



CS 499 README Category 3

About Project

This project creates an interactive dashboard for the data stored in the Austin Animal Center (AAC) database. The dashboard is used to determine which dogs are good candidates for various types of search-and-rescue training. When the user selects what type of dog training, they are looking for the system updates and will display a pie chart which the breed breakdown, and a map with the location of the first animal in the results.

Motivation

This system exists to showcase my growth with the Mongo database, PyMongo, as well as the Dash system to create a web-based dashboard that the client can use.

Getting Started

How to start using it

1. Enter MongoDB Compass and import the csv file "aac_shelter_outcomes.csv"
 - a. Open MongoDB Compass and press the + button, name the database AAC and the collection animals.
 - b. Click Add Data and select Import JSON or CSV file and select the CSV file that can be found at the link under installation.
 - c. Unselect Ignore empty strings, and then Import
2. Create index(es) in MongoDB Compass by selecting Open MongoDB shell:
 - a. Simple Index command: `db.animals.createIndex({"breed":1})`
 - b. Complex Index command: `db.animals.createIndex({"breed":1, "outcome_type":1})`
3. Ensure you have the Logo.png file in the same file directory as the .py and ipynb files
 - a. This image will load Grazioso Salvare logo at the top of the page when the ipynb file is ran
4. Run the .py and ipynb files
 - a. Update your port number in the Project2CRUD.py file, which can be retrieved by opening MongoDB Compass and selecting the ':' and choosing Show Connection information.
5. Click on link
 - a. When you click on the link the dashboard should appear in a new tab. It will have some options at the top, a list of animals, and then a pie chart and a map.

Installation

A list of items you will need prior to using the system:

1. Python: Ensure you have latest version
2. MongoDB Compass: [MongoDB Compass Download \(GUI\) | MongoDB](#)
3. PyMongo: install using pip: `pip install pymongo`
4. CSV file "aac_shelter_outcomes.csv": You can get it from Austin Animal Center Open Data Portal. <https://doi.org/10.26000/025.000001>

Usage

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).



This dashboard has 3 very important sections. The filter options, the interactive data table, and the charts (a Pie chart and a Geolocation Chart)

Filters

To begin the system offers various filters options available to the user based on the information provided in the dashboard specifications document. Each filter has an associated value, that is used to update the interactive data table.

Code Example

```
app.layout = html.Div([
    # Adds the image to the webpage
    html.Div(className='row',
              style={'display' : 'flex'},
              children=[
                html.Img(src='data:image/png;base64,{}'.format(encoded_image.decode()), style={'height':'5%', 'width':'5%'}),
                html.Center(html.B(html.H1('Animal Dashboard')))
              ]),
    html.B(html.H3('Created by Juan Cervantes Ortiz')),
])
```

This code shows the Grazioso Salvare logo at the top of the page.

It also shows the title of the Dashboard 'Animal Dashboard' next to the logo, with 'Created by Juan Cervantes Ortiz' directly under the image.

```
html.Hr(),
html.Div(
    # Add code for the interactive filtering options
    dcc.RadioItems(
        id='filter-type',
        options=[{'label': 'Water Rescue', 'value': 'water'},
                  {'label': 'Mountain/Wilderness Rescue', 'value': 'mount'},
                  {'label': 'Disaster Rescue or Individual Tracking', 'value': 'dis'},
                  {'label': 'Reset', 'value': 'reset'}],
        # this ensure that it is all on one line
        inline=True
    )
)
```

