Java Based Banking Fraud Detection System

ML-Based Anomaly Detection with Real-Time Transaction Simulation

PRESENTED BY

Zainab Nisa J

Java Internship • Infosys SpringBoard

February 23, 2026

▶ Rule-Based Detection

▶ ML Anomaly Detection

Real-Time Simulation

Table of Contents

Introduction

In the digital banking era, **fraudulent transactions** pose significant threats to financial institutions and customers. Traditional rule-based systems struggle to detect **sophisticated fraud patterns** and **zero-day attacks**.
This project presents an **Intelligent Banking Fraud Detection System** that combines:

- **Rule-based detection** for known fraud patterns
- **Machine Learning models** for anomaly detection
- **Real-time transaction simulation** for testing
- **Interactive dashboards** for fraud monitoring
- **RESTful API gateway** for system integration

The system provides **real-time alerts**, **comprehensive reporting**, and **scalable deployment** for modern banking infrastructure.

Problem Statement

Financial institutions face increasing challenges in fraud detection:

- **Rising Fraud Complexity:** Fraudsters employ sophisticated techniques that evade traditional detection systems.
- **High False Positive Rates:** Rule-based systems generate excessive alerts, causing alert fatigue.
- **Delayed Detection:** Manual review processes result in delayed fraud identification.
- **Limited Adaptability:** Static rules cannot adapt to evolving fraud patterns.
- **Scalability Issues:** Legacy systems struggle with high transaction volumes.

Solution: A hybrid system combining **rule-based detection** with **ML-based anomaly detection**, integrated with **real-time simulation** and **comprehensive dashboards**.

Motivation

- **Financial Impact:** Global banking fraud losses exceeded $32 billion in 2024, emphasizing urgent need for advanced detection.
- **Customer Trust:** Effective fraud prevention is crucial for maintaining customer confidence in digital banking.
- **Regulatory Compliance:** Financial institutions must comply with PCI-DSS, GDPR, and local banking regulations.
- **AI Integration:** Machine learning enables detection of complex patterns invisible to traditional systems.
- **Real-time Response:** Immediate fraud detection prevents unauthorized transactions before completion.
- **Scalability Requirement:** Modern systems must handle millions of transactions daily while maintaining accuracy.

Objectives

Primary Objectives:

1. Develop a **hybrid fraud detection system** combining rule-based and ML approaches.
2. Create a **transaction simulation engine** for fraud scenario testing.
3. Implement **real-time anomaly detection** using machine learning models.
4. Design **interactive dashboards** for fraud trend visualization.
5. Build a **RESTful API gateway** for seamless system integration.

Technology Stack

Backend Development:

- **Programming Language:** Java (Spring Boot Framework)
- **Database:** MySQL
- **ORM:** Hibernate

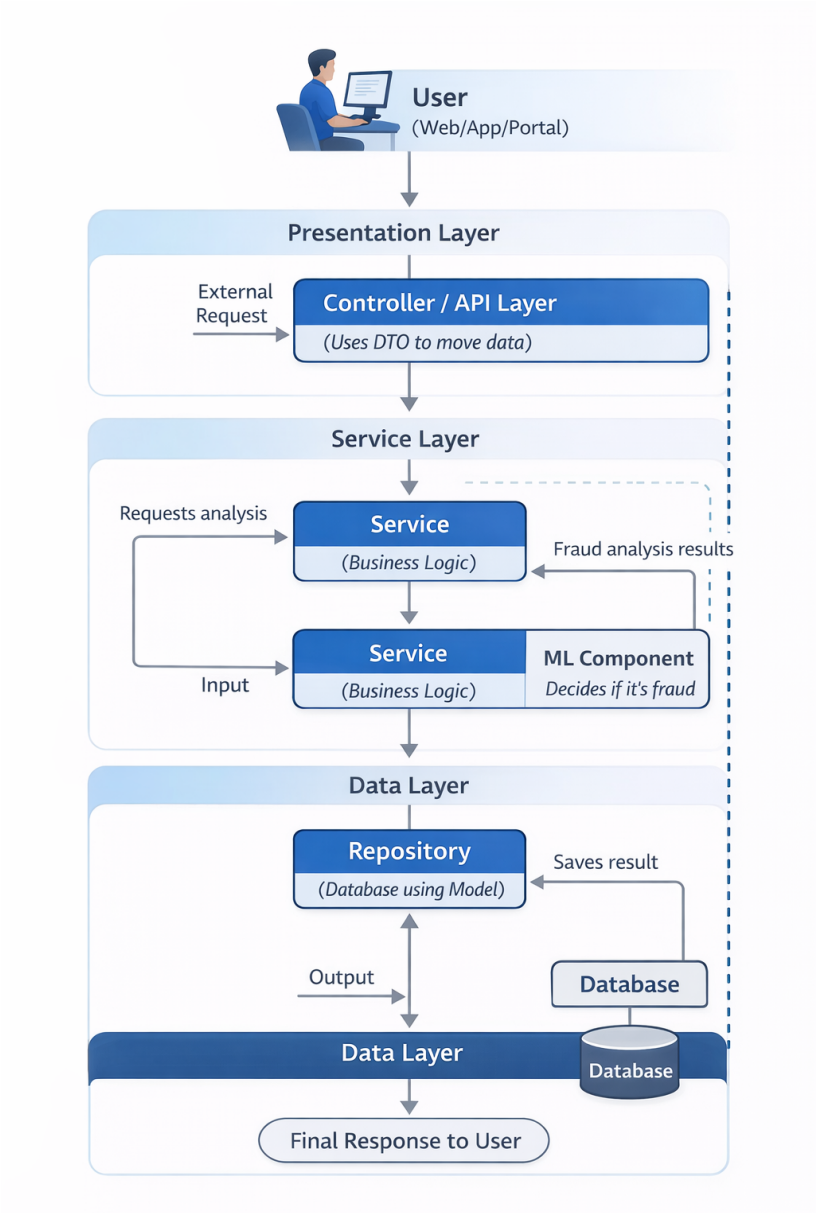
Frontend & Visualization:

