WORKED 1

import folium

import pandas as pd

# Function to create the map

def create\_interactive\_map(csv\_file, output\_map\_file='interactive\_map.html'):

try:

# Load the CSV file into a DataFrame

data = pd.read\_csv('Project data2\_out.csv')

# Print the column names for debugging

print(f"Columns in CSV: {data.columns.tolist()}")

# Required columns

required\_columns = ['latitude', 'longitude', 'name', 'location', 'type']

# Check if all required columns exist (case insensitive)

data.columns = data.columns.str.strip() # Strip any leading/trailing spaces

missing\_columns = [col for col in required\_columns if col not in data.columns]

if missing\_columns:

raise ValueError(f"CSV file is missing the following required columns: {', '.join(missing\_columns)}")

# Create a base map centered around the average of all provided coordinates

start\_coords = (data['latitude'].mean(), data['longitude'].mean())

interactive\_map = folium.Map(location=start\_coords, zoom\_start=10)

# Plot each data point

for \_, row in data.iterrows():

folium.Marker(

location=(row['latitude'], row['longitude']),

popup=f"Name: {row['name']}<br>Location: {row['location']}<br>Type: {row['type']}",

icon=folium.Icon(color='blue', icon='info-sign')

).add\_to(interactive\_map)

# Save the map to an HTML file

interactive\_map.save(output\_map\_file)

print(f"Map has been created and saved as {output\_map\_file}")

except FileNotFoundError:

print(f"Error: The file {csv\_file} was not found.")

except pd.errors.EmptyDataError:

print("Error: The CSV file is empty.")

except ValueError as ve:

print(f"ValueError: {ve}")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

csv\_file\_path = 'Project data2\_out.csv' # Ensure this is the correct path to your CSV file

create\_interactive\_map(csv\_file\_path)

#display map

#save map

WORKED 2

import folium

import pandas as pd

# Function to create the map

def create\_interactive\_map(csv\_file, output\_map\_file='interactive\_map.html'):

try:

# Load the CSV file into a DataFrame

data = pd.read\_csv(csv\_file)

# Print the column names for debugging

print(f"Columns in CSV: {data.columns.tolist()}")

# Required columns

required\_columns = ['latitude', 'longitude', 'name', 'location', 'type']

# Check if all required columns exist (case insensitive)

data.columns = data.columns.str.strip() # Strip any leading/trailing spaces

missing\_columns = [col for col in required\_columns if col not in data.columns]

if missing\_columns:

raise ValueError(f"CSV file is missing the following required columns: {', '.join(missing\_columns)}")

# Create a base map centered around the average of all provided coordinates

start\_coords = (data['latitude'].mean(), data['longitude'].mean())

interactive\_map = folium.Map(location=start\_coords, zoom\_start=10,

tiles='https://{s}.tile.stamen.com/terrain/{z}/{x}/{y}.png',

attr='Map tiles by <a href="http://stamen.com">Stamen Design</a>,

under <a href="http://creativecommons.org/licenses/by/3.0">CC BY 3.0</a>.

Data by <a href="http://openstreetmap.org">OpenStreetMap</a>, under <a href="http://www.openstreetmap.org/copyright">ODbL</a>.')

# Plot each data point with a label

for \_, row in data.iterrows():

folium.Marker(

location=(row['latitude'], row['longitude']),

popup=f"Name: {row['name']}<br>Location: {row['location']}<br>Type: {row['type']}",

icon=folium.Icon(color='blue', icon='info-sign')

).add\_to(interactive\_map)

# Add a permanent label for the name

folium.Marker(

location=(row['latitude'], row['longitude']),

icon=folium.DivIcon(

html=f"""<div style="font-size: 12pt; color : black">{row['name']}</div>"""

)

).add\_to(interactive\_map)

# Save the map to an HTML file

interactive\_map.save(output\_map\_file)

print(f"Map has been created and saved as {output\_map\_file}")

except FileNotFoundError:

print(f"Error: The file {csv\_file} was not found.")

except pd.errors.EmptyDataError:

print("Error: The CSV file is empty.")

except ValueError as ve:

print(f"ValueError: {ve}")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

csv\_file\_path = 'Project data2\_out.csv' # Ensure this is the correct path to your CSV file

create\_interactive\_map(csv\_file\_path)

