import streamlit as st

import pandas as pd

import folium

from folium.plugins import MarkerCluster

from streamlit\_folium import st\_folium

import altair as alt

from wordcloud import WordCloud

from io import BytesIO

st.set\_page\_config(page\_title="GeoCycle KE – Eldoret", layout="wide")

@st.cache\_data

def load\_csv(path):

df = pd.read\_csv(path)

# safety

df["Latitude"] = pd.to\_numeric(df["Latitude"], errors="coerce")

df["Longitude"] = pd.to\_numeric(df["Longitude"], errors="coerce")

df = df.dropna(subset=["Latitude","Longitude"])

# helpers if missing

for c in ["\_WasteCategory","\_Alert","\_ReasonsNormalized","\_InterventionsNormalized"]:

if c not in df.columns: df[c] = "" if c != "\_Alert" else False

return df

df = load\_csv("GeoCycle\_Dashboard\_Ready\_Final.csv")

st.title("♻️ GeoCycle KE – Eldoret Waste Decision-Support")

st.caption("Photos • Actors • Alerts • Folium + Kepler.gl • Aligned with Kenya’s Sustainable Waste Management Act (2022)")

# ---- Sidebar filters

with st.sidebar:

st.header("Filters")

wards = sorted(df["Ward"].dropna().astype(str).unique())

wastes = sorted(df["Waste Types"].dropna().astype(str).unique())

actors = sorted(df["Waste Management Actors"].dropna().astype(str).unique())

sel\_wards = st.multiselect("Ward(s)", wards, default=wards)

sel\_wastes = st.multiselect("Waste type(s)", wastes, default=wastes)

sel\_actors = st.multiselect("Actors", actors, default=actors)

alerts\_only = st.checkbox("Only show Health/Burning alerts", value=False)

mask = pd.Series(True, index=df.index)

if sel\_wards: mask &= df["Ward"].astype(str).isin(sel\_wards)

if sel\_wastes: mask &= df["Waste Types"].astype(str).isin(sel\_wastes)

if sel\_actors: mask &= df["Waste Management Actors"].astype(str).isin(sel\_actors)

if alerts\_only: mask &= df["\_Alert"] == True

dfv = df[mask].copy()

# ---- KPIs

c1,c2,c3,c4 = st.columns(4)

c1.metric("Dumpsites (filtered)", len(dfv))

c2.metric("Wards", dfv["Ward"].nunique())

c3.metric("Waste types", dfv["Waste Types"].nunique())

c4.metric("Alerts (health/burning)", int(dfv["\_Alert"].sum()))

# ---- Folium map

def color\_for(cat, alert=False):

if alert: return "red"

return {

"Organic":"green","Plastic":"red","Paper":"orange","Glass":"blue",

"Metal":"gray","E-waste":"black","Mixed":"purple","Others":"lightgray",

"Unknown":"beige"

}.get(str(cat), "blue")

center = [dfv["Latitude"].mean() if len(dfv) else 0.5167,

dfv["Longitude"].mean() if len(dfv) else 35.2833]

m = folium.Map(location=center, zoom\_start=12)

mc = MarkerCluster().add\_to(m)

for \_, r in dfv.iterrows():

photo = r.get("Photo URL","")

img\_html = f'<br><img src="{photo}" width="220">' if isinstance(photo,str) and photo.strip() else ""

popup = (

f"<b>Dumpsite:</b> {r.get('Dumpsite Name','Unnamed')}"

f"<br><b>Ward:</b> {r.get('Ward','Unknown')}"

f"<br><b>Waste Types:</b> {r.get('Waste Types','—')}"

f"<br><b>Actors:</b> {r.get('Waste Management Actors','—')}"

f"<br><b>Community Action:</b> {r.get('Community Interventions','—')}"

f"<br><b>Reasons:</b> {r.get('Reasons for Dumping','—')}"

f"<br><b>Proposed Interventions:</b> {r.get('Proposed Interventions','—')}"

f"{img\_html}"

)

folium.Marker(

[float(r["Latitude"]), float(r["Longitude"])],

popup=popup,

icon=folium.Icon(

color=color\_for(r.get("\_WasteCategory","Unknown"), bool(r.get("\_Alert", False))),

icon='exclamation-sign' if bool(r.get("\_Alert", False)) else 'trash',

prefix='fa'

)

).add\_to(mc)

st\_folium(m, height=680, width=None)

# ---- Insights (stacked bar + reasons/interventions)

tab1, tab2 = st.tabs(["📊 Waste by Ward","🧩 Reasons & Interventions"])

with tab1:

d = dfv.copy()

d["\_cat"] = d["\_WasteCategory"].replace("", "Unknown")

if not d.empty:

agg = d.groupby(["Ward","\_cat"]).size().reset\_index(name="count")

chart = alt.Chart(agg).mark\_bar().encode(

x=alt.X("count:Q", title="Dumpsites"),

y=alt.Y("Ward:N", sort='-x', title="Ward"),

color=alt.Color("\_cat:N", title="Waste Type"),

tooltip=["Ward","\_cat","count"]

).properties(height=420)

st.altair\_chart(chart, use\_container\_width=True)

else:

st.info("No data in current filter.")

with tab2:

# Reasons bar

reasons\_tokens = []

for x in dfv["\_ReasonsNormalized"].dropna().astype(str):

reasons\_tokens += [p.strip() for p in x.split(",") if p.strip() and p.strip() != "<NA>"]

if reasons\_tokens:

reasons = pd.Series(reasons\_tokens).value\_counts().reset\_index()

reasons.columns = ["Reason","Count"]

st.subheader("Top Reasons for Dumping")

st.altair\_chart(

alt.Chart(reasons).mark\_bar().encode(

y=alt.Y("Reason:N", sort='-x'),

x=alt.X("Count:Q"),

tooltip=["Reason","Count"]

).properties(height=420),

use\_container\_width=True

)

else:

st.info("No non-empty 'Reasons for Dumping' yet — chart hidden.")

# Interventions word cloud

inter\_tokens = []

for x in dfv["\_InterventionsNormalized"].dropna().astype(str):

inter\_tokens += [p.strip() for p in x.split(",") if p.strip() and p.strip() != "<NA>"]

if inter\_tokens:

text = " ".join(inter\_tokens)

wc = WordCloud(width=1000, height=400, background\_color="white").generate(text)

buf = BytesIO()

wc.to\_image().save(buf, format="PNG")

st.subheader("Proposed Interventions — Word Cloud")

st.image(buf.getvalue(), use\_container\_width=True)

else:

st.info("No non-empty 'Proposed Interventions' yet — word cloud hidden.")

# ---- Data table + download

with st.expander("Show filtered data table"):

st.dataframe(dfv.reset\_index(drop=True))

@st.cache\_data

def to\_csv\_bytes(df\_in: pd.DataFrame) -> bytes:

return df\_in.to\_csv(index=False).encode("utf-8")

st.download\_button(

"⬇️ Download filtered CSV",

data=to\_csv\_bytes(dfv),

file\_name="GeoCycle\_filtered.csv",

mime="text/csv"

)

# ---- Kepler.gl WOW tab (optional later)

st.info("To add the Kepler.gl wow tab, we’ll enable it after deployment (it needs extra packages which are already listed).")