- **Template Engine:** Thymeleaf

Build & Deployment:

- **Build Tool:** Maven

System Architecture



Module Explanation

1. **Data Modeling Module**
   - Transaction schema design with user profiles and fraud labels
   - Feature engineering pipeline (transaction patterns, velocity checks)
2. **Simulation Engine Module**
   - Generates synthetic transactions (normal and fraudulent)
   - Configurable fraud scenarios (card-not-present, account takeover)
3. **Rule-Based Detection Module**
   - Predefined rule sets (velocity, amount thresholds, geographic anomalies)
   - Real-time rule evaluation with configurable severity levels
4. **ML-Based Detection Module**
   - Trained models for anomaly scoring
   - Ensemble approach combining multiple algorithms

Module Explanation (Continued)

5. **Dashboard Module**
   - Real-time fraud alerts and transaction monitoring
   - Trend analysis and pattern visualization
   - Configurable alert thresholds and notifications
6. **API Gateway Module**
   - RESTful endpoints for transaction submission
   - Webhook support for fraud alerts
   - Authentication and rate limiting
7. **Reporting Module**
   - Automated daily/weekly fraud reports
   - Compliance audit trail generation
   - Performance metrics and KPI tracking
8. **Database Management Module**
   - Efficient indexing for fast query performance

Simulation Module

The **Transaction Simulation Engine** enables comprehensive testing of fraud detection capabilities before production deployment.

Key Features:

- **Normal Transaction Generation:** Creates realistic transaction patterns based on user behavior models
- **Fraud Scenario Simulation:** Generates various fraud types:
  - Card testing (multiple small transactions)
  - Account takeover (sudden pattern change)
  - Geographic anomalies (impossible travel)
  - High-value fraud (unusual amount patterns)
- **Configurable Parameters:** Transaction volume, fraud ratio, time windows
- **Real-time Injection:** Simulated transactions flow through detection pipeline
- **Performance Testing:** Validates system scalability under load

Execution Flow

Transaction Processing Pipeline

1. **Transaction Ingestion**
   - API receives transaction data (amount, merchant, location, time)
   - Initial validation and schema verification
2. **Feature Extraction**
   - Calculate transaction velocity (transactions per hour)
   - Compute statistical features (deviation from user average)
   - Geographic and temporal features
3. **Rule-Based Evaluation**
   - Apply predefined rules sequentially
   - Generate alert if rules triggered
   - Assign risk score based on rule severity
4. **ML-Based Scoring**
   - Pass features to trained ML models
   - Anomaly score calculation
   - Ensemble prediction aggregation

Execution Flow (Continued)

5. **Decision Making**
   - Combine rule-based and ML scores
   - Apply decision threshold (configurable)
   - Classify as: FRAUD/NORMAL
6. **Dashboard Update**
   - Real-time metrics update (fraud rate, alert count)
   - Visualization refresh
   - Store in database for historical analysis

