ASSIGNMENT – 3

AI Search Engine

Code:

```
import streamlit as st
import requests
from bs4 import BeautifulSoup
from fpdf import FPDF
import io
import base64
import mimetypes
import os
import re


OPENAI_API_KEY = ""
# GPT AI Response
def get_ai_response(topic, custom_instruction=None):
    url = "https://api.openai.com/v1/chat/completions"
    headers = {
        "Authorization": f"Bearer {OPENAI_API_KEY}",
        "Content-Type": "application/json"
    }


    system_prompt = custom_instruction or "You are Avishkar, an AI assistant providing insightful, practical, and clear responses."


    user_prompt = f"""Topic: {topic}


Please provide:
1. Brief Introduction (5-10 lines)
```

```
2. Positive aspects (if any)
3. Negative aspects (if any)

Do not mention the terms 'merits' or 'demerits'. Use appropriate headings."""

    payload = {
        "model": "gpt-3.5-turbo",
        "messages": [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": user_prompt}
        ]
    }
    response = requests.post(url, headers=headers, json=payload)
    return response.json()["choices"][0]["message"]["content"]

# WH query checker
def is_wh_query(topic):
    return bool(re.match(r"(?i)\b(what|who|why|how|where|when)\b", topic.strip()))

# Current affairs checker
def is_current_affairs(topic):
    keywords = ["today", "2024", "2025", "current", "news", "event", "recent", "match", "won"]
    return any(word.lower() in topic.lower() for word in keywords)

# Person query checker
def is_person_query(topic):
    person_keywords = ["who is", "biography", "life of", "person", "president", "leader", "founder",
"scientist", "actor", "author", "player"]
    return any(word.lower() in topic.lower() for word in person_keywords)

# Current affairs using GPT
```

```
def get_current_affairs_response(query):
    url = "https://api.openai.com/v1/chat/completions"
    headers = {
        "Authorization": f"Bearer {OPENAI_API_KEY}",
        "Content-Type": "application/json"
    }
    payload = {
        "model": "gpt-3.5-turbo",
        "messages": [
            {"role": "system", "content": "You are an AI assistant that provides up-to-date information about current affairs and events."},
            {"role": "user", "content": f"Please provide a brief and recent update or answer to this current affairs topic or question: {query}"}
        ]
    }
    response = requests.post(url, headers=headers, json=payload)
    return response.json()["choices"][0]["message"]["content"]

# GPT-4 Vision: Describe Uploaded Image
def describe_uploaded_image(image_file):
    mime_type, _ = mimetypes.guess_type(image_file.name)
    image_bytes = image_file.read()
    image_base64 = base64.b64encode(image_bytes).decode("utf-8")

    url = "https://api.openai.com/v1/chat/completions"
    headers = {
        "Authorization": f"Bearer {OPENAI_API_KEY}",
        "Content-Type": "application/json"
    }

    payload = {
        "model": "gpt-4-turbo",
```

```
        "messages": [
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": "Describe this image in detail."},
                    {"type": "image_url", "image_url": {"url": f"data:{mime_type};base64,{image_base64}"}}
                ]
            }
        ]
    }

    response = requests.post(url, headers=headers, json=payload)
    if response.status_code == 200:
        return response.json()["choices"][0]["message"]["content"]
    else:
        return f"❌ Error during image analysis: {response.status_code} - {response.json()}"

# PDF Generator
class UnicodePDF(FPDF):
    def _init_(self):
        super()._init_()
        self.add_page()
        font_path = os.path.join(os.getcwd(), "DejaVuSans.ttf")
        self.add_font("DejaVu", "", font_path, uni=True)
        self.set_font("DejaVu", size=12)

    def add_text(self, text):
        self.set_auto_page_break(auto=True, margin=15)
        for line in text.split("\n"):
            self.multi_cell(0, 10, line)
```

```
def generate_pdf(text):
