Lesson 3 Demo 5

AutoGen Agent Collaboration for Smarter Troubleshooting

Objective: To demonstrate how AutoGen agent collaboration improves IT support by enabling specialized agents to analyze, diagnose, and resolve user issues efficiently

You are building an AI-driven IT support system where multiple agents work together to troubleshoot technical issues. When a user reports a problem, a Diagnostic Agent first gathers relevant details and identifies possible causes. The issue is then passed to a Resolution Agent, which provides a structured, step-by-step solution. By leveraging agent-to-agent communication, this system enhances troubleshooting accuracy, reduces resolution time, and delivers a seamless support experience.

Prerequisites:

1. Create a virtual environment
2. Install dependencies

Steps to be followed:

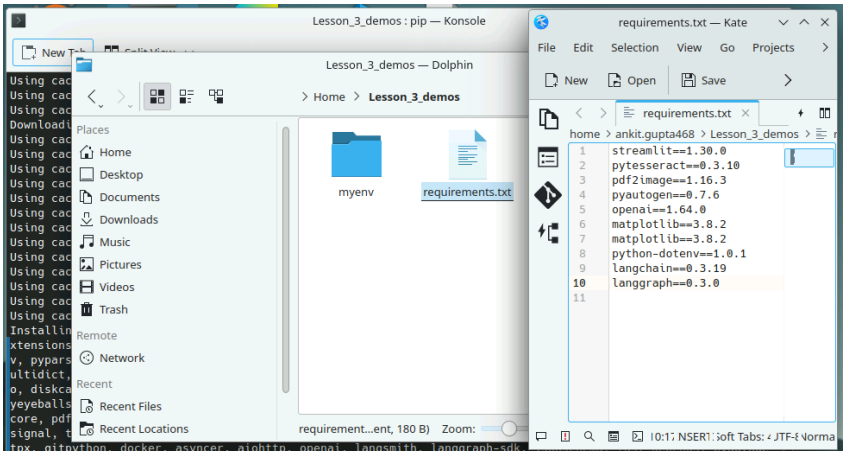
1. Set up the environment
2. Define the Diagnostic Agent
3. Define the Resolution Agent
4. Set up the Streamlit interface for user input
5. Capture user input and run the code

Step 1: Set up the environment

1.1 Open command prompt and go to the "**Lesson_3_demos**" folder (which we created in Demo_1) using the command given below:
mkdir Lesson_3_demos (not needed if the folder is already created in Demo1)
cd Lesson_3_demos

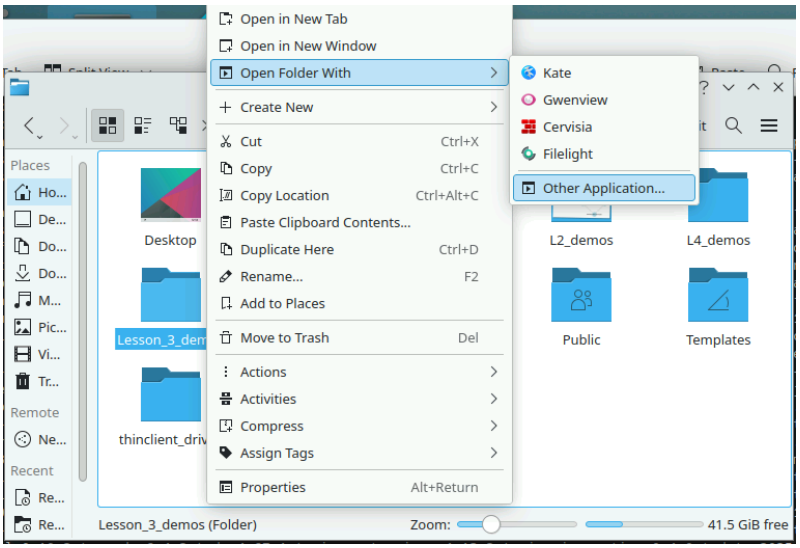
1.2 After this, activate the virtual environment using the command below:
python3 -m venv venv (not needed if the virtual env. is already created in Demo1)
source venv/bin/activate

1.3 Now, create a `requirements.txt` file inside the folder with required libraries (not needed if already done in Demo1):



1.4 Install all the required libraries using the command below:
pip install -r requirements.txt (not needed if already done in Demo1)

1.5 Now, open the folder using VS Code editor:



After this, open a new Python file using the "**New File**" option and name it as "**Demo5**".

Step 1: Define the Diagnostic Agent

1.1 The **Diagnostic Agent** receives user input describing an IT issue and analyzes it to determine the potential causes.

