



CS 340 README Project 2

About Project 2

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 Mongo and import the csv file "aac_shelter_outcomes.csv"
2. Create index(es) in MongoDB:
 - a. Simple Index command: **db.animals.createIndex({"breed":1})**
 - b. Complex Index command: **db.animals.createIndex({"breed":1, "outcome_type":1})**
3. Adding an authenticated user for the AAC database (optional but recommended)
 - a. A user can be created by going into your admin database and running the code:
db.createUser({ user:"aacuser", pwd: "SNHU1234", roles: [{role: "readWrite", db: "AAC"}]})
4. 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
5. Run the .py and ipynb files
 - a. Update your port number in the Project2CRUD.py file, which can be retrieved by running:
printenv | grep -I mongo
6. 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: [Install MongoDB - MongoDB Manual v7.0](#)
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>

Use the command: **cd /usr/local/datasets** to set into the correct directory then use the below command to load in the file to the database

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](#).



```
mongoimport --username="${MONGO_USER}" --password="${MONGO_PASS}" --  
port=${MONGO_PORT} --host=${MONGO_HOST} --db=AAC --collection animals --type=csv --  
headerline --authenticationDatabase admin --drop ./aac_shelter_outcomes.csv
```

Usage

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

```
# Changed the height and width of the image to better fit on the page  
html.Img(src='data:image/png;base64,{}'.format(encoded_image.decode()), style={'height':'10%', 'width':'10%'}),  
html.Center(html.B(html.H1('CS-340 Dashboard'))),  
html.Center(html.B(html.H3('Created by Juan Cervantes Ortiz'))),  
html.Hr(),  
html.Div(  
#FIXME Add in code for the interactive filtering options. For example, Radio buttons, drop down, checkboxes, etc.  
    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  
    )  
)
```

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

It also shows the title of the Dashboard 'CS-340 Dashboard' in the top center of the page, with 'Created by Juan Cervantes Ortiz' directly under it.

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')
    #return (data,columns)
```

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

← → ↻ 127.0.0.1:15885



CS-340 Dashboard

Created by Juan Cervantes Ortiz

☐ Water Rescue ☐ Mountain/Wilderness Rescue ☐ Disaster Rescue or Individual Tracking ☐ Reset

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.

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.

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](#).

Code Example

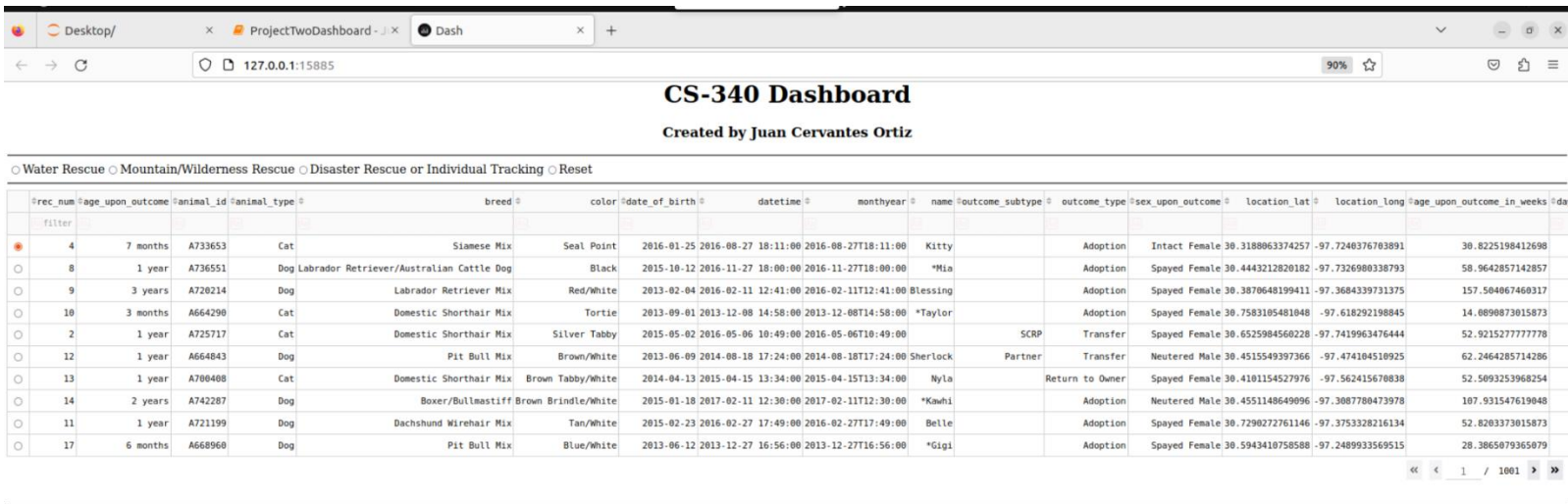
```

