UNDERSTANDING AI ANTI-FRAUD AND AML SYSTEM

FinTech Compliance AI Gateway with Fraud and AML screening

Understanding Bliink AI's ecosystem

Introduction:

- **This** is an AI-powered financial transaction monitoring system designed to see **money laundering (AML)** and **fraudulent activities** are detected using both **rule-based** and **machine learning** methods.
- It processes **ISO 20022-like financial messages**, analyzes them, scores each transaction for risk, and provides **real-time dashboards** for investigators and compliance teams.

The goal of this prototype was to **understand and simulate** how enterprise-grade tools like **Bliink AI** products might operate, and to recreate a simplified, educational version of their core features.

Bliink AI Product	Feature to Cover in Prototype	Implementation Idea
Bliink AML Pro	Rule-Based Detection & Sanctions Screening	Build a simple **lookup table of "sanctioned accounts"** and flag matches. Add a basic **risk score** based on amount/transaction frequency.
Bliink Fraud Pro	AI/ML Model & ISO 20022 Compliance	Build a **small ML anomaly detection model** (Isolation Forest / Random Forest) on transaction data. Ensure parser handles ISO 20022 fields. Show **risk scores** and a simple **explainable feature importance chart**.

What to learn	
Non-technical aspect	Understand how the system simulates, detects, and visualizes risky transactions in plain language.
Technical aspect	Learn the full architecture, dependencies, data flow, and how to run, train, and extend the system.

🧩 **System Summary and Data Artifacts:**

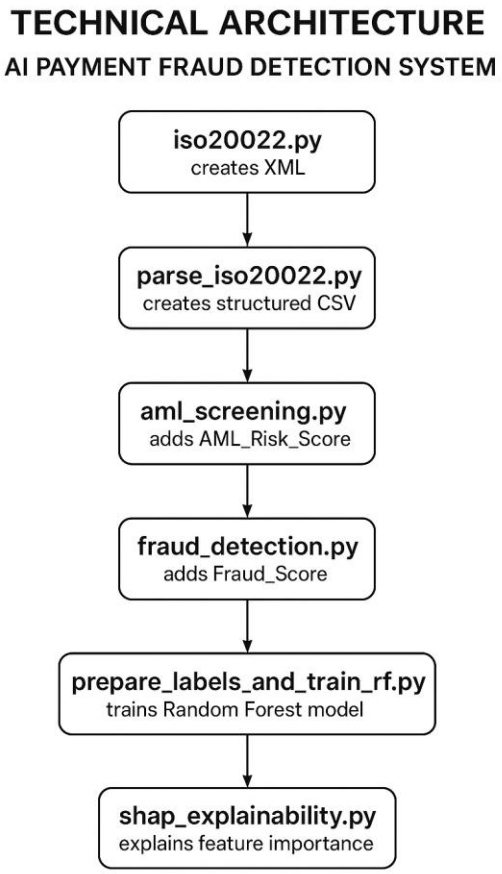
It's like a mini version of a full enterprise fraud detection system, built to learn the complete journey —> from data creation to AI-driven fraud insights in a way that is interactive and easy to understand.

Step	Module	Function
1️⃣	generate_iso20022.py	Generates fake ISO 20022 transaction XMLs
2️⃣	parse_iso20022.py	Converts XMLs into structured data (CSV / DataFrame)
3️⃣	aml_screening.py	Applies rule-based AML scoring and flags high-risk activity
4️⃣	fraud_detection.py	Detects outliers using an unsupervised ML model (Isolation Forest)
5️⃣	prepare_labels_and_train_rf.py	Trains a supervised Random Forest model for improved accuracy
6️⃣	streamlit_app.py	Interactive web dashboard for visualization and review
🧠	shap_explainability.py	Explains ML model decisions using SHAP feature importance

File	Description
transactions/*.xml	Synthetic ISO 20022 XML files
transactions_parsed.csv	Structured transaction dataset
transactions_aml_screened.csv	AML rule-based results
transactions_fraud_detected.csv	ML-based fraud detection results
rf_fraud_model.joblib	Saved supervised model
global_shap_importance.csv	Explainability summary
sanctioned_accounts.csv	Demo sanctions list

Conceptual Flow of the System:

1. **Simulate real banking transactions -** The system begins by creating sample payments between randomly generated bank accounts. Each payment includes realistic details, such as who sent it, who received it, the amount, currency, country, and reason for the transfer.
2. **Read and structure those payments -** These transactions are formatted using ISO 20022, the same international standard that real banks use. The system then reads and organizes this data into a structured table that's easier to analyze.
3. **Check for suspicious patterns -** Next, it performs basic rule-based checks. For example, flagging any transaction involving a blacklisted account, unusually large transfers, or high-risk keywords like *crypto*, *refund*, or *loan*. Each of these adds to a transaction's risk score.
4. **Machine learning for anomaly detection** - An **AI model** looks for transactions that **don't fit the normal pattern**, like outliers in behavior to spot activity that might be fraudulent or unusual.
5. **Supervised learning for accuracy** - A second model learns from previous results and refines the detection process, making future predictions more accurate and reducing false alarms.
6. **Explainability & Transparency -** Transparency is key. The system uses explainable AI (SHAP) to show why a transaction was marked as risky, highlighting which factors (amount, country, purpose, etc.) influenced that decision.
7. **Real-time dashboard -** Finally, all the results appear in a visual dashboard, where investigators and compliance teams can explore data, view alerts, and understand insights in real time.



Concept Exploration

I also explored how an advanced system that could help fintech companies automatically detect payment fraud in real time. How to continuously watch transactions as they happen, identify anything unusual or suspicious, and alert analysts right away. How to build a platform that would combine modern web tools, cloud services, and artificial intelligence to make fraud detection faster, more accurate, and easier to manage at scale.

In simpler terms, it is kind of a smart, automated security system for payments, one that uses AI to protect money transfers and give human analysts clear insights into what's happening and why.

This is simply a concept exploration of what a realistic, large-scale architecture for an AI-powered payment fraud detection system might look like. A possible blueprint that shows how all the tools (AI, microservices, event streaming, cloud infrastructure) can work together to detect fraud in real time.

- A web app (**frontend**) for customers to see dashboards and alerts — like a control center where analysts can watch what's happening.
- A main service (Spring Boot) that receives every payment, saves it to a secure database, and sends it out for checking.
- A messaging system (Kafka) that works like a fast conveyor belt, passing transactions in real time to other parts of the system.
- An AI engine (Python) that acts as the brain, analyzing transactions, comparing them with patterns of known fraud, and giving each one a "risk score." It also explains why something looks suspicious.
- Cloud infrastructure (AWS + Kubernetes) that keeps everything running smoothly, securely, and at scale — meaning it can handle thousands of transactions per second.