```
import base64
import os

import requests
from dotenv import load_dotenv

from swarm_models import OpenAIChat
from swarms.structs import Agent

# Load environment variables
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")

# Define prompts for various tasks
MEAL_PLAN_PROMPT = (
    "Based on the following user preferences: dietary restrictions as"
    " vegetarian, preferred cuisines as Italian and Indian, a total"
    " caloric intake of around 2000 calories per day, and an"
    " exclusion of legumes, create a detailed weekly meal plan."
    " Include a variety of meals for breakfast, lunch, dinner, and"
    " optional snacks."
)
IMAGE_ANALYSIS_PROMPT = (
    "Identify the items in this fridge, including their quantities"
    " and condition."
)
```

```
# Function to encode image to base64
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode("utf-8")


# Initialize Language Model (LLM)
llm = OpenAIChat(
    openai_api_key=openai_api_key,
    max_tokens=3000,
)


# Function to handle vision tasks
def create_vision_agent(image_path):
    base64_image = encode_image(image_path)
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {openai_api_key}",
    }
    payload = {
        "model": "gpt-4-vision-preview",
        "messages": [
            {
```

```
            "role": "user",
            "content": [
                {"type": "text", "text": IMAGE_ANALYSIS_PROMPT},
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"{base64_image}"
                    },
                },
            ],
        }
    ],
    "max_tokens": 300,
}
response = requests.post(
    "https://api.openai.com/v1/chat/completions",
    headers=headers,
    json=payload,
)
return response.json()


# Function to generate an integrated shopping list considering meal plan and fridge contents
def generate_integrated_shopping_list(
    meal_plan_output, image_analysis, user_preferences
):
```

```
    # Prepare the prompt for the LLM
    fridge_contents = image_analysis["choices"][0]["message"][
        "content"
    ]
    prompt = (
        f"Based on this meal plan: {meal_plan_output}, and the"
        f" following items in the fridge: {fridge_contents},"
        " considering dietary preferences as vegetarian with a"
        " preference for Italian and Indian cuisines, generate a"
        " comprehensive shopping list that includes only the items"
        " needed."
    )

    # Send the prompt to the LLM and return the response
    response = llm(prompt)
    return response  # assuming the response is a string


# Define agent for meal planning
meal_plan_agent = Agent(
    llm=llm,
    sop=MEAL_PLAN_PROMPT,
    max_loops=1,
    autosave=True,
    saved_state_path="meal_plan_agent.json",
)
```

```
# User preferences for meal planning
user_preferences = {
    "dietary_restrictions": "vegetarian",
    "preferred_cuisines": ["Italian", "Indian"],
    "caloric_intake": 2000,
    "other notes": "Doesn't eat legumes",
}

# Generate Meal Plan
meal_plan_output = meal_plan_agent.run(
    f"Generate a meal plan: {user_preferences}"
)

# Vision Agent - Analyze an Image
image_analysis_output = create_vision_agent("full_fridge.jpg")

# Generate Integrated Shopping List
integrated_shopping_list = generate_integrated_shopping_list(
    meal_plan_output, image_analysis_output, user_preferences
)

# Print and save the outputs
print("Meal Plan:", meal_plan_output)
print("Integrated Shopping List:", integrated_shopping_list)
```

```
with open("nutrition_output.txt", "w") as file:
    file.write("Meal Plan:\n" + meal_plan_output + "\n\n")
    file.write(
        "Integrated Shopping List:\n"
        + integrated_shopping_list
        + "\n"
    )

print("Outputs have been saved to nutrition_output.txt")
```