VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Machhe, Belagavi, Karnataka-590018

A Project Report

On

“Anomaly-Based Intrusion Detection System in Financial Transaction using Ensemble Machine Learning and Blockchain”

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Science & Engineering

Submitted by

Refa Samrin	**4GW22CS092**
Shreya M A	**4GW22CS103**
Supreetha M S	**4GW22CS110**
Thrupthi K N	**4GW22CS116**

Under the Guidance of

Dr. Rajashekar M B
Associate Professor
Dept. of CSE, GSSSIETW

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Accredited by NBA, New Delhi, Validity 01.07.2023 to 30.06.2026)

GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi & Govt. of Karnataka)

K.R.S ROAD, METAGALLI, MYSURU-570016, KARNATAKA

Accredited by NAAC

2025-26

Geetha Shishu Shikshana Sangha (R)
GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN
K.R.S Road, Mysuru-570016, Karnataka
(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)
Accredited by NAAC

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
(Accredited by NBA, New Delhi, Validity 01.07.2023 to 30.06.2026)

CERTIFICATE

Certified that the 7th Semester Project titled **"Anomaly-Based Intrusion Detection System in Financial Transactions Using Ensemble Machine Learning and Blockchain"** is a bonafide work carried out by **Refa Samrin (4GW22CS092), Shreya M A (4GW22CS103), Supreetha M S(4GW22CS110) and Thrupthi K N(4GW22CS116)** in partial fulfilment for the award of degree of Bachelor of Engineering in **Computer Science & Engineering** of the Visvesvaraya Technological University, Belagavi, during the year 2025-26. The Project report has been approved as it satisfies the academic requirements with respect to the project work prescribed for Bachelor of Engineering Degree.

Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
(Dr. Rajashekar M B)	**(Dr. Raviraj P)**	**(Dr. Shivakumar M)**

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

Certificate from the Guide

This is to certify that the project report entitled Anomaly-Based Intrusion Detection System in Financial Transactions Using Ensemble Machine Learning and Blockchain submitted by **Refa Samrin, Shreya M A, Supreetha M S, Thrupthi K N** to the Visvesvaraya Technological University, Belagavi, in partial fulfillment for the award of the degree of B. E in Computer Science and Engineering is a bonafide record of project work carried out by her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

Signature:

Name of the Guide: **Dr. Rajashekar M B**

Department: **Computer science and Engineering**

Counter signature of HOD/Programme coordinator with seal

DECLARATION BY STUDENTS

DECLARATION

We declare that this project report titled Anomaly-Based Intrusion Detection System in Financial Transactions Using Ensemble Machine Learning and Blockchain submitted in partial fulfillment of the degree of **B. E in Computer Science and Engineering** is a record of original work carried out by me under the supervision of Dr. Rajashekar M B, and has not formed the basis for the award of any other degree, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Signature of the student(s)

Name	Signature
1. Refa Samrin	
2. Shreya M A	
3. Supreetha M S	
4. Thrupthi K N	

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible.

First and foremost we offer our sincere phrases of thanks to **Late Smt. Vanaja B Pandit, Former Honorary Secretary, GSSS(R)** Mysuru for the blessings and support.

We offer our sincere thanks to **Smt. Anupama B Pandit, Secretary, GSSS(R)** and the **Management of GSSSIETW, Mysuru** for their invaluable support and guidance in carrying out this project.

We would like to express our gratitude to our **Principal, Dr. Shivakumar M** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the project.

We consider it is a privilege and honour to express our sincere thanks to **Dr.Raviraj P Professor and Head,** Department of Computer Science& Engineering for his support and invaluable guidance throughout the tenure of this project.

We would like to thank our Guide **Dr. Rajashekar M B,** Associate Professor, Department of Computer Science & Engineering for his support, guidance, motivation, encouragement for the successful completion of this project.

We would like to thank our Project Co-ordinators **Dr. Rajashekar M B,** Associate Professor & **Ms. Nischitha B S,** Assistant Professor, Department of Computer Science & Engineering for their constant monitoring, guidance & motivation throughout the tenure of this project.

We intend to thank all the teaching and non-teaching staffs of our Computer Science & Engineering department for their immense help and co-operation.

Finally we would like to express our gratitude to our parents and friends who always stood with us to complete this work successfully.

Refa Samrin (4GW22CS092)
Shreya M A (4GW22CS103)
Supreetha M S (4GW22CS110)
Thrupthi K N (4GW22CS116)

ABSTRACT

The rapid growth of digital payment systems such as UPI, mobile banking, and online transactions has significantly increased the risk of financial fraud, demanding intelligent and adaptive security mechanisms. Traditional fraud detection systems rely heavily on rule-based or signature-based approaches, which lack real-time decision-making, transparency, and the ability to detect evolving fraud patterns. To address these limitations, this project proposes an Anomaly-Based Intrusion Detection System (IDS) that integrates Ensemble Machine Learning models and Blockchain technology to enhance security, accuracy, and trust in financial transactions.

The system employs a multi-stage methodology that begins with data preprocessing, feature selection, and categorical encoding to prepare financial transaction datasets for machine learning. A hybrid ML architecture combining Convolutional Neural Networks (CNN) for sequential pattern extraction, Support Vector Machines (SVM) for classification, and Naïve Bayes for probability-based evaluation forms the Ensemble Detection Engine. This integrated model analyzes user transactions in real time, identifying suspicious behavioral anomalies with improved accuracy and reduced false positives. Once a transaction is classified as safe, it is stored on a blockchain ledger, ensuring immutability, transparency, and protection against tampering. Fraudulent or unknown transactions trigger instant alerts, blocking mechanisms, and user notifications through a Flask-based web interface.

Experimental results demonstrate the system's capability to detect fraudulent senders, receivers, and unknown users while maintaining a user-friendly interface and non-intrusive verification methods such as time-locks and micro-transactions. By combining ensemble ML accuracy with blockchain's tamper-proof storage, the proposed IDS delivers an effective, reliable, and proactive framework for financial fraud prevention. The system significantly enhances trust, reduces financial risks, and provides a scalable foundation for future extensions in banking, e-commerce, and digital payment ecosystems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SNAPSHOTS

Chapter 1

INTRODUCTION

1.1 Overview

The rapid digital transformation within the financial sector has revolutionized the way monetary transactions are conducted. Modern payment ecosystems—such as online banking, UPI transactions, mobile wallets, internet banking, and e-commerce platforms—have enabled users to perform financial activities seamlessly, anytime and anywhere. As convenience has increased, transaction volumes have grown exponentially, driven by technological advancements, government initiatives, and widespread smartphone penetration. This digital shift has brought tremendous benefits but has also exposed the ecosystem to a wide range of cybersecurity threats. Criminals continuously exploit system vulnerabilities, user negligence, and weak authentication mechanisms to perform fraud, identity theft, phishing attacks, and unauthorized transactions. The magnitude and frequency of such incidents highlight the pressing need for intelligent, real-time intrusion detection mechanisms tailored specifically for financial systems.

Traditional fraud detection mechanisms, which are still widely used in many financial institutions, predominantly rely on rule-based models. These systems function by detecting predefined patterns or threshold breaches, such as unusually high transaction amounts or access from suspicious locations. While these methods are efficient for identifying known fraud patterns, they fail to detect dynamic, evolving, and sophisticated cyberattacks. Fraudsters frequently adapt their techniques to bypass these static rules, making traditional systems inadequate for modern digital transaction environments. As a result, the financial sector faces significant challenges in ensuring robust security, maintaining user trust, and preventing financial losses.

To address these limitations, this project proposes a novel hybrid Intrusion Detection System (IDS) tailored for real-time financial transactions. The system integrates the strengths of Ensemble Machine Learning algorithms—such as Convolutional Neural Networks (CNN), Support Vector Machines (SVM), and Naïve Bayes—to improve detection accuracy and adaptability. These models analyse transactional patterns, user behaviour, and historical data to identify anomalies that may indicate fraudulent activity. By combining multiple models, the system leverages the complementary strengths of each technique, resulting in a more reliable and accurate classification of transactions as benign or suspicious.

In parallel, the project incorporates Blockchain technology to address issues related to data integrity, transparency, and security of transaction records. Blockchain’s decentralized ledger ensures that every transaction is stored immutably, making it nearly impossible for attackers or insiders to manipulate records without detection. Each transaction is timestamped and cryptographically secured, creating a trustworthy environment that significantly reduces the possibility of tampering. This integration not only fortifies the intrusion detection process but also enhances overall system reliability by ensuring that the transaction logs used for ML training and audit purposes remain authentic.

Together, Machine Learning and Blockchain technologies create a synergistic framework capable of providing intelligent decision-making and secure transaction logging. Machine learning models detect anomalies with high precision, while blockchain ensures the trustworthiness of the underlying data. The system is designed to flag suspicious activities instantly, enabling financial institutions and users to take timely preventive actions. This hybrid approach aims to uplift the overall security posture of financial transaction environments, reduce the risk of fraud, and build greater confidence among users.

1.2 Existing System

Traditional financial fraud detection relies heavily on mechanisms that are outdated and unable to keep pace with evolving digital threats. One of the most common ways to do this is through rule-based detection, which flags suspicious activity off of pre-set rules, such as a transaction in an unusually large amount, transferring funds suddenly to foreign countries, multiple rapid transactions, or access from unfamiliar locations. Rules like these have to be manually created or updated, and thus tend to be quite rigid, unable to spot new or sophisticated fraud methods.

Another common technique is signature-based intrusion detection, in which current activities are matched against historical records of known fraudulent behaviour. While this is powerful for determining previously seen attacks, it has a hard time dealing with zero-day threats or unknown patterns of fraud, leading to an alarming rate of false negatives. Strong dependence on attack signatures from the past restricts adaptability, making it unreliable in fast-changing financial environments where attackers modify their techniques constantly.

Another limitation with traditional systems is that they rely on centralized databases to store transaction data. A centralized server introduces a single point of failure, which is then prone to cyberattacks, unauthorized changes, and system downtime. Furthermore, nearly all traditional fraud detection methods flag a transaction well after the transaction has been complete, establishing time delays in threat identification. This lack of real-time monitoring results in financial losses, reduced customer confidence levels, and reduced overall security. As such, all these issues-rigid rule-based systems, signature limitations, centralization risks, and delayed responses-mean that existing financial fraud detection systems are plagued by an inability to offer strong, transparent, and real-time protection in today's ever-changing digital world.

1.3 Scope and Objectives

- To create a dynamic anomaly detection framework focused on providing a fraud user detection and transaction detection of suspicious activity, using ensemble machine learning algorithms such as CNN and SVM.
- To evaluate if the sender and receiver accounts are credible and trustworthy, to prevent undesirable high-risk transactions.
- To develop blockchain technology for transaction record storage, ensuring security, transparency, and immutability to eliminate data double-entry.
- To send alerts and block suspicious or fraudulent transactions before they process, reducing the risk of money lost instantly.
- To create a minimal disruption user-facing web interface on Flack, with minimal user-friendliness during secure transactions.
- To support use for the banking system during banking transactions, digital payment applications, e-commerce systems, and financial institutions needing advanced fraud prevention.
- To assess the systems level of performance using observational metrics of accuracy, precision, recall and F1, for reliability and effectiveness for real-world application.

1.4 Limitations of the Existing System

- Rule-based systems cannot keep up with changing fraud strategies, AI-driven fraud schemes, and deviations in typical behavior.
- They produce high rates of false positives and false negatives that lead to incorrect fraud alerts, users being frustrated, and delayed transactions.
- Rule-based systems use centralized databases that are subject to modification, corruption, or hacking, with no way to guarantee they cannot be modified.
- They do not offer real-time decision-making, as many systems identify fraud only after a transaction is completed.
- Many systems struggle to identify complex fraud patterns like when attackers mimic normal behavior, conduct micro-transactions, or use multiple accounts to conduct transactions.
- There is no consolidated solution, which results in delays and weaknesses from using to approaches to address fraud detection and secure data management.

1.5 Problem Statement

- The rise in complexity of financial fraud activities requires better security measures. Traditional fraud detection systems do not provide enough transparency, flexibility, or real-time capabilities. The main problem is that they cannot keep up with changing fraud patterns and fail to maintain a secure, tamper-proof record of transactions.
- To develop an Anomaly-Based Intrusion Detection System for financial transactions that combines Ensemble Machine Learning techniques to boost accuracy in detecting anomalies and uses Blockchain technology to ensure transparency, security, and unchangeable transaction records.

1.6 Motivation

The increase in financial fraud is a significant issue as digital transactions grow rapidly across banking platforms, online payment systems, and e-commerce. With more people using UPI, mobile banking, and online wallets, cybercriminals exploit system weaknesses, poor monitoring, and outdated fraud detection tools. The lack of effective real-time monitoring, along with the rise of sophisticated AI-driven cyberattacks, has led to serious financial losses for banks, payment service providers, and customers. Fraud not only results in financial harm but also damages user trust and confidence in digital financial systems. This troubling trend pushes the need to create a smart fraud detection system that can understand and spot hidden issues in transaction patterns. A modern system must move beyond traditional rule-based checks and continuously evolve to identify new fraud behaviors.

Additionally, transparency and trust are vital in financial settings. Blockchain technology offers a secure, unchangeable, and verifiable ledger that meets these needs by ensuring transaction records remain intact. There is also a need for proactive fraud prevention—detecting and stopping suspicious activity before a transaction is completed rather than after the damage is done. User convenience is crucial; security features should not be intrusive or disrupt the transaction process. Real-time alerts, micro-verifications, and intelligent time-locks can improve protection while allowing a smooth user experience. Combining data science, machine learning, and blockchain creates a strong defense mechanism that meets the needs of modern financial systems. This project aims to develop a system that is reliable, transparent, proactive, and user-friendly, ensuring better financial security in an increasingly digital world.

1.7 Proposed System

The proposed system offers a hybrid intrusion detection framework that combines machine learning models with blockchain technology to ensure secure and precise fraud detection in financial transactions. The machine learning part integrates Convolutional Neural Networks (CNN) for identifying behavioral patterns, Support Vector Machines (SVM) for accurate anomaly classification, and Naïve Bayes for making decisions based on probability. This combination reduces false alarms and improves reliability.

To secure data storage, the system uses blockchain. Every verified transaction is stored as a permanent, transparent block linked through cryptographic hashing. A Flask-based web application provides an easy-to-use interface that includes login and registration, transaction input, real-time fraud analysis, instant alerts, blocking features, and access to transaction history. The real-time detection process allows users to input transaction details. The ML model analyzes these details to determine if the transaction is safe or fraudulent. If fraud is detected, it triggers immediate alerts and blocking actions; safe transactions are securely added to the blockchain. The system also enhances accuracy through feature engineering and fine-tuning of parameters, which helps reduce both false positives and false negatives.

Additionally, users receive real-time notifications about fraudulent senders, fraudulent receivers, suspicious transaction amounts, and unknown users. By combining machine learning and blockchain technology, the proposed system provides a secure and reliable solution to combat modern financial fraud.

1.8 Organization of the Report

The report is organized in the following manner:

- **Chapter 1** gives the introduction of this project that is what the project does and how it is helpful.
- **Chapter 2** gives the literature survey of the project. In order to understand the project in a cleaner manner, a survey was done to know about existing systems.
- **Chapter 3** gives the system requirements and design of the project and this chapter briefs out the requirements required to fulfill the project.
- **Chapter 4** gives the implementation of the project, with the help of the flow chart and algorithms, project flow can be understood clearly and this chapter also includes the test case description.
- **Chapter 5** gives the testing of the project; it measures the quality of the software we are developing.

****

Chapter 2

LITERATURE SURVEY

Literature Survey is the most important step in the software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need a lot of external support. This support can be obtained from senior programmers, from books or from websites. Before building the system, the above considerations are taken into account for developing the proposed system.

2.1 Survey Findings

[1] Optimization Scheme of Collaborative Intrusion Detection System Based on Blockchain Technology

Jiachen Huang, Yuling Chen, Xuewei Wang, Zhi Ouyang, Nisou Du — 2025

This paper presents an improved collaborative intrusion detection system (CIDS) that combines ensemble learning with blockchain to boost attack detection and secure threat intelligence exchanges. The main issue is that individual IDS nodes face limited visibility and trust problems when sharing alerts. The new system decentralizes decision-making and uses blockchain to validate and timestamp contributions from various detection agents.

The authors introduce a Weighted Random Forest (WRF) as the main detection engine. WRF gives dynamic importance to feature subsets from network and transaction logs, which helps it detect subtle anomalies. Alongside WRF, the authors create the Alternating Random Assignment Selection Mechanism (ARASM) to distribute detection tasks among nodes. This approach balances workload while ensuring diversity in the models learned. The blockchain serves as a secure coordination layer, where detection votes, model updates, and consensus outcomes are recorded as transactions. This guarantees tamper resistance and traceability. The evaluation uses simulated distributed environments and standard intrusion datasets, primarily reporting experiments on a curated NSL-KDD-like dataset and a synthetic multi-node dataset. The results show improved detection accuracy and consensus stability compared to single-node IDS and basic voting ensembles. The blockchain layer stops retroactive changes to alert logs, leading to verifiable audit trails. However, the authors recognize scalability and configuration complexity as key challenges. As the number of collaborating nodes grows, consensus delays and blockchain throughput limits increase detection latency. The evaluation also falls short in real-time performance testing on large-scale modern network datasets, which limits claims about practical deployment. The paper suggests future work on lightweight consensus protocols and adaptive block batching to lower communication overhead and support real-time operations in large federated financial networks.

[2] Real-Time Fraud Detection Using Streaming Data in Financial Transactions

Amarnath Immasidiyetty — 2025

This study focuses on the urgent need for detecting fraud in fast financial transaction streams. The authors create a streaming analytics pipeline that helps with feature extraction, temporal aggregation, and ongoing model updates to quickly classify transactions. The main idea combines deep learning with traditional classifiers. Recurrent or temporal-aware neural networks identify patterns in user transaction histories, while tree-based models and SVMs deliver quick classification for each transaction. The paper emphasizes feature engineering using sliding and tumbling windows to capture both short-term spikes and long-term behavior changes. Engineered features include rolling averages, time intervals between transactions, speed metrics, device fingerprints, and location variability.

The system runs on a streaming platform that allows online learning. Models are continuously retrained with newly labeled data using simple incremental updates to keep up with changing behavior. The experiments utilize synthetic high-throughput streams and a partially anonymized banking dataset to measure latency and accuracy. Results show a significant decrease in detection latency compared to batch methods, with good precision and recall when combining neural embedding features with quick classifiers. However, the authors point out several important limitations: (1) high computational costs for continuous model updates, especially for deep models; (2) ongoing risk of high false-positive rates that can hurt user experience; and (3) the risk of overfitting unless strong regularization and diverse training samples are included. The paper concludes that while streaming machine learning offers hopeful real-time detection, implementing it demands careful planning for scaling, online labeling, and human oversight to handle alerts and build trust.

[3] Detecting Anomalies in Blockchain Transactions Using Machine Learning Classifiers and Explainability Analysis

Mohammad Hasana, Mohammad Shahriar Rahman, Helge Janicke, Iqbal H. Sarker — 2024

This paper looks at detecting anomalies specifically in blockchain transaction datasets. It combines a group of tree-based classifiers with tools that explain the models. The authors are motivated by the rise in illegal activities on public ledgers, such as money laundering, phishing transfers, and mixing services. They explore how features from transactions and graphs can help supervised classifiers identify unusual behavior. The method involves detailed feature engineering, including temporal features like frequency and cadence, graph features such as degree centrality and clustering coefficient, economic features like transaction value adjusted to local currency, and behavioral embeddings based on these factors. The classifiers used in the study include Decision Trees, Random Forests, Gradient Boosting, AdaBoost, and XGBoost. The authors suggest using both stacked ensembles, through hard and soft voting, and layered boosting to increase detection effectiveness.

A key part of the study is the use of explainability with SHAP (SHapley Additive exPlanations) and feature importance ranking to clarify model decisions. This is crucial for forensic analysis and meeting regulatory requirements. The experiments show that ensemble models that include features related to graph data can distinguish many types of malicious transactions from legitimate ones. However, there are practical difficulties. The dataset is highly imbalanced, with very few labeled fraudulent cases compared to legitimate transactions. This imbalance can create bias and lead to unreliable metrics. To address this, the paper examines oversampling techniques and class weighting but points out the risks of overfitting and issues with synthetic sample artifacts. Additionally, the computational costs are significant for large ledger datasets, especially when calculating graph metrics at scale. The study suggests incorporating unsupervised pre-screening to lessen the labeling workload and adopting scalable graph-processing frameworks for practical blockchain analytics.

[4] A Decentralized and Self-Adaptive Intrusion Detection Approach Using Continuous Learning and Blockchain Technology

Ahmed Abubakar Aliyu, Jinshou Liu, Ezekia Gillard — 2024

This research presents a self-adaptive IDS architecture that combines continuous learning models, such as LSTM/RNN and autoencoders, with blockchain-based recording and coordination. The goal is to allow IDS nodes to learn continuously from streaming telemetry and to validate anomaly detections together without depending on a central authority. The detection pipeline uses the reconstruction error from autoencoders for unsupervised anomaly scoring. It employs LSTM networks to model temporal dependencies and sequence anomalies in transaction streams. Autoencoders learn compressed representations of normal behavior. Large reconstruction errors suggest potential anomalies, which are then confirmed through consensus recorded on a lightweight blockchain ledger.

The blockchain component secures model updates, detection votes, and audit logs. This ensures the integrity of the learning history and detection results. Continuous learning occurs through periodic local model fine-tuning using streaming windows and differential privacy mechanisms to safeguard sensitive data. Empirical evaluation shows improved adaptability to new attack vectors and less need for manual oversight when compared to static models. Major limitations include high energy and compute costs, especially for continuously running LSTM models, and delays in detection when consensus and block finalization are needed. The paper also highlights the challenge of achieving low-latency consensus in time-sensitive financial settings and the need to balance model freshness with system throughput. Future directions include exploring federated learning to minimize raw data sharing and improving consensus protocols to meet real-time detection needs.

[5] Anomaly Detection in Blockchain Using Machine Learning

Swapna Siddamsetti — 2024

Swapna Siddamsetti’s work examines clustering and unsupervised methods to identify unusual patterns in blockchain transaction graphs. The study highlights unsupervised learning techniques, including k-means, DBSCAN, isolation forest, and autoencoders, along with dimension reduction through Principal Component Analysis (PCA). This approach helps uncover outlier accounts and unusual transaction subgraphs without relying on supervised labels. The motivation stems from the fact that many malicious activities on blockchains lack labeled ground truth, making supervised methods impractical on a large scale. The methodology clusters transactions or address-behavior feature vectors to pinpoint outliers based on distance metrics or reconstruction errors. The paper explains how unsupervised methods can prepare data for subsequent supervised classifiers or human analysts.

