Smarter Reconciliation and Anomaly Detection

System Overview

In the first use case, The Anomaly Detection System is designed to identify discrepancies in financial reconciliation data using a combination of Machine Learning (ML) models and Gen AI (Llama-3.3-70B). The system analyzes historical transaction data to detect patterns and compares them with new transaction records to flag anomalies.

This solution aims to automate reconciliation by:

- Detecting irregularities in financial transactions
- Explaining anomalies using an LLM (Llama-3.3-70B)
- Suggesting next steps for resolution
- Generating Excel reports with identified anomalies

This second use case of the Anomaly Detection System focuses on financial reconciliation using a hybrid approach:

- Rule-based classification for quick filtering of obvious cases.
- AI-driven classification using Mistral SABA-24B via GROQ API for complex anomaly detection.
- JSON-based AI responses to explain anomalies and suggest root causes.
- Automated CSV file processing to analyze large-scale financial records.

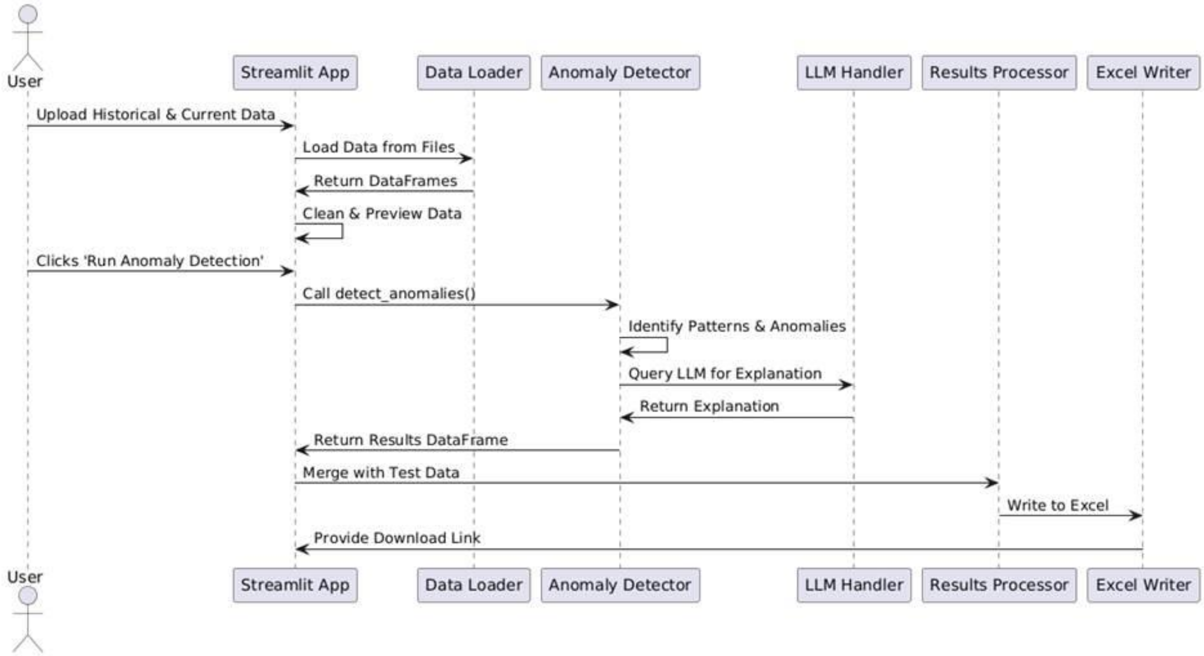
Technology Stack

- **Programming Language**: Python
- **Libraries Used**: SciPy (for curve fitting), scikit-learn, Pandas, NumPy, Requests, JSON, RegEx
- **AI Model**: LangChain (for LLAMA 3.3), Mistral SABA-24B via GROQ API
- **Web UI**: Streamlit
- **File Handling**: XlsxWriter, CSV processing with Pandas

USE CASE 1: IHub Reconciliation

System Architecture

Sequence Diagram



This sequence diagram represents the step-by-step process flow in the system:

