Nama Kelompok : Capcai
KELOMPOK : 2
Dosen : Pak Kelvin S.Kom, M.Kom Anggota dan
Pembagian Tugas :
1. Justin Kosasi (221110323) = membantu pembuatan code combine.py, Bahan presentasi
2. Jordan (221111018) = membantu pembuatan code combine.py, presentasi
3. Sanjaya Citra (221111926) = membantu pembuatan code combine.py, Edit Video
4. Venderson Egy Agatran (221110627) = membantu pembuatan code combine.py, dokumentasi

Deskripsi Project : Membuat Chat-Bot, Image AI Descriptor, dan image generator dengan Google API Key menggunakan Bahasa pemograman Python

Terlebih dahulu lakukan :
buka command prompt pada windows
ketik pip install :
streamlit
google-generativeai
python-dotenv
textwrap3
transformers
diffusers
huggingface-hub
torch
torchvision
torchaudio
accelerate
Contoh : pip install streamlit

Cara menjalakankan code :
pada terminal ketik : Streamlit run combine.py

Screenshot code :

combine.py > get_gemini_image_response

```python
import streamlit as st
from PIL import Image
import google.generativeai as genai
import textwrap3
import os
from dotenv import load_dotenv
from huggingface_hub import login
from diffusers import StableDiffusionPipeline
import torch

# Load environment variables
load_dotenv()
api_key = os.getenv("GOOGLE_API_KEY")
genai.configure(api_key=api_key)

# Hugging Face token
hf_token = os.getenv("HUGGING_FACE_TOKEN")

# Authenticate to Hugging Face
login(hf_token)

def get_gemini_image_response(input_text, image):
    model = genai.GenerativeModel('gemini-pro-vision')
    if input_text != "":
        response = model.generate_content([input_text, image])
    else:
        response = model.generate_content(image)
    return response.text

def get_gemini_question_response(question):
    model = genai.GenerativeModel('gemini-pro')
    if question != "":
        response = model.generate_content(question)
        return response.text
    else:
        return "Need to input something"
```

combine.py

combine.py > get_gemini_image_response

```python
def to_markdown(text):
    text = text.replace('•', '  *')
    return textwrap3.indent(text, '> ', predicate=lambda _: True)

def generate_image_with_huggingface(prompt):
    pipe = StableDiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4", use_auth_token=hf_token)

    if torch.cuda.is_available():
        pipe.to("cuda")  # if you have a GPU
    else:
        pipe.to("cpu")  # if you don't have a GPU

    image = pipe(prompt).images[0]

    output_dir = "D:\\tmp"
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    image_path = os.path.join(output_dir, "generated_image.png")
    image.save(image_path)

    return Image.open(image_path)

st.set_page_config(page_title="Gemini Demo")

if 'chat_history' not in st.session_state:
    st.session_state['chat_history'] = []

options = ["Gemini Image Demo", "Q&A Demo", "Image response", "History"]
selected_option = st.sidebar.multiselect("Select Demo", options)

if "Gemini Image Demo" in selected_option:
    st.header("Gemini Image Demo")

    input_text = st.text_input("Input Prompt: ", key="input_image")
    uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])
```

combine.py

combine.py > get_gemini_image_response

```python
    if uploaded_file is not None:
        image = Image.open(uploaded_file)
        st.image(image, caption="Uploaded Image.", use_column_width=True)

    submit = st.button("Tell me about the image")
    if submit and uploaded_file is not None:
        response = get_gemini_image_response(input_text, image)
        st.subheader("The Response is")
        if input_text is None:
            st.session_state['chat_history'].append(("You", "-"))
        else:
            st.session_state['chat_history'].append(("You", input_text))
            st.session_state['chat_history'].append(("Bot", response))
        st.write(response)
    elif submit and uploaded_file is None:
        st.write("Please upload images")

if "Q&A Demo" in selected_option:
    st.header("Q&A Demo")

    input_question = st.text_input("Input: ", key="input_question")
    submit_question = st.button("Ask the question")
    if submit_question:
        response_question = get_gemini_question_response(input_question)
        st.subheader("The Response is")
        st.markdown(to_markdown(response_question))
        st.session_state['chat_history'].append(("You", input_question))
        st.session_state['chat_history'].append(("Bot", response_question))

if "Image response" in selected_option:
    st.header("Image Response Demo")

    prompt = st.text_input("Input Prompt for Image Generation: ", key="input_generate_image")
    generate_image = st.button("Generate Image")
    if generate_image:
        generated_image = generate_image_with_huggingface(prompt)
```

combine.py

combine.py > get_gemini_image_response

```python
    st.header("Q&A Demo")

    input_question = st.text_input("Input: ", key="input_question")
    submit_question = st.button("Ask the question")
    if submit_question:
        response_question = get_gemini_question_response(input_question)
        st.subheader("The Response is")
        st.markdown(to_markdown(response_question))
        st.session_state['chat_history'].append(("You", input_question))
        st.session_state['chat_history'].append(("Bot", response_question))

if "Image response" in selected_option:
    st.header("Image Response Demo")

    prompt = st.text_input("Input Prompt for Image Generation: ", key="input_generate_image")
    generate_image = st.button("Generate Image")
    if generate_image:
        generated_image = generate_image_with_huggingface(prompt)
        st.image(generated_image, caption="Generated Image.", use_column_width=True)
        st.session_state['chat_history'].append(("You", prompt))
        st.session_state['chat_history'].append(("Bot", "Generated an image based on the prompt"))

if "History" in selected_option:
    st.header("History")
    for role, text in st.session_state['chat_history']:
        st.write(f"{role}: {text}")
        st.image

```