ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-2/01-10
Gajula Sadhika *et. al.*, /International Journal of Engineering & Science Research

Full Length Article

UPI Guard: Intelligent UPI Fraud Detection System

S Revathi[1], Gajula Sadhika[2], Nelapatla Sahithi[3], Kanduri Sri Navya[4]
[1]Assistant Professor; Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India.
[2,3,4]B.Tech Students ; Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India.
Mail Id; nelapatlasahithi@gmail.com[3], gajulasadhika18@gmail.com[2], srinavya182005@gmail.com[4]
Accepted 05-04-2026

Abstract
The rapid growth of digital payment platforms has transformed financial transactions by enabling fast and convenient money transfers. Among these platforms, the Unified Payments Interface (UPI) has gained widespread adoption due to its simplicity and real-time transaction capability. However, the increasing reliance on UPI has also led to a surge in fraudulent activities such as phishing attacks, unauthorized transfers, and scam-based payment requests. Detecting these fraudulent activities in real time has become a critical challenge for financial systems.This paper proposes a machine learning-based UPI fraud detection system designed to identify suspicious transactions and prevent financial losses. The system evaluates transaction attributes including amount, time, and receiver information to estimate the probability of fraud. A classification model is trained to categorize transactions as legitimate or fraudulent based on learned behavioral patterns. To improve detection accuracy, rule-based validation is integrated alongside machine learning prediction. Suspicious receiver identifiers and unusually high transaction values are flagged as potential threats. Transactions identified as fraudulent are automatically blocked, and repeated fraudulent attempts lead to account suspension. The system is implemented using Python, Flask, and MySQL, with an administrative dashboard for monitoring fraud trends and high-risk users. The proposed solution enhances digital payment security through intelligent and real-time fraud detection.

Keywords: *UPI Fraud Detection, Machine Learning, Digital Payments, Fraud Prevention, Transaction Analysis, Real-Time Detection, Cybersecurity.*

Introduction

Digital payment technologies have become an essential component of modern financial systems. With the widespread use of smartphones and internet connectivity, electronic payment methods have replaced traditional cash-based transactions. Among these technologies, the Unified Payments Interface (UPI) has emerged as one of the most popular digital payment systems due to its convenience, accessibility, and real-time processing capability. It enables users to transfer funds instantly between bank accounts using mobile devices, thereby simplifying everyday financial transactions.To address these challenges, this project proposes a machine learning-based UPI fraud detection system capable of analyzing transaction behavior in real time. The system learns from historical transaction data and identifies patterns associated with fraudulent activities. By evaluating key parameters such as transaction amount, time, and receiver details, the system predicts the likelihood of fraud. Additionally, rule-based checks are incorporated to enhance detection accuracy. If a transaction is classified as fraudulent, it is blocked immediately to prevent financial loss. The system also includes an account-level protection mechanism that blocks users after repeated fraudulent attempts. An administrative dashboard provides insights into fraud statistics, trends, and high-risk users, enabling effective monitoring and decision-making.

Purpose of the Project

The primary purpose of this project is to develop an intelligent system capable of detecting fraudulent transactions in UPI-based digital payment platforms. With the increasing use of online payments, the risk of fraud has grown significantly. The proposed system aims to provide a reliable solution that identifies suspicious transactions in real time and prevents financial losses. By analyzing transaction attributes such as amount, time, and receiver information, the system determines whether a transaction is safe or fraudulent.Another objective of the project is to improve fraud detection accuracy using machine learning techniques. Unlike traditional rule-based systems, the proposed model learns from transaction data and identifies patterns associated with fraudulent behavior. This enables the system to detect complex and evolving fraud activities more effectively. In addition, rule-based validation such as identifying suspicious receiver identifiers and high-value transactions is incorporated to strengthen the detection mechanism.

The project also focuses on enhancing security through an account-blocking feature and providing an admin dashboard for monitoring fraud trends and high-risk users.

Existing System

In the current digital payment ecosystem, fraud detection mechanisms primarily rely on predefined rules and manual monitoring processes. Financial institutions use fixed conditions such as transaction limits, unusual activity patterns, and blacklist databases to identify suspicious transactions. Although these methods provide a basic level of security, they are not sufficient to handle the increasing complexity and volume of digital transactions. These systems often detect fraud only after the transaction has been completed, making it difficult to recover lost funds.Another limitation of traditional systems is the lack of adaptability. Rule-based approaches cannot effectively identify new fraud patterns since they depend on static conditions. This leads to high false positives, where legitimate transactions are flagged as fraud, and false negatives, where fraudulent transactions go undetected. Additionally, manual verification processes increase response time and reduce overall system efficiency.

Problems in Existing System

Existing fraud detection systems suffer from several limitations. Most systems lack real-time detection capabilities and identify fraud only after transaction completion. They depend heavily on predefined rules that cannot adapt to evolving fraud techniques. Detection accuracy is often low, and these systems do not learn from past data. Furthermore, there is no mechanism to block users after repeated fraudulent attempts, and monitoring tools for analyzing fraud trends are limited. These shortcomings highlight the need for an intelligent and adaptive fraud detection system.

