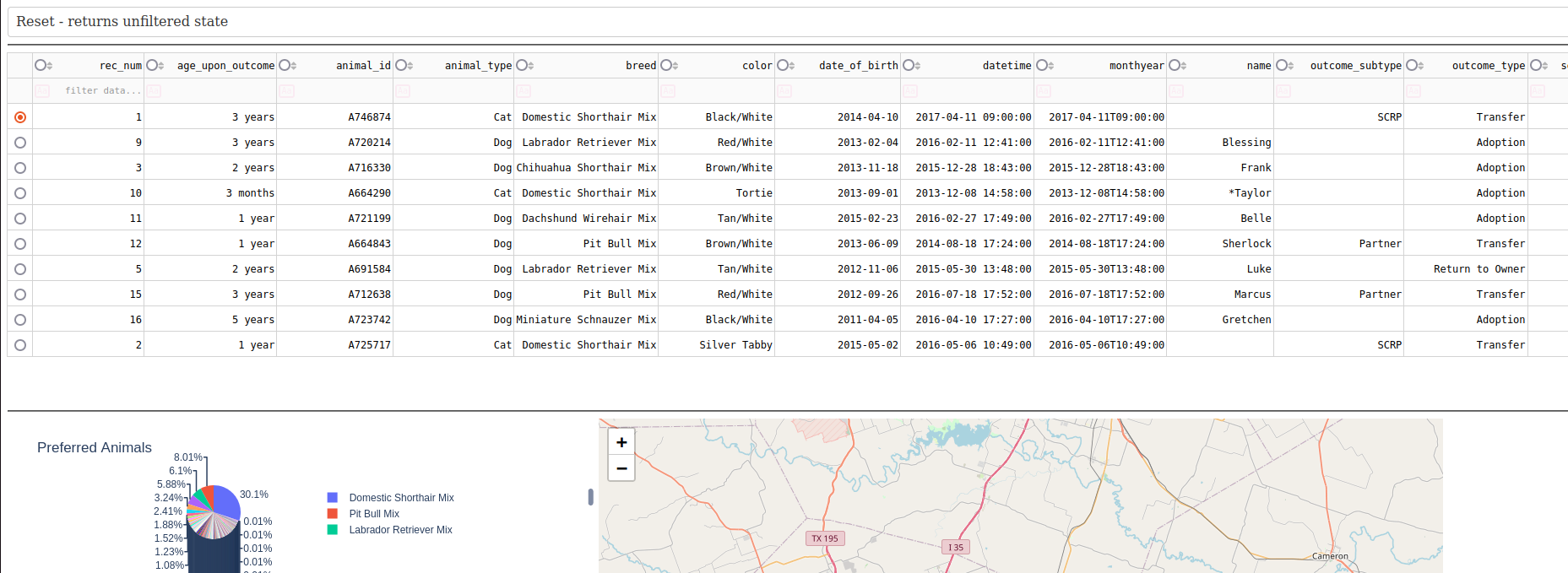
# CS 340 README Template

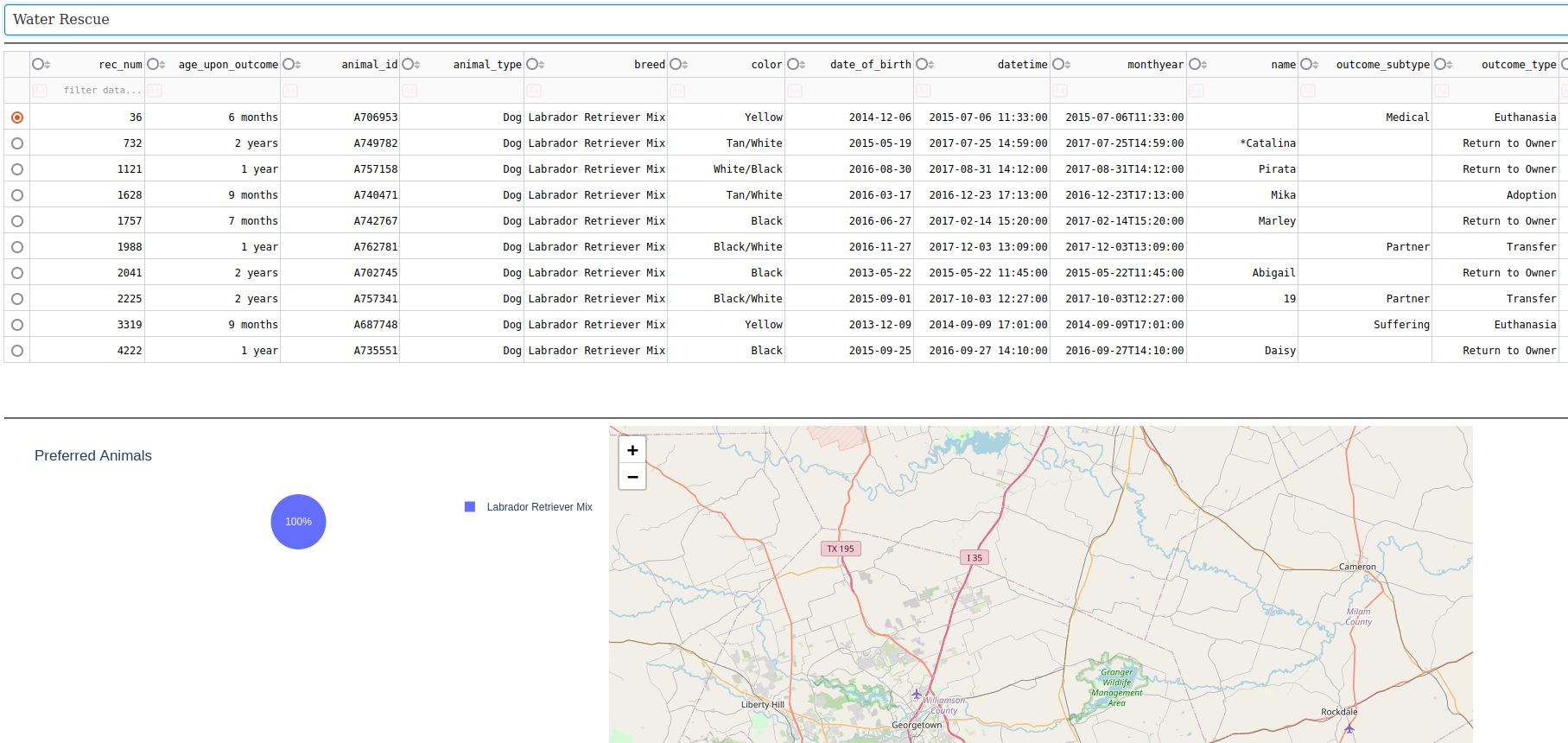
## About the Project/Project Title

Grazioso Salvare Dashboard

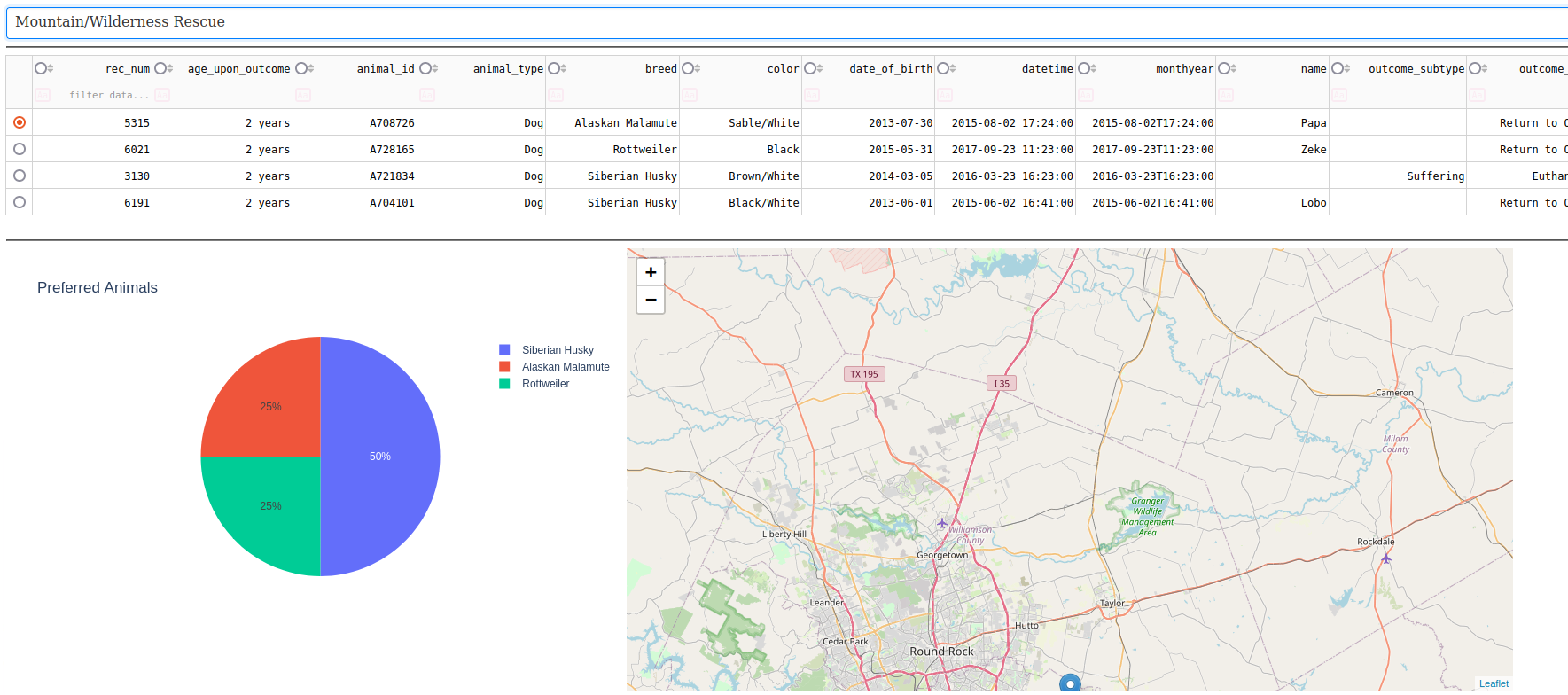
*Default query*

**

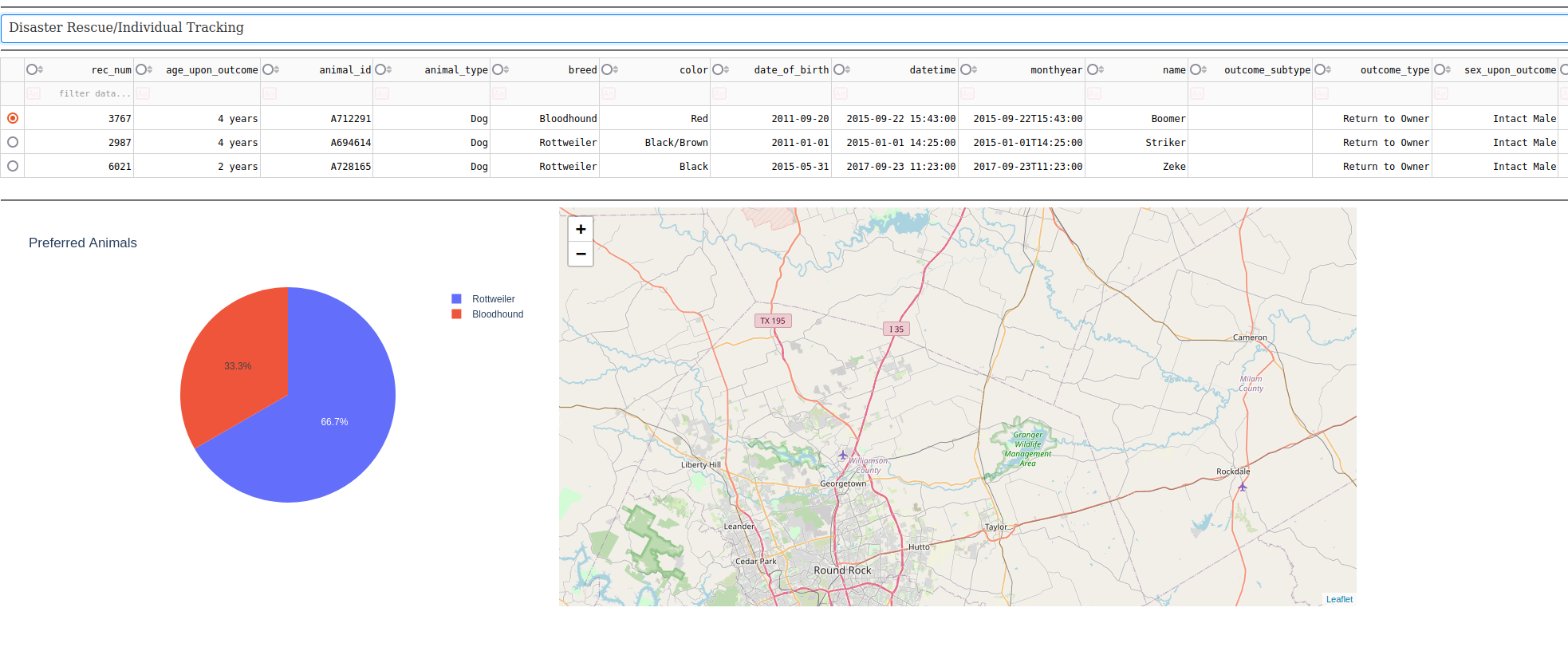
*Water Rescue*

**

*Mountain/Wilderness Rescue*

**

*Disaster Rescue/Individual Training*

**

## Motivation

Provides a CRUD framework for data access and manipulation for the AnimalShelter Dataset for Grazioso Salvare.

# Getting Started

## Requirements

* Requires Python 3.0 or newer.
* MongoDB
* Python Packages
  + jupyter-dash
  + jupyter-plotly-dash
  + matplotlib
  + dash-leaflet
  + pandas
  + numpy
  + dash
* Jupyter Notebook 6.4.8 or newer

## Installation

* project files loaded into Jupyter Notebook

## Usage

shelter.HOST: the db host

shelter.PORT: the port for accessing the db

shelter.DB: the database name

shelter.COL: the collection to access

