```
import os
import json
from dotenv import load_dotenv
from openai import OpenAI
import gradio as gr
```

Hàm / Lệnh	Chức năng
import os	Import module os để thao tác với biến môi trường.
from dotenv import load_dotenv	Import load_dotenv để tải biến môi trường từ tệp .env.
from openai import OpenAI	Import OpenAI SDK để gọi API OpenAI.
import google.generativeai as genai	Import Google Generative AI SDK để sử dụng mô hình của Google.
import gradio as gr	Import Gradio để tạo giao diện người dùng.

```
load_dotenv(override=True)

openai_api_key = os.getenv('OPENAI_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")

MODEL = "gpt-4o-mini"
openai = OpenAI()
```

Hàm / Lệnh	Chức năng
load_dotenv(override=True)	Tải các biến môi trường từ file .env, ghi đè giá trị cũ nếu có.
openai_api_key = os.getenv('OPENAI_API_KEY')	Lấy giá trị API key của OpenAI từ biến môi trường.
if openai_api_key: print(f"OpenAI API Key exists and begins {openai_api_key[:8]}")	Kiểm tra nếu API key tồn tại, in ra 8 ký tự đầu của key.
else: print("OpenAI API Key not set")	Nếu API key không tồn tại, in thông báo "OpenAI API Key not set".
MODEL = "gpt-4o-mini"	Gán giá trị "gpt-4o-mini" cho biến MODEL để chỉ định mô hình AI sử dụng.
openai = OpenAI()	Khởi tạo đối tượng OpenAI để tương tác với API của OpenAI.

```
system_message = "You are a helpful assistant for an Airline called
FlightAI. "
system_message += "Give short, courteous answers, no more than 1
sentence. "
system_message += "Always be accurate. If you don't know the answer, say
so."
def chat(message, history):
    messages = [{"role": "system", "content": system_message}] + history
+ [{"role": "user", "content": message}]
    response = openai.chat.completions.create(model=MODEL,
messages=messages)
    return response.choices[0].message.content

gr.ChatInterface(fn=chat, type="messages").launch()
```

Hàm / Lệnh	Chức năng
system_message = "You are a	Khởi tạo chuỗi hướng dẫn hệ thống cho trợ lý, chỉ định bối

helpful assistant for an Airline called FlightAI. "	cảnh là hãng hàng không FlightAI.
system_message += "Give short, courteous answers, no more than 1 sentence. "	Thêm hướng dẫn trả lời ngắn gọn, lịch sự và không quá một câu.
system_message += "Always be accurate. If you don't know the answer, say so."	Thêm yêu cầu luôn chính xác; nếu không biết, hãy thừa nhận.
def chat(message, history):	Định nghĩa hàm chat nhận đầu vào là tin nhắn người dùng và lịch sử hội thoại.
messages = [{"role": "system", "content": system_message}] + history + [{"role": "user", "content": message}]	Xây dựng danh sách tin nhắn gửi đến API, bao gồm tin hệ thống, lịch sử hội thoại và tin nhắn người dùng mới.
response = openai.chat.completions.create(model=MODEL, messages=messages)	Gửi yêu cầu tạo phản hồi đến API OpenAI sử dụng mô hình được chỉ định (MODEL) và các tin nhắn đã xây dựng.
return response.choices[0].message.content	Trả về nội dung tin nhắn phản hồi đầu tiên từ kết quả của API.
gr.ChatInterface(fn=chat, type="messages").launch()	Tạo và khởi chạy giao diện chat của Gradio, sử dụng hàm chat để xử lý hội thoại.

```
ticket_prices = {"london": "$799", "paris": "$899", "tokyo": "$1400", "berlin": "$499"}

def get_ticket_price(destination_city):
    print(f"Tool get_ticket_price called for {destination_city}")
    city = destination_city.lower()
    return ticket_prices.get(city, "Unknown")

get_ticket_price("Berlin")
```

Hàm / Lệnh	**Chức năng**
ticket_prices = {"london": "$799", "paris": "$899", "tokyo": "$1400", "berlin": "$499"}	Định nghĩa dictionary chứa giá vé cho các thành phố.
def get_ticket_price(destination_city):	Định nghĩa hàm get_ticket_price nhận tên thành phố đích để trả về giá vé tương ứng.
print(f"Tool get_ticket_price called for {destination_city}")	In ra thông báo rằng hàm đã được gọi với tên thành phố đích.
city = destination_city.lower()	Chuyển đổi tên thành phố đích thành chữ thường để đảm bảo tính nhất quán khi tra cứu trong dictionary.
return ticket_prices.get(city, "Unknown")	Trả về giá vé từ dictionary, hoặc "Unknown" nếu thành phố không có trong danh sách.
get_ticket_price("Berlin")	Gọi hàm get_ticket_price với tham số "Berlin" để lấy giá vé cho Berlin.

```
price_function = {
    "name": "get_ticket_price",
    "description": "Get the price of a return ticket to the destination
city. Call this whenever you need to know the ticket price, for example
when a customer asks 'How much is a ticket to this city'",
    "parameters": {
        "type": "object",
        "properties": {
            "destination_city": {
                "type": "string",
                "description": "The city that the customer wants to
travel to",
            },
        },
        "required": ["destination_city"],
        "additionalProperties": False
    }
}
tools = [{"type": "function", "function": price_function}]
```

