CHAPTER 01

INTRODUCTION

The Unified Payments Interface (UPI) has transformed India's digital payment ecosystem by enabling instant, secure, and user-friendly transactions across banks, merchants, and individuals. Its widespread adoption has accelerated financial inclusion and reshaped the way financial interactions occur in the digital economy. However, the rapid growth of UPI has also introduced vulnerabilities, with fraudulent activities such as phishing, unauthorized access, and anomalous transaction patterns posing serious risks to user trust and financial stability. Detecting and mitigating fraud in real time has therefore become a critical requirement for safeguarding the integrity of UPI transactions.

The UPI Fraud Detection System is designed to address these challenges by combining machine learning techniques with secure system design. Transaction data is ingested, preprocessed, and enriched through contextual and graph-based reputation features to capture suspicious behavioral patterns. Fraud scoring is achieved using supervised learning models—most notably Random Forest, which has demonstrated superior accuracy compared to Logistic Regression, SVM, and Decision Trees—as well as anomaly detection through IsolationForest. Together, these approaches enable precise identification of fraudulent transactions while minimizing false positives.

In addition to detection, the system emphasizes privacy, security, and transparency. Sensitive UPI identifiers are deterministically tokenized and encrypted at rest, ensuring confidentiality of transaction data. Authentication mechanisms integrate password hashing, Time-based One-Time Password (TOTP) multi-factor authentication, and WebAuthn (FIDO2) protocols to strengthen account security. Explainability hooks provide per-transaction insights, allowing stakeholders to understand why a transaction was flagged and adapt to evolving fraud strategies.

The framework is lightweight yet scalable, supporting asynchronous processing through Celery/Redis while maintaining synchronous fallbacks for developer-friendly experimentation. This balance of simplicity and robustness makes it suitable for exploration, evaluation, and incremental hardening toward production deployment.

CHAPTER 02

LITERATURE REVIEW

Jagadeesan et al. (2025) emphasize that detecting fraud in UPI transactions requires a combination of supervised, unsupervised, and semi-supervised machine learning approaches to capture unusual transaction patterns. They argue that effective feature engineering and anomaly detection are central to improving accuracy, and their survey of prior work supports this claim. Lakshmi et al. (2019) analyzed UPI-based mobile banking applications and recommended stronger encryption and multi-factor authentication to mitigate risks, while Mohapatra et al. (2017) examined UPI's role in India's transition toward a "less cash" society, stressing the importance of user trust and transparency in preventing fraud. Kumar et al. (2020) provided a technical security analysis of UPI and payment apps, identifying weaknesses in authentication flows and advocating for adaptive learning and anomaly detection strategies. Together, these studies reinforce Jagadeesan et al.'s argument that effective UPI fraud detection must integrate robust machine learning models, anomaly-based monitoring, and secure authentication mechanisms to ensure resilience against evolving threats.

Rani, Alam, and Javed (2024) explored the application of advanced machine learning techniques for fraud detection in UPI transactions, focusing particularly on the XGBoost algorithm due to its efficiency in handling complex and imbalanced datasets. Their work highlights the importance of preprocessing methods such as the Synthetic Minority Oversampling Technique (SMOTE) to balance fraudulent and legitimate transaction data, and Principal Component Analysis (PCA) to reduce dimensionality while retaining critical information. In their survey of prior research, Ibrahim (2023) demonstrated the effectiveness of Random Forest and Decision Tree models but noted performance issues with imbalanced datasets, while Boulieris (2023) highlighted the role of anomaly detection and NLP-based features in improving fraud detection metrics. Baesens (2021) argued that intelligent feature engineering can often outperform complex analytical techniques, showing that careful data design is as important as algorithmic sophistication. By synthesizing these perspectives, Rani, Alam, and Javed underline that robust fraud detection requires not only powerful algorithms like XGBoost but also careful data balancing, feature extraction, and continuous model retraining to adapt to evolving fraud strategies. Their contribution lies in demonstrating that ensemble learning, supported by preprocessing and dimensionality reduction, can achieve high accuracy in real-time fraud detection systems.