),
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'),
    #FIXME: Set up the features for your interactive data table to make it user-friendly for your client
    #If you completed the Module Six Assignment, you can copy in the code you created here
    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,
),
html.Br(),

```

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.

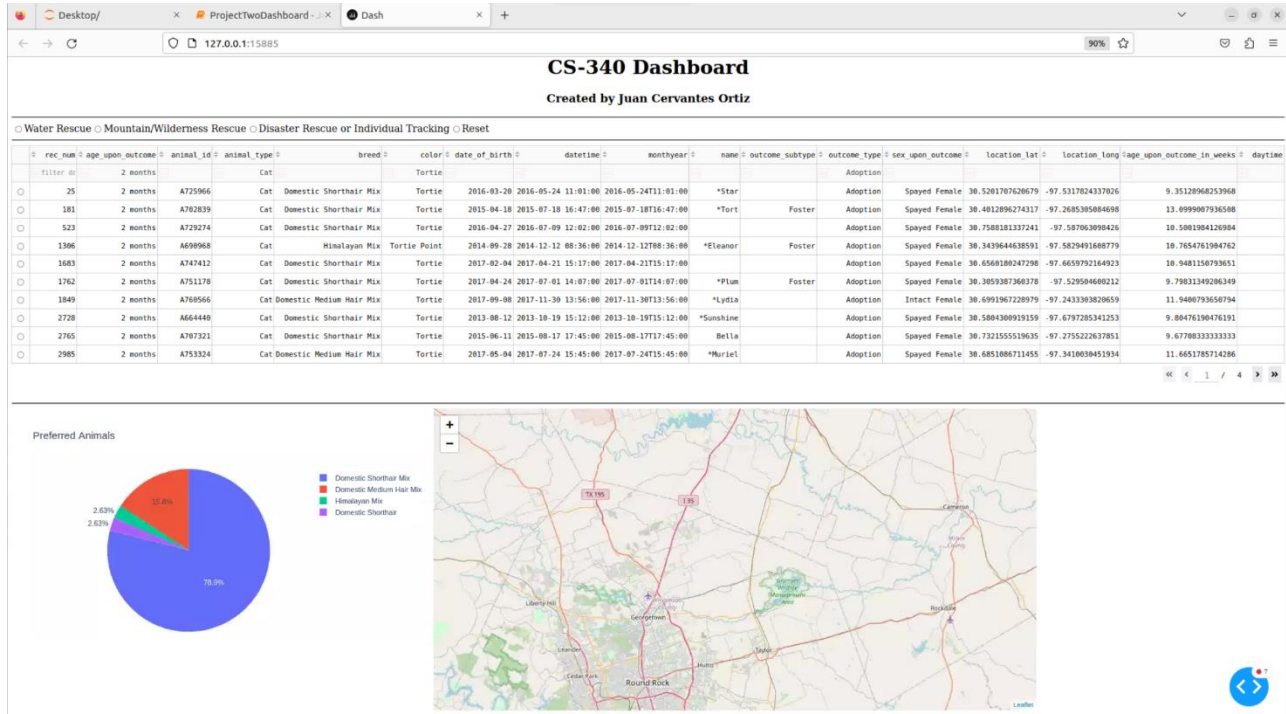
Tests and Screenshots



filter	rec_num	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_long	age_upon_outcome_in_weeks	day
<input checked="" type="radio"/>	4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25	2016-08-27 18:11:00	2016-08-27T18:11:00	Kitty		Adoption	Intact Female	30.3188063374257	-97.7240376703891	30.8225198412698	
<input type="radio"/>	8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12	2016-11-27 18:00:00	2016-11-27T18:00:00	Mia		Adoption	Spayed Female	30.4443212820182	-97.7326980338793	58.9642857142857	
<input type="radio"/>	9	3 years	A720214	Dog	Labrador Retriever Mix	Red/White	2013-02-04	2016-02-11 12:41:00	2016-02-11T12:41:00	Blessing		Adoption	Spayed Female	30.3878648199411	-97.3684339731375	157.584867460317	
<input type="radio"/>	10	3 months	A664298	Cat	Domestic Shorthair Mix	Tortie	2013-09-01	2013-12-08 14:58:00	2013-12-08T14:58:00	Taylor		Adoption	Spayed Female	30.7583185481048	-97.618292198845	14.6898873015873	
<input type="radio"/>	2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05-06T10:49:00		SCRIP	Transfer	Spayed Female	30.6525984560228	-97.7419963476444	52.9215277777778	
