Mongodb Library Utilizing CRUD in Python Coding.

Explanation: This is a demonstration on how to create a web client interface that allows users to interact with a dashboard, pie graph and geological graph from a data that is collected in a MongoDB Database which is called ‘animal’ which is located in the AAC database.

## Motivation

The main purpose of the project is to make it possible for a web application that connects a client-side user interface to a database using python programing language as a way to enable CRUD from a Mongodb for these connections and therefor have an illustrated dashboard, geological graph and pie graph that will interact with the data located in the MongoDB database.

## Getting Started

The steps to set your project locally you will need to download mongodb in order for you to be able to set up your new database which will contain your data, in this case will be the database ‘AAC’ and within the ‘AAC’ we will ensure that it has the ‘animal’ collection data in order for us to store our information. Then we install python and use the Jupiter notebook IDE so we can take advantage of the CRUD functionality with the help of Monogodb library in order to implement a storage application. Later we use Dash to make an interface between the data in a dashboard, pie graph and geological graph.

## Installation

Mongodb: this will be your database that will ensure to query and index your data and we can install this in: https://www.mongodb.com/docs/manual/installation/

Python programming: This will be the programming language in order to utilize the Mongodb library and other libraries such as Dash and pandas and this programming language can be installed in: https://www.python.org/

Dash Framework: constructs an interface that allows for users to interact with the data and it utilizes a dashboard with many filters that can organized the data efficiently as recommended by the user preference as well as build graphs that can demonstrate the data purpose in many ways which in this project will be the geological graph and the pie graph, installation can be done through pip installation which in this case will be “pip install Dash”

Jupyter notebook: This IDE will allow you to create python coding and in order to install it we will need either launch anaconda navigator which will allow for Jupyter notebook to be easily install or pip installing Jupiter notebook in a terminal in Linux or cmd in windows

## Usage

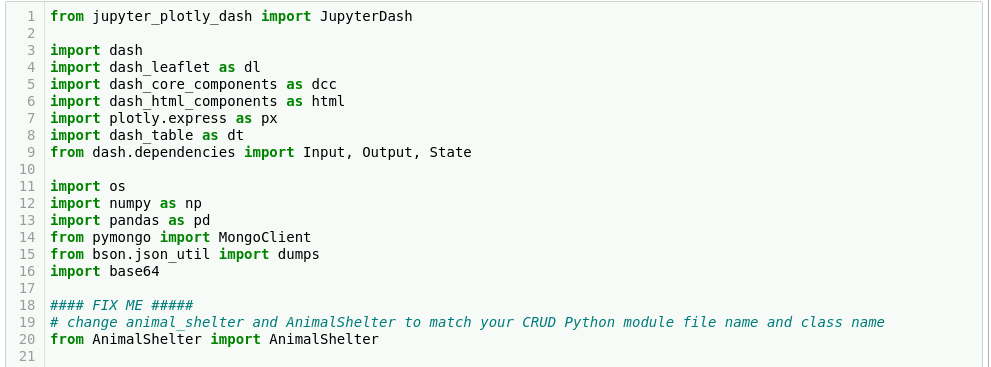
**STEPs:**

1. **Imports:**

We first imported the necessary libraries such as MongoDB client, Dash multiple functions, pandas data frame and the CRUD functionalities that were built in the AnimalShelter Module so with the help of python inside jupyter notebook we insert all of these libraries together:

Example code:

Import dash ….

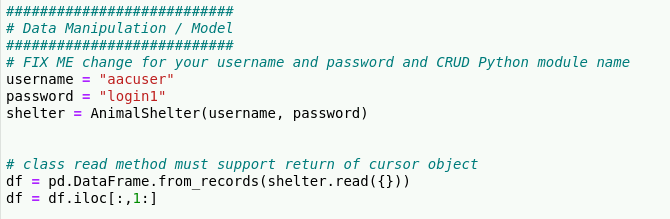


**2. Data Import:**

We import our data and our credentials to the pandas dataframe by using the AnimalShelter CRUD functionality of MongoDB which holds the database name ‘AAC’ and its collection called animals.

Example code:

df.pdDataFrame.from\_records(AnimalShelter(username,password).read({}))



**1.Import logo and layout**:

We then import the image logo which is “Gravioso salvare logo.png” and create the layout for the dash as “project 2”.

