Grazioso Salvare's Dashboard README Template

About the Project

This milestone is titled "Grazioso Salvare's Dashboard". It is the last milestone for the full stack software application project that we have been developing for our client, Grazioso Salvare. We completed the back-end development with the CRUD Python Module milestone at an earlier stage in the development process. This milestone imports the CRUD Python Module into a client-facing web application dashboard that allows our client to work with data imported from the Austin Animal Center's database (AAC).

Requirements

The following are a list of our client's requirements for the dashboard:

- The Grazioso Salvare logo that includes a URL anchor tag to the client's homepage: www.snhu.edu. The screenshots below show this logo at the top of the dashboard.
- A unique identifier that credits the creator of the dashboard. The developer's name and link to Global Rain is placed at the bottom of the dashboard. This can also be seen in the screenshots below.
- Buttons that provide interactive filter options to filter the AAC data by preferred dogs that are best fit for the three different rescue types provided by our client. The table below provides the dog specifications for each of these rescue types.

Table 1: Rescue Type and Preferred Dog Characteristics

Rescue Type	Preferred Breeds	Preferred Sex	Training Age*
Water	Labrador Retriever Mix, Chesapeake Bay Retriever, Newfoundland	Intact Female	26 weeks to 156 weeks
Mountain or Wilderness	German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler	Intact Male	26 weeks to 156 weeks
Disaster or Individual Tracking	Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler	Intact Male	20 weeks to 300 weeks

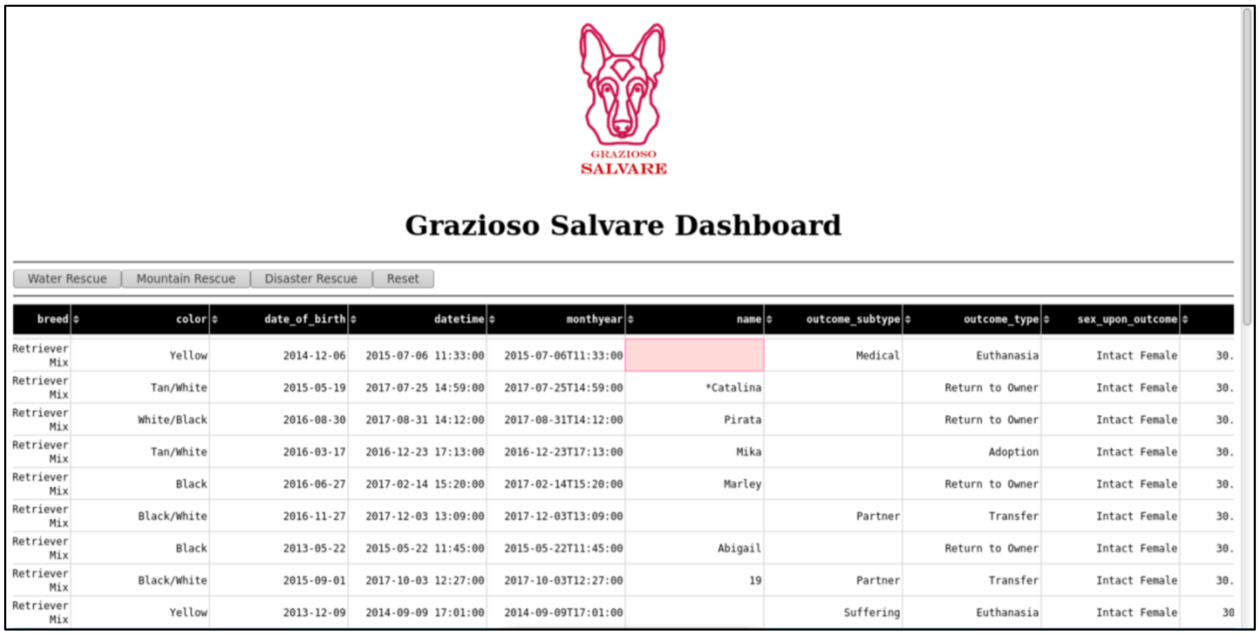
- The screenshots below show a button for each of the three rescue types at the top left corner of the dashboard. For a clean and simple UI experience, we used only one adjective in the button label for each rescue type. For example, the "mountain or wilderness" rescue type is shortened to "Mountain Rescue". There is a fourth filter option that serves as a reset button. This reset button allows all widgets to return to their original, unfiltered state.
- A data table that dynamically responds to the filtering options. Screenshots 1 through 4 below shows the table's response each time a rescue-type button is pressed.
- A geolocation and pie chart that dynamically responds to the filtering options.