Li et al. (2024) advanced the field of fraud detection with SEFraud, a self-explainable framework that jointly learns detection and interpretable masks inside a heterogeneous graph transformer. By integrating mask learning and a task-aware triplet loss, SEFraud produces compact, human-readable explanations for each alert while improving detection performance on large industrial datasets, including deployment at ICBC. This work influenced the emphasis on built-in explainability rather than post-hoc methods, encouraging the adoption of per-transaction

explanation hooks and lightweight feature importance artifacts so investigators can quickly inspect why a transaction was flagged. The integration of explainability into fraud detection systems ensures transparency and builds user trust, which is critical in financial environments where false positives can undermine confidence.

Zhuo et al. (2024) addressed heterophily and label imbalance challenges in fraud graphs through Partitioning Message Passing (PMP). Their approach partitions neighbor aggregation and uses distinct node-specific aggregation functions for homophilic versus heterophilic neighbors. The authors provided theoretical grounding through spectral analysis and demonstrated substantial gains over naïve graph neural network message-passing on fraud benchmarks. This work directly informed the treatment of graph-derived features in fraud detection prototypes, particularly the decision to avoid naïve neighbor-averaging. Instead, reputation and connectivity metrics were computed, with future iterations designed to integrate heterophily-aware GNN layers to better capture fraud signals buried among benign neighbors. By tackling the relational complexity of fraud detection, PMP contributes to more accurate and resilient systems.

Taken together, these studies illustrate the evolution of fraud detection in UPI and digital payments. Jagadeesan et al. highlight the foundational role of supervised and anomaly-based learning, while Rani, Alam, and Javed demonstrate the effectiveness of ensemble learning combined with SMOTE and PCA for handling imbalanced datasets. Li et al. advance the field with explainable graph-based detection, ensuring transparency in fraud alerts, and Zhuo et al. tackle heterophily in fraud graphs, improving relational modeling and detection performance. This literature review shows that effective UPI fraud detection requires a multi-layered approach: robust machine learning models for transaction-level analysis, preprocessing and dimensionality reduction for data quality, and graph-based frameworks for relational fraud patterns. The integration of explainability and adaptive learning strategies ensures that fraud detection systems remain resilient, transparent, and trustworthy in the face of evolving threats.

CHAPTER 03

OBJECTIVES

This work focuses on building a secure and intelligent fraud detection system for UPI transactions. It applies machine learning and anomaly detection to identify suspicious activities in real time. The system is designed to ensure transaction security, user trust, and scalability in digital payments.

1. Securely ingest and store UPI transactions

- Build a reliable ingestion pipeline with deterministic tokenization and field-level encryption.
- Protect sensitive identifiers (e.g., account numbers, VPAs) while ensuring compliance with privacy and security standards.
- Guarantee scalability and resilience to handle high transaction volumes without compromising data integrity.

