```
import os
import requests
from bs4 import BeautifulSoup
from typing import List
from dotenv import load_dotenv
from openai import OpenAI
import google.generativeai
import anthropic
import gradio as gr
```

Hàm	Chức năng
os	Quản lý biến môi trường, đường dẫn file.
requests	Gửi yêu cầu HTTP để lấy dữ liệu web.
BeautifulSoup	Phân tích HTML để trích xuất dữ liệu.
load_dotenv()	Tải biến môi trường từ file .env.
OpenAI	Kết nối OpenAI API để xử lý văn bản/gọi mô hình AI.
google.generativeai	Dùng AI của Google để tạo văn bản.
anthropic	Gọi API của Claude AI từ Anthropic.
gradio	Tạo giao diện web đơn giản cho chatbot.

```
load_dotenv(override=True)
openai_api_key = os.getenv('OPENAI_API_KEY')
anthropic_api_key = os.getenv('ANTHROPIC_API_KEY')
google_api_key = os.getenv('GOOGLE_API_KEY')

if openai_api_key:
    print(f"OpenAI API Key exists and begins {openai_api_key[:8]}")
else:
    print("OpenAI API Key not set")

if anthropic_api_key:
    print(f"Anthropic API Key exists and begins
{anthropic_api_key[:7]}")
else:
    print("Anthropic API Key not set")

if google_api_key:
    print(f"Google API Key exists and begins {google_api_key[:8]}")
else:
    print("Google API Key not set")
```

Hàm / Lệnh	Chức năng
load_dotenv(override=True)	Tải biến môi trường từ file .env, cho phép ghi đè giá trị cũ.
os.getenv('OPENAI_API_KEY')	Lấy giá trị khóa API của OpenAI từ biến môi trường.
os.getenv('ANTHROPIC_API_KEY')	Lấy giá trị khóa API của Anthropic từ biến môi trường.
os.getenv('GOOGLE_API_KEY')	Lấy giá trị khóa API của Google từ biến môi trường.
if openai_api_key:	Kiểm tra xem khóa API của OpenAI có tồn tại không.
print(f"OpenAI API Key exists and begins {openai_api_key[:8]}")	In ra thông báo nếu khóa API của OpenAI tồn tại (hiển thị 8 ký tự đầu).
if anthropic_api_key:	Kiểm tra xem khóa API của Anthropic có tồn tại

	không.
print(f"Anthropic API Key exists and begins {anthropic_api_key[:7]}")	In ra thông báo nếu khóa API của Anthropic tồn tại (hiển thị 7 ký tự đầu).
if google_api_key:	Kiểm tra xem khóa API của Google có tồn tại không.
print(f"Google API Key exists and begins {google_api_key[:8]}")	In ra thông báo nếu khóa API của Google tồn tại (hiển thị 8 ký tự đầu).
print(google_api_key)	In toàn bộ khóa API của Google (có thể gây rủi ro bảo mật).

```
openai = OpenAI()

claude = anthropic.Anthropic()

google.generativeai.configure()

system_message = "You are a helpful assistant"
```

Hàm / Lệnh	Chức năng
openai = OpenAI()	Khởi tạo đối tượng OpenAI để sử dụng API của OpenAI.
claude = anthropic.Anthropic()	Khởi tạo đối tượng Anthropic để sử dụng API của Claude.
google.generativeai.configure()	Cấu hình API của Google Generative AI.
system_message = "You are a helpful assistant"	Thiết lập thông điệp hệ thống để hướng dẫn trợ lý AI.

```
openai.chat.completions.create(
        model='gpt-4o-mini',
        messages=messages,
    )
    return completion.choices[0].message.content
```

Hàm / Lệnh	Chức năng
openai.chat.completions.create(...)	Gửi yêu cầu đến mô hình gpt-4o-mini để tạo phản hồi dựa trên danh sách tin nhắn messages.
return completion.choices[0].message.content	Trả về nội dung tin nhắn phản hồi đầu tiên từ mô hình AI.

```
def shout(text):
    print(f"Shout has been called with input {text}")
    return text.upper()

shout("hello")
```

