***Course Code/Course Title:*** ICT2140/INTRODUCTION TO SOFTWARE ENGINEERING

***Group Number:*** Group9

***Project Topic:*** FRAUD DETECTION SYSTEM

*Link to GitHub Repository:*

***Group Leader:*** NJUME BRIAN EPIE

***Group Information***

|  |  |  |  |
| --- | --- | --- | --- |
| *SN* | *Member’s Name* | *Registration Number* | *Team Role* |
| *1* | NJUME BRIAN EPIE | ICTU20241420 | Scrum Master |
| *2* | ABOUBAKAR GARBA MAMOUDOU | ICTU20241612 | Team member |
| *3* | MOHAMMED HAIBAI MOUSTAPHA ABOUBAKAR | ICTU20241807 | Team member |
| *4* | MOUSTAPHA SIDI MOHAMED | ICTU20241409 | Team member |
| *5* | EBODE KESHA FARELLE | ICTU20241783 | Team member |

[1 INTRODUCTION 6](#_Toc208565658)

[1.1 General Introduction: 6](#_Toc208565659)

[1.2 Aim and Objectives: 6](#_Toc208565660)

[1.3 Problem Statement: 6](#_Toc208565661)

[2 LITERATURE REVIEW 7](#_Toc208565662)

[2.1 Review of Concepts Related to the Project 7](#_Toc208565663)

[2.1.1 Data Analysis and Feature Engineering: 7](#_Toc208565664)

[Critical for extracting meaningful insights from raw financial data, such as transaction type, account balances, and transaction amount. 7](#_Toc208565665)

[2.1.2 Machine Learning Models for Classification: 7](#_Toc208565666)

[2.1.3 Imbalanced Datasets: 7](#_Toc208565667)

[2.2 Review of Software Development Methodologies 7](#_Toc208565668)

[2.2.1 Waterfall Model: 7](#_Toc208565669)

[2.2.2 Agile Methodology: 7](#_Toc208565670)

[2.2.3 Spiral Model: 8](#_Toc208565671)

[2.2.4 V-Model (Verification and Validation Model): 8](#_Toc208565672)

[2.3 Reason for Using V-Model Methodology 8](#_Toc208565673)

[2.4 Review of Related Literature 8](#_Toc208565674)

[3 METHODOLOGY AND MATERIALS 9](#_Toc208565675)

[3.1 RESEARCH METHODOLOGY 9](#_Toc208565676)

[3.1.1 1. Research Design 9](#_Toc208565677)

[3.1.2 2. Data Collection 10](#_Toc208565678)

[3.1.2.1 Primary Data: 10](#_Toc208565679)

[3.1.2.2 Secondary Data: 10](#_Toc208565680)

[3.1.3 3. Data Analysis 10](#_Toc208565681)

[3.1.3.1 Exploratory Data Analysis (EDA): 10](#_Toc208565682)

[3.1.3.2 Feature Engineering: 10](#_Toc208565683)

[3.1.4 4. Model Development 10](#_Toc208565684)

[3.1.5 5. System Design and Implementation 11](#_Toc208565685)

[3.1.6 6. Testing and Validation 11](#_Toc208565686)

[3.1.6.1 Unit Testing: 11](#_Toc208565687)

[3.1.6.2 Integration Testing: 11](#_Toc208565688)

[3.1.6.3 Performance Testing: 11](#_Toc208565689)

[3.1.6.4 Validation: 11](#_Toc208565690)

[3.1.7 7. Ethical Considerations 12](#_Toc208565691)

[3.1.7.1 Privacy and Confidentiality: 12](#_Toc208565692)

[3.1.7.2 Bias Mitigation: 12](#_Toc208565693)

[3.1.7.3 Transparency and Explainability: 12](#_Toc208565694)

[3.1.8 8. Deployment and Continuous Monitoring 12](#_Toc208565695)

[3.2 System Requirements 12](#_Toc208565696)

[3.2.1 1. Functional Requirements 12](#_Toc208565697)

[3.2.1.1 1.1 Data Input and Preprocessing 13](#_Toc208565698)

[3.2.1.2 1.2 Model Training 13](#_Toc208565699)

[3.2.1.3 1.3 Model Evaluation 13](#_Toc208565700)

[3.2.1.4 1.4 Fraud Detection and Prediction 13](#_Toc208565701)

[3.2.1.5 1.6 Visualization and Reporting 14](#_Toc208565702)

[3.2.2 2. Non-Functional Requirements 14](#_Toc208565703)

[3.2.2.1 2.1 Performance 14](#_Toc208565704)

[3.2.2.2 2.2 Accuracy 15](#_Toc208565705)

[3.2.2.3 2.3 Scalability 15](#_Toc208565706)

[3.2.2.4 2.4 Usability 15](#_Toc208565707)

[3.2.2.5 2.5 Reliability 15](#_Toc208565708)

[3.2.2.6 2.6 Maintainability 16](#_Toc208565709)

[3.2.2.7 2.7 Security 16](#_Toc208565710)

[3.2.2.8 2.8 Portability 16](#_Toc208565711)

[3.3 System Design 16](#_Toc208565712)

[3.3.1 4.1 Architecture of the System (High-Level Design – HLD) 16](#_Toc208565713)

[3.3.1.1 4.1.1 Data Layer 17](#_Toc208565714)

[3.3.1.1.1 Input Sources: 17](#_Toc208565715)

[3.3.1.1.2 Data Preprocessing Module: 17](#_Toc208565716)

[3.3.1.1.3 Data Storage: 17](#_Toc208565