# CS 340 README

## About the Project/Project Title

*Grazioso Salvare rescue. Is working with five shelters to identify dogs that are good candidates for search and rescue or other life-threatening conditions.*

## Motivation



## Getting Started

* *To get started you must install MongoDB on your machine.*
* *Must have python 3 installed.*
* *You will also need the Pymongo library.*
* *download the aac\_shelter\_outcomes.csv*
* *install the Dash library*
* *install Jupyter notebook*

## Installation

* *Use the official MongoDB website and follow the download and installation instructions.* [*Install MongoDB - MongoDB Manual*](https://www.mongodb.com/docs/manual/installation/)*.*
* *Install the libraries for your machine using* [*Download and Install - PyMongo v4.10 (mongodb.com)*](https://www.mongodb.com/docs/languages/python/pymongo-driver/current/get-started/download-and-install/?msockid=160c0f3bb1a963ad390d1d47b054623e)*.*
* *Use this command to import the aac\_shelter\_outcomes.csv*
  + *mongoimport --username="${MONGO\_USER}" --password="${MONGO\_PASS}" --port=${MONGO\_PORT} --host=${MONGO\_HOST} --db AAC --collection animals --authenticationDatabase admin --type csv --file aac\_shelter\_outcomes.csv –headerline*
* *In the terminal use this command “pip install dash”*
* *Download jupyter from their official website* [*Project Jupyter | Home*](https://jupyter.org/)

## Usage & tests

*Uses the R of the CRUD operations to read the data from the csv file.*

*The dashboard will format to make it simple to view and sort the data.*

*The data can be sorted by various rescue types from the dynamic buttons.*

*The dashboard loads 10 animals per page to make it easier to load and view.*

### *Code Example and tests*

from jupyter\_plotly\_dash import JupyterDash

import import\_ipynb

import dash

import dash\_leaflet as dl

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.express as px

import dash\_table as dt

from dash.dependencies import Input, Output, State

import base64

import os

import numpy as np

import pandas as pd

from pymongo import MongoClient

from bson.json\_util import dumps

#### import the crud script #####

import module\_script

from module\_script import AnimalShelter

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

# Data Manipulation / Model

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

username = "accuser"

password = "test123"

shelter = AnimalShelter(username, password)

# was getting a typeError, to fix it \_id field from ObjectId needs to be converted to a string.

# this way it can be serialized for JSON.

df = pd.DataFrame.from\_records(

        [{\*\*record, '\_id': str(record['\_id'])} for record in shelter.read({})])

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

# Dashboard Layout / View

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

app = JupyterDash('SimpleExample')

#Adds in Grazioso Salvare’s logo

image\_filename = 'GSLogo.png'  # replace with your own image

encoded\_image = base64.b64encode(open(image\_filename, 'rb').read()).decode()

# Adds image and decodes the encoded image.

#html.Img(src='data:image/png;base64,{}'.format(encoded\_image.decode()))

app.layout = html.Div([

    html.Center(html.Img(src = f'data:image/png;base64,{encoded\_image}')),

    html.Center(html.B(html.H1('Aneudi M. SNHU CS-340 Dashboard'))),

    html.Hr(),

    html.Div(

        # Filter by all animals or animal type.

        dcc.RadioItems(

            id = 'filter-type',

            options = [

                {'label': 'All Rescues', 'value': 'ALL'},

                {'label': 'Water Rescue', 'value': 'WATER'},

                {'label': 'Mountain or Wilderness Rescue', 'value': 'MOUNTAIN'},

                {'label': 'Disaster or Individual Tracking', 'value': 'DISASTER'}

            ],

            value = 'ALL',

            labelStyle = {'display': 'inline-block'}

        )

    ),

    html.Hr(), #Dashboard features.

    dt.DataTable(

        id = 'datatable-id',

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

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

        page\_size = 10,

        selected\_rows = [0],

        editable = False,

        filter\_action = "native",

        sort\_action = "native",

        sort\_mode = "multi",

        column\_selectable = False,

        row\_selectable = "single",

        row\_deletable = False,

        selected\_columns = [],

        page\_action = "native",

        page\_current = 0

    ),

    html.Br(),

    html.Hr(),

#This sets up the dashboard so that your chart and your geolocation chart are side-by-side

    html.Div(className = 'row',

         style = {'display' : 'flex'},

             children = [

        html.Div(

            id = 'graph-id',

            className = 'col s12 m6',

            ),

        html.Div(

            id = 'map-id',

            className = 'col s12 m6',

            )

        ])

])

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

# Interaction Between Components / Controller

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

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

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

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

def update\_dashboard(filter\_type):

### FIX ME Add code to filter interactive data table with MongoDB queries

    if filter\_type == 'ALL':

        query = {}

    elif filter\_type == 'WATER':

        query = {'rescue\_type': 'Water Rescue'}

    elif filter\_type == 'MOUNTAIN':

        query = {'rescue\_type': 'Mountain or Wilderness Rescue'}

    elif filter\_type == 'DISASTER':

        query = {'rescue\_type': 'Disaster or Individual Tracking'}

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

    data = df.to\_dict('records')

    return (data,columns)

@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]

#Piechart to show the percentage of animal type in the data.