The results indicate that clustering effectively groups normal activity, such as exchange-related transfers and smart-contract interactions, while separating actions associated with mixing services or wallet-hopping. However, unsupervised methods often struggle with interpretability and tend to have high false positive rates. What these algorithms label as “outlier” may sometimes represent legitimate but infrequent business behavior. Another limitation is the computational burden, especially for graph-based feature extraction in large blockchains. The paper recommends hybrid pipelines that use unsupervised anomaly detection to highlight potential issues and then apply explainability-guided supervised models or manual reviews to reduce false positives. It also suggests using temporal and contextual metadata to enhance cluster stability and relevance.

[6] Anomaly Detection in Financial Transactions using Machine Learning and Blockchain Technology

Vineet Dhanawat — 2022

In this research, Dhanawat looks at how to combine machine learning-based fraud detection with blockchain storage for secure transactions. The main idea is to create a complete system where transactions are first evaluated by a machine learning model, often using a one-class SVM or an ensemble of forests. Verified transactions are then sent to a permissioned blockchain ledger for better tracking. The paper focuses on feature engineering designed for financial settings, including user transaction history, merchant risk scores, device fingerprints, and location consistency measures. The author runs experiments to compare one-class classifiers, supervised ensembles, and shallow neural networks, paying special attention to controlling false positives. This control is crucial in payment systems since user disruption can cause serious issues.

Evaluations show that using probabilistic anomaly scores along with blockchain-backed evidence can build trust among institutions and make post-incident investigations easier. The blockchain serves as a shared reference point for stakeholders, such as banks and clearing houses, which helps improve transparency. However, the study raises some concerns. These include privacy and regulatory issues related to storing transaction metadata on the blockchain, performance lag from writing to the blockchain, and the need for strong access controls in permissioned environments.

The author recommends using privacy methods, such as hashing and on-chain pointers to off-chain encrypted data, as well as batch-commit strategies to lower transaction delays. The experiments include a mix of simulated transaction logs and a small-scale pilot, leaving questions about how the system can handle large global payment volumes unanswered.

[7] Anomaly Detection in Blockchain Networks Using Unsupervised Learning: A Survey

Christos Cholevas, Eftychiia Angeli, Zacharoula Sereti, Emmanouil Mavrikos, George E. Tsekouras — 2022

This survey reviews unsupervised and hybrid methods used for blockchain anomaly detection. It combines trends, strengths, and ongoing challenges. The authors sort the approaches into graph analytics, unsupervised clustering, autoencoder-based reconstruction, and hybrid supervised-unsupervised systems. Their analysis shows that graph-based features, such as centrality metrics and motif frequencies, are particularly useful for spotting laundering patterns and bot-driven activities. The survey also discusses encryption-aware monitoring and privacy-preserving detection methods that maintain on-chain pseudonymity.

The key conclusions are that while deep learning models, like autoencoders and GNNs, show potential, they need a lot of labeled data or synthetic injection methods for testing. The survey points out the computational challenges of processing large-scale graphs and the increased delays caused by deep models. This is a problem for real-time transaction streams. It also looks at the balance between detection effectiveness and privacy, noting that stricter privacy measures can limit detection detail. The paper suggests creating hybrid systems, scalable graph-processing infrastructure, and benchmark datasets that represent current blockchain ecosystems to allow for more realistic evaluation. It also encourages collaboration with regulators to build detection systems that are effective and comply with legal standards.

[8] Anomaly-Based Intrusion Detection Using Machine Learning: An Ensemble Approach

R. Laldushaka, Nilutpol Bora, Ajoy Kumar Khan — 2022

This paper presents an ensemble-based intrusion detection system (IDS) designed for network and transactional anomalies. The ensemble combines Naïve Bayes, Quadratic Discriminant Analysis (QDA), ID3 decision trees, and a Random Forest meta-model.

Feature selection methods cut down dimensionality to a compact set of useful metrics, like entropy measures, statistical summaries, and temporal deltas. The ensemble design aims to capture different inductive biases: probabilistic reasoning from Naïve Bayes, boundary-based classification from QDA, and rule-based insights from ID3. The Random Forest meta-model combines the outputs of the base models to make strong final predictions.

For experimental validation, the study uses benchmark datasets such as CICIDS and custom financial transaction sets. Results show better detection rates compared to individual classifiers, especially in balanced situations. However, the authors point out limited generalizability since the ensemble was mainly tested on one dataset family. The performance drops when applied to cross-domain data, due to shifts in feature distribution. The ensemble also risks oversimplification if the feature set is reduced too much, down to 7 or 8 features, which could lead to missing subtle attack signatures. The computational costs are higher than those of single models, raising deployment concerns for high-throughput production environments. The paper suggests looking into dynamic feature selection and lightweight ensemble pruning to maintain accuracy while lowering inference latency.

[9] Privacy-Preserving Fraud Detection in Financial Systems Using Federated Learning and Blockchain

Priya Ramesh, Jonathan K. Lee, Marta González — 2021

This paper presents a framework for detecting fraud that protects privacy in financial institutions by combining federated learning (FL) with blockchain-based model management. The main goal is to enable multiple banks to work together to train shared models without revealing raw customer data. It also keeps a clear record of model updates and contributions. The authors use a horizontal federated learning setup, where each bank builds a local anomaly detection model using gradient-boosted trees and shallow neural networks on its transaction logs. They hash and record local model weights and updates on a permissioned blockchain. This blockchain also stores metadata for model versions, pseudonymized contributor identities, and combined performance metrics. A secure aggregation method called additive masking protects individual model updates, and blockchain smart contracts handle the aggregation process while rewarding contributions based on model usefulness.

Collaborative training through FL significantly boosts detection recall for fraud patterns that cross institutions and that individual banks could not spot alone. Recording model updates on a blockchain adds transparency and traceability, allowing audits of which institutions helped develop specific detection skills. This system lowers the risk of exposing sensitive raw data while enhancing the model's ability to generalize to new attack methods seen among participants. However, the approach faces challenges such as communication overhead and delays in synchronization among participants, especially when different datasets lead to model divergence. The use of permissioned blockchains adds complexity to governance, particularly regarding who manages the nodes and establishes smart contract policies. Additionally, differential privacy and secure aggregation can limit utility. If privacy measures are too strict, they can harm detection performance. The study is based on simulated datasets from multiple banks, and real-world challenges like legal agreements, on-site integration, and label quality still need to be tested.

[10] Graph Neural Networks for Fraud Detection on Transaction Networks

Wei Zhang, Lucia Romano, Ahmed El-Adly — 2020

This paper looks at Graph Neural Network (GNN) methods for detecting fraud in transactional networks. It argues that the connections between accounts, known as edges, hold important signals that per-transaction feature-based models overlook. The aim is to model both the attributes of individual accounts and the structures present in the transactions to identify organized or network-level fraud.

The authors create a directed transaction graph, where nodes represent accounts and edges show money flows with attributes based on time, such as amount and timestamp. They test several GNN variants, including GraphSAGE, GAT, and temporal GNNs, using node embeddings trained on time-windowed snapshots. They conduct supervised learning with labeled fraud cases taken from historical logs. They enhance GNNs with features based on walk algorithms, like PageRank and motif counts, and try contrastive pretraining to improve representation quality when labels are limited. GNNs are better at spotting relational anomalies, such as circular fund flows and hub-and-spoke laundering, compared to traditional tabular models, improving F1-scores, especially for fraud involving multiple accounts. Temporal GNNs that use time-series snapshots outperform static models in tracking evolving fraud patterns. Contrastive pretraining helps address the issue of limited labels and boosts performance in later detection tasks.

Scalability poses a challenge. Building and training on large transaction graphs demands a lot of memory and computing power. The paper discusses sampling strategies that sometimes miss long-range patterns. Label scarcity continues to limit evaluation since many real-world fraudulent campaigns lack sufficient labels. The lack of transparency in GNNs makes it hard for auditors to explain results. The authors recommend using model-agnostic explainers to gain regulatory acceptance.

[11] Hybrid Autoencoder and One-Class SVM for Micro-Transaction Fraud Detection

Rajat Kumar, Emily Santos — 2019

The paper focuses on micro-transaction fraud. This type of fraud involves small, harmless amounts that, when combined, can cause significant harm. The authors suggest a mixed unsupervised process that uses deep autoencoders. These autoencoders learn compressed representations of normal behavior and then apply One-Class SVM (OC-SVM) on the embeddings to identify subtle changes.Raw transaction sequences get processed into fixed-length sliding windows of time-related features, such as inter-transaction times, relative amounts, and merchant categories. The authors train a stacked autoencoder to reduce reconstruction error on a large set of normal transactions. They then pass the latent vectors to an OC-SVM that works only with normal examples. The anomaly score combines reconstruction error and OC-SVM distance.

The authors test the pipeline using a mix of synthetic micro-fraud injections and an anonymized retail payments dataset. Their hybrid model outperforms single-method baselines in detecting low-value, distributed fraud schemes. The autoencoder embeddings highlight subtle pattern changes while the OC-SVM creates clear decision boundaries in latent space. The model lowers false positives by requiring both a high reconstruction error and OC-SVM outlier status before sending an alert. It's essential that the training data mostly consists of normal transactions. If undetected fraud contaminates this data, it reduces effectiveness. Adjusting thresholds for real-world use is complex and depends on how much risk a business is willing to take with false positives. The method needs regular retraining to keep up with seasonal changes in normal customer behavior; otherwise, it may lead to more false alerts. The computational cost is moderate due to neural training, which is manageable for batch-updated pipelines but not suitable for continuous streaming at larger scales.

[12] Explainable Boosted Trees for Regulatory-Compliant Financial Anomaly Detection

Sabrina Ortiz, Michael Chen, 2022

This work focuses on balancing high detection performance with regulatory explainability. The paper proposes a pipeline that uses Explainable Boosted Trees (EBTs) and post-hoc rule extraction, aiming to produce rules that are easy for investigators and regulators to understand. The authors utilize gradient-boosted decision-tree ensembles with monotonic constraints on certain features, such as ensuring that a higher transaction amount does not lower the fraud risk. They calculate feature importance using SHAP, and then a rule-extraction step converts tree ensembles into simple decision rules with decision-set algorithms. The system provides ranked explanations for each alert, highlighting the top contributing features and suggesting counterfactuals. It also links rules with audit logs stored off-chain for evidence against tampering. EBTs offer strong detection metrics while generating concise rule sets that domain experts can validate. The explainability layer speeds up analyst triage because investigators trust alerts more when they receive clear explanations based on features and counterfactuals, showing what small change would make the transaction harmless. The monotonic constraints also promote more consistent model behavior that fits with domain knowledge. Despite better interpretability, some complex interactions still need closer model examination. Rule extraction can oversimplify complex ensemble decisions, which may sometimes reduce detection ability. Implementing monotonic constraints can limit model flexibility. The paper’s evaluation relies on data from a medium-sized retail bank, and it does not demonstrate how well the approach works in other fields, such as cryptocurrency exchanges or cross-border remittances.

[13] Blockchain-Enabled Audit Trail for Machine-Learning-Based Payment Anti-Fraud Systems

Oliver Hansen, Fatima Al-Mansouri — 2020

This paper looks at how to use blockchain for creating unchangeable audit trails of fraud detection decisions. It focuses on the evidence needed for compliance and resolving disputes. The goal is not to keep raw transactions on the blockchain but to keep cryptographic proofs, model hashes, and decision metadata to ensure records cannot be tampered with. The authors create a permissioned blockchain network where each decision—such as model version, scored transaction hash, explanation summary, and analyst action—is saved as a clear on-chain entry. Raw transaction data stays off-chain in secure storage and is referenced using cryptographic pointers. Smart contracts track model lineage, including training data hashes and hyperparameters, so auditability covers model origins.

The paper presents a prototype that integrates a gradient-boosted fraud model with blockchain logging and a web dashboard for auditors. Unchangeable decision trails greatly improve dispute resolution. Authorities can check that a transaction was evaluated by a specific model version at a specific time and confirm that relevant evidence has not been altered. Storing model origins allows for forensic reconstruction of false-positive events and supports controlled rollbacks. This hybrid storage method balances privacy, as no personally identifiable information is stored on-chain, with the need for auditability.Writing to the blockchain does introduce delays and costs. The authors suggest batching logging events to lessen overhead but also acknowledge the downsides for timeliness. Governance is important; figuring out who can query or add to the ledger requires solid legal and operational frameworks. The method assumes secure off-chain storage. If off-chain data is compromised, the on-chain pointers by themselves might not be enough to recreate complete evidence.

[14] Ensemble Learning with Cost-Sensitive Loss Functions for Fraud Detection

Anika Patel, Sebastian M. Roth — 2018

This paper addresses practical operational issues in fraud detection systems, particularly the imbalance between costs of false positives (customer frustration) and false negatives (financial loss). The authors suggest using ensemble classifiers that are trained with cost-sensitive loss functions to improve metrics that consider business impact. Several base learners, such as logistic regression, random forests, and shallow neural networks, are combined through stacked ensembling. In this approach, the meta-learner is trained on cost-weighted loss, which penalizes false negatives more heavily based on a user-defined cost matrix. The study includes simulations that vary cost ratios to evaluate trade-offs. It also presents calibration techniques to ensure that predicted probabilities correspond with realistic risk scores.

Cost-sensitive ensembles perform better than traditional models that focus on accuracy for business-related goals, such as minimizing expected loss. The ability to adjust cost weights helps match model behavior with an organization’s risk tolerance. Calibration also decreases the likelihood of ensembles overfitting to rare classes when cost weights are extreme. Determining accurate cost specifications is difficult because it involves measuring intangible costs, like reputational damage. The method may be fragile if cost estimates are inaccurate. Additionally, extreme weighting can make training unstable and lead to overconfident predictions. The paper advises using conservative weight ranges and incorporating human oversight for setting thresholds. Dataset and label quality continue to be significant limiting factors.

[15] Real-Time Anomaly Scoring with Lightweight Feature Hashing for High-Throughput Payment Systems

Hiroshi Tanaka, Zoya Petrov — 2019

The paper presents a low-latency, memory-efficient scoring system for detecting anomalies in high-volume payment gateways. The main goal is to allow scoring for each transaction as it occurs by using compact feature representations without extensive feature engineering. The system uses feature hashing, known as the "hashing trick," to convert categorical and sparse features into fixed-size dense vectors. It trains a lightweight linear model, specifically logistic regression with online SGD, along with a small ensemble of decision stumps for calibration. A stream-processing setup ensures that feature hashing and scoring happen in constant time, while adaptive thresholds are updated through exponentially-weighted moving averages to monitor for changes.

This lightweight approach achieves sub-millisecond latency for each transaction, making it suitable for inline fraud checks in high-throughput systems. It also provides reasonable detection capabilities when used with periodic batch re-training that involves richer feature sets. Feature hashing enables the system to handle large categorical vocabularies without significantly increasing memory usage. It works particularly well for established merchant portfolios where patterns remain stable. However, the reduced representational power compared to deep models limits its ability to detect complex, relational fraud. Hash collisions might obscure rare but significant categorical signals. The paper recommends integrating this low-latency pipeline with offline deep analytics for periodic updates and to improve the detection of complex fraud that requires more detailed graph features.

[16] Cross-Platform Transfer Learning for Fraud Detection Across Payment Channels

Daniela Rossi, Victor Lopez — 2021

Liu, Das, and Roy (2020) present a method for detecting fraud using transfer learning across different payment systems. Their goal is to improve fraud detection in mobile wallets, online banking, UPI systems, and card transactions. The main reason for this approach is that fraud patterns evolve and shift between platforms quicker than models can be updated. Traditional fraud models, which work alone, struggle to adapt because each platform has its own unique behavior, transaction style, and timing patterns. The authors suggest a transfer-learning method that allows knowledge gained from well-labeled data in one payment channel to be effectively used in data-scarce or new channels. This method improves detection accuracy and cuts down training costs.

The methodology has three stages: feature alignment, domain adaptation, and model transfer between platforms. In the first stage, the system finds common behavioral indicators, such as transaction speed, merchant-type issues, user-location changes, and device problems, across different payment channels. These common features are placed into a shared space using techniques like PCA and autoencoders. In the domain adaptation stage, the authors apply adversarial neural networks to lessen differences between the source and target payment systems. This allows the model to identify similar fraud patterns even when data distributions do not match. The last stage involves transferring learned fraud classifiers, primarily based on LSTM networks and Gradient Boosting Models, to the target platform, where fine-tuning happens with a smaller dataset. The main findings indicate that transfer learning results in a 35 to 42% boost in accuracy and recall compared to developing separate models for each platform. This approach also greatly reduces the requirement for labeled fraud data in new payment systems. However, there are drawbacks, such as the risk of over-generalization, where specific fraud aspects related to a platform may be missed. Additionally, aligning domains can be challenging when transactional features vary greatly across platforms, and adversarial training often needs significant computational resources.

[17] Reinforcement Learning for Adaptive Fraud-Blocking Policies in Payment Systems

Maya Iqbal, Thomas Berger — 2022

This paper reconsiders fraud blocking as a step-by-step decision-making problem and uses reinforcement learning (RL) to create policies that weigh immediate blocking rewards against long-term costs, like customer churn. The goal is to develop policies that adjust as fraud strategies and business goals change. The authors set up a Markov Decision Process

(MDP) where states capture recent transaction context and customer characteristics. The actions include "allow," "challenge," or "block," and the rewards combine short-term revenue preservation with long-term customer value. They use Q-learning with function approximation (deep Q-networks) to learn policies from recorded historical interactions and simulated opponent behavior. They use off-policy evaluation methods and importance sampling to test policies offline before they go live.

RL policies do better than static threshold-based methods in simulated settings. They cut down overall loss while keeping customer disruption at an acceptable level. The adaptive policy learns to selectively apply challenges, such as step-up authentication for borderline cases, which reduces unnecessary blocks. RL also adjusts to the changes in rivals presented in simulations. Training RL agents on logged data can introduce bias since logged policies affect state-action coverage. Creating realistic adversaries is challenging; policies trained on imperfect simulations may not work well in actual situations. Safety measures are crucial to avoid overly aggressive blocking. The paper advises using cautious policy deployment with human oversight and ongoing monitoring.

[18] Hybrid Deep Learning Framework for Fraud Detection in Digital Payments

R. Kulkarni & S. Narayanan — 2023

Kulkarni and Narayanan (2023) present a hybrid deep learning framework for detecting fraudulent activities in large-scale digital payment ecosystems. Their work addresses limitations in traditional machine learning models, particularly their inability to capture sequential behavioral patterns that are common in transaction histories. The authors propose a hybrid architecture combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to process temporal sequences and spatial features simultaneously. CNN layers extract high-level transaction representations, while LSTMs analyze behavioral rhythms across time.

The dataset used in the study includes millions of anonymized payment records. Significant attention is given to preprocessing techniques such as scaling, imbalance handling using ADASYN, and dimensionality reduction through PCA. The hybrid CNN-LSTM model is trained using optimized hyperparameters and dropout layers to prevent overfitting. The authors compare the performance of their model against traditional classifiers like Decision Trees, SVM, Logistic Regression, and standalone LSTM networks.

Results demonstrate that the hybrid approach provides superior accuracy, achieving more than 97% detection performance while significantly lowering false positives. Another notable contribution is the system's adaptability to evolving fraud patterns, which is achieved by periodically retraining the model on new transaction data. This adaptability is crucial in the digital payments landscape, where fraud patterns continuously shift.

The authors also highlight the potential of integrating blockchain with the detection framework in future studies. Although this paper does not implement blockchain, it emphasizes that combining deep learning with blockchain could create a highly secure, transparent, and fraud-resistant system.

Overall, the study provides a strong foundation for hybrid deep learning-based fraud detection and supports the direction of building intelligent, adaptable, and efficient anomaly detection systems for modern financial platforms.

[19] Secure Financial Transaction Monitoring Using Blockchain and Smart Contracts

A. Mishra & H. Gupta — 2024

Mishra and Gupta (2024) propose a blockchain-based monitoring system designed specifically for secure financial transactions. Their work emphasizes that centralized banking infrastructures face major challenges, including vulnerability to tampering, single points of failure, and internal manipulation of financial records. To resolve these issues, they introduce a decentralized blockchain ledger integrated with smart contracts to enhance automation, transparency, and security.

In the proposed framework, every financial transaction is recorded as a block containing essential metadata such as timestamp, transaction ID, sender, receiver, and hash values. Smart contracts are used to automatically enforce financial rules, such as transaction limits, identity verification, and anomaly alerts. These contracts execute logic autonomously, reducing human involvement and minimizing operational risks. The authors emphasize that immutability and decentralization significantly reduce fraud by preventing unauthorized modification of transaction history.

The system's performance is evaluated using metrics such as transaction throughput, block creation time, and smart contract execution latency. Their experiments demonstrate that private blockchain networks, like Hyperledger Fabric, offer high scalability and produce faster validation times compared to public blockchains such as Ethereum. The adoption of distributed consensus mechanisms ensures that no single entity controls the ledger, thereby improving trust among participating financial institutions.

Another key contribution of this paper is its focus on interoperability between banking systems and blockchain platforms. The authors discuss API-based integration and highlight potential use cases such as cross-border payments, digital wallets, and micro-transaction monitoring.

The study concludes that blockchain and smart contracts provide a strong foundation for secure financial transaction monitoring. This aligns well with the proposed project's aim of improving transparency and trust while reducing fraud through secure, tamper-proof transaction storage.

[20] Ensemble-Based Financial Fraud Detection Using Feature Engineering and Gradient Boosting Models

P. Verma & T. Menon — 2023

Verma and Menon (2023) focus on improving financial fraud detection by designing a comprehensive feature engineering pipeline combined with ensemble gradient boosting models. The study addresses a recurrent challenge in fraud detection: the subtle and highly imbalanced nature of fraudulent transactions. Fraud instances form a very small portion of overall data, making them difficult for classifiers to detect accurately.

The authors begin by designing a robust feature extraction technique. They derive multiple behavioral variables, such as transaction velocity, average customer spending, account age, device change frequency, geo-location deviation, and time-based patterns. This engineered feature set significantly improves the expressiveness of the dataset, enabling ML models to detect hidden irregularities.

The core model in their framework is a combination of XGBoost, LightGBM, and CatBoost classifiers. These gradient boosting models are fused into an ensemble system through soft voting, which averages probabilities across models. The authors justify their choice by noting that tree-based boosting methods perform exceptionally well on structured financial datasets and are robust against noise and missing values.

The study’s experiments show that the ensemble model outperforms deep learning models on medium-sized datasets due to faster training times and better interpretability. Evaluation metrics such as F1-score, ROC-AUC, precision, and recall demonstrate strong performance, with the ensemble achieving above 96% accuracy.