The above code shows the various button options provided, being Water Rescue, Mountain/Wilderness Rescue, Disaster rescue or Individual Tracking, and Reset. When each button is clicked the system will update the interactive table, based on the value associated with the button.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

```
#####
# Interaction Between Components / Controller
#####
@app.callback(Output('datatable-id', 'data'),
              [Input('filter-type', 'value')])
def update_dashboard(filter_type):
    ## FIX ME Add code to filter interactive data table with MongoDB queries

    #water rescues
    if filter_type == 'water':
        df = pd.DataFrame.from_records(animals.read({"animal_type" : "Dog",
            "breed" : {"$in": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]},
            "sex_upon_outcome": "Intact Female",
            "age_upon_outcome_in_weeks": {"$gte":26.0, "$lte":156.0}}))

    #mountain or wilderness
    elif filter_type == 'mount':
        df = pd.DataFrame.from_records(animals.read({"animal_type" : "Dog",
            "breed" : {"$in": ["German Shepherd", "Alaskan Malamute", "Old English Sheepdog",
            "Siberian Husky", "Rottweiler"]},
            "sex_upon_outcome": "Intact Male",
            "age_upon_outcome_in_weeks": {"$gte":26.0, "$lte":156.0}}))

    #disaster or individual tracking
    elif filter_type == 'dis':
        df = pd.DataFrame.from_records(animals.read({"animal_type" : "Dog",
            "breed" : {"$in": ["Doberman Pinscher", "German Shepherd", "Golden Retriever", "Bloodhound",
            "Rottweiler"]},
            "sex_upon_outcome": "Intact Male",
            "age_upon_outcome_in_weeks": {"$gte":20.0, "$lte":300.0}}))

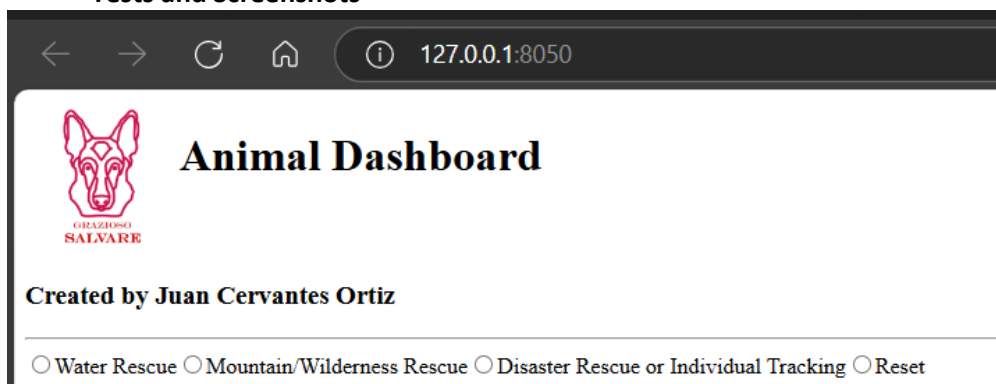
    #rest value
    else:
        df = pd.DataFrame.from_records(animals.read({}))

    columns=[{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns]
    data=df.to_dict('records')

    df.drop(columns=['_id'],inplace=True)
    return df.to_dict('records')
```

This section shows how the different values of the interactive table will be updated. Each time a filter option is selected the system will update the read query to the one that matches the filter type of the button.

Tests and Screenshots



This shows the various filter options. At initialization the system doesn't select any option. Once the user selects an option, the bubble will be red.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

Interactive Data Table

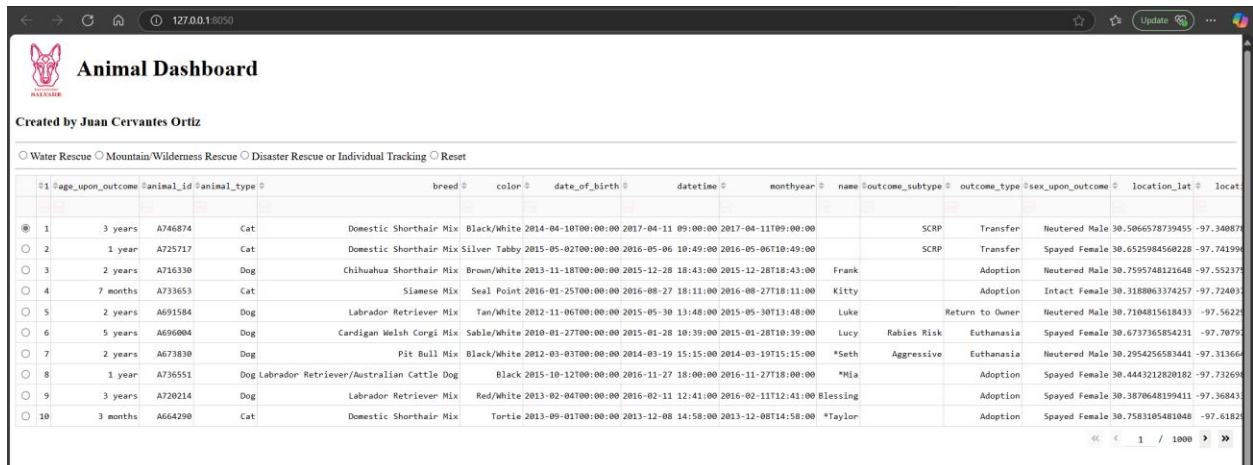
The interactive data table has various functionalities and utilizes the Project2CRUD.py to make read queries. At initialization the system will run a read all query and display the results. When the user uses the various filters the data table will update to show the results for those selected filters.

