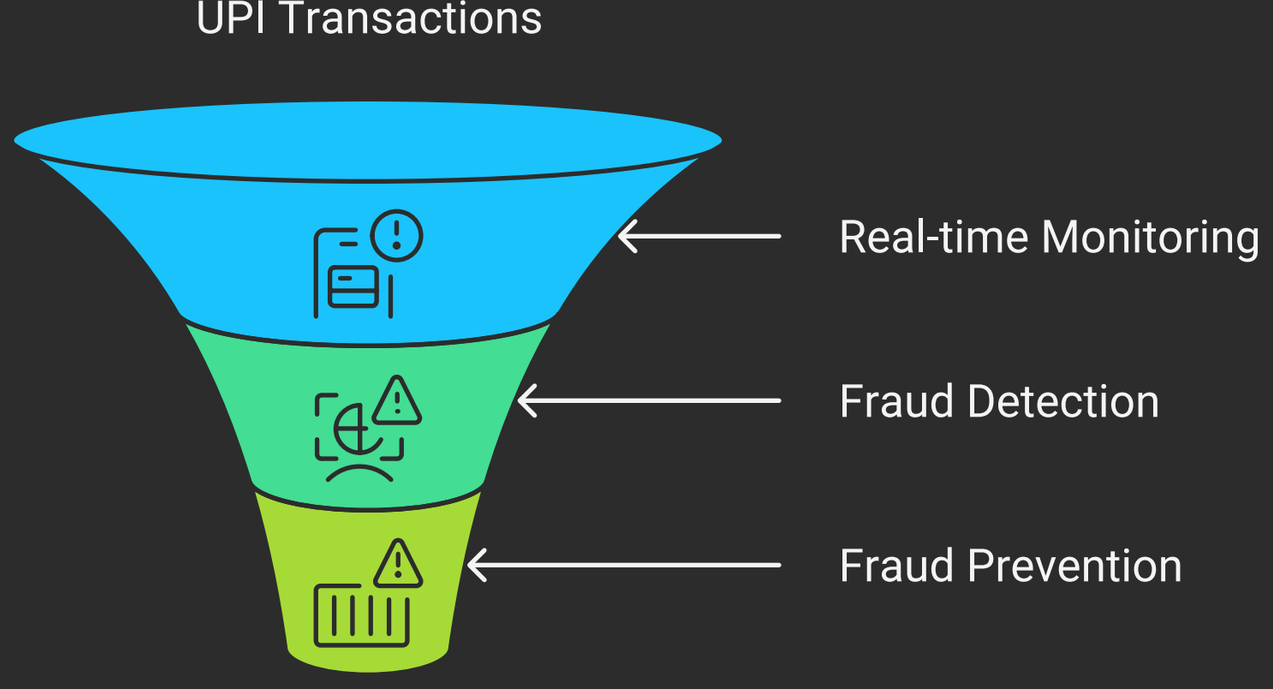
AI-Based Suspicious Transaction Detection on UPI (NPCI Workflow + Our Enhancement)

