FINFRAUD DOCUMENTATION v1.0.0 **Name: Kashmira Patil**

REPORT WITH MVP kashmiraspatil@gmail.com

---

1. Executive Summary

FinFraud is a deployable fintech solution that detects fraudulent transactions in real time using machine learning, ensures transparency with explainable AI, and guarantees integrity with a blockchain-backed immutable audit log. It also provides actionable recommendations for customers, analysts, and product teams.
The system is built with FastAPI, MySQL, React/Tailwind, Docker, and industry-grade ML models.

2. Introduction

Problem Statement

Financial fraud continues to rise with digital transactions, leading to massive losses and eroded trust. Traditional fraud detection systems are often opaque, centralized, and lack reliable audit trails.

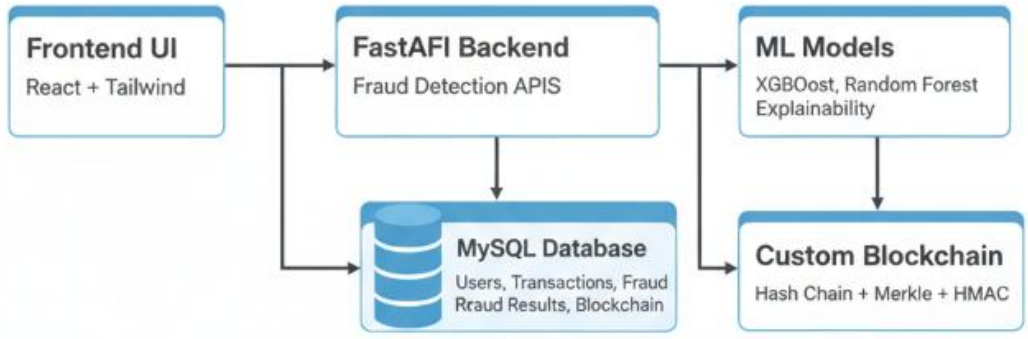
Objectives

- Real-time, accurate fraud detection.
- Explainable decisions with reason codes.
- Tamper-proof logging with blockchain.
- Actionable recommendations for multiple stakeholders.

3. System Architecture

The system integrates four major components:

1. FastAPI Backend – Fraud detection API, blockchain integration, DB persistence.
2. ML Models – XGBoost & Random Forest with explainability (SHAP).
3. Custom Blockchain Ledger – Immutable, hash-chained log with Merkle + HMAC.
4. React/Tailwind Frontend – Dashboard for fraud alerts, blockchain log, recommendations.

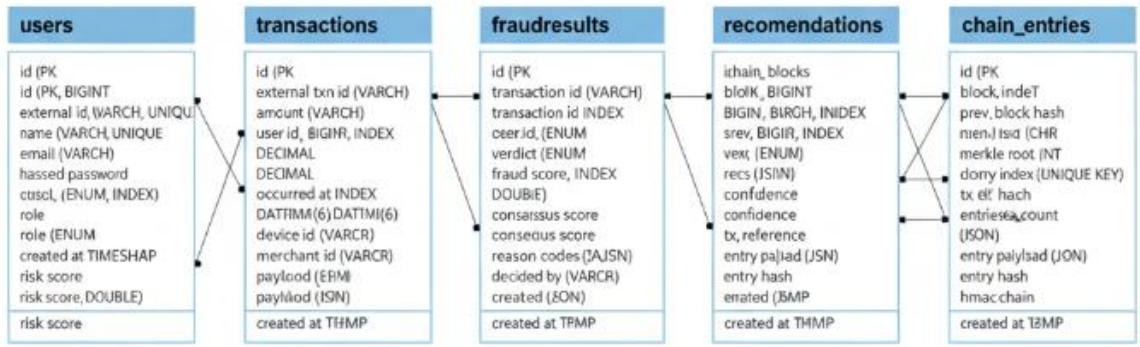


4. Database Design

MySQL stores structured data for users, transactions, fraud verdicts, recommendations, and blockchain logs.

Key Tables

- users – user details & roles.
- transactions – transaction data.
- fraudresults – fraud verdicts, fraud score, reason codes.
- chain_blocks – immutable blockchain blocks.
- chain_entries – detailed ledger entries.

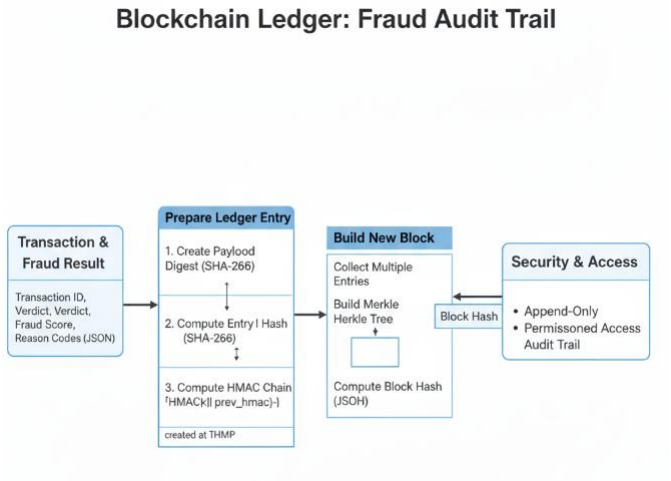


5. Fraud Detection Module

- Data Preprocessing – Handle imbalance (SMOTE), feature engineering.
- Models – XGBoost + Random Forest (trained on transaction datasets).
- Explainability – SHAP values + reason codes (e.g., “Unusual location”).
- Consensus – Multi-agent voting; fraud verdict committed if majority models agree.

6. Blockchain Ledger

- Design – SHA-256 hash chaining, Merkle root, HMAC chaining per entry.
- Purpose – Immutable, tamper-proof fraud audit trail.
- Security – Append-only, private ledger; permissioned access.
- Stored Data – Txn reference, verdict, fraud score, reason codes.



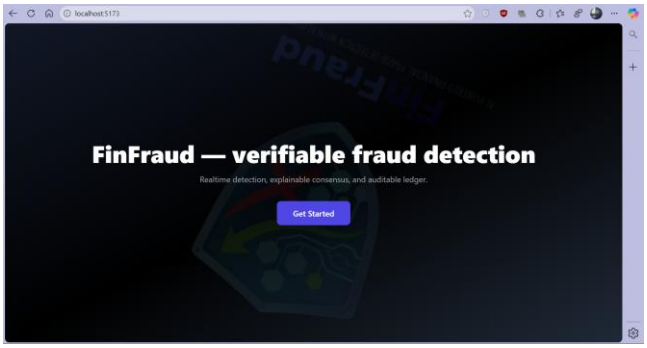
7. Recommendation Engine

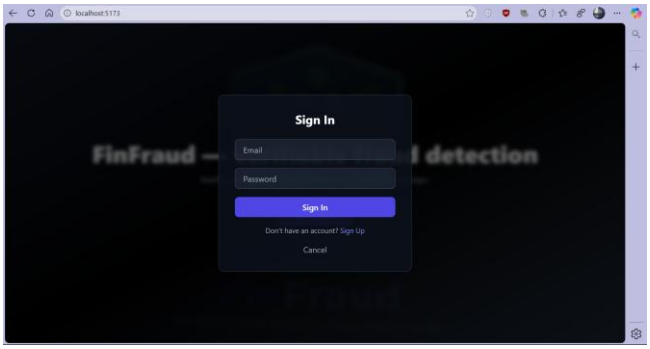
- For Customers – Safety tips (e.g., use secure devices, avoid risky regions).
- For Analysts – Ranked transaction review queues.
- For Product Teams – Suspicious user/merchant cluster analysis.

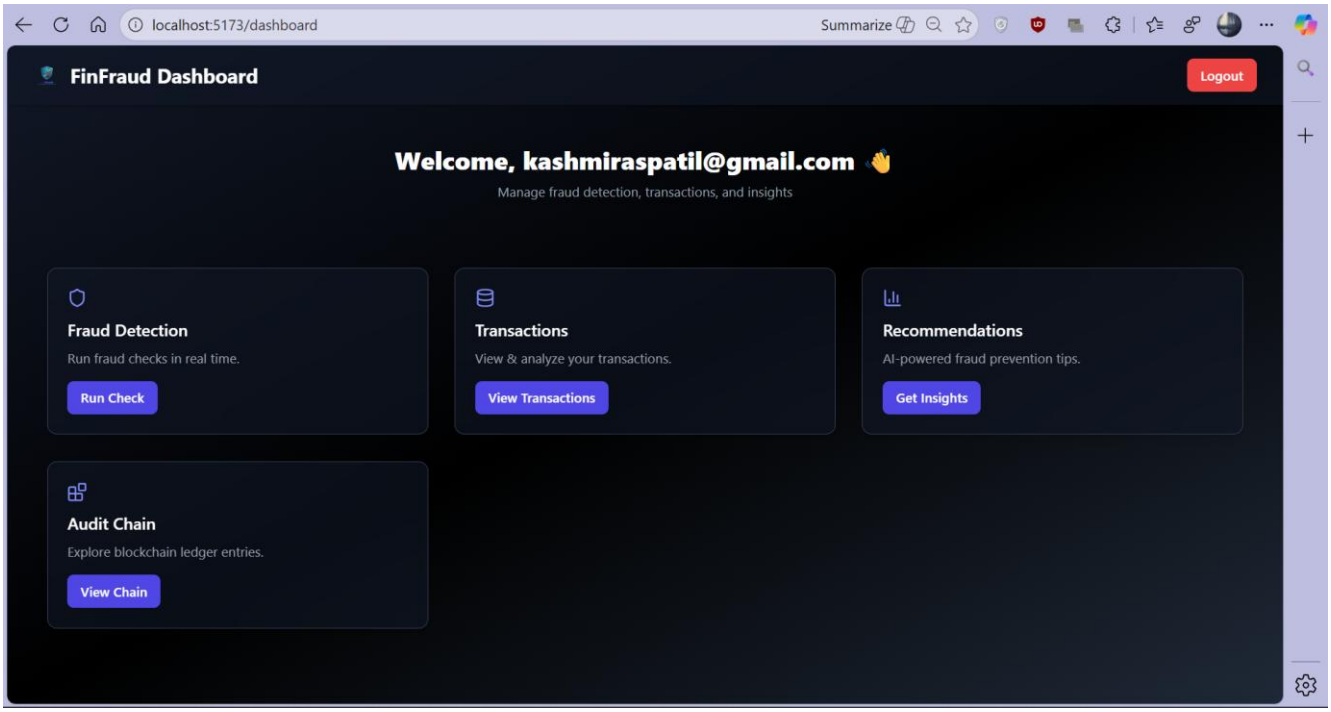
8. Frontend UI

Built with React + Tailwind, the dashboard includes:

- Transaction submission form.
- Fraud alert panel (status, fraud score, reason codes).
- Blockchain log viewer.
- Recommendation panel.







9. Deployment & Setup

Requirements

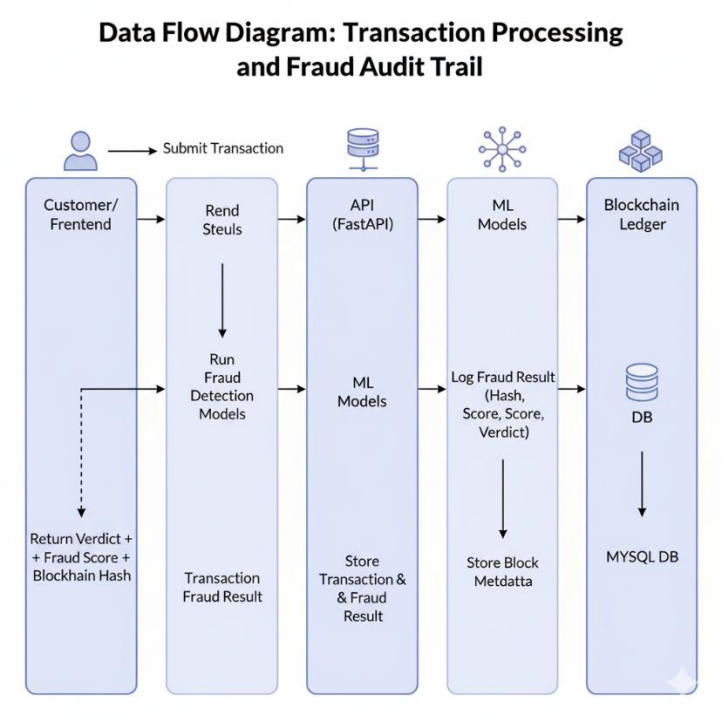
- Python 3.10+
- Docker & Docker Compose
- Node.js

```
# Start MySQL
docker compose up -d mysql
# Initialize schema
docker exec -i <mysql-container> mysql -uuser -pfraudpass frauddb <
database/schema.sql
# Backend setup
cd backend
```

```
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
uvicorn app.main:app --reload --port 8000
# Frontend setup
cd frontend
npm install && npm start
```

10. Testing & Validation

- Unit Tests – ML predictions, blockchain integrity.
- API Testing – Swagger UI, Postman.
- End-to-End Tests – Submit txn → fraud detection → blockchain log → UI visualization.



11. Security & Compliance

- Minimal PII stored on-chain (only hashes/reason codes).
- Strong encryption in DB & blockchain ledger.
- Role-based access control: Admin, Analyst, User.
- Transparent audit logs → compliance with fintech regulations.

12. Limitations & Future Work

- Current blockchain = single-node (future: multi-node PoA/BFT).
- Dataset = synthetic demo; production will need real banking datasets.
- Possible extension → Ethereum/Hyperledger for enterprise compliance.
- Real-time scalability → Kafka integration for high throughput.

13. References

- Stripe: *Fintech Fraud Detection Explained*
- Cossack Labs: *Audit Logs Security*
- IRJET: *Fraud Detection using Blockchain*
- FastAPI Best Practices (Zhanymkanov repo, Tiangolo docs)

Appendix A – API Endpoints

- GET /health → API & DB status.
- POST /transaction → Submit transaction, get fraud verdict + blockchain hash.
- GET /blockchain → Retrieve blockchain ledger.
- POST /recommend → Get recommendations.


```
{
    "transaction_id": "txn123",
    "user_id": "u001",
    "amount": 10000,
    "currency": "INR",
    "occurred_at": "2025-09-23T14:30:00Z",
    "location": "Mumbai",
    "device_id": "device123",
    "merchant_id": "m001"
}
```