Code Example

```
html.Hr(),
dash_table.DataTable(id='datatable-id',
                      columns=[{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns],
                      data=df.to_dict('records'),
                      editable=True,
                      filter_action="native",
                      sort_action="native",
                      sort_mode='multi',
                      row_selectable='single',
                      row_deletable=False,
                      selected_rows=[0],
                      page_action='native',
                      page_current= 0,
                      page_size= 10,
                      ),
```

The interactive data table has many useful features for the client. First the table is limited to 10 results on each page. This helps ensure that everything can be seen on one page with minimal scrolling. Second it uses the native sort mode that dash provides. This allows the user to filter based on what they are looking for.

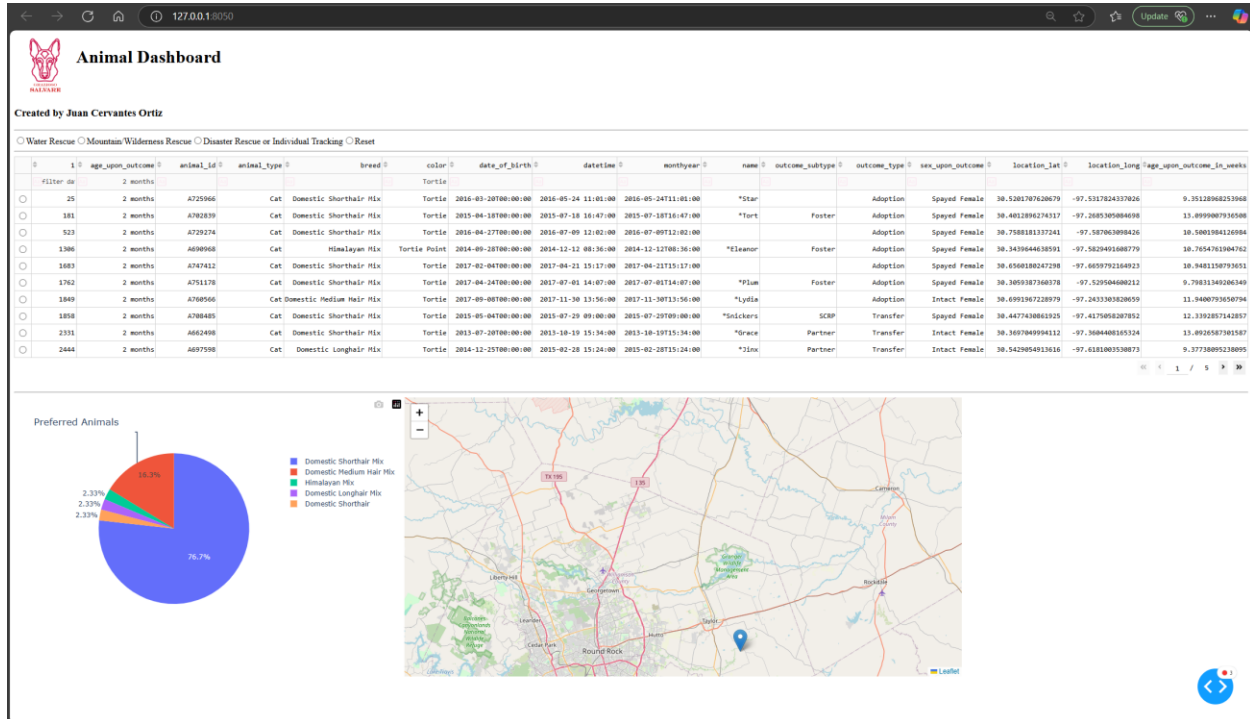
Tests and Screenshots



	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	month/year	name	outcome_subtype	outcome_type	sex_upon_outcome	location_lat	location_lon
1	3 years	A746874	Cat	Domestic Shorthair Mix	Black/White	2014-04-10T00:00:00	2017-04-11 09:00:00	2017-04-11T09:00:00		SCRIP	Transfer	Neutered Male	30.5066578739455	-97.34887
2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02T00:00:00	2016-05-06 10:49:00	2016-05-06T10:49:00		SCRIP	Transfer	Spayed Female	30.6525984560228	-97.74199
3	2 years	A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18T00:00:00	2015-12-28 18:43:00	2015-12-28T18:43:00	Frank		Adoption	Neutered Male	30.7595748121648	-97.55237
4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25T00:00:00	2016-08-27 18:11:00	2016-08-27T18:11:00	Kitty		Adoption	Intact Female	30.3188063374257	-97.72403
5	2 years	A691584	Dog	Labrador Retriever Mix	Tan/White	2012-11-06T00:00:00	2015-05-30 13:48:00	2015-05-30T13:48:00	Luke		Return to Owner	Neutered Male	30.7104815618433	-97.5622
6	5 years	A696004	Dog	Cardigan Welsh Corgi Mix	Sable/White	2010-01-27T00:00:00	2015-01-28 10:39:00	2015-01-28T10:39:00	Lucy	Rabies Risk	Euthanasia	Spayed Female	30.6737365854231	-97.7879
