

AI-Powered Trading Reconciliation Agent

Overview

This project is an **AI-powered Trading Reconciliation Agent** that detects and analyzes anomalies in trading data. It utilizes **Retrieval-Augmented Generation (RAG)** to enhance AI-driven anomaly detection by incorporating historical trade data for improved accuracy and contextual reasoning.

Features

- **Automated Anomaly Detection:** Identifies mismatches in trade data using predefined tolerance levels.
- **Retrieval-Augmented Generation (RAG):** Fetches similar past trades from a database to provide contextual insights.
- **AI-Driven Resolution Suggestions:** Uses an LLM (via **Ollama**) to classify anomalies and recommend resolutions.
- **Database Integration:** Stores trade history in **SQLite** for fast retrieval.
- **Modular & Scalable:** Built using **Python**, **Streamlit**, and **LangChain**.

Architecture

1. Data Storage & Retrieval

- **SQLite database** (`database.py`) stores all historical trade records.
- **Vector Embeddings** (using `all-MiniLM-L6-v2`) enable efficient similarity-based retrieval of past trades.

2. Anomaly Detection (`reconciliation.py`)

- Scans for **quantity mismatches** exceeding tolerance thresholds.
- Triggers the retrieval pipeline for similar past cases.

3. RAG-Enhanced AI Reasoning (`reasoning.py`)

- Retrieves relevant historical trades before AI processing.
- Constructs a **structured prompt** with past case data for the LLM.
- AI model (via Ollama) **analyzes anomalies and suggests resolutions**.

Key Components and How RAG is Applied

1. Database & Retrieval Module (**database.py**)

- “This module stores all trade records and supports embedding-based search.
- When an anomaly is detected, we **fetch similar historical trades using vector embeddings** before passing them to the LLM.
- We use **sentence embeddings (MiniLM)** to measure trade similarity and **fetch relevant cases dynamically**.

This retrieval step is what enables **RAG to enhance reasoning**—instead of the AI generating generic responses, it receives actual trade data for context-aware analysis.”

2. Reconciliation Logic (**reconciliation.py**)

- “This module scans trade records and flags mismatches where actual quantities **breach predefined tolerance thresholds**.
 - Once a break is detected, the system triggers the **retrieval pipeline**, fetching similar past trades for comparison.”
-

3. AI Reasoning & RAG Integration (**reasoning.py**)

- “Once historical trades are retrieved, they are packaged into a structured prompt.
- The **RAG pipeline then feeds this contextual data into the AI model (running on Ollama with **gemma:latest**)**.
- The model classifies the anomaly and suggests a resolution by leveraging both historical trade patterns and its trained knowledge.”

Why RAG?

This approach ensures that every AI-generated insight is **contextually relevant**, rather than a generic response.

Setup Instructions

1. Clone the Repository

```
git clone https://github.com/ewfx/sradg-ai-challengers.git  
cd sradg-ai-challengers/code/src
```

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Run the Streamlit UI

```
streamlit run ui.py
```

Steps to Run the app

Step 1: Upload Trade Data

- “We start by uploading a CSV file containing trade data. The system reads and inserts it into our SQLite database.”

Step 2: Detect Breaks

- “Once the file is processed, the system detects mismatches using predefined tolerance levels and flags them in a table.”

Step 3: Apply RAG for AI-Driven Analysis

- “For each flagged trade, clicking ‘Analyze Trade’ retrieves the most **similar past cases** from the database.”
- “These retrieved trades are then **embedded into a structured AI prompt**, enabling the model to provide precise, data-backed recommendations.”

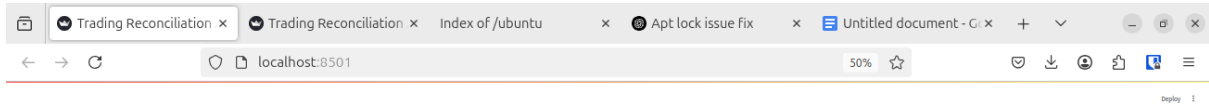
Step 4: AI-Generated Resolution

- “The model classifies the break (e.g., data entry error, late settlement, corporate action issue).”
- “It then suggests a **contextual resolution**, explaining how similar past cases were resolved.”

Usage

1. **Upload a trade data file** (CSV or JSON).
2. The system **detects anomalies** and flags mismatches.
3. **RAG fetches relevant past trades** from the database.
4. AI **analyzes the anomaly** and provides a **context-aware resolution**.