Example code :

#Dashboard layout

app = JupyterDash(“….”)



**3. Build the layout**:

We insert the ids, build upon the layout with the help of html syntax and set up the filter options for Water rescue, mountain rescue, disaster rescue and reset as well as the dashboard features filters that allow the user to interact with the data.

Example code:

Html layout:

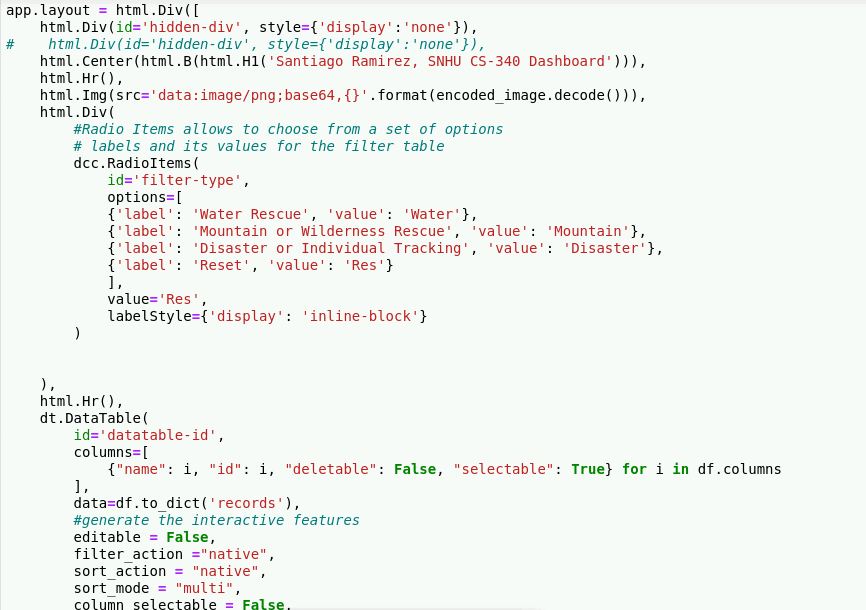
app.layout = html.DIV({

html.Div(Id = “….”, style ={display = “”},

html.center(….),

Html.Hr(),

)}



**4. Build Dashboard Filter Type:**

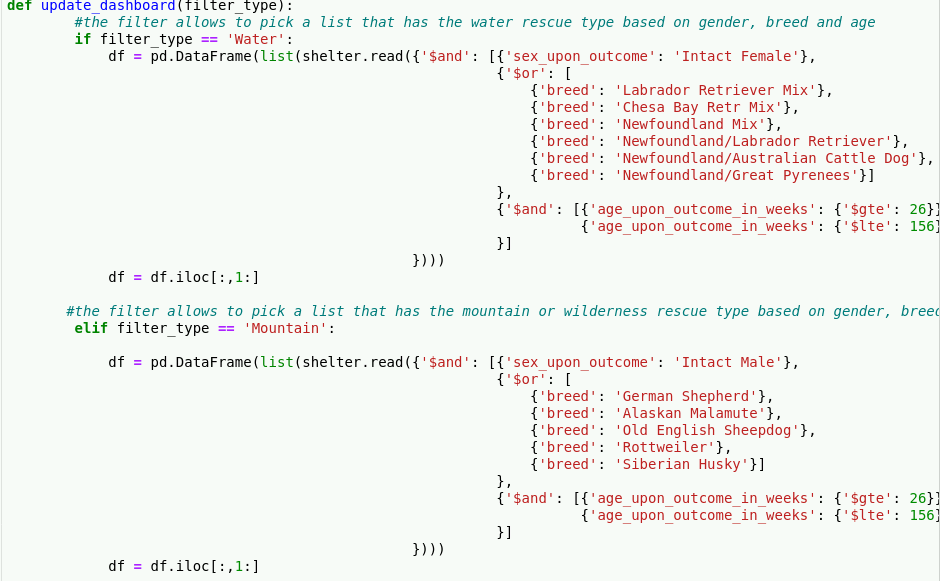
We build the dashboard filter option which will update the information of the dashboard whenever it changes from each rescue type such as water rescue, mountain rescues… etc:

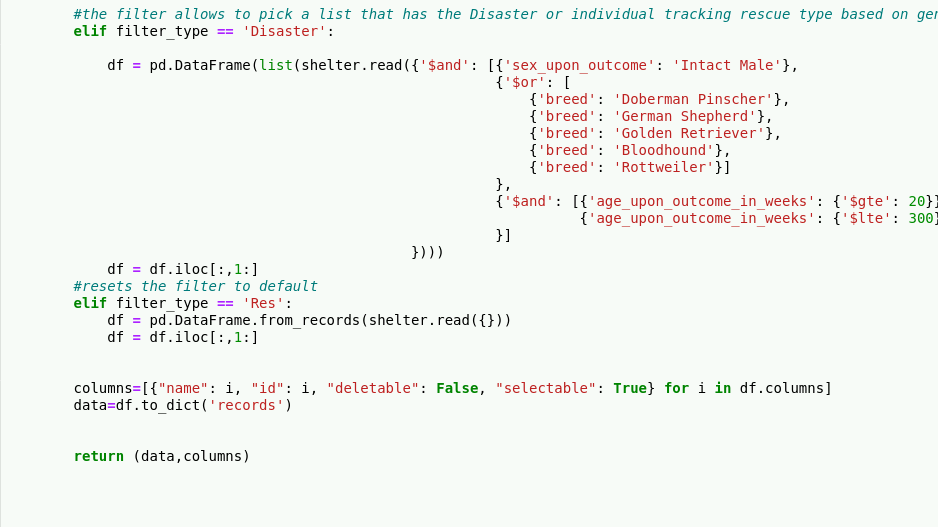
Example code:

def update\_dashboard(filter\_type):

if filter\_type == “water”:

df.pdDataframe(list(data.read({……})





**5. Pie graph**:

This section we build the pie graph which main purpose is to show the breed of the dog and the percentage of the breed that appears in the data frame which in this case gets filter by the rescue types (water, disaster, mountain, reset).

Example Code:

def update\_graphs(viewData):

dff.pd.DataFrame.from\_dict(viewData)

names = breed\_type

values = amount

return [ dcc.Graph(

figure=……..,

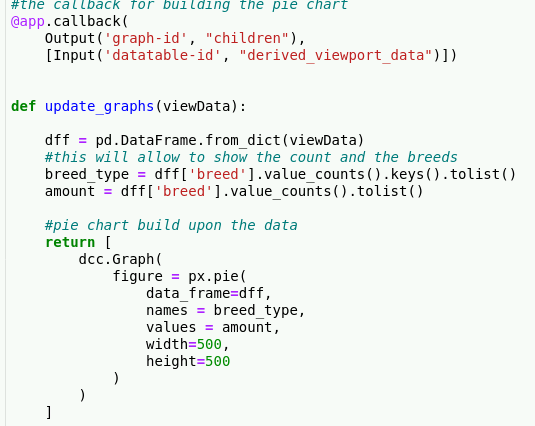
data\_frame = dff,

names = breed\_type,

values = amount,

width =…

height=….)]



**6. Geological graph:**

This final section builds upon the geological graph for the data that will be store in the dashboard as well as the pie graph, but it will show the first instance of the animal that is in the dashboard list as well as update to the rescue types(water, disaster, mountain, reset).

Example code:

def update\_map(viewData)

dff.pd.DataFrame.from\_dict(viewData)

return [

dl.map(style = ….

dl.TileLayer( id= …)

dl.market(position = [….]

dl.popup([

html.H1(“…..”),

html.p(dff.iloc[…,…])

]) ]) ]) ]