Screenshots

The following screenshots display the interactive activities when each button is pressed.

1) Water Rescue: The first screenshot displays the filtered data table showing all of the dogs that meet the requirements for water rescue. The second screenshot displays the geolocation of the first entry of this filter; a yellow Labrador Retriever Mix with no name reported.

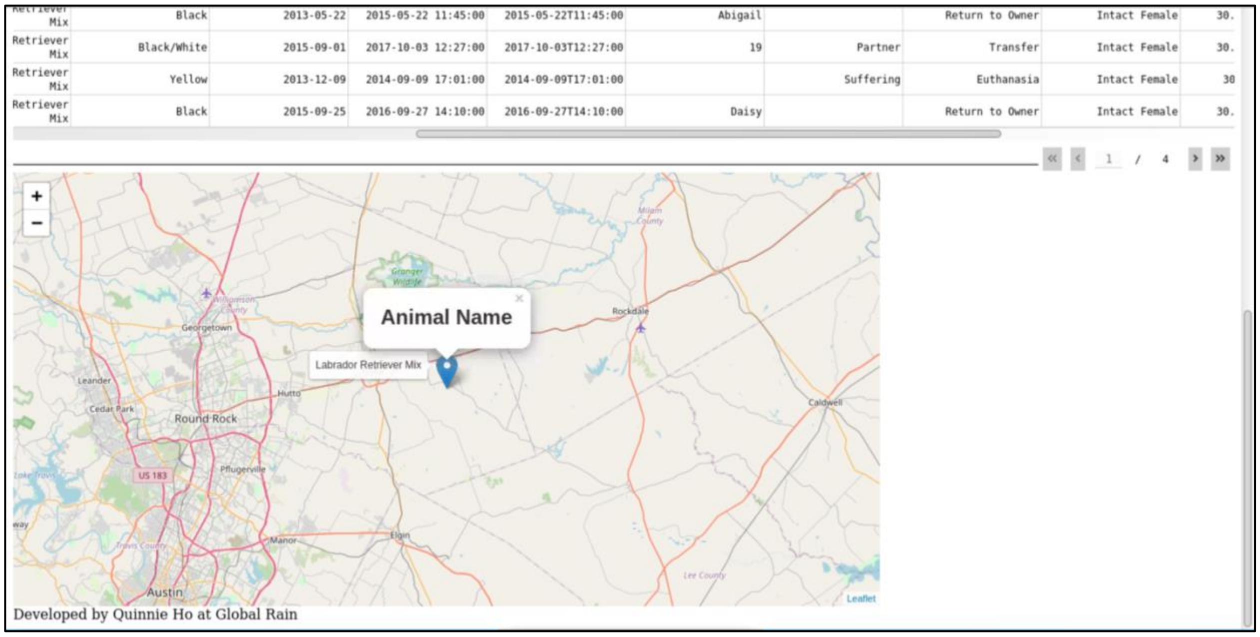
Screenshot #1: Water Rescue Filtered Data Table



Grazioso Salvare Dashboard

breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	outcome_type	sex_upon_outcome	
Retriever Mix	Yellow	2014-12-06	2015-07-06 11:33:00	2015-07-06T11:33:00		Medical	Euthanasia	Intact Female	30.
Retriever Mix	Tan/White	2015-05-19	2017-07-25 14:59:00	2017-07-25T14:59:00	*Catalina		Return to Owner	Intact Female	30.
Retriever Mix	White/Black	2016-08-30	2017-08-31 14:12:00	2017-08-31T14:12:00	Pirata		Return to Owner	Intact Female	30.
Retriever Mix	Tan/White	2016-03-17	2016-12-23 17:13:00	2016-12-23T17:13:00	Mika		Adoption	Intact Female	30.
Retriever Mix	Black	2016-06-27	2017-02-14 15:20:00	2017-02-14T15:20:00	Marley		Return to Owner	Intact Female	30.
Retriever Mix	Black/White	2016-11-27	2017-12-03 13:09:00	2017-12-03T13:09:00		Partner	Transfer	Intact Female	30.
Retriever Mix	Black	2013-05-22	2015-05-22 11:45:00	2015-05-22T11:45:00	Abigail		Return to Owner	Intact Female	30.
Retriever Mix	Black/White	2015-09-01	2017-10-03 12:27:00	2017-10-03T12:27:00	19	Partner	Transfer	Intact Female	30.
Retriever Mix	Yellow	2013-12-09	2014-09-09 17:01:00	2014-09-09T17:01:00		Suffering	Euthanasia	Intact Female	30

