Fraud Detection Engine – Technical Rationale & Scaling Roadmap

1. Project Objective

- The goal of this project is to simulate real-time fraud detection in a fintech environment, demonstrating:
- Foundational fraud rules (rules-based detection)
- Event-driven architecture design principles
- A clear roadmap to production readiness and scalability

2. Tech Stack & Rationale

Backend

Node.js (Apollo GraphQL)

✅ Fast development, non-blocking I/O (handles transaction streams well).

✅ GraphQL ensures frontend queries fetch exactly what they need, avoiding over-fetching.

PostgreSQL

✅ ACID-compliant → guarantees data consistency (critical for financial systems).

✅ Rich support for JSON & indexing (helpful for fraud metadata).

In-memory cache / queue

✅ Super-fast reads/writes for generated & suspicious transactions.

Email Alerts (SendGrid)

✅ Shows alerting mechanism exists.

Frontend

React Dashboard

✅ Gives visibility & control → not just a backend service.

🚀 Could be extended with WebSockets for real-time alerts & dashboards with charts.

3. Scaling Strategy (Production Readiness)

Layer	Current (POC)	Production-Ready Scaling
Transactions Ingest	Batch insert from Node.js	Kafka → partitioned topics by user/region
Rules Engine	Hardcoded rules + one custom rule	ML scoring
Cache	In-memory array	Redis
Database	Single PostgreSQL instance	Sharding / Partitioning + Read Replicas
Alerts	Manual email trigger	Automated threshold-based alerting (Kafka consumer + Email/SMS service)
API	Monolithic Node.js GraphQL	Microservices: Ingest Service, Rule Service, Alert Service, Reporting Service
Infrastructure	Hosted on Render and Vercel (demo)	Kubernetes on AW

4. Future Functionalities

Event-driven architecture (Kafka)

- All services consume/produce events, ensuring loose coupling.
- Replay capability → investigate historical fraud at any time.

Real-time Alerts

- WebSockets + push notifications when thresholds exceeded.

Machine Learning Integration

- Anomaly detection models → combine rules + ML for hybrid detection.

User Behavior Analytics

- Track velocity (e.g. 5 failed attempts in 30 seconds).
- Location anomalies (sudden login from multiple countries).

Admin Control Panel

- Create / edit fraud rules dynamically via UI.

5. Security Considerations

- JWT + Role-Based Access Control (RBAC) for API
- Data encryption at rest
- Hashing sensitive values before storage (e.g., card numbers)
- Rate limiting at API gateway