2. Detect and explain fraudulent or anomalous transactions

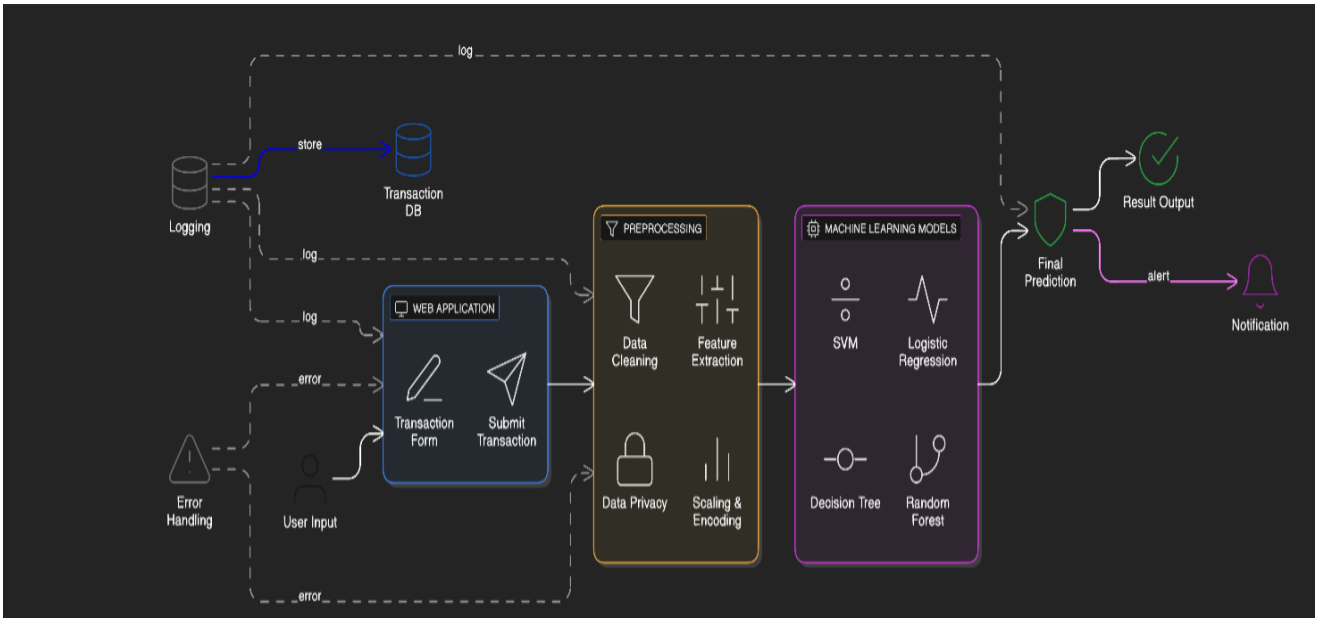
- Integrate graph-derived features, supervised models, and IsolationForest anomaly scoring.
- Provide per-transaction explainability hooks and lightweight feature importance artifacts for transparency.
- Enable investigators and institutions to understand why alerts are triggered, improving trust and accountability.

3. Provide secure, extensible operations and developer ergonomics

- Implement strong authentication (passwords, TOTP, WebAuthn) and reliable synchronous/asynchronous processing scaffolds.
- Use Celery + Redis fallback, testing frameworks, and clear documentation for production readiness.
- Ensure long-term scalability, adaptability to new fraud patterns, and ease of use for developers.

CHAPTER 04

METHODOLOGY



The system is designed to identify fraudulent UPI transactions by combining user input, data processing, and machine learning techniques in a structured workflow.

1. User Input and Web Application

The process begins when a user enters transaction details through a web-based application. The form collects essential information such as payer and payee details, transaction amount, time, and category. Once submitted, the data is validated to ensure correctness before further processing.

2. Transaction Storage and Logging

Submitted transactions are stored in a transaction database for record-keeping and analysis. At the same time, system activities such as successful submissions, processing steps, and system events are logged. This logging mechanism helps track system behavior and supports monitoring and debugging.

3. Error Handling

If invalid inputs or system errors occur during submission or processing, they are captured by the error-handling module. This ensures that incorrect data does not move forward in the pipeline and helps maintain system reliability.

4. Data Preprocessing

Before applying machine learning models, the transaction data is prepared through preprocessing. This includes cleaning the data to remove inconsistencies, extracting relevant features, and converting raw values into formats suitable for model input. Data scaling and encoding are also applied to ensure uniformity. Privacy measures are integrated at this stage to protect sensitive transaction information.

5. Machine Learning Models

The processed data is passed to multiple machine learning models, including Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest. These models analyze transaction patterns and learn to distinguish between normal and fraudulent behavior. Running multiple models improves prediction reliability.

6. Prediction and Decision Making

The outputs from the models are used to generate a final prediction indicating whether a transaction is fraudulent or not. A confidence score is also calculated to show how certain the system is about its decision.

7. Result Output and Notification

The final prediction is displayed to the user in a clear and understandable format. If fraud is detected, an alert or notification is generated to ensure timely action. Normal transactions are confirmed without interruption.

CHAPTER 05

DESIGN TECHNIQUES

The system is designed using a structured and secure approach to ensure accuracy, reliability, and scalability.

