# README

# About the Project

This project aimed to develop a functional web-based dashboard for Grazioso Salvare, enabling them to explore animals available at the Austin Animal Shelter to suit their business requirements. The project comprises three layers:

1. A MongoDB layer housing the necessary data for visualization.
2. A Python middleware layer for control.
3. A visualization layer based on Plotly, Dash, and Leaflet to generate HTML.

The Python middleware utilizes a custom class to streamline Create-Read-Update-Delete (CRUD) operations in MongoDB through the Pymongo driver. This class encompasses all the abstracted getter/setter methods required for CRUD operations and MongoDB initialization.

# Component Selection Methodology

MongoDB was chosen for its flexibility and user-friendly dynamic schema, ideal for web environments. Python's extensive toolset, including the Pymongo driver and Plotly-Dash/Dash Leaflet libraries, enables seamless backend-to-frontend development. While MongoDB lacks some SQL ACID properties, the low transaction volume mitigates this concern.

#### CRUD Class

CRUD operations are commonplace when using any database but formatting these queries and any error handling can be tedious. To help alleviate the tedium of formatting each new query or re-code for each error type these operations have been abstracted to simple get and set methods.

The class contains the following methods:

* Constructor (password, username)
* create(data)
* read(criteria = none)
* update(initial, change)
* delete(remove)

And properties:

* records\_updated
* records\_matched
* records\_deleted

For more information on the CRUD class refer to the specific readme.

# Installation/Pre-Conditions

To use this software the following tools must be installed:

* Python 3.6
* Pymongo 4.2: [link](https://pymongo.readthedocs.io/en/stable/)
* MongoDB 4.2: [link](https://www.mongodb.com/)
* Plotly Dash: [link](https://dash.plotly.com/)
* Dash Leaflet: [link](https://dash-leaflet.herokuapp.com/)
* Database admin user account with read/write permissions

## Product Walkthrough

Grazioso Salvare requested the dashboard to have a few key components:

- Branding elements, including the company logo.

- A direct link to their website.

- A visually informative pie chart depicting the distribution of available breeds.

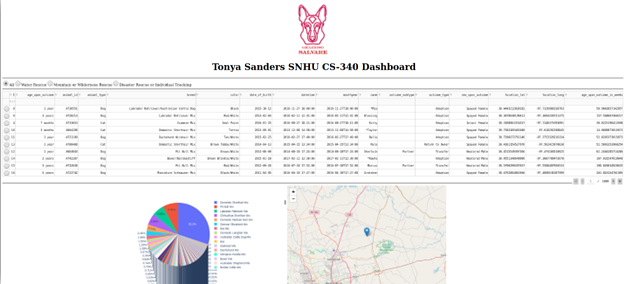
- An interactive map pinpointing the location of selected animals.

- A customizable data table presenting detailed information about the animals housed at the Austin Animal Shelter. Tailored filters enable targeted searches to fulfill specific business needs, such as:

- Water Rescue

- Mountain or Wilderness Rescue

- Disaster Rescue or Individual Tracking



# Challenges:

## The primary challenge encountered during this project revolved around effectively presenting the dataset's information on the dashboard. Outdated initial instructions contributed to confusion, necessitating additional time to pinpoint the source of bugs. The elongated stack traces further complicated the debugging process, requiring meticulous tracking to identify and resolve the issues accurately. Once it was identified that some of the syntax was outdated, it took a bit of time to find the correct updated code but the solutions became clear nonetheless.

## Contact

Tonya Sanders, [tonya.sanders@snhu.edu](mailto:tonya.sanders@snhu.edu)