**8. Results:**

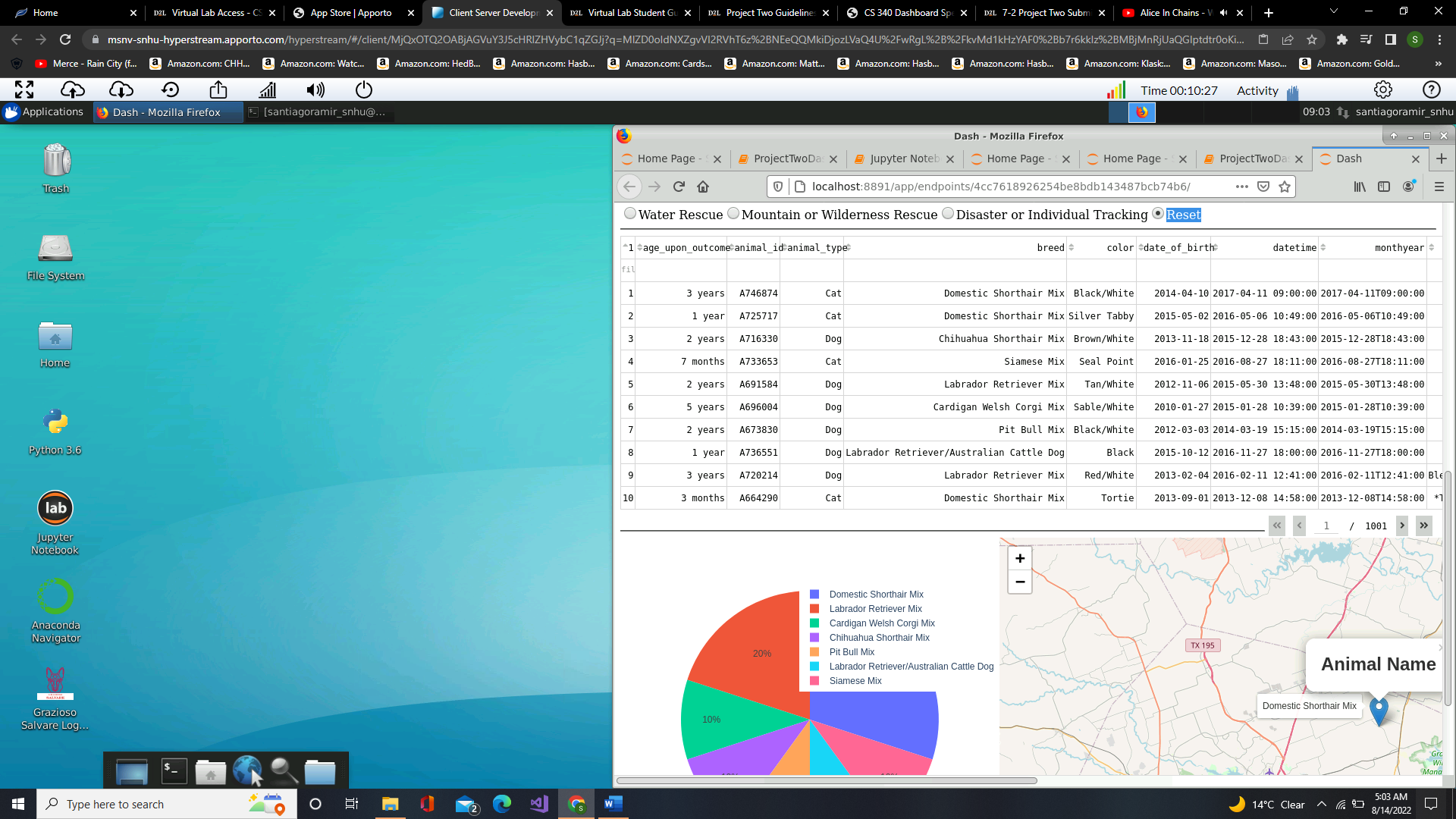
**Logo:**

Demonstrates the unique name of the dashboard which is my name as well as the logo that is display with it

