Fraud Transaction Detection System - Technical Specifications

1. Project Overview

1.1 Objective

Develop a real-time fraud detection system that classifies transactions as fraudulent or legitimate using machine learning techniques, with the ability to detect multiple fraud scenarios including amount-based, terminal-based, and customer behavior-based fraud patterns.

1.2 Business Value

- **Risk Mitigation**: Prevent financial losses from fraudulent transactions
- **Real-time Protection**: Block suspicious transactions before completion
- **Customer Trust**: Maintain customer confidence through secure transactions
- **Regulatory Compliance**: Meet financial industry fraud prevention requirements
- **Cost Reduction**: Minimize manual review processes and false positives

1.3 Fraud Scenarios to Detect

1. **Amount-based Fraud**: Transactions > $220 (baseline pattern)
2. **Terminal Compromise**: Fraudulent activity on compromised terminals (28-day windows)
3. **Card-not-Present Fraud**: Customer credential theft with inflated transaction amounts

2. System Architecture

2.1 High-Level Architecture

```
Transaction Stream → Feature Engineering → ML Model → Risk Scoring → Decision Engine →
Action
```

2.2 System Components

- **Data Ingestion Pipeline**: Real-time transaction processing
- **Feature Engineering Engine**: Creates behavioral and temporal features
- **Model Inference Service**: Fraud prediction and scoring
- **Risk Assessment Module**: Combines model outputs with business rules
- **Alert Management System**: Handles fraud notifications and case management
- **Model Management Platform**: Handles model updates and A/B testing

- **Monitoring Dashboard**: Real-time performance and fraud metrics

3. Data Specifications

3.1 Input Dataset Schema

Feature	Type	Description	Example Values
TRANSACTION_ID	String	Unique transaction identifier	"TXN_001234567"
TX_DATETIME	Datetime	Transaction timestamp	"2024-01-15 14:30:25"
CUSTOMER_ID	String	Customer identifier	"CUST_5678"
TERMINAL_ID	String	Terminal/merchant identifier	"TERM_9012"
TX_AMOUNT	Float	Transaction amount ($)	125.50
TX_FRAUD	Binary	Fraud label (target)	0 (legitimate), 1 (fraud)

3.2 Data Quality Requirements

- **Completeness**: No missing values in core fields
- **Timeliness**: Transaction data available within 100ms of occurrence
- **Accuracy**: Timestamps must be precise to the second
- **Consistency**: Customer/Terminal IDs follow consistent format

4. Feature Engineering Specifications

4.1 Temporal Features

- **Time-based**:
  - Hour of day, Day of week, Month
  - Is weekend/holiday transaction
  - Time since last transaction (customer/terminal)

4.2 Customer Behavioral Features

- **Spending Patterns** (rolling windows: 1, 7, 14, 30 days):
  - Average transaction amount
  - Standard deviation of amounts
  - Transaction frequency
  - Maximum transaction amount
  - Spending velocity changes

- **Customer Risk Indicators**:
  - Days since first transaction
  - Number of unique terminals used
  - Geographic diversity (if location data available)
  - Ratio of current amount to historical average

4.3 Terminal-based Features

- **Terminal Activity** (rolling windows: 1, 7, 28 days):
  - Transaction volume
  - Average transaction amount
  - Number of unique customers
  - Fraud rate (for model updates)
- **Terminal Risk Indicators**:
  - Terminal age (days since first transaction)
  - Anomalous activity patterns
  - Customer concentration ratio

4.4 Transaction-specific Features

- **Amount Features**:
  - Raw transaction amount
  - Amount rounded to nearest 10/100
  - Log-transformed amount
  - Amount percentile (customer/terminal/global)
- **Derived Features**:
  - Amount deviation from customer average
  - Amount deviation from terminal average
  - Transaction velocity (amount/time since last)

4.5 Aggregated Risk Features

- **Customer Risk Score**: Historical fraud indicators
- **Terminal Risk Score**: Terminal-based risk factors
- **Network Features**: Customer-terminal interaction patterns