Screenshot #2: Water Rescue Filtered Geolocation Map



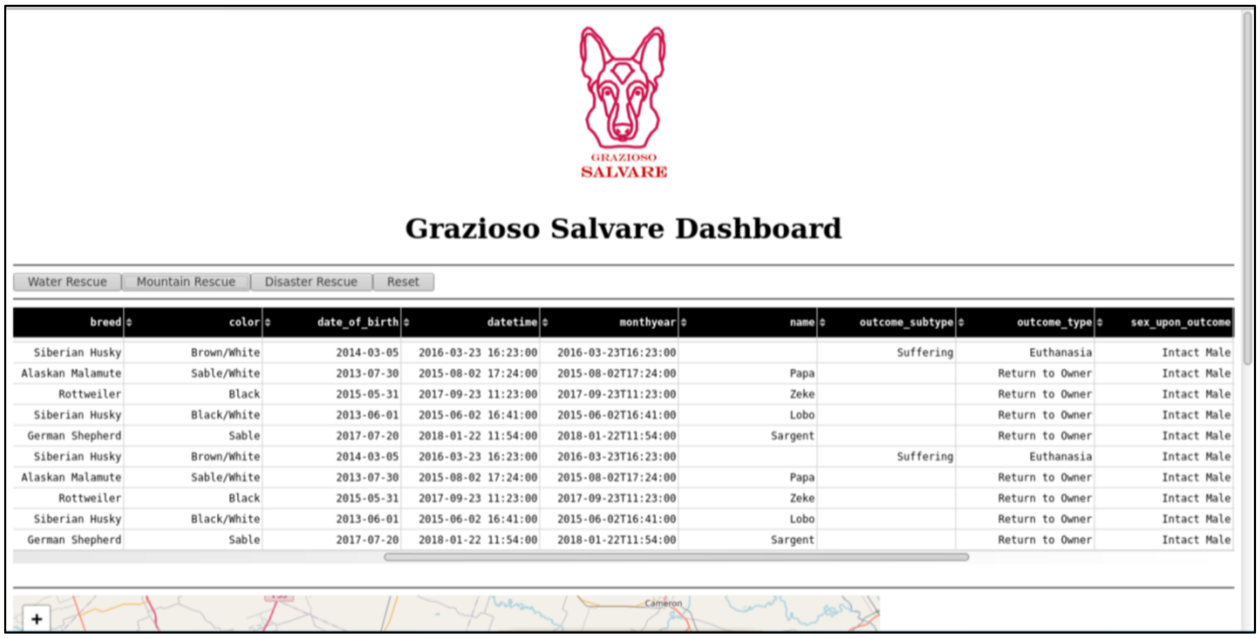
Retriever Mix	Black	2013-05-22	2015-05-22 11:45:00	2015-05-22T11:45:00	Abigail		Return to Owner	Intact Female	30.
Retriever Mix	Black/White	2015-09-01	2017-10-03 12:27:00	2017-10-03T12:27:00	19	Partner	Transfer	Intact Female	30.
Retriever Mix	Yellow	2013-12-09	2014-09-09 17:01:00	2014-09-09T17:01:00		Suffering	Euthanasia	Intact Female	30
Retriever Mix	Black	2015-09-25	2016-09-27 14:10:00	2016-09-27T14:10:00	Daisy		Return to Owner	Intact Female	30.

Developed by Quinnie Ho at Global Rain

2) Mountain Rescue: Screenshot #3 displays the filtered data table showing all of the dogs that meet the requirements for mountain rescue. Screenshot #4 displays the geolocation of the first entry of this filter; a brown/white Siberian Husky with no name reported.

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.

