```
import os
from swarms import Agent, ConcurrentWorkflow
from swarm_models import OpenAIChat
from loguru import logger

from dotenv import load_dotenv

# Load environment variables
load_dotenv()

# Retrieve the OpenAI API key from the environment variable
api_key = os.getenv("GROQ_API_KEY")

# Initialize the model for OpenAI Chat
model = OpenAIChat(
    openai_api_base="https://api.groq.com/openai/v1",
    openai_api_key=api_key,
    model_name="llama-3.1-70b-versatile",
    temperature=0.1,
)


logger.add("swarms_example.log", rotation="10 MB")

agents = [
    Agent(
```

```
        agent_name=f"Term-Sheet-Analysis-Agent-{i}",
        system_prompt="Analyze the term sheet for investment opportunities.",
        llm=model,
        max_loops=1,
        autosave=True,
        dashboard=False,
        verbose=True,
        dynamic_temperature_enabled=True,
        saved_state_path=f"term_sheet_analysis_agent_{i}.json",
        user_name="swarms_corp",
        retry_attempts=1,
        context_length=200000,
        return_step_meta=False,
    )
    for i in range(3)  # Adjust number of agents as needed
]

# Initialize the workflow with the list of agents
workflow = ConcurrentWorkflow(
    agents=agents,
    metadata_output_path="term_sheet_analysis_metadata.json",
    return_str_on=True,
    auto_generate_prompts=True,
    auto_save=True,
)
```

```
# Define the task for all agents
task = "Analyze the term sheet for investment opportunities and identify key terms and conditions."

# Run the workflow and save metadata
metadata = workflow.run(task)
logger.info(metadata)



# # Example usage of the run_batched method
# tasks = [
#     "What are the benefits of a ROTH IRA?",
#     "How do I open a ROTH IRA account?",
# ]
# results = workflow.run_batched(tasks)
# print("\nRun Batched Method Output:")
# print(results)

# # Example usage of the run_async method
# async def run_async_example():
#     future = workflow.run_async(task)
#     result = await future
#     print("\nRun Async Method Output:")
#     print(result)

# # Example usage of the run_batched_async method
# async def run_batched_async_example():
```

```
#     futures = workflow.run_batched_async(tasks)
#     results = await asyncio.gather(*futures)
#     print("\nRun Batched Async Method Output:")
#     print(results)

# # Example usage of the run_parallel method
# parallel_results = workflow.run_parallel(tasks)
# print("\nRun Parallel Method Output:")
# print(parallel_results)

# # Example usage of the run_parallel_async method
# async def run_parallel_async_example():
#     parallel_futures = workflow.run_parallel_async(tasks)
#     parallel_results = await asyncio.gather(*parallel_futures)
#     print("\nRun Parallel Async Method Output:")
#     print(parallel_results)

# # To run the async examples, you would typically use an event loop
# if __name__ == "__main__":
#     asyncio.run(run_async_example())
#     asyncio.run(run_batched_async_example())
#     asyncio.run(run_parallel_async_example())
```