<input type="radio"/>	12	1 year	A664843	Dog	Pit Bull Mix	Brown/White	2013-06-09	2014-08-18 17:24:00	2014-08-18T17:24:00	Sherlock	Partner	Transfer	Neutered Male	30.4515549397366	-97.474104510925	62.2464285714286	
<input type="radio"/>	13	1 year	A708408	Cat	Domestic Shorthair Mix	Brown Tabby/White	2014-04-13	2015-04-15 13:34:00	2015-04-15T13:34:00	Nyla		Return to Owner	Spayed Female	30.4101154527976	-97.562415670838	52.5093253968254	
<input type="radio"/>	14	2 years	A742287	Dog	Boxer/Bulmastiff	Brown Brindle/White	2015-01-18	2017-02-11 12:30:00	2017-02-11T12:30:00	Kawhi		Adoption	Neutered Male	30.4551148649096	-97.3087780473978	107.931547619048	
<input type="radio"/>	11	1 year	A721199	Dog	Dachshund Wirehair Mix	Tan/White	2015-02-23	2016-02-27 17:49:00	2016-02-27T17:49:00	Belle		Adoption	Spayed Female	30.7290272761146	-97.3753328216134	52.8203373015873	
<input type="radio"/>	17	6 months	A668960	Dog	Pit Bull Mix	Blue/White	2013-06-12	2013-12-27 16:56:00	2013-12-27T16:56:00	Gigi		Adoption	Spayed Female	30.5943410758588	-97.2489933569515	28.3865079365079	

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
@app.callback(Output('graph-id', "children"),
              [Input('datatable-id', "derived_virtual_data")])
def update_graphs(viewData):
    ###FIX ME ###
    # add code for chart of your choice (e.g. pie chart) #
    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

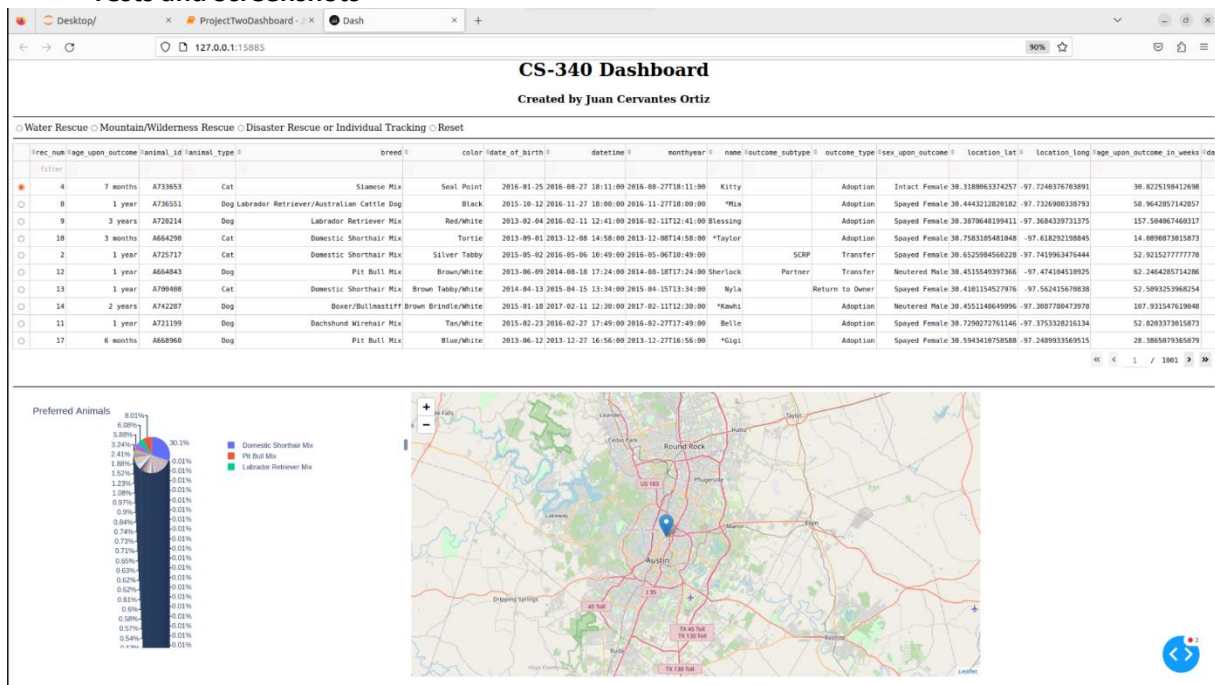
    dff = pd.DataFrame.from_dict(viewData)
    # Because we only allow single row selection, the list can be converted to a row index here
    if index is None:
        row = 0
    else:
        row = index[0]

    # 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])
                ])
            ])
        ])
    ]
```

