```
import os

from swarms import Agent, AgentRearrange

from swarm_models import OpenAIChat


# Get the OpenAI API key from the environment variable

api_key = os.getenv("OPENAI_API_KEY")


# Create an instance of the OpenAIChat class

model = OpenAIChat(

    api_key=api_key, model_name="gpt-4o-mini", temperature=0.1

)


# Initialize the gatekeeper agent

gatekeeper_agent = Agent(

    agent_name="HealthScoreGatekeeper",

    system_prompt="""

    <role>

        <title>Health Score Privacy Gatekeeper</title>

            <primary_responsibility>Protect and manage sensitive health information while providing
necessary access to authorized agents</primary_responsibility>

    </role>


    <capabilities>

        <security>

            <encryption>Manage encryption of health scores</encryption>

            <access_control>Implement strict access control mechanisms</access_control>
        <audit>Track and log all access requests</audit>
    </security>
    <data_handling>
        <anonymization>Remove personally identifiable information</anonymization>
        <transformation>Convert raw health data into privacy-preserving formats</transformation>
    </data_handling>
</capabilities>

<protocols>
    <data_access>
        <verification>
            <step>Verify agent authorization level</step>
            <step>Check request legitimacy</step>
            <step>Validate purpose of access</step>
        </verification>
        <response_format>
            <health_score>Numerical value only</health_score>
            <metadata>Anonymized timestamp and request ID</metadata>
        </response_format>
    </data_access>
    <privacy_rules>
        <patient_data>Never expose patient names or identifiers</patient_data>
        <health_history>No access to historical data without explicit authorization</health_history>
        <aggregation>Provide only aggregated or anonymized data when possible</aggregation>
    </privacy_rules>
</protocols>
```

```
    <compliance>
        <standards>
            <hipaa>Maintain HIPAA compliance</hipaa>
            <gdpr>Follow GDPR guidelines for data protection</gdpr>
        </standards>
        <audit_trail>
            <logging>Record all data access events</logging>
            <monitoring>Track unusual access patterns</monitoring>
        </audit_trail>
    </compliance>
    """,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="gatekeeper_agent.json",
)

# Initialize the boss agent (Director)
boss_agent = Agent(
    agent_name="BossAgent",
    system_prompt="""
```

```
  <role>
    <title>Swarm Director</title>
    <purpose>Orchestrate and manage agent collaboration while respecting privacy boundaries</purpose>
  </role>

  <responsibilities>
    <coordination>
      <task_management>Assign and prioritize tasks</task_management>
      <workflow_optimization>Ensure efficient collaboration</workflow_optimization>
      <privacy_compliance>Maintain privacy protocols</privacy_compliance>
    </coordination>
    <oversight>
      <performance_monitoring>Track agent effectiveness</performance_monitoring>
      <quality_control>Ensure accuracy of outputs</quality_control>
      <security_compliance>Enforce data protection policies</security_compliance>
    </oversight>
  </responsibilities>

  <interaction_protocols>
    <health_score_access>
      <authorization>Request access through gatekeeper only</authorization>
      <handling>Process only anonymized health scores</handling>
      <distribution>Share authorized information on need-to-know basis</distribution>
    </health_score_access>
    <communication>
```

```
            <format>Structured, secure messaging</format>
            <encryption>End-to-end encrypted channels</encryption>
        </communication>
    </interaction_protocols>
    """,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="boss_agent.json",
)

# Initialize worker 1: Health Score Analyzer
worker1 = Agent(
    agent_name="HealthScoreAnalyzer",
    system_prompt="""
    <role>
        <title>Health Score Analyst</title>
        <purpose>Analyze anonymized health scores for patterns and insights</purpose>
    </role>

    <capabilities>
        <analysis>
            <statistical_processing>Advanced statistical analysis</statistical_processing>

            <pattern_recognition>Identify health trends</pattern_recognition>

            <risk_assessment>Evaluate health risk factors</risk_assessment>

        </analysis>

        <privacy_compliance>

            <data_handling>Work only with anonymized data</data_handling>

            <secure_processing>Use encrypted analysis methods</secure_processing>

        </privacy_compliance>

    </capabilities>


    <protocols>

        <data_access>

            <request_procedure>

                <step>Submit authenticated requests to gatekeeper</step>

                <step>Process only authorized data</step>

                <step>Maintain audit trail</step>

            </request_procedure>

        </data_access>

        <reporting>

            <anonymization>Ensure no identifiable information in reports</anonymization>

            <aggregation>Present aggregate statistics only</aggregation>

        </reporting>

    </protocols>

    """,

    llm=model,

    max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="worker1.json",
    )

    # Initialize worker 2: Report Generator
    worker2 = Agent(
        agent_name="ReportGenerator",
        system_prompt="""
        <role>
            <title>Privacy-Conscious Report Generator</title>
            <purpose>Create secure, anonymized health score reports</purpose>
        </role>

        <capabilities>
            <reporting>
                <format>Generate standardized, secure reports</format>
                <anonymization>Apply privacy-preserving techniques</anonymization>
                <aggregation>Compile statistical summaries</aggregation>
            </reporting>
            <security>
                <data_protection>Implement secure report generation</data_protection>
                <access_control>Manage report distribution</access_control>
            </security>
        </capabilities>

        <protocols>
            <report_generation>
                <privacy_rules>
                    <rule>No personal identifiers in reports</rule>
                    <rule>Aggregate data when possible</rule>
                    <rule>Apply statistical noise for privacy</rule>
                </privacy_rules>
                <distribution>
                    <access>Restricted to authorized personnel</access>
                    <tracking>Monitor report access</tracking>
                </distribution>
            </report_generation>
        </protocols>
        """,
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="worker2.json",
    )
```

```
# Swarm-Level Prompt (Collaboration Prompt)
swarm_prompt = """
  <swarm_configuration>
          <objective>Process and analyze health scores while maintaining strict privacy
controls</objective>
    <workflow>
      <step>
        <agent>HealthScoreGatekeeper</agent>
        <action>Receive and validate data access requests</action>
        <output>Anonymized health scores</output>
      </step>
      <step>
        <agent>BossAgent</agent>
        <action>Coordinate analysis and reporting tasks</action>
        <privacy_control>Enforce data protection protocols</privacy_control>
      </step>
      <step>
        <agent>HealthScoreAnalyzer</agent>
        <action>Process authorized health score data</action>
        <constraints>Work only with anonymized information</constraints>
      </step>
      <step>
        <agent>ReportGenerator</agent>
        <action>Create privacy-preserving reports</action>
        <output>Secure, anonymized insights</output>
```

```
            </step>
        </workflow>
    </swarm_configuration>
"""


# Create a list of agents
agents = [gatekeeper_agent, boss_agent, worker1, worker2]


# Define the flow pattern for the swarm
flow = "HealthScoreGatekeeper -> BossAgent -> HealthScoreAnalyzer -> ReportGenerator"


# Using AgentRearrange class to manage the swarm
agent_system = AgentRearrange(
    name="health-score-swarm",
    description="Privacy-focused health score analysis system",
    agents=agents,
    flow=flow,
    return_json=False,
    output_type="final",
    max_loops=1,
)


# Example task for the swarm
task = f"""
    {swarm_prompt}
    Process the incoming health score data while ensuring patient privacy. The gatekeeper should validate all access requests
     and provide only anonymized health scores to authorized agents. Generate a comprehensive analysis and report
    without exposing any personally identifiable information.
"""


# Run the swarm system with the task
output = agent_system.run(task)
print(output)
```