""" Resume Matcher & Personality Analyzer Created by Ferdaus — Built to help job seekers reflect and improve """

import streamlit as st import requests import json import time

--- Page Setup ---

st.set_page_config(page_title="Resume Matcher", layout="centered") st.title("🧠 Resume Matcher & Personality Analyzer")

--- Input Fields ---

job_description = st.text_area("📄 Job Description", height=200) resume_text = st.text_area("📝 Your Resume", height=200)

--- API Setup ---

API_KEY = "" # Add your API key here MODEL_ID = "resume-matcher-v1" API_ENDPOINT = f"https://api.example.com/models/{MODEL_ID}/analyze?key={API_KEY}"

--- Prompt Construction ---

request_text = f""" Compare the following resume with the job description. Highlight keyword matches, missing terms, and describe the candidate’s tone and personality. Return results in JSON format.

Job Description: {job_description}

Resume: {resume_text} """

--- Retry Logic ---

def get_analysis(prompt, retries=5): payload = { "contents": [{"parts": [{"text": prompt}]}], "generationConfig": { "responseMimeType": "application/json", "responseSchema": { "type": "OBJECT", "properties": { "match_percentage": {"type": "NUMBER"}, "matched_keywords": {"type": "ARRAY", "items": {"type": "STRING"}}, "missing_keywords": {"type": "ARRAY", "items": {"type": "STRING"}}, "tone_summary": {"type": "STRING"}, "resume_focus_summary": {"type": "STRING"} } } } }

```
headers = {"Content-Type": "application/json"}
for attempt in range(retries):
    try:
        response = requests.post(API_ENDPOINT, headers=headers,
data=json.dumps(payload))
        if response.status_code != 200:
            raise Exception(f"HTTP {response.status_code}")
        result = response.json()
        raw_text =
result['candidates'][0]['content']['parts'][0]['text']
        return json.loads(raw_text)
    except Exception as e:
        st.warning(f"Attempt {attempt+1} failed: {e}")
        time.sleep(2 ** attempt)
return None
```

--- Button ---

if st.button("🔍 Analyze Resume"): if not job_description or not resume_text: st.error("🛑 Please fill in both the job description and your resume.") else: with st.spinner("Analyzing..."): analysis = get_analysis(request_text) if analysis: score = round(analysis.get("match_percentage", 0)) st.subheader("📊 Match Score") st.progress(score / 100) st.metric("Match %", f"{score}%")

```
           if score >= 80:
                st.success("🎉 Excellent match! Your resume aligns
well.")
```

```
            elif score >= 50:
                st.info("👍 Good match. Consider adding missing
keywords.")
            else:
                st.warning("⚠️ Low match. Try tailoring your resume
more closely.")

            st.subheader("✅ Matched Keywords")
            st.write(analysis.get("matched_keywords", ["No strong
matches found."]))

            st.subheader("❌ Missing Keywords")
            st.write(analysis.get("missing_keywords", ["All key terms
appear present."]))

            st.subheader("👤 Personality & Tone")
            st.write(analysis.get("tone_summary", "Could not assess
tone."))

            st.caption(f"**Resume Focus:**
{analysis.get('resume_focus_summary', 'N/A')}")
        else:
            st.error("❌ Could not complete analysis. Please check
your API key or try again.")
```