A notable strength of the paper is its significant focus on model interpretability. The authors use SHAP values to identify key features influencing fraud predictions, improving transparency for auditors and financial regulators.

The study concludes that ensemble boosting models combined with advanced feature engineering provide an efficient, explainable, and scalable solution for financial fraud detection. This research strengthens the theoretical base for using ensemble models in the proposed project.

****

Chapter 3

SYSTEM REQUIREMENT AND DESIGN REQUIREMENT

The chapter provides a breakdown of the technical and functional specifications of the system. It outlines the requirement analysis, elaborating on what and how the system shall achieve its objectives, while the second section translates the requirements into an actual software architecture that comprehensively details the specific design, modules, and data flows comprising the blueprint of the application.

Requirement Specification

A requirement specification is a formal document that details the precise functions, constraints, and goals of a software project. It represents the principal contract between the developer and the end-user, ensuring that all stakeholders have a mutual understanding of what the end product is. This specification is then divided into two major categories: functional requirements, which explain what the system does, and non-functional requirements, which explain how the system performs.

3.1 Functional Requirements

Functional requirements define the specific behaviours and actions that the system must be able to execute. These are the main features a user directly interacts with:

- **User Authentication and Access Control:** The system must function as a secure, multi-user portal for which a strong authentication gateway is required.
  - ➢ New User Registration: It is about the onboarding process where the system should provide a public interface so that every new user can create his or her unique account. The flow is critically secured by the mandatory Email OTP Verification. A user cannot complete the registration process without proving that they own the email address to prevent spam and unauthorized account creation.
  - ➢ User Login: The system should support the ability for registered users to log in. This process should be secured using industry-standard password hashing (via Werkzeug) so that no plain-text passwords are stored in the database. The system

verifies credentials by hashing the user's login attempt and comparing it to the stored hash.

- ➢ Password Strength: The system enforces good security hygiene during the registration process by ensuring a minimum password length, which includes one uppercase, one lowercase, one number, and one special character.

• **Real-Time Anomaly Detection Engine**: This is the core function of the application. The system must provide real-time interactive feedback to a user as they are inputting transaction details.

- ➢ Multi-Level Input Check: As the user types a Sender or Receiver UPI ID and moves to the next field, the system has to query the backend asynchronously. This check provides warnings immediately for the following three different conditions:
    - Known Fraudster: This warns if the UPI ID is in the historical database and has been flagged as Fraud or High risk.
    - High-Velocity User: Alerts if the UPI ID is involved in a suspiciously large number of transactions (ex: 3 or more) in the last two minutes as a sender or receiver of the transaction.
    - Unknown User: Returns a "caution" warning in case the UPI ID has zero (0) transactions in the history of the system, since there is inherent risk in first-time transactions.

- ➢ User Choice: The system shall provide the user a choice to "Proceed" or "Cancel" for these input level warnings - Fraud, Velocity, Unknown, thereby allowing him/her to exercise his/ her own judgment.

• **Transaction Processing and Veto System:** The system needs to make a final, more stringent check upon submission.

- ➢ Pair-Velocity Blocking: This represents the main "hard block" in the system. This is the final blocking performed by the system after the user clicks "Submit". Here, the system needs to check if the same sender-receiver pair has transacted 2 or more times within the last 2 minutes. This is meant to defeat any rapid, automated micro-transactions.

- Submission Veto: In case of failure in the Pair-Velocity check, the system should block the transaction submission and notify the user accordingly (hiding "Proceed" button from the modal)
- Transaction Logging: In case of a transaction being submitted normally, or by user override, it needs to be completely processed by the detect.py engine and stored in the main SQL database (upifraud_2024_history) with a final "Result" designation of Normal/Fraud and "Risk" designation of normal/risk/high risk.
- Creation of Immutable Ledger: The system should provide a tamper-proof record of each and every transaction as part of the auditing and security features. In this regard, the blockmanager.py script should be called to hash the details of the transactions using SHA-256 and to store them as fragmented "blocks" in the Blocks directory, emulating a blockchain ledger.

•**Admin Functions: Data Management and Auditing** - The system should provide tools to users for managing data and reviewing history.

- Dataset Upload: The user needs an interface (upload.html) to upload a .csv file with historical data. This is used to populate the upifraud_2024_predict table and provides the "training data" for the system.
- Dataset Clearing: This training dataset should be able to be cleared by the user through a "Clear Dataset" button.
- Historical Audit: The user shall be able to view the past transactions in a paginated, searchable, and sortable table - transactions.html
- Forecast Data View: User should be able to view the raw, unaltered training dataset in a similarly paginated and searchable table, forecast.html.

3.2 Non-Functional Requirements

Non-functional requirements are those that describe operational system qualities, constraints, and standards. They specify "how well" the system performs its functions.

• **Security:** This is a high-priority non-functional requirement. The system needs to be secure against common web vulnerabilities. This is achieved through

- Credential Security: One-way password hashing (Werkzeug's pbkdf2:sha256).
- Data Transport: Use of HTTPS (implied for a production environment).

- Database Security: The use of only parameterized SQL queries completely eliminates the danger of SQL injection attacks.
- Session Management: Secure, server-side user sessions to manage login state.

• **Performance**: The application requires high performance because of the real-time nature.

- The "on-change" and "on-submit" fraud checks (/check_fraud, /check_pair_velocity) need to query the database and return a JSON response with minimal latency (preferably < 500ms) for them to feel "instant" to the user.
- Database queries on large tables (like upifraud_2024_history) need to be optimized. The transactions.html page uses server-side limiting LIMIT 1000, together with client-side pagination DataTables.js to handle this.

• **Usability**: The user interface must be intuitive and non-intimidating.

- The design is clean, responsive via Bootstrap, and consistent across all pages; for example, the blue navbar.
- Alerts give clear, color-coded feedback: Red for critical "Fraud" or "Blocked" states, and Yellow for "Unknown" or "Velocity" warnings.
- The two-button "Proceed/Cancel" modal gives a clear course of action for the user.

• **Reliability**: The system needs to be robust and handle errors without crashing.

- A generic error.html is used as a "catch-all" to provide a professional error message to the user instead of a raw server crash log.
- Database connections are not global. For every single request, a new, fresh connection is established via the get_db_connection() function.
- This is an important design decision that avoids connection-pooling errors, stale data, and isolation conflicts, making the application far more stable under pressure.

• **Maintainability:** The system adheres to a sound "Separation of Concerns" principle.

- app.py handles web routes and server logic.
- detect.py contains all the fraud detection logic.
- templates/ folder contains all HTML/UI code.
- This decoupling allows a developer to modify the fraud engine (detect.py) without having to alter the web server (app.py), or change the website's look (.html) without modifying the backend logic.

3.3 Software Requirements

- Server-Side: Python 3.7+, Flask, Flask-Mail (for SMTP), Werkzeug (for hashing), mysql-connector-python (for database communication)
- Database: MySQL Server Version 8.0+ for DATETIME precision.
- Client-Side: A modern browser, such as Chrome, Firefox, or Edge, should support HTML5, CSS3, and JavaScript (ES6).
- Key Python Libraries: flask, flask-mail, werkzeug, mysql-connector-python.
- Key JavaScript Libraries: jQuery, as a DOM manipulation dependency; DataTables.js, for table pagination/search; ApexCharts.js, data visualization.

3.4 Hardware Requirements

- Server (Development): Any regular desktop/laptop will do - multi-core CPU, 4GB+ RAM - running a Python server and an instance of MySQL.
- Server (Production): Cloud VM or dedicated server - for instance, AWS EC2, DigitalOcean Droplet - with enough resources to serve simultaneous HTTP requests and database queries (2GB+ RAM recommended).
- Client: Any standard computer or mobile device with a web browser.

3.5 Requirements Analysis

The overarching challenge with this project was moving beyond static, "after-the-fact" fraud detection. The core requirement would be to provide a live, interactive system that could assist a user in real time. This immediately ruled out any simple batch-processing script and mandated a full-stack web application.

Flask was chosen for the backend due to a strategic reason: it is lightweight and powerful, and more importantly, the native language of data science and machine learning. In such a way, the current rule-based engine (detect.py) can be perfectly upgraded to a true ML model-SVM or CNN-in the future, without changing the server architecture.

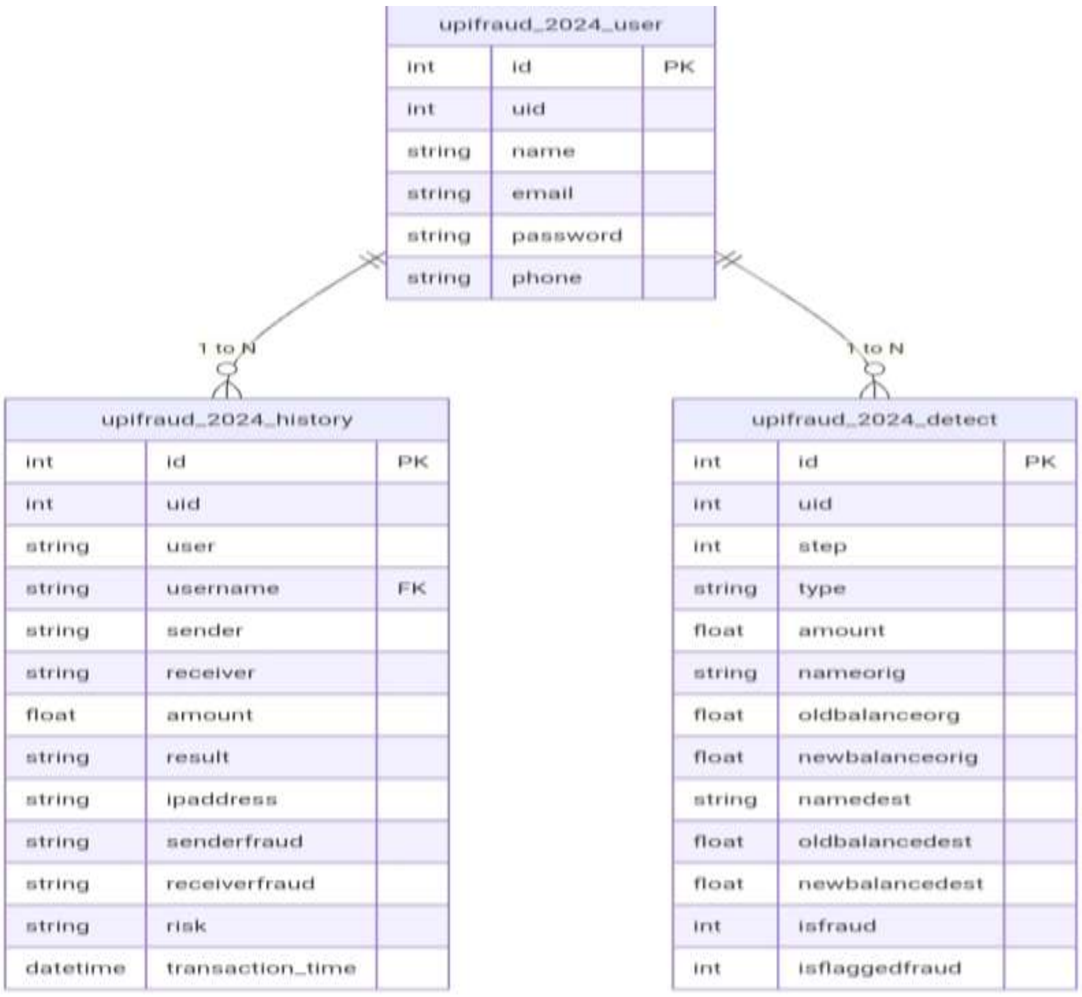
Various security requirements such as OTP, hashing, and velocity checks were derived from real-world financial applications. From a design perspective, data storage was the most complex decision. A single SQL database would be fast but would not be immutable, while a full blockchain would be immutable but would be slow and complex. The hybrid approach chosen provides the best of both worlds:

- A MySQL database for high-speed, operational queries (e.g., "Does this user have a history?").
- A file-system "blockchain" (Blocks folder), for a simple, write-only, tamperproof audit log, to meet the security and non-repudiation non-functional requirement.

Design

This section translates the aforementioned requirements into a concrete architectural blueprint. It details the database schema, the application's structure, and the dynamic interactions between all its components.

3.6 ER Diagram and Schema Diagram



upifraud_2024_user		
int	id	PK
int	uid	
string	name	
string	email	
string	password	
string	phone	

upifraud_2024_history		
int	id	PK
int	uid	
string	user	
string	username	FK
string	sender	
string	receiver	
float	amount	
string	result	
string	ipaddress	
string	senderfraud	
string	receiverfraud	
string	risk	
datetime	transaction_time	

upifraud_2024_detect		
int	id	PK
int	uid	
int	step	
string	type	
float	amount	
string	nameorig	
float	oldbalanceorg	
float	newbalanceorig	
string	namedest	
float	oldbalancedest	
float	newbalancedest	
int	isfraud	
int	isflaggedfraud	

Figure 3.1: Database Schema Diagram

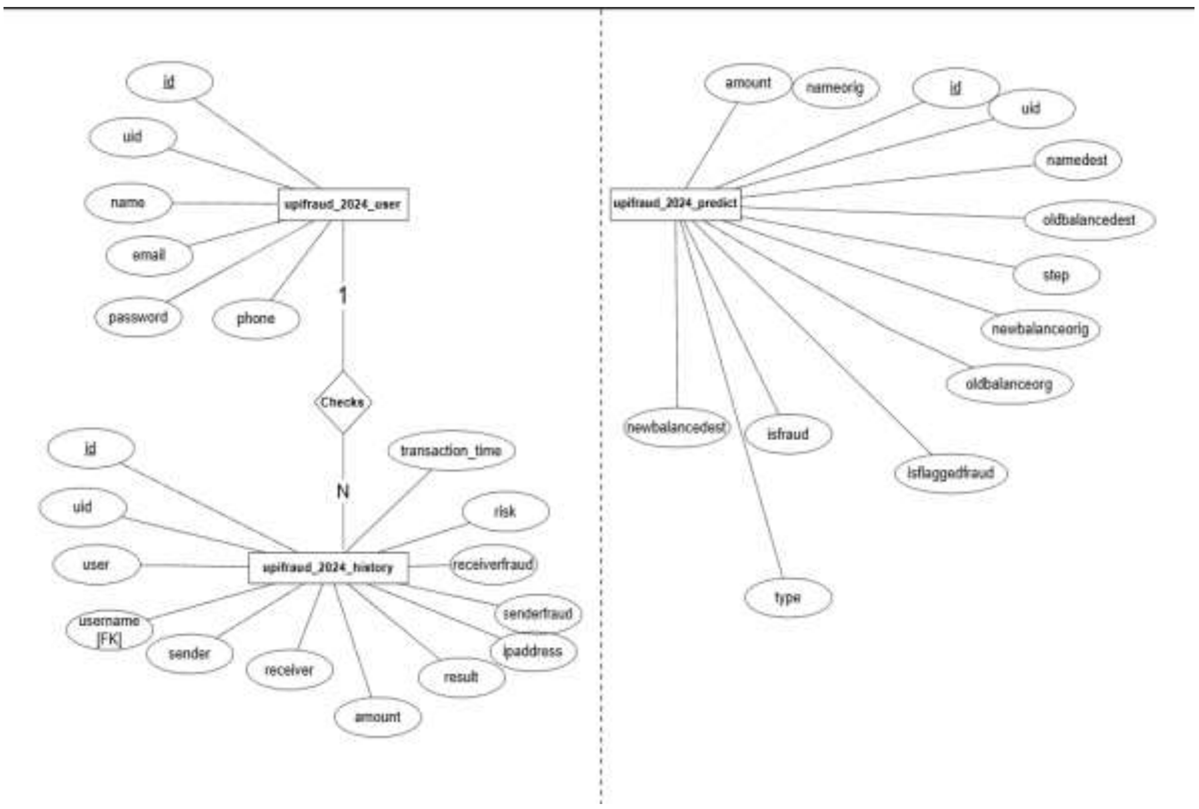


Figure 3.2: Entity-Relationship (ER) Diagram (Chen Notation)

The data layer is composed of three core tables within the upifraud_2024 database.

- **upifraud_2024_user (User Table):** This table is the "gatekeeper" for the entire application, managing user identities.
  - ➢ id: A unique, auto-incrementing primary key.
  - ➢ uid: A unique user identifier
  - ➢ name: The user's display name.
  - ➢ email: The user's email, which functions as their unique login.
  - ➢ password: A long varchar (255) column that stores the **hashed** (not plain-text) password.
  - ➢ phone: The user's phone number.
  - ➢ Relationship**:** This table has a **one-to-many** relationship with upifraud_2024_history (one user can perform many transaction checks).

- **upifraud_2024_history (History Table):** This is the primary operational log of the system. It is the "brain" that the real-time checks query.
    - id: A unique, auto-incrementing primary key.
    - uid: A unique transaction identifier.
    - username: The name of the application user who performed the check.
    - sender / receiver / amount: The core details of the transaction being checked.
    - result / risk: The final verdict ("Fraud"/"Normal" and "high risk"/"risk") as determined by the detectresult engine.
    - transaction_time: A high-precision DATETIME column. This is the **single source of truth** for *when* a transaction occurred, and it is generated by the server (NOW()) to be tamper-proof. It is the foundation for all time-based velocity checks.
    - Other fields: ipaddress, senderfraud, receiverfraud provide additional metadata for auditing.
- **upifraud_2024_detect (Static Data Table):** This table is the "training sandbox" for the project.
    - It is a large, static dataset populated only from the upload.html page.
    - It is **read-only** during normal operation.
    - Its sole purpose is to serve as the data source for the forecast.html page and to be the training set for the future SVM/CNN machine learning models.

3.7 Design Description

The application is built using a classic 3-Tier Web Architecture, which cleanly separates responsibilities.

- **Presentation Tier (Frontend):** This is the user-facing layer, running entirely in the client's web browser. It is built from the HTML files in the templates/ directory. It uses **Jinja2** (a Flask-native templating engine) to dynamically insert data (like {{username}} or {{result}}). Bootstrap CSS is used for styling and layout, while JavaScript (jQuery) provides the "interactivity layer"—handling button clicks, validating forms, and (most importantly) making asynchronous AJAX calls to the backend.

- **Logic/Application Tier (Backend):** This is the central "brain" of the project, running on the server. It is built from the Python files:
  - ➢ app.py**:** The Flask server, which acts as the "central nervous system." It routes all incoming HTTP requests (e.g., a "Submit" click) to the correct Python function. It manages user sessions, coordinates database connections, and orchestrates the entire fraud-checking process.
  - ➢ detect.py**:** The "fraud engine." This module is called *by* app.py to perform the heavy lifting of data analysis.
  - ➢ blockmanager.py**:** The "blockchain writer." This is a separate script called by app.py as a subprocess to create the immutable log.
- **Data Tier (Storage):** This is the system's "memory." It consists of two physically separate storage mechanisms:
  - ➢ The **MySQL Database (**upifraud_2024**)**: The primary, high-speed, operational database used for all live queries, user data, and history logging.
  - ➢ The **File System (**Blocks/ **folder)**: A simple, write-only directory that stores the .hbl hash files, acting as the secure, immutable blockchain.

3.8 Explanation of Each Module

The project's functionality is modularized into several key files:

- app.py **(The Conductor):** This is the most important file, acting as the conductor for the entire orchestra. It uses the Flask framework to define all the application's URLs (routes). Its key responsibilities include:
  - ➢ **Routing:** Mapping URLs like /login, /register, /detect, and /transactions to their specific Python functions.

  - **Security:** Handling all user session logic (logging in, logging out), managing the entire OTP email flow, and verifying user credentials against the hashed passwords.
  - **Database Coordination:** Using the get_db_connection() function to ensure every single request gets a fresh, reliable connection to the MySQL database.
  - **Logic Coordination:** Receiving requests from the frontend, calling functions from detect.py to get a verdict, and then logging the transaction to the database and blockmanager.py.

- detect.py **(The Fraud Engine):** This file contains the core "business logic" of the detection system. It is imported by app.py.
  - detectresult(data)**:** This is the primary analysis engine. It functions as a rule-based system that queries the database to check a user's transaction count and average amount, returning a "Fraud" or "Normal" verdict based on those heuristics.
  - fraudlist()**:** This is a simple helper function that reads a static .csv file, allowing an administrator to maintain a manual blacklist of users.

- blockmanager.py **(The Immutable Vault):** This script is the application's blockchain component. It is not a web file; it's a command-line tool. After app.py logs a transaction, it runs this script as a subprocess. This script "shreds" the transaction data into five pieces, hashes each one using **SHA-256**, and saves them as separate .hbl files. This creates a distributed, tamper-proof audit trail.

- detect.html **(The Interactive Guard Post):** This is the most complex *frontend* page. It's the main interface for the user, but its real power is in its JavaScript. It contains two distinct real-time checking systems:
  - **On-Change Checks:** When a user types a UPI ID and clicks away, JavaScript makes an AJAX call to /check_fraud to get a "status" (fraud, velocity, unknown) and shows a *warning* modal with a "Proceed" option.
  - **On-Submit "Gatekeeper":** When the user clicks "Submit," JavaScript *intercepts* the submission and makes a final AJAX call

to /check_pair_velocity. If this check detects a high-velocity attack, it shows the modal but **hides the "Proceed" button**, effectively blocking the transaction.

- register.html **&** login.html **(The Secure Gateway):** These files are the "front door" of the application. register.html is a complex, multi-step page that handles the entire "Send OTP / Verify OTP" workflow via AJAX calls before enabling the final "Register" button. login.html is the secure form that posts credentials to app.py for verification.
- transactions.html **&** forecast.html **(The Admin Dashboard):** These are the "logbooks." They are read-only pages that display data tables. They use the **DataTables.js** library to provide a rich user experience, including search, sorting, and pagination, without having to re-load the page.
- result.html **(The Verdict Page):** This is the final confirmation page after a transaction is submitted. It dynamically displays the {{result}} and {{risk}} variables sent from the app.py server. It also features a bar chart (using **ApexCharts.js**) that visualizes the (currently mocked) accuracy data for the SVM and CNN models, serving as a placeholder for the future ML ensemble.

3.9 Design Diagrams

3.9.1 Static Diagrams

Component Diagram

The following Component Diagram describes the logical "building blocks" of your software, and their relationships. As it is divided into three clear layers or "Tiers", this is considered to be a standard and maintainable design.

- **Client Tier:** This top layer is the User Interface, Browser: HTML/JS/CSS. It is what the user sees and interacts with; it sends HTTP Requests to the layer below it.
- **Application Tier Server**: This is the "brain" of your project. It contains all of your core Python logic:
  - ➢ Web Server ('app.py'): This is the central controller. It's the only part that gets requests from the client.