7	2 years	A673830	Dog	Pit Bull Mix	Black/White	2012-03-03T00:00:00	2014-03-19 15:15:00	2014-03-19T15:15:00	*Seth	Aggressive	Euthanasia	Neutered Male	30.2954256583441	-97.31366
8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12T00:00:00	2016-11-27 18:00:00	2016-11-27T18:00:00	*Mia		Adoption	Spayed Female	30.4443212820182	-97.73269
9	3 years	A720214	Dog	Labrador Retriever Mix	Red/White	2013-02-04T00:00:00	2016-02-11 12:41:00	2016-02-11T12:41:00	Blessing		Adoption	Spayed Female	30.3870648199411	-97.36843
10	3 months	A664290	Cat	Domestic Shorthair Mix	Tortie	2013-09-01T00:00:00	2013-12-08 14:58:00	2013-12-08T14:58:00	*Taylor		Adoption	Spayed Female	30.7583105481048	-97.6182

This is an image of what the interactive data table looks like when running the program. It has a row at the top that lists what variable is in that column. On the left most column is selectable bubbles that when pressed will update the geolocation map below to show the location of that specific animal. Only one row can be selected at any given point. Lastly in the bottom right corner of the table is a page selector. You can scroll from page to page, jump to the first or last page, or jump to a specific page number. Each page only displays a maximum of 10 entries to help with visibility.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).



To further help with visibility the user is also able to filter by the value they decide by entering what they are looking for in each column in the second row. In the case above the system is filtering for Cats that are 2 months old and Tortie in color, that are up for adoption. The pie chart and geolocation map will also update.

Charts

The chart functions show the user graphical data. There are two types of charts in this dashboard, a pie chart, and then a geolocation chart. The pie chart is created by using the breed of the read queries, and the geolocation chart displays the location of the selected entry.

Code Example

```
# Display the breeds of animal based on quantity represented in
# the data table on a pie chart
@app.callback(Output('graph-id', "children"),
              [Input('datatable-id', "derived_virtual_data")])
def update_graphs(viewData):
    df = pd.DataFrame.from_dict(viewData)

    return [
        dcc.Graph(
            figure = px.pie(df, names='breed', title='Preferred Animals')
        )
    ]
```

