```
import asyncio
from typing import List


from swarm_models import OpenAIChat


from swarms.structs.async_workflow import (
    SpeakerConfig,
    SpeakerRole,
    create_default_workflow,
    run_workflow_with_retry,
)
from swarms.prompts.finance_agent_sys_prompt import (
    FINANCIAL_AGENT_SYS_PROMPT,
)
from swarms.structs.agent import Agent



async def create_specialized_agents() -> List[Agent]:
    """Create a set of specialized agents for financial analysis"""


    # Base model configuration
    model = OpenAIChat(model_name="gpt-4o")


    # Financial Analysis Agent
    financial_agent = Agent(
        agent_name="Financial-Analysis-Agent",
        agent_description="Personal finance advisor agent",
        system_prompt=FINANCIAL_AGENT_SYS_PROMPT
         + "Output the <DONE> token when you're done creating a portfolio of etfs, index, funds, and
more for AI",
        max_loops=1,
        llm=model,
        dynamic_temperature_enabled=True,
        user_name="Kye",
        retry_attempts=3,
        context_length=8192,
        return_step_meta=False,
        output_type="str",
        auto_generate_prompt=False,
        max_tokens=4000,
        stopping_token="<DONE>",
        saved_state_path="financial_agent.json",
        interactive=False,
    )

    # Risk Assessment Agent
    risk_agent = Agent(
        agent_name="Risk-Assessment-Agent",
        agent_description="Investment risk analysis specialist",
          system_prompt="Analyze investment risks and provide risk scores. Output <DONE> when
analysis is complete.",
        max_loops=1,
        llm=model,
        dynamic_temperature_enabled=True,
        user_name="Kye",
        retry_attempts=3,
        context_length=8192,
        output_type="str",
        max_tokens=4000,
        stopping_token="<DONE>",
        saved_state_path="risk_agent.json",
        interactive=False,
    )

    # Market Research Agent
    research_agent = Agent(
        agent_name="Market-Research-Agent",
        agent_description="AI and tech market research specialist",
         system_prompt="Research AI market trends and growth opportunities. Output <DONE> when
research is complete.",
        max_loops=1,
        llm=model,
        dynamic_temperature_enabled=True,
        user_name="Kye",
        retry_attempts=3,
        context_length=8192,
        output_type="str",
        max_tokens=4000,
        stopping_token="<DONE>",
        saved_state_path="research_agent.json",
        interactive=False,
    )

    return [financial_agent, risk_agent, research_agent]


async def main():
    # Create specialized agents
    agents = await create_specialized_agents()

    # Create workflow with group chat enabled
    workflow = create_default_workflow(
        agents=agents,
        name="AI-Investment-Analysis-Workflow",
        enable_group_chat=True,
    )

    # Configure speaker roles
    workflow.speaker_system.add_speaker(
        SpeakerConfig(
            role=SpeakerRole.COORDINATOR,
            agent=agents[0],  # Financial agent as coordinator
            priority=1,
            concurrent=False,
            required=True,
        )
    )

    workflow.speaker_system.add_speaker(
        SpeakerConfig(
            role=SpeakerRole.CRITIC,
            agent=agents[1],  # Risk agent as critic
            priority=2,
            concurrent=True,
        )
    )

    workflow.speaker_system.add_speaker(
        SpeakerConfig(
            role=SpeakerRole.EXECUTOR,
            agent=agents[2],  # Research agent as executor
            priority=2,
            concurrent=True,
        )
    )

    # Investment analysis task
    investment_task = """
        Create a comprehensive investment analysis for a $40k portfolio focused on AI growth opportunities:
```

```
1. Identify high-growth AI ETFs and index funds

2. Analyze risks and potential returns

3. Create a diversified portfolio allocation

4. Provide market trend analysis

Present the results in a structured markdown format.
"""


try:
    # Run workflow with retry
    result = await run_workflow_with_retry(
        workflow=workflow, task=investment_task, max_retries=3
    )


    print("\nWorkflow Results:")
    print("================")


    # Process and display agent outputs
    for output in result.agent_outputs:
        print(f"\nAgent: {output.agent_name}")
        print("-" * (len(output.agent_name) + 8))
        print(output.output)


    # Display group chat history if enabled
    if workflow.enable_group_chat:
        print("\nGroup Chat Discussion:")
        print("=====================")
```

```
            for msg in workflow.speaker_system.message_history:
                print(f"\n{msg.role} ({msg.agent_name}):")
                print(msg.content)

        # Save detailed results
        if result.metadata.get("shared_memory_keys"):
            print("\nShared Insights:")
            print("===============")
            for key in result.metadata["shared_memory_keys"]:
                value = workflow.shared_memory.get(key)
                if value:
                    print(f"\n{key}:")
                    print(value)

    except Exception as e:
        print(f"Workflow failed: {str(e)}")

    finally:
        await workflow.cleanup()


if __name__ == "__main__":
    # Run the example
    asyncio.run(main())
```