1.2 This agent uses OpenAI to generate a structured diagnosis by identifying likely problems and returning a list of possible causes based on the user's description.

```
import streamlit as st

import autogen

import openai

from dotenv import load_dotenv

import os




client = openai.AzureOpenAI(


api_key="2ABecnfxzhRg4M5D6pBKiqxXVhmGB2WvQ0aYKkbTCPsj0JLKsZPfJQQJ99BDAC77bzfXJ3w3AAABACOGi
3sC",

    api_version="2023-12-01-preview",

    # azure_endpoint="https://openai-api-management-gw.azure-api.net/"

    azure_endpoint="https://openai-api-management-gw.azure-api.net/deployments/gpt-4o-
mini/chat/completions?api-version=2023-12-01-preview"

)
```

Step 2: Define the Resolution Agent

2.1 The **Resolution Agent** takes the diagnosis provided by the Diagnostic Agent and generates a set of troubleshooting steps.

2.2 The Resolution Agent's job is to suggest a detailed, step-by-step solution based on the identified problem. It collaborates with the Diagnostic Agent by using the analysis to recommend solutions that resolve the issue.

```
# Step 1: Define the Diagnostic Agent
class DiagnosticAgent(autogen.AssistantAgent):
    def __init__(self, name="DiagnosticAgent", model="gpt-4o-mini"):
        super().__init__(name=name)
        self.model = model

    def diagnose_issue(self, message):
        """
        Step 2: The Diagnostic Agent analyzes the issue and determines possible causes.
        """
        prompt = f"""
        The user has reported an IT issue: "{message}".
        As the Diagnostic Agent, analyze the problem and list possible causes.
        """
        response = client.chat.completions.create(
            model=self.model,
            messages=[
                {"role": "system", "content": "You are an IT diagnostic expert."},
                {"role": "user", "content": prompt}
            ]
        )
        return response.choices[0].message.content.strip()
```

```
# Step 3: Define the Resolution Agent
class ResolutionAgent(autogen.AssistantAgent):
    def __init__(self, name="ResolutionAgent", model="gpt-4o-mini"):
        super().__init__(name=name)
        self.model = model

    def provide_solution(self, diagnosis):
        """
        Step 4: The Resolution Agent suggests a fix based on the diagnosis from the
Diagnostic Agent.
        """
        prompt = f"""
        The Diagnostic Agent has identified the following possible causes:
        "{diagnosis}".
        As the Resolution Agent, suggest step-by-step troubleshooting solutions.
        """
        response = client.chat.completions.create(
            model=self.model,
            messages=[
                {"role": "system", "content": "You are an IT troubleshooting expert."},
                {"role": "user", "content": prompt}
            ]
        )
        return response.choices[0].message.content.strip()
```

Step 3: Set up the Streamlit interface for user input

3.1 Create the Streamlit app interface for users to input their IT problems.

3.2 The interface includes a text area where users can describe the issue and a button to trigger the support process.

3.3 Upon receiving the issue, the system interacts with both the Diagnostic Agent and the Resolution Agent to handle the troubleshooting process.

```
# Step 5: Deploy via Streamlit

st.title("AutoGen IT Support Chatbot - Agent Collaboration")

st.write("An AI-powered chatbot where agents collaborate to diagnose and resolve IT
issues.")


# User Input

user_input = st.text_area("Describe your IT issue:")

if st.button("Get Support"):

    if user_input.strip():

        diagnostic_agent = DiagnosticAgent()

        resolution_agent = ResolutionAgent()


        diagnosis = diagnostic_agent.diagnose_issue(user_input)

        solution = resolution_agent.provide_solution(diagnosis)


        st.subheader("Diagnosis:")

        st.write(diagnosis)


        st.subheader("Suggested Solution:")

        st.write(solution)

    else:

        st.warning("Please enter a valid IT issue.")
```

Step 4: Capture user input and run the code

4.1 When the **"Get Support"** button is clicked, the app captures the user's description of the issue, processes it using both agents, and displays the results.

4.2 First, the Diagnostic Agent diagnoses the issue, listing potential causes.Then, the Resolution Agent suggests a detailed solution to the problem based on the diagnosis.

4.3 **User feedback**: After generating the diagnosis and solution, the chatbot provides the user with an opportunity to review and confirm the suggested resolution.

4.4 Save the file and then run the streamlit webapp from command prompt using the command given below:

streamlit run Demo5.py

Output:

By following the above-mentioned steps, you have successfully illustrated how AutoGen agents collaborate to improve chatbot responses. By splitting responsibilities between a diagnostic agent and a resolution agent, you achieve a more structured and effective troubleshooting approach. This method enhances problem-solving efficiency in helpdesk and customer support applications.