- **Modular Layered Architecture**

  The system follows a layered design where data flows through input, preprocessing, feature handling, model analysis, and output layers. Each layer works independently, making the system easy to maintain, test, and upgrade without affecting other components.

- **Privacy-by-Design Approach**

  Sensitive user information is protected using tokenization instead of storing raw identifiers. Only required data is processed, reducing privacy risks and ensuring safe handling of transaction information.

- **Data Security and Encryption**

  Critical transaction fields are encrypted before storage to prevent unauthorized access. Security keys are managed externally using environment configurations, improving protection and key management practices.

- **Hybrid Machine Learning Models**

  Multiple machine learning models are used together to improve fraud detection accuracy. Supervised models identify known fraud patterns, while anomaly detection models detect unusual or new transaction behaviors.

- **Graph-Based Feature Analysis**

  Transactions are represented as graphs to capture relationships between users. This helps identify coordinated or repeated fraudulent activities that may not be detected through individual transactions.

- **Explainable Predictions**

  The system provides feature-based explanations for each prediction. This helps users and

analysts understand why a transaction was flagged, supporting transparency and informed decision-making.

- **Fault Tolerance and Developer Support**

  The system supports both synchronous and asynchronous processing to ensure smooth operation even if some services fail. Developer-friendly tools and scripts simplify testing and system setup.

- **Test-Driven and Reliable Design**
  Automated testing is used to validate core functionalities such as security, data processing, and model execution. This ensures stability and reduces system errors.

- **Operational Readiness and Monitoring**
  The design supports deployment using containers and includes logging and health monitoring features. These help track system performance and prepare the system for real-world usage.

CHAPTER 06

EXECUTION

This chapter gives the detailed execution and the implementation details for the predictionof the upi fraud detection.