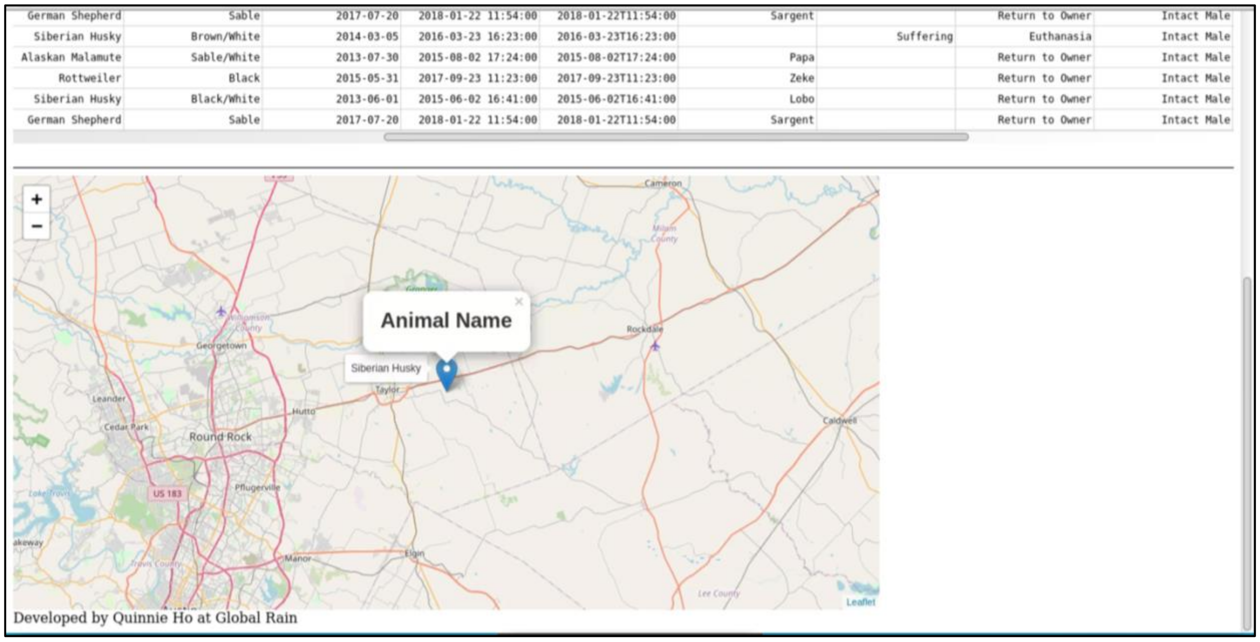
Screenshot #3: Mountain Rescue Filtered Data Table



Grazioso Salvare Dashboard

breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	outcome_type	sex_upon_outcome
Siberian Husky	Brown/White	2014-03-05	2016-03-23 16:23:00	2016-03-23T16:23:00		Suffering	Euthanasia	Intact Male
Alaskan Malamute	Sable/White	2013-07-30	2015-08-02 17:24:00	2015-08-02T17:24:00	Papa		Return to Owner	Intact Male
Rottweiler	Black	2015-05-31	2017-09-23 11:23:00	2017-09-23T11:23:00	Zeke		Return to Owner	Intact Male
Siberian Husky	Black/White	2013-06-01	2015-06-02 16:41:00	2015-06-02T16:41:00	Lobo		Return to Owner	Intact Male
German Shepherd	Sable	2017-07-20	2018-01-22 11:54:00	2018-01-22T11:54:00	Sargent		Return to Owner	Intact Male
Siberian Husky	Brown/White	2014-03-05	2016-03-23 16:23:00	2016-03-23T16:23:00		Suffering	Euthanasia	Intact Male
Alaskan Malamute	Sable/White	2013-07-30	2015-08-02 17:24:00	2015-08-02T17:24:00	Papa		Return to Owner	Intact Male
Rottweiler	Black	2015-05-31	2017-09-23 11:23:00	2017-09-23T11:23:00	Zeke		Return to Owner	Intact Male
Siberian Husky	Black/White	2013-06-01	2015-06-02 16:41:00	2015-06-02T16:41:00	Lobo		Return to Owner	Intact Male
German Shepherd	Sable	2017-07-20	2018-01-22 11:54:00	2018-01-22T11:54:00	Sargent		Return to Owner	Intact Male