Hàm / Lệnh	Chức năng
price_function	Định nghĩa thông tin mô tả về hàm get_ticket_price, bao gồm tên, mô tả, và cấu trúc tham số (chỉ yêu cầu "destination_city" kiểu string) để lấy giá vé khứ hồi đến thành phố đích.
tools = [{"type": "function", "function": price_function}]	Tạo danh sách tools chứa hàm được mô tả, dùng để tích hợp vào hệ thống công cụ hoặc trợ lý khi cần gọi hàm này.

```
def chat(message, history):
    messages = [{"role": "system", "content": system_message}] + history
+ [{"role": "user", "content": message}]
    response = openai.chat.completions.create(model=MODEL,
messages=messages, tools=tools)

    if response.choices[0].finish_reason=="tool_calls":
        message = response.choices[0].message
        response, city = handle_tool_call(message)
        messages.append(message)
        messages.append(response)
        response = openai.chat.completions.create(model=MODEL,
messages=messages)

    return response.choices[0].message.content
```

Hàm / Lệnh	Chức năng
def chat(message, history):	Định nghĩa hàm chat nhận đầu vào là tin nhắn người dùng (message) và lịch sử hội thoại (history).
messages = [{"role": "system", "content": system_message}] + history + [{"role": "user", "content": message}]	Xây dựng danh sách tin nhắn gửi đến API, bao gồm tin nhắn hệ thống, lịch sử hội thoại và tin nhắn mới từ người dùng.
response = openai.chat.completions.create(model=MODEL, messages=messages, tools=tools)	Gửi yêu cầu đến API OpenAI với mô hình được chỉ định (MODEL), các tin nhắn đã xây dựng và tích hợp các công cụ (tools).
if response.choices[0].finish_reason=="tool_calls":	Kiểm tra nếu API đã hoàn thành với lý do gọi công cụ (tool_calls), tức là cần xử lý lời gọi hàm từ công cụ.
message =	Lấy tin nhắn từ phản hồi của API chứa lời gọi công cụ.

response.choices[0].message	
response, city = handle_tool_call(message)	Gọi hàm handle_tool_call để xử lý lời gọi công cụ, trả về phản hồi và giá trị liên quan (ví dụ: tên thành phố).
messages.append(message)	Thêm tin nhắn chứa lời gọi công cụ vào lịch sử hội thoại.
messages.append(response)	Thêm phản hồi từ công cụ (đã xử lý) vào lịch sử hội thoại.
response = openai.chat.completions.create(model=MODEL, messages=messages)	Gửi lại danh sách tin nhắn cập nhật cho API để nhận phản hồi cuối cùng từ mô hình.
return response.choices[0].message.content	Trả về nội dung tin nhắn phản hồi cuối cùng từ API.

```
def handle_tool_call(message):
    tool_call = message.tool_calls[0]
    arguments = json.loads(tool_call.function.arguments)
    city = arguments.get('destination_city')
    price = get_ticket_price(city)
    response = {
        "role": "tool",
        "content": json.dumps({"destination_city": city,"price":
price}),
        "tool_call_id": tool_call.id
    }
    return response, city

gr.ChatInterface(fn=chat, type="messages").launch()
```

Hàm / Lệnh	**Chức năng**
def handle_tool_call(message):	Định nghĩa hàm handle_tool_call để xử lý lời gọi công cụ từ tin nhắn.
tool_call = message.tool_calls[0]	Lấy lời gọi công cụ đầu tiên từ thuộc tính tool_calls của tin nhắn.
arguments = json.loads(tool_call.function.arguments)	Giải mã chuỗi JSON chứa các tham số của lời gọi công cụ thành một đối tượng Python (dictionary).
city = arguments.get('destination_city')	Trích xuất tên thành phố đích từ dictionary các tham số.
price = get_ticket_price(city)	Gọi hàm get_ticket_price để lấy giá vé dựa trên tên thành phố.
response = { "role": "tool", "content": json.dumps({"destination_city": city, "price": price}), "tool_call_id": tool_call.id }	Tạo phản hồi dưới dạng dictionary, chứa kết quả (tên thành phố và giá vé) được đóng gói lại dưới dạng JSON, kèm theo ID của lời gọi công cụ.
return response, city	Trả về phản hồi từ công cụ cùng với tên thành phố.
gr.ChatInterface(fn=chat, type="messages").launch()	Tạo và khởi chạy giao diện chat Gradio, sử dụng hàm chat để xử lý hội thoại và hiển thị kết quả.