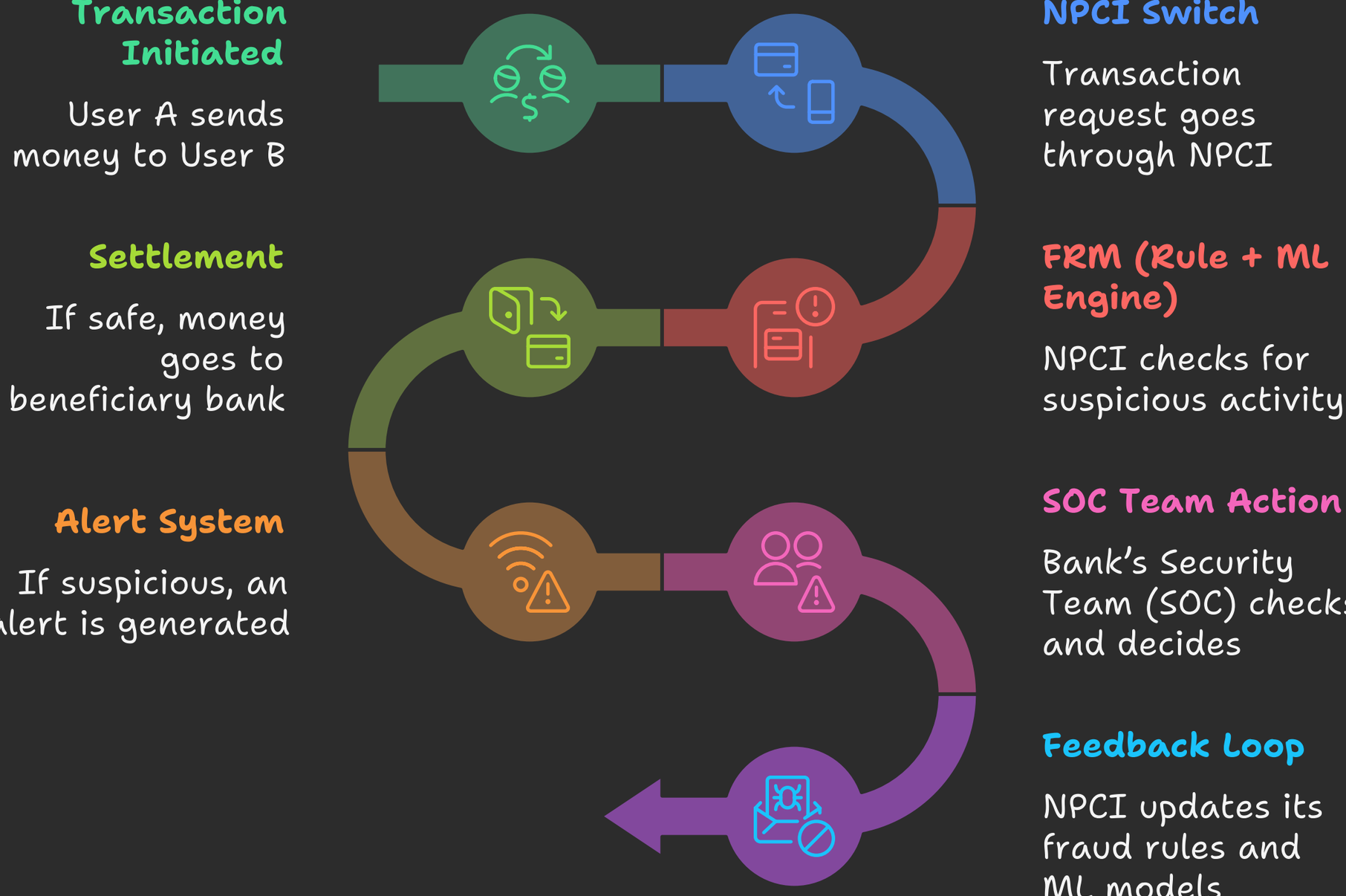
1. Background

- NPCI (National Payments Corporation of India) manages UPI transactions.
- UPI has a transaction limit (1L per day per account).
- Fraudsters use tricks like **mule accounts, split transactions, and social engineering (OTP sharing, fake links)** to bypass limits and steal money.
- NPCI already has a **Fraud Risk Management (FRM) system** which checks transactions in real-time.



2. Current NPCI Workflow (Simplified)

1. **Transaction Initiated** → User A sends money to User B.
2. **NPCI Switch** → Transaction request goes through NPCI.
3. **FRM (Rule + ML Engine)** → NPCI checks for suspicious activity.
4. **Settlement** → If safe, money goes to beneficiary bank.
5. **Alert System** → If suspicious, an alert is generated.
6. **SOC Team Action** → Bank's Security Team (SOC) checks and decides to allow, hold, or freeze.
7. **Feedback Loop** → NPCI updates its fraud rules and ML models for future



3. Problem with Current System

- Mostly **rule-based** (e.g., “>1L in one go suspicious”).
- **Reactive approach** – detects after fraud attempt.
- Limited **behavioral analysis** (can’t fully track mule accounts or unusual timings).

4. Our AI-Enhanced Workflow

We propose adding an **AI Anomaly Engine** on top of NPCI’s system:

Key Features:

1. **Behavioral Pattern Detection**
    - Learns normal user behavior (time of transactions, usual amount, regular accounts).
    - Flags abnormal cases (e.g., midnight ₹90k transfer when user usually does daytime ₹2k transfers).
2. **Auto-Freeze Button**
    - Suspicious cases (3–4 AM txn, sudden ₹1L+ via mule accounts, OTP misuse) → Account auto-freeze.
    - Prevents fraudster from withdrawing stolen money.
3. **Mule Account Tracing**
    - Tracks money movement between accounts.
    - Identifies **culprit ↔ victim ↔ mule network.**
    - Builds a transaction graph to highlight connections.
4. **AI Alerts + SOC Dashboard**
    - SOC team gets detailed alert with:
        - Account IDs
        - Time
        - Location

        - Risk Score
        - Suspected mule links
    - Helps faster investigation.
5. **Self-Learning Feedback Loop**
    - Every decision (Allow/Hold/Freeze) is fed back to the AI.
    - AI continuously improves fraud detection accuracy.