Screenshot #4: Mountain Rescue Filtered Geolocation Map

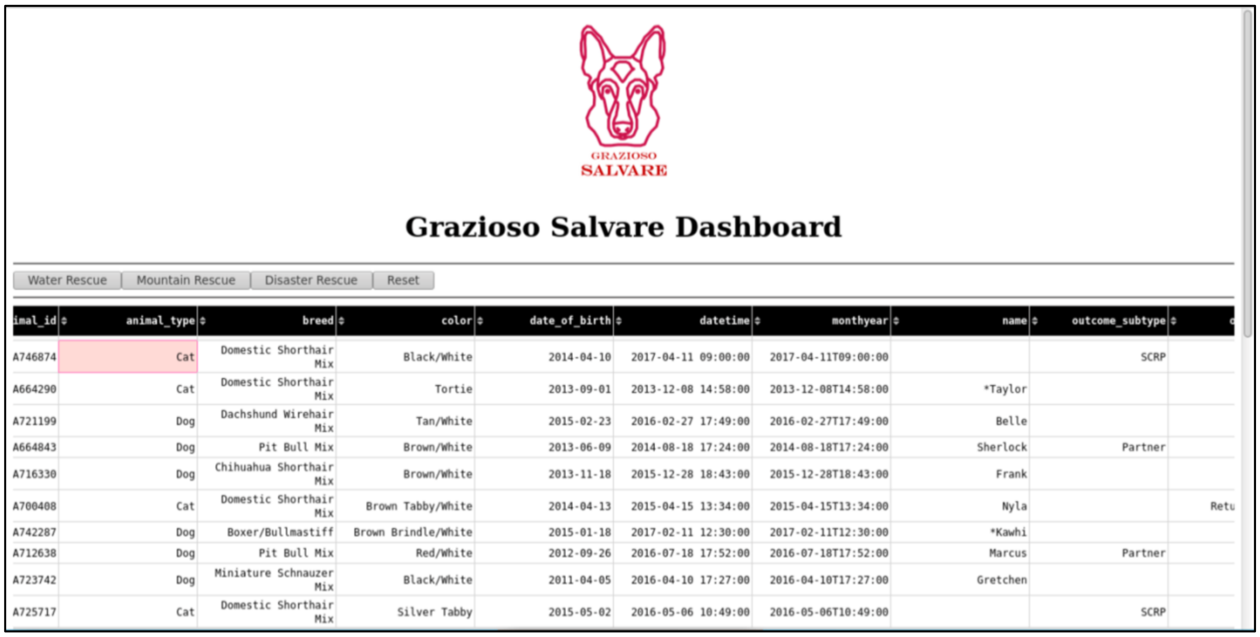


German Shepherd	Sable	2017-07-20	2018-01-22 11:54:00	2018-01-22T11:54:00	Sargent		Return to Owner	Intact Male
Siberian Husky	Brown/White	2014-03-05	2016-03-23 16:23:00	2016-03-23T16:23:00		Suffering	Euthanasia	Intact Male
Alaskan Malamute	Sable/White	2013-07-30	2015-08-02 17:24:00	2015-08-02T17:24:00	Papa		Return to Owner	Intact Male
Rottweiler	Black	2015-05-31	2017-09-23 11:23:00	2017-09-23T11:23:00	Zeke		Return to Owner	Intact Male
Siberian Husky	Black/White	2013-06-01	2015-06-02 16:41:00	2015-06-02T16:41:00	Lobo		Return to Owner	Intact Male
German Shepherd	Sable	2017-07-20	2018-01-22 11:54:00	2018-01-22T11:54:00	Sargent		Return to Owner	Intact Male

Developed by Quinnie Ho at Global Rain

3) Reset button: Screenshot #5 below displays a reset of the data table after the mountain rescue filter was applied. The reset button takes the table back to its original, unfiltered state. Screenshot #6 displays the geolocation of the first entry of this filter; a Domestic Shorthair Mix cat with no name reported.

