🏦 Explainable Transaction Anomaly Monitoring System

Complete Project Workflow & System Explanation

---

1. Introduction

Banks process **millions of transactions every day**, and a small fraction of them may involve: - Fraudulent activity - Account takeover - Misuse or suspicious behaviour.

The challenge is not only *detecting anomalies*, but also **explaining clearly why a transaction was flagged**, in a way that: - Compliance teams understand - Auditors can verify - Systems can run securely (offline)

This project presents a **fully explainable, rule-based transaction anomaly detection system** that mirrors how real banking fraud & AML engines work—without relying on black-box machine learning models.

---

2. Why Not Traditional Machine Learning?

In real banking environments:

- Fraud labels are **rare, delayed, or incorrect**
- Regulators require **transparent decision logic**
- Models must be **auditable years later**
- Secure networks often **do not allow internet access**

Because of this, many real systems still rely heavily on **rules + explainable scoring**, especially as a *first line of defence*.

Our system is designed exactly for this layer.

---

3. High-Level Workflow Overview

The complete system workflow is:

1. Load raw transaction data
2. Process data in parallel (multiprocessing)
3. Engineer behavioural & contextual features
4. Convert features into binary risk signals
5. Compute an explainable risk score

6. Apply confirmation logic to reduce noise
7. Generate human-readable outputs & statistics

Each step is **independent, modular, and auditable**.

---

4. Input Data Description

4.1 Dataset Characteristics

- ~150,000 transactions
- ~138,000 users (very sparse history)
- Most users have **only 1–2 transactions**

This sparsity strongly influences the system design.

4.2 Core Input Fields

Field	Purpose
transaction_id	Unique identifier
sender_account	User/account reference
amount	Transaction value
timestamp	Time of transaction
device_hash	Device fingerprint
ip_address	Network context
location	City-level location
fraud_label	Used **only for evaluation**, not detection

---

5. Multiprocessing Design (Performance & Scalability)

Why Multiprocessing?

Transaction datasets are large, and feature engineering is: - CPU-intensive - Independent per user or per transaction

To make the system **production-realistic**, we use **Python multiprocessing**.

How It Works

- Dataset is split into chunks
- Each process handles:
    - Feature computation
    - Risk signal generation

- Results are merged safely

Benefits

- Faster processing on large datasets
- Mirrors real batch-processing pipelines
- Safe for offline & secure environments

(No shared state, no race conditions)

---

6. Feature Engineering – Signal Design Logic

Each feature answers **one simple banking question**.

6.1 Amount Deviation

Question: Is this amount unusual for this user?

- Compute user-level mean and standard deviation
- Calculate z-score
- Even moderate deviation is meaningful due to sparse history

Why it matters: Fraudsters often test accounts with unusually high or low amounts.

6.2 New Device Detection

Question: Has this device been used before?

- First-time device → risk
- Known device → safe

Bank relevance: Strong signal for account takeover and credential compromise.

6.3 New IP Address

Question: Is the network context unfamiliar?

- New IP → weak risk signal
- Never used alone to flag

Why weak? Mobile networks change IPs frequently.

6.4 Location Change

Question: Did the transaction location suddenly change?

- City-level comparison
- First transaction → unknown baseline

Use case: Detects abnormal geographic behaviour.

6.5 Off-Hour Activity

Question: Is the transaction happening at an unusual time?

- Determine the user's dominant transaction hour
- Flag deviations

Example: User transacts mostly at 10–11 AM, but the transaction occurs at 3 AM.

6.6 Velocity Anomaly (Simulation Mode)

Problem: The Real dataset is sparse

Solution: - Introduce controlled simulation - Create short bursts of transactions - Detect rapid activity within small time windows

Important: Simulation mode is **clearly labelled** and used only for demonstration.

---

7. Risk Signal Layer

Each engineered feature is converted into a **binary risk flag**:

Risk Flag	Meaning
risk_amount	Unusual transaction amount
risk_new_device	New device
risk_new_ip	New IP address
risk_location_change	Location changed
risk_off_hour	Unusual time
risk_velocity_sim	Rapid transaction burst

Each signal alone is weak — **strength comes from combination**.

---

8. Explainable Risk Scoring Engine

Additive Scoring Logic

```
Risk Score = Sum of active risk flags
```

- No weights
- No ML
- Fully transparent

Risk Interpretation

Score	Meaning
0–2	Low risk
3	Medium (novel behaviour)
4	High risk
5+	Critical risk

This mirrors how real rule engines escalate alerts.