Project Milestones

Milestone 1: Weeks 1–2 — Setup & Data Modeling

- **Objective:** Establish project environment and transaction models
- **Tasks:** Configure APIs, design fraud simulation datasets, define rule sets
- **Evaluation (Week 2):** Data models created; APIs configured; fraud scenarios defined

Milestone 2: Weeks 3–4 — Simulation Engine & Detection Core

- **Objective:** Build transaction generator and rule-based anomaly detector
- **Tasks:** Develop fraud scenarios, test detection accuracy with rules
- **Evaluation (Week 4):** Simulation and detection engine functional with rule-based alerts

Project Milestones (Continued)

Milestone 3: Weeks 5–6 — Dashboards & API Gateway

- **Objective:** Visualize fraud trends and enable API integration
- **Tasks:** Implement dashboards, connect APIs, test reporting functions
- **Evaluation (Week 6):** Dashboards and APIs operational; reports accurate

Milestone 4: Weeks 7–8 — ML Plug-in & Deployment

- **Objective:** Integrate ML-based detection and finalize system
- **Tasks:** Plug in ML models, run accuracy benchmarks, deploy fraud simulator
- **Evaluation (Week 8):** ML plug-in integrated; system deployed; anomaly detection effective

Algorithms

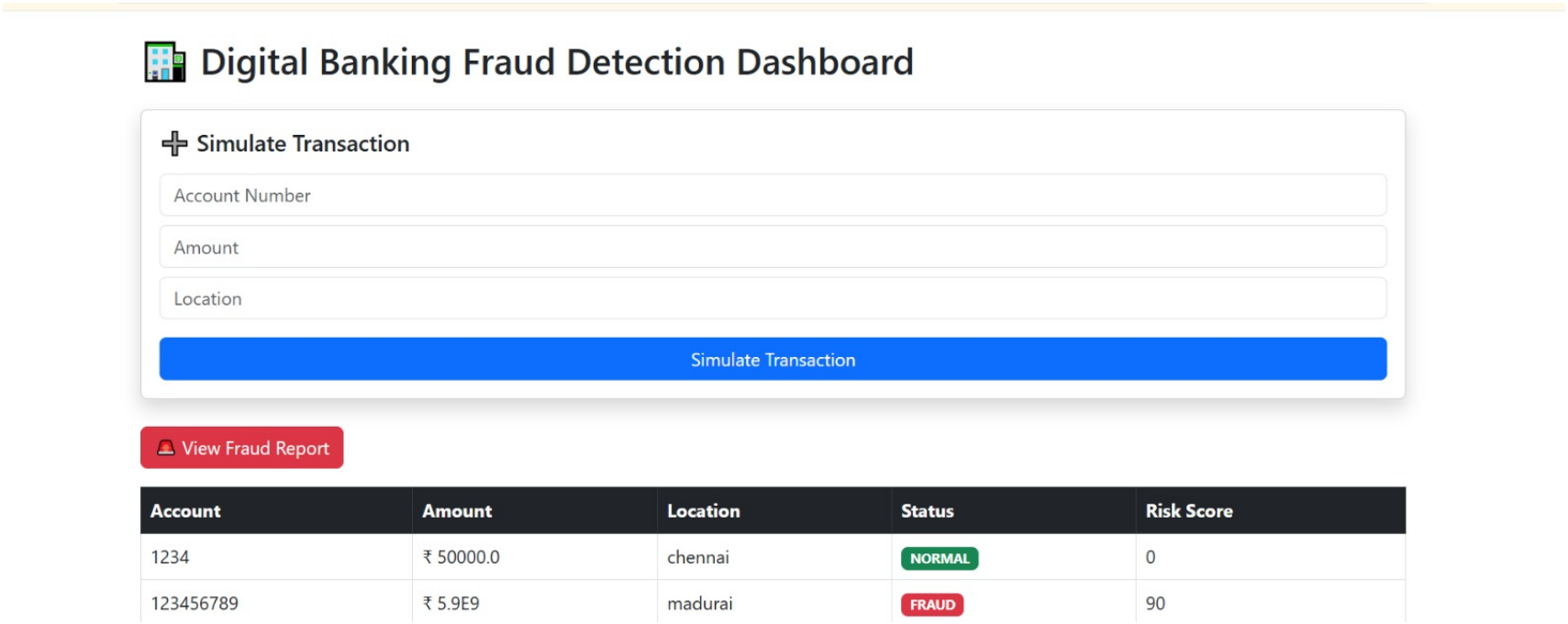
1. Rule-Based Detection Algorithm

1. Input transaction T with features ($amount$, $location$, $time$, $merchant$)
2. For each rule R in rule set:
   - Evaluate condition: if $R(T) = \text{TRUE}$, increment risk score
3. If total risk score $>$ threshold θ_r, flag as suspicious
4. Generate alert with triggered rules and severity
5. Output: decision ($APPROVED|REVIEW|DECLINED$) and risk score

