# CS 340 README

**Grazioso Salvare Animal Information and Map**

This project queries the animal database for Gravioso Salvare, allowing for filtering and visual display of the information found. Along with the visual graphic display (a pie chart), a map with the physical location of each animal is also displayed. A user can filter results by animal type, including cats, dogs, or all animals and can also interact with the map by zooming out to see more animals or selecting an animal from the table which then focuses the map on this animal.

## Motivation

This project exists to satisfy requirements given by Gravioso Salvare in creating a visual display for animal information in the given database. This site queries animals from the database, and this information is then displayed allowing for user interaction with the data. This will be useful for any analyst or individual curious about animal information in the database.

**Structure for Project**

This website displays a table with animal information with filtering capabilities, a pie chart display, and a map using Python, MongoDB, PyMongo, and the Jupyter Dash Framework. PyMongo allows for easy data manipulation in MongoDB with Python scripting, hence the choice for using PyMongo. The Dash framework targets interactive development directly in Python such as automatic updating upon changes, without needing separate servers or coding languages to handle webpage code. Dash is useful in this case since a user can filter table data or select single records to view. Dash automatically updates the visual displays locally, creating a satisfactory interaction for the user.

## Steps to Complete the Project

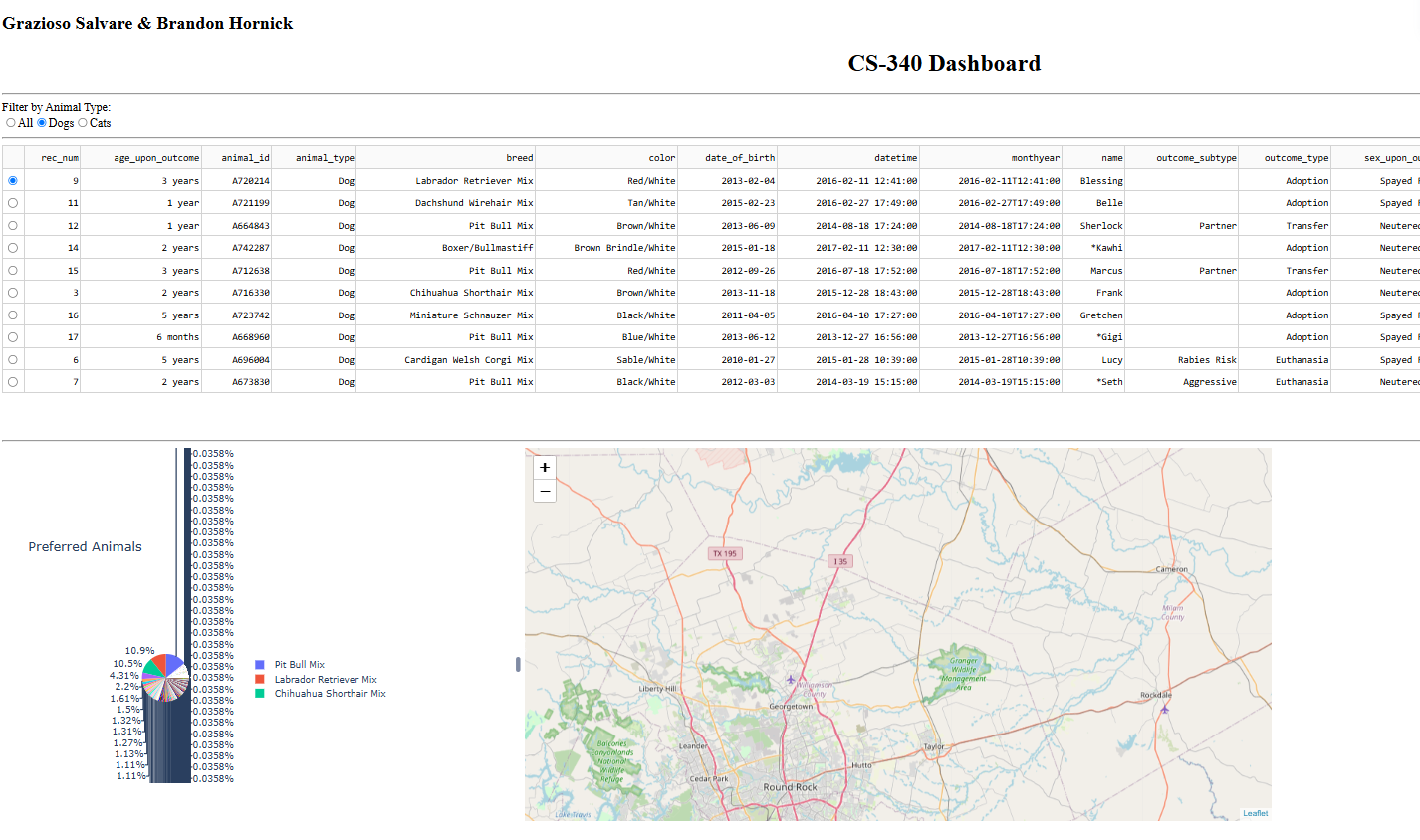
To complete this project the following steps were taken:

1. Create and Setup MongoDB
   1. Create MongoDB Instance
   2. Deploy data to the MongoDB instance
   3. Add credentials for accessing data in database
2. Create Python CRUD Script to Access MongoDB data
   1. Create functions passing in admin credentials for MongoDB manipulation
   2. Implement Create, Read, Update, and Delete functions in script.
   3. Test the CRUD functions in separate script to ensure no issues with CRUD functions
3. Create Python Jupyter Dash Script to host website
   1. Modify Dash Template to access our database with credentials for Read capabilities
   2. Create table based on queried data with filter options (passes in other queries)
   3. Add graphic display, with auto updating capabilities based on table filter
   4. Add map display, again auto updating based on interactions with the table

**Challenges to Overcome**

This project was not without challenges, primarily with errors when trying to display results on the dashboard table. I repeatedly saw errors when modifying data in the table, chaining from the table to both displays. I began by solving some logic issues I had in the graphic display, returning the incorrect information in the update graph function. Upon fixing this, I realized I was still receiving other errors related to the table displaying data. This time I had to drop the ‘id’ column from the results, so the data was compatible with the table. After fixing this, the map not updating fixed itself and the webpage ran well. I had multiple other little issues that were solved with repeated debugging. Thankfully, the framework we used displays detailed errors in widgets in the top right of the screen allowing easy debugging of these simple issues. Overall, when approaching an unfamiliar framework you will have many issues. This is part of the learning process and has led me to become a better developer overall.

### Screenshots



## Contact

Brandon Hornick – [brandon.hornick@snhu.edu](mailto:brandon.hornick@snhu.edu)