This code sample above shows how the pie chart is created. It takes in the values of breed from the read query and converts them into a pie chart labeled 'Preferred Animals'.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).



```
# This callback will update the geo-location chart for the selected data entry
# derived_virtual_data will be the set of data available from the datatable in the form of
# a dictionary.
# derived_virtual_selected_rows will be the selected row(s) in the table in the form of
# a list. For this application, we are only permitting single row selection so there is only
# one value in the list.
# The iloc method allows for a row, column notation to pull data from the datatable
@app.callback(
    Output('map-id', "children"),
    [Input('datatable-id', "derived_virtual_data"),
     Input('datatable-id', "derived_virtual_selected_rows")]
)
def update_map(viewData, index):
    if viewData is None:
        return
    elif index is None:
        return

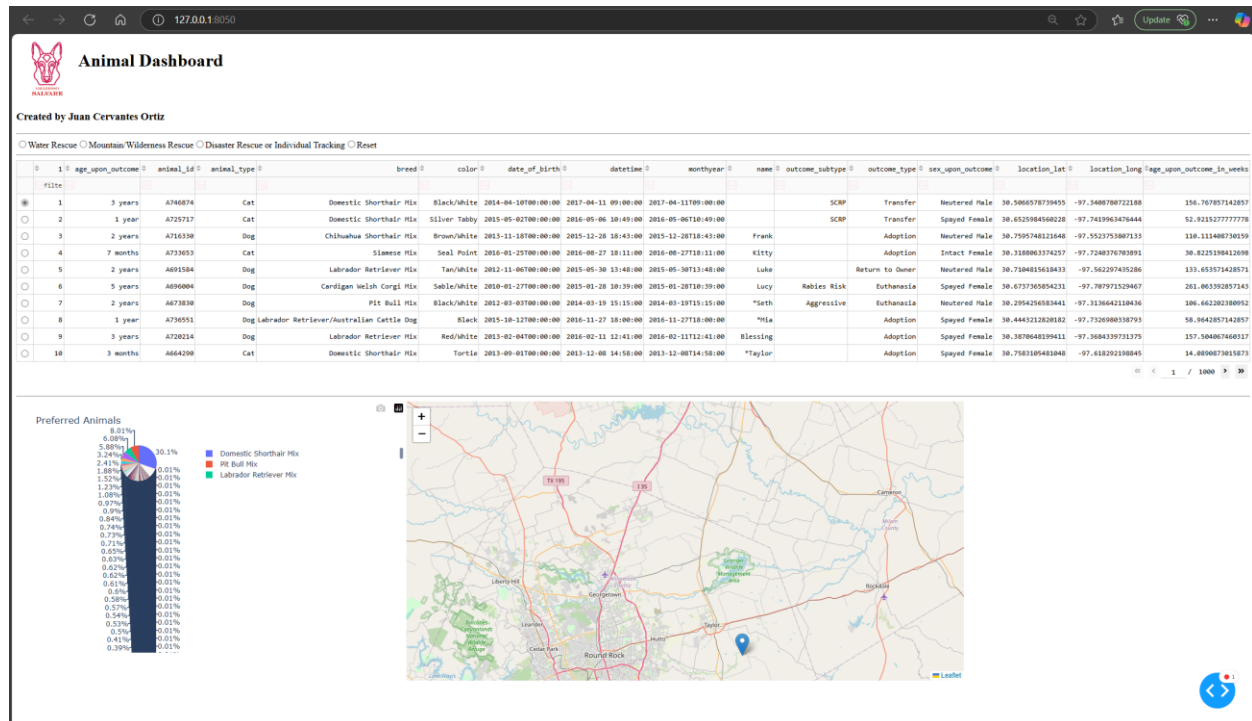
    dff = pd.DataFrame.from_dict(viewData)
    # Because we only allow single row selection, the list can be converted to a row index here
    if index is None:
        row = 0
    else:
        row = index[0]

    # Austin TX is at [30.75,-97.48]
    return [
        dl.Map(style={'width': '1000px', 'height': '500px'}, center=[30.75,-97.48], zoom=10, children=[
            dl.TileLayer(id="base-layer-id"),
            # Marker with tool tip and popup
            # Column 13 and 14 define the grid-coordinates for the map
            # Column 4 defines the breed for the animal
            # Column 9 defines the name of the animal
            dl.Marker(position=[dff.iloc[row,13],dff.iloc[row,14]], children=[
                dl.Tooltip(dff.iloc[row,4]),
                dl.Popup([
                    html.H1("Animal Name"),
                    html.P(dff.iloc[row,9])
                ])
            ])
        ])
    ]
```

This code sample shows how the geolocation map is used and updated. When the user selects an entry from the interactive table the map will update to show the location of the animal. If the user selects the blue pin a pop-up with the animal's name will appear.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