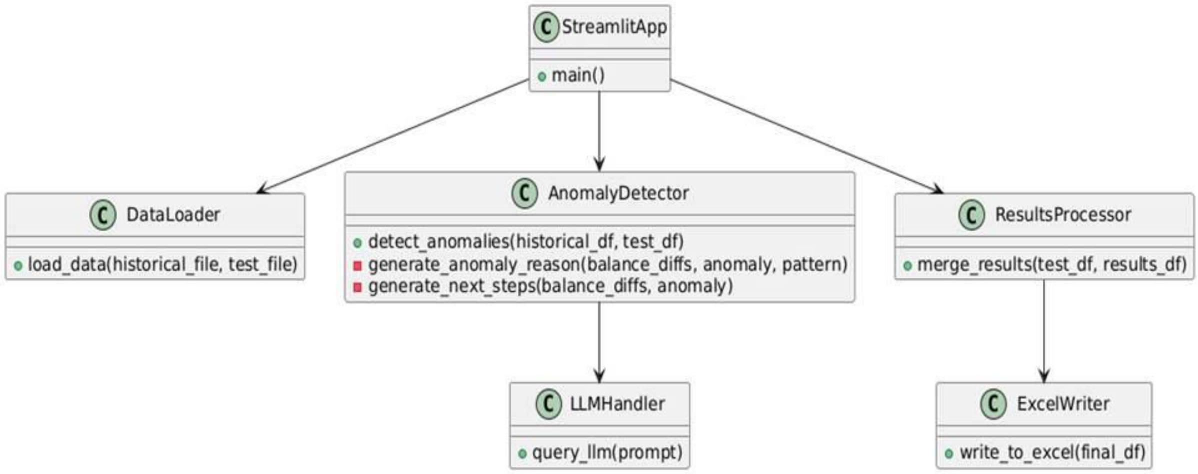
1. User uploads historical and test data: The user selects files through the Streamlit interface.
2. Data Loading: The system reads and processes the uploaded files into Pandas DataFrames.
3. Data Preview: The user can view a preview of the loaded datasets.
4. User initiates anomaly detection: The system runs the anomaly detection algorithm when the user clicks the Run Anomaly Detection button.
5. Anomaly detection process:
    - The system checks for anomalies using historical trends.
    - It evaluates the data against predefined patterns (linear, sinusoidal, quadratic, logarithmic).

- If anomalies are found, it queries the LLM (Llama-3.3-70B) for explanation.

6. Results Processing: The detected anomalies are merged with the test dataset.

7. Exporting results: The final processed data is written to an Excel file, which the user can download.

Class Diagram



The class diagram illustrates the object-oriented structure of the system:

1. StreamlitApp:
   - The main entry point for the application.
   - Handles user interactions, file uploads, and UI display.
2. DataLoader:
   - Loads and processes historical and test datasets.
   - Calculates balance differences between GL and iHub balances.
3. AnomalyDetector:
   - Identifies anomalies based on historical trends.
   - Applies curve-fitting models to detect patterns in data.
   - Uses LLMHandler to provide anomaly explanations.
4. LLMHandler:

- Sends anomaly-related queries to the Llama-3.3-70B model using LangChain.
- Receives and returns explanations for detected anomalies.

5. ResultsProcessor:
    - Merges test data with detected anomalies.
    - Ensures final output includes anomaly classifications and recommendations.
6. ExcelWriter:
    - Exports the final anomaly detection results to an Excel file.
    - Provides a downloadable link in the UI.

Approach

1. Data Preprocessing

The system loads historical reconciliation data in CSV/XLSX format. Since column structures often vary, a standardization process is applied:

- Column names are converted to lowercase, and spaces are removed to ensure consistency.
- The **balance difference** is computed for each account as:

Balance Difference=GL Balance−iHub Balance\text{Balance Difference} = \text{GL Balance} - \text{iHub Balance}Balance Difference=GL Balance−iHub Balance

This difference is crucial in identifying anomalies in reconciliation.

2. Anomaly Detection Process

The system applies multiple regression models to detect unusual patterns:

- **Linear Model:** Identifies trends in balance differences.
- **Sinusoidal Model:** Detects periodic fluctuations in financial records.
- **Quadratic Model:** Flags parabolic trends in balance movements.

- **Logarithmic Model:** Recognizes sudden changes that stabilize over time.

If the predicted balance difference significantly deviates from the expected trend, it is flagged as an **anomaly**.

3. **AI-Generated Explanations**

When an anomaly is detected, a structured prompt is created and sent to **Llama-3.3-70B**, which:

- **Analyzes historical patterns** to determine why the anomaly occurred.
- **Generates a detailed explanation** for reconcilers.
- **Suggests corrective actions** based on historical trends and past resolutions.

4. **Results Processing & Report Generation**

Once anomalies are detected, they are merged with the test dataset to provide a comprehensive reconciliation report.

Challenges & Solutions

1. **Handling Dynamic Dataset Structures**

- Issue: Datasets from different sources had inconsistent column names and structures.
- Solution: Implemented automated column mapping to standardize input data dynamically.

2. **Model Selection & Performance**

- Issue: Some anomalies did not follow simple linear trends, leading to misclassification.
- Solution: Introduced hybrid regression models (sinusoidal, quadratic, logarithmic) to capture complex patterns.

3. **AI Response Optimization**

- Issue: AI-generated explanations were sometimes too generic or verbose.

- Solution: Improved prompt engineering to ensure concise and actionable insights.

