Solution Architecture

Overview

This project provides an end-to-end regulatory data profiling solution. It processes regulatory instructions and transaction data to:

1. Extract profiling rules using LLM-based AI.
2. Perform anomaly detection on transactions.
3. Generate Python validation code for compliance checks.
4. Compute risk scores for transactions and provide suggested remediations.

System Architecture

The solution is designed with a modular and scalable architecture, consisting of the following components:

Frontend (Streamlit UI)

- Users upload regulatory instructions and transaction data.
- Displays flagged transactions, anomalies, risk scores, and generated rules.
- Communicates with the backend via REST API calls.

Backend (Flask API)

- Handles data ingestion and processing.
- Uses LLMs to generate profiling rules dynamically.
- Performs anomaly detection using unsupervised ML techniques.
- Generates Python validation code for compliance.
- Computes risk scores and provides suggested remediation actions.

AI/ML Models

- LLM (Claude, OpenRouter API): Extracts rules from regulatory text.
- Anomaly Detection (Isolation Forest, DBSCAN): Flags unusual transactions.
- Risk Scoring (Custom Algorithm): Assigns risk scores to transactions.

Database/Storage

- CSV-based storage for simplicity (Can be extended to SQL databases).
- Stores:
  - Regulatory Instructions
  - Transaction Data

- Generated Rules
- Flagged Transactions
- Risk Scores & Anomaly Reports

Data Flow

1. User uploads regulatory instructions and transaction data.
2. Backend processes data, generates rules using LLMs, and applies anomaly detection.
3. Flagged transactions, anomalies, and risk scores are computed and stored.
4. Results are sent back to the frontend for display.