Tests and Screenshots



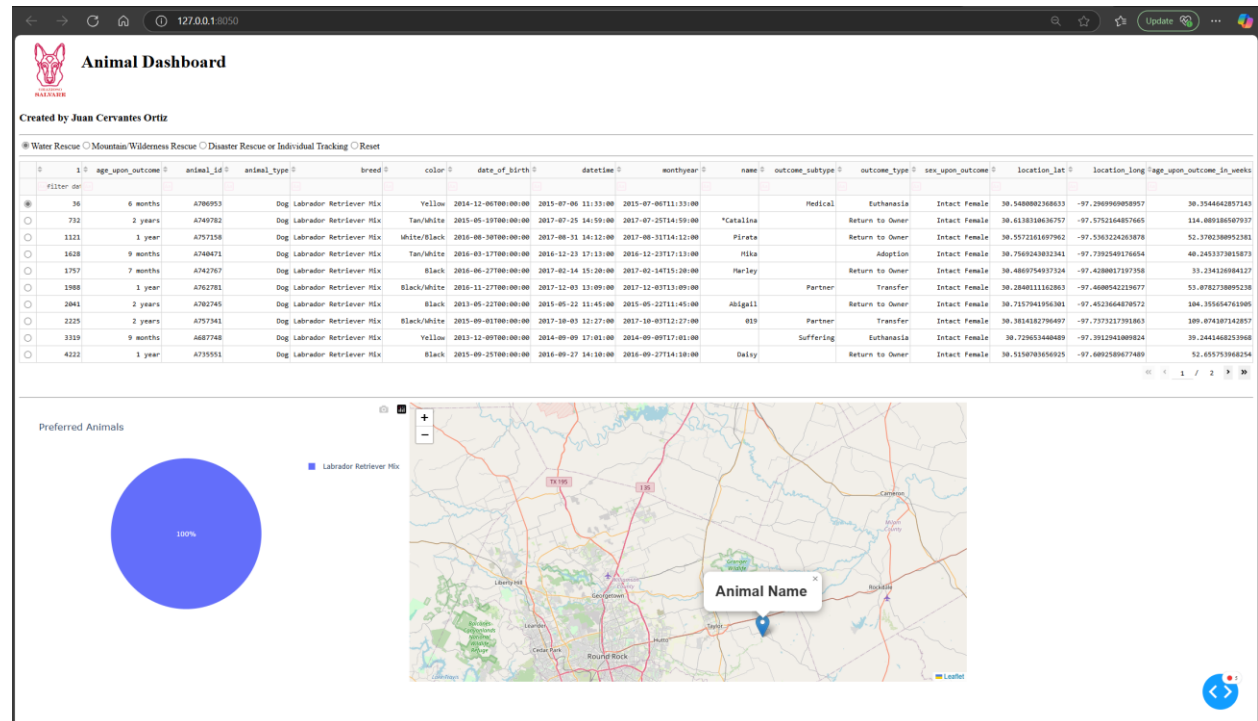
This screen shot shows how the initial pie chart looks like at initialization of the system. The system generates the chart using all 1000 entries in the database. The geolocation map shows the location for the first entry in the database.

System Operation for Each Filter Option

When the user selects a filter from the top, aside from reset, both the pie chart and geolocation map update. In this examples below the system updated the pie chart and map. The map will show the location of the first animal returned in the table.

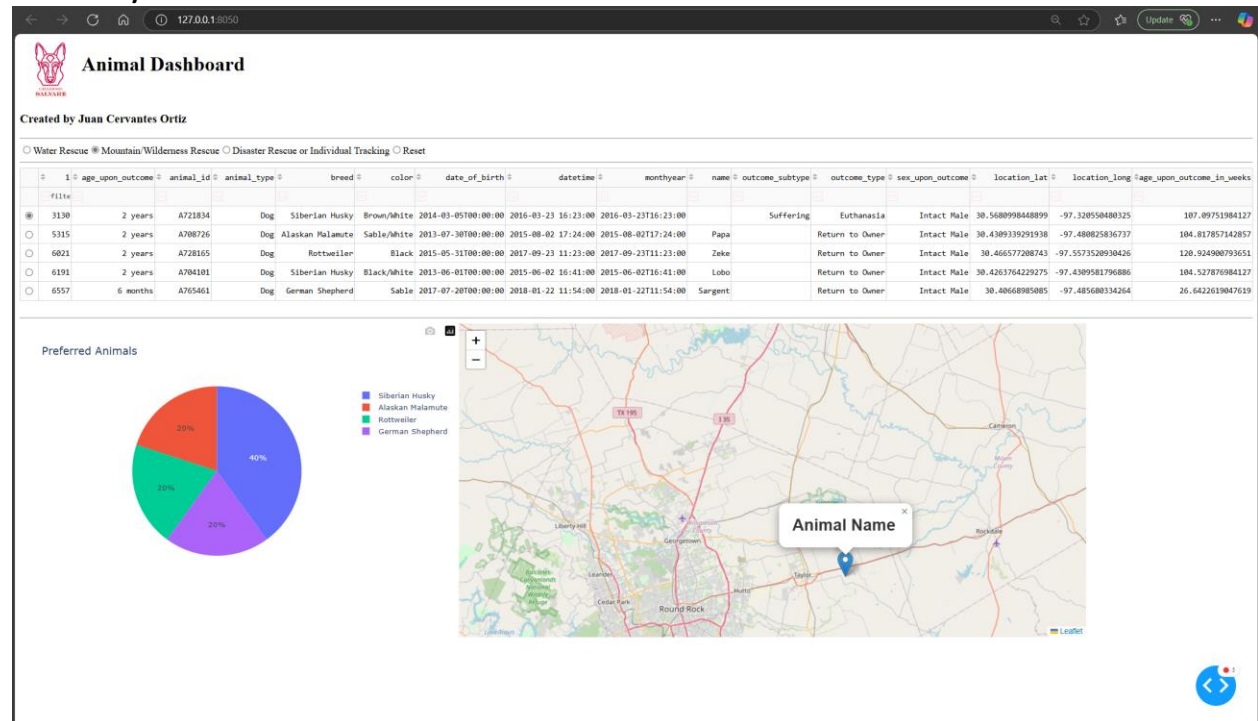
Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