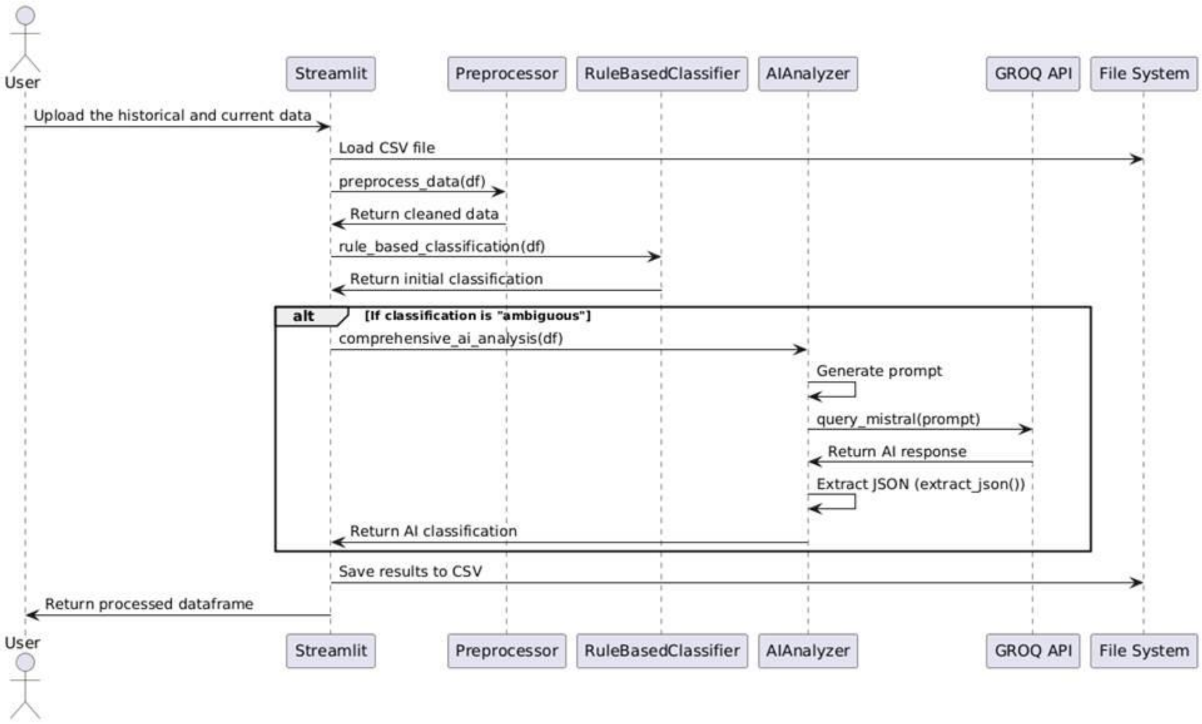
Results & Outcomes

- Successfully detected anomalies in financial reconciliation.
- AI-powered explanations provided clear context for flagged anomalies.
- Reduced manual efforts by automating reconciliation analysis.
- Streamlined report generation, making it easy for analysts to review anomalies.
- A user-friendly Streamlit dashboard provided real-time insights

USE CASE 2: Catalyst reconciliation

System Architecture

Sequence Diagram



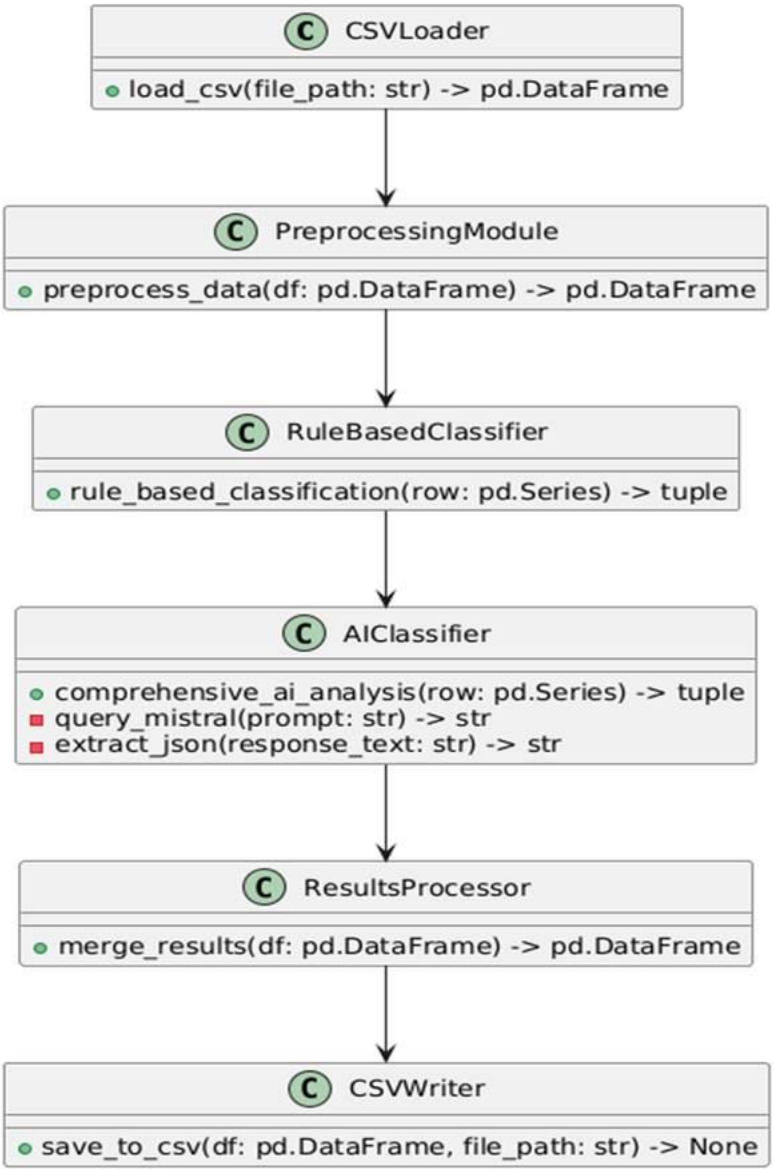
The sequence diagram outlines the flow of execution:

1. User initiates by upload the historical and current data on streamlit.
2. The datasets are loaded (pandas.read_csv).
3. Preprocessing (preprocess_data):
    - Converts date columns to datetime.
    - Converts numeric columns to numeric format.
    - Computes price and quantity differences.
4. Rule-Based Classification (rule_based_classification):
    - Determines if reconciliation is "good_to_go" or "ambiguous."
5. AI-Based Classification (comprehensive_ai_analysis) (only for ambiguous cases):

- Calls query_mistral(), which sends a request to the GROQ API.
- Receives the AI-generated classification.
- Extracts JSON response (extract_json()).

6. Saves Results (pandas.to_csv).

Class Diagram



CSVLoader → Loads reconciliation data from a CSV file.
PreprocessingModule → Cleans and structures the data.
RuleBasedClassifier → Applies predefined rules to classify reconciliation breaks.
AIClassifier → Uses AI to analyze ambiguous cases.
ResultsProcessor → Merges rule-based and AI classifications.
CSVWriter → Saves the final results to a CSV file

Approach & Methodology

1. **Data Preprocessing**

The system ingests transaction data from financial records and applies preprocessing:

- **Date fields** are converted to datetime format for consistency.
- **Numeric fields** such as price and quantity are standardized.
- **Price and quantity differences** are computed:

$$\text{Price Difference} = |\text{Catalyst Price} - \text{Impact Price}|$$

$$\text{Quantity Difference} = |\text{Catalyst Quantity} - \text{Impact Quantity}|$$

These computed values help in identifying **transaction anomalies**.

2. **Rule-Based & AI Classification**

The classification process has **two stages**:

- **Rule-Based Classification:**
    - § If Catalyst or Impact price fields are empty/zero, the transaction is automatically marked as “good_to_go” (indicating a backend issue or network delay).
    - § If price and quantity differences are within acceptable thresholds, the transaction is also approved.
    - § If the transaction does not meet predefined rules, it is classified as "ambiguous" and sent for AI-based classification.

- **AI-Based Classification:**

Transactions marked as ambiguous are sent to Mistral SABA-24B via GROQ API.

The AI model analyzes multiple attributes (price, quantity, trade date, inventory codes) and returns a structured JSON response with:

§ Classification: “good_to_go” or “anomaly.”

§ Reason: Justification for classification.

§ Root Cause: If marked as an anomaly, the probable reason is provided.

§ Rule-based and AI classifications are merged into a final dataset for reporting.

Challenges & Solutions

1. **Handling Complex Financial Patterns**

- Issue: Some financial transactions had irregular price/quantity differences that did not fit predefined rules.
- Solution: Introduced hybrid AI + rule-based filtering to handle ambiguous cases.

2. **JSON Extraction & Parsing Issues**

- Issue: AI sometimes returned unstructured text instead of a clean JSON response.
- Solution: Implemented RegEx-based JSON extraction to ensure structured data processing.

3. **Performance Optimization**

- Issue: Querying the AI model for every transaction increased processing time.
- Solution: Introduced rule-based pre-filtering, reducing unnecessary AI calls and improving performance.

Results & Outcomes

- Successfully classified financial transactions into “good_to_go” and “anomaly” categories.

- Reduced AI calls by filtering non-ambiguous cases using rule-based logic.

- AI-generated explanations helped analysts understand anomaly root causes.

- Automated CSV report generation, making classification insights easily accessible.