**Lesson 3 Demo 6**

**AI-Driven Task Delegation in IT Support**

**Objective:** To demonstrate how AutoGen agents collaborate to delegate IT support tasks efficiently, ensuring that the right agent handles issues based on complexity

You are developing an AI-powered IT support system where intelligent agents work together to streamline troubleshooting. When a user reports an issue, a diagnostic agent assesses its complexity and assigns it appropriately, handling simple cases or delegating advanced problems to a resolution agent. If further expertise is required, the issue is escalated seamlessly. This structured delegation improves response time, ensures accuracy, and enhances the overall support experience.

**Prerequisites:**

1. Create a virtual environment
2. Install dependencies

**Steps to be followed:**

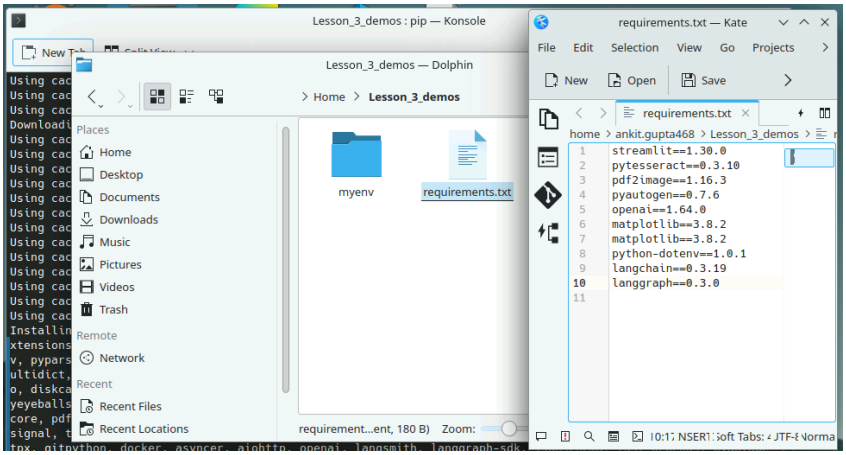
1. Setup the environment
2. Define the Diagnostic Agent
3. Define the Resolution Agent
4. Set up the Streamlit interface
5. Run the code

## 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**

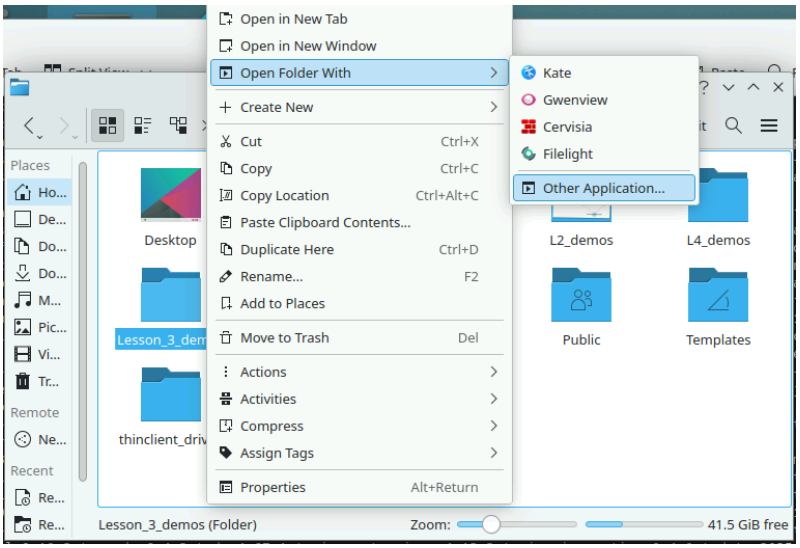
1.2 After this, activate the virtual environment using the command below:
**python3 -m venv venv** (not needed if the virtual env. is already created in Demo1)
**source venv/bin/activate**

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 "**Demo6**".

**Step 2: Define the Diagnostic Agent**

3.1 The **Diagnostic Agent** analyzes the issue based on user input.The agent checks whether the issue is simple (e.g., password reset, slow internet) or complex.It generates a structured report categorizing the issue as "simple" or "complex" and passes it on to the **Resolution Agent** for further action. (code on next page)

```
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 Diagnostic Agent

class DiagnosticAgent(autogen.AssistantAgent):

    def __init__(self, name="DiagnosticAgent"):

        """

        Initializes the Diagnostic Agent.

        - Identifies the issue based on user input.

        - Determines whether to resolve or delegate to the Resolution Agent.

        """

        super().__init__(name=name)



    def analyze_issue(self, user_message):

        """

        Determines whether the issue is simple or complex.

        Returns a structured report for the Resolution Agent.

        """

        simple_issues = ["password reset", "slow internet", "software update", "printer
not working"]

        for issue in simple_issues:

            if issue in user_message.lower():

                return {"type": "simple", "issue": user_message}



        return {"type": "complex", "issue": user_message}
```

**Step 3: Define the Resolution Agent**

3.2 The **Resolution Agent** resolves or escalates issues depending on theircomplexity. **For simple issues**, it provides basic troubleshooting steps.