Water Rescue



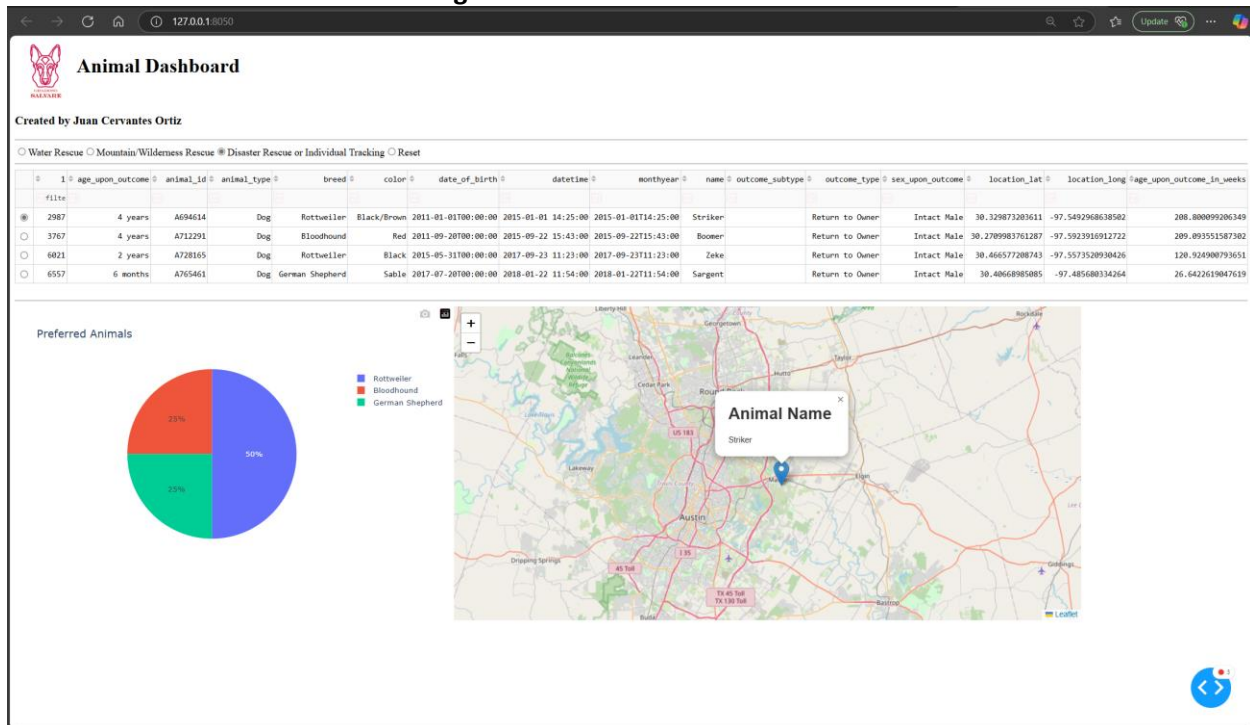
This shows what the system will show when the user selects water rescue as their filter type.

