousands of entries into the database, Grazioso Salvare needed a more visual representation of this data to more easily find the dogs they required for their training.

## Getting Started

To get started using this web application you must run the Jupyter source file called ProjectTwoDashboard. You may need to update the credentials that were hardcoded into the source file to match the credentials for your database however once you do this will launch the web app, allowing you to connect to localhost.

## Functional Requirements

The project fits the needs of Grazioso Salvare through the following means. Firstly, upon loading the webpage we start on the “Reset” category for filtering, this includes all of the filtered data, showing a pie chart to represent the distribution of different animals in the working set and a geolocation chart for a specific animal selected from the rows: A screenshot of a computer

Description automatically generated

Additionally, there are sorting options for each column allowing for further customization of the presented data to meet the customer needs. Lastly, I have implemented backend pagination such that there is pagination based on the size of data present, however that data remains on the server until requested, ensuring that the website is capable of running more seamlessly for clients that may have lower-end machines.

Furthermore, each of the radio buttons at the top of the data table enable further filtering for the specific type of rescue dogs that they are looking to train. Here are the three screenshots to show the different types of dogs that could be filtered, based on Grazioso Salvare’s requirements. Note that the geolocation chart, pie chart and pagination will update to reflect the newly pulled data.

A screenshot of a computer screen

Description automatically generated

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

## Tools Used

For this project we decided to use a NoSQL database known as MongoDB. This database is uniquely equipped to handle we applications and non-conforming data as each entry is considered a document. Therefore this database seems suitable for this application where each animal shelter may be using slightly different formats for their entry of animals.

Furthermore, we utilized the Dash framework for the viewer and controller to perform the tasks of fetching data from the database and displaying that data for the web app. This was a very useful framework as this allowed for the web application to quickly and efficiently be deployed with the needed functionality. The dash framework has many useful features backed into it allowing for quick deployments of simple websites such as the one we built here.

Links for the tools used:

MongoDB - <https://www.mongodb.com>

Dash Framework - <https://dash.plotly.com>

## Project Creation

In creating this project there were several steps that had to be addressed, working from the bottom up with the database. Firstly, I began by creating the database that we would be utilizing for this project. This meant creating a user account for the Grasiozo Salvare group and importing the CSV file into the MongoDB database.

After importing the file into the database I ran a few test queries to ensure that the database would work as intended, being able to pull the needed data that we would expect to need. As an example here is a query for pulling the water rescue dogs from the mongodb database:



After ensuring that this created the expected results, I continued to test the other queries as needed before moving on the creating the CRUD middleware in Python. I created the file called AnimalShelter in python which acts as the CRUD middleware, being able to read, write, update, and delete entries from our database with a user-friendly API. This utility was created using pymongo and I was able to test this utility using Jupyter source files (see my previous README on that interface).

Finally, for creating the front-end of the web application, I was given a template file from a Jupyter source file and began to create the different aspects of the requirements. There were some challenges along the way of having to research the different aspects of the framework that I was weak in and determine how to best use them with what I had currently created.

Particularly, it was challenging to incorporate backend pagination with frontend visuals such as the pie chart and table sorting. Overall, I was able to merge all of the view aspects into a single callback function that works with gathering the data from the database and displaying that data to the widgets based on the filtering and sorting options applied.

## Contact

Aaron McDonald

Email: aaron.mcdonald1@snhu.edu