Proposed System

The proposed system introduces a machine learning-based approach for detecting fraudulent UPI transactions in real time. The system analyzes transaction details such as amount, time, and receiver information to predict whether a transaction is safe or fraudulent. A trained classification model evaluates the input features and generates a fraud probability score. In addition to machine learning prediction, rule-based validation is applied to identify suspicious receiver identifiers and high-value transactions.

If a transaction is identified as fraudulent, it is blocked immediately to prevent financial loss. The system also monitors user behavior and blocks accounts after multiple fraudulent attempts. This feature enhances security by preventing repeated misuse. An administrative dashboard is provided to display transaction statistics, fraud percentage, high-risk users, and fraud trends. The proposed system offers a proactive and scalable solution for securing digital payment transactions.

Related Work

Several research studies have explored fraud detection using machine learning and anomaly detection techniques. Previous works demonstrate that classification algorithms such as Logistic Regression, Decision Trees, and Random Forest are effective for identifying fraudulent transactions. Anomaly detection techniques are commonly used to detect unusual transaction patterns associated with fraud. Comparative studies indicate that Logistic Regression performs well for binary classification problems.Most existing research focuses on offline detection and lacks real-time implementation. The proposed system builds upon these approaches by combining machine learning with rule-based validation and implementing real-time fraud detection using a web-based architecture. The system also integrates database storage and analytics to monitor fraud trends and user behavior.

Requirement Analysis

Requirement analysis defines the operational and performance expectations of the proposed UPI Fraud Detection System. Based on the comparative study, the system requirements are categorized into functional and non-functional requirements. These requirements ensure that the system performs real-time fraud detection, maintains security, and provides a reliable user experience. The analysis also includes tools, technologies, and computational requirements necessary for implementation.

Functional Requirements

The system provides secure user authentication to ensure that only authorized individuals can access the application. Users and administrators log in using valid credentials, and session management is implemented to maintain secure access and prevent unauthorized usage. The system supports transaction processing by allowing users to enter details such as receiver UPI ID and transaction amount. These inputs are structured and forwarded for further analysis.Each transaction is evaluated using a trained machine learning model. The system analyzes parameters such as transaction amount, frequency, and time to determine whether the transaction is legitimate or fraudulent. Based on the prediction, the system calculates a probability score and assigns a risk level such as low, medium, or high. This risk evaluation helps users understand the likelihood of fraud.The system controls transactions by allowing safe transactions and blocking suspicious ones in real time. Fraudulent attempts are prevented immediately to reduce financial risk. All transaction details, including user information, time, amount, probability, and prediction results, are stored in a MySQL database. This ensures proper record management and supports future analysis. The system also tracks repeated fraudulent attempts and identifies high-risk users. When a predefined

threshold is exceeded, an automatic account blocking mechanism is activated to prevent further misuse.An administrator dashboard is provided to display key statistics such as total transactions, number of fraud cases, fraud percentage, and system risk level. Graphical visualizations are generated to analyze fraud trends over time. Users can also access their transaction history, which improves transparency and accountability. Secure logout functionality is implemented to clear session data and enhance system security.

Non-Functional Requirements

The system is designed to be scalable so that it can handle increasing numbers of users and transactions without performance degradation. Multiple transactions can be processed simultaneously, ensuring smooth operation even during high load conditions. High performance is achieved by implementing real-time fraud detection with minimal delay. The system processes user inputs quickly and generates prediction results instantly using the trained model.Usability is emphasized by providing a simple and intuitive interface. Users can easily enter transaction details and view results without requiring technical expertise. Reliability is ensured through stable system performance and secure storage of transaction data. The database maintains records consistently, preventing data loss.Security is enhanced through authentication, session management, input validation, and automatic account blocking after repeated fraud attempts. These measures protect both user information and system integrity. The system is also designed for maintainability using a modular architecture. Individual components such as the machine learning model, database, and frontend interface can be updated independently without affecting overall functionality.

Design

System Architecture

The architecture of the UPI Fraud Detection System is designed to process user transactions and detect fraudulent activities in real time. The system integrates the web interface, backend processing engine, machine learning model, and database components. The process begins when a user enters transaction details through the web interface. These inputs are sent to the Flask backend, where validation and preprocessing are performed. The processed data is then forwarded to the trained machine learning model, which calculates the probability of fraud.Based on the prediction and additional rule-based checks, the system classifies the transaction as safe or fraudulent. The result is displayed to the user immediately. All transaction details and prediction results are stored in the MySQL database. The admin dashboard retrieves stored data to generate analytics such as fraud trends and high-risk users. This architecture ensures real-time processing, secure monitoring, and efficient data management.