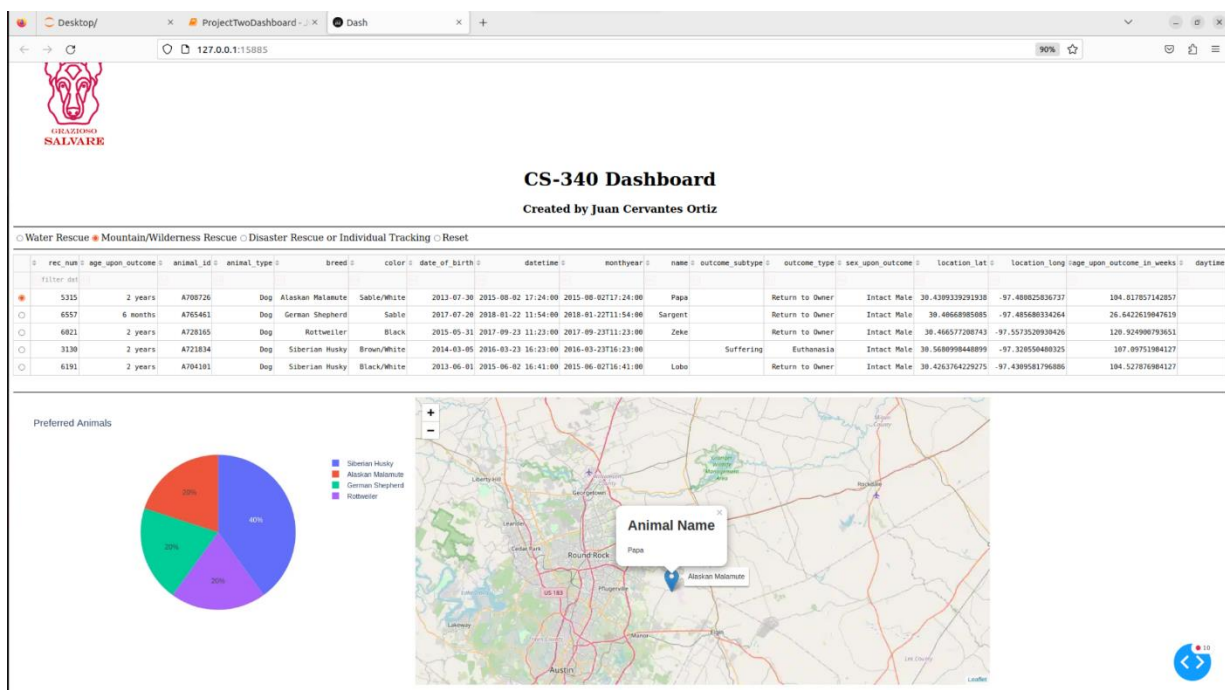
This code sample shows how the geolocation map is used and updated. When the user selects an entry from the interactive table the map will update to show the location of the animal. If the user selects the blue pin a pop-up with the animal's name will appear.