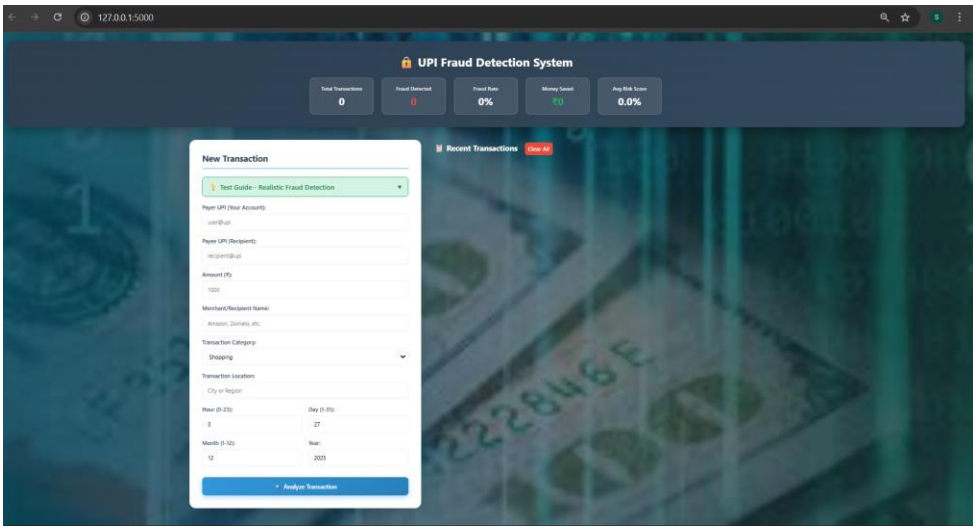


Fig 6.1 UPI Fraud Detection System – Web Application Interface

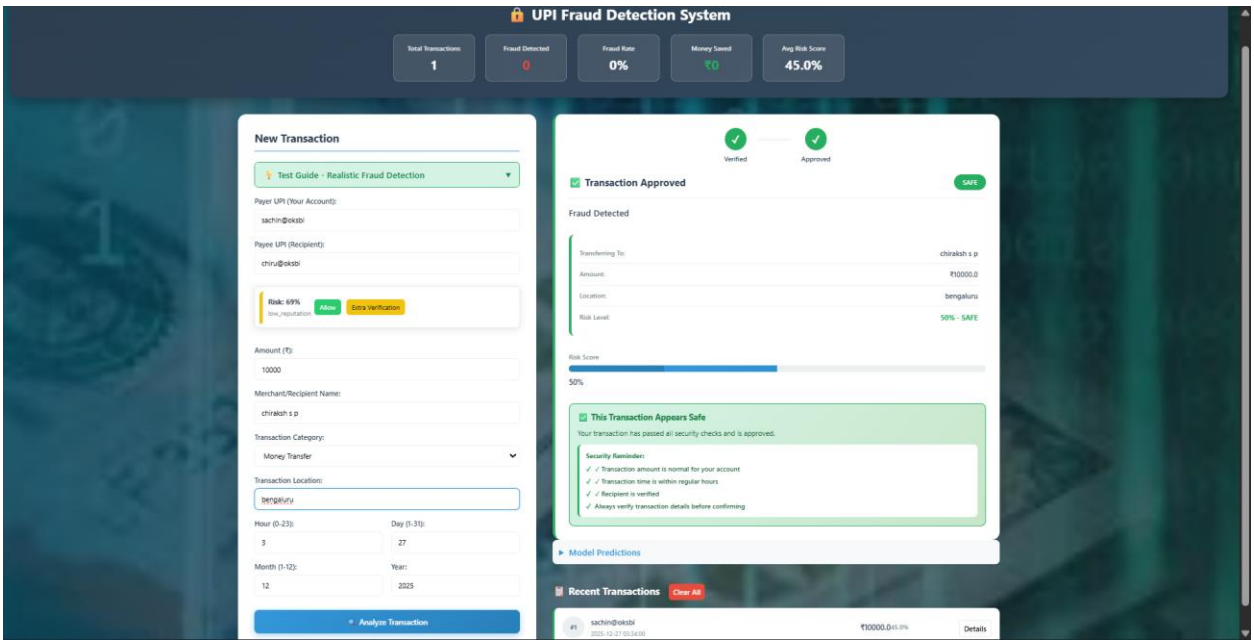


Fig 6.2 Safe UPI Transacations

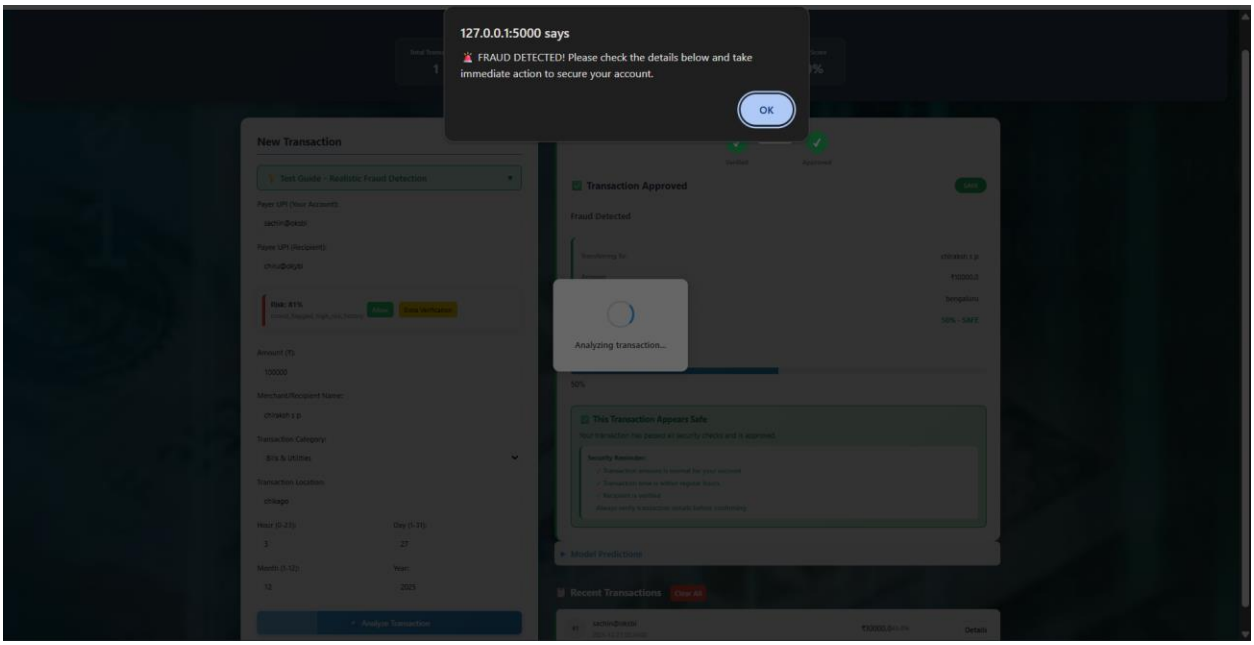


Fig 6.3 Notifying Fraud Detected with Alarm

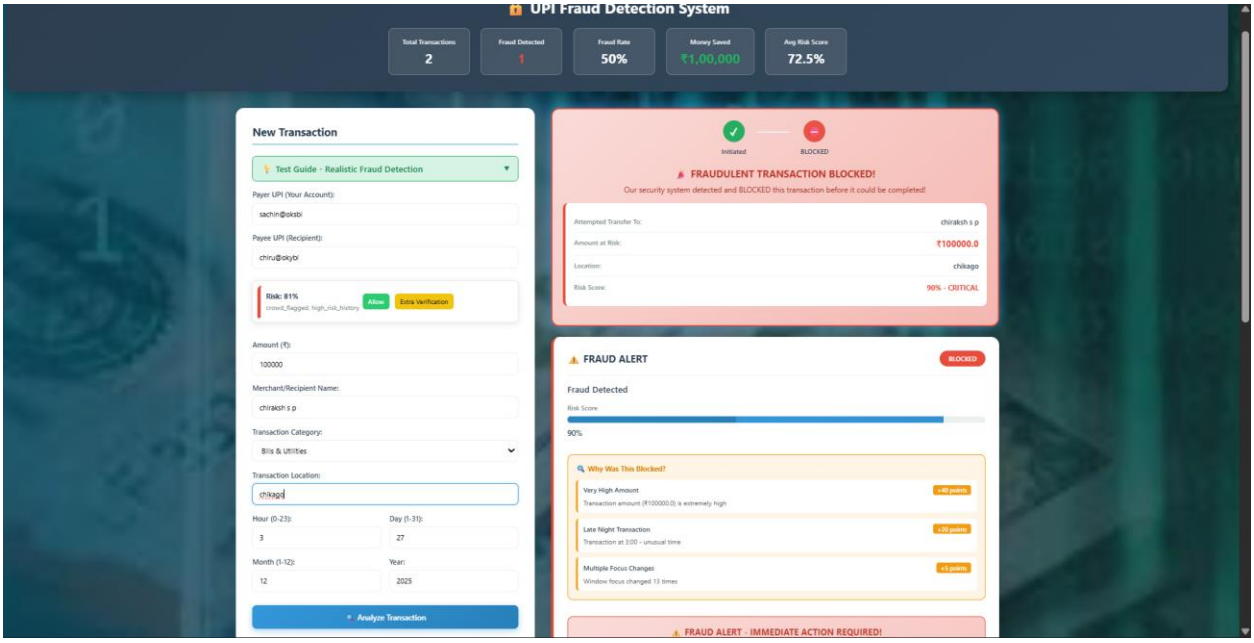


Fig 6.4 Fraud detected and fraud alerts

CHAPTER 07

DESCRIPTION OF TOOLS

1. Application Development & Data Management

- **Flask** is used to build the web application and REST APIs for transaction input, processing, and result display.
- **SQLite** acts as the database to store transactions, user details, and logs in a lightweight and portable manner.
- **Docker / Docker Compose (recommended)** helps create a reproducible environment for services like databases and background workers during deployment.

2. Machine Learning & Analytics

- **scikit-learn** is used to train and deploy machine learning models such as Logistic Regression, Decision Tree, Random Forest, and Isolation Forest.
- **Isolation Forest** detects anomalous or unusual transaction patterns when labeled fraud data is limited.
- **NetworkX** enables graph-based analysis of transactions to extract relationship and behavioral features.
- **joblib / pickle** are used to save and load trained machine learning models efficiently.
- **SHAP (optional)** provides explainability by highlighting feature importance behind fraud predictions.