****

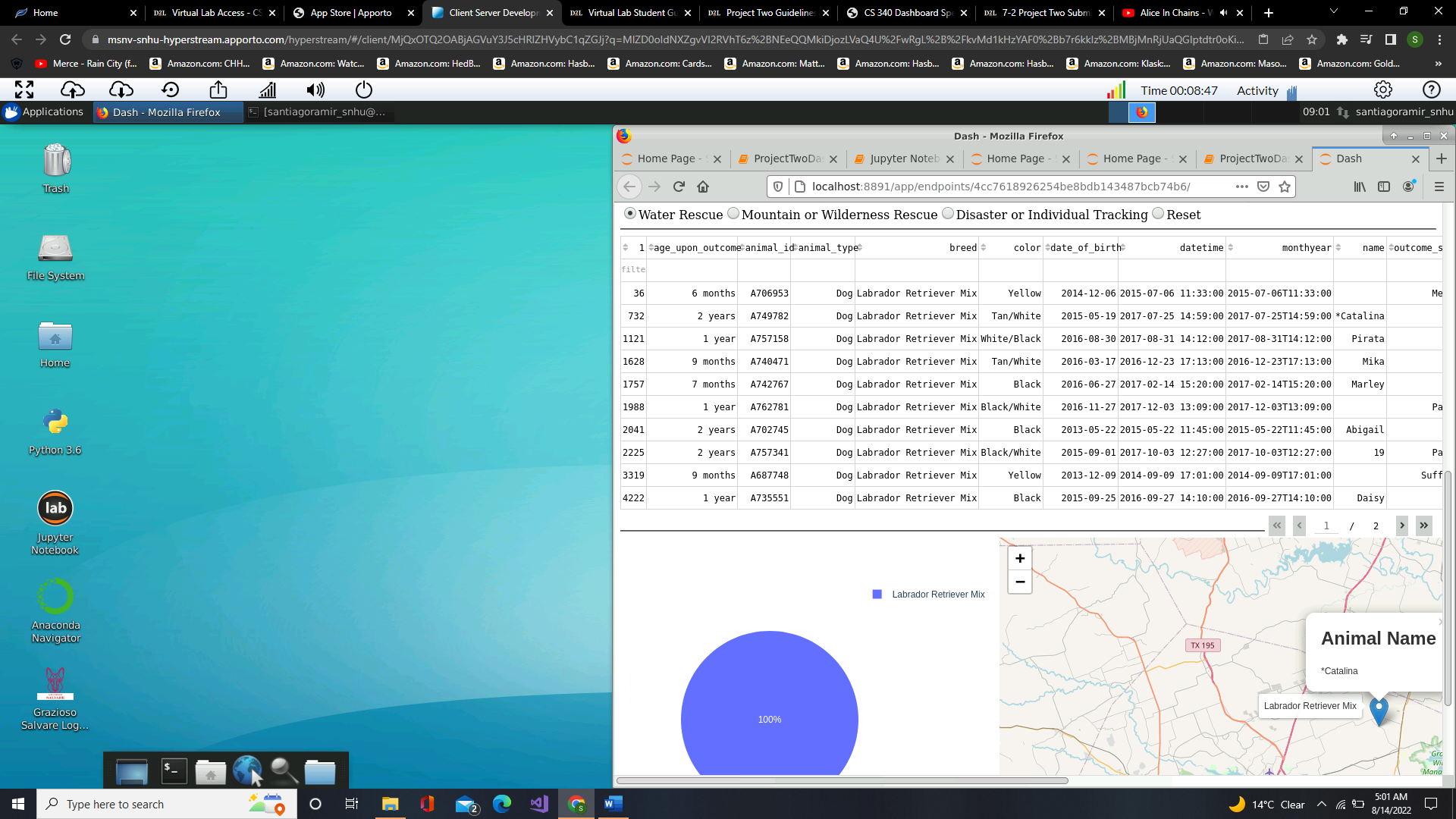
**Dashboard and Reset type:**

Has multiple filters such as organizing data from Ascending and Descending order from any column of choice, it has minimum 10 records at most for the dashboard display it can scroll to the next line of data which is the one at the bottom right of the dashboard and it has radio items which are the water rescue, mountain.wilderness rescue, disaster or individual tracking and reset which in this case is at the reset button.