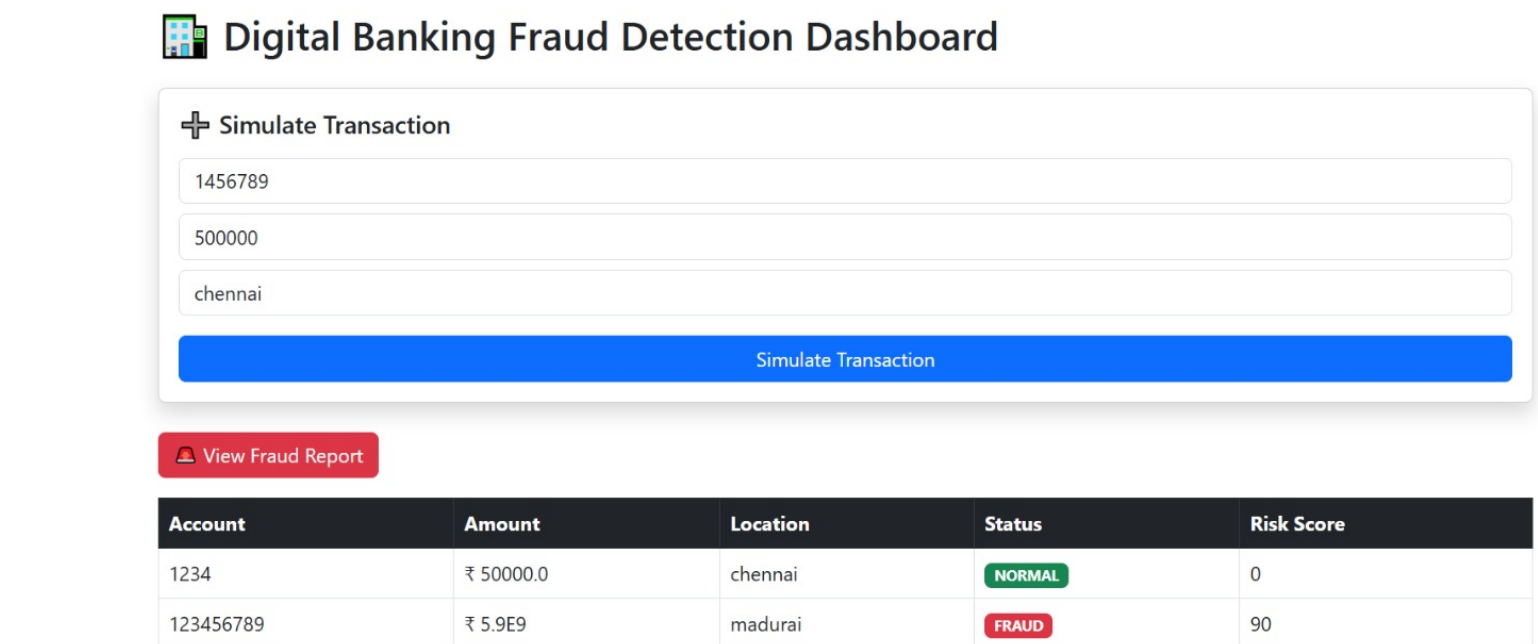
Example Rules:

- Transaction amount $>$ \$5000 (high-value rule)
- Velocity $>$ 5 transactions in 10 minutes (rapid fire)
- Geographic distance $>$ 500km from last transaction within 1 hour (impossible travel)

