## **Overview**

A full-stack intelligent platform to process, monitor, and detect fraudulent financial transactions in real-time using rule-based and ML-based approaches. It includes dashboards for analysts and role-based secured access for all users.

## **Tech Stack**

| **Layer** | **Stack Used** |
| --- | --- |
| Frontend | React (Vite) + Tailwind + ShadCN UI |
| Backend | Python (FastAPI) + Node.js (for payment APIs / stream handling) |
| Database | MongoDB (primary) + SQLite3 (local logs/cache) |
| ML Model | Graph-based detection with DBSCAN + Isolation Forest |
| Auth | JWT / OAuth2 + Role-Based Access Control (RBAC) + Secure OTP Login |

## **Auth System & Login**

* ✅ JWT based login
* ✅ Role-based access (admin, analyst, viewer)
* ✅ IP tracking and behavioral fingerprinting
* ✅ Device/IP/location risk scoring

## **Modules & Features**

### **1️⃣ Core Functional Features**

* Create/send transactions (mock/live mode)
* View history with filters (date, amount, status)
* Real-time transaction streaming
* Live status checks (success, failed, flagged)
* Razorpay integration
* Webhooks for payment status sync
* Retry mechanism for failed tx

### **Fraud Detection & ML**

* Rule-based checks (amount, frequency, velocity)
* ML scoring engine (Graph-based features)
* Risk scoring system (0–100)
* Risk categories: Low, Medium, High
* Auto-block on risk threshold breach

### **Analyst / Officer Dashboard**

* View flagged transactions
* Approve / Block / Investigate actions
* Mark false positives
* Notification bell UI (real-time alerts)
* Live alert feed with stream
* Action logging (audit trail)
* Signed logs (user + action + timestamp)

### **Security Layer**

* AES or ECC payload encryption
* JWT login mechanism
* Role-based access
* IP tracking for session integrity

### **User Marketplace**

A built-in transaction marketplace for users to either:

* **Create a transaction plan** (custom thresholds, rules, streaming behavior)
* **Buy pre-configured plans** (e.g., "Low Volume Plan", "High-Risk Monitor Plan")

#### **Features**

* User dashboard to manage plans
* Purchase plans via Razorpay

#### **UI Bits**

* Marketplace page (grid of plans)

## **ML Model Details (Graph-Based Anomaly Detection)**

### **📈 Overview**

The fraud detection engine uses a graph-based approach to capture hidden patterns in transaction metadata (sender, device, IP, session, etc.) and combines classical anomaly detection models to identify suspicious nodes.

### **Components**

#### **1. Graph Construction**

* Nodes: sender, device, ip\_address, product
* Edges: Represent relationships (e.g., used\_device, used\_ip, bought\_product)
* Sessions grouped using 30-minute gaps between sender actions

#### **2. Embedding Generation**

from node2vec import Node2Vec

* Generated using Node2Vec (random walks over the graph)
* Parameters: dimensions=64, walk\_length=10, num\_walks=100

#### **3. Feature Engineering**

* Session count per user
* Avg. transactions per session

### **🧪 Anomaly Detection Models**

#### **✅ Isolation Forest**

from sklearn.ensemble import IsolationForest

* Uses contamination-based outlier scoring
* Outputs risk scores + anomaly flags

#### **✅ DBSCAN**

from sklearn.cluster import DBSCAN

* Flags low-density (noise) nodes as anomalies
* Great for unsupervised, density-based filtering

### **📝 Output**

* anomalies\_iforest.txt and anomalies\_dbscan.txt files store flagged node IDs
* Scores returned for UI color-coding or further classification

## **DB Structure (MongoDB)**

**Collections:**

* users: RBAC roles, login info
* transactions: full tx record, flags, score
* logs: analyst actions
* alerts: high risk event