3. Security, Privacy & System Reliability

- **hashlib (SHA-256)** performs deterministic hashing of UPI identifiers to protect user privacy.
- **cryptography (Fernet)** encrypts sensitive data fields before storing them in the database.
- **pyotp** supports multi-factor authentication using time-based one-time passwords.
- **pytest** is used for automated testing to ensure system correctness and reliability.

CHAPTER 08

NEW LEARNINGS FROM THE TOPIC

1. Graph Features are High-Value Signals

- **Learning:** Metrics like degree, reciprocity, and connected-component reputation uncover collusive behavior missed by per-transaction features.
- **Implication:** Prioritize graph feature pipelines; integrate embeddings/GNNs (e.g., node2vec, PMP/DHMP) for richer account representations.

2. Heterophily Challenges

- **Learning:** Fraud graphs often violate homophily; fraudsters link to benign accounts, limiting vanilla GNNs.
- **Implication:** Use heterophily-aware architectures or partitioned message passing; benchmark on realistic fraud graphs.

3. Hybrid Detection Boosts Coverage

- **Learning:** Supervised models detect known patterns; unsupervised models (IsolationForest) capture novel/rare behaviors.
- **Implication:** Combine via score-fusion strategies; calibrate alert thresholds with human-in-the-loop feedback.

4. Built-in Explainability

- **Learning:** Investigators need immediate, compact explanations (feature importance, graph context).
- **Implication:** Include lightweight SHAP-style hooks or masked feature artifacts with alerts; avoid slow post-hoc methods.

5. Privacy-by-Design Trade-offs

- **Learning:** Deterministic tokenization aids analytic joins but requires careful salt management; field encryption complicates queries.
- **Implication:** Define KMS policies, enforce strict access, index tokenized fields for query efficiency.

6. Security UX Matters

- **Learning:** MFA (TOTP, backup codes) and WebAuthn enhance trust but must consider recovery flows and platform compatibility.

- **Implication:** Provide fallbacks, clear recovery steps, and optional strong authenticators in documentation.

7. Developer Ergonomics Reduce Friction

- **Learning:** Synchronous fallbacks (Celery/Redis) and lightweight model-create scripts enable fast local iteration.
- **Implication:** Maintain dev-friendly toggles/configs; document production path (docker-compose for Redis/Postgres).

8. Testing & Infra Quirks Show Early

- **Learning:** Flask request-context issues and SQLite transient locks often cause flakiness.
- **Implication:** Use Flask test client, retries/init patterns; stage to Postgres in CI to catch concurrency issues early.

9. Operational Readiness > Prototype

- **Learning:** Health endpoints, monitoring, and infra (Redis/workers, key rotation) are essential for production.
- **Implication:** Deploy minimal docker-compose + CI pipeline that runs unit tests, infra smoke tests, and optional WebAuthn integration.

CHAPTER 09

RESULTS AND DISCUSSION

The prototype demonstrates end-to-end functionality: models can be trained and loaded, the ingestion pipeline generates graph features, and the IsolationForest anomaly scorer flags complementary outliers. Authentication mechanisms—including passwords, TOTP, and WebAuthn—are implemented and tested alongside field-level encryption, while health and smoke checks confirm basic operational readiness. Celery/Redis scaffolding exists, though full async validation was limited locally; synchronous fallbacks ensure usability during development.

Combining supervised models with IsolationForest enhances coverage, as supervised learners detect known fraud patterns while the unsupervised detector surfaces novel behaviors. Graph-derived metrics such as VPA reputation, degree, and reciprocity provide contextual signals missing from single-transaction features, and built-in SHAP-compatible explainability significantly aids investigators by delivering concise, actionable rationale for alerts. This hybrid, graph-aware strategy proves effective for triage and early detection.

