# ==============================================

# Enterprise-Grade Agentic AI App with Gemini LLM

# Streamlit Chat Interface + Prism Metrics Dashboard

# Deployed on Google Colab Enterprise (GCP) with Cloud Storage Integration

# ==============================================

# --- 1. Install & Import Dependencies ---

import streamlit as st

import google.generativeai as genai

import pandas as pd

import os

from google.cloud import storage

import io

# --- 2. Configure Gemini LLM ---

# Use secrets management in production (Colab Enterprise supports st.secrets)

# In Colab: Add GEMINI\_API\_KEY to secrets via UI or .streamlit/secrets.toml

genai.configure(api\_key=st.secrets["GEMINI\_API\_KEY"])

# Initialize Gemini model (agentic behavior enabled)

model = genai.GenerativeModel('gemini-1.5-pro')

# --- 3. GCP Cloud Storage Client ---

@st.cache\_resource

def get\_gcs\_client():

return storage.Client()

client = get\_gcs\_client()

# --- 4. Load Prism Data from basedatadmin Bucket ---

@st.cache\_data(ttl=3600) # Cache for 1 hour

def load\_prism\_data(bucket\_name="basedatadmin", file\_path="prism\_data.csv"):

try:

bucket = client.bucket(bucket\_name)

blob = bucket.blob(file\_path)

data = blob.download\_as\_bytes()

df = pd.read\_csv(io.BytesIO(data))

st.success(f"Loaded prism\_data from gs://{bucket\_name}/{file\_path}")

return df

except Exception as e:

st.error(f"Failed to load data: {e}")

return pd.DataFrame()

# --- 5. Load Subcon Data (Assume scalar or aggregated; adjust path if structured) ---

@st.cache\_data(ttl=3600)

def load\_subcon\_data(bucket\_name="basedatadmin", file\_path="subcon\_data.csv"):

try:

bucket = client.bucket(bucket\_name)

blob = bucket.blob(file\_path)

data = blob.download\_as\_bytes()

subcon\_df = pd.read\_csv(io.BytesIO(data))

# Assume subcon\_data is total subcontractor FTE (scalar)

total\_subcon = subcon\_df['subcon\_fte'].sum()

return total\_subcon

except Exception as e:

st.warning(f"Subcon data not found. Using 0. Error: {e}")

return 0

# --- 6. Calculate Utilization Metrics ---

def calculate\_utilization\_metrics(prism\_df, subcon\_fte):

if prism\_df.empty:

return pd.DataFrame()

# Ensure required columns exist

required\_cols = ['EDL\_Name', 'Rev\_Mapping', 'fte', 'status', 'dbo\_grade']

if not all(col in prism\_df.columns for col in required\_cols):

st.error("Missing required columns in prism\_data.")

return pd.DataFrame()

# Group by EDL\_Name and Rev\_Mapping

grouped = prism\_df.groupby(['EDL\_Name', 'Rev\_Mapping']).agg(

total\_fte=('fte', 'sum'),

billed\_fte=('fte', lambda x: prism\_df.loc[x.index, 'fte'][prism\_df.loc[x.index, 'status'] == 'billed'].sum()),

unbilled\_fte=('fte', lambda x: prism\_df.loc[x.index, 'fte'][prism\_df.loc[x.index, 'status'] == 'unbilled'].sum())

).reset\_index()

# TL+ Grades Filter

tl\_grades = ['AVP', 'DGM', 'Dir', 'GM', 'SD', 'SDM', 'TL', 'TM', 'VP']

tl\_mask = prism\_df['dbo\_grade'].isin(tl\_grades) & (prism\_df['status'] == 'unbilled')

tl\_df = prism\_df[tl\_mask]

tl\_grouped = tl\_df.groupby(['EDL\_Name', 'Rev\_Mapping']).agg(

unbilled\_fte\_tl=('fte', 'sum')

).reset\_index()

# Merge

metrics = pd.merge(grouped, tl\_grouped, on=['EDL\_Name', 'Rev\_Mapping'], how='left').fillna(0)