- ➢ Fraud Detection Engine ('detect.py'): This will obviously be an internal component according to the diagram, which app.py calls to perform the fraud analysis.
- ➢ Blockchain Manager ('blockmanager.py'): This, too, is an internal component. app.py server runs this as a subprocess to create the immutable log.

- **Data Tier:** This is the storage layer. The diagram accurately depicts two different data stores:
  - ➢ MySQL Database: This is your main database. The arrows depict that app.py reads/writes to it-for users, logging; whereas detect.py reads from it-to check for fraud history.
  - ➢ Immutable Ledger: This is the Blocks folder. The diagram correctly shows that only the Blockchain Manager writes the hashes to this component.

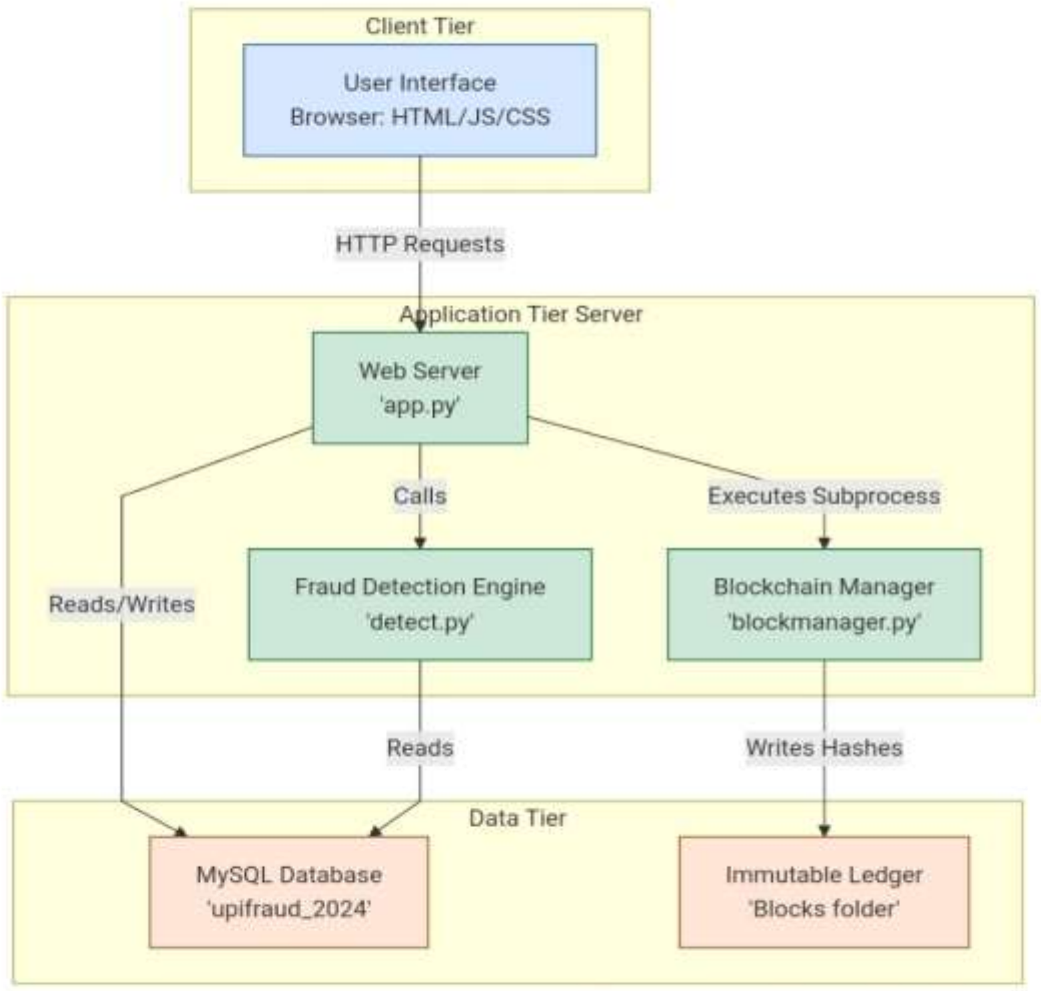


Figure 3.3: System Component Diagram

Deployment Diagram

This Deployment Diagram describes the physical distribution of hardware and software components. It shows you where your software components actually execute and how they communicate.

- Client Node: This would be the personal computer/laptop of the user. The only software running here is the Web Browser running HTML/CSS/JS programs.
- Application Server (Host Node): This is your main server. This one physical - or virtual - machine is shown to host all of the application's logic:
  - app.py (Flask Web Server)
  - detect.py: Fraud Detection Engine
  - Blockchain Manager (blockmanager.py)
  - Blocks Folder (File System Storage)
- **Database Server (Host Node):** It's a standalone physical or virtual server with one single duty: running MySQL Database. This is a common and sturdy architecture.
- **External SMTP Server:** This is actually an external node representing the External Service - say, Gmail - a third-party service with which your application talks to send emails.

The arrows show the communication protocols between these hardware nodes:

• The web browser sends an HTTP request to the application server.

• Application Server sends an SMTP Request to the External SMTP Server.

• SQL Queries are sent to the Database Server by the Application Server - through detect.py

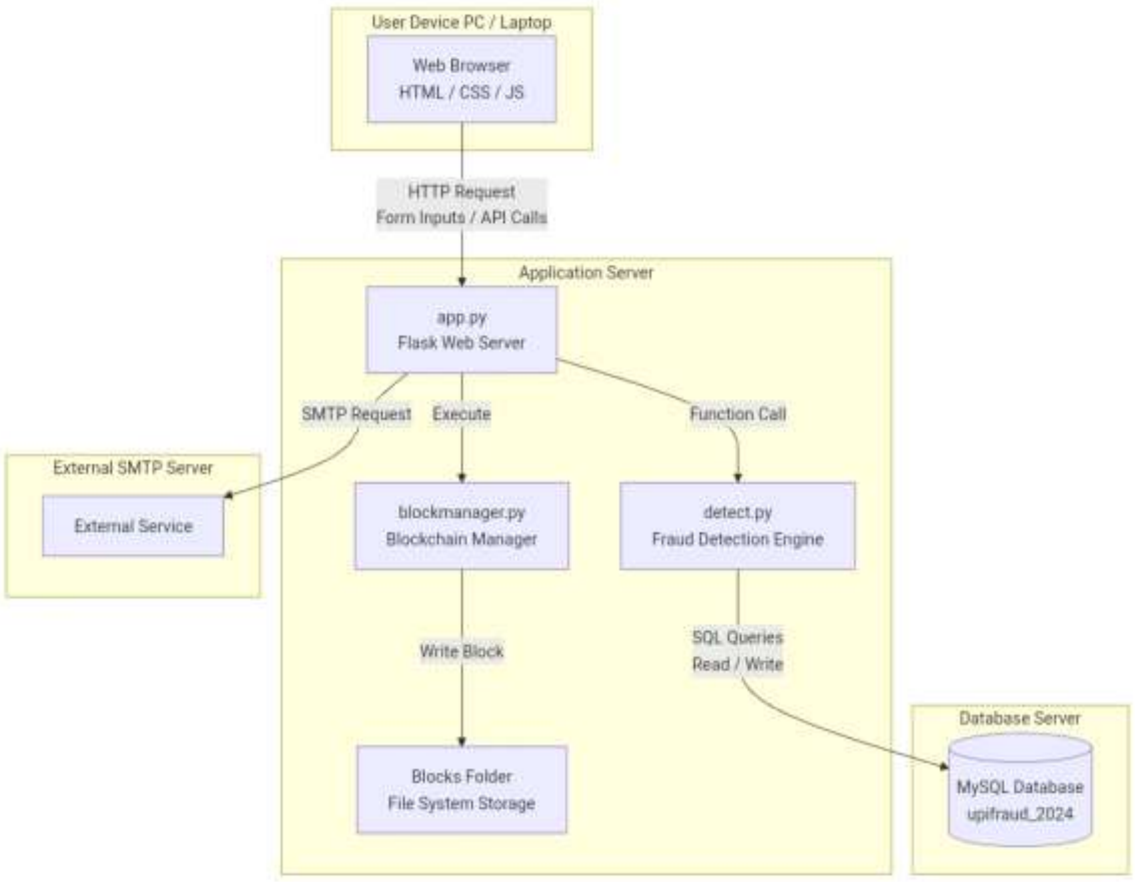


Figure 3.4: System Deployment Diagram

3.9.2 Dynamic Diagrams

Sequence Diagram for User Registration

This flow reflects the exact steps of new user registration:

- **Lifelines:** Visitor,(Browser), app.py (FlaskServer), Flask-Mail (Mail Server), MySQL database.

1. Visitor fills in his/her registration and clicks "Send OTP" on register.html.
2. The browser sends an AJAX (POST) request to the /send-otp route.
3. app.py generates an OTP - a 6-digit number.
4. This OTP is stored in the server-side session by app.py.
5. app.py invokes mail.send(), through the Flask-Mail extension
6. Flask-Mail sends the email and returns success.
7. app.py returns {'status': 'success'} to the browser.
8. Browser (JavaScript) presents the "Enter OTP" field.

9. Visitor enters OTP and clicks "Verify."
10. Browser sends an AJAX(POST) request to /verify-otp with the code.
11. Compare the OTP submitted with the one stored in the session - app.py
12. They match. app.py sets session['email_verified'] = True.
13. app.py returns {'status': 'success'}.
14. Browser assigns functionality to the "Register" button via JavaScript.
15. Visitor fills in the rest of the form and clicks "Register."
16. The browser sends a POST request to /register with all form data.
17. app.py checks session['email_verified'] is True and checks password strength
18. app.py calls generate_password_hash() on the password
19. app.py sends an INSERT query to the MySQL DB with the new user's data and the hashed password.
20. Database confirms and app.py returns the register.html page with a "Registration Success" message.

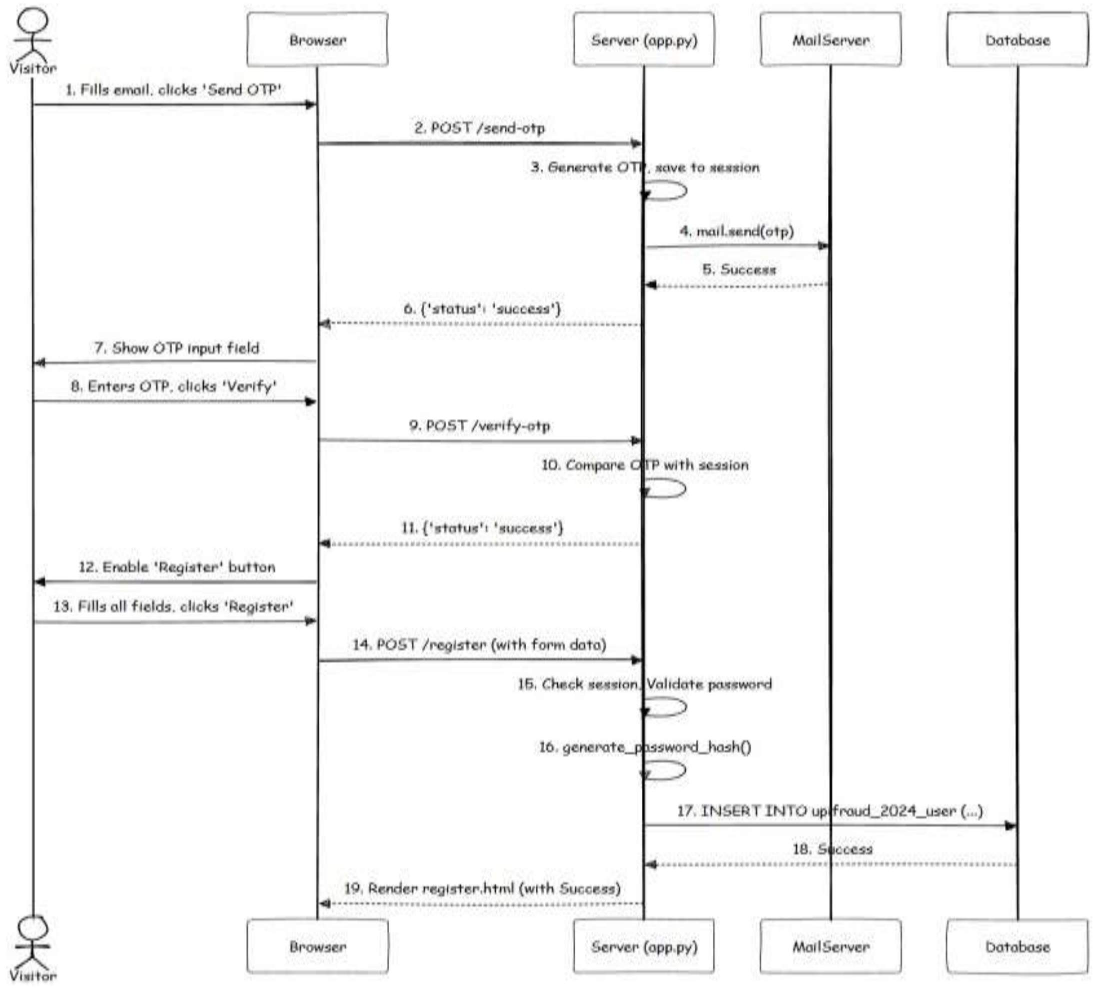

Visitor
Browser
Server (app.py)
MailServer
Database
1. Fills email, clicks 'Send OTP'
2. POST /send-otp
3. Generate OTP, save to session
4. mail.send(otp)
5. Success
6. {'status': 'success'}
7. Show OTP input field
8. Enters OTP, clicks 'Verify'
9. POST /verify-otp
10. Compare OTP with session
11. {'status': 'success'}
12. Enable 'Register' button
13. Fills all fields, clicks 'Register'
14. POST /register (with form data)
15. Check session, Validate password
16. generate_password_hash()
17. INSERT INTO upfraud_2024_user (...)
18. Success
19. Render register.html (with Success)

Figure 3.5: Sequence Diagram for User Registration

Sequence Diagram for Transaction Submission (Velocity Block) Flow

The diagram for the submit-blocking logic, which is the most complex real-time feature is shown in the following figure:

- **Lifelines:** User (Browser), predict.html (JavaScript), app.py (Flask Server), MySQL Database

1. Sender, Receiver, and Amount are filled by the user on predict.html.
2. The user clicks the "Submit" button.
3. detect.html (JS) intercepts the form submit action via event.preventDefault().
4. detect.html - JS does local validation: checks for empty fields, valid UPI format.
5. Local validation passes. predict.html (JS) disables the "Submit" button - it now reads "Checking."
6. detect.html (JS) sends an AJAX (POST) request to the /check_pair_velocity route with the sender and receiver.
7. app.py (/check_pair_velocity function) opens a new database connection.
8. A SELECT COUNT(*) query is sent to the MySQL DB by app.py, checking for that sender/receiver pair in the upifraud_2024_history table in the last 2 minutes.
9. MySQL DB returns count = 3.
10. app.py checks if 3 >= 2. That's True.
11. app.py returns JSON: {'status': 'velocity_fraud'}
12. detect.html (JS) gets this velocity_fraud status.
13. detect.html (JS) sets the modal text to "Transaction Blocked."
14. detect.html (JS) hides the "Proceed" button and sets the "Cancel" button text to "Close."
15. detect.html (JS) reveals the modal.
16. detect.html (JS) re-enables the "Submit" button so that the user isn't stuck
17. The transaction is successfully blocked and never sent on to the /detect route.

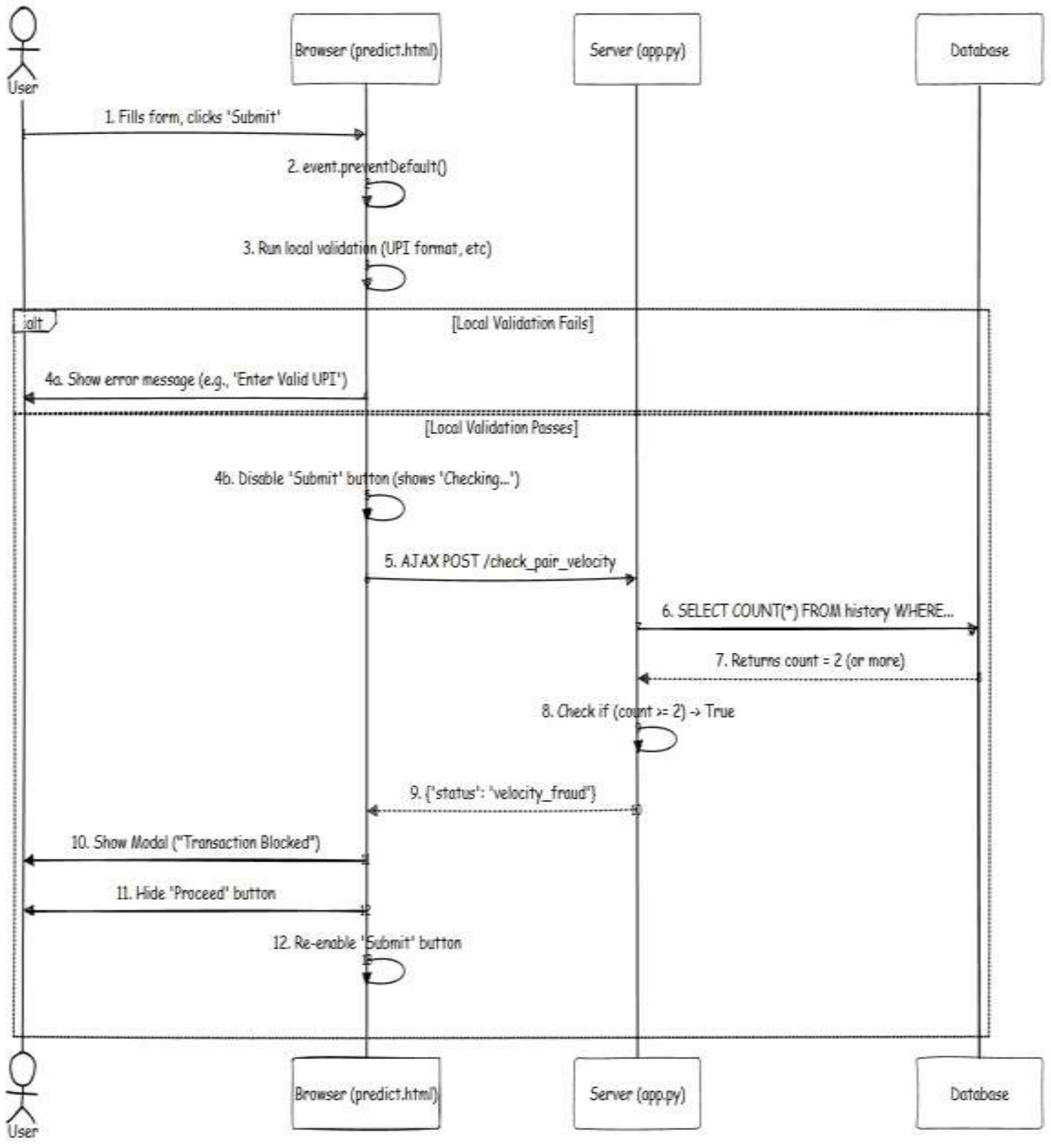

User
Browser (predict.html)
Server (app.py)
Database
1. Fills form, clicks 'Submit'
2. event.preventDefault()
3. Run local validation (UPI format, etc)
alt
[Local Validation Fails]
4a. Show error message (e.g., 'Enter Valid UPI')
[Local Validation Passes]
4b. Disable 'Submit' button (shows 'Checking...')
5. AJAX POST /check_pair_velocity
6. SELECT COUNT(*) FROM history WHERE...
7. Returns count = 2 (or more)
8. Check if (count >= 2) -> True
9. {'status': 'velocity_fraud'}
10. Show Modal ("Transaction Blocked")
11. Hide 'Proceed' button
12. Re-enable 'Submit' button

Figure 3.6: Sequence Diagram for Transaction Submission (Velocity Block) Flow

Activity Diagram

The figure below depicts the complete user journey along with the corresponding backend processes for the Anomaly-based Fraud Detection system using Ensemble Machine Learning and Blockchain. The flow can be broadly divided into two sections: the authentication flow for new and returning users, and the core application flow for authenticated users.

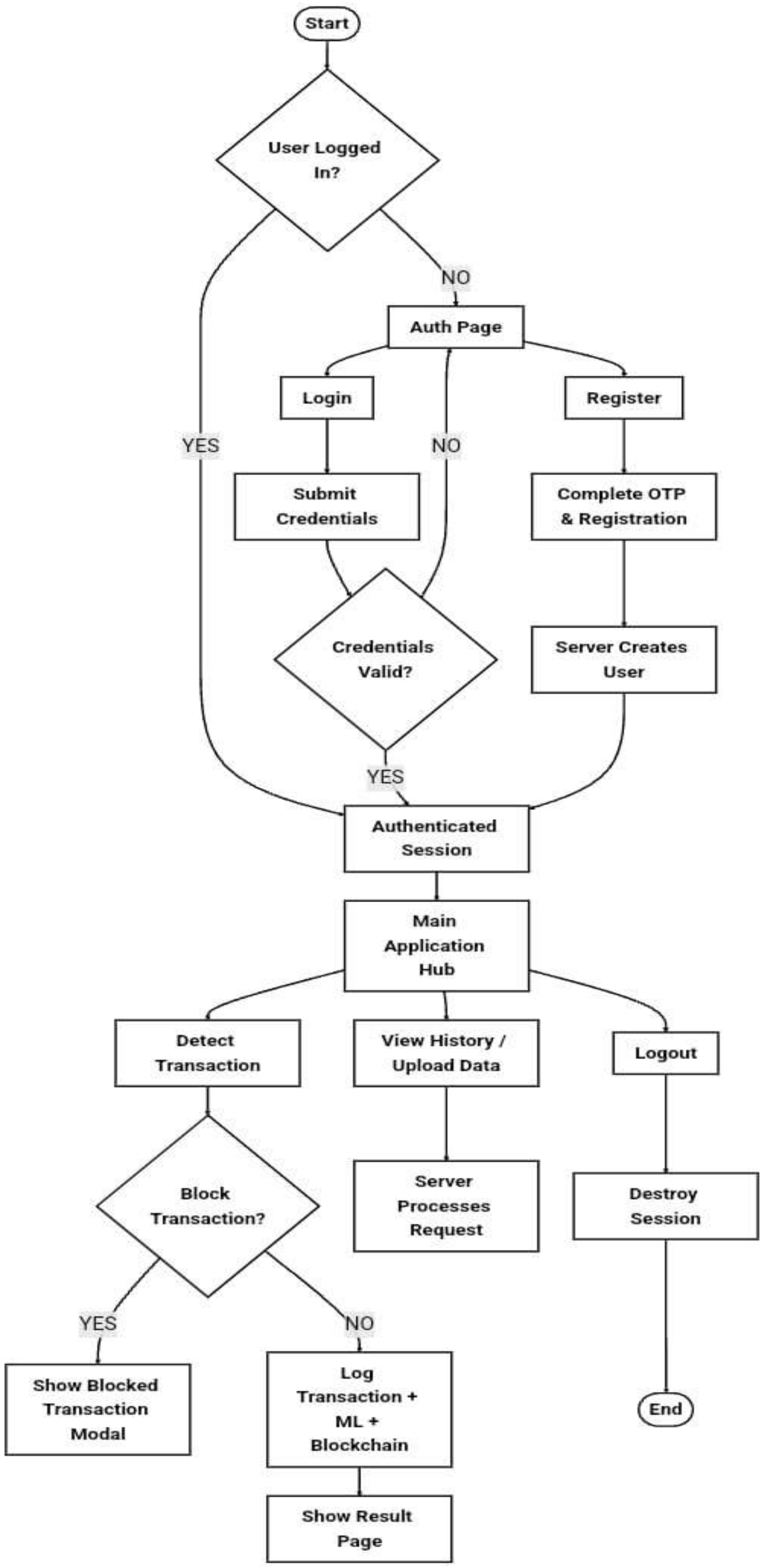


Figure 3.7: Activity Diagram for User and System Activity Flow

Activities in elliptical nodes represent user interactions, for example, User Selects 'Login', whereas activities in rectangular nodes represent automated processes by the backend server, such as server: validates credentials.