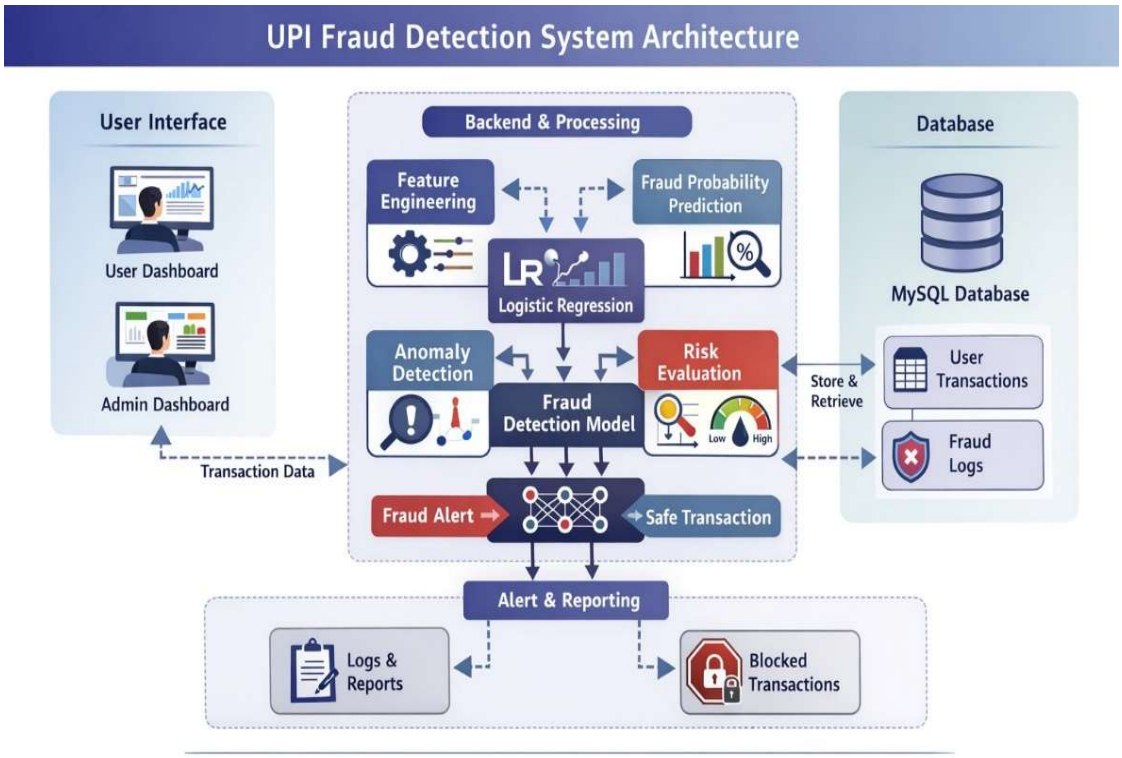


Fig 1: System Architecture.

Technical Architecture

The technical architecture connects the frontend interface, Flask backend, machine learning model, and MySQL database. The frontend collects transaction data from users and sends it to the backend. The backend validates the input and forwards it to the machine learning model for prediction. Rule-based checks are applied to improve detection accuracy. The final decision is returned to the user and stored in the database. The admin dashboard retrieves stored data to display statistics and trends, enabling monitoring and decision-making.