Hàm / Lệnh	Chức năng
def shout(text):	Định nghĩa hàm shout nhận một chuỗi text làm tham số.
print(f"Shout has been called with input {text}")	In ra thông báo rằng hàm shout đã được gọi cùng với giá trị đầu vào.

return text.upper()	Chuyển chuỗi text thành chữ in hoa và trả về kết quả.
shout("hello")	Gọi hàm shout với chuỗi "hello", in ra thông báo và trả về "HELLO".

```
gr.Interface(fn=shout, inputs="textbox", outputs="textbox").launch()
gr.Interface(fn=shout, inputs="textbox", outputs="textbox",
flagging_mode="never").launch(share=True)
gr.Interface(fn=shout, inputs="textbox", outputs="textbox",
flagging_mode="never").launch(inbrowser=True)
```

Hàm / Lệnh	Chức năng
gr.Interface(fn=shout, inputs="textbox", outputs="textbox").launch()	Tạo giao diện Gradio với hàm shout, nhập liệu từ ô văn bản và hiển thị kết quả ở ô văn bản. Khởi chạy giao diện cục bộ.
gr.Interface(fn=shout, inputs="textbox", outputs="textbox", flagging_mode="never").launch(share=True)	Tạo giao diện Gradio và vô hiệu hóa chế độ gắn cờ (flagging_mode="never"). Khởi chạy với liên kết chia sẻ công khai.
gr.Interface(fn=shout, inputs="textbox", outputs="textbox", flagging_mode="never").launch(inbrowser=True)	Tạo giao diện Gradio, vô hiệu hóa chế độ gắn cờ và tự động mở trong trình duyệt khi chạy.

```
view = gr.Interface(
    fn=shout,
    inputs=[gr.Textbox(label="Your message:", lines=6)],
    outputs=[gr.Textbox(label="Response:", lines=8)],
    flagging_mode="never"
)
view.launch()

view = gr.Interface(
    fn=message_gpt,
    inputs=[gr.Textbox(label="Your message:", lines=6)],
    outputs=[gr.Textbox(label="Response:", lines=8)],
    flagging_mode="never"
)
view.launch()
```

Hàm / Lệnh	Chức năng
view = gr.Interface(fn=shout, inputs=[gr.Textbox(label="Your message:", lines=6)], outputs=[gr.Textbox(label="Response:", lines=8)], flagging_mode="never")	Tạo giao diện Gradio với hàm shout, gồm một ô nhập và một ô xuất văn bản có nhiều dòng. Vô hiệu hóa chế độ gắn cờ.
view.launch()	Khởi chạy giao diện Gradio đã tạo.
view = gr.Interface(fn=message_gpt, inputs=[gr.Textbox(label="Your message:", lines=6)],	Tạo giao diện Gradio với hàm message_gpt, gồm một ô nhập và một ô xuất văn bản có nhiều dòng. Vô hiệu hóa chế độ gắn cờ.

outputs=[gr.Textbox(label="Response:", lines=8)], flagging_mode="never")	
view.launch()	Khởi chạy giao diện Gradio đã tạo.

```
def stream_gpt(prompt):
    messages = [
        {"role": "system", "content": system_message},
        {"role": "user", "content": prompt}
      ]
    stream = openai.chat.completions.create(
        model='gpt-4o-mini',
        messages=messages,
        stream=True
    )
    result = ""
    for chunk in stream:
        result += chunk.choices[0].delta.content or ""
        yield result

view = gr.Interface(
    fn=stream_gpt,
    inputs=[gr.Textbox(label="Your message:")],
    outputs=[gr.Markdown(label="Response:")],
    flagging_mode="never"
)
view.launch()
```