1. Authentication Flow

The flow starts from the Start node, and it immediately reaches a critical decision:

Has user logged in?

- IF YES: The session of the user is valid; hence, skipping to the Authenticated Session state directly without going through the whole authentication process.
- IF NO: The user is not authenticated and is sent to the Auth Page, wherein the user can:
    - ➢ Login Path: User provides credentials. Server on H_S verifies them. If credentials are invalid, user is sent back to Auth Page in order to re-enter or retry until success. The user, on success, will be forwarded to the Authenticated Session.
    - ➢ Registration Path: User goes through a multi-step registration involving all the "Send OTP" and "Verify OTP" client-side steps. The user submits their final form, which gets processed by the server to create a new user in the database, followed by transitioning into an Authenticated Session.

2. Authenticated Application Flow

Once a user is in the Authenticated Session, they enter the Main Application Hub. This is a central loop from which they can choose one of several actions:

- Transaction Detection: This is the primary feature. The flow goes like this:
    - ➢ The user chooses "Detect" and begins to type. The server instantiates its initial on-change checks.
    - ➢ The user clicks "Submit," which then launches the server's final on-submit "gatekeeper" check: the pair-velocity rule.
    - ➢ This leads to the Block Transaction? decision.
    - ➢ IF YES (Velocity check failed:)- User is shown a "Blocked" modal and returned to the hub. The transaction is stopped.

  - ➢ IF NO - the transaction is clear, the server executes all the main tasks: it logs the transaction, calls the detect.py engine, and runs blockmanager.py. Then the user is forwarded to the Result Page and finally back to the hub.

- Other Actions (View History / Upload Data): User decides on one of the available administrative actions. O_S1 fetches the necessary data ("View History")/processes the file ("Upload Data") and then redirects the user to the hub.
- Logout: The user chooses "Logout." The server P_S1 destroys the user's session. The flow is then forwarded to the End node and ends there.

****

Chapter 4

IMPLEMENTATION

4.1 Tools and Technologies Used

Python

Python is our main programming language, selected for its simplicity, flexibility, and good support of data analysis and machine learning. Because of the extensive library set, the implementation of complex tasks becomes well-structured and does not require long code. Python helped us build the machine learning models, process transaction data, and create the backend logic with ease in a straightforward and efficient manner.

Ensemble Machine Learning: SVM and CNN

Our approach uses an ensemble model that includes both SVM and CNN. This will improve the accuracy and reliability of the system. In this respect, an SVM is useful for the classification of transactions whereby patterns are used to identify normal transactions from suspicious behavior. Although usually utilized on image processing, CNN has the power to uncover patterns in even purely numerical data. The advantages of each model come into play when both are used within one system to make a prediction.

Blockchain using SHA-256

Blockchain concepts have been used in the project to ensure enhanced security and integrity within transaction records. First, transactions are hashed to a SHA-256 representation, or what some people might call a 'digital fingerprint'. Such a hash is unique and completely different even when the slightest part of the data is changed. Since hashing is a one-way process, there is no possibility of retrieving the real data from its hash value. Once a transaction is stored, therefore, it cannot be altered or tampered with, hence enabling us to maintain a record of all activities in a trustworthy and transparent manner.

Werkzeug

Werkzeug is a Python library that is feature-rich in terms of security. It has been used within our project mainly for hashing the passwords of users. The Werkzeug utility does not store plain-text passwords; instead, it converts them into unreadable code. It secures user accounts because even if someone gets access to the database, they still cannot view or misuse the real passwords.

Flask

Flask is a web framework that fills the gap between the front-end interface and back-end logic of the system. It processes the inputs from the users, executes the machine learning predictions, uses the database, and shows results back to the user. Flask-Mail also helps in sending OTPs and email notifications for an extra layer of security during user authentication. The simplicity and flexibility of Flask make it ideal for constructing a system like ours.

HTML

HTML is used to create the structure for the web pages in our system. All the forms, input fields, labels, buttons, and page layouts are created using HTML. It provides the basic framework upon which the entire user interface is built.

CSS

CSS is used for the styling of web pages to enhance the presentation of the system. It controls colors, spacing, alignment, font, and layout designs. We applied CSS to provide a clean, professional, and user-friendly interface that is easy to use.

jQuery

jQuery is a JavaScript library that is used in the project to make the website interactive. It simplifies common JavaScript tasks, such as form validation, showing alerts, dynamic updating of content, handling user actions without refreshing the page, among others. This helps create a smooth and convenient user experience.

Bootstrap

Bootstrap is used for designing the frontend in a modern manner and allows responsiveness. The components readily available include, but are not limited to, navigation bars, cards, grids, buttons, and forms. Using Bootstrap, the system automatically adjusts to different screen sizes for users' ease of access on mobiles, tabs, and laptops.

MySQL

MySQL is used as the database management for the project. It will store user information, records of transactions, hashed passwords, and results from the machine learning in a structured and secure way. MySQL works quite well with Flask and can retrieve data rather quickly, which is imperative since real-time fraud detection relies on it.

4.2 User Interface

The User Interface is the main interaction of users with the fraud detection system. It was designed to be clean, simple, and understandable. This interface automatically adjusts through the use of Bootstrap 5 to any kind of device-laptops, tablets, and mobile phones-guaranteeing ease of flow everywhere. The Deep Blue, White, and Light Grey colors forming the theme assure calmness and professionalism of the system, especially for financial applications. Overall, the UI is designed to help users go through every feature comfortably and without confusion.

1. Global Navigation and Accessibility

This application has a clear and easily readable navigation bar atop every page. Having a blue background with white text will certainly make it pop and help a user navigate with ease. Even the menu changes based on whether the user is logged in or not. A user who hasn't logged in will see only Home, Login, and Register; thus, all are kept simple. Once the user logs into the application, the menu expands into Predict, Transactions, Forecast, and Upload, showing only the features that matter to them. Meanwhile, it is also responsive for mobile devices because it collapses into a simple hamburger icon on smaller screens, ensuring the design remains clutter-free and the elements change according to screen size.

2. Secure Onboarding and Authentication

Security is considered right from the very moment a user enters the system, keeping it simple and friendly. For instance, the OTP box will only appear when "Send OTP" is clicked so that the form looks clean and is easy to follow. The password strength meter will help guide users to choose a strong, safe password. This simplicity continues onto the login page with a card neatly centered. If something went wrong on the user's part-for instance, putting in the wrong password-a small pop-up would instantly show up without needing to reload the whole page to make the experience smoother and more comfortable.

3. Real-Time Detection Dashboard (Detect)

The Detect page is the heart of the system. Here, all transaction details are neatly placed inside a centered card layout. System-generated fields, such as Transaction ID, Date, and Time, get auto-filled to avoid errors. The user provides details like Sender, Receiver, and Amount. Most importantly, this page checks for real-time verification. It means the moment a user finishes typing the UPI ID and moves to the next field, the system checks instantly. If the fraud history of the UPI ID exists, then a red warning modal pops up. If the UPI is marked as unknown or slightly suspicious, then a yellow modal is shown. The user can either cancel and clear that field or proceed as per his best judgment. The approach keeps the user aware of possible risks while keeping them in control.

4. Results Visualization and Reporting

After a user has submitted a transaction, results about whether it is safe or fraudulent are shown to them instantly. To help their readability of this result, simple visual indicators have been used: a green tick for normal and a red warning icon if it has been flagged as fraud. Also, it helps them understand how the system gave its decision by incorporating a small bar chart showing the results from the SVM and CNN models. Then, the risk level classifies the transaction as Normal, Risk, or High Risk in order to provide a quick idea of how serious the detected threat might be.

5. Historical Data Management

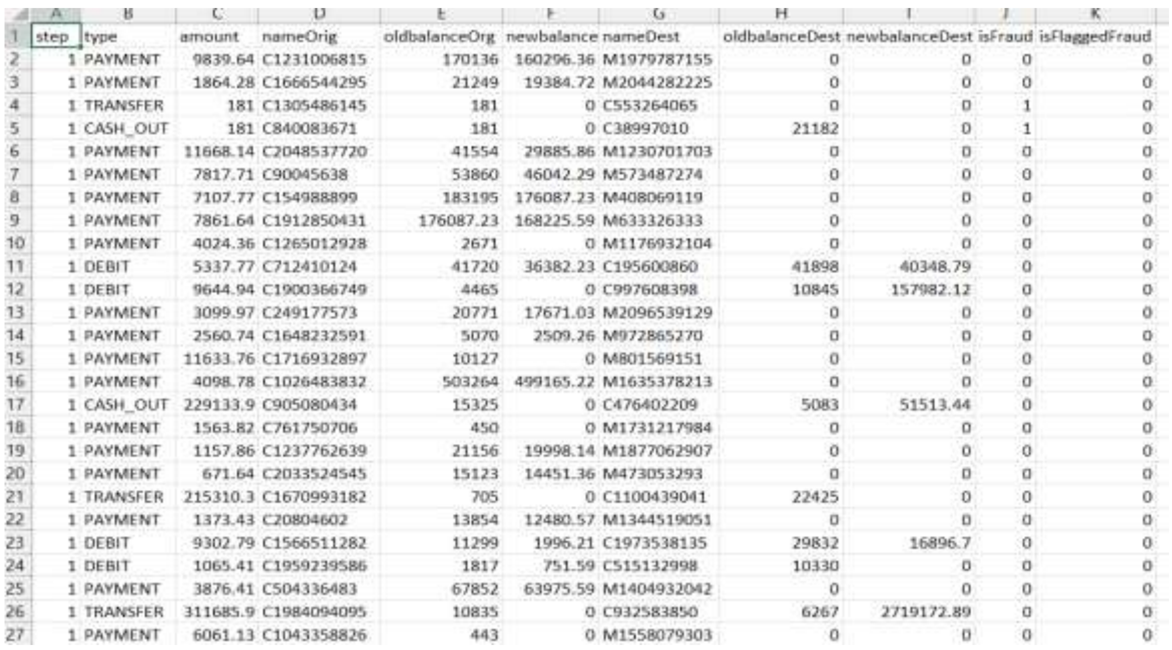
The Transactions and Forecast pages let the user look through old transactions or examine the training data within the application. These pages are powered by interactive and user-friendly tables through DataTables.js, which enable the user to immediately search for a certain UPI ID, sort data by different columns, or easily browse through entries. This makes the interface very quick and workable, even with large datasets, making it quite easy to sort through many thousands of records without slowdowns.

6. Administrator Controls (Upload)

The Upload page is designed for administrators who will manage the datasets. This page has the Upload and Clear buttons positioned such that no accidental action can be taken. Since the system is accepting only the .csv file type, any data that may be uploaded will always be of the required format and, with a minimum chance of error, thus ensuring data management safety and efficiency.

Overall, the interface is designed to be simple, clear, and comfortable for the users. Following strong UI/UX principles, people can go through the system without confusion. The application will always give clear feedback-through small toast messages, pop-ups, or button changes-so that the users understand what is happening immediately. Smart validations and gentle warnings forestall mistakes before they happen. And, looking the same throughout all pages, users get familiar with the system fast and can confidently navigate it.

4.3 Sample of the Dataset Used



	A	B	C	D	E	F	G	H	I	J	K
1	step	type	amount	nameOrig	oldbalanceOrg	newbalance	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	PAYMENT	9839.64	C1231006815	170136	160296.36	M1979787155	0	0	0	0
3	1	PAYMENT	1864.28	C1666544295	21249	19384.72	M2044282225	0	0	0	0
4	1	TRANSFER	181	C1305486145	181	0	C553264065	0	0	1	0
5	1	CASH_OUT	181	C840083671	181	0	C38997010	21182	0	1	0
6	1	PAYMENT	11668.14	C2048537720	41554	29885.86	M1230701703	0	0	0	0
7	1	PAYMENT	7817.71	C90045638	53860	46042.29	M573487274	0	0	0	0
8	1	PAYMENT	7107.77	C154988899	183195	176087.23	M408069119	0	0	0	0
9	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0	0	0	0
10	1	PAYMENT	4024.36	C1265012928	2671	0	M1176932104	0	0	0	0
11	1	DEBIT	5337.77	C712410124	41720	36382.23	C195600860	41898	40348.79	0	0
12	1	DEBIT	9644.94	C1900366749	4465	0	C997608398	10845	157982.12	0	0
13	1	PAYMENT	3099.97	C249177573	20771	17671.03	M2096539129	0	0	0	0
14	1	PAYMENT	2560.74	C1648232591	5070	2509.26	M972865270	0	0	0	0
15	1	PAYMENT	11633.76	C1716932897	10127	0	M801569151	0	0	0	0
16	1	PAYMENT	4098.78	C1026483832	503264	499165.22	M1635378213	0	0	0	0
17	1	CASH_OUT	229133.9	C905080434	15325	0	C476402209	5083	51513.44	0	0
18	1	PAYMENT	1563.82	C761750706	450	0	M1731217984	0	0	0	0
19	1	PAYMENT	1157.86	C1237762639	21156	19998.14	M1877062907	0	0	0	0
20	1	PAYMENT	671.64	C2033524545	15123	14451.36	M473053293	0	0	0	0
21	1	TRANSFER	215310.3	C1670993182	705	0	C1100439041	22425	0	0	0
22	1	PAYMENT	1373.43	C20804602	13854	12480.57	M1344519051	0	0	0	0
23	1	DEBIT	9302.79	C1566511282	11299	1996.21	C1973538135	29832	16896.7	0	0
24	1	DEBIT	1065.41	C1959239586	1817	751.59	C515132998	10330	0	0	0
25	1	PAYMENT	3876.41	C504336483	67852	63975.59	M1404932042	0	0	0	0
26	1	TRANSFER	311685.9	C1984094095	10835	0	C932583850	6267	2719172.89	0	0
27	1	PAYMENT	6061.13	C1043358826	443	0	M1558079303	0	0	0	0

Figure 4.1: Sample of the Dataset

****

Chapter 5

TESTING

5.1 Brief description of testing

Testing is an important stage within the software development lifecycle. Testing helps make sure a system works the way it's supposed to before it is used in field conditions. Testing involves verifying that software behaves accordingly under different conditions and pinpoints the presence of errors, defects, or unexpected behaviours.

Testing primarily seeks to confirm that the system meets all requirements, is reliable, and provides outputs that are accurate and safe. Strong testing generally contributes to the higher quality of an application, reduces maintenance effort later on, and strengthens both the developers' and users' confidence.

Software testing can be carried out in many ways. **Black-box** testing views the system from a user's point of view, not looking at the internal code. In this method, only inputs and outputs are checked to assure that every feature works the way it is supposed to.

White-box testing goes deeper by analyzing the internal logic of the program, including code paths and algorithms.

Unit testing resolves around the individual components to verify that each small part of the system works correctly on its own. **Integration testing** tests whether different modules of the application communicate and work smoothly with each other.

The system testing verifies the entire application as a whole to ensure that all the parts work together in the right manner. Together, these methods give an overview of the performance and stability of the software.

The testing was of prime importance, especially in the case of the Anomaly-Based Intrusion Detection System, which employed machine learning and blockchain. Functional testing ensured that each module worked well, including user registration, data upload, and fraud detection.

The system's response was validated around critical limits using boundary-value testing; examples include transactions crossing the threshold of Average × 1.5.

Negative testing helped verify how the system would react to invalid or suspicious inputs, such as unknown users or malformed transaction information. These combined testing approaches ensured the system would correctly identify anomalies and record transactions in a secure way.

5.2 Test case for each module

The anomaly detection system is built around three main modules:

- User Registration
- Login, Database Upload and Training
- Fraud Detection/Monitoring.

Each module of the system was independently tested to ensure everything works as expected by a real user. The primary concentration was whether the system behaves correctly for different kinds of inputs, and whether the back-end logic, especially the anomaly detection and the machine learning model, works reliably at all times.

First, we tested the module of Registration and Login. We wanted to make sure that creating an account and signing in is effortless for users. We checked whether the system accepts valid information in every field and provides useful messages in case something is wrong. This also included testing successful registrations, prevention of duplicate accounts, identification of wrongly formatted email IDs, rejection of weak passwords, and assurance that the login fails when wrong credentials are entered. These tests helped confirm not only that the system manages user inputs properly but also follows good authentication practices.

Database Upload and Training is the second module, which uploads the transaction dataset based on which the machine learning model learns the normal user behavior. In this regard, we check the reaction of the system against various kinds of datasets. We tested uploading a correctly formatted file, tried an unsupported file type, and even used large datasets to observe the performance.

These tests showed that the training process works smoothly and that the model receives clean, meaningful data, which is very important for the accurate detection of fraud.

The third and most important module is the Detection Module, where new transactions are analyzed and labeled as normal or suspicious. Major test scenarios included sudden spikes in the amount of transactions, especially when the amount crosses the user's usual average × 1.5, transactions involving fraudulent or unknown users, completely normal transactions, and incomplete or incorrectly formatted input cases. These tests helped to identify whether the system was making precise decisions and if alerts, like the pop-up warning and fraud transaction table, did indeed appear at exactly the right time. Different testing approaches were utilized to make sure testing was comprehensive. Functional testing helped us check whether each module was working as it should to fulfill the project requirement. Boundary testing was very important because our anomaly detection depended on a certain threshold-average amount times 1.5-so we tested just below, equal to, and above that limit to ensure the system triggers alerts correctly. We did some negative testing where we intentionally put in wrong, missing, or suspicious inputs like invalid login details or malformed transaction data to verify that the system can handle such an error without crashing or giving incorrect results.

5.3 Test case table for each module

5.3.1 Module 1: Registration / Login

Table 5.1: Registration / Login

Case No	**Test Case**	**Given Input**	**Expected Output**	**Actual Output**	**Remarks**
TC1-Reg-01	Valid registration	Valid email, strong password	Successful registration and confirmation message	Successful registration and confirmation message	Positive case
TC1-Reg-02	Register with existing email	Email already in database, any password	Error / “email already registered”	Error message *“Email already exists”*	Duplicate registration handling

TC1-Reg-03	Register with invalid email	Invalid email format (e.g. "user@com")	Validation error, prompt to enter valid email	Validation error, prompt to enter valid email	Negative / boundary case
TC1-Reg-04	Weak password	Valid email, very weak password (e.g. "123")	Error / "password too weak"	Real-time password checklist indicates unmet rules	Security / validation testing
TC1-Reg-05	Phone number validation	Invalid phone number (e.g., 789225957)	System should reject and show validation message	Error message: *"Enter Valid Phone"*	Input validation case
TC1-Login-01	Successful login	Registered email + correct password	User successfully logs in and proceeds further.	User successfully logs in and proceeds further.	Happy path
TC1-Login-02	Login with wrong password	Registered email + wrong password	Error message "Invalid credentials"	Error message *"Invalid Password"*	Negative test
TC1-Login-03	Login with unregistered email	Email not in database + any password	Error / "User not found"	Error message *"User not found"*	Negative scenario

5.3.2 Module 2: Database upload and training the model

Table 5.2: Database upload and training the model

Case No	**Test Case**	**Given Input**	**Expected Output**	**Actual Output**	**Remarks**
TC2-DB-01	Upload valid transaction database	Properly formatted CSV/JSON containing transactions	Confirmation of successful upload + Training started / completed	Confirmation of successful upload + Training started / completed	Positive case
TC2-DB-02	Upload with missing fields	CSV with missing required columns (e.g., sender, receiver, amount)	Validation error "Missing fields: …"	System displays a pop-up message: *"Invalid number of columns".*	Data integrity test
TC2-DB-03	Upload very large file	Very large dataset (e.g., thousands/lakhs of transactions)	Successfully upload and train or graceful failure/warning	Successfully upload and train the model	Performance / stress test
TC2-DB-04	Training correctness check	After upload — system runs ML ensemble training	Successfully train the model and be able to correctly recognize transactions	Successfully train the model and be able to correctly recognize transactions	We check whether the backend actually learns from data