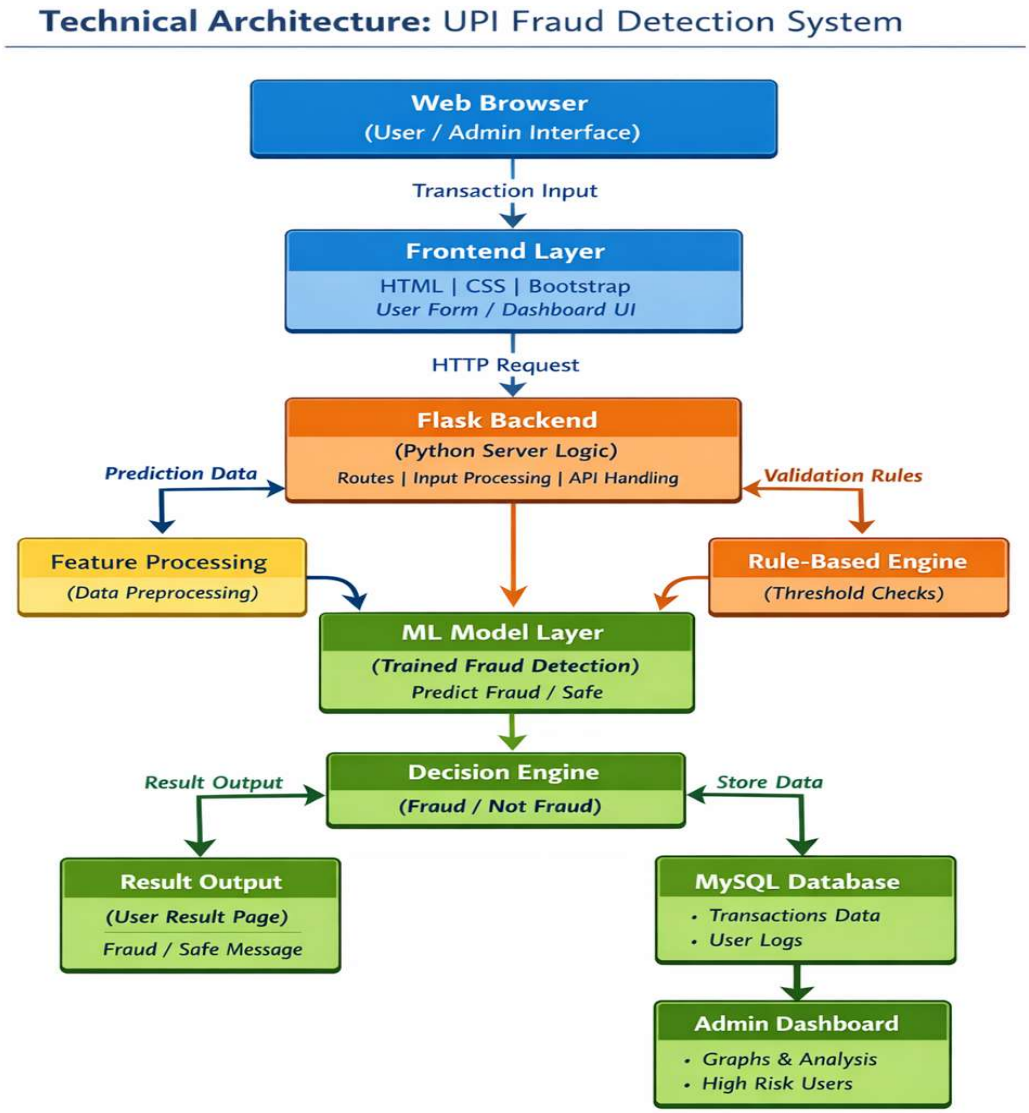


Fig 2: Technical Architecture

Flow Chart Description

The working process begins when a user initiates a transaction. Transaction details such as UPI ID and amount are collected and preprocessed. The Logistic Regression model evaluates the input and generates a fraud probability. If the probability exceeds a predefined threshold, the transaction is classified as fraudulent and an alert is generated. Otherwise, the transaction is marked as safe and allowed to proceed. This structured evaluation ensures that every transaction is analyzed before execution.

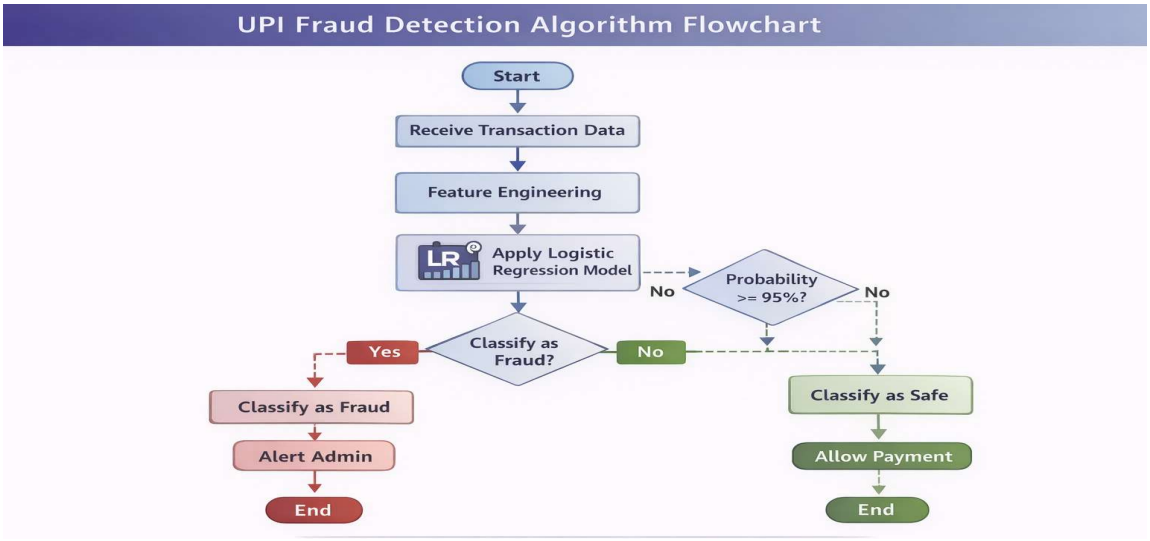


Fig 3 Flow Chart

Algorithm

The algorithm begins with input data collection, where the user enters transaction details through the web interface. The data is sent to the Flask backend for validation. After validation, preprocessing converts the data into a numerical format suitable for the machine learning model. Feature representation is performed to extract relevant attributes for prediction. The trained model analyzes the features and classifies the transaction as fraud or safe. The backend processes the result and displays it to the user. Finally, transaction details and prediction results are stored in the MySQL database for future reference. This step-by-step process ensures accurate and efficient fraud detection.

Implementation

The implementation of the proposed UPI Fraud Detection System integrates machine learning techniques, backend web technologies, database management, and frontend visualization tools. The system is designed to perform real-time fraud detection by combining predictive analytics with rule-based validation. The architecture ensures seamless communication between modules, enabling efficient transaction monitoring and secure decision-making.

Technologies

The core fraud detection mechanism is built using machine learning techniques. Logistic Regression is employed as the primary classification algorithm to distinguish between fraudulent and legitimate transactions. The model is trained using historical transaction data to identify patterns associated with suspicious activities. During prediction, the model generates probability values using the predict_proba() function. These probabilities are compared against a predefined threshold value of 0.8 to determine whether a transaction should be classified as fraud.

