```
"""
Todo
- Add more data in RAG for hydroponic based solutions with images and very detailed captions
- Introduce JSON function calling for the diagnoser -> good / bad -> if bad then disease detecter agent
- List of common desases -> if agent picks one of those diseases -> select another of available treatments
- Fix error choice
"""

import os

from dotenv import load_dotenv

from examples.demos.plant_biologist_swarm.prompts import (
    diagnoser_agent,
    disease_detector_agent,
    growth_predictor_agent,
    harvester_agent,
    treatment_recommender_agent,
)
from swarms import Agent
from swarm_models.gpt_o import GPT4VisionAPI

# Load the OpenAI API key from the .env file
load_dotenv()
```

```
# Initialize the OpenAI API key

api_key = os.environ.get("OPENAI_API_KEY")




# llm = llm,

llm = GPT4VisionAPI(

    max_tokens=3000, openai_api_key=os.getenv("OPENAI_API_KEY")

)


# Initialize Diagnoser Agent

diagnoser_agent = Agent(

    agent_name="Diagnoser Agent",

    system_prompt=diagnoser_agent(),

    llm=llm,

    max_loops=1,

    dashboard=False,

    # streaming_on=True,

    # verbose=True,

    # saved_state_path="diagnoser.json",

    multi_modal=True,

    autosave=True,

    streaming_on=True,

)


# Initialize Harvester Agent
```

```
harvester_agent = Agent(
    agent_name="Harvester Agent",
    system_prompt=harvester_agent(),
    llm=llm,
    max_loops=1,
    dashboard=False,
    # streaming_on=True,
    # verbose=True,
    # saved_state_path="harvester.json",
    multi_modal=True,
    autosave=True,
    streaming_on=True,
)


# Initialize Growth Predictor Agent
growth_predictor_agent = Agent(
    agent_name="Growth Predictor Agent",
    system_prompt=growth_predictor_agent(),
    llm=llm,
    max_loops=1,
    dashboard=False,
    # streaming_on=True,
    # verbose=True,
    # saved_state_path="growth_predictor.json",
    multi_modal=True,
    autosave=True,
    streaming_on=True,
)


# Initialize Treatment Recommender Agent
treatment_recommender_agent = Agent(
    agent_name="Treatment Recommender Agent",
    system_prompt=treatment_recommender_agent(),
    llm=llm,
    max_loops=1,
    dashboard=False,
    # streaming_on=True,
    # verbose=True,
    # saved_state_path="treatment_recommender.json",
    multi_modal=True,
    autosave=True,
    streaming_on=True,
)


# Initialize Disease Detector Agent
disease_detector_agent = Agent(
    agent_name="Disease Detector Agent",
    system_prompt=disease_detector_agent(),
    llm=llm,
    max_loops=1,
    dashboard=False,
    # streaming_on=True,
    # verbose=True,
    # saved_state_path="disease_detector.json",
    multi_modal=True,
    autosave=True,
    streaming_on=True,
)
agents = [
    diagnoser_agent,
    disease_detector_agent,
    treatment_recommender_agent,
    growth_predictor_agent,
    harvester_agent,
]


task = "Conduct a diagnosis on the plants's symptoms, this wasn't grown in dirt, it grew from hydroponics"
img = "bad_tomato.jpg"


loop = 0
for i in range(len(agents)):
    if i == 0:
        output = agents[i].run(task, img)
        print(output)


    else:
        output = agents[i].run(output, img)
        print(output)


    # Add extensive logging for each agent
    print(f"Agent {i+1} - {agents[i].agent_name}")
    print("--------------------------------")
```