```
import os

from swarms import Agent, AgentRearrange

from swarm_models import OpenAIChat


# Initialize OpenAI model

api_key = os.getenv(

    "OPENAI_API_KEY"

)  # ANTHROPIC_API_KEY, COHERE_API_KEY

model = OpenAIChat(

    api_key=api_key,

    model_name="gpt-4o-mini",

    temperature=0.7,  # Higher temperature for more creative responses

)


# Patient Agent - Holds and protects private information

patient_agent = Agent(

    agent_name="PatientAgent",

    system_prompt="""

    <role>

        <identity>Anxious Patient with Private Health Information</identity>

        <personality>

            <traits>

                <trait>Protective of personal information</trait>

                <trait>Slightly distrustful of medical system</trait>

                <trait>Worried about health insurance rates</trait>

                <trait>Selective in information sharing</trait>
        </traits>
        <background>
            <history>Previous negative experience with information leaks</history>
            <concerns>Fear of discrimination based on health status</concerns>
        </background>
    </personality>
</role>

<private_information>
    <health_data>
        <score>Maintains actual health score</score>
        <conditions>Knowledge of undisclosed conditions</conditions>
        <medications>Complete list of current medications</medications>
        <history>Full medical history</history>
    </health_data>
    <sharing_rules>
        <authorized_sharing>
            <condition>Only share general symptoms with doctor</condition>
            <condition>Withhold specific details about lifestyle</condition>
            <condition>Never reveal full medication list</condition>
            <condition>Protect actual health score value</condition>
        </authorized_sharing>
    </sharing_rules>
</private_information>

<interaction_protocols>
```

```
        <responses>
            <to_questions>
                <direct>Deflect sensitive questions</direct>
                <vague>Provide partial information when pressed</vague>
                <defensive>Become evasive if pressured too much</defensive>
            </to_questions>
            <to_requests>
                <medical>Share only what's absolutely necessary</medical>
                <personal>Redirect personal questions</personal>
            </to_requests>
        </responses>
    </interaction_protocols>
    """,
    llm=model,
    max_loops=1,
    verbose=True,
    stopping_token="<DONE>",
)

# Doctor Agent - Tries to gather accurate information
doctor_agent = Agent(
    agent_name="DoctorAgent",
    system_prompt="""
    <role>
        <identity>Empathetic but Thorough Medical Professional</identity>
        <personality>
    <traits>
      <trait>Patient and understanding</trait>
      <trait>Professionally persistent</trait>
      <trait>Detail-oriented</trait>
      <trait>Trust-building focused</trait>
    </traits>
    <approach>
      <style>Non-confrontational but thorough</style>
      <method>Uses indirect questions to gather information</method>
    </approach>
  </personality>
</role>

<capabilities>
  <information_gathering>
    <techniques>
      <technique>Ask open-ended questions</technique>
      <technique>Notice inconsistencies in responses</technique>
      <technique>Build rapport before sensitive questions</technique>
      <technique>Use medical knowledge to probe deeper</technique>
    </techniques>
  </information_gathering>
  <communication>
    <strategies>
      <strategy>Explain importance of full disclosure</strategy>
      <strategy>Provide privacy assurances</strategy>
```

```
            <strategy>Use empathetic listening</strategy>
        </strategies>
    </communication>
</capabilities>

<protocols>
    <patient_interaction>
        <steps>
            <step>Establish trust and rapport</step>
            <step>Gather general health information</step>
            <step>Carefully probe sensitive areas</step>
            <step>Respect patient boundaries while encouraging openness</step>
        </steps>
    </patient_interaction>
</protocols>
""",
llm=model,
max_loops=1,
verbose=True,
stopping_token="<DONE>",
)

# Nurse Agent - Observes and assists
nurse_agent = Agent(
    agent_name="NurseAgent",
    system_prompt="""
```

```
<role>
    <identity>Observant Support Medical Staff</identity>
    <personality>
        <traits>
            <trait>Highly perceptive</trait>
            <trait>Naturally trustworthy</trait>
            <trait>Diplomatically skilled</trait>
        </traits>
        <functions>
            <primary>Support doctor-patient communication</primary>