To improve model performance, feature scaling is applied using StandardScaler, and class imbalance is handled using balanced class weights. Logistic Regression is selected due to its computational efficiency, interpretability, and effectiveness for binary classification problems.

The model performance is evaluated using multiple evaluation techniques. These include the classification report, which provides accuracy, precision, recall, and F1-score, and the confusion matrix, which summarizes prediction outcomes. The ROC-AUC score is used to measure model discrimination capability, and the ROC curve visually represents the trade-off between true positive rate and false positive rate. These evaluation methods ensure reliable fraud detection performance.

Data Processing

Data processing plays an important role in preparing transaction data for machine learning. The Pandas library is used to load and analyze the dataset. It facilitates reading data from CSV files, identifying missing values, understanding class distribution, and preparing structured input for model training. NumPy is used for numerical computations and for generating feature arrays required by the machine learning model.

Data preprocessing includes splitting the dataset into training and testing sets using stratified sampling to maintain class balance. Feature scaling is applied to normalize numerical values and improve model accuracy. Class imbalance is addressed by assigning

balanced weights to the model, ensuring that minority fraud cases are properly learned.

Backend Development

The backend of the system is implemented using the Flask framework. Flask handles routing, request processing, user authentication, and communication between the frontend interface and machine learning model. It processes user inputs, performs validation, executes prediction logic, and renders HTML templates. Flask acts as the central controller that coordinates system operations.

Session management is implemented using Flask sessions. This mechanism maintains user login states across pages and stores information such as username and role. It also supports role-based access control, allowing administrators to view analytics dashboards while users perform transaction checks. Session management ensures secure interaction and prevents unauthorized access.

Frontend Interface

The frontend interface is developed using HTML, CSS, and Bootstrap. HTML is used to design web pages such as login screens, user dashboards, transaction forms, and result pages. Bootstrap enhances the visual appearance by providing responsive layouts, predefined components, and consistent styling. This ensures that the system is accessible across different devices.

Chart.js is integrated into the admin dashboard to visualize fraud trends. It generates line graphs showing fraud occurrences over time. These visualizations help administrators monitor suspicious activity and analyze system performance.

System Implementation

The system is implemented using Python-based backend logic integrated with machine learning prediction. The Flask application manages authentication, transaction processing, and database communication. When a user logs in, the system assigns roles and redirects them to appropriate dashboards. The administrator dashboard displays total transactions, fraud count, fraud percentage, risk level, recent transactions, and high-risk users. It also generates fraud trend graphs using stored database records.

When a user submits a transaction, the system first checks whether the account is blocked due to repeated fraud attempts. If the account is active, transaction details such as amount and receiver ID are processed. The system generates feature vectors and passes them to the trained model. Additional rule-based checks are applied to detect suspicious receiver IDs or unusually high transaction values.

Based on prediction results, the system classifies the transaction as safe or fraudulent. Fraudulent transactions are blocked, while legitimate transactions are allowed. The system assigns a risk level and stores the transaction details in the MySQL database. If repeated fraudulent behavior is detected, the account blocking mechanism is triggered automatically.

Testing and Validation

Testing and validation were performed to ensure that the UPI Fraud Detection System operates correctly under different scenarios and user interactions. The testing phase focused on verifying fraud prediction accuracy, backend processing, database operations, and user interface functionality. Both manual testing and logical validation techniques were applied to confirm that the system behaves reliably when handling normal inputs, edge cases, and invalid data. This structured testing process helped identify issues during development and ensured stable performance.

Test Strategy

The testing strategy defines the overall methodology used to evaluate the system. The approach emphasizes validating key modules such as user authentication, transaction handling, fraud prediction, database integration, and dashboard analytics. The system was tested to ensure accurate classification of transactions and smooth communication between components. Different scenarios, including safe transactions, suspicious inputs, and repeated fraud attempts, were evaluated. The strategy also ensured that both frontend and backend modules function correctly together, providing a consistent user experience.

Testing Approach

Testing was conducted in a local development environment where the application was executed using a Flask server. A web browser was used to interact with the user interface, while MySQL handled transaction storage. The pre-trained machine learning model was integrated for prediction. Test data included multiple transaction amounts, valid and suspicious UPI IDs, and repeated fraud attempts.

Various testing techniques were used to validate system behavior. Black-box testing was performed to verify user interface functionality such as login, payment processing, and result display without considering internal code. White-box testing focused on backend logic including fraud detection thresholds, rule-based validation, and account blocking mechanisms. Edge case testing examined high-value transactions, suspicious IDs, and repeated fraud attempts. Input validation testing ensured that incorrect or incomplete inputs were handled appropriately.

Testing tools included manual testing through web browsers, Flask debug mode for identifying runtime errors, SQL queries for validating database records, and logging statements for debugging backend logic. Any identified bugs were documented and corrected. Errors in logic, database inconsistencies, or interface issues were resolved and retested to ensure system reliability.

