**CS 340 README Template**

**About the Project/Project Title**

This project is a “Dash” dashboard developed for visualizing and interacting with data from an animal shelter MongoDB database. The dashboard provides insights into rescue types, outcome types, geolocations of animals in the shelter, age, and more.

**Motivation**

The motivation behind this project is to create a user-friendly and interactive tool for exploring and understanding data from an animal shelter database. It aims to facilitate decision-making and analysis for stakeholders involved in animal welfare.

**Getting Started**

To get a local copy up and running, follow these simple steps.

**Installation**

1. Clone the repository:
2. Install dependencies:
3. Run the Dash application:

**Usage**

The dashboard allows users to filter data based on rescue types, view outcome types through a pie chart, and explore geolocations on an interactive map. Below are some examples of how the project can be used.

**Code Example:** # Update the data table based on the selected filter type

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

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

@app.callback(

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

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

Input('filter-type', 'value')] # Add the dropdown input here

)

def update\_graphs(viewData, filter\_type):

if viewData is None:

return []

# Check if viewData is a list and convert it to a DataFrame

if isinstance(viewData, list):

dff = pd.DataFrame(viewData)

else:

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

# Apply filtering based on the selected option from the dropdown

if filter\_type == 'Water Rescue':

dff\_filtered = dff[dff['breed'].isin(['Labrador Retriever Mix', 'Chesapeake Bay Retriever', 'Newfoundland'])]

elif filter\_type == 'Mountain or Wilderness Rescue':

dff\_filtered = dff[dff['breed'].isin(['German Shepherd', 'Alaskan Malamute', 'Old English Sheepdog', 'Siberian Husky', 'Rottweiler'])]

elif filter\_type == 'Disaster Rescue or Individual Tracking':

dff\_filtered = dff[dff['breed'].isin(['Doberman Pinscher', 'German Shepherd', 'Golden Retriever', 'Bloodhound', 'Rottweiler'])]

else:

dff\_filtered = dff # No filtering if 'Reset'

# Count the occurrences of each outcome type

outcome\_counts = dff\_filtered['outcome\_type'].value\_counts()

# Create a pie chart using Plotly Express

pie\_chart = dcc.Graph(

figure=px.pie(names=outcome\_counts.index, values=outcome\_counts.values, title='Outcome Types')

)

return data\_table

**Tests**

To run tests, you can use the following command:

pytest tests/

**Screenshots:**

A screenshot of a computer

Description automatically generated

Initial dashboard, logo, identifier, table, and drop down widget

A screenshot of a computer

Description automatically generatedInitial dashboard table and charts

A screenshot of a computer

Description automatically generated

Water rescue filter

A screenshot of a computer

Description automatically generated

Mountain filter

A screenshot of a computer

Description automatically generated

Disaster rescue

A screenshot of a computer

Description automatically generated

“reset” filter and highlighted cell