# CS 340 README Template

## About the Project/Project Title

We were assigned a project for Grazioso Salvare to perform sort tasks on the Austin Animal Shelter database. The dashboard utilizes the AnimalShelter.py module which enables a connection to the database, and provides Create, Read, Update, and Delete functionality. The dashboard utilizes Plotly’s Dash, which is a tool that offers a way for users to interact with and visualize data with their Data Table. We also utilize Plotly Express for graphs and Leaflet for the map. Lastly, Pandas is utilized to create the DataFrame that holds our data to be manipulated.

## Motivation

This project allows Grazioso Salvare to quickly filter through the database to assess whether there are dogs available for training. The dashboard also creates a graph of available animals as well as a map for the selected animal’s location.

## Getting Started

PyMongo is the driver that allows a connection between MongoDB and the Python program. PyMongo is a lightweight tool that provides many functions that simplify the way this program works with MongoDB. To get a local copy running, you must download and import the AnimalShelter module. The program then creates an AnimalShelter object with the correct credentials and makes a “read” call to the AAC database. The search results are saved as a cursor in a Pandas DataFrame. Within the app object, the various html attributes can be edited as required, and the filter buttons may be renamed by changing their labels. The DataTable simply lists all queried rows, which happens when the app is ran. Within the callback section, the filters for the rescue types may be changed as required, by simply adding or removing breeds and other various attributes.

# Create method to implement the R in CRUD.

def read(self, data):

if data is not None:

return self.database.animals.find(data, { '\_id': 0} )

else:

raise Exception("Parameters returned no records")

I had started with a snippet of code that was provided to me from my professor. This was a starting point and information had to be changed and added to complete the python file. I used the module’s resources and researched more into what I didn’t understand from the resources into online searching. I watched a few videos on YouTube. Challenges I faced were figuring out where certain calls needed to be in the test file. Matching up what the python file was doing with what the test file needed to pull actions on was a challenge. Apporto semmed to be acting up at times. I spent time logging out and restarting. I finally started over with fresh files. As I worked through it it made more sense to me. I was able to see the bigger picture of the project.

## Installation

You will need Terminal, MongoDB, Python and Jupiter Notebook. I also used the libraries pymongo and bson (ObjectID). You must have access to the MongoDB server with valid credentials. You then need to have Python and Jupyter Notebook installed to run the application. Download the AnimalShelter.py module and the ProjectTwoDashboard KW2.ipynb script. You may need to edit the directories for AnimalShelter.py and the logo within the dashboard application. Here are my import settings.

“

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

import base64

“

## Usage

### Functionality

### To use the program, simply run the application and if everything connects properly, you should see a DataTable containing the AAC database rows. You should then be able to click on the various filter buttons, which will sort and filter the database with certain attributes. The map and graph should also change based on your filter request.

Graphical user interface, table

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with low confidence

Table

Description automatically generated

Graphical user interface

Description automatically generated

### Code Example

I really liked the way this code worked for creating the radio buttons.

html.Div(

dcc.RadioItems(

id='filter-type',

options=[

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

{'label': 'Mountain/Wilderness Rescue', 'value': 'mount'},

{'label': 'Disaster Rescue and Individual Tracking', 'value': 'disaster'},

{'label': 'Reset', 'value': 'reset'}

],

value='reset'

)

),

## Contact

Your name: Matthew Wilt