Hàm / Lệnh	Chức năng
def stream_gpt(prompt)	Hàm xử lý yêu cầu nhập (prompt), gửi đến OpenAI GPT-4o-mini và trả về phản hồi theo luồng.
messages = [{"role": "system", "content": system_message}, {"role": "user", "content": prompt}]	Tạo danh sách tin nhắn gồm tin hệ thống và tin người dùng để gửi đến API OpenAI.
stream = openai.chat.completions.create(model='gpt-4o-mini', messages=messages, stream=True)	Gửi tin nhắn đến API OpenAI để nhận phản hồi theo dạng luồng (stream=True).
for chunk in stream:	Lặp qua từng phần dữ liệu (chunk) của phản hồi từ API.
result += chunk.choices[0].delta.content or ""	Cập nhật nội dung phản hồi từng phần vào biến result.
yield result	Trả về phản hồi theo luồng để hiển thị dần dần.
view = gr.Interface(fn=stream_gpt, inputs=[gr.Textbox(label="Your message:")], outputs=[gr.Markdown(label="Response:")], flagging_mode="never")	Tạo giao diện Gradio với ô nhập văn bản và ô hiển thị kết quả dưới dạng Markdown.
view.launch()	Khởi chạy giao diện Gradio để người dùng nhập tin nhắn và nhận phản hồi từ GPT.

```
def stream_claude(prompt):
    result = claude.messages.stream(
        model="claude-3-haiku-20240307",
        max_tokens=1000,
        temperature=0.7,
        system=system_message,
        messages=[
            {"role": "user", "content": prompt},
        ],
    )
    response = ""
    with result as stream:
        for text in stream.text_stream:
            response += text or ""
            yield response

view = gr.Interface(
    fn=stream_claude,
    inputs=[gr.Textbox(label="Your message:")],
    outputs=[gr.Markdown(label="Response:")],
    flagging_mode="never"
)
view.launch()
```

Hàm / Lệnh	Chức năng
def stream_claude(prompt)	Hàm xử lý yêu cầu nhập (prompt), gửi đến API Claude-3 Haiku và trả về phản hồi theo luồng.
result = claude.messages.stream(...)	Gửi tin nhắn đến API Claude-3 Haiku với cấu hình mô hình, số token tối đa, nhiệt độ và tin hệ thống.
with result as stream:	Mở luồng dữ liệu nhận phản hồi từ Claude.
for text in stream.text_stream:	Lặp qua từng phần dữ liệu phản hồi từ Claude.
response += text or ""	Cập nhật nội dung phản hồi vào biến response.
yield response	Trả về phản hồi theo luồng để hiển thị dần dần.
view = gr.Interface(...)	Tạo giao diện Gradio với ô nhập văn bản và ô hiển thị kết quả dưới dạng Markdown.
view.launch()	Khởi chạy giao diện Gradio để người dùng nhập tin nhắn và nhận phản hồi từ Claude.

```
def stream_model(prompt, model):
    if model=="GPT":
        result = stream_gpt(prompt)
    elif model=="Claude":
        result = stream_claude(prompt)
    else:
        raise ValueError("Unknown model")
    yield from result

view = gr.Interface(
    fn=stream_model,
    inputs=[gr.Textbox(label="Your message:"), gr.Dropdown(["GPT",
"Claude"], label="Select model", value="GPT")],
    outputs=[gr.Markdown(label="Response:")],
    flagging_mode="never"
)
view.launch()
```

Hàm / Lệnh	Chức năng
def stream_model(prompt, model)	Hàm chọn mô hình AI (GPT hoặc Claude) và trả về phản hồi theo luồng.
if model=="GPT": result = stream_gpt(prompt)	Nếu chọn GPT, gọi hàm stream_gpt để xử lý yêu cầu.
elif model=="Claude": result = stream_claude(prompt)	Nếu chọn Claude, gọi hàm stream_claude để xử lý yêu cầu.
else: raise ValueError("Unknown model")	Ném lỗi nếu mô hình không hợp lệ.
yield from result	Trả về dữ liệu phản hồi theo luồng từ mô hình đã chọn.
view = gr.Interface(...)	Tạo giao diện Gradio với ô nhập văn bản và menu chọn mô hình AI.
view.launch()	Khởi chạy giao diện Gradio để người dùng nhập tin nhắn, chọn mô hình và nhận phản hồi.

```
class Website:
    url: str
    title: str
    text: str

    def __init__(self, url):
        self.url = url
        response = requests.get(url)
        self.body = response.content
        soup = BeautifulSoup(self.body, 'html.parser')
        self.title = soup.title.string if soup.title else "No title
found"
        for irrelevant in soup.body(["script", "style", "img", "input"]):
            irrelevant.decompose()
        self.text = soup.body.get_text(separator="\n", strip=True)

    def get_contents(self):
        return f"Webpage Title:\n{self.title}\nWebpage Contents:\
n{self.text}\n\n"
```