shelter.connect(username, passwork): for security the credentials are never stored and must be passed with the connect() procedure to create a connection with the database. will return None if there is a connection failure.

shelter.close(): closes the current connection to the db.

shelter.create({data}): return a bool value giving the result of the data being saved to the dataset. Data must match the schema of the dataset. Returns True or False based on if the data was committed correctly.

shelter.read({query}): searches the dataset given the mongo db search criteria. There is no filtration of the data applied to the search criteria. Returns the results of the query.

shelter.update({query}{data}): searches for the for query criteria and applies the data changes provided to the result. returns True or False based on the success of the data update.

shelter.delete({query}): removes the results of the query from the database. Return True or false based on the success of the removal.

## Demo/Front End

The [Dash](https://dash.plotly.com/) framework is used with [Plotly](https://plotly.com/python/plotly-express/) to create the graphs, maps and data tables. Please refer to their individual documentations. Dash was chosen to house the view models used due to its easy to use setup and event response model. With the data being filtered, searched and view changing, Dash allows for a call to action for each event creating the hook to update the view model as needed. [Pandas](https://pandas.pydata.org/) allows for easy formatting of data pulled from an object database like [Mongo](https://www.mongodb.com/docs/), allowing for the data to be tied to a table in Dash for the view model. All frameworks used are in python allowing for a unified coding experience along with consistent data handling. With Mongo user management can be achieved with the ability to provide access control to specific datasets.

in the Demo the data table has preselected search options in a drop down menu for each of the search and rescue programs. The data table allows for further sorting just below each column where the user can type in specific sorting criteria. The map and pie chart will update with the data table filter allowing the user to see focused visuals based on the given criteria. This was designed to give the user accurate data based on the filter criteria on the data table for better decision making from the provided info.

## Project challenges

During the original setup of the database user and access control, there was a case where the ACL was lost on the database for a specific user. This should not have occurred but reapplying the user ACL to the database allowed it to work again. This proved an odd challenge connecting when accessing the database. This led to development of connection validation features in the API, there is now a check that the connection variables are set. There is also feedback when calling the connect procedure to further error handle any failed connections.

## Roadmap/Features (Optional)

* CRUD features
  + Create (Completed)
  + Read (Completed)
  + Update (Completed)
  + Delete (Completed)
* input validation

## Contact

Matthew Courts