# --- Metric Calculations ---

metrics['overall\_utilization\_%'] = (metrics['billed\_fte'] / metrics['total\_fte']) \* 100

metrics['nbl\_utilization\_%'] = (metrics['unbilled\_fte'] / (metrics['total\_fte'] + subcon\_fte)) \* 100

metrics['nbl\_utilization\_tl\_%'] = (metrics['unbilled\_fte\_tl'] / metrics['total\_fte']) \* 100

metrics['nbl\_utilization\_tl\_subcon\_%'] = (metrics['unbilled\_fte\_tl'] / (metrics['total\_fte'] + subcon\_fte)) \* 100

# Round to 2 decimals

pct\_cols = [col for col in metrics.columns if '%' in col]

metrics[pct\_cols] = metrics[pct\_cols].round(2)

return metrics

# --- 7. Streamlit App Layout ---

st.set\_page\_config(page\_title="Agentic AI - Utilization Dashboard", layout="wide")

st.title("Agentic AI Utilization Dashboard with Gemini")

# Load data

with st.spinner("Loading data from GCP Cloud Storage..."):

prism\_df = load\_prism\_data()

subcon\_fte = load\_subcon\_data()

# Calculate metrics

metrics\_df = calculate\_utilization\_metrics(prism\_df, subcon\_fte)

# --- 8. Default Metrics Display (on load) ---

if not metrics\_df.empty:

st.subheader("Utilization Metrics by EDL\_Name & Rev\_Mapping")

st.dataframe(

metrics\_df[[

'EDL\_Name', 'Rev\_Mapping',

'overall\_utilization\_%',

'nbl\_utilization\_%',

'nbl\_utilization\_tl\_%',

'nbl\_utilization\_tl\_subcon\_%'

]].style.format("{:.2f}%", subset=[col for col in metrics\_df.columns if '%' in col]),

use\_container\_width=True

)

else:

st.info("No data available. Check GCS bucket and file paths.")

# --- 9. Chat Interface (Agentic AI Powered by Gemini) ---

st.markdown("---")

st.subheader("Ask Anything About Utilization, Trends, or Insights")

# Initialize chat history

if "messages" not in st.session\_state:

st.session\_state.messages = []

# Display chat messages

for message in st.session\_state.messages:

with st.chat\_message(message["role"]):

st.markdown(message["content"])

# Chat input

if prompt := st.chat\_input("Ask about utilization, EDL performance, or forecasts..."):

st.session\_state.messages.append({"role": "user", "content": prompt})

with st.chat\_message("user"):

st.markdown(prompt)

with st.chat\_message("assistant"):

with st.spinner("Thinking..."):

# Contextualize prompt with data summary

context = f"""

Current Utilization Summary:

- Total FTE: {prism\_df['fte'].sum():.2f}

- Billed FTE: {prism\_df[prism\_df['status']=='billed']['fte'].sum():.2f}

- Unbilled FTE: {prism\_df[prism\_df['status']=='unbilled']['fte'].sum():.2f}

- Subcontractor FTE: {subcon\_fte}

- Metrics table available above.

User Question: {prompt}

Respond concisely and actionably. Use tables if needed.

"""

response = model.generate\_content(context)

st.markdown(response.text)

st.session\_state.messages.append({"role": "assistant", "content": response.text})

# --- 10. Sidebar: Configuration & Info ---

with st.sidebar:

st.header("Configuration")

st.write("\*\*GCS Bucket:\*\* `basedatadmin`")

st.write("\*\*Prism File:\*\* `prism\_data.csv`")

st.write("\*\*Subcon File:\*\* `subcon\_data.csv`")

st.caption("Update paths in code if needed.")

st.markdown("---")

st.write("\*\*Gemini Model:\*\* gemini-1.5-pro")

st.write("\*\*Platform:\*\* Google Colab Enterprise (GCP)")

st.write("\*\*Auth:\*\* Service Account + Secrets")

if st.button("Clear Cache & Reload Data"):

st.cache\_data.clear()

st.success("Cache cleared. Reload the app.")

st.experimental\_rerun()