**The Memes Creater**

import os, random﻿from io import BytesIO﻿from datetime import datetime﻿import requests﻿from PIL import Image, ImageDraw, ImageFont﻿from openai import OpenAI﻿# ---- API KEY ----﻿API\_KEY = "sk-proj-xxxx"  # 🔑 replace with your OpenAI API key﻿if not API\_KEY.startswith("sk-"):﻿    raise SystemExit("Please set your OpenAI API key in API\_KEY variable.")﻿client = OpenAI(api\_key=API\_KEY)﻿# ---- Folder for backgrounds ----﻿bg\_dir = "/storage/emulated/0/Pictures/meme\_backgrounds"﻿os.makedirs(bg\_dir, exist\_ok=True)﻿# ---- Background URLs (only 6 safe ones) ----﻿urls = [﻿    "https://picsum.photos/800/800?random=1",﻿    "https://picsum.photos/800/800?random=2",﻿    "https://picsum.photos/800/800?random=3",﻿    "https://picsum.photos/800/800?random=4",﻿    "https://picsum.photos/800/800?random=5",﻿    "https://picsum.photos/800/800?random=6",﻿]﻿# ---- Download once and cache ----﻿local\_images = []﻿for i, url in enumerate(urls):﻿    path = os.path.join(bg\_dir, f"bg\_{i}.jpg")﻿    if not os.path.exists(path):  # download if missing﻿        r = requests.get(url, timeout=30)﻿        with open(path, "wb") as f:﻿            f.write(r.content)﻿    local\_images.append(path)﻿# ---- Font Loader ----﻿def get\_font(size):﻿    for path in ["/system/fonts/Roboto-Bold.ttf", "/system/fonts/Roboto-Regular.ttf"]:﻿        if os.path.exists(path):﻿            return ImageFont.truetype(path, size)﻿    cache\_dir = "/storage/emulated/0/Pydroid3/deps"﻿    os.makedirs(cache\_dir, exist\_ok=True)﻿    font\_path = os.path.join(cache\_dir, "DejaVuSans-Bold.ttf")﻿    if not os.path.exists(font\_path):﻿        r = requests.get(﻿            "https://github.com/dejavu-fonts/dejavu-fonts/raw/master/ttf/DejaVuSans-Bold.ttf",﻿            timeout=30﻿        )﻿        with open(font\_path, "wb") as f:﻿            f.write(r.content)﻿    return ImageFont.truetype(font\_path, size)﻿# ---- AI Caption Generator ----﻿def generate\_captions(topic):﻿    prompt = (﻿        f"Create a funny meme about '{topic}' in exactly two lines.\n"﻿        "Do NOT use 'Top:', 'Bottom:', numbers, or labels.\n"﻿        "Just write two short, punchy lines (under 8 words each).\n"﻿        "Line 1 is the top caption, line 2 is the bottom caption."﻿    )﻿    resp = client.chat.completions.create(﻿        model="gpt-4o-mini",﻿        messages=[{"role": "user", "content": prompt}],﻿        temperature=0.9,﻿    )﻿    text = resp.choices[0].message.content﻿    lines = [line.strip() for line in text.split("\n") if line.strip()]﻿    clean = []﻿    for line in lines:﻿        line = line.replace("Top:", "").replace("Bottom:", "")﻿        line = line.lstrip("0123456789.-: ").strip()﻿        clean.append(line)﻿    return clean[0] if len(clean) > 0 else "", clean[1] if len(clean) > 1 else ""﻿# ---- Text Drawing ----﻿def draw\_text\_centered(draw, text, y, font, image\_width):﻿    words = text.split()﻿    lines, line = [], ""﻿    for word in words:﻿        test = (line + " " + word).strip()﻿        if draw.textbbox((0, 0), test, font=font)[2] < image\_width \* 0.9:﻿            line = test﻿        else:﻿            lines.append(line)﻿            line = word﻿    if line:﻿        lines.append(line)﻿    for line in lines:﻿        w = draw.textbbox((0, 0), line, font=font)[2]﻿        x = (image\_width - w) // 2﻿        draw.text((x, y), line, font=font,﻿                  fill="white", stroke\_width=3, stroke\_fill="black")﻿        y += font.size + 5﻿    return y﻿# ---- Main ----﻿topic = input("🎭 Enter meme topic: ").strip()﻿top\_text, bottom\_text = generate\_captions(topic)﻿num\_images = random.choice([2, 3, 4])﻿print(f"📸 Creating meme with {num\_images} images...")﻿chosen\_paths = random.sample(local\_images, num\_images)﻿images = [Image.open(p).convert("RGB").resize((800, 800)) for p in chosen\_paths]﻿final\_img = Image.new("RGB", (800, 800 \* num\_images))﻿for i, img in enumerate(images):﻿    final\_img.paste(img, (0, i \* 800))﻿draw = ImageDraw.Draw(final\_img)﻿font = get\_font(60)﻿draw\_text\_centered(draw, top\_text, 20, font, 800)﻿draw\_text\_centered(draw, bottom\_text, (num\_images - 1) \* 800 + 20, font, 800)﻿timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")﻿out\_path = f"/storage/emulated/0/Download/meme\_{timestamp}.jpg"﻿final\_img.save(out\_path, format="JPEG", quality=95)﻿print("✅ Meme saved as:", out\_path)