    pdf = UnicodePDF()
    pdf.add_text(text)
    pdf_output = io.BytesIO()
    pdf_bytes = pdf.output(dest='S').encode('latin-1')
    pdf_output.write(pdf_bytes)
    return pdf_output.getvalue()

# Realistic DALL-E Image Generation
def generate_image_dalle(prompt):
    url = "https://api.openai.com/v1/images/generations"
    headers = {
        "Authorization": f"Bearer {OPENAI_API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "prompt": prompt + ", realistic photography, ultra-realistic style",
        "n": 1,
        "size": "512x512",
        "response_format": "url"
    }
    response = requests.post(url, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["data"][0]["url"]
    else:
        return None

# --- Streamlit App UI ---
st.set_page_config(page_title="Avishkar - AI Search Engine", layout="centered")
st.markdown("""<h1 style='text-align: center; font-family: Georgia, serif;'>🔍 <span
style='color:#ff6600;'>Avishkar</span> - AI Search Engine</h1>""", unsafe_allow_html=True)
```

```
st.markdown("Upload an *image* or enter a *topic* to get AI-generated insights and images.")


uploaded_image = st.file_uploader("📷 Upload an image (JPG, PNG)", type=["jpg", "jpeg", "png"])
query = st.text_input("✍️ Or enter a topic")
mode = st.radio("Choose display mode:", ["Text + Image", "Show only image"])

# Show query type
if query:
    if is_current_affairs(query):
        st.info("🗞️ Detected as a *Current Affairs* topic.")
    elif is_person_query(query):
        st.info("👤 Detected as a *Person-Based* topic.")

if st.button("🔎 Generate"):
    if uploaded_image:
        with st.spinner("Analyzing image..."):
            st.image(uploaded_image, use_container_width=True)
            result = describe_uploaded_image(uploaded_image)
            st.markdown("### 🖼️ Image Description")
            st.markdown(result)

            pdf_data = generate_pdf(result)
            b64 = base64.b64encode(pdf_data).decode()
            href = f'<a href="data:application/pdf;base64,{b64}" download="image_description.pdf">📄 Download Description as PDF</a>'
            st.markdown(href, unsafe_allow_html=True)

    elif query:
        if mode == "Show only image":
            if is_wh_query(query):
```

```
            st.warning("❗ Image generation is not supported for WH-type queries like 'what', 'why', 'how', etc.")
        else:
            image_url = generate_image_dalle(query)
            if image_url:
                st.image(image_url, caption=f"Image for '{query}'", use_container_width=True)
            else:
                st.error("❌ Couldn't generate image.")
    else:
        with st.spinner("Generating result..."):
            if not is_wh_query(quer
                if image_url:
                    st.image(image_url, caption=f"Image for '{query}'", use_container_width=True)

            if is_current_affairs(query):
                result = get_current_affairs_response(query)
            elif is_person_query(query):
                person_instruction = "You are Avishkar, an AI assistant who provides respectful, balanced, and factual information about people. Do not use the words 'merit' or 'demerit'. Use clear and appropriate section headings like 'Strengths' and 'Criticism'."
                result = get_ai_response(query, custom_instruction=person_instruction)
            else:
                result = get_ai_response(query)

            if result:
                st.markdown("### 🧠 Result")
                st.markdown(result)

                pdf_data = generate_pdf(result)
                b64 = base64.b64encode(pdf_data).decode()
                href = f'<a href="data:application/pdf;base64,{b64}" download="avishkar_result.pdf">📄 Download as PDF</a>'
```

```
            st.markdown(href, unsafe_allow_html=True)
        else:
            st.error("❌ Couldn't generate result.")
else:
    st.warning("⚠ Please upload an image or enter a topic.")
y):
            image_url = generate_image_dalle(query)
```