# Appendix

# Dash callbacks

#############################################

# Interaction Between Components / Controller

#############################################

*@app.callback([Output('datatable-id','data'),*

*Output('datatable-id','columns')],*

*[Input('filter-type', 'value')])*

*def update\_dashboard(filter\_type):*

*#set up an if/else if/else block to respond to the radio buttons*

*if filter\_type == 'All':*

*df = pd.DataFrame.from\_records(shelter.getRecordCriteria({}))*

*elif filter\_type == 'Water':*

*#data isn't that clean, use regex for pattern matching*

*#build the regex patterns for the different filters*

*labRegex = re.compile(\".\*lab.\*\", re.IGNORECASE)*

*chesaRegex = re.compile(\".\*chesa.\*\", re.IGNORECASE)*

*newRegex = re.compile(\".\*newf.\*\", re.IGNORECASE)*

*df = pd.DataFrame.from\_records(shelter.getRecordCriteria({*

*'$or':[ #Regex isn't allowed in an $in helper so use $or*

*{\"breed\": {'$regex': newRegex}}, #pass the regex to the filter*

*{\"breed\": {'$regex': chesaRegex}},*

*{\"breed\": {'$regex': labRegex}},*

*],*

*\"sex\_upon\_outcome\": \"Intact Female\",*

*\"age\_upon\_outcome\_in\_weeks\": {\"$gte\":26.0, \"$lte\":156.0}*

*}))*

*elif filter\_type == 'Mountain':*

*germanRegex = re.compile(\".\*german.\*\", re.IGNORECASE)*

*alaskanRegex = re.compile(\".\*mala.\*\", re.IGNORECASE)*

*oldRegex = re.compile(\".\*old engilish.\*\", re.IGNORECASE)*

*huskyRegex = re.compile(\".\*husk.\*\", re.IGNORECASE)*

*rottRegex = re.compile(\".\*rott.\*\", re.IGNORECASE)*

*df = pd.DataFrame.from\_records(shelter.getRecordCriteria({*

*'$or':[*

*{\"breed\": {'$regex': germanRegex}},*

*{\"breed\": {'$regex': alaskanRegex}},*

*{\"breed\": {'$regex': oldRegex}},*

*{\"breed\": {'$regex': huskyRegex}},*

*{\"breed\": {'$regex': rottRegex}},*

*],*

*\"sex\_upon\_outcome\": \"Intact Male\",*

*\"age\_upon\_outcome\_in\_weeks\": {\"$gte\":26.0, \"$lte\":156.0}*

*}))*

*elif filter\_type == 'Disaster':*

*germanRegex = re.compile(\".\*german.\*\", re.IGNORECASE)*

*goldenRegex = re.compile(\".\*golden.\*\", re.IGNORECASE)*

*bloodRegex = re.compile(\".\*blood.\*\", re.IGNORECASE)*

*doberRegex = re.compile(\".\*dober.\*\", re.IGNORECASE)*

*rottRegex = re.compile(\".\*rott.\*\", re.IGNORECASE)*

*df = pd.DataFrame.from\_records(shelter.getRecordCriteria({*

*'$or':[*

*{\"breed\": {'$regex': germanRegex}},*

*{\"breed\": {'$regex': goldenRegex}},*

*{\"breed\": {'$regex': bloodRegex}},*

*{\"breed\": {'$regex': doberRegex}},*

*{\"breed\": {'$regex': rottRegex}},*

*],*

*\"sex\_upon\_outcome\": \"Intact Male\",*

*\"age\_upon\_outcome\_in\_weeks\": {\"$gte\":20.0, \"$lte\":300.0}*

*}))*

*else:*

*raise Exception(\"Unknown filter\")*

*columns=[{\"name\": i, \"id\": i, \"deletable\": False, \"selectable\": True} for i in df.columns]*

*data=df.to\_dict('records')*

*return (data,columns)*

*#change the color of a selected cell*

*@app.callback(*

*Output('datatable-id', 'style\_data\_conditional'),*

*[Input('datatable-id', 'selected\_columns')]*

*)*

*def update\_styles(selected\_columns):*

*return [{*

*'if': { 'column\_id': i },*

*'background\_color': '#D2F3FF'*

*} for i in selected\_columns]*

*#call back for pie chart*

*#set to plot all of the data across all of the pages instead of the viewable data*

*#change to derived\_viewport\_data if other behavior is wanted*

*@app.callback(*

*Output('graph-id', \"children\"),*

*[Input('datatable-id', \"derived\_virtual\_data\")])*

*def update\_graphs(viewData):*

*dffPie = pd.DataFrame.from\_dict(viewData)*

*return [*

*dcc.Graph(*

*figure = px.pie(dffPie, names='breed',)*

*)*

*]*

*#call back for slecting a row and then plotting the geomarker*

*@app.callback(*

*Output('map-id', \"children\"),*

*[Input('datatable-id', \"derived\_virtual\_selected\_rows\")])*

*def update\_map(virtualRows):*

*#austin Texas is [30.75, -97.48]*

*#create the views*

*if not virtualRows: #build a default view if there are no selected lines*

*markerArray = (30.75,-97.48) #default marker at Austin Animal Shelter*

*toolTip = "Austin Animal Center"*

*popUpHeading = "Austin Animal Center"*

*popUpParagraph = "Shelter Home Location"*

*else: #build the contextual views based on the selection*

*dff = pd.DataFrame(df.iloc[virtualRows]) #convert the datatable to a dataframe*

*coordLat = float(dff['location\_lat'].to\_string().split()[1]) #strip out the lat*

*coordLong = float(dff['location\_long'].to\_string().split()[1]) #strip out the long*

*markerArray = (coordLat, coordLong) #build the array based on selection*

*toolTip = dff['breed']*

*popUpHeading = "Animal Name"*

*popUpParagraph = dff['name']*

*#return the map with a child marker*

*#marker is set to the values found in markerArray*

*#map centers/moves to view the new marker instead of holding a fixed center*

*return [dl.Map(style={'width': '700px', 'height': '450px'}, center=markerArray,*

*zoom=10, children=[dl.TileLayer(id=\"base-layer-id\"),*

*dl.Marker(position=markerArray, children=[*

*dl.Tooltip(toolTip),*

*dl.Popup([*

*html.H1(popUpHeading),*

*html.P(popUpParagraph)*

*])*

*])*

*])*

*]*