Types of Testing

Unit testing was conducted to verify individual modules such as authentication, transaction processing, and prediction logic. Each component was tested independently to ensure correct input-output behavior. Integration testing validated communication between modules, including interaction between the frontend, backend, machine learning model, and database. This ensured that all components functioned together correctly.

Functional testing confirmed that system features operate according to requirements. Valid inputs were accepted, invalid inputs were rejected, and output results were verified. System testing evaluated the application in an end-to-end environment to ensure that the entire workflow operates smoothly. This included testing user interactions, data processing, and prediction accuracy.

White-box testing was used to analyze internal logic, including decision branches, loops, and probability thresholds. This helped detect hidden logical errors. Black-box testing evaluated the system from the user perspective, ensuring correct outputs for given inputs without considering internal implementation. These combined testing methods ensured comprehensive validation of the application.

Result Analysis

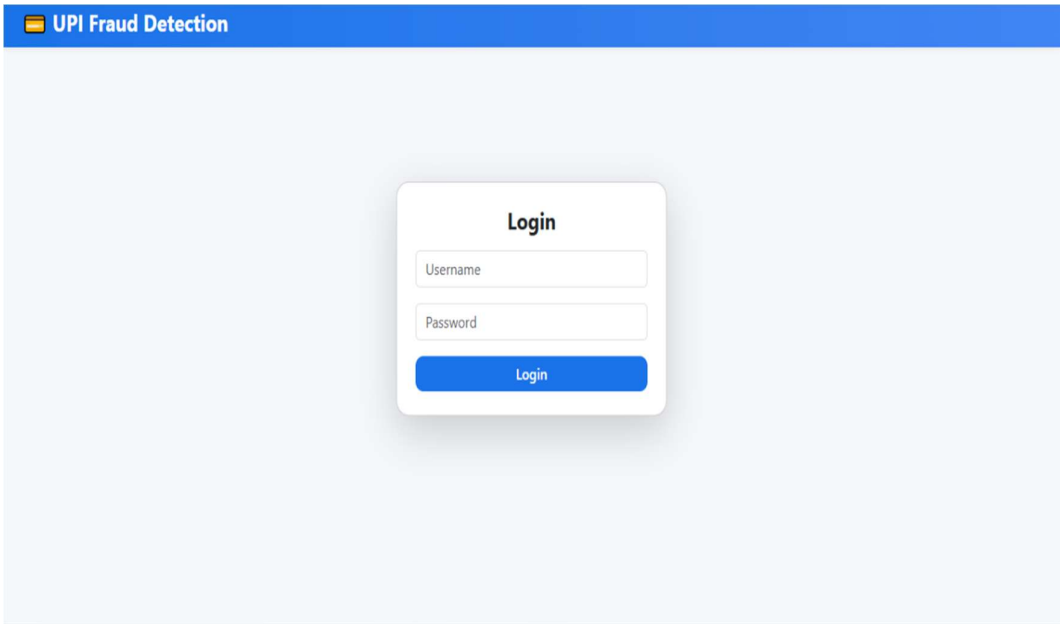
The UPI Fraud Detection System was successfully tested using multiple transaction scenarios. The Logistic Regression model demonstrated reliable performance in classifying fraudulent transactions. By utilizing probability-based prediction, the system assigned risk scores to each transaction. A threshold-based classification approach improved detection accuracy by identifying high-risk transactions.

Class imbalance in the dataset was handled using balanced weights during model training, which improved detection of rare fraud cases. Rule-based validation further enhanced accuracy by flagging suspicious UPI IDs and high-value transactions. This hybrid approach improved reliability compared to using machine learning alone.

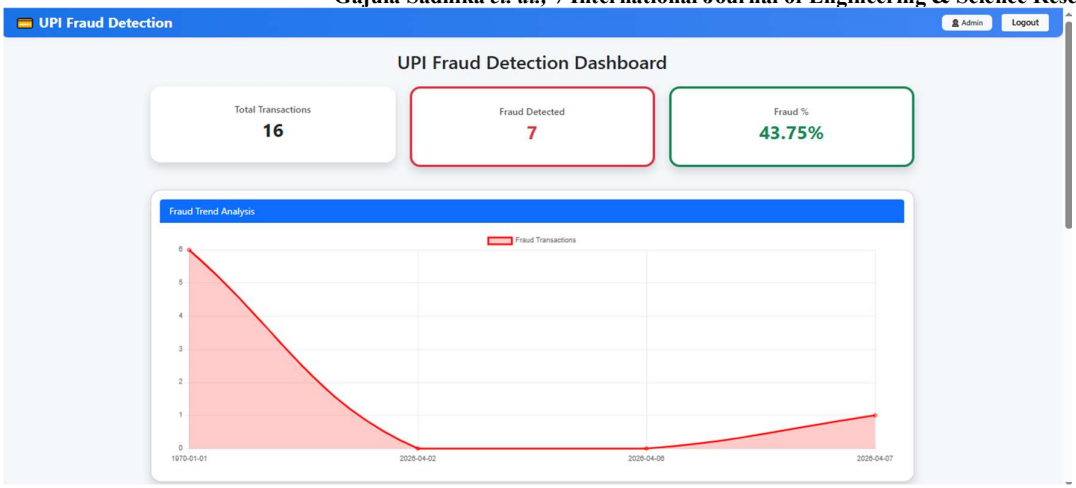
The system also implemented dynamic risk classification, categorizing transactions as low, medium, or high risk. This improved user understanding and decision-making. The admin dashboard provided insights into total transactions, fraud cases, fraud percentage, and trends. Graphical visualization enabled easier pattern recognition.