@app.callback(

    Output('graph-id', "children"),

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

def update\_graphs(viewData):

    df = pd.DataFrame(viewData)

    if not df.empty:

        pie\_chart = px.pie(df, names = 'animal\_type', title = 'Animal Type Distribution')

        return [dcc.Graph(figure = pie\_chart)]

    return []

@app.callback(

    Output('map-id', "children"),

    [Input('datatable-id', "derived\_viewport\_data"),

     Input('datatable-id', 'selected\_rows')])

def update\_map(viewData, selected\_rows):

#FIXME Add in the code for your geolocation chart

    if viewData is None or len(viewData) == 0:

        return []

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

    if selected\_rows is None or len(viewData) == 0:

        row = 0

    else:

        row = selected\_rows[0]

    lat, lon = 30.75, -97.486

    if not pd.isna(dff.iloc[row, 13]) and not pd.isna(dff.iloc[row, 14]):

        lat = dff.iloc[row, 13]

        lon = dff.iloc[row, 14]

    # Austin TX is at [30.75, -97.48]

    return[

        dl.Map(style={'width': '1000px', 'height': '500px'}, center = [lat, lon], zoom = 10, children = [

            dl.TileLayer(id = "base-layer-id"),

            # Marker with tool tip and popup

            dl.Marker(position = [lat, lon], children = [

                dl.Tooltip(dff.iloc[row,4]),

                dl.Popup([

                    html.H1("Animal Name"),

                    html.P(dff.iloc[row,9])

                ])

            ])

        ])

    ]

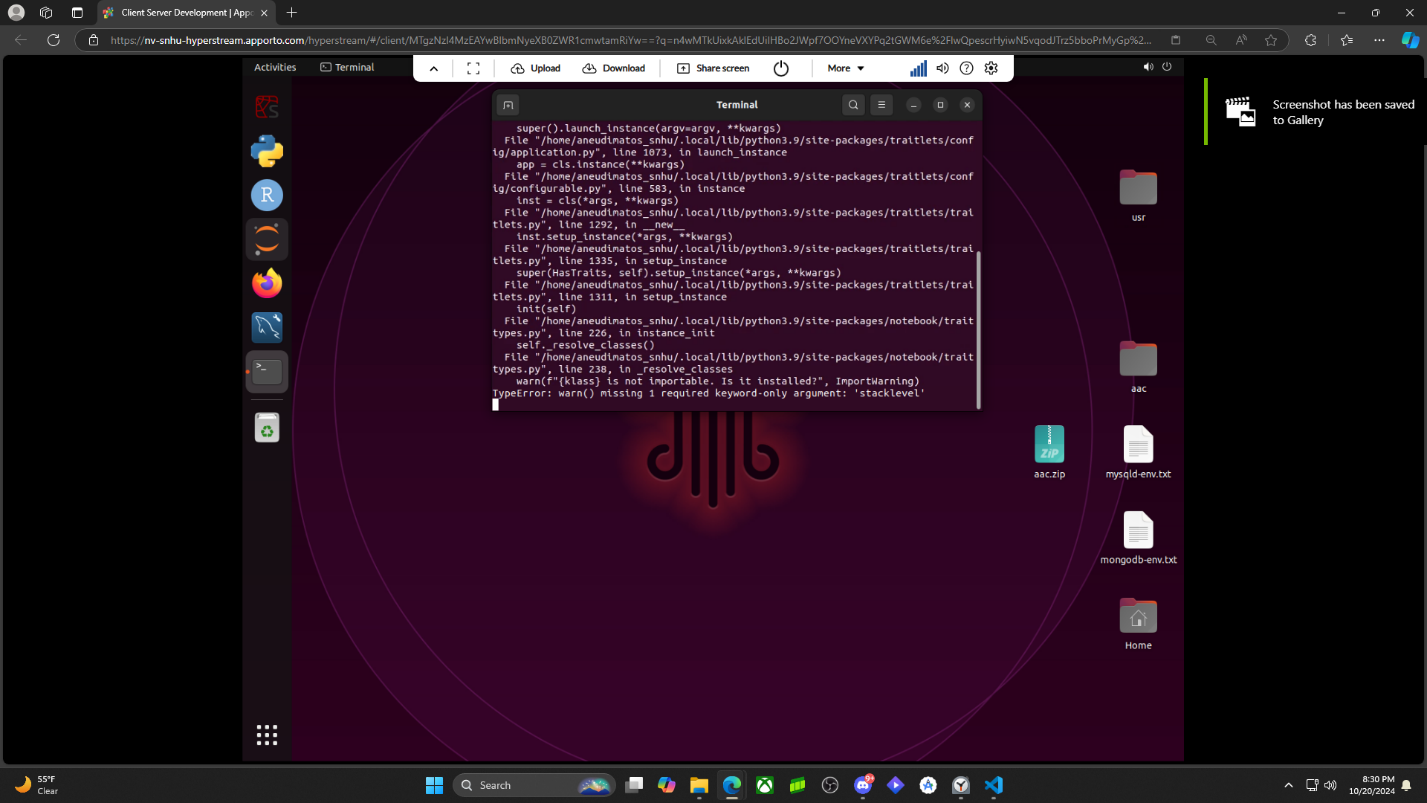
app

### Challenges and improvements

In this project I was not able to get the map to render, I am still not sure if it was a virtual environment or if it was an issue with the code. I attempted to fix all the errors of the map that would appear in the shell, but even after those errors were fixed the map would not render. I attempted to hard code the Coordinates given for Austin Texas and the map would still not render. Another challenge I faced was Dynamic filter buttons. I was not able to find what they needed to filter from the dataset as there was no columns that held the data was to be filtered like WATER, or MOUNTAIN.

# Screenshots

I was not able to get screenshots of the dashboard running because after I had spent enough time trying to render the map my virtual environment stopped working and I couldn’t access jupyter without it crashing. I have a screenshot of it crashing instead.



Contact

Your name: Aneudi Matos