App Screen Shots



Trading Reconciliation Agent

Upload CSV File

Drag and drop file here
Limit 200MB per file • CSV

Browse Files

All Trades

id	tradeDate	quantity_a	quantity_b	tolerance	comment	embedding
0	1 2025-03-25		300	50	5 Slight discrepancy	173.181,182,189,35,25,209,189,190,102,81,209,29,215,189,2,172,189,39,140,104,243,57,41,229,133,189,139,249,63,4
1	2 2025-03-25		250	250	5 No issue	230,231,187,189,77,230,131,189,82,111,39,61,136,111,241,187,233,42,32,189,190,124,6,61,240,34,103,189,39,189,181
2	3 2025-03-24		300	310	10 Minor overage	291,140,140,189,144,26,63,189,18,10,9,81,41,40,278,29,133,86,180,180,797,717,25,41,14,2781,17,189,279,181,131,82,
3	4 2025-03-23		250	140	5 Under reporting detected	173,254,236,189,39,113,187,187,189,118,26,205,124,244,03,202,05,89,61,240,20,30,61,67,86,77,189,211,240,86,61,33
4	5 2025-03-27		400	395	8 Close match	41,79,279,189,18,36,26,180,222,189,119,188,84,34,209,187,209,3,176,180,18,29,117,8,39,131,189,149,134,88,82,
5	6 2025-03-21		200	205	5 Minor adjustment	106,255,91,189,14,237,159,59,25,131,135,61,235,235,209,59,35,82,231,189,161,219,255,61,228,164,196,189,126,115,54,6
6	7 2025-03-20		300	360	10 Slight overage	188,245,183,189,1,176,247,187,119,165,233,80,120,19,190,60,189,15,244,188,26,154,214,188,34,231,189,189,180,65,154
7	8 2025-03-19		500	480	15 Potential reporting error	137,206,179,189,135,49,146,60,31,10,75,60,77,18,2,61,30,31,15,188,183,189,176,60,139,52,93,189,200,184,144,61,53,255

Detected Breaks

▲ Trade on 2025-03-25 - Expected: 100.0, Actual: 90.0, Tolerance: 5.0, Comment: Slight discrepancy

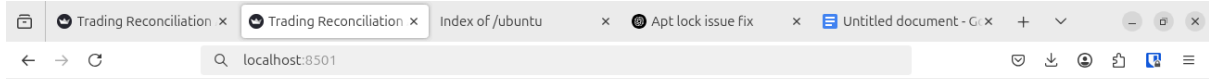
Analyze Trade

▲ Trade on 2025-03-23 - Expected: 150.0, Actual: 140.0, Tolerance: 5.0, Comment: Under reporting detected

Analyze Trade

▲ Trade on 2025-03-19 - Expected: 500.0, Actual: 480.0, Tolerance: 15.0, Comment: Potential reporting error

Analyze Trade



Detected Breaks

▲ Trade on 2025-03-25 - Expected: 100.0, Actual: 90.0, Tolerance: 5.0, Comment: Slight discrepancy

Analyze Trade

▲ Trade on 2025-03-23 - Expected: 150.0, Actual: 140.0, Tolerance: 5.0, Comment: Under reporting detected

Analyze Trade

LLM Analysis:

Classification: Potential Reporting Error

Reasoning:

- The reconciliation break indicates an under-reporting of 10 (150 - 140) units.
- The comment associated with the break suggests a potential reporting error.
- Historical similar cases contain a record with a similar situation (500-480-15) classified as a potential reporting error.

Suggested Resolution:

- Review the underlying data for System A to identify any potential reporting errors.
- Compare the reporting methodology used for System A and System B to ensure consistency.
- Investigate whether the under-reporting is a systemic issue or an isolated event.
- Consider adjusting the reconciliation break quantity for System A based on the identified reporting error.

Additional Considerations:

- The tolerance level of 5.0 is relatively low, which may increase the sensitivity of the reconciliation process to minor discrepancies.
- The historical similar cases provide evidence that under-reporting of this magnitude has occurred in the past, suggesting it is a potential explanation for the reconciliation break.

▲ Trade on 2025-03-19 - Expected: 500.0, Actual: 480.0, Tolerance: 15.0, Comment: Potential reporting error

Analyze Trade