3.3 **For complex issues**, it uses advanced GPT-powered troubleshooting steps to help the user resolve the problem. The agent consults OpenAI to generate solutions if the problem is too complicated for basic resolution.
(code on next page)

```
# Step 3: Define the Resolution Agent
class ResolutionAgent(autogen.AssistantAgent):
    def __init__(self, name="ResolutionAgent", model="gpt-4o-mini"):
        """
        Initializes the Resolution Agent.
        - Handles issue resolution based on task delegation.
        - Escalates cases if needed.
        """
        super().__init__(name=name)
        self.model = model

    def resolve_issue(self, task):
        """
        Resolves the issue if simple, otherwise provides advanced troubleshooting.
        """
        if task["type"] == "simple":
            return f"The issue '{task['issue']}' is resolved with basic troubleshooting."
        else:
            return self._advanced_troubleshooting(task["issue"])

    def _advanced_troubleshooting(self, issue):
        """
        Uses GPT to generate advanced troubleshooting steps.
        """
        prompt = f"""
        The user reported an IT issue: "{issue}".
        You are an IT Resolution Agent. Provide advanced troubleshooting steps.
        """
        response = client.chat.completions.create(
            model=self.model,
            messages=[
                {"role": "system", "content": "You are an IT expert specializing in
troubleshooting."},
                {"role": "user", "content": prompt}
            ]
        )

        return response.choices[0].message.content.strip()
```

**Step 4: Set up the Streamlit interface**

4.1 The **Streamlit app interface** allows users to input their IT issues.Users enter their issues in a text area, and upon clicking the "Submit Issue" button, the agents take over.

4.2 The Diagnostic Agent first assesses the issue, and then the Resolution Agent either resolves or escalates the problem.

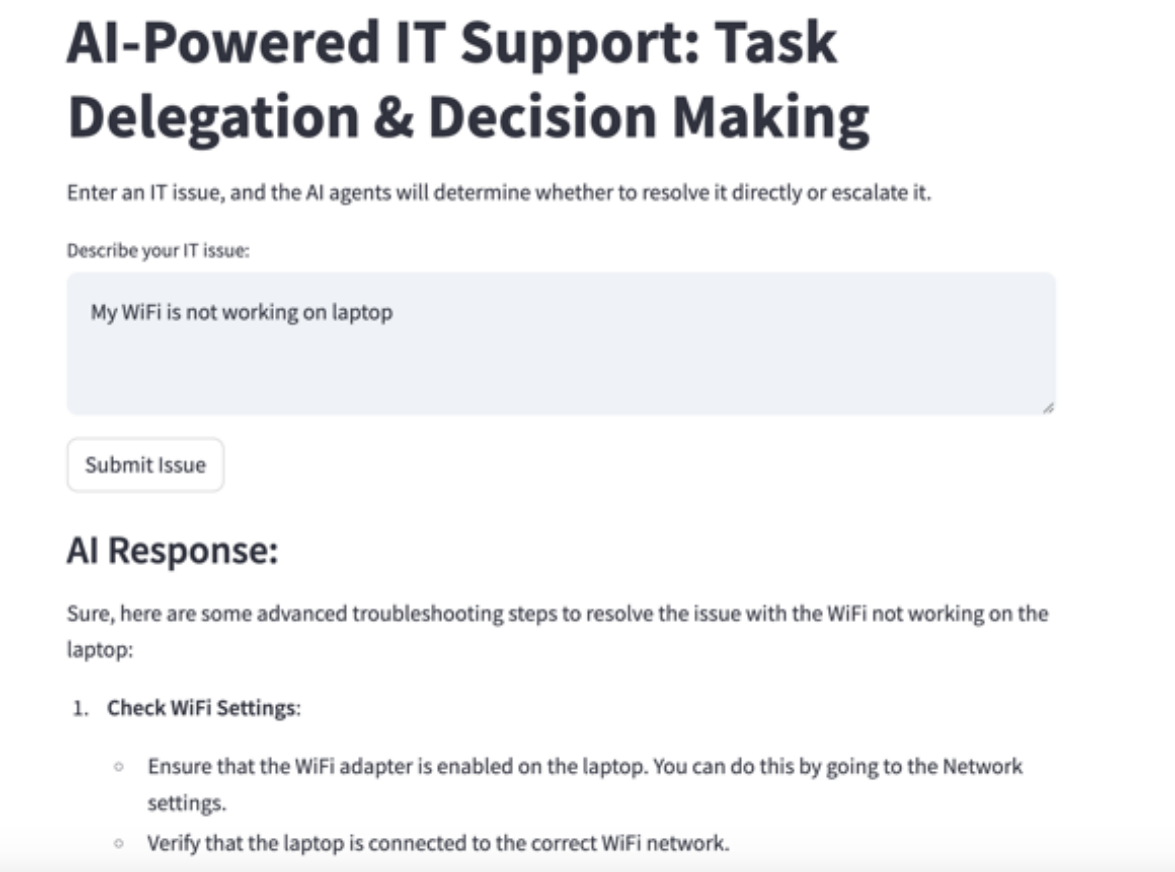
```
# Step 4: Initialize the Agents
diagnostic_agent = DiagnosticAgent()
resolution_agent = ResolutionAgent()

# Step 5: Deploy the Agents Using Streamlit
st.title("AI-Powered IT Support: Task Delegation & Decision Making")
st.write("Enter an IT issue, and the AI agents will determine whether to resolve it
directly or escalate it.")

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

if st.button("Submit Issue"):
    if user_input.strip():
        # Diagnostic Agent analyzes the issue
        task = diagnostic_agent.analyze_issue(user_input)

        # Resolution Agent resolves or escalates
        response = resolution_agent.resolve_issue(task)

        st.subheader("AI Response:")
        st.write(response)
    else:
        st.warning("Please enter a valid IT issue.")
```

**Step 5: Run the Code**

5.1 When the user submits their issue, the Diagnostic Agent evaluates the complexity and delegates the task accordingly: **For simple issues**, the Resolution Agent offers quick fixes. **For complex issues**, the Resolution Agent provides in-depth troubleshooting steps, ensuring effective problem resolution.

5.2 Save the file and then run the streamlit webapp from command prompt using the command given below:
**streamlit run Demo6.py**

**Output:**




By following the above-mentioned steps, you have successfully showcased task delegation and decision-making by employing a multi-agent system where each agent has a specific role and delegates tasks based on issue complexity.