---

9. Two-Layer Decision Framework

Layer 1: Broad Anomaly Detection

- Flags transactions with multiple risk signals
- High recall

Layer 2: Confirmation Layer

A transaction is confirmed anomalous only if: - Risk score ≥ threshold - AND strong behavioural evidence exists

Why this matters: Reduces false positives and analyst fatigue.

---

10. Example Walkthrough – One Transaction End-to-End

This section provides a **deep, narrative-style walkthrough** of how the system processes a *single transaction from raw input to final decision*. It is intentionally detailed so that **any judge, auditor, or finance professional can trace the logic without a technical background**.

The goal of this section is to answer one question clearly:

“Why exactly did the system flag this transaction, and can I defend this decision in front of a regulator?”

This section explains **one transaction step-by-step**, showing how the system arrives at a decision. This is written especially for **non-ML and finance-background reviewers**.

Example Transaction

Field	Value
Transaction ID	TXN001
Sender Account	ACC123
Amount	₹5,000
Timestamp	2025-01-08 14:30
Location	Mumbai
Device	New device
IP Address	New IP

Step-by-Step Decision Flow

Step 1: Amount Deviation Check

- User’s historical transaction range: ₹500 – ₹2,000
- Current transaction: ₹5,000
- Amount is **significantly higher than normal**

➡️ `risk_amount = 1`

Step 2: Device Check

- Device has **never been seen before** for this account
- Strong indicator of potential account takeover

➡️ `risk_new_device = 1`

Step 3: IP Address Check

- IP address is **new for this user**
- Treated as a weak supporting signal

➡️ `risk_new_ip = 1`

Step 4: Location Consistency Check

- Previous transaction location: Delhi
- Current location: Mumbai
- Sudden city-level change detected

➡️ `risk_location_change = 1`

Step 5: Time-of-Day Behaviour Check

- User usually transacts between 9–11 AM
- Current transaction occurred at 2:30 PM
- Outside dominant behaviour window

➡️ `risk_off_hour = 1`

Risk Score Calculation

All active risk signals are added:

```
Risk Score = 1 + 1 + 1 + 1 + 1 = 5
```

Final Decision

Criterion	Result
Risk score ≥ threshold	✅ Yes
Multiple independent signals	✅ Yes
Strong behavioural evidence	✅ Yes

➡️ **Transaction is confirmed anomalous**

Explainable Output Generated

```json
{
  "transaction_id": "TXN001",
  "risk_score": 5,
  "is_anomalous": true,
  "reasons": [
    "Unusual transaction amount",
    "New device detected",
    "New IP address detected",
    "Transaction location changed",
```

```
    "Transaction at an unusual time"
  ]
}
```

This explanation is exactly what a **bank analyst or auditor would see**, making the decision transparent and reviewable.

---

11. Output Generation

10.1 Transaction-Level Output

Each flagged transaction includes:

- Risk score
- Final anomaly decision
- Human-readable reasons

Example:

```
"New device detected"
"Transaction at an unusual time"
```

10.2 Output Files

File	Purpose
summary.json	Overall system health
flagged_transactions.csv	Investigation-ready file
flagged_transactions.parquet	Fast analytics
stats_reasons.csv	Common fraud patterns
stats_risk_scores.csv	Risk distribution

---

11. Interpreting Anomaly Rate

Rate	Interpretation
< 0.1%	Very strict
0.1–1%	Production-balanced
1–5%	Sensitive
> 5%	Very sensitive

Thresholds can be adjusted based on bank policy.

12. Dashboard & Demonstration

The Streamlit dashboard provides:

- Offline execution
- Risk score filtering
- Simulation toggle
- Detailed reason inspection

This allows judges to **see decisions live**, not just code.

13. Why This System Is Bank-Ready

✓ Explainable ✓ Auditable ✓ Offline-capable ✓ Handles sparse users ✓ Multiprocessing-enabled ✓ Mirrors real fraud pipelines

14. Future Extensions

- Learnable weights
- Investigator feedback loop
- Session-based behaviour
- Integration with alert queues

15. Final Summary

This project demonstrates how a bank-grade, explainable transaction anomaly detection system can be built using transparent rules, parallel processing, and multi-signal behavioural logic — without relying on black-box machine learning.