# CS 340 Project Two README

## About the Project/Project Title

*The purpose of this assignment was to create a fully functional MongoDB dashboard that successfully displays/filters data, and displays a geo-location map and a chart of choice, which in this case is a pie chart. This project must use the CRUD functions developed in project one to interact with the database.*

## Motivation

*The motivation behind this project is to complete the “client-side” of our project. We have already developed the backend side of the app in our project where all the data is stored and handled, now in this project we are developing where the user interacts with the data stored in our database.*

## Getting Started

*First, we need to start our MongoDB database in the shell commands.*

*-Open up the shell terminal and start the program with authentication.*

*Text

Description automatically generated*

*-Open Jupyter Notebooks, and create a new Notebook script. Import the python script from project One so we can use the CRUD functions.*

## Installation

*MongoDB, the shell command line, and jupyter notebooks are required for this project. Installation is already provided by the apporto Linux virtual machine. We will also need to import: mongoclient to connect our MongoDB base, dash/dash\_leaflet/dash\_core/dash\_html\_components to create our dashboard, base64 for our encoded image, and plotly.express to create our pie chart.*

## Usage

*We developed a jupyter notebooks script to create our dashboard and access the data from our python script. See the code below for reference:*

### Code:

from jupyter\_plotly\_dash import JupyterDash  
  
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 CRUD Python module file name and class name  
from project import AnimalShelter  
  
###########################  
# Data Manipulation / Model  
###########################  
  
# Instantiate AAC Database with our username and password  
username = "aacuser"  
password = "joan1124"  
shelter = AnimalShelter(username, password)  
  
# class read method must support the return of cursor object  
df = pd.DataFrame.from\_records(shelter.read({}))  
  
#########################  
# Dashboard Layout / View  
#########################  
app = JupyterDash('SimpleExample')  
  
# Adds in Grazioso Salvare’s logo  
# DONE  
image\_filename = 'Grazioso.png' # replace with your own image  
encoded\_image = base64.b64encode(open(image\_filename, 'rb').read())  
  
# FIX ME Place the HTML image tag in the line below into the app.layout code according to your design  
# FIX ME Also remember to include a unique identifier such as your name or date  
# html.Img(src='data:image/png;base64,{}'.format(encoded\_image.decode()))  
  
# Place GRAZIOSO salvatore logo in the middle of the screen  
app.layout = html.Div([  
 html.Div(id='hidden-div', style={'display': 'none'}),  
 html.A([  
 html.Img(  
 src='data:image/png;base64,{}'.format(encoded\_image.decode()),  
 style={  
 'height': '25%',  
 'width': '25%',  
 'float': 'center',  
 'position': 'relative',  
 'padding-top': 0,  
 'padding-right': 0,  
 'padding-left': '35%'  
  
 })  
 ]),  
 html.Br(),  
 html.Center(html.B(html.H1('Rescue Animal Locator'))), # set title  
 html.Hr(),  
 # Create dropdown menu for filter type options  
 dcc.Dropdown(  
 id='demo-dropdown',  
 options=[  
  
 {'label': 'Water Rescue', 'value': 'W'},  
 {'label': 'Mountain or Wilderness Rescue', 'value': 'M'},  
 {'label': 'Tracking Rescue', 'value': 'TR'},  
 {'label': 'Reset', 'value': 'R'}  
 ],  
 value='NYC'  
 ),  
 # *FIXME Add in code for the interactive filtering options. For example, Radio buttons, drop down, checkboxes, etc.* dt.DataTable(  
 id='datatable-id',  
 columns=[  
 {"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns  
 ],  
 data=df.to\_dict('records'),  
 # Set up the features for your interactive data table to make it user-friendly for your client  
 editable=False,  
 filter\_action="native",  
 sort\_action="native",  
 sort\_mode="multi",  
 column\_selectable=False,  
 row\_selectable=True,  
 row\_deletable=False,  
 selected\_columns=[],  
 selected\_rows=[],  
 page\_action="native",  
 page\_current=0,  
 page\_size=10,  
 ),  
 html.Br(),  
 # Set up the pie graph and geo map layout",  
 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',  
 ),  
  
 ]),  
 html.Br(),  
 html.Hr(),  
])  
  
  
#############################################  
# Interaction Between Components / Controller  
#############################################  
  
  
@app.callback([Output('datatable-id', 'data'),  
 Output('datatable-id', 'columns')],  
 [Input('demo-dropdown', 'value')])  
def update\_dashboard(value):  
 ###code to filter interactive data table with MongoDB queries  
 # Select the right query for the filter options in the dropdown menu  
 if (value == 'R'):  
 df = pd.DataFrame.from\_records(shelter.readAll({}))  
 elif (value == 'W'):  
 df = pd.DataFrame(list(shelter.readAll(  
 {"breed": {"$in": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]},  
 "sex\_upon\_outcome": "Intact Female", "age\_upon\_outcome\_in\_weeks": {"$lte": 156, "$gte": 26}})))  
 elif (value == 'M'):  
 df = pd.DataFrame(list(shelter.readAll({"breed": {  
 "$in": ["German Shephard", "Alaskan Malamute", "Old English Sheepdog", "Siberian Husky", "Rottweiler"]},  
 "sex\_upon\_outcome": "Intact Male",  
 "age\_upon\_outcome\_in\_weeks": {"$lte": 156, "$gte": 26}})))  
 elif (value == 'TR'):  
 df = pd.DataFrame(list(shelter.readAll(  
 {"breed": {"$in": ["Doberman Pinscher", "German Shephard", "Golden Retriever", "Bloodhound", "Rottweiler"]},  
 "sex\_upon\_outcome": "Intact Male", "age\_upon\_outcome\_in\_weeks": {"$lte": 300, "$gte": 20}})))  
  
 columns = [{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns]  
 data = df.to\_dict('records')  
  
 # return query  
 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]  
  
  
@app.callback(  
 Output('graph-id', "children"),  
 [Input('datatable-id', "derived\_viewport\_data")])  
# create pie chart graph  
def update\_graphs(viewData):  
 dff = pd.DataFrame.from\_dict(viewData) # read data  
 return [  
 dcc.Graph(  
 id='graph\_id',  
 figure=px.pie(dff,  
 values='age\_upon\_outcome\_in\_weeks', # values to display in pie chart  
 names='age\_upon\_outcome', # label names  
 title='Animal Ages Data Chart'  
 ),  
  
 )]  
  
  
@app.callback(  
 Output('map-id', "children"),  
 [Input('datatable-id', "derived\_viewport\_data")])  
def update\_map(viewData):  
 # *FIXME Add in the code for your geolocation chart* dff = pd.DataFrame.from\_dict(viewData)  
 # 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  
 dl.Marker(position=[dff.loc[0, 'location\_lat'], dff.loc[0, 'location\_long']], children=[  
 dl.Tooltip(dff.iloc[0, 4]),  
 dl.Popup([  
 html.H1("Animal Name"),  
 html.P(dff.loc[0, 'name'])  
 ])  
 ])  
 ])  
 ]  
  
  
app

### Results Screenshots

*Graphical user interface, text, application, Word

Description automatically generated*

*Pie Chart*

*Chart, pie chart

Description automatically generated*

*Geo-location Map*

*Graphical user interface, application

Description automatically generated*

*Drop-down Menu*

*Graphical user interface, text, application, email

Description automatically generated*

## Contact

Joan Estepan