Next steps involve validating the asynchronous pipeline with Docker/Redis, migrating from SQLite to Postgres for improved concurrency, and strengthening privacy and operational readiness. Planned enhancements include API-level masking of raw UPI, KMS-backed key rotation, integration of heterophily-aware graph models (PMP/DHMP or node embeddings), and a model monitoring and retraining workflow to achieve production-level maturity.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

The UPI Fraud Detection Prototype delivers a compact, privacy-aware, and extensible pipeline demonstrating essential capabilities for practical transaction-risk systems. It features secure ingestion (deterministic tokenization + Fernet encryption), contextual feature engineering (transactional and VPA graph metrics), layered detection (supervised models + IsolationForest), per-transaction explainability hooks, and hardened account security (TOTP + WebAuthn scaffold). Graph-derived signals revealed coordination patterns missed by single-transaction features, while the hybrid detector increased coverage for both known and novel fraud patterns. Although the current implementation is lightweight (SQLite, scikit-learn pickles, synchronous Celery fallback), it establishes a reproducible foundation for production with targeted infrastructure and privacy hardening.

Future Enhancements & Roadmap

Infra & Reliability

- Replace SQLite with PostgreSQL and implement robust migrations (Alembic).
- Add docker-compose for Redis + Celery + Postgres, with CI jobs for infra smoke-tests and unit tests.

Privacy & Security Hardening

- Mask raw UPI in all API outputs, adopt KMS-backed key rotation for DB_ENCRYPTION_KEY, and formalize access/audit logging for token-to-raw mappings.

Model & Graph Upgrades

- Integrate heterophily-aware graph models or node embeddings (PMP, DHMP, node2vec, GNNs) to improve detection of collusive fraud.
- Replace demo models with production training pipelines, cross-validation, calibration, and implement model monitoring with automated retraining.

Explainability & Operations

- Add explainability dashboards (SHAP summaries + graph context) and embed investigator feedback workflows.
- Instrument metrics, alerting, and drift detection to operationalize model performance.

Testing & Integration

- Add WebAuthn integration tests with virtual authenticators and migrate tests to the Postgres + Redis stack to catch concurrency issues early.

REFERENCES

- Jagadeesan, S., Arjun, K.S., Dhanika, G., Karthikeyan, G., & Deepika, K. (2025). *UPI Fraud Detection Using Machine Learning*. In V. Sharmila et al. (Eds.), *Challenges in Information, Communication and Computing Technology*. Taylor & Francis.
- Rani, R., Alam, A., & Javed, A. (2024). *Secure UPI: Machine Learning-Driven Fraud Detection System for UPI Transactions*. Proceedings of the 2nd International Conference on Disruptive Technologies (ICDT), IEEE. DOI: 10.1109/ICDT61202.2024.10489682.
- Ibrahim, M.A. (2023). *Fraud Detection Model for Illegitimate Transactions*. *Kabale University Interdisciplinary Research Journal*, 2(2), 21-37.
- Boulieris, P., Pavlopoulos, J., Xenos, A., & Vassalos, V. (2023). *Fraud Detection with Natural Language Processing*. *Machine Learning Journal*, 1(22).
- Baesens, B., Höppner, S., & Verdonck, T. (2021). *Data Engineering for Fraud Detection*. *Decision Support Systems*, 150, 113492.
- Mohapatra, S., et al. (2017). *Unified Payment Interface (UPI): A Cashless Indian Transaction Process*. *International Journal of Applied Science and Engineering*, 5, 29-42.
- Kumar, R., Kishore, S., Lu, H., & Prakash, A. (2020). *Security Analysis of Unified Payments Interface and Payment Apps in India*. *USENIX Security Symposium*, 1499-1516.
- Li, K., et al. (2024). *SEFraud: Self-Explainable Fraud Detection in Graphs*. *KDD 2024 Proceedings*. arXiv:2406.11389.
- Zhuo, W., et al. (2024). *Partitioning Message Passing for Fraud Graphs*. arXiv:2412.00020.
- Rastogi, S., Sharma, A., Panse, C., & Bhimavarapu, V.M. (2021). *Unified Payment Interface (UPI): A Digital Innovation and Its Impact on Financial Inclusion*. *Universal Journal of Accounting and Finance*, 9(3), 518-530.