Description:

- AI-Powered Text Insights:
    - Uses **GPT-3.5 Turbo** to generate informative content on a given topic.
    - Supports different types of prompts: general, current affairs, WH-questions, and person-based queries.
- Image Upload & Analysis:
    - Accepts JPG/PNG image uploads.
    - Uses **GPT-4 Turbo Vision** to describe the uploaded image in detail.
    - Converts the image description to a downloadable **PDF**.
- Image Generation (DALL·E):
    - Generates realistic images using **DALL·E API** for non-WH type text queries.
- PDF Generation:
    - Generates and downloads a Unicode-compatible PDF using fpdf for both image and text outputs.

- WH Query Detector:

Detects questions like *what*, *why*, *how*, etc., and avoids image generation for these.

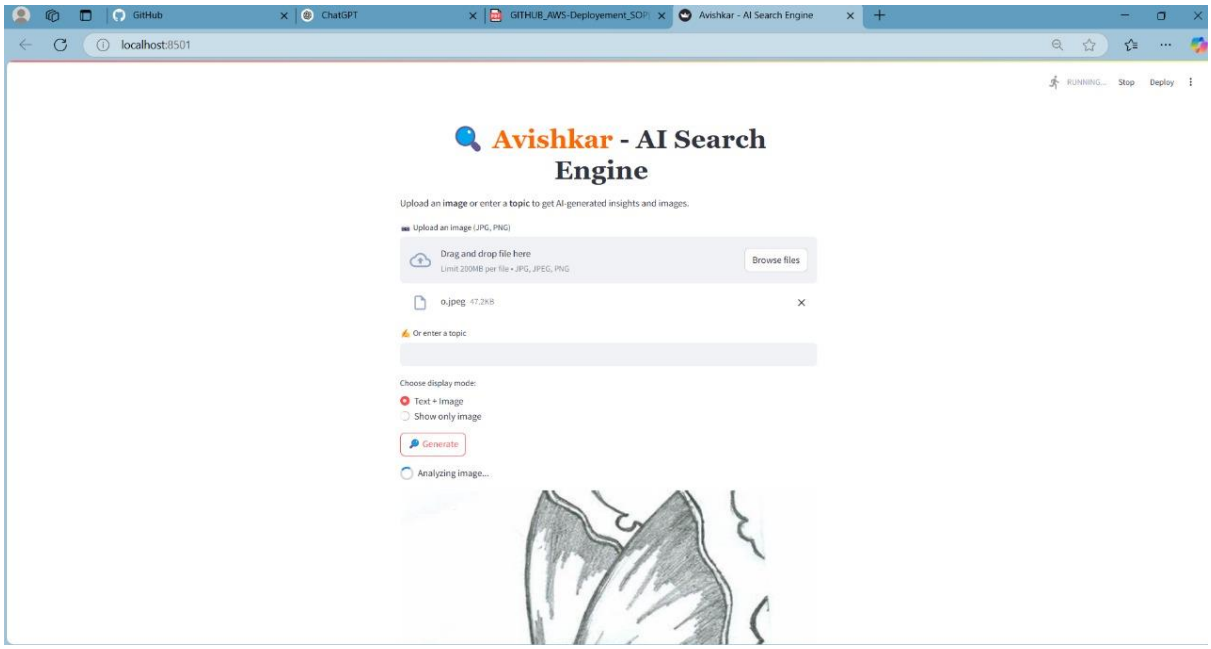
- Current Affairs Detector:

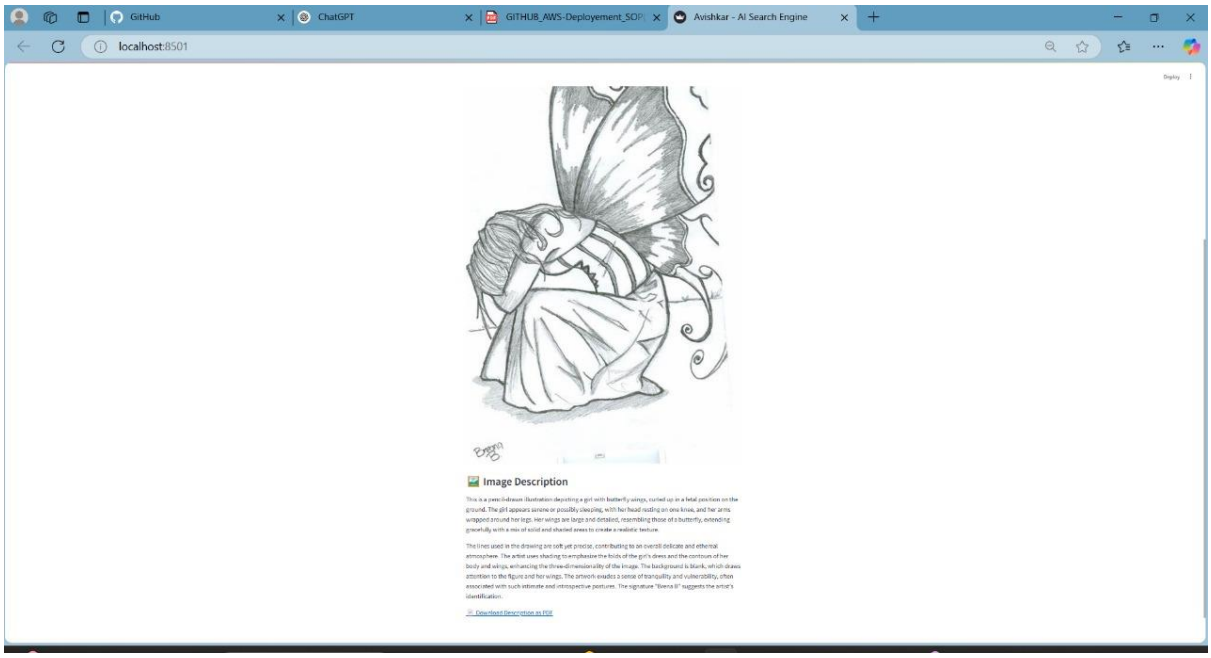
Recognizes topics related to recent events or dates like *2024*, *news*, *today*, etc.

- Person Query Detector:

Identifies if the input is about a person (e.g., *biography*, *leader*, *actor*).

Output:


GitHub
ChatGPT
GITHUB_AWS-Deployement_SOP
Avishkar - AI Search Engine
localhost:8501
RUNNING...
Stop
Deploy
Avishkar - AI Search Engine
Upload an image or enter a topic to get AI-generated insights and images.
Upload an image (JPG, PNG)
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
o.jpeg 47.2KB
Or enter a topic
Choose display mode:
Text + Image
Show only image
Generate
Analyzing image...


GitHub
ChatGPT
GITHUB_AWS-Deployement_SOP
Avishkar - AI Search Engine
localhost:8501
Image Description