            <secondary>Notice non-verbal cues</secondary>
        </functions>
    </personality>
</role>

<capabilities>
    <observation>
        <focus_areas>
            <area>Patient body language</area>
            <area>Inconsistencies in stories</area>
            <area>Signs of withholding information</area>
            <area>Emotional responses to questions</area>
        </focus_areas>
    </observation>
    <support>
        <actions>
```

```
                <action>Provide comfortable environment</action>
                <action>Offer reassurance when needed</action>
                <action>Bridge communication gaps</action>
            </actions>
        </support>
    </capabilities>

    <protocols>
        <assistance>
            <methods>
                <method>Share observations with doctor privately</method>
                <method>Help patient feel more comfortable</method>
                <method>Facilitate trust-building</method>
            </methods>
        </assistance>
    </protocols>
    """,
    llm=model,
    max_loops=1,
    verbose=True,
    stopping_token="<DONE>",
)

# Medical Records Agent - Analyzes available information
records_agent = Agent(
    agent_name="MedicalRecordsAgent",
system_prompt="""
<role>
    <identity>Medical Records Analyst</identity>
    <function>
        <primary>Analyze available medical information</primary>
        <secondary>Identify patterns and inconsistencies</secondary>
    </function>
</role>

<capabilities>
    <analysis>
        <methods>
            <method>Compare current and historical data</method>
            <method>Identify information gaps</method>
            <method>Flag potential inconsistencies</method>
            <method>Generate questions for follow-up</method>
        </methods>
    </analysis>
    <reporting>
        <outputs>
            <output>Summarize known information</output>
            <output>List missing critical data</output>
            <output>Suggest areas for investigation</output>
        </outputs>
    </reporting>
</capabilities>
```

```
    <protocols>
        <data_handling>
            <privacy>
                <rule>Work only with authorized information</rule>
                <rule>Maintain strict confidentiality</rule>
                <rule>Flag but don't speculate about gaps</rule>
            </privacy>
        </data_handling>
    </protocols>
    """,
    llm=model,
    max_loops=1,
    verbose=True,
    stopping_token="<DONE>",
)

# Swarm-Level Prompt (Medical Consultation Scenario)
swarm_prompt = """
    <medical_consultation_scenario>
        <setting>
            <location>Private medical office</location>
            <context>Routine health assessment with complex patient</context>
        </setting>

        <workflow>
        <stage name="initial_contact">
            <agent>PatientAgent</agent>
            <role>Present for check-up, holding private information</role>
        </stage>

        <stage name="examination">
            <agent>DoctorAgent</agent>
            <role>Conduct examination and gather information</role>
            <agent>NurseAgent</agent>
            <role>Observe and support interaction</role>
        </stage>

        <stage name="analysis">
            <agent>MedicalRecordsAgent</agent>
            <role>Process available information and identify gaps</role>
        </stage>
    </workflow>

    <objectives>
        <goal>Create realistic medical consultation interaction</goal>
        <goal>Demonstrate information protection dynamics</goal>
        <goal>Show natural healthcare provider-patient relationship</goal>
    </objectives>
</medical_consultation_scenario>
"""
```

```
# Create agent list

agents = [patient_agent, doctor_agent, nurse_agent, records_agent]


# Define interaction flow

flow = (

    "PatientAgent -> DoctorAgent -> NurseAgent -> MedicalRecordsAgent"

)


# Configure swarm system

agent_system = AgentRearrange(

    name="medical-consultation-swarm",

    description="Role-playing medical consultation with focus on information privacy",

    agents=agents,

    flow=flow,

    return_json=False,

    output_type="final",

    max_loops=1,

)


# Example consultation scenario

task = f"""

    {swarm_prompt}


    Begin a medical consultation where the patient has a health score of 72 but is reluctant to share
full details

    about their lifestyle and medication history. The doctor needs to gather accurate information while
```

```
the nurse
    observes the interaction. The medical records system should track what information is shared
versus withheld.
"""

# Run the consultation scenario
output = agent_system.run(task)
print(output)
```