The account blocking mechanism strengthened system security by restricting users after repeated fraudulent attempts. During testing, the system performed consistently across normal, high-risk, and edge-case scenarios. Integration between frontend, backend, machine learning model, and database was stable. Overall, the system successfully achieved accurate fraud detection and demonstrated readiness for real-world applications.

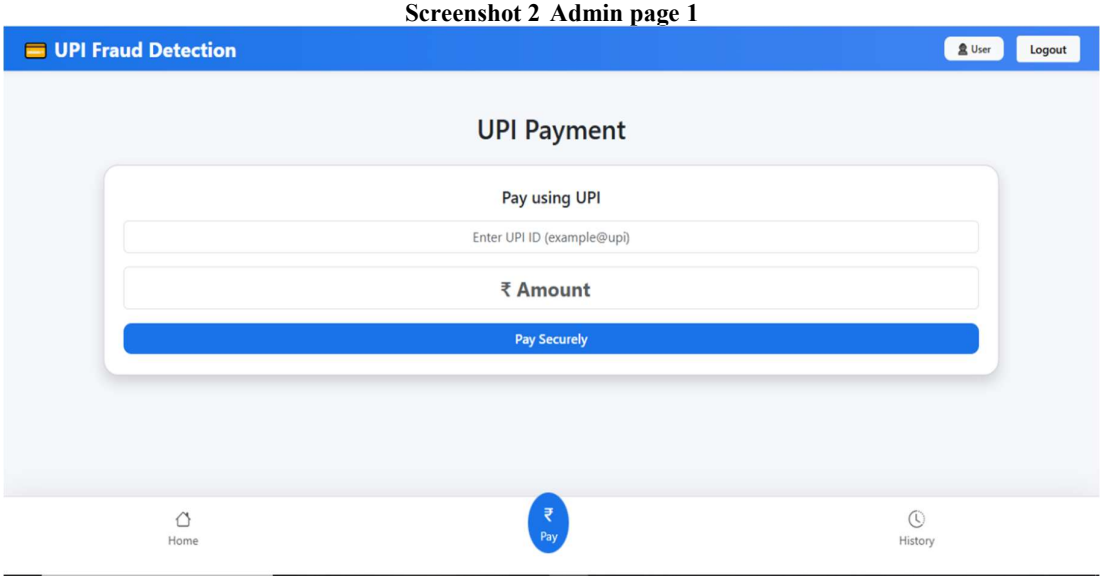
Screenshots



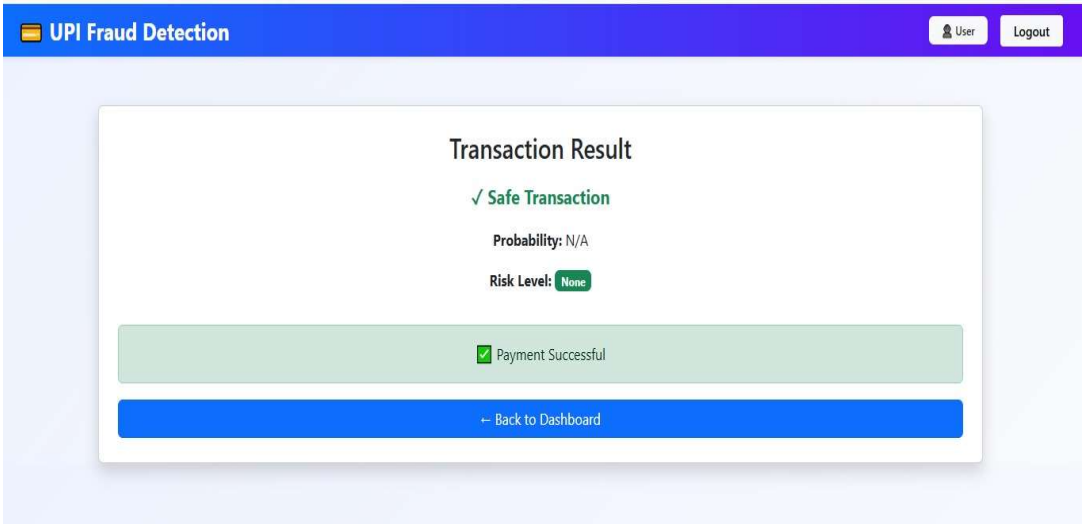
Screenshot 1: Login Page



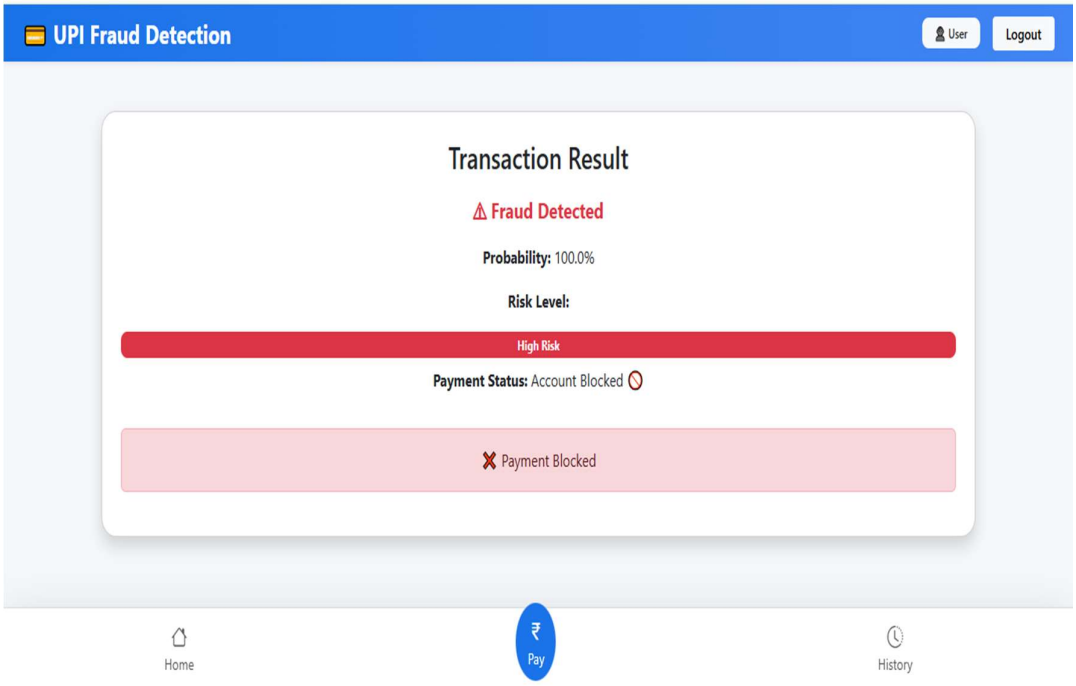
Screenshot 2 Admin page 1



Screenshot 3; Payment Page



Screenshot 4: Resul Page 1



Screenshot 5: Result Page

Conclusion

The proposed UPI Fraud Detection System was successfully designed and implemented using a combination of machine learning and rule-based techniques. The system effectively identifies fraudulent transactions by applying a Logistic Regression model along with predefined validation rules such as suspicious UPI IDs and high transaction amounts. This hybrid approach improves detection accuracy and reduces the chances of false predictions.The probability-based prediction mechanism enables the system to assign risk levels including low, medium, and high. This enhances usability by providing meaningful insights into transaction safety. Security features such as account blocking after repeated fraud attempts further strengthen system reliability. The admin dashboard provides real-time analytics, including fraud trends and high-risk users, supporting effective monitoring.Overall, the system demonstrates that integrating machine learning with practical business logic can significantly enhance fraud detection performance. The architecture is scalable, efficient, and capable of handling real-time transaction analysis, making it suitable for modern digital payment environments.

Future Scope

Although the system performs effectively, several enhancements can further improve its functionality. Advanced machine learning algorithms such as Random Forest and gradient boosting techniques can be implemented to improve prediction accuracy. Deep learning models may also be explored for identifying complex fraud patterns.