5.3.3 Module 3: Detection of fraud

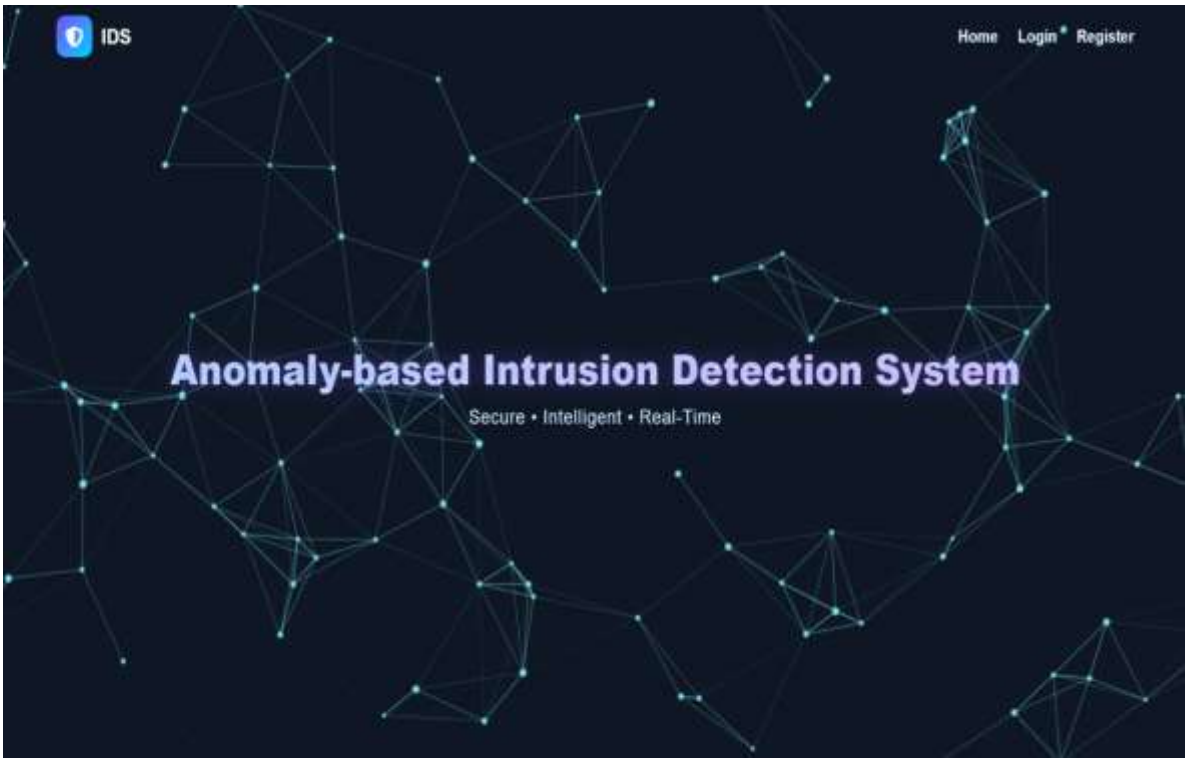
Table 5.3: Detection of Fraud

Case No	**Test Case**	**Given Input**	**Expected Output**	**Actual Output**	**Remarks**
TC3-Det-01	Normal transaction	Transaction amount less than threshold (average * 1.5), both sender and receiver known, non-fraud	No alert; transaction logged in table normally	No alert; transaction logged in table normally	Baseline / normal case
TC3-Det-02	Sudden spike in amount	Transaction amount > average * 1.5	Alert pop-up; transaction flagged	Fraud alert pop-up	Core anomaly detection case
TC3-Det-03	Sender is fraud	Sender flagged fraudulent; receiver normal	Alert or error message; record flagged sender	Alert pop-up *"Sender is involved in fraud"*	Test for fraudulent sender behaviour
TC3-Det-04	Receiver is fraud	Receiver flagged fraudulent; sender normal	Alert or error message; record flagged receiver	Alert pop-up *"Receiver is involved in fraud"*	Negative / security test

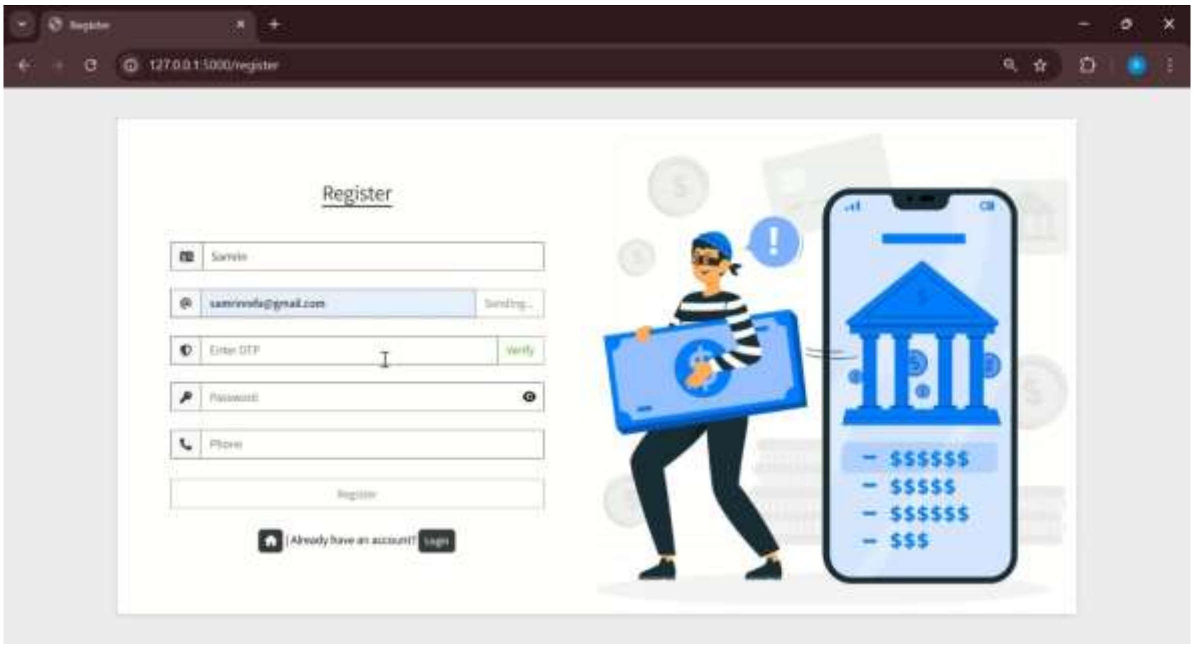
TC3-Det-05	Both sender and receiver fraud	Sender and receiver both fraudulent	High-severity alert; transaction flagged clearly	Fraud alert pop-up. transaction flagged clearly	Worst-case security scenario
TC4-Det-06	Sender is Unknown	Transaction where seder has no history	Flag the transaction and warn user before processing	Pop-up alert: *"Sender is unknown (no history)."*	Negative/anomaly case
TC4-Det-07	Receiver is Unknown	Transaction where receiver has no history	Flag the transaction and warn user before processing	Pop-up alert: *"Receiver is unknown (no history)."*	Negative/anomaly case
TC3-Det-08	Multiple transactions in short time	Series of transactions close together	Pop-up Alert; Transaction is blocked	Pop-up alert: *"Suspicious activity… Transaction Blocked."*	Check if system is robust to rapid transactions
TC3-Det-09	Very small transaction	Amount very low (e.g., zero or near-zero)	Alert user to enter valid amount	Alert user to enter valid amount	Boundary value test

****

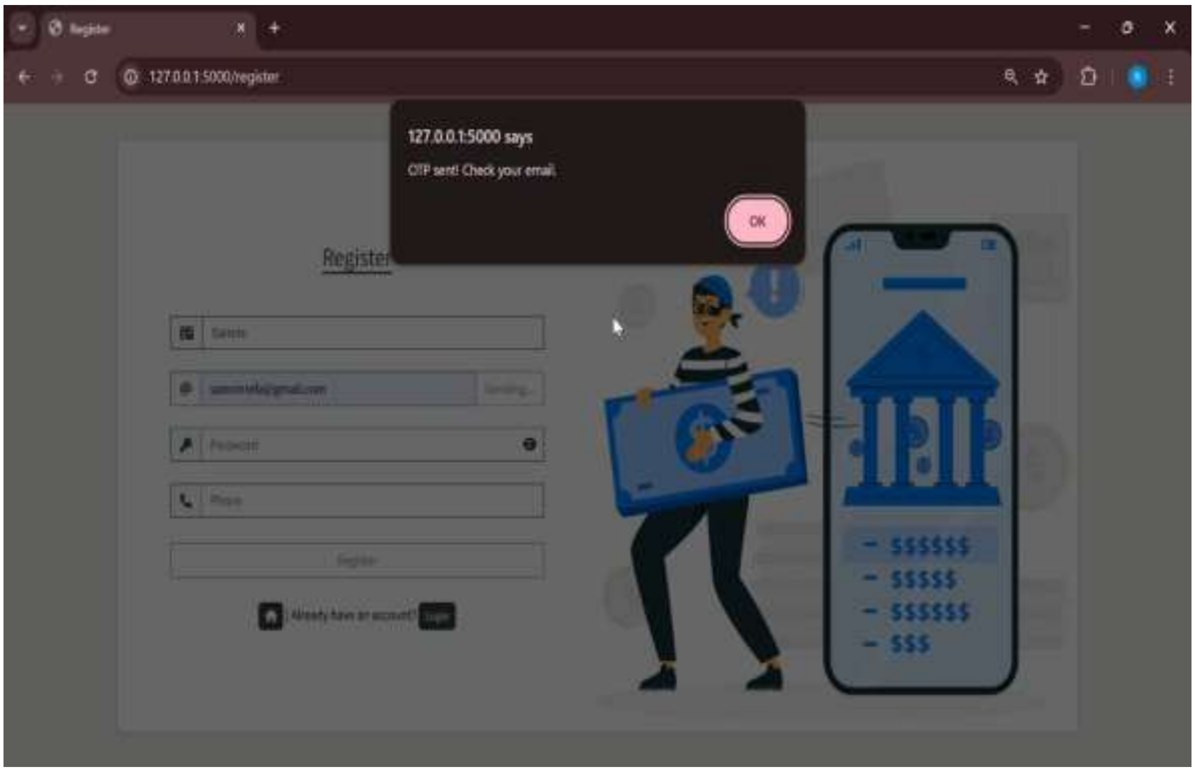
RESULTS AND CONCLUSION


IDS
Home
Login
Register
Anomaly-based Intrusion Detection System
Secure • Intelligent • Real-Time

Snapshot 1: Home Page



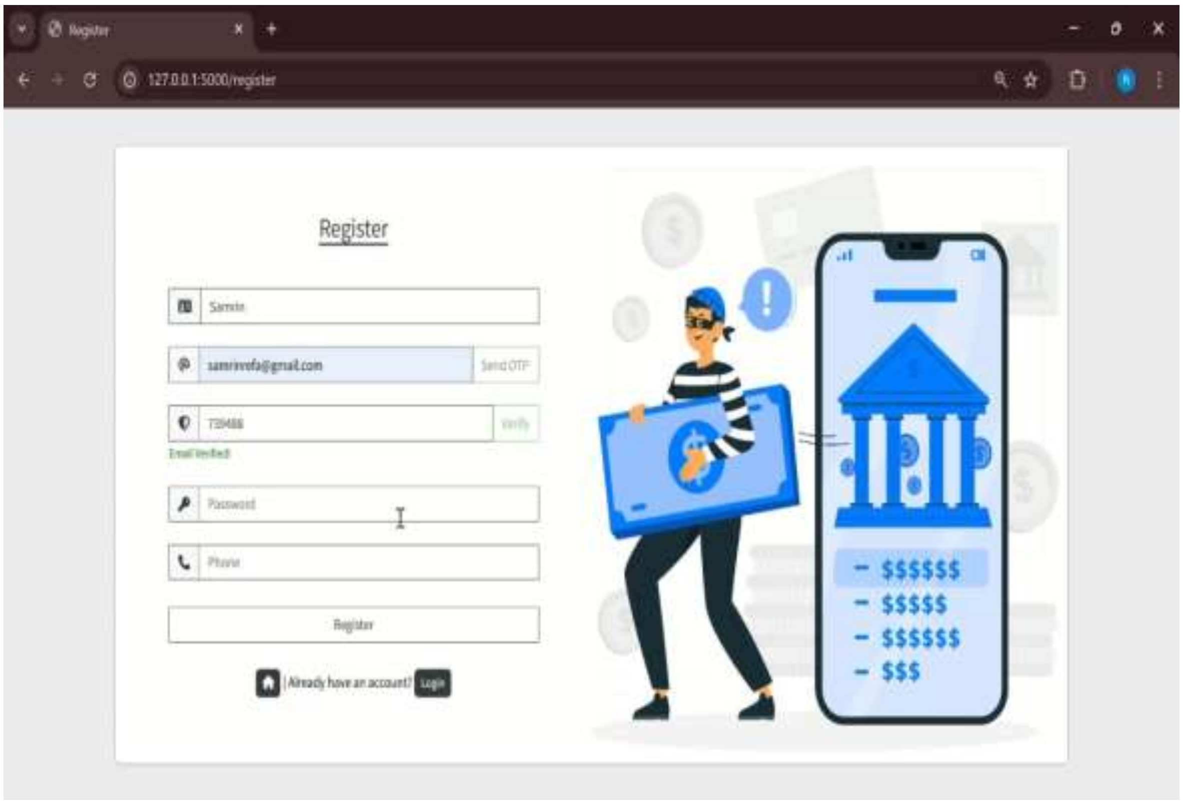
Snapshot 2: Registration Page


127.0.0.1:5000/register

127.0.0.1:5000 says

OTP sent! Check your email.

Snapshot 3: Email Validation


127.0.0.1:5000/register

Register

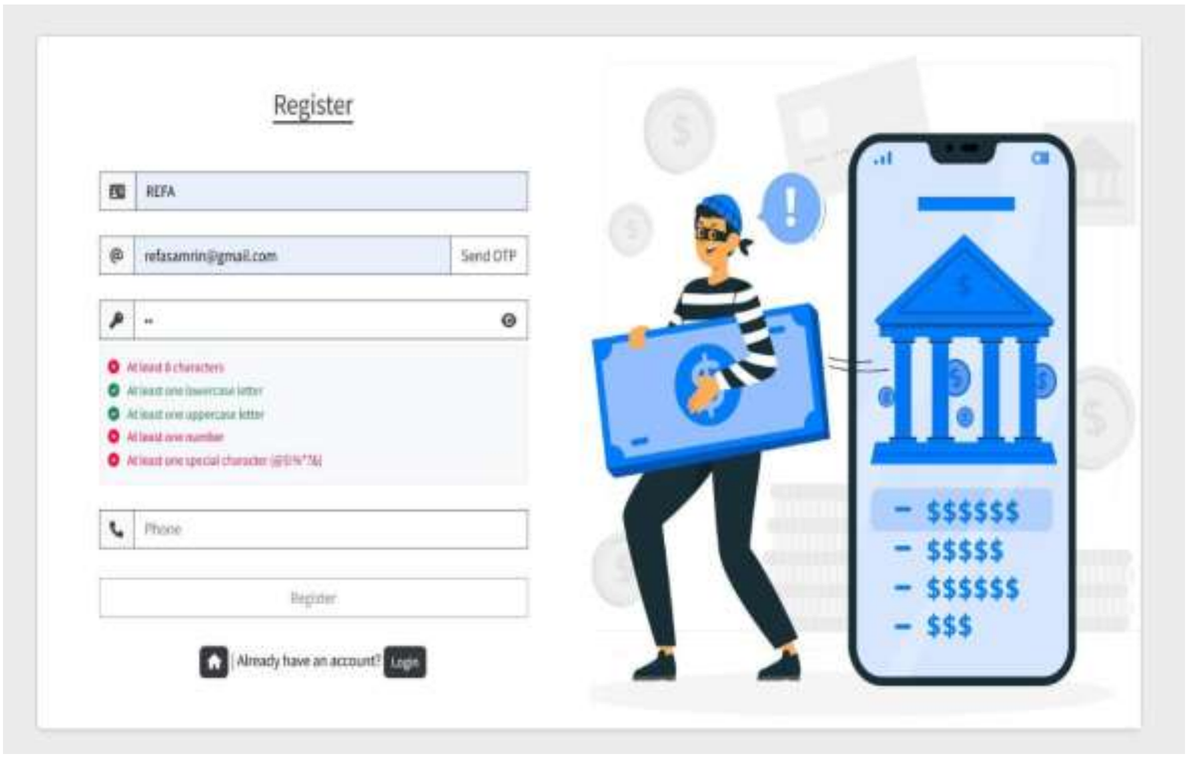
Samin

samrinvefa@gmail.com

739488

Email Verified

Snapshot 4: Email Verified



Register

REFA

refasamrin@gmail.com | Send OTP

••

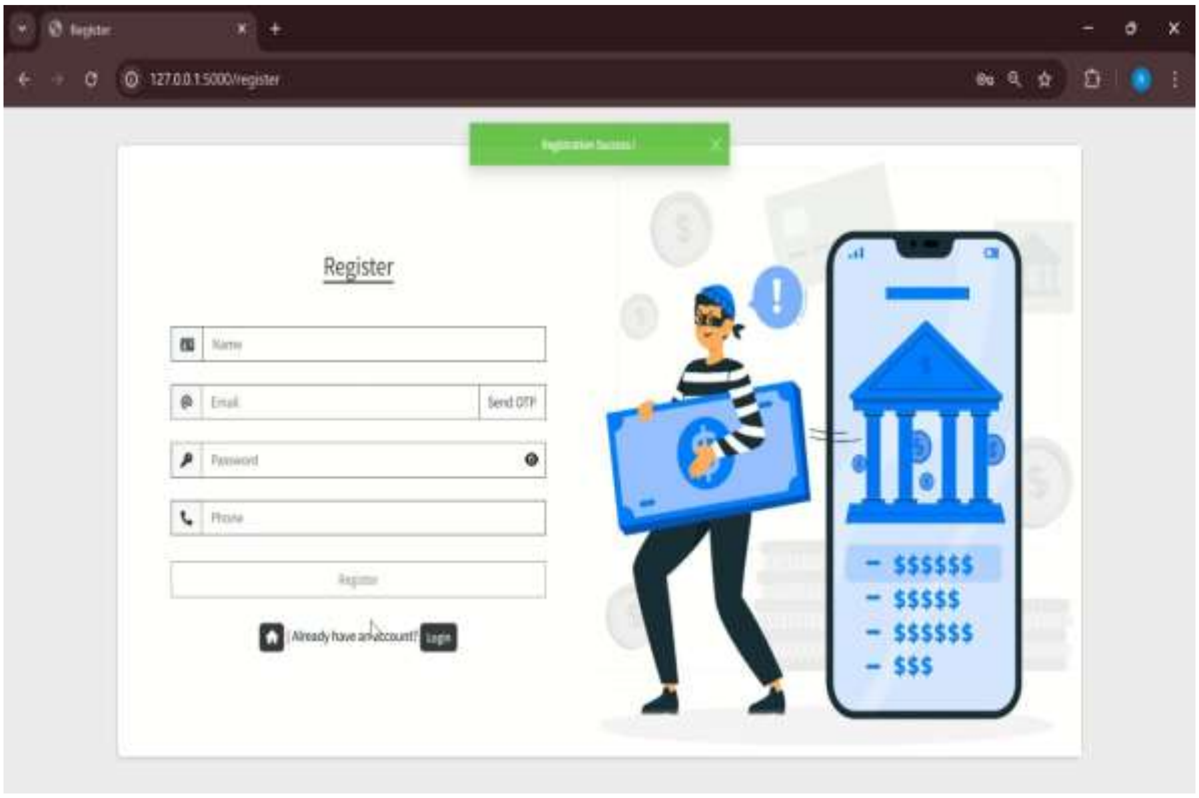
- ✗ At least 8 characters
- ✓ At least one lowercase letter
- ✓ At least one uppercase letter
- ✗ At least one number
- ✗ At least one special character (@$!%*?&)

Phone

Register

Already have an account? Login

Snapshot 5: Creating Strong Password



Registration Success!

Register

Name

Email | Send OTP

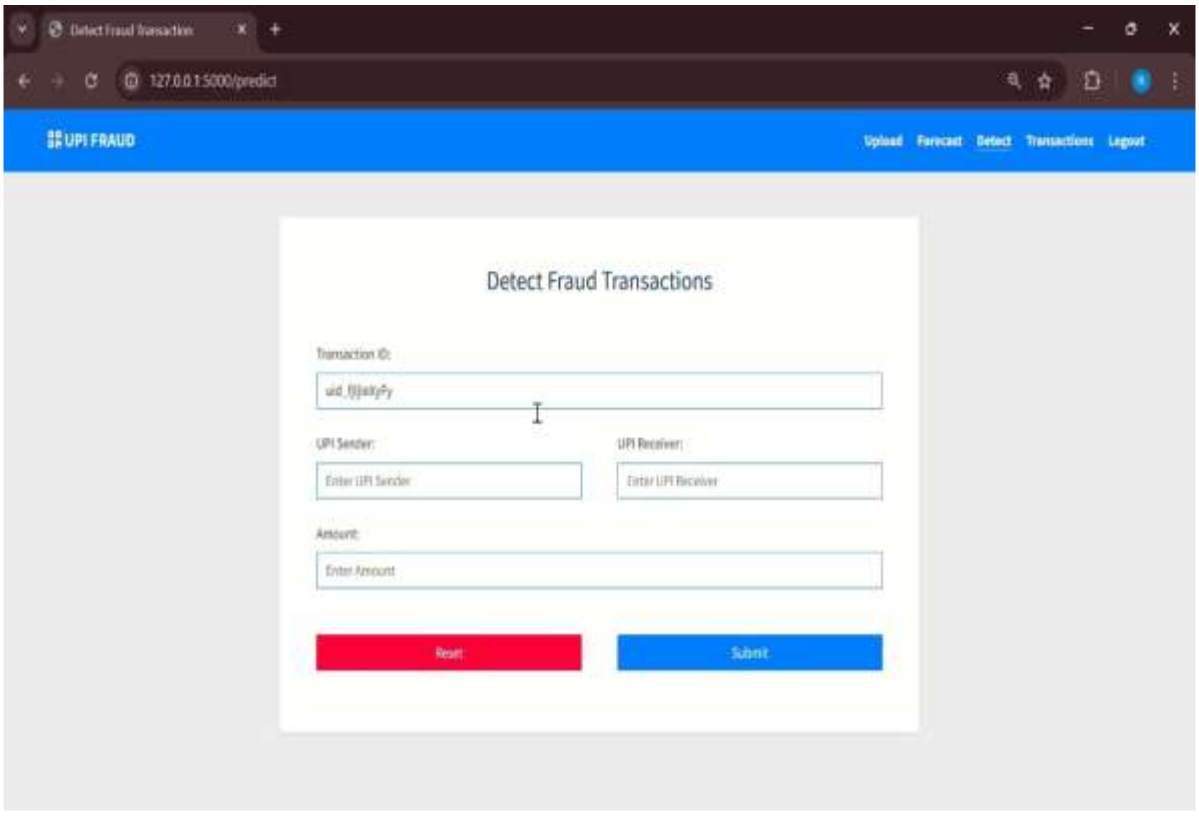
Password

Phone

Register

Already have an account? Login

Snapshot 6: Registration Successful



Detect Fraud Transactions

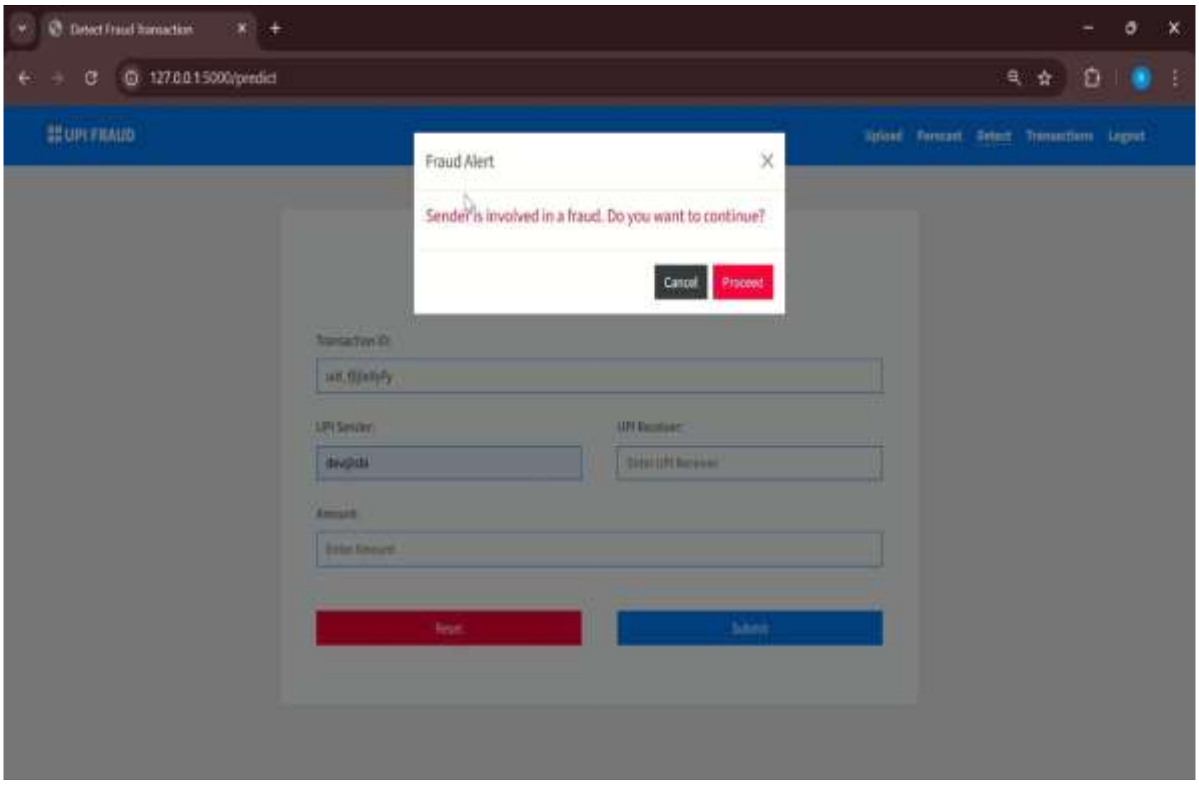
Transaction ID:
uid_fjljallyFy

UPI Sender:
Enter UPI Sender

UPI Receiver:
Enter UPI Receiver

Amount:
Enter Amount

Snapshot 7: Detection Page



Fraud Alert

Sender is involved in a fraud. Do you want to continue?