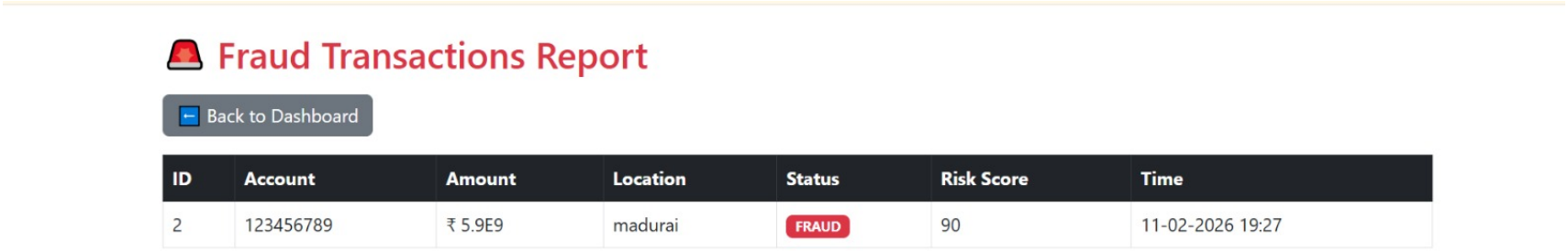
Implementation Screenshot 1



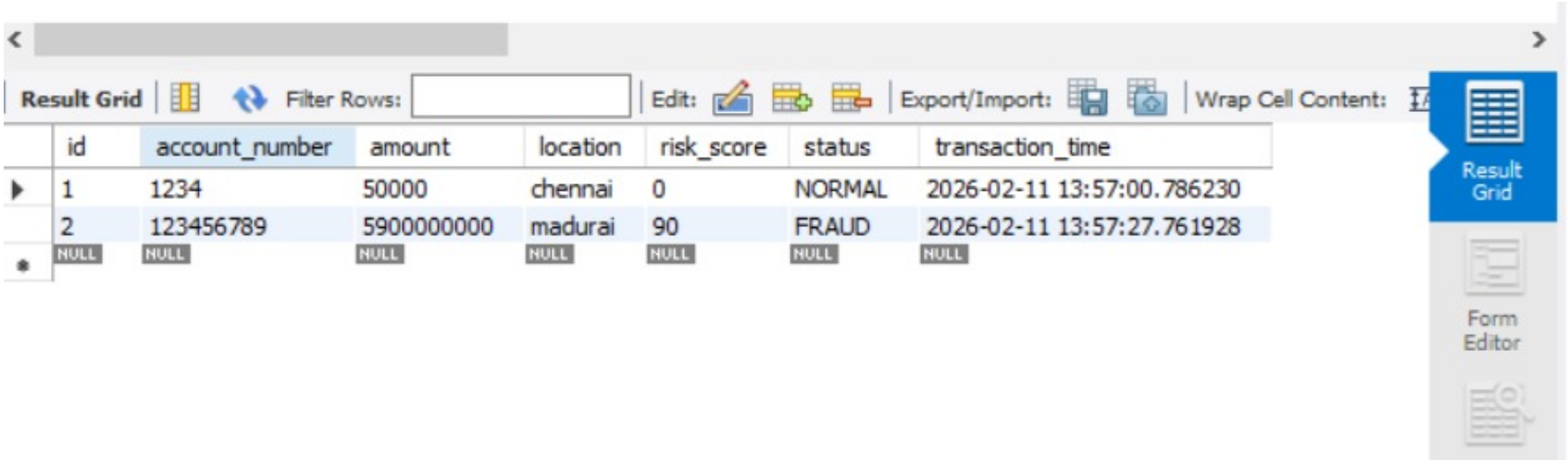
Implementation Screenshot 2



Implementation Screenshot 3



Implementation Screenshot 4



Advanced Discussion

1. **Model Interpretability**
   - SHAP values reveal feature importance for individual predictions
   - Helps fraud analysts understand why transactions were flagged
   - Enables rule refinement based on ML insights

2. **Handling Imbalanced Data**
   - Fraud typically $< 1\%$ of transactions (severe class imbalance)
   - Applied SMOTE for synthetic minority oversampling
   - Class weights adjusted in loss function
   - Ensemble methods reduce bias toward majority class

3. **Real-time Streaming Architecture**
   - Apache Kafka ensures message durability and ordering
   - Micro-batching for efficient ML inference
   - Stateful processing for velocity calculations

Advanced Discussion (Continued)

4. **Feature Engineering Techniques**
   - **Temporal Features:** Time since last transaction, day of week, hour of day
   - **Aggregate Features:** Rolling averages, standard deviations over time windows
   - **Categorical Encoding:** One-hot encoding for merchant categories
   - **Geographic Features:** Distance from home location, country risk scores

