**Introduction**

The Agent Contracts project, created by Relari, is designed to allow users to implement and verify simple AI agents against predefined contracts. In this project, the goal was to simulate a refund processing AI agent that evaluates refund requests based on user input (order number, refund reason). The agent is implemented and tested against a contract specifying the conditions for refund eligibility and agent outputs. The integration of Relari-otel for telemetry and trace visualization via Jaeger was an additional component of the project.

**Project Overview**

* **Task Objective: To implement a simple refund processing AI agent and verify its contract against predefined conditions.**
* **Technologies Used:**
  + **Relari for agent contract management.**
  + **Relari-otel for instrumentation and trace reporting.**
  + **Jaeger for visualizing agent traces.**
  + **Docker for offline contract verification.**

**Project Steps and Execution**

**1. Setup and Installation**

**Cloning the Repo and Installing Dependencies**

* **The project began with cloning the Agent Contracts repository from *GitHub (git clone https://github.com/relari-ai/agent-contracts.git).***
* The Poetry package manager was used to install the dependencies (poetry install), which includes all necessary dependencies for running the agent and contracts.

**Installing Relari-otel**

* After setting up the dependencies, the next step was to install Relari-otel (pip install relari-otel[langchain,openai]) for the purpose of integrating OpenTelemetry with the agent.
* This step involved configuring trace reporting to Jaeger for visualization.
  + Challenges: The installation of relari-otel and proper Jaeger setup required careful configuration to ensure correct trace reporting.

**2. Agent Implementation**

**Refund Agent Logic**

* The core functionality involved implementing a simple RefundAgent class to simulate refund processing. The agent takes in an order number and refund reason as input, and outputs a refund confirmation or denial message based on predefined refund conditions.
  + Predefined refund conditions:
    - product defect
    - wrong item shipped
    - item not received
* If the refund reason matched one of the conditions, the agent would approve the refund; otherwise, the refund was denied.

**Contract Definition**

* A contract file (contracts.py) was defined with the following conditions:
  + Precondition: "User provides valid order number."
  + Pathcondition: "Agent confirms refund eligibility."
  + Postcondition: "Agent outputs refund confirmation or denial message."
* The contract file also handled the verification of whether the agent met the required conditions.

**3. Integration of Relari-otel**

**Relari-otel for Telemetry**

* The agent script was instrumented with Relari-otel to trace the agent's execution and send telemetry data.
* Challenges: There were a few hiccups with ensuring that the trace data was correctly captured and sent to Jaeger. Specifically, the ***Relari.init(project\_name="RefundTest")*** method was used, but there were issues with trace batching and reporting initially. After tweaking configurations and ensuring that the correct trace context was initialized, the system successfully reported traces to Jaeger.

**4. Docker and Contract Verification**

**Docker Verification**

* To verify the agent’s compliance with the contract, offline Docker verification was used by running the make docker-verification command.
* This verification checks if the agent's behavior aligns with the defined contract and runs in a controlled, reproducible environment.
  + Challenges: Docker setup and ensuring the container environment was correctly configured to run the contract verification was tricky. After multiple attempts to resolve dependency issues and container configurations, the agent passed the verification step.

**5. Jaeger Trace Visualization**

**Jaeger Setup**

* After successfully running the agent and its trace data, Jaeger was set up to visualize the traces from the agent's execution.
  + Challenges: Initial issues with Jaeger configuration and making sure the trace data was visible in the Jaeger dashboard. Adjustments to the trace context and communication between Relari-otel and Jaeger were necessary.

**6. Testing and Final Validation**

* Once the agent, contract, and verification were successfully integrated, testing was carried out to ensure everything was working as expected.
  + Key testing steps included:
    - Running test cases with various refund reasons (both eligible and non-eligible).
    - Verifying that the contract conditions were met.
    - Checking the trace outputs in Jaeger for visualization.
  + Challenges: There were instances where the trace outputs in Jaeger were delayed, and the verification needed to be adjusted to handle edge cases like invalid or ambiguous refund reasons.

**Challenges Faced**

1. Trace Reporting Issues: Configuring Relari-otel to send trace data to Jaeger was tricky. It required adjusting trace batch sizes and ensuring that the trace context was properly set up.
2. Docker Configuration: Running the offline verification with Docker had a few issues with dependency management and proper container configuration.
3. Refund Logic and Contract Verification: The logic for refund eligibility and contract validation needed to be carefully designed to meet the contract's conditions. Additionally, handling edge cases such as invalid refund reasons required extra attention.
4. Jaeger Setup: Configuring Jaeger to display the trace data required troubleshooting with the Jaeger UI to ensure the traces were displayed correctly.

**Total Time Breakdown**

1. **Setup and Installations: 30 minutes (Cloning repo and installing dependencies)**
2. **Relari-otel Installation and Configuration: 2 hours**
3. **Refund Agent Logic Implementation: 2.5 hours**
4. **Contract Definition and Verification: 1.5 hours**
5. **Docker Setup and Offline Verification: 5 hours**
6. **Jaeger Setup and Trace Visualization: 3 hours**
7. **Testing and Final Validation: 1.5 hours**

**Total Time: Approximately 16 hours over a span of 3 days.**

**Conclusion**

The Agent Contracts project was successfully implemented and verified. By following the detailed steps outlined in the report, the Refund Processing Agent was able to process refund requests and meet the contract conditions. Integration of Relari-otel for traceability and visualization in Jaeger added a layer of monitoring that was helpful in understanding the agent's behavior and ensuring compliance with the contract. The Docker-based offline contract verification ensured that the agent's performance was consistent across different environments.

However, there were multiple challenges during the project, such as configuring Relari-otel, resolving Docker issues, and setting up Jaeger. These required careful troubleshooting and configuration adjustments.

**The project took approximately 16 hours to complete over the span of 3 days, with the majority of the time spent on troubleshooting configuration issues.**

A screenshot of a computer

AI-generated content may be incorrect.A cartoon blue animal with hat

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Output Implementation

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.