****

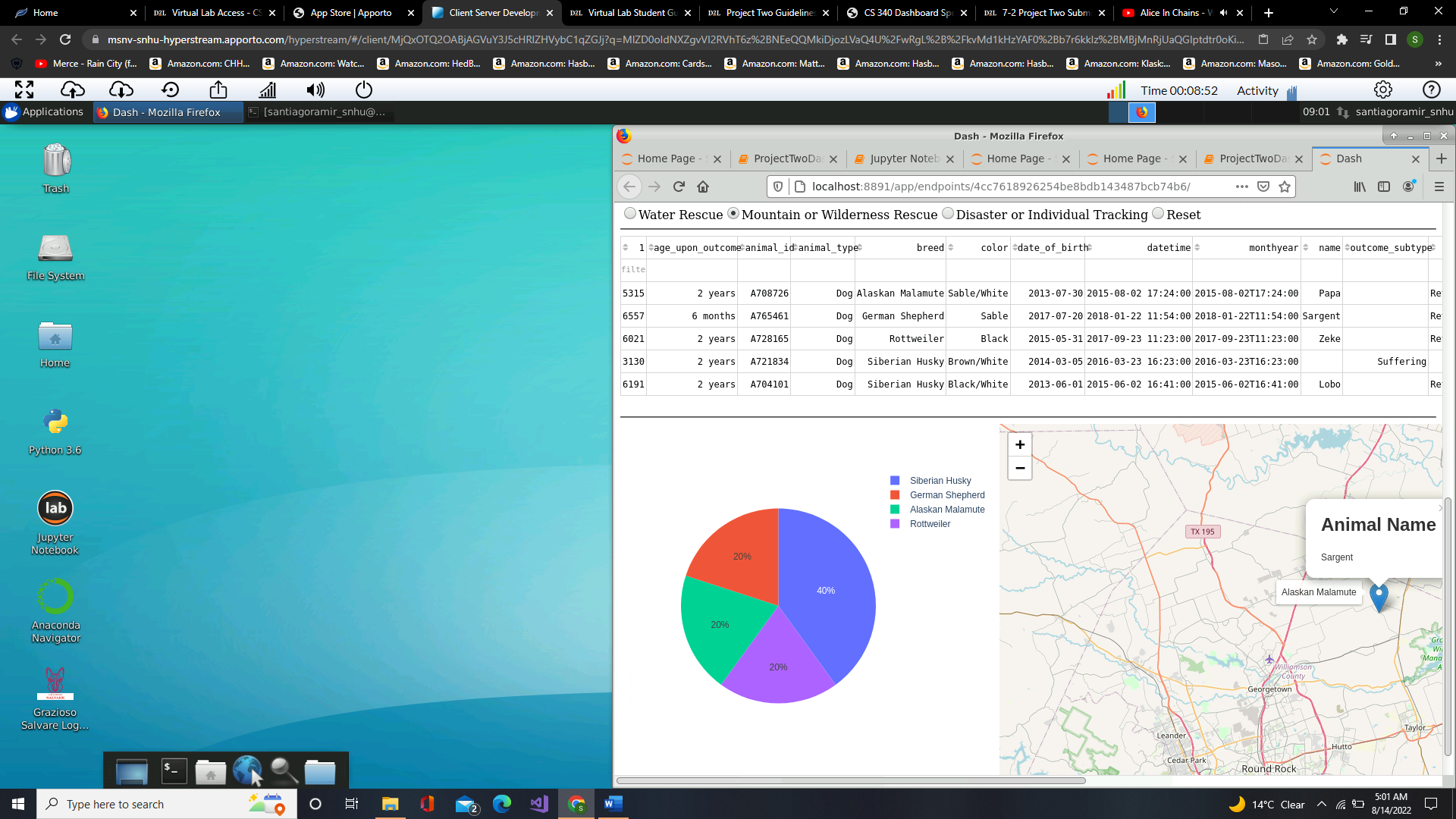
**Water Rescue type:**

This is how the dashboard, pie graph and geological graph look after using the water rescue type in the radio item options.