Mountain/Wilderness Rescue



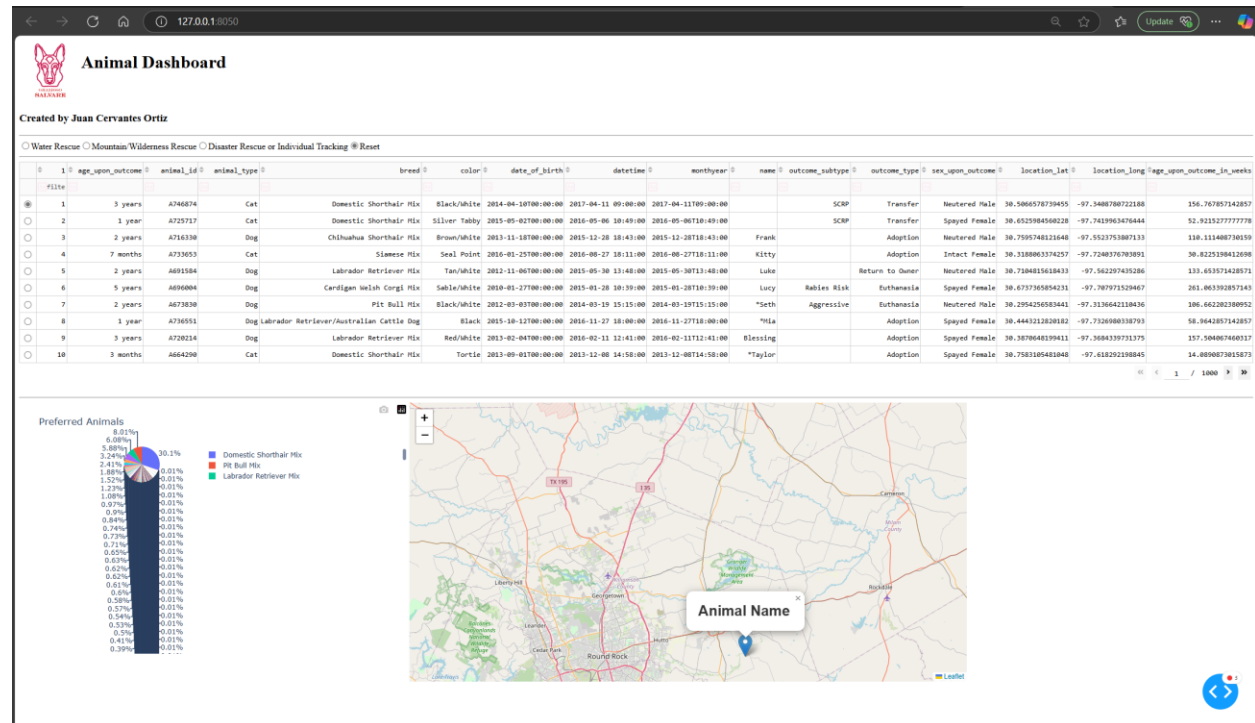
This shows what the system will show when the user selects mountain/wilderness rescue as their filter type.

Disaster Rescue or Individual Tracking



This shows what the system will show when the user selects disaster rescue or individual tracking as their filter type.

Reset



This shows what the system will show when the user selects reset as their filter type. This filter returns the system back to how it was at the initialization of the system.

Common Errors

Some common errors that can occur while using this system stem from improper initialization of the database. Ensure that your username and password are correct if you choose to add verified users. Furthermore ensuring that the proper port number is entered in the Project2CRUD.py file is imperative for the system to work.

This is a web-based dashboard which means that a stable connection is recommended to ensure proper system functionality.

Roadmap/Features

There are various features that could be potentially added in future releases.

1. A way to add, remove, and update entries right from the dashboard: This would make the system a more robust and usable system since the user would no longer need to run various applications or systems to be able to do those sorts of changes.
2. New chart options: Currently the system only supports Pie charts, but future releases could allow the user to select which type of chart or graph works best for them. They could be line charts, or bar graphs. The various filters in these tests had a small amount of results, so the quantities of each dog breed could be easily calculated based on the Pie chart percentages, but if there were hundreds of results other graphs or charts might be best.

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

**Contact**

Juan Cervantes Ortiz

juan.cervantesortiz@snhu.edu

Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).