5. Differentiation from Existing Products

- Current FRM = **Rule-based + Reactive**.
- Our AI System = **Behavioral + Proactive + Auto Action**.
- Competitors mainly alert SOC, but our system can:
    - Auto-freeze funds instantly.
    - Map mule networks visually.
    - Provide real-time behavioral anomaly detection.

6. Use Case Example

- User A clicks on phishing link and shares OTP.
- Fraudster tries ₹90,000 transfer at 3:15 AM to mule account B.
- **AI detects anomaly** (time + amount unusual, mule account flagged).
- **System auto-freezes account** before money is withdrawn.
- SOC team alerted with full trace (A → B → Mule C → Culprit D).

7. Benefits

- Protects users from OTP scams, mule fraud, and UPI limit bypass.
- Prevents money loss by freezing funds instantly.
- Gives banks & NPCI more visibility on fraud networks.
- Builds trust in UPI system.

🔧 Tech Stack for AI-Based Suspicious Transaction Detection

1. Data Layer (Transaction Data Capture)

- **Sources**: NPCI transaction mirrors / Bank logs / PSP data (Google Pay, PhonePe, Paytm, etc.)
- **Tech**:
    - **Kafka / RabbitMQ** → real-time transaction streaming
    - **PostgreSQL / MongoDB** → structured + semi-structured data storage
    - **ElasticSearch** → fast searching (for trace module & SOC dashboard)

2. AI / ML Engine

- **Algorithms**:
    - **Anomaly Detection** → Isolation Forest, Autoencoders, One-Class SVM
    - **Behavioral Modeling** → LSTM/GRU (user transaction patterns over time)
    - **Graph Analysis** → Neo4j / NetworkX for mule account detection
- **Frameworks**:
    - **Python (Scikit-learn, TensorFlow, PyTorch)**
    - **Spark MLlib** (if large-scale, millions of txns per sec)

3. Fraud Rules Layer (Complementing AI)

- Predefined rules for quick checks:
    - ₹ > 1L in short span
    - Txns between 3–4 AM
    - Same device/IP multiple users
    - OTP phishing pattern
- **Tech**:
    - Drools / Open Policy Agent (OPA)
    - Redis (for super-fast in-memory rule checks)

4. Real-Time Decision Engine

- Yahan pe AI + Rules ka output combine hota hai:
    - If Risk Score > threshold → **Auto-Freeze**
    - Else if suspicious but not confirmed → **SOC Alert**
    - Else → **Allow**
- **Tech**:
    - Apache Flink / Spark Streaming (real-time processing)
    - REST APIs / gRPC for integration with NPCI + Banks

5. Account Freeze & Trace Module

- **Freeze API** → integrate with **Core Banking Systems (CBS)** to lock funds.
- **Trace**:
    - Build a **transaction graph** using **Neo4j** or **TigerGraph**.
    - Show **Culprit** → Victim → Mule Chains visually.

6. SOC Dashboard (Visualization Layer)

- Web App (for Bank SOC Teams)
    - **Frontend** → React.js / Angular
    - **Backend** → Node.js / Django / FastAPI
    - **Visualization** → D3.js, Grafana, Kibana
    - **Auth** → OAuth2 / Keycloak

SOC team can:✅ See alerts with details (account ID, IP, location, device, mule chain)

✅ Freeze account with 1 click✅ Override AI decision (allow/hold/freeze)

7. Feedback & Model Retraining

- Every decision (true fraud / false alarm) is stored.
- Weekly retraining pipeline:
    - **Airflow / Kubeflow** → automates retraining
    - **MLflow** → version control of models

🔄 Implementation Flow (Step by Step)

1. Transaction hits NPCI → **mirrored to our system** in real-time.
2. Data goes into **AI Engine + Rules Engine** simultaneously.
3. AI calculates **risk score** + mule account detection.
4. Decision Engine:
    - **High risk** → Freeze instantly
    - **Medium risk** → Alert SOC

- **Low risk** → Allow txn

5. SOC Dashboard shows trace + alert.
6. Feedback stored → model retrains → system smarter.

⚡ **Differentiator**:

- Most fraud systems = just rules + alerts.
- Tumhara = **AI + Behavioral patterns + Auto-freeze + Mule detection graph** → next-gen solution 🚀