5. Machine Learning Model Specifications

5.1 Model Architecture Options

5.1.1 Ensemble Approach (Recommended)

- **Level 1 Models**:
  - Gradient Boosting (XGBoost/LightGBM) - Primary
  - Random Forest - Secondary
  - Logistic Regression - Baseline
- **Level 2 Meta-learner**: Logistic Regression for final prediction

5.1.2 Alternative Single Models

- **Deep Learning**: Neural Network with embedding layers
- **Isolation Forest**: For anomaly detection
- **One-Class SVM**: For outlier detection

5.2 Model Evaluation Metrics

5.2.1 Primary Metrics

- **Precision**: Minimize false positives (legitimate transactions blocked)
- **Recall**: Minimize false negatives (fraudulent transactions missed)
- **F1-Score**: Balance between precision and recall
- **AUC-ROC**: Overall model discrimination ability

5.2.2 Business Metrics

- **False Positive Rate**: < 1% (customer experience)
- **True Positive Rate**: > 95% (fraud detection)
- **Cost-based Evaluation**: Weighted by fraud losses vs. investigation costs

5.3 Model Training Strategy

- **Temporal Split**: Train on historical data, validate on recent data
- **Cross-Validation**: Time-series split respecting temporal order
- **Class Imbalance**: Handle using SMOTE, class weights, or threshold optimization
- **Feature Selection**: Recursive feature elimination and importance analysis

6. Real-time Processing Requirements

6.1 Latency Requirements

- **Model Inference**: < 50ms
- **Feature Engineering**: < 30ms
- **Total Processing Time**: < 100ms per transaction

6.2 Throughput Requirements

- **Peak Load**: 10,000 transactions per second
- **Average Load**: 1,000 transactions per second
- **Scalability**: Auto-scaling based on transaction volume

6.3 Data Pipeline Architecture

```python
# Streaming Pipeline Structure
Transaction Event → Feature Store → Model Inference → Risk Decision → Response
```

7. Technical Implementation

7.1 Technology Stack

- **Programming Language**: Python 3.9+
- **ML Framework**: scikit-learn, XGBoost, LightGBM
- **Stream Processing**: Apache Kafka, Apache Flink
- **Feature Store**: Redis, Apache Cassandra
- **Model Serving**: MLflow, TensorFlow Serving
- **API Framework**: FastAPI
- **Monitoring**: Prometheus, Grafana
- **Containerization**: Docker, Kubernetes

7.2 Code Structure

```
fraud_detection_system/
├── data/
│   ├── raw/
│   ├── processed/
│   └── features/
├── src/
│   ├── data_pipeline/
│   │   ├── ingestion.py
│   │   ├── preprocessing.py
│   │   └── feature_engineering.py
│   ├── models/
│   │   ├── train_model.py
│   │   ├── ensemble_model.py
│   │   └── model_evaluation.py
│   ├── api/
│   │   ├── fraud_api.py
│   │   └── model_serving.py
│   ├── monitoring/
│   │   └── model_monitoring.py
│   └── utils/
│       └── data_utils.py
├── tests/
├── config/
├── models/
├── requirements.txt
└── README.md
```

8. API Specifications

8.1 Fraud Detection Endpoint

```python
# Real-time Fraud Detection API
POST /api/v1/fraud/detect
{
    "transaction_id": "TXN_001234567",
    "customer_id": "CUST_5678",
    "terminal_id": "TERM_9012",
    "tx_amount": 125.50,
    "tx_datetime": "2025-06-23T14:30:25Z"
}

Response:
{
    "transaction_id": "TXN_001234567",
    "fraud_probability": 0.23,
    "risk_score": 0.15,
    "decision": "APPROVE",
    "risk_factors": [
        "amount_deviation: 0.12",
        "customer_velocity: 0.08"
    ],
    "processing_time_ms": 45,
    "model_version": "v2.1.0",
    "timestamp": "2025-06-23T14:30:25.123Z"
}
```

