# CS 340 README

## About the Project/Project Title

This project is building a database to handle the information stored about the animals that pass through the Austin Animal Center. At this point CRUD methodology has been implemented in the project allowing for all foreseeable editing need for the information in the database. The dashboard has been implemented to allow for exploration of the information in the database. This dashboard displays a data table that contains all the information from the database, a pie chart that displays a visual representation of the breeds currently displayed on the current page of the data table, and a map showing the location of the currently selected animal. In addition to these features a radio button is in place to allow the user to select different filters to find animals best suited for different service tasks that Grazioso Salvare is looking to train animals for.

## Motivation

This project was started to create a more convenient and easier to use system to keep track of data stored about the animals that have come through the animal center. This project will also aid in maintaining the information held in Austin Animal Center’s database as well as make it easier to sort out animals best suited for tasks that Grazioso Salvare is looking to train for.

## Getting Started

To get a local copy of the project working start by installing MongoDB, load the CSV file containing the Animal Center’s outcomes dataset using the MongoDB import tool. Next the animal\_shelter.py file should be added. This file acts as the go between to access the database on MongoDB and will aid in adding new information to the database and reading information within the database that matches prompts given.

**Tools Used**

MongoDB, Dash Framework

MongoDB was used for the model component as it integrates very easily with Python due to the utility prebuilt for Python pymongo. It also easily handles different document structures without any problems.

Dash Framework was used to integrate the tools that HTML can add in a Python app so that the dashboard is as visually attractive as it is with the integration of the widgets for the pie chart along with the map. It also allowed for the integration of the radio buttons so that filters on the data could be easily toggled through.

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

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

## Installation

MongoDB, aac\_shelter\_outcomes.csv, animal\_shelter.py, ProjectTwoDashboard.ipynd

MongoDB should be installed as directed and can be accessed at mongodb.com, next aac\_shelter\_outcomes.csv can be added into MongoDB with the following command entered into your command line:

mongoimport --port=(your port number) --db=AAC(You may name the database as you wish but this is what is used in this project) --collection=animals(same as the database name) --type=csv --headerline --file=(ensure the whole file path is included here)/aac\_shelter\_outcomes.csv

Once the file has been imported into MongoDB it should be accessible through the animal\_shelter.py file which will allow for information in the database to be upkept and allow for the data to be easily handled by the user.

Using ProjectTwoDashboard.ipynd will allow the user to access the information that animal\_shelter.py is handling and make it usable and understandable.

## Usage

### Code Example

At this point CRUD has been implemented in the code for the information included in the database. Create will add data if data is given when the method is called. Read will retrieve data from the database that matches a given search item, read will give only the first example that matches the search item. Update will take a indicated piece of information that needs to be changed and update it with the given new information. Delete will remove items from the database. The dashboard gives the user the ability to look at the data from the database and see a visual representation of the data currently displayed with a pie chart and the currently selected animals location is displayed on a map. It also gives the user the option to filter the data to see only animals that meet the preferred requirements for different service animal categories.

### Tests

Create, Read, Update, and Delete have all been tested, the test for create passed information to the create class and if the information was successfully added the method returns true, so the test took this and if true was returned a message printed declaring that “animal added”. Read was tested by passing information pulled from the database to the read class, if the item was found then the class returns the data found, the test passed this information to a variable that was then printed by the test. Update was tested by passing the function an indicated item in the database and updated the specified information. Delete was tested by first adding a new item in the database and then deleting this item. Both update and delete return information in JSON format and this information was printed in the tests. The dashboard was tested incrementally as different widgets were added, starting with the data table this was added and tested to be sure that the amount of items displayed at a time was manageable for users. The map was then added and tested to ensure that the selected animal location was displayed. The pie chart was then added, initially it displayed all animals included in the database and was very difficult to parse out the information being displayed. The radio buttons to filter animals out of the data were added and tested. The pie chart was updated after the addition of the radio buttons to only display the animals that are displayed on the current page making the data much easier to understand and parse through.

### Screenshots

MongoDB import execution

Text

Description automatically generated

User Authentication Execution

Text

Description automatically generated

Example of test code to test the create and read methods that have been developed

Text

Description automatically generated

Starting state of dashboard

A picture containing map

Description automatically generated

Dashboard when Water Rescue selected

A picture containing graphical user interface

Description automatically generated

Dashboard when Mountain or Wilderness Rescue selected

A picture containing graphical user interface

Description automatically generated

Dashboard when Disaster Rescue or Individual Tracking selected

A picture containing graphical user interface

Description automatically generated

Dashboard after Reset selected

A picture containing graphical user interface

Description automatically generated

**Challenges**

Building the application was straight forward for the most part. My biggest difficulty came from trying to find the best way of implementing the radio buttons. I initially ran into an error where if I changed the filter to Mountain or Wilderness Rescue or Disaster Rescue or Individual Tracking no animals were displayed I went through and changed how I had my filters structured and double checked that the dog breeds that were in the dashboard specification were listed with their full names which was not the case for Chesapeake Bay Retrievers or Doberman Pinschers both of which had been shortened in the database. After correcting for this the filters were working as expected. I also had to make some changes to my pie chart to make sure that it would change to display the correct information when one of the filters was selected and in the process the pie chart now only displays the animals on the page being viewed which makes for a much less cluttered and easier to understand chart than having all of the animals in the database being displayed all at once.

## Contact

Your name: Gunnar Dulle