ALMOST WORKED

import pandas as pd

import gmplot

# Function to create an interactive Google map

def create\_google\_map(csv\_file, output\_map\_file='google\_map.html', api\_key='AIzaSyB3VGClTSNkdcqiz6woFpSXsycio9lCzoE'):

# Load the CSV file into a DataFrame

data = pd.read\_csv('Project data2\_out.csv')

# Check if required columns exist

required\_columns = ['latitude', 'longitude', 'name', 'location', 'type']

if not all(column in data.columns for column in required\_columns):

raise ValueError(f"CSV file must contain the following columns: {', '.join(required\_columns)}")

# Extract latitude and longitude columns

latitudes = data['latitude']

longitudes = data['longitude']

# Create a Google Map centered around the first data point

gmap = gmplot.GoogleMapPlotter(latitudes.mean(), longitudes.mean(), 10, apikey=api\_key)

# Plot each point on the map

for \_, row in data.iterrows():

gmap.marker(row['latitude'], row['longitude'], title=f"{row['name']} - {row['location']} ({row['type']})")

# Save the map to an HTML file

gmap.draw(output\_map\_file)

print(f"Google Map has been created and saved as {output\_map\_file}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

csv\_file\_path = 'Project data2\_out.csv' # Path to your CSV file

google\_maps\_api\_key = 'AIzaSyB3VGClTSNkdcqiz6woFpSXsycio9lCzoE' # Replace with your actual Google Maps API Key

create\_google\_map(csv\_file\_path, api\_key=google\_maps\_api\_key)

!pip install 'gmplot'

FINAL CODE

import folium

import pandas as pd

# Function to create the map

def create\_interactive\_map(csv\_file, output\_map\_file='interactive\_map.html'):

try:

# Load the CSV file into a DataFrame

data = pd.read\_csv(csv\_file)

# Print the column names for debugging

print(f"Columns in CSV: {data.columns.tolist()}")

# Required columns

required\_columns = ['latitude', 'longitude', 'name', 'location', 'type']

# Check if all required columns exist (case insensitive)

data.columns = data.columns.str.strip() # Strip any leading/trailing spaces

missing\_columns = [col for col in required\_columns if col not in data.columns]

if missing\_columns:

raise ValueError(f"CSV file is missing the following required columns: {', '.join(missing\_columns)}")

# Create a base map centered around the average of all provided coordinates

start\_coords = (data['latitude'].mean(), data['longitude'].mean())

interactive\_map = folium.Map(location=start\_coords, zoom\_start=10)

# Add tile layers for road networks and satellite view

folium.TileLayer('openstreetmap', name='Road Map').add\_to(interactive\_map) # Default road network

folium.TileLayer('Stamen Toner', name='Stamen Toner').add\_to(interactive\_map) # Black and white road map

folium.TileLayer('Stamen Terrain', name='Terrain Map').add\_to(interactive\_map) # Terrain map

folium.TileLayer(

tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World\_Imagery/MapServer/tile/{z}/{y}/{x}',

attr='Esri',

name='Satellite',

overlay=False,

control=True

).add\_to(interactive\_map) # Satellite view

# Add Layer control to toggle between different map views

folium.LayerControl().add\_to(interactive\_map)

# Plot each data point with a label

for \_, row in data.iterrows():

folium.Marker(

location=(row['latitude'], row['longitude']),

popup=f"Name: {row['name']}<br>Location: {row['location']}<br>Type: {row['type']}",

icon=folium.Icon(color='blue', icon='info-sign')

).add\_to(interactive\_map)

# Add a permanent label for the name

folium.Marker(

location=(row['latitude'], row['longitude']),

icon=folium.DivIcon(

html=f"""<div style="font-size: 12pt; color : black">{row['name']}</div>"""

)

).add\_to(interactive\_map)

# Save the map to an HTML file

interactive\_map.save(output\_map\_file)

print(f"Map has been created and saved as {output\_map\_file}")

except FileNotFoundError:

print(f"Error: The file {csv\_file} was not found.")

except pd.errors.EmptyDataError:

print("Error: The CSV file is empty.")

except ValueError as ve:

print(f"ValueError: {ve}")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

csv\_file\_path = 'Project data2\_out.csv' # Ensure this is the correct path to your CSV file

create\_interactive\_map(csv\_file\_path)