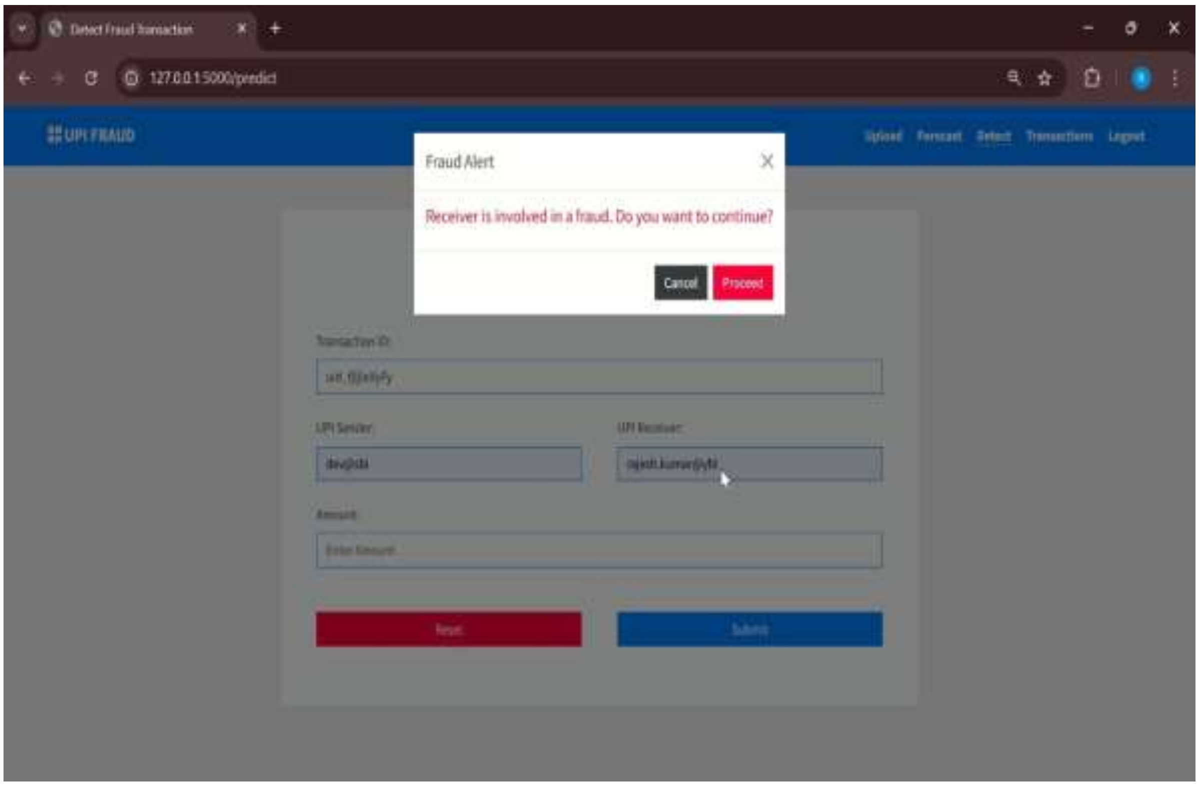
Transaction ID:
uid_fjljallyFy

UPI Sender:
devjlsbi

UPI Receiver:
Enter UPI Receiver

Amount:
Enter Amount

Snapshot 8: Detecting Sender as Fraud


Fraud Alert

Receiver is involved in a fraud. Do you want to continue?

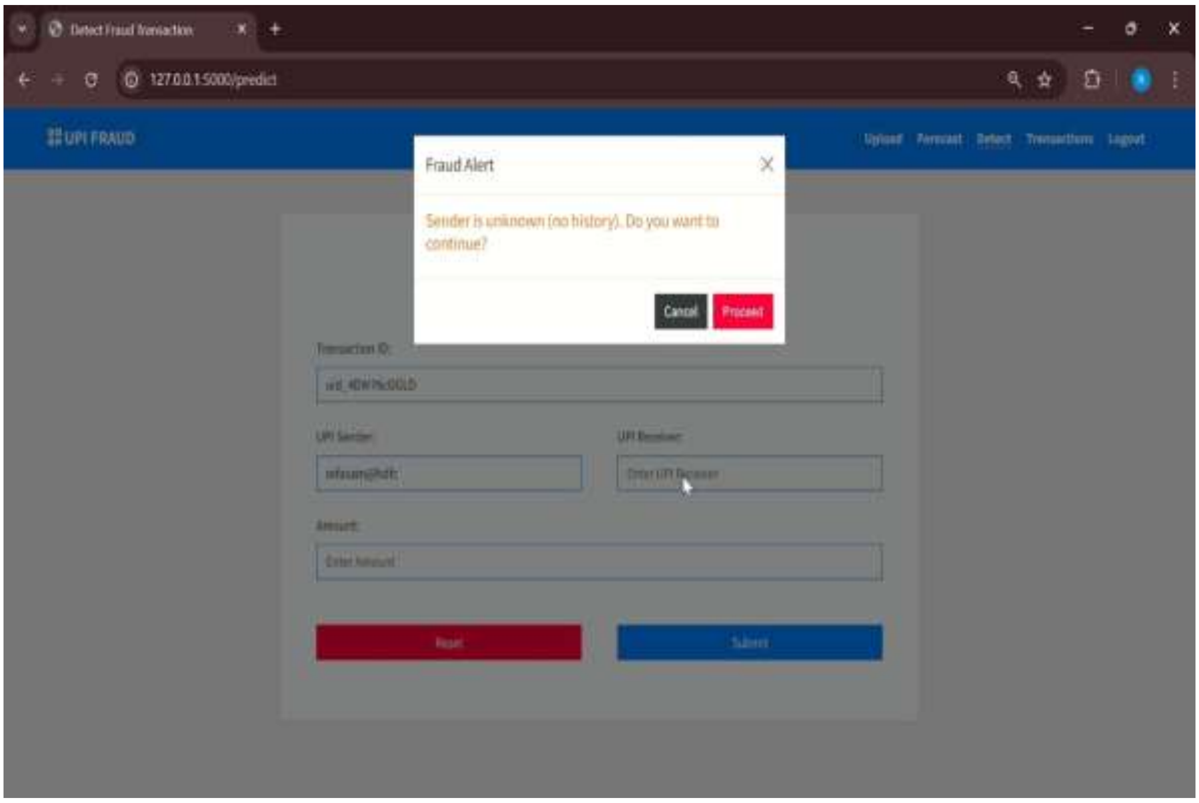
Transaction ID:

UPI Sender: dev@sbi

UPI Receiver: rajesh.kumar@ybl

Amount:

Snapshot 9: Detecting Receiver as Fraud


Fraud Alert

Sender is unknown (no history). Do you want to continue?

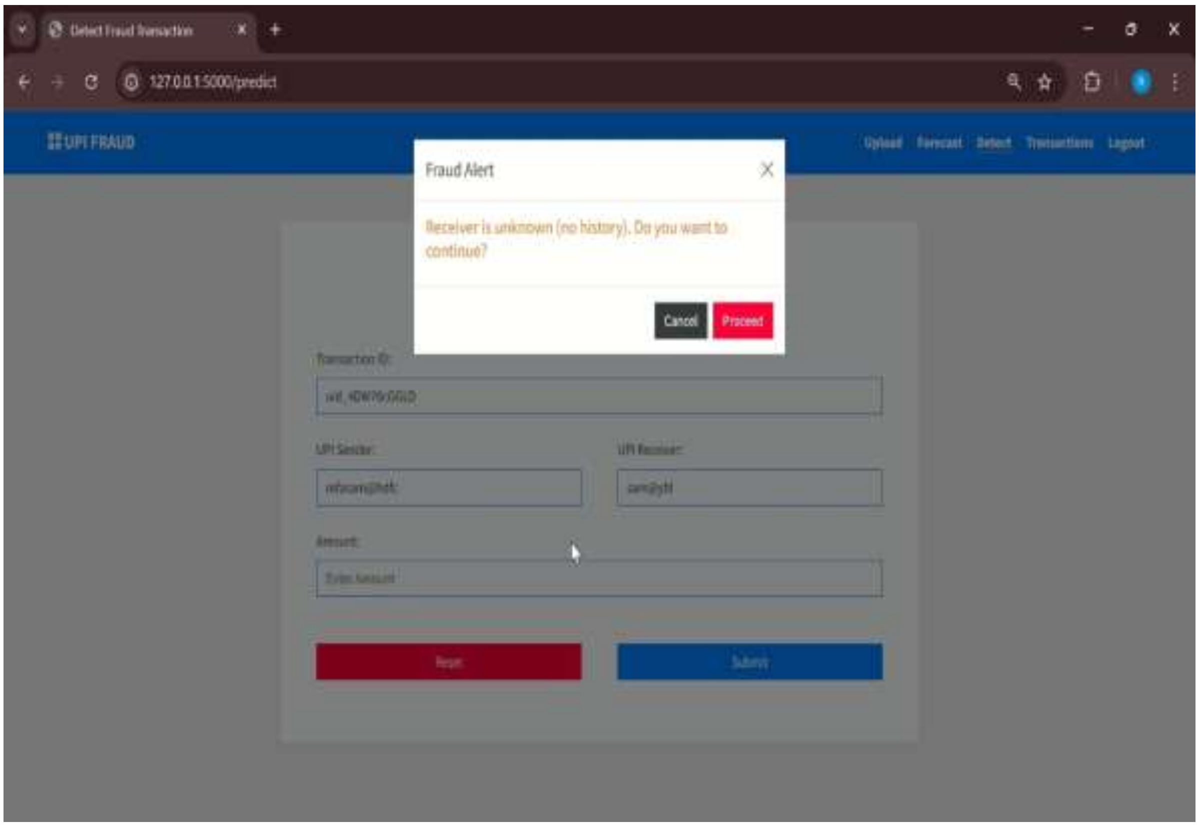
Transaction ID:

UPI Sender:

UPI Receiver:

Amount:

Snapshot 10: Sender Unknown Alert



Fraud Alert

Receiver is unknown (no history). Do you want to continue?

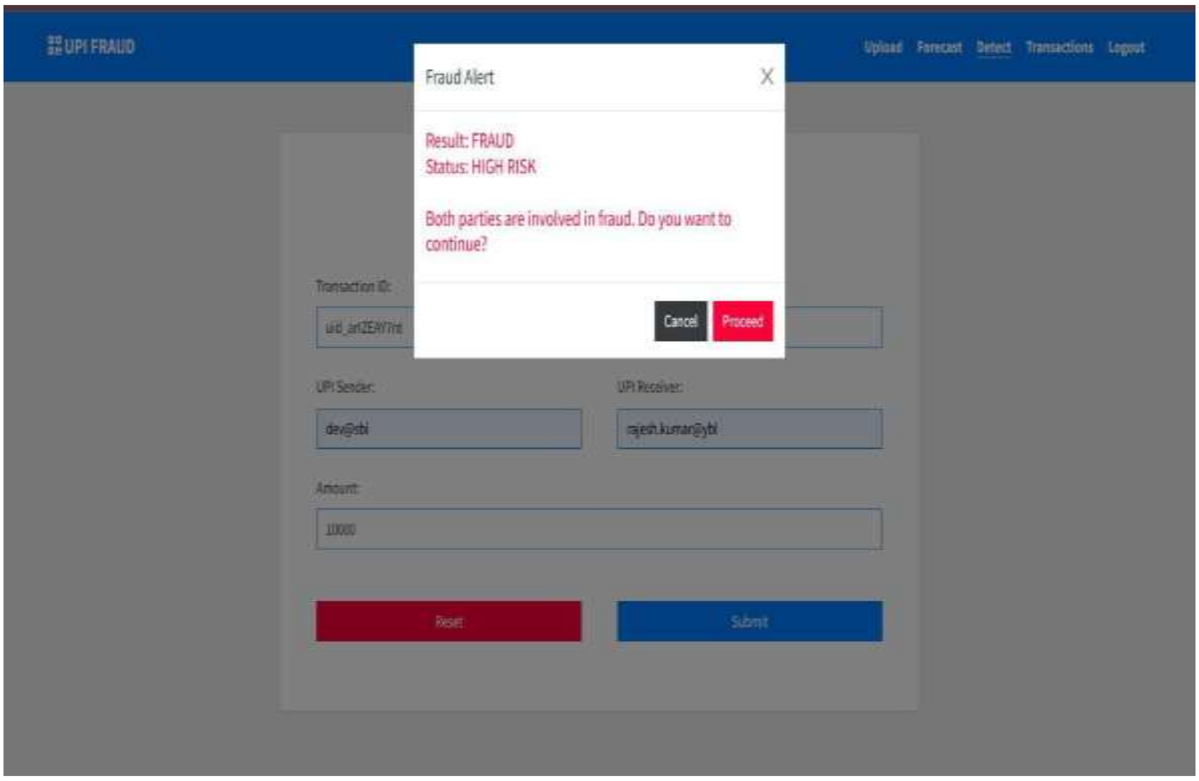
Transaction ID: uid_4DW76i5GLD

UPI Sender: mfxsam@hdfc

UPI Receiver: sam@ybl

Amount: Enter Amount

Snapshot 11: Receiver Unknown Alert



Fraud Alert

Result: FRAUD
Status: HIGH RISK

Both parties are involved in fraud. Do you want to continue?

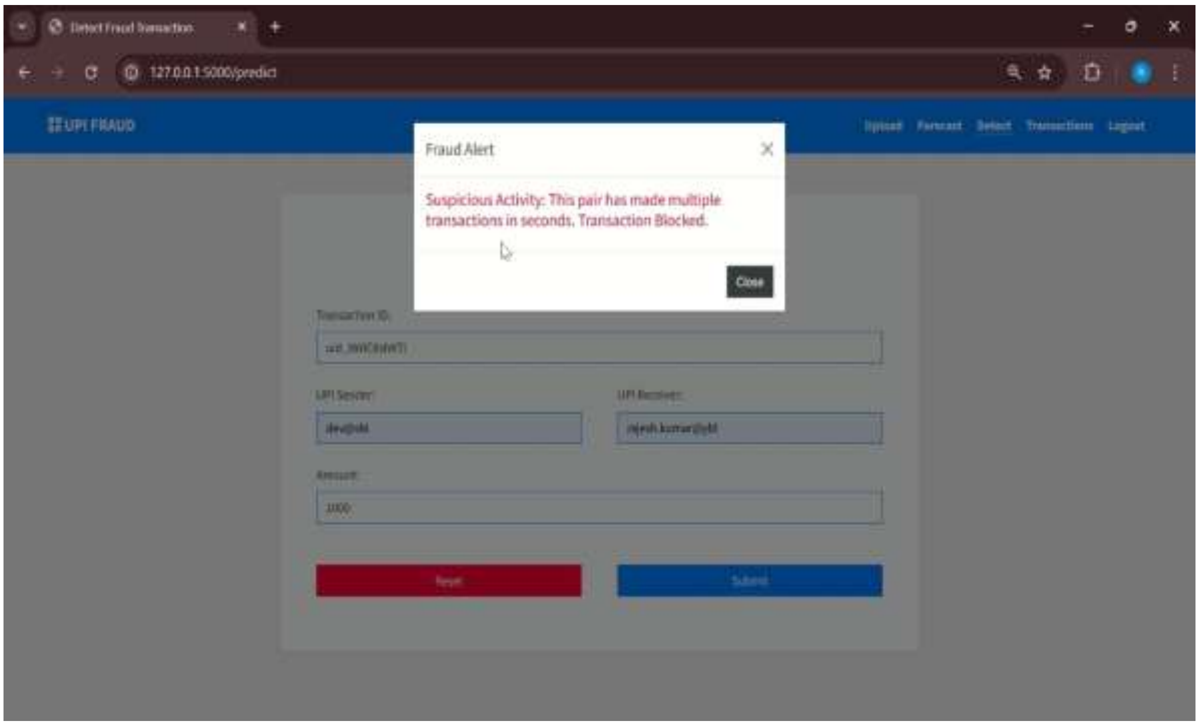
Transaction ID: uid_arIZEAY7mt

UPI Sender: dev@sbi

UPI Receiver: rajesh.kumar@ybl

Amount: 10000

Snapshot 12: Transaction Status



Fraud Alert

Suspicious Activity: This pair has made multiple transactions in seconds. Transaction Blocked.

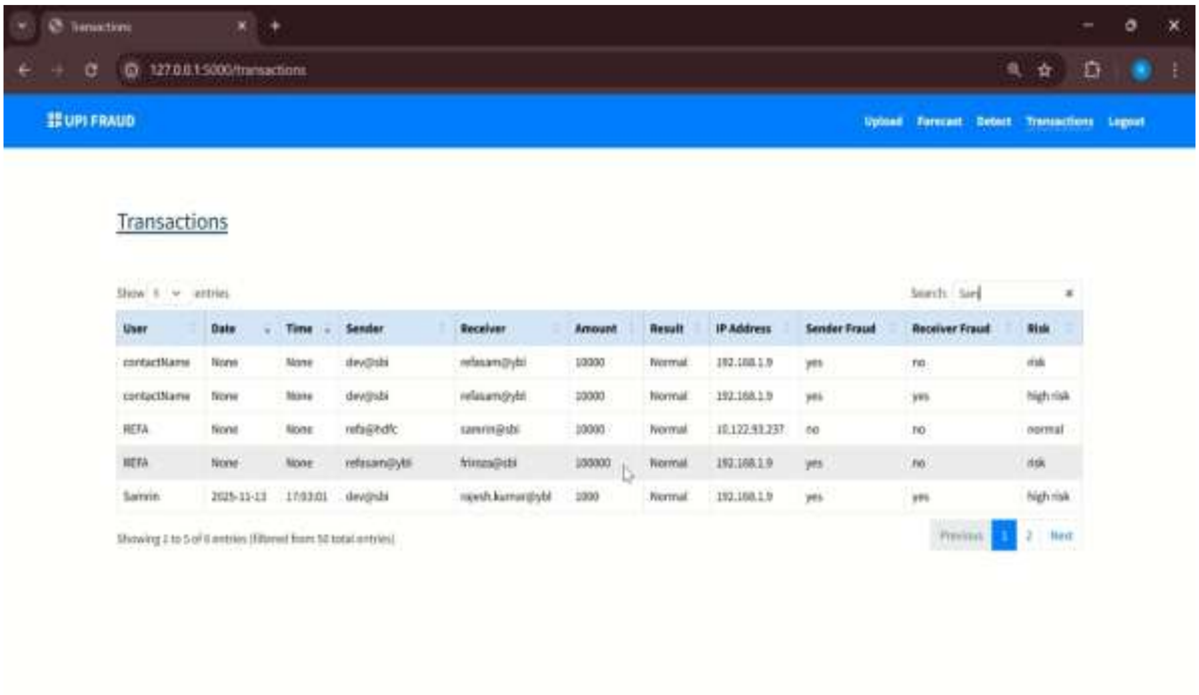
Transaction ID: [illegible]

UPI Sender: dev@sbi

UPI Receiver: rajesh.kumar@ybl

Amount: 1000

Snapshot 13: Blocking the Transaction


Transactions
127.0.0.1:5000/transactions
UPI FRAUD
Upload Forecast Detect Transactions Logout
Previous 1 2 Next

Transactions

Show 5 entries

Search: Sam

User	Date	Time	Sender	Receiver	Amount	Result	IP Address	Sender Fraud	Receiver Fraud	Risk
contactName	None	None	dev@sbi	refasam@ybl	10000	Normal	192.168.1.9	yes	no	risk
contactName	None	None	dev@sbi	refasam@ybl	10000	Normal	192.168.1.9	yes	yes	high risk
REFA	None	None	refa@hdfc	samrin@sbi	10000	Normal	10.122.93.237	no	no	normal
REFA	None	None	refasam@ybl	frinza@sbi	100000	Normal	192.168.1.9	yes	no	risk
Samrin	2025-11-13	17:03:01	dev@sbi	rajesh.kumar@ybl	1000	Normal	192.168.1.9	yes	yes	high risk

Showing 1 to 5 of 6 entries (filtered from 50 total entries)

Snapshot 14: Transaction History

****

CONCLUSION

This project presents a practical and intelligent way to detect fraudulent financial transactions using an Anomaly-Based Intrusion Detection System. By combining multiple machine learning models like CNN, SVM, and Naïve Bayes, the system learns real transaction patterns and identifies anything unusual with better accuracy than traditional methods. These models work together to reduce mistakes, allowing the system to make quick and reliable decisions. This makes it highly suitable for real-time financial environments where early detection can prevent major losses.