5. **Model Deployment Strategy**
   - A/B testing for new model versions
   - Shadow mode deployment (parallel evaluation without affecting decisions)
   - Gradual rollout with monitoring
   - Automated rollback on performance degradation

6. **Challenges & Solutions**
   - **Challenge:** Concept drift (fraud patterns evolve)
   - **Solution:** Scheduled model retraining (weekly) with recent data

Challenges and Limitations

Technical Challenges:

- **Data Quality:** Incomplete or noisy transaction data affects model accuracy
- **Concept Drift:** Fraud patterns evolve, requiring continuous model updates
- **Real-time Constraints:** Balancing detection accuracy with low latency
- **Scalability:** Handling peak transaction loads during high-traffic periods

Operational Limitations:

- **False Positives:** Legitimate transactions flagged cause customer friction
- **Model Bias:** Risk of discriminatory patterns in training data
- **Adversarial Attacks:** Sophisticated fraudsters may learn to evade detection
- **Regulatory Compliance:** Must explain automated decisions (GDPR, fair lending)

Mitigation Strategies: Regular audits, human-in-the-loop review, explainable AI, continuous monitoring

Conclusion

The **Intelligent Banking Fraud Detection System** successfully demonstrates a **hybrid approach** combining rule-based and ML-based detection methods.

Key Achievements:

- Created comprehensive **simulation environment** for scenario testing
- Developed **interactive dashboards** for fraud trend analysis
- Built **scalable API gateway** for system integration

Impact: The system enhances fraud prevention capabilities, reduces financial losses, and improves customer trust through rapid detection and response.

Limitations: Performance depends on data quality, requires ongoing model maintenance, and faces challenges with evolving fraud tactics.

Future Work

The system can be further enhanced through:

- **Deep Learning Integration:** Implement LSTM/GRU networks for sequential pattern detection across transaction histories.
- **Graph-Based Analysis:** Use graph neural networks to detect fraud rings and network-based patterns.
- **Federated Learning:** Enable collaborative model training across institutions without sharing sensitive data.
- **Explainable AI:** Expand LIME/SHAP integration for better decision transparency and regulatory compliance.
- **Mobile Application:** Develop mobile app for real-time fraud alerts and customer verification.
- **Blockchain Integration:** Explore distributed ledger for immutable fraud event logging.
- **Advanced Simulation:** Add adversarial attack simulation to test system robustness.
- **Multi-language Support:** Extend dashboard and reporting to multiple languages.

Project Deliverables

As per Infosys SpringBoard internship requirements, the following two deliverables are submitted at project completion:

Deliverable 1: GitHub Repository

- Contains complete project source code (Backend + Frontend + supporting files)
- Includes **MIT License** file in the root of the repository
- Repository is publicly accessible for review and evaluation

Deliverable 2: Agile Documentation

- Documents project execution in **dated format** for each milestone week
- Records tasks completed, outputs produced, and evaluation criteria met per sprint
- Included **inside the GitHub repository** as `AGILE_DOCS.md`
- Covers all 4 milestones (Weeks 1–2, 3–4, 5–6, 7–8)

References

1. Dal Pozzolo, A., et al. (2024). "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," IEEE Transactions on Neural Networks.
2. Bahnsen, A. C., et al. (2025). "Feature Engineering Strategies for Credit Card Fraud Detection," Expert Systems with Applications.
3. Wang, Y., et al. (2024). "Real-time Fraud Detection in Financial Transactions using Stream Processing," ACM Computing Surveys.
4. Liu, F. T., et al. (2023). "Isolation Forest for Anomaly Detection," IEEE Transactions on Knowledge and Data Engineering.
5. Chen, T., & Guestrin, C. (2024). "XGBoost: A Scalable Tree Boosting System," Proceedings of ACM SIGKDD.

References (Continued)

6. Bhattacharyya, S., et al. (2025). "Data Mining for Credit Card Fraud: A Comparative Study," Decision Support Systems.
7. Jurgovsky, J., et al. (2024). "Sequence Classification for Credit Card Fraud Detection," Expert Systems with Applications.
8. Carcillo, F., et al. (2023). "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection," Information Fusion.
9. Randhawa, K., et al. (2024). "Credit Card Fraud Detection using AdaBoost and Majority Voting," IEEE Access.
10. Alowais, S. A., et al. (2025). "Revolutionizing Banking: The Role of Artificial Intelligence in Fraud Detection," Journal of Financial Technology.