# CS 340 README Template

## About the Project/Project Title

Global rain Dashboard

## Motivation

Global Rain is custom software design and development. To work on a project for an innovative international rescue-animal training company, Grazioso Salvare. He is seeking a software application that can work with existing data from the animal shelters to identify and categorize available dogs. Global Rain has contracted for a full stack development of this application, including a database and a client-facing web application dashboard through which users at Grazioso Salvare will access the database. Additionally, Grazioso Salvare has requested that the code for this project be open source and accessible on GitHub, so that it may be used and adapted by similar organizations

## Getting Started

Most of the crud process was done and code can be viewed on github. I will have to code the dashboard and the database interface logic. This will include dashboard attributes. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and training time.

## Installation

Tools: Jupyter Notebook to write and edit code. The software can also test code once completed

Pymongo

Mongodb

## Usage

*.*

### Code Example

Imports are important for starting:

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 os

import numpy as np

import pandas as pd

from pymongo import MongoClient

from bson.json\_util import dumps

# change animal\_shelter and AnimalShelter to match your CRUD Python module file name and class name

from Animal\_Shelter import AnimalShelter

import base64

#as well as your source CRUD file

#allowing secure access to the data

username = "aacuser"

password = "123456"

shelter = AnimalShelter(username,password)

# class read method must support return of cursor object

df = pd.DataFrame.from\_records(shelter.read({}))

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

# Dashboard Layout / View

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

app = JupyterDash('Andres Module')

#FIX ME Add in Grazioso Salvare’s logo

image\_filename = 'Grazioso Salvare Logo.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()))

app.layout = html.Div([

html.Div(id='hidden-div', style={'display':'none'}),

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

html.Center(html.B(html.H1('Andres Module CS-340 Dashboard'))),

html.Hr(),

html.Div(

#FIXME Add in code for the interactive filtering options. For example, Radio buttons, drop down, checkboxes, etc.

className='row',

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

children=[

html.Button(id='submit-button-one',n\_clicks=0, children= 'Water Rescue'),

html.Button(id='submit-button-two',n\_clicks=0, children= 'Mountain Rescue'),

html.Button(id='submit-button-three',n\_clicks=0, children='Disaster Rescue'),

html.Button(id='submit-button-four',n\_clicks=0, children='Scent Rescue'),

html.Button(id='submit-button-five', n\_clicks=0, children='reset')

]

),

html.Hr(),

dt.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

page\_size=100,

style\_table={'height':'300px','overflowY':'auto','overflowX':'auto'},

style\_header={

'backgroundColor':'rgb(255,255,180)',

'fontWeight':'bold'

},

style\_data={

'whiteSpace':'normal',

'height':'auto'

},

##these codes set the UI

#tooltips

tooltip ={i: {

'value': i,

'use\_with': 'both' # both refers to header & data cell

} for i in df.columns},

tooltip\_delay=0,

tooltip\_duration = None,

#sorting features that we are going to use

sort\_action='native',

sort\_mode='multi',

filter\_action='native',

editable=False,

column\_selectable=False,

row\_selectable='single',

row\_deletable=False,

selected\_rows=[],

),

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')],

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

Input('submit-button-one', 'n\_clicks'),Input('submit-button-two','n\_clicks'),

Input('submit-button-three','n\_clicks'),Input('submit-button-four','n\_clicks'),Input('submit-button-five', 'n\_clicks')])

def update\_dashboard(bt1,bt2,bt3,bt4,bt5):

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

if (int(bt1) >= 1):

df = pd.Dataframe.from\_records(shelter.read({'$and': [

{'$or': [ {'breed':'Labrador Retriever Mix'}, {'breed':'Chesapeake Bay Retriever'},

{'breed':'Newfoundland'}]},

{'sex\_upon\_outcome':'Intact Female'}, {'age\_upon\_outcome\_in\_weeks':{'$lte':26, 'gte':156}}]}))

bt2, bt3, bt4 = 0

elif (int(bt2)>= 1):

df = pd.Dataframe.from\_records(shelter.read({'$and': [

{'$or': [ {'breed':'German Shepherd'}, {'breed':'Alaskan Malamute'},

{'breed':'Old English Sheepdog'},{'breed':'Siberian Husky'},{'breed':'Rottweiler'}]},

{'sex\_upon\_outcome':'Intact Male'}, {'age\_upon\_outcome\_in\_weeks':{'$lte':26, 'gte':156}}]}))

bt1, bt3 ,bt4 = 0

elif (int(bt3)>=1):

df = pd.Dataframe.from\_records(shelter.read({'$and': [

{'$or': [ {'breed':'Doberman Pinscher'}, {'breed':'German Sheperd'},

{'breed':'Golden Retriever'},{'breed':'Bloodhound'},{'breed':'Rottweiler'}]},

{'sex\_upon\_outcome':'Intact Male'}, {'age\_upon\_outcome\_in\_weeks':{'$lte':20, 'gte':300}}]}))

bt1, bt2, bt4 = 0

elif(int(bt4)>=1):

df = pd.Dataframe.from\_records(shelter.read())

bt1, bt2, bt3 = 0

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

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

return data

@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': '#D3D2FF'

} for i in selected\_columns]

@app.callback(

Output('graph-id', "children"),

[Input('datatable-id', "derived\_viewport\_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, values=values, names=names, title='Percentage of breeds available')

)

]

@app.callback(Output('map-id', "children"),

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

Input('datatable-id',"derived\_viewport\_selected\_rows")])

def update\_map(viewData):

#FIXME: Add in the code for your geolocation chart

#If you completed the Module Six Assignment, you can copy in the code you created here.

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

dff = viewDF.loc[rows]

return [ dl.Map(style={'width': '1000px', 'height': '500px'}, center=[30.75,-97.48], zoom=15, children=[

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

# Marker with tool tip and pop up

dl.Marker(position=[dff.loc[0,'location\_lat'],dff.loc[0,'location\_long']], children=[

dl.Tooltip(dff['breed']),

dl.Popup([

html.H1("Animal Name"),

html.P(dff.loc[0,'name'])

])

])

])]

App

Ultimately I could not get my code to run a dashboard the closest I got was the sign in page that tested the username and password. However, at that step there still was not data incorporated into the jupyter file. I think a reviewing the section on the CRUD file and the setup within the jupyter file would help correct the error I was receiving.