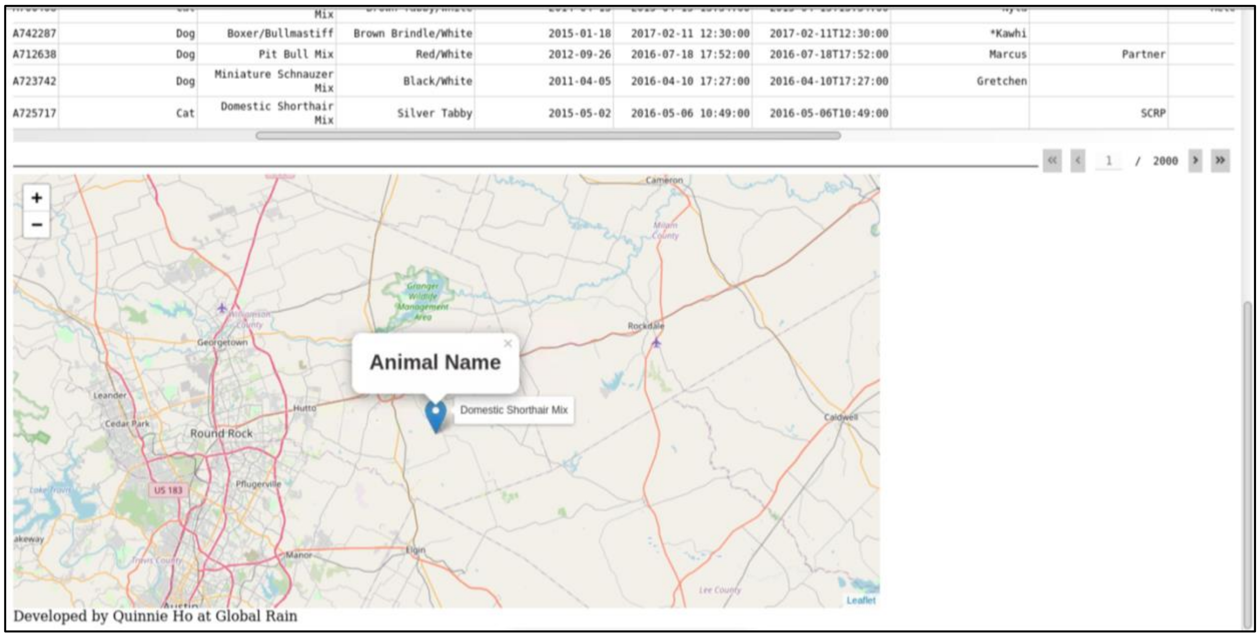
Screenshot #5: Reset Unfiltered Data Table



Grazioso Salvare Dashboard

imal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	
A746874	Cat	Domestic Shorthair Mix	Black/White	2014-04-10	2017-04-11 09:00:00	2017-04-11T09:00:00		SCRP	
A664290	Cat	Domestic Shorthair Mix	Tortie	2013-09-01	2013-12-08 14:58:00	2013-12-08T14:58:00	*Taylor		
A721199	Dog	Dachshund Wirehair Mix	Tan/White	2015-02-23	2016-02-27 17:49:00	2016-02-27T17:49:00	Belle		
A664843	Dog	Pit Bull Mix	Brown/White	2013-06-09	2014-08-18 17:24:00	2014-08-18T17:24:00	Sherlock	Partner	
A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18	2015-12-28 18:43:00	2015-12-28T18:43:00	Frank		
A700408	Cat	Domestic Shorthair Mix	Brown Tabby/White	2014-04-13	2015-04-15 13:34:00	2015-04-15T13:34:00	Nyla		Retu
A742287	Dog	Boxer/Bullmastiff	Brown Brindle/White	2015-01-18	2017-02-11 12:30:00	2017-02-11T12:30:00	*Kawhi		
A712638	Dog	Pit Bull Mix	Red/White	2012-09-26	2016-07-18 17:52:00	2016-07-18T17:52:00	Marcus	Partner	
A723742	Dog	Miniature Schnauzer Mix	Black/White	2011-04-05	2016-04-10 17:27:00	2016-04-10T17:27:00	Gretchen		
A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05-06T10:49:00		SCRP	

Screenshot #6: Reset Unfiltered Geolocation Map



A742287	Dog	Boxer/Bullmastiff	Brown Brindle/White	2015-01-18	2017-02-11 12:30:00	2017-02-11T12:30:00	*Kawhi		
A712638	Dog	Pit Bull Mix	Red/White	2012-09-26	2016-07-18 17:52:00	2016-07-18T17:52:00	Marcus	Partner	
A723742	Dog	Miniature Schnauzer Mix	Black/White	2011-04-05	2016-04-10 17:27:00	2016-04-10T17:27:00	Gretchen		
A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05-06T10:49:00		SCRP	

Developed by Quinnie Ho at Global Rain

4) Disaster Rescue: Unfortunately, the screenshots for the disaster rescue filter can't be captured at this time due to the current connectivity issue I'm having with Apporto, the environment I've been using to run MongoDB and develop this project.