Along with machine learning, the use of Blockchain technology adds an extra layer of trust and security. Every transaction stored on the blockchain becomes transparent and tamper-proof, ensuring that data cannot be changed or misused. This creates confidence for users and strengthens the overall reliability of the system. Together, machine learning and blockchain provide a strong, proactive approach to financial safety, helping reduce fraud and offering a solid foundation that can grow and adapt to future banking and digital payment needs.

****

FUTURE SCOPE

In the future, this system can become much stronger by training it on bigger and real-time financial datasets. Because our project used only a limited dataset, adding live transaction data will help the model learn continuously and detect new types of fraud more accurately. We also wanted to include advanced deep learning models like LSTMs or Transformers, but we couldn’t add them due to limited time and resources. Adding these models later will help the system understand user behavior better and catch even small, unusual activities that might be missed now.

There are also many features we planned but could not include in this version. For example, creating a mobile app for instant fraud alerts would make the system more convenient for users, but we couldn’t implement it due to UI and time constraints. The blockchain part can also be upgraded with smart contracts to automatically block suspicious transactions, which was not possible within our current project scope. We also hope to deploy the system on cloud platforms in the future so it can handle large amounts of data and support real banks and payment systems. With these additions, the project can grow into a complete, real-time, and user-friendly fraud prevention system.

****

REFERENCES

[1] J. Huang, Y. Chen, X. Wang, Z. Ouyang, and N. Du, “Optimization Scheme of Collaborative Intrusion Detection System Based on Blockchain Technology,” *Electronics*, vol. 14, no. 2, art. 261, 2025.

[2] A. Immadisetty, “Real-Time Fraud Detection Using Streaming Data in Financial Transactions,” *Journal of Recent Trends in Computer Science and Engineering (JRTCSE)*, vol. 13, no. 1, pp. 66–76, 2025. DOI: 10.70589/JRTCSE.2025.13.1.9.

[3] M. Hasan, M. S. Rahman, H. Janicke, and I. H. Sarker, “Detecting Anomalies in Blockchain Transactions Using Machine Learning Classifiers and Explainability Analysis,” *Blockchain: Research and Applications*, vol. 5, no. 3, art. 100207, 2024. DOI: 10.1016/j.bcra.2024.100207.

[4] A. A. Aliyu, J. Liu, and E. Gilliard, “A Decentralized and Self-Adaptive Intrusion Detection Approach Using Continuous Learning and Blockchain Technology,” *Journal of Data Science and Intelligent Systems*, Oct. 2024, Online First. DOI: 10.47852/bonviewJDSIS42023803.

[5] S. Siddamsetti, “Anomaly Detection in Blockchain Using Machine Learning,” *Journal of Electrical Systems*, vol. 20, pp. 619–634, 2024. DOI: 10.52783/jes.2988.

[7] C. Cholevas, E. Angeli, Z. Sereti, E. Mavrikos, and G. E. Tsekouras, “Anomaly Detection in Blockchain Networks Using Unsupervised Learning: A Survey,” *Algorithms*, vol. 17, no. 5, p. 201, 2024.

[8] R. Lalduhsaka, N. Bora, and A. K. Khan, “Anomaly-Based Intrusion Detection Using Machine Learning: An Ensemble Approach,” *International Journal of Information Security and Privacy (IJISP)*, vol. 16, no. 1, pp. 1–15, 2022.

[9] P. Ramesh, J. K. Lee, and M. González, “Privacy-Preserving Fraud Detection in Financial Systems Using Federated Learning and Blockchain,” *International Journal of Secure Financial Computing*, vol. 12, no. 3, pp. 145–158, 2021.

[10] W. Zhang, L. Romano, and A. El-Adly, “Graph Neural Networks for Fraud Detection on Transaction Networks,” *Journal of Computational Intelligence in Finance*, vol. 8, no. 2, pp. 67–84, 2020.

[11] R. Kumar and E. Santos, “Hybrid Autoencoder and One-Class SVM for Micro-Transaction Fraud Detection,” *Proceedings of the International Conference on Financial Data Mining*, pp. 201–214, 2019.

[12] S. Ortiz and M. Chen, “Explainable Boosted Trees for Regulatory-Compliant Financial Anomaly Detection,” *IEEE Transactions on Financial Machine Learning*, vol. 5, no. 1, pp. 25–40, 2022.

[13] O. Hansen and F. Al-Mansouri, “Blockchain-Enabled Audit Trail for Machine-Learning-Based Payment Anti-Fraud Systems,” *International Conference on Blockchain Applications in Finance*, pp. 98–112, 2020.

[14] A. Patel and S. M. Roth, “Ensemble Learning with Cost-Sensitive Loss Functions for Fraud Detection,” *Journal of Applied Machine Learning in Banking*, vol. 3, no. 4, pp. 210–226, 2018.

[15] H. Tanaka and Z. Petrov, “Real-Time Anomaly Scoring with Lightweight Feature Hashing for High-Throughput Payment Systems,” *IEEE Symposium on High-Speed Transaction Processing*, pp. 55–69, 2019.

[16] D. Rossi and V. Lopez, “Cross-Platform Transfer Learning for Fraud Detection Across Payment Channels,” *International Journal of Deep Learning for Finance*, vol. 9, no. 1, pp. 88–103, 2021.

[17] M. Iqbal and T. Berger, “Reinforcement Learning for Adaptive Fraud-Blocking Policies in Payment Systems,” *ACM Conference on Financial Security and AI*, pp. 130–144, 2022.

[18] R. Kulkarni and S. Narayanan, “Hybrid Deep Learning Framework for Fraud Detection in Digital Payments,” Journal of Intelligent Financial Systems, vol. 5, no. 2, pp. 112–130, 2023.

[19] A. Mishra and H. Gupta, “Secure Financial Transaction Monitoring Using Blockchain and Smart Contracts,” International Journal of Blockchain and Distributed Ledger Technology, vol. 8, no. 1, pp. 45–62, 2024.

[20] P. Verma and T. Menon, “Ensemble-Based Financial Fraud Detection Using Feature Engineering and Gradient Boosting Models,” Journal of Machine Learning in Finance, vol. 11, no. 3, pp. 154–172, 2023.

****

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 12, December 2025

|DOI: 10.15680/IJIRSET.2025.1412001|

Anomaly-Based Intrusion Detection System in Financial Transactions using Ensemble Machine Learning and Blockchain

[1]Shreya M A, [2]Thrupthi K N, [3]Supreetha M S, [4]Refa Samrin, [5]Dr. Rajashekar M B

[1 2 3 4]Department of Computer Science & Engineering,

GSSS Institute of Engineering & Technology for Women, Mysuru, Affiliated to VTU, Belagavi, Karnataka, India

[5]Associate Professor, Department of Computer Science & Engineering,

GSSS Institute of Engineering & Technology for Women, Mysuru, Affiliated to VTU, Belagavi, Karnataka, India

ABSTRACT: With the rapid expansion of digital payment systems, the possibility of financial fraud is also growing fast. Most traditional rule-based Intrusion detection systems lack the ability to identify new and evolving fraud patterns within a real-time context. Therefore, this paper proposes an Anomaly-Based Intrusion Detection System for financial transactions using Ensemble Machine Learning with Blockchain technology. It processes transaction data to learn normal behavior of the users, and detects unusual activities such as unexpected spikes in the amount of transfers initiated or interaction with unknown users. Genuine transactions in this regard go on to be recorded on a blockchain ledger for total transparency of these records against alteration risks. The trials indicated that integrating ensembles with blockchain will enhance the accuracy of detection and reduction of false alerts for a reliable real-time method of protection for the digital financial transaction.

KEYWORDS: Anomaly Detection; Intrusion Detection System (IDS); Financial Fraud Detection; Ensemble Machine Learning; Blockchain Security; Transaction Monitoring; Digital Payments; Real-Time Fraud Prevention; Data Analytics.

I. INTRODUCTION

The rapid digital transition witnessed in the financial segment has brought a change in the manner of conducting transactions. Because of online banking, UPI, mobile wallets, and E-commerce sites, people have been able to carry out transactions at any given time and from anywhere. Although these facilities have accelerated the more number of transactions, they have also made the financial area more vulnerable to cyber threats. Fraudsters and thieves take advantage of weaknesses and ignorance on the part of these people. The rising number depicts the importance of an efficient Real-time Intrusion Detection System.

Conventional intrusion detection systems work on rule-based models and check for specific patterns, like abnormally high transaction values and unusual access locations. Traditional approaches are efficient and capable of detecting existing threats, but they are not effective for identifying new and sophisticated cyber threats, as cyber criminals adapt and change incessantly.

Keeping these problems in mind, there is a proposal for a hybrid solution based on an IDS with Ensemble Machine Learning algorithms, CNN and SVM. These models focus on examining transactional activities and historical data insights.

Adding blockchain technology will help us to safeguard and record transactions effectively and securely. Its decentralized and immutable structure enhances data integrity and transparency.

Together, Machine Learning and Blockchain enable anomaly detection and record-keeping functionalities. The result is an increase in financial security and user trust.

II. LITERATURE SURVEY

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 12, December 2025

|DOI: 10.15680/IJIRSET.2025.1412001|

The papers surveyed analyze research on applying machine learning and blockchain techniques in intrusion and fraudulent activity detection on finance and blockchain. Some papers highlight the superiority of ensemble learning techniques involving Weighted Random Forests, voting classifiers, and CNN, SVM, Naïve Bayes, and Decision Trees hybrids for improving accuracy on anomaly detection on different feature sets and inductive bias. Other papers deal with streaming analytics involving temporal models RNN and LSTM for online transaction streams for fraud detection. Cost computation, false positives, and drift remain challenges.

A common thread among the articles that emerged here was the implementation of blockchain technologies for secured logging, coordination, and audit trails. The blockchain technologies raise transparency but they also ensure that all efforts within an IDS are validated and enable collaboration but maintain privacy, specifically within federated learning.

Unsupervised methods, clustering, autoencoders, isolation forest, and graph analytics are known for discovering novel attack behaviors but suffer from interpretation and high rates of false positives.

However, some common problems that have been related with all these works include scalability, delay caused by consensus, computational complexity, imbalance in datasets, regulatory compliance, and the need for an explainable and real-time solution.

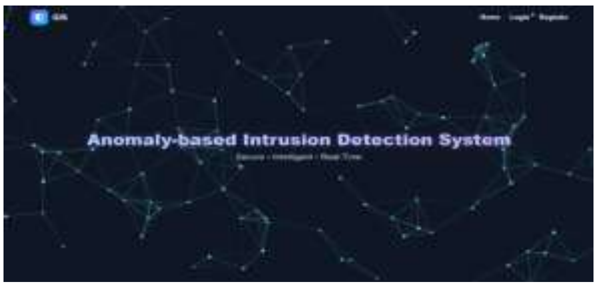
III. METHODOLOGY

Our system adopts a step-by-step method that analyzes user spending behavior and identifies anything unusual. We start with cleansing and structuring the transaction data into sets with useful and consistent information. We then represent these transactions as sequences so that the model can identify meaningful patterns such as the frequency with which an individual transfers money, the timing associated with these transfers, and the average amount they normally send. We then proceed to feed these sequences into a Convolutional Neural Network.

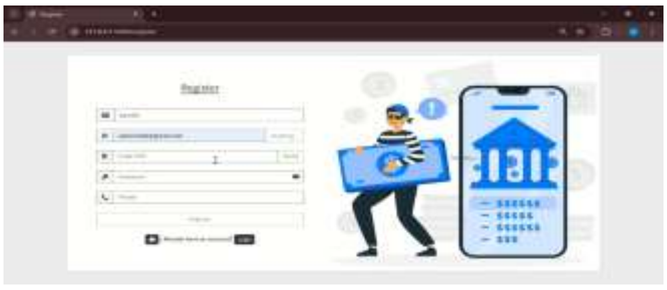
Once the CNN is done with the extraction of these patterns, these feature extractions are passed on to a Support Vector Machine, which works as a final decision-maker. It assists a transaction classification process as it identifies these discovered patterns. A transaction can be labelled once it undergoes verification. The verified and legitimate ones are then stored within a blockchain ledger. All entries are secured with a cryptographic hash so that no party can make any surreptitious changes to it at a later stage. The above-mentioned mechanism lets an anomalous pattern be detected with precision while also making valid ones tamper-proof and authentic.

IV. EXPERIMENTAL RESULTS

Figures shows the results of Anomaly-Based Intrusion Detection System in Financial Transactions using Ensemble Machine Learning and Blockchain. Figs. (a) Home Page (b) Registration Page (c) Email Validation (d) Email Verified (e) Creating strong Password (f) Registration Successful (g) Detection page (h) Detecting Sender as Fraud (i) Receiver Unknown Alert (j) Transaction Status (k) Blocking the transaction



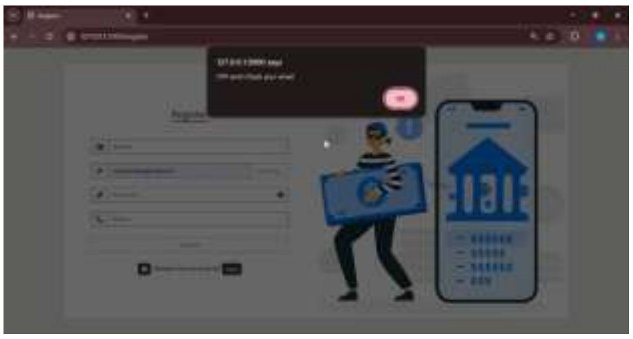
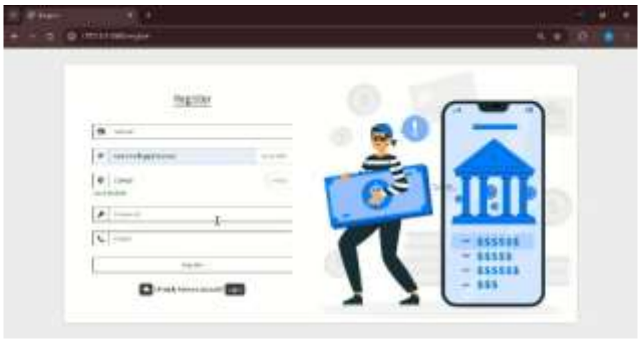
(a)



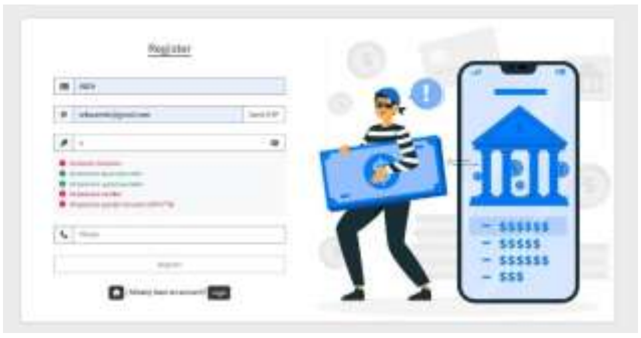
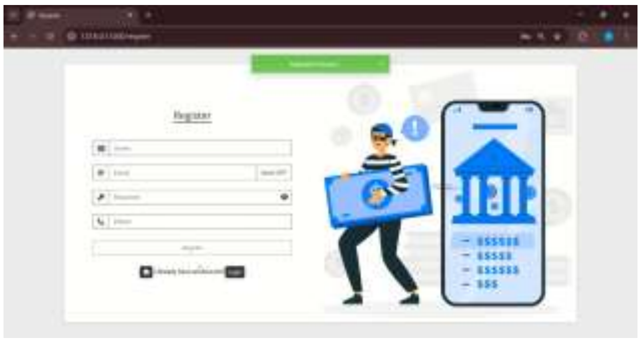
(b)

Volume 14, Issue 12, December 2025

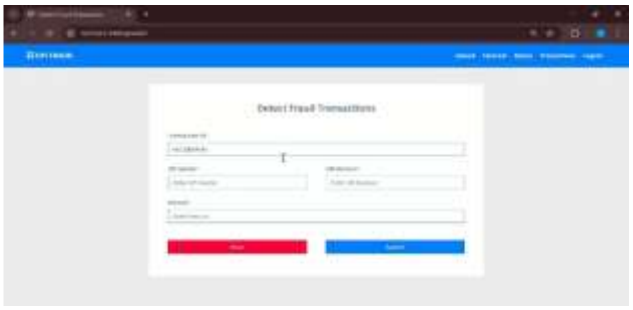
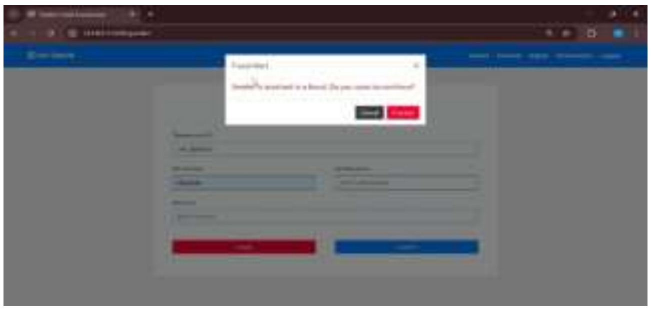
|DOI: 10.15680/IJIRSET.2025.1412001|

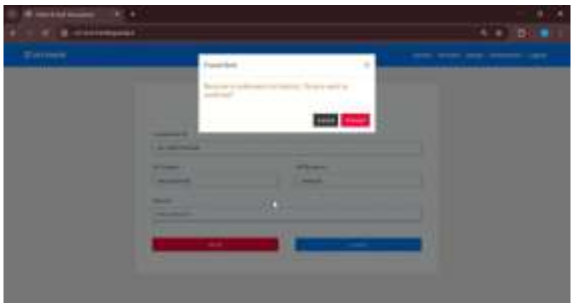
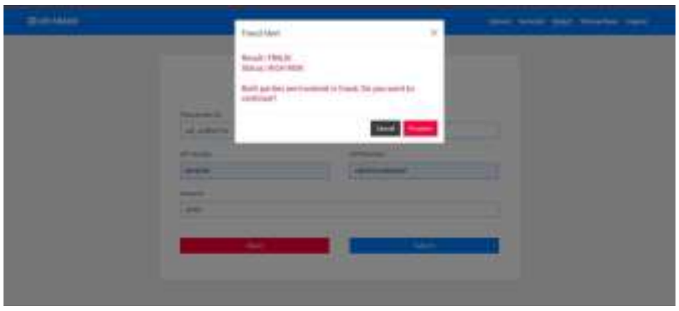
(c) (d)

(e) (f)

(g) (h)

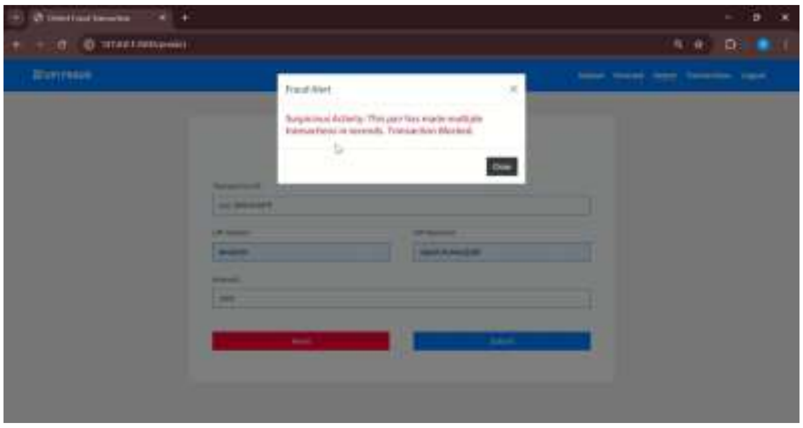



(i) (j)

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 12, December 2025

|DOI: 10.15680/IJIRSET.2025.1412001|



Fraud Alert

Suspicious Activity: This user has made multiple transactions in seconds. Transaction Blocked.

(k)

V. CONCLUSION

As a contribution to research on blockchain and artificial intelligence, we have designed and implemented a system that uses machine learning and blockchain concepts together for flagging unusual/suspicious financial transactions. Our CNN-SVM-based system identifies unusual spending activities based on user regularities not yet experienced and learned before, making it optimized than traditional rule-based systems. By storing legitimate transactions on a blockchain, we have ensured that critical data will not be altered or abused.

REFERENCES

[1] J. Huang, Y. Chen, X. Wang, Z. Ouyang, and N. Du, “Optimization Scheme of Collaborative Intrusion Detection System Based on Blockchain Technology,” Electronics, vol. 14, no. 2, art. 261, 2025.

[2] A. Immadisetty, “Real-Time Fraud Detection Using Streaming Data in Financial Transactions,” Journal of Recent Trends in Computer Science and Engineering (JRTCSE), vol. 13, no. 1, pp. 66–76, 2025. DOI: 10.70589/JRTCSE.2025.13.1.9.

[3] M. Hasan, M. S. Rahman, H. Janicke, and I. H. Sarker, “Detecting Anomalies in Blockchain Transactions Using Machine Learning Classifiers and Explainability Analysis,” Blockchain: Research and Applications, vol. 5, no. 3, art. 100207, 2024. DOI: 10.1016/j.bcra.2024.100207.

[4] A. A. Aliyu, J. Liu, and E. Gilliard, “A Decentralized and Self-Adaptive Intrusion Detection Approach Using Continuous Learning and Blockchain Technology,” Journal of Data Science and Intelligent Systems, Oct. 2024, Online First. DOI: 10.47852/bonviewJDSIS42023803.

[5] S. Siddamsetti, “Anomaly Detection in Blockchain Using Machine Learning,” Journal of Electrical Systems, vol. 20, pp. 619–634, 2024. DOI: 10.52783/jes.2988.

[7] C. Cholevas, E. Angeli, Z. Sereti, E. Mavrikos, and G. E. Tsekouras, “Anomaly Detection in Blockchain Networks Using Unsupervised Learning: A Survey,” Algorithms, vol. 17, no. 5, p. 201, 2024.

[8] R. Lalduhsaka, N. Bora, and A. K. Khan, “Anomaly-Based Intrusion Detection Using Machine Learning: An Ensemble Approach,” International Journal of Information Security and Privacy (IJISP), vol. 16, no. 1, pp. 1–15, 2022.

[9] P. Ramesh, J. K. Lee, and M. González, “Privacy-Preserving Fraud Detection in Financial Systems Using Federated Learning and Blockchain,” International Journal of Secure Financial Computing, vol. 12, no. 3, pp. 145–158, 2021.

[10] W. Zhang, L. Romano, and A. El-Adly, “Graph Neural Networks for Fraud Detection on Transaction Networks,” Journal of Computational Intelligence in Finance, vol. 8, no. 2, pp. 67–84, 2020.