Hàm / Lệnh	Chức năng
class Website	Định nghĩa lớp Website để lấy nội dung từ một trang web.
def __init__(self, url)	Khởi tạo đối tượng Website với URL, tải nội dung trang web và trích xuất tiêu đề, nội dung.
response = requests.get(url)	Gửi yêu cầu HTTP GET để lấy nội dung trang web.
soup = BeautifulSoup(self.body, 'html.parser')	Phân tích cú pháp HTML của trang web bằng BeautifulSoup.
self.title = soup.title.string if soup.title else "No title found"	Lấy tiêu đề trang web, nếu có.
for irrelevant in soup.body(["script", "style", "img", "input"]): irrelevant.decompose()	Loại bỏ các thẻ không cần thiết (script, style, img, input).
self.text = soup.body.get_text(separator="\n", strip=True)	Trích xuất nội dung văn bản từ trang web.
def get_contents(self)	Trả về tiêu đề và nội dung văn bản của trang web dưới dạng chuỗi.

```
system_message = "You are an assistant that analyzes the contents of a
company website landing page \
and creates a short brochure about the company for prospective
customers, investors and recruits. Respond in markdown."
def stream_brochure(company_name, url, model):
    prompt = f"Please generate a company brochure for {company_name}.
Here is their landing page:\n"
    prompt += Website(url).get_contents()
    if model=="GPT":
        result = stream_gpt(prompt)
    elif model=="Claude":
        result = stream_claude(prompt)
    else:
        raise ValueError("Unknown model")
    yield from result
```

Hàm / Lệnh	Chức năng
system_message	Định nghĩa thông điệp hệ thống, chỉ dẫn AI tạo brochure ngắn về công ty dựa trên trang web chính.
def stream_brochure(company_name, url, model)	Tạo luồng dữ liệu sinh brochure từ trang web của công ty.
prompt = f"Please generate a company brochure for {company_name}..."	Xây dựng prompt để yêu cầu AI tạo brochure.
prompt += Website(url).get_contents()	Lấy nội dung trang web của công ty và đưa vào prompt.
if model=="GPT": result = stream_gpt(prompt)	Gọi mô hình GPT nếu được chọn.
elif model=="Claude": result = stream_claude(prompt)	Gọi mô hình Claude nếu được chọn.
else: raise ValueError("Unknown model")	Kiểm tra mô hình hợp lệ, nếu không sẽ báo lỗi.
yield from result	Trả về kết quả từng phần từ mô hình AI theo luồng.

```
view = gr.Interface(
    fn=stream_brochure,
    inputs=[
        gr.Textbox(label="Company name:"),
        gr.Textbox(label="Landing page URL including http:// or
https://"),
        gr.Dropdown(["GPT", "Claude"], label="Select model")],
    outputs=[gr.Markdown(label="Brochure:")],
    flagging_mode="never"
)
view.launch()
```

Hàm / Lệnh	Chức năng
view = gr.Interface(...)	Tạo giao diện Gradio để gọi stream_brochure.
fn=stream_brochure	Xác định hàm xử lý yêu cầu tạo brochure.
inputs=[...]	Định nghĩa các ô nhập liệu gồm tên công ty, URL trang web và lựa chọn mô hình AI.
gr.Textbox(label="Company name:")	Ô nhập tên công ty.
gr.Textbox(label="Landing page	Ô nhập URL trang web công ty.

URL including http:// or https://")	
gr.Dropdown(["GPT", "Claude"], label="Select model")	Dropdown chọn mô hình AI (GPT hoặc Claude).
outputs=[gr.Markdown(label="Brochure:")]	Hiển thị kết quả dưới dạng Markdown.
flagging_mode="never"	Tắt chức năng đánh dấu nội dung.
view.launch()	Khởi chạy giao diện web.