Installation

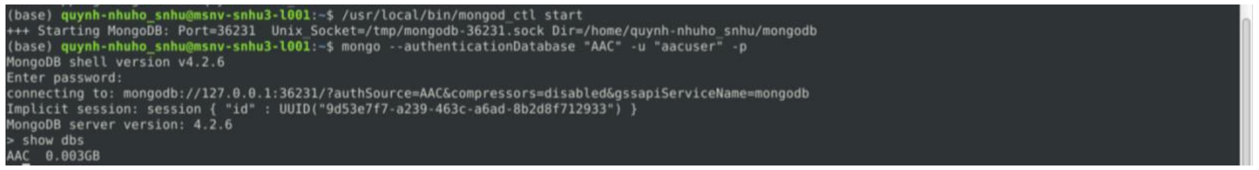
The tools required to complete this project are a Terminal application, MongoDB version 4.2, PyMongo, Jupyter Notebook, and Dash framework. Here is why each tool is required and directions on how to install them:

1. The Terminal application is used to host MongoDB. The Linux terminal was used for this project.
2. MongoDB is used as our NoSQL database tool. More specifically, we are using version 4.2 of the application. This version can be downloaded by going to THIS link and selecting the download option that matches your operating platform. One advantage of using it as a database tool is that it is open sourced, simple, and straightforward. Its simplicity is the reason that we chose it for this project, since our project only requires developing a simple dashboard.
3. PyMongo is the official MongoDB driver for Python. We used this tool to create the CRUD Python Module. More information on it and how to install it can be found HERE.
4. Jupyter Notebook is the Python IDE used in this project to create the Python file that contains our development code and Python Notebook file that contains the test code. This is the environments that hosts our development and testing for both the CRUD Python Module and interactive dashboard. This IDE can be downloaded HERE.
5. The Dash framework provides a simplified way to construct the view and controller structures for the web application. One way that it achieves this is through a Pandas data frame. Pandas makes it easy to translate data between MongoDB and other libraries, therefore, importing the AAC data from MongoDB into Pandas is the first step in our dashboard development.

Getting Started
Follow these example steps to get a local copy up and running:

Part I: Set Up

1. Ensure that all tools mentioned in the "Installation" section above are installed and set up correctly.
2. Start the Terminal application.
3. Log into the user account created during the CRUD Python Module milestone. We used our aacuser user account for this project. Once you have successfully logged into the account, enter Enter "use AAC" in the Mongo shell – where "AAC" should be replaced with the name of your database.



```
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ /usr/local/bin/mongod_ctl start
+++ Starting MongoDB: Port=36231  Unix_Socket=/tmp/mongodb-36231.sock Dir=/home/quynh-nhuho_snhu/mongodb
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ mongo --authenticationDatabase "AAC" -u "aacuser" -p
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:36231/?authSource=AAC&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9d53e7f7-a239-463c-a6ad-8b2d8f712933") }
MongoDB server version: 4.2.6
> show dbs
AAC   0.003GB
```

4. Now, open Jupyter Notebook to upload the attached CRUD Python Module as a ".py" file and the "Grazioso Salvare Dashboard" as a Notebook IPYNB file. Make sure that they are uploaded into the same folder/directory.
5. Open the imported "Grazioso Salvare Dashboard" file. This file hosts the dashboard developmental code and test code.
6. The top of the file includes all of the libraries required to connect with other tools, such as "from pymongo import MongoClient" to access the aacuser and AAC database in MongoDB. We need to add two libraries in this step:
    1. Add "import base64" to link the base64 library to handle the encoded logo.
    2. Link the CRUD Python Module by entering "from CRUD_Python_Module import AnimalShelter". This pulls in the "AnimalShelter" class created in the "CRUD_Python_Module" file.

