# Import NLTK library to use stopwords module

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

# Import web\_pilot plugin to enhance web search and query functionality

import web\_pilot

# Import gpt\_3 library to use OpenAI's GPT-3 natural language generation model

import gpt\_3

# Import instagram\_format library to use Instagram's formatting options for captions and cover photos

import instagram\_format

# Import image\_generator library to use different image generation models, such as StyleGAN, BigGAN, or CLIP

import image\_generator

# Import image\_editor library to use different image editing options, such as cropping, resizing, rotating, or adding filters or stickers

import image\_editor

# Import image\_previewer library to preview the generated image before sending it to the chat

import image\_previewer

# Import image\_saver library to save or share the generated image on social media platforms, such as Instagram, Facebook, or Twitter

import image\_saver

# Import bing\_image\_viewer library to display images in the chat box

import bing\_image\_viewer

# Define a function to get user inputs

def get\_user\_inputs():

# Prompt user to enter topic URL

topic\_url = input("Enter the topic: ")

# Prompt user to enter any additional details

details = input("Enter any additional details: ")

# Prompt user to enter any keywords

keywords = input("Enter any keywords: ").split(",")

# Prompt user to enter the context or purpose

context = input("Enter the context: ")

# Prompt user to enter the tone

tone = input("Enter the tone: ")

# Prompt user to enter the style

style = input("Enter the style: ")

# Prompt user to enter the personality

personality = input("Enter the personality: ")

# Prompt user to enter the format

format = input("Enter the format: ").split(",")

# Prompt user to enter the resolution

resolution = input("Enter the resolution: ")

# Prompt user to enter the quality

quality = input("Enter the quality: ")

# Prompt user to enter the style

style = input("Enter the style: ")

# Prompt user to enter the textures

textures = input("Enter the textures: ").split(",")

# Prompt user to enter the feedback

feedback = input("Enter the feedback: ")

# Prompt user to enter Instagram account URL

ig\_account\_url = input("Enter your Instagram account URL: ")

# Prompt user to choose an image generation model

image\_model = input("Choose an image generation model from StyleGAN, BigGAN, or CLIP: ")

# Prompt user to customize the image generation parameters

image\_params = input("Customize the image generation parameters (number of images, diversity, style transfer, color scheme): ").split(",")

# Convert all inputs to lowercase

topic\_url = topic\_url.lower()

details = details.lower()

keywords = [keyword.lower() for keyword in keywords]

context = context.lower()

tone = tone.lower()

style = style.lower()

personality = personality.lower()

format = [item.lower() for item in format]

resolution = resolution.lower()

quality = quality.lower()

style = style.lower()

textures = [texture.lower() for texture in textures]

feedback = feedback.lower()

ig\_account\_url = ig\_account\_url.lower()

image\_model = image\_model.lower()

image\_params = [param.lower() for param in image\_params]

Remove any stop words from the inputs

stop\_words = set(stopwords.words('english'))

topic\_url = ' '.join([word for word in topic\_url.split() if word not in stop\_words])

details = ' '.join([word for word in details.split() if word not in stop\_words])

keywords = [word for word in keywords if word not in stop\_words]

context = ' '.join([word for word in context.split() if word not in stop\_words])

Combine all inputs into one string

input\_string = f'{topic\_url} {details} {" ".join(keywords)} {context}'

Create a dictionary of user inputs

user\_inputs = {

"topic\_url": topic\_url,

"details": details,

"keywords": keywords,

"context": context,

"tone": tone,

"style": style,

"personality": personality,

"format": format,

"resolution": resolution,

"quality": quality,

"style": style,

"textures": textures,

"feedback": feedback,

"ig\_account\_url": ig\_account\_url,

"input\_string": input\_string,

"image\_model": image\_model,

"image\_params": image\_params

}

# Define a function to generate the content type

def generate\_content\_type(user\_inputs):

# Use web\_pilot to search for relevant information based on the user's inputs and sources

web\_pilot.search(user\_inputs["input\_string"], sources = [user\_inputs["topic\_url"]])

# Use gpt\_3 to generate the caption based on the user's inputs and sources, and the web search results

caption = gpt\_3.generate\_caption(user\_inputs, web\_pilot.results)

# Use image\_generator to generate the image based on the user's inputs and sources, and the web search results

image = image\_generator.generate\_image(user\_inputs, web\_pilot.results)

# Use image\_editor to edit or modify the image based on the user's preferences

image = image\_editor.edit\_image(image, user\_inputs)

# Use image\_previewer to preview the image before sending it to the chat

image\_previewer.preview\_image(image)

# Use bing\_image\_viewer to display the image in the chat box

bing\_image\_viewer.display\_image(image)

# Use instagram\_format to format the caption and the image according to Instagram's standards

instagram\_format.format\_caption(caption)

instagram\_format.format\_image(image)

# Use image\_saver to save or share the image on social media platforms, such as Instagram, Facebook, or Twitter

image\_saver.save\_image(image, user\_inputs["ig\_account\_url"])

image\_saver.share\_image(image, user\_inputs["ig\_account\_url"])

# Return the caption and the image as the final output

return caption, image

# Call the get\_user\_inputs function to get the user's inputs

user\_inputs = get\_user\_inputs()

# Call the generate\_content\_type function to generate the content type based on the user's inputs

caption, image = generate\_content\_type(user\_inputs)

# Print the caption and display the image in the chat box

print(caption)

bing\_image\_viewer.display\_image(image)