****

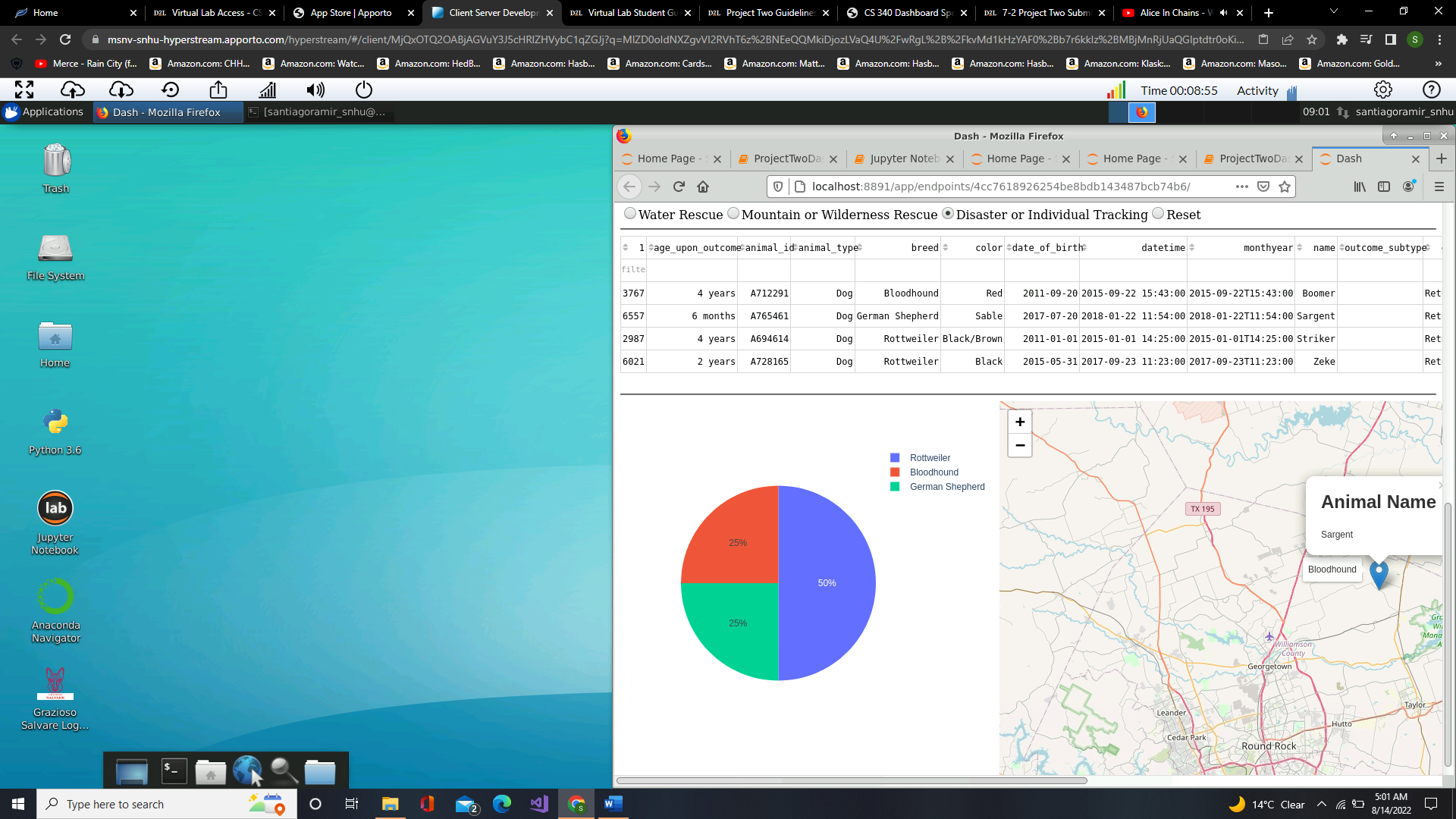
**Mountain or Wilderness Rescue:**

This is how the dashboard, pie graph and geological graph look after using the mountain or wilderness Rescue rescue type in the radio item options.



**Disaster or Individual Tracking:**

This is how the dashboard, pie graph and geological graph look after using the Disaster or Individual Tracking rescue type in the radio item options.

****

**Challenges:**

The challenges that I had encounter creating the interface is trying to use the callbacks correctly since I didn’t understand at first how the callbacks worked, and I quickly overcame this challenges by reading and trying to figured that the first parameter in the callback refers to the id of course of the graphs that needed modification the output of the callback is what the interface sees and the input is what is being applied like updates and therefor the second parameters refers to the updates and changes that were needed for the graphs to change and that is what I was mostly stuck but I somewhat understood what its purpose does. Another challenge I had encountered is that the data was taking very long to load and to overcome this issues I made sure to first keep the data as original to MongoDb cursor and also remove the id column from the collection animals since I noticed it takes more time to process these ids than I had imagined so removing this column was essential to make the dash load faster.

## Contact

Your name: Santiago Ramirez Jimenez