Part II: Data Manipulation / Model

7. We add in our username and password for the “aacuser” user account in MongoDB that we are using to access the AAC database. This section uses the inputted account credentials to authenticate access to the AAC database in Mongo and read in the data using the Pandas framework.

Part III: Dashboard Layout / View

8. This section is where we design the layout of our dashboard. The first to-do item is to add in the requested logo for the client. We do this by ensuring that the image of the logo is saved in the same directory as the Notebook file we are using, and then code in the name of the image file.
9. Next, we add in an HTML image tag and URL anchor tag per our client’s request.
10. Our next development is the buttons for each of the rescue type of the interactive filtering feature. These are displayed in a row, one next to the other. There are a total of four buttons, therefore, we need to declare a variable ID for each button, assign a variable that reports back when the button has been clicked, and the display name of the button. We assign ‘n_clicks’ to each button to hold the amount of clicks for each button. To keep it simple, each variable ID is ‘button_one’, ‘button_two’, etc., and the display names are the name of the rescue types and the reset button (i.e., “Water Rescue”, “Mountain Rescue”, “Disaster Rescue”, and “Reset”).
11. The next specification is for the layout of the data table that hosts the AAC data from MongoDB. We add in user-friendly features to make the database organized and intuitively easy for our client to use with little to no training. Some of these features are to ensure that the table is not editable, displaying only 10 entries per page, allowing users to sort by columns, etc.
12. The next set of code is to add a placeholder in the dashboard to include the pie chart and geolocation map that is coded in the callback section later in the file.
13. Our last layout code is adding in the client-requested unique identifier to the bottom left corner of the dashboard.

Part IV: Interaction Between Components / Controller / Callbacks

14. This last section is the controller of the Dash framework and deals with callbacks that allow us to activate features that we designed in the layout section, such as sortable data table, interactive buttons that create custom filters when clicked, etc. We start with developing the button callbacks to filter the data table accordingly when each button is clicked.
    1. The first step is to include all of the input IDs and values (‘n_clicks’).
    2. Then we create the “update_dashboard” function by passing in four variables, one for each button.
    3. The function set ups 4 conditional statements, one for each button.
        i. If ‘button1’ (“Water Rescue”) is pressed, then the data table will filter by any of the following breeds: Labrador Retriever Mix, or Chesapeake Bay Retriever, Newfoundland. Within these preferred breeds, the data will also filter down by animals that are labeled as “Intact Female” and are within 26 and 156 weeks of age.
        ii. The second button is reserved for dogs best fit for mountain rescues. When pressed, ‘button2’ will filter the data table to show only the following breeds: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler. Moreover, only “Intact Male” that are 26 weeks to 156 weeks of any of these breeds are displayed.

iii. The third or “Disaster Rescue” button displays the following breed of “Intact Male” that are 20 to 300 weeks old: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler.

4. The last button is the restart button. Pressing this button will return any sorted/filtered data back to its original unfiltered state.

15. We also added in code to display a pie chart that breakdowns the percentage of each breed available when the rescue-type button is pressed. This is the callback function for the pie chart placeholder that is in the layout section.
16. The last callback function is for the geolocation map. This is developed to show the geolocation of the animal displayed on the first row of the data table along with its breed and name (if displayed in the data table). This map works alongside the data table, so it is updated when the table is filtered.
17. Finally, we click “Run” in the Notebook file’s toolbar to activate the dashboard.

Challenges

One big problem I encountered is the inability to populate a graph such as a bar graph or pie chart. I have looked through many resources and blogs, and tried different methods to render different types of graphs without avail. The main problem I run into is that my problem seems to be running in the background, however, the dashboard is completely white and nothing is displayed. I’m unsure if the root cause is my code or the unreliable Apporto connection; the virtual environment that I’ve been using to access all of the tools required to build this project.

The unreliable and extremely slow connection in Apporto caused issues with freezes, getting kicked out of the system, and log-in, authentication, and idle connections in MongoDB. Naturally, these issues resulted in a snowball effect that caused major delays and/or inability to run the dashboard. I filed a ticket for these issues, however, there was not more that could be done besides logging out of Apporto and/or giving it sometime before restarting and logging back in.

Contact

Quinnie Ho