Transaction Anomaly Alert System - Technical Report

Objective:

Implement a basic monitoring system capable of identifying transaction anomalies in real time and issuing alerts. The system classifies transaction statuses and detects outliers that may indicate incidents or system failures.

Data:

Two CSV files were provided: `transactions.csv` and `transactions_auth_codes.csv`. The first contains the core transaction logs including timestamps and statuses, while the second maps auth codes for enriched analysis.

Approach:

The backend was built using Flask and Python. An endpoint `/alert` was implemented to identify anomalies based on status frequency comparison against historical baselines. If error-type statuses (e.g., 'failed', 'denied', 'reversed') exceed 50% above average per-minute rates in the last 5 minutes, the system returns a JSON alert.

Model Type:

This solution uses a rule-based model. No machine learning was applied due to the simplicity and clear threshold expectations defined in the challenge.

Endpoints:

- `/` : Health check. - `/transactions` : All transaction data. - `/auth_codes` : All auth codes. - `/summary` : Count per status. - `/alert` : Anomaly detection over the last 5 minutes.

Visualization:

Grafana was used for real-time visualization. The API's JSON endpoints can be consumed using the SimpleJSON plugin to render panels that monitor each transaction status and highlight when alerts are triggered.

How to Run:

Run `pip install -r requirements.txt` and start the API with `python app.py`. The system will load the CSV files and be ready to serve real-time queries on port 5000.

Conclusion:

This monitoring prototype demonstrates the core capability of real-time anomaly detection in financial transactions. Its modular design allows for integration with real dashboards and alerting systems.