ChatGPT

Project Title: Real-Time Financial Fraud Detection with Explainable AI Dashboard

---

Objective: Build a real-time fraud detection system that:

- Ingests streaming transaction data
- Predicts fraudulent activities with high recall
- Explains model predictions using SHAP
- Enables user feedback and model retraining
- Deploys on a scalable, containerized infrastructure

Dataset:

- **Name:** PaySim (Synthetic Financial Transactions)
- **Source:** Kaggle
- **Reason:** Realistic structure, labeled, temporal and transactional behavior present

---

System Architecture Overview:

```
Frontend (React + shadcn)
      |
FastAPI Backend (REST APIs)
      |
Real-Time Scoring (Kafka/Redis Consumer + ML Model)
      |
Kafka/Redis Stream (Simulated Transaction Feed)
      |
Data Producer (pandas to stream)
```

Tech Stack:

- **ML Models:** XGBoost, Autoencoder, GNN (optional), Temporal Transformer (optional)
- **Backend:** FastAPI, SHAP, Redis/Kafka
- **Frontend:** React.js, Tailwind CSS, shadcn/ui
- **Streaming:** Apache Kafka / Redis Streams
- **Explainability:** SHAP
- **Containerization:** Docker
- **Orchestration:** Kubernetes (Minikube -> AWS EKS)
- **MLOps:** MLflow, Evidently AI, GitHub Actions

Modeling & Evaluation:

- **XGBoost:** Tabular feature-based baseline
- **Autoencoder:** Unsupervised anomaly detection
- **Graph Neural Network:** User-to-user or user-to-merchant graph detection
- **Transformer:** Temporal sequential fraud patterns

Evaluation Metrics:

- Recall (critical)
- Precision, F1 Score
- PR-AUC (for imbalanced classes)

Feature Engineering:

- Delta features: balance_before - amount, etc.
- Time features: hour of day, recency
- Transaction type encoding
- Graph features (if GNN used)
- User behavior stats: avg. transaction size, frequency

Backend API:

- `POST /predict`: Returns fraud score + SHAP
- `POST /feedback`: Accepts user-labeled data
- `GET /stats`: Model + transaction summary metrics

Frontend Dashboard:

- Built with React + shadcn
- **Components:**
- Live Transaction Feed (color-coded fraud risk)
- Explainability Modal (SHAP plot)
- Feedback Buttons (False Positive/Negative)
- Admin Analytics (charts and summaries)
- Optional D3-based network graph for fraud rings

Deployment:

- **Docker:** Backend, frontend, inference service
- **Kubernetes:** Local (Minikube), cloud (EKS)
- **CI/CD:** GitHub Actions (build, test, deploy)
- **Monitoring:**
- MLflow for experiments
- Evidently AI for data drift
- Prometheus + Grafana (optional)

Folder Structure:

```
fraud-detection-project/
├── backend/
├── frontend/
├── streaming/
├── ml_pipeline/
│   ├── eda.py
│   ├── model_train.py
│   └── shap_explain.py
├── docker/
├── k8s/
└── README.md
```

Learning Outcomes:

- End-to-end ML pipeline development
- Real-time data engineering with Kafka
- SHAP-based explainability in production
- Full-stack ML system with FastAPI + React
- MLOps tools: MLflow, Docker, K8s, GitHub Actions
- Fraud-specific domain modeling and mitigation strategies

Next Steps:

1. Complete EDA (`eda.py`)
2. Perform feature engineering and model training
3. Build model inference and SHAP explanation API
4. Connect Kafka/Redis with FastAPI pipeline
5. Create React frontend with shadcn components
6. Containerize and orchestrate
7. Deploy with monitoring and feedback loop