Future improvements may include integration with banking APIs or UPI gateways for real-time fraud detection in live payment systems. Additional security mechanisms such as multi-factor authentication, OTP verification, and behavioral analysis can strengthen system protection. Anomaly detection techniques may also be incorporated to detect unknown fraud patterns.Deployment on cloud platforms can improve scalability and accessibility. Mobile application support can provide a convenient interface for users. Continuous model retraining using real-time data can help adapt to evolving fraud strategies. These enhancements will make the system more robust and suitable for large-scale financial applications.

References

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar, “Anomaly Detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[2] Andrew Ng and Michael I. Jordan, “On Discriminative vs Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes,” *Advances in Neural Information Processing Systems*, 2002.

[3] Sanjay Bhattacharyya, Sanjib Jha, K. Tharakunnel, and J. C. Westland, “Data Mining for Credit Card Fraud: A Comparative Study,” *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, Feb. 2011.

[4] European Card Fraud Dataset, "Credit Card Fraud Detection Dataset," Kaggle, 2013. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud

[5] Fabian Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] Palash Goyal, "Building a Fraud Detection System using Machine Learning," *Towards Data Science*, 2023. Available: https://towardsdatascience.com

[7] Miguel Grinberg, *Flask Web Development: Developing Web Applications with Python*, O'Reilly Media, 2018.

[8] Oracle Corporation, "MySQL Documentation," 2024. Available: https://dev.mysql.com/doc/

[9] Bootstrap Team, "Bootstrap 5 Documentation," 2024. Available: https://getbootstrap.com/

[10] Chart.js Contributors, "Chart.js Documentation," 2024. Available: https://www.chartjs.org/docs/