8.2 Decision Thresholds

- **High Risk**: Probability > 0.8 → BLOCK
- **Medium Risk**: 0.3 < Probability ≤ 0.8 → REVIEW
- **Low Risk**: Probability ≤ 0.3 → APPROVE

9. Model Monitoring and Maintenance

9.1 Performance Monitoring

- **Model Drift Detection**: Feature distribution changes
- **Performance Degradation**: Track precision/recall over time
- **Data Quality Monitoring**: Missing values, outliers, schema changes
- **Latency Monitoring**: Response time tracking

9.2 Retraining Strategy

- **Scheduled Retraining**: Weekly model updates
- **Trigger-based Retraining**: Performance drop > 5%
- **Incremental Learning**: Online learning for rapid adaptation
- **A/B Testing**: Gradual rollout of new models

9.3 Alerting System

- **Critical Alerts**: Model failures, API downtime
- **Performance Alerts**: Degraded accuracy, increased latency
- **Business Alerts**: Fraud spike detection, unusual patterns

10. Security and Compliance

10.1 Data Security

- **Encryption**: Data at rest and in transit
- **Access Control**: Role-based permissions
- **Audit Logging**: Complete transaction trail
- **Data Retention**: Configurable retention policies

10.2 Regulatory Compliance

- **PCI DSS**: Payment card industry standards
- **GDPR**: Data privacy regulations
- **Fair Credit Reporting**: Model explainability
- **Anti-Money Laundering**: Transaction monitoring

11. Testing Strategy

11.1 Model Testing

- **Unit Tests**: Feature engineering functions
- **Integration Tests**: End-to-end pipeline testing
- **Performance Tests**: Load testing, latency validation
- **Shadow Testing**: Compare with existing system

11.2 Fraud Scenario Testing

- **Amount-based**: Transactions > $220 detection
- **Terminal Compromise**: Multi-day fraud pattern detection
- **Customer Behavior**: Spending anomaly detection
- **Edge Cases**: Boundary condition testing

12. Deployment Strategy

12.1 Infrastructure Requirements

- **Compute**: 8 CPU cores, 32GB RAM per instance
- **Storage**: 1TB SSD for feature store and models
- **Network**: High-bandwidth, low-latency connections
- **Redundancy**: Multi-region deployment for HA

12.2 Deployment Pipeline

```
Development → Staging → Canary → Production
```

- **Blue-Green Deployment**: Zero-downtime updates
- **Feature Flags**: Gradual feature rollout
- **Rollback Strategy**: Immediate rollback capability

13. Success Criteria

13.1 Technical KPIs

- **Model Performance**: F1-Score > 0.90
- **System Latency**: < 100ms 99th percentile
- **Availability**: 99.99% uptime
- **Throughput**: Handle peak loads without degradation

13.2 Business KPIs

- **Fraud Detection Rate**: > 95%
- **False Positive Rate**: < 1%
- **Cost Savings**: Measurable reduction in fraud losses
- **Customer Satisfaction**: Minimal impact on legitimate transactions

14. Risk Management

14.1 Technical Risks

- **Model Bias**: Regular bias testing and mitigation
- **Adversarial Attacks**: Fraud pattern evolution
- **System Failures**: Fallback to rule-based systems
- **Data Quality Issues**: Robust data validation

14.2 Business Risks

- **Regulatory Changes**: Compliance monitoring
- **Competitive Pressure**: Continuous improvement
- **Fraud Evolution**: Adaptive learning capabilities
- **Customer Impact**: Careful threshold management

15. Future Enhancements

15.1 Advanced Features

- **Graph Neural Networks**: Customer-merchant relationship analysis
- **Reinforcement Learning**: Adaptive decision making
- **Federated Learning**: Privacy-preserving model updates
- **Real-time Feature Engineering**: Stream processing optimization

15.2 Integration Opportunities

- **External Data Sources**: Credit bureaus, fraud databases
- **Multi-channel Analysis**: Cross-channel fraud patterns
- **Behavioral Biometrics**: User interaction patterns
- **Geographic Intelligence**: Location-based risk assessment