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 1001 entries in the database. The geolocation map shows the location for the first entry in the database.

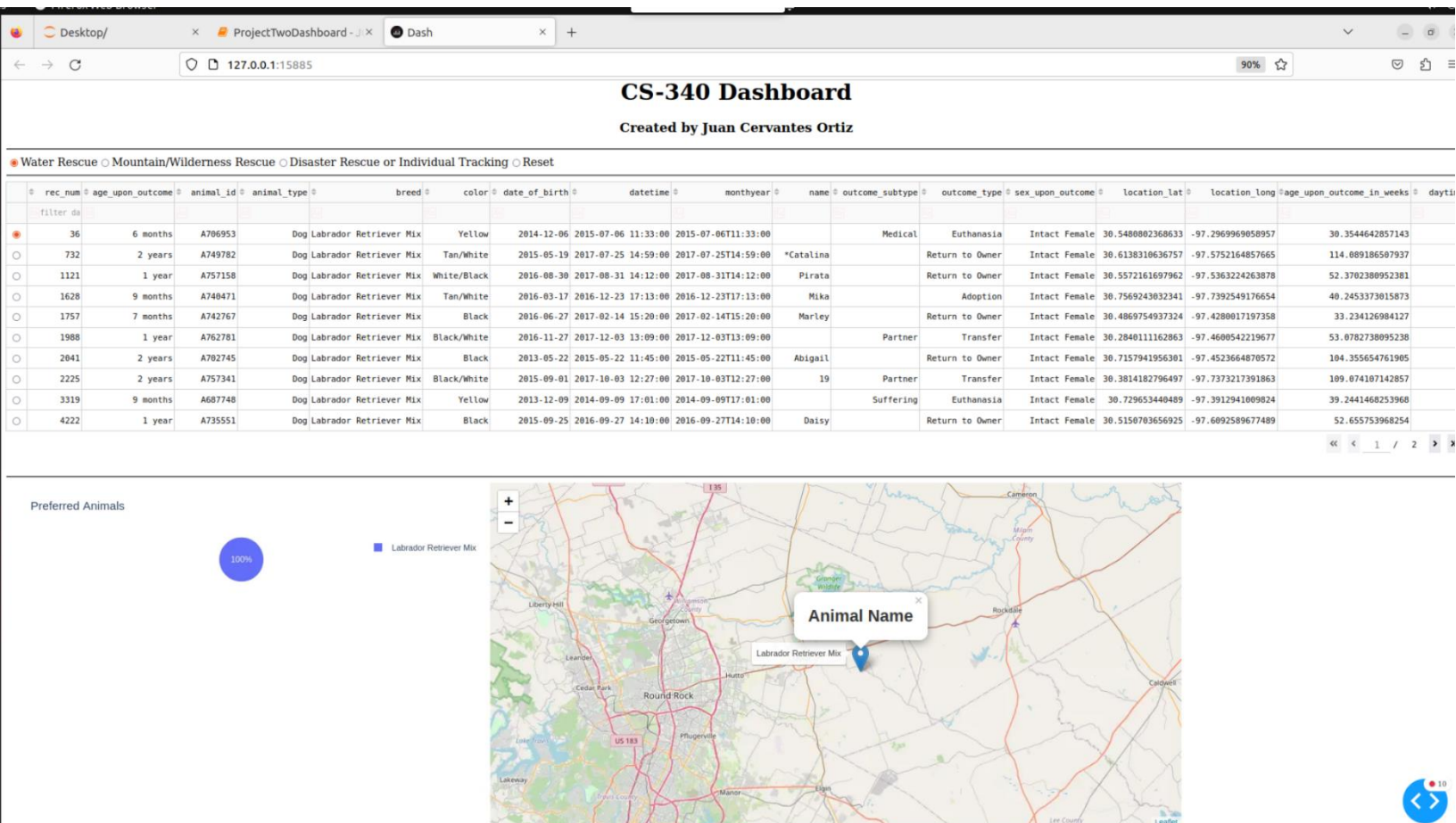


When the user selects a filter from the top, aside from reset, both the pie chart and geolocation map update. In this example when the user selects the Mountain/Wilderness Rescue filter we get 5 results in our interactive table, but the pie chart only shows the breakdown of our results based on breed. So 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](#).

instead of showing us the data for 1001 entries, it only shows us the data for the 5 returned. The geolocation map also updates to show the current location of the first entry in the query which is an Alaskan Malamute named Papa.

System Operation for Each Filter Option

Water Rescue



The screenshot displays the CS-340 Dashboard, created by Juan Cervantes Ortiz. The dashboard is titled "CS-340 Dashboard" and "Created by Juan Cervantes Ortiz". It features a navigation bar with options: "Water Rescue", "Mountain/Wilderness Rescue", "Disaster Rescue or Individual Tracking", and "Reset". The main content area shows a table of animal rescue data with columns: rec_num, age_upon_outcome, animal_id, animal_type, breed, color, date_of_birth, datetime, monthyear, name, outcome_subtype, outcome_type, sex_upon_outcome, location_lat, location_long, age_upon_outcome_in_weeks, and dayti. The table lists 10 animals, including Labrador Retrievers and a Pirata. Below the table, there is a section titled "Preferred Animals" showing a map of the area around Round Rock, Texas. A blue pin indicates the location of a "Labrador Retriever Mix" named "Animal Name".

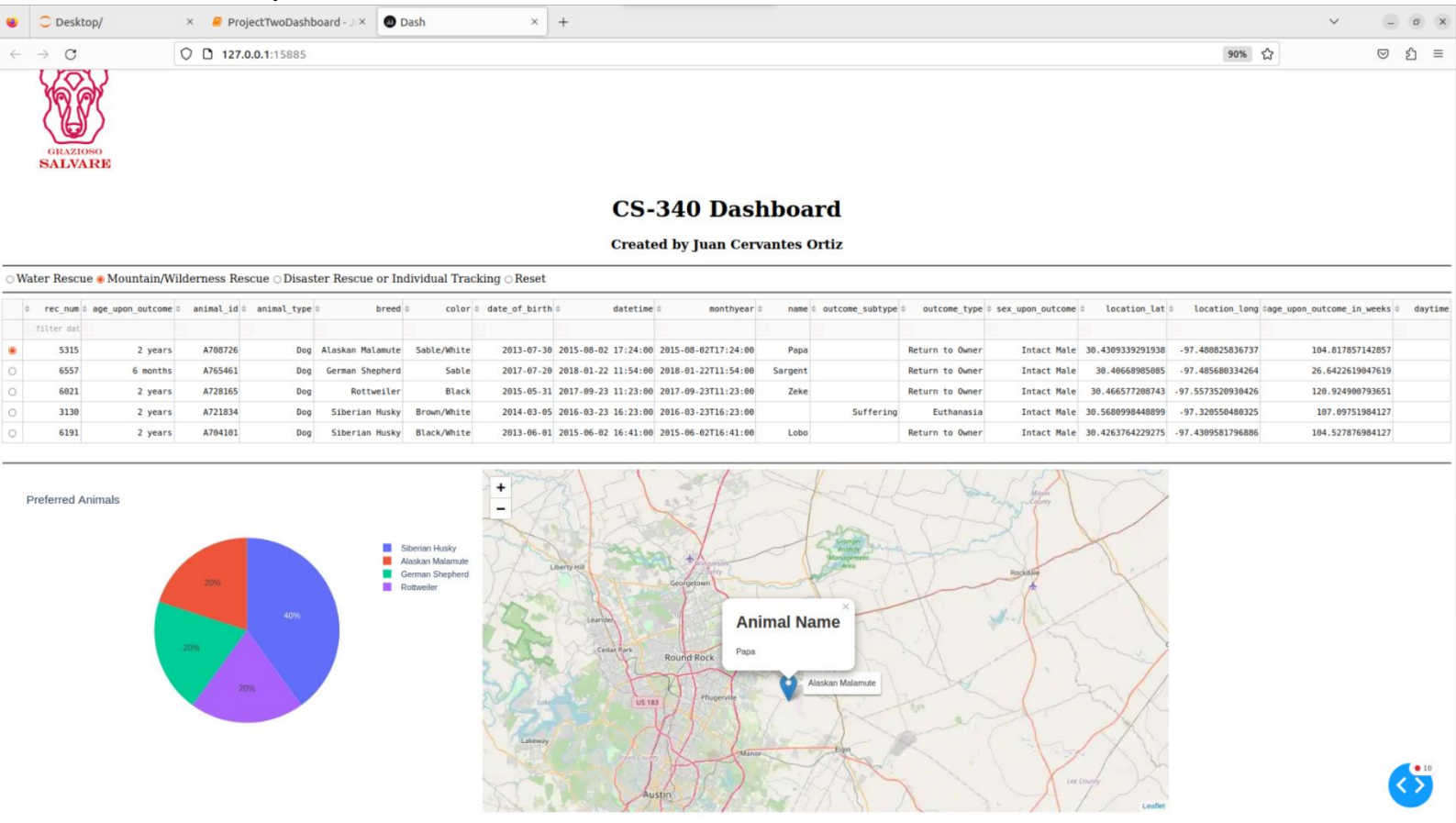
rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	outcome_type	sex_upon_outcome	location_lat	location_long	age_upon_outcome_in_weeks	dayti
36	6 months	A706953	Dog	Labrador Retriever Mix	Yellow	2014-12-06	2015-07-06 11:33:00	2015-07-06T11:33:00		Medical	Euthanasia	Intact Female	30.5480802368633	-97.296996050957	30.3544642857143	
732	2 years	A749782	Dog	Labrador Retriever Mix	Tan/White	2015-05-19	2017-07-25 14:59:00	2017-07-25T14:59:00	*Catalina	Return to Owner	Intact Female	Intact Female	30.6138310636757	-97.5752164857665	114.089186507937	
1121	1 year	A757158	Dog	Labrador Retriever Mix	White/Black	2016-08-30	2017-08-31 14:12:00	2017-08-31T14:12:00	Pirata	Return to Owner	Intact Female	Intact Female	30.5572161697962	-97.5363224263878	52.3782380952381	
1628	9 months	A748471	Dog	Labrador Retriever Mix	Tan/White	2016-03-17	2016-12-23 17:13:00	2016-12-23T17:13:00	Mika	Adoption	Intact Female	Intact Female	30.7569243832341	-97.7392549176654	40.2453373815873	
1757	7 months	A742767	Dog	Labrador Retriever Mix	Black	2016-06-27	2017-02-14 15:20:00	2017-02-14T15:20:00	Marley	Return to Owner	Intact Female	Intact Female	30.4869754937324	-97.4280017197358	33.234126984127	
1988	1 year	A762781	Dog	Labrador Retriever Mix	Black/White	2016-11-27	2017-12-03 13:09:00	2017-12-03T13:09:00		Partner	Transfer	Intact Female	30.2840111162863	-97.4600542219677	53.0782738095238	
2041	2 years	A792745	Dog	Labrador Retriever Mix	Black	2013-05-22	2015-05-22 11:45:00	2015-05-22T11:45:00	Abigail	Return to Owner	Intact Female	Intact Female	30.7157941956381	-97.4523664870572	104.355654761905	
2225	2 years	A757341	Dog	Labrador Retriever Mix	Black/White	2015-09-01	2017-10-03 12:27:00	2017-10-03T12:27:00	19	Partner	Transfer	Intact Female	30.3814182796497	-97.7373217391863	109.074187142857	
3319	9 months	A687748	Dog	Labrador Retriever Mix	Yellow	2013-12-09	2014-09-09 17:01:00	2014-09-09T17:01:00		Suffering	Euthanasia	Intact Female	30.729653440489	-97.3912941009824	39.2441468253968	
4222	1 year	A735551	Dog	Labrador Retriever Mix	Black	2015-09-25	2016-09-27 14:10:00	2016-09-27T14:10:00	Daisy	Return to Owner	Intact Female	Intact Female	30.5150703656925	-97.6092589677489	52.655753968254	

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

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](#).



Mountain/Wilderness Rescue



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

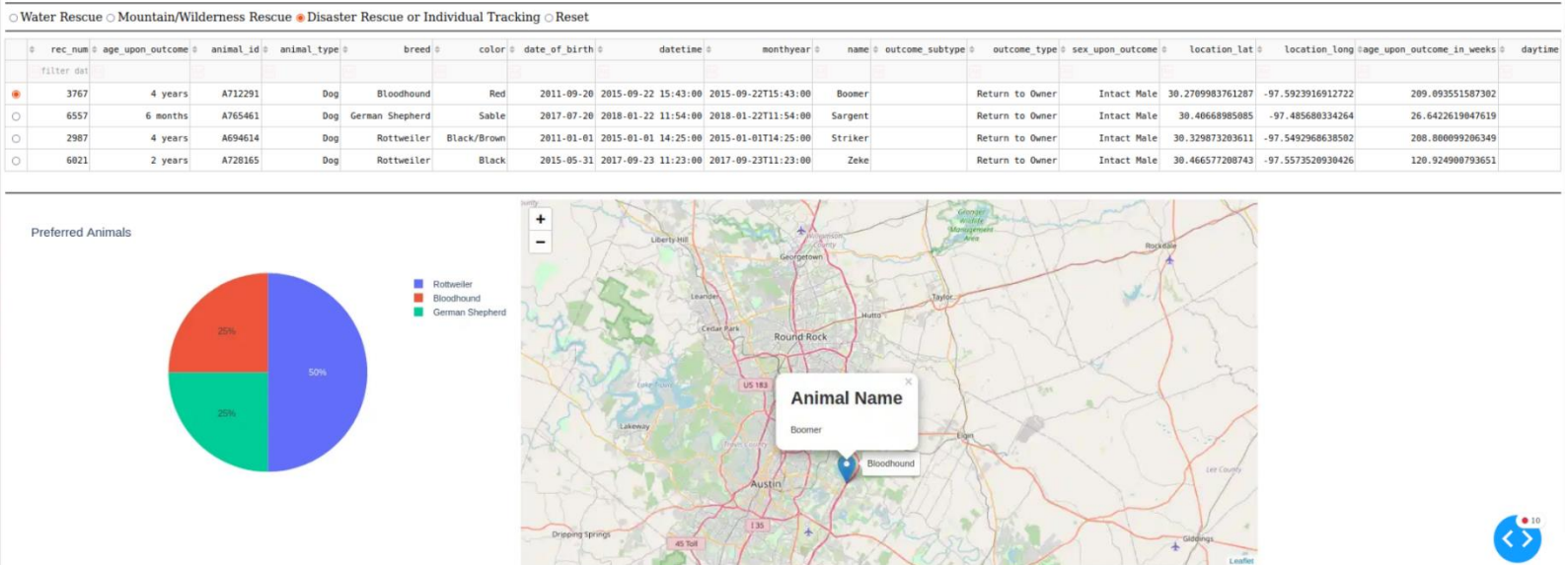
Note: This template has been adapted from the following sample templates: [Make a README](#), [Best README Template](#), and [A Beginners Guide to Writing a Kickass README](#).

Disaster Rescue or Individual Tracking



CS-340 Dashboard

Created by Juan Cervantes Ortiz

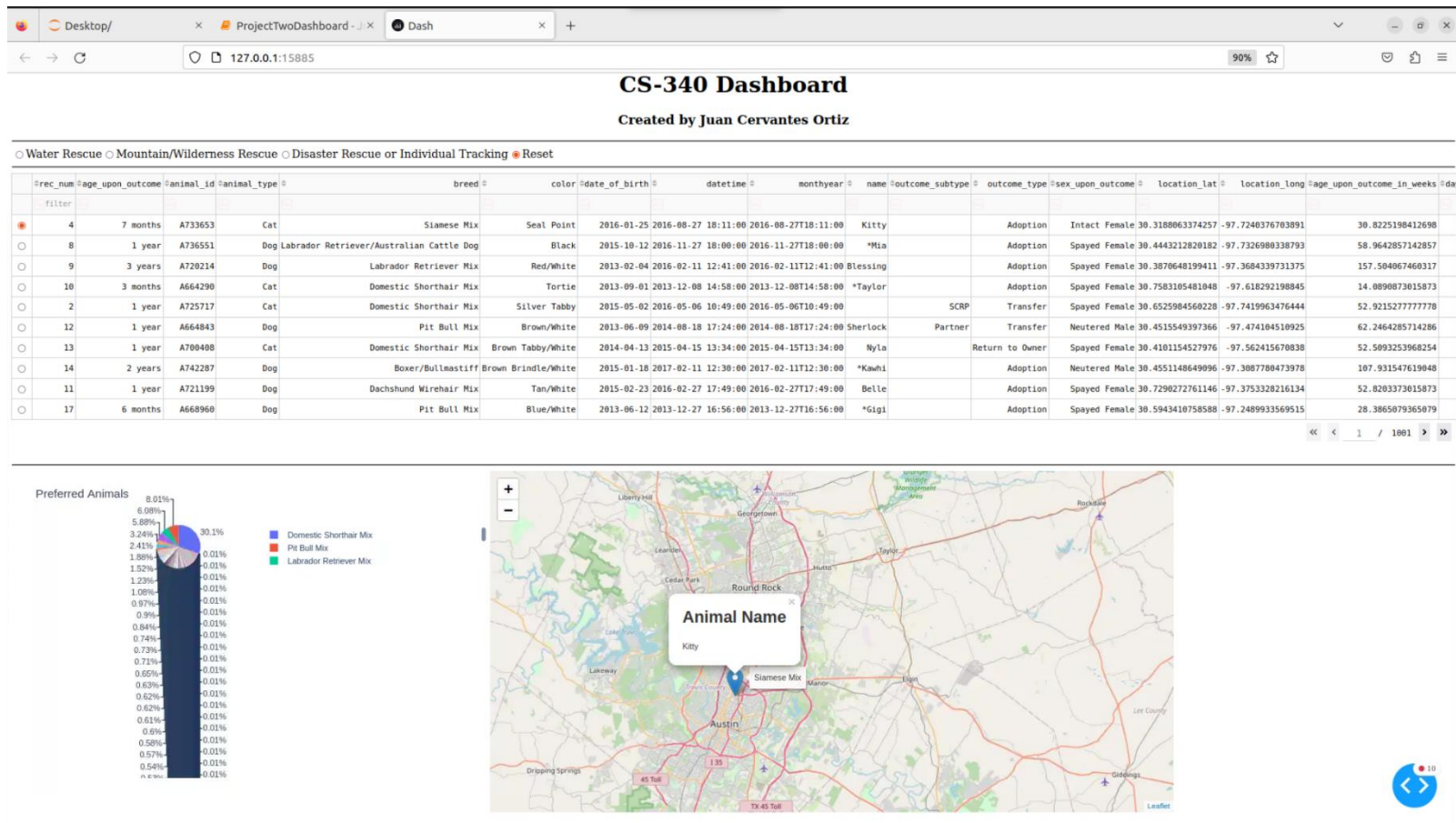


This shows what the system will show when the user selects disaster rescue or individual tracking as their filter type.

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](#).



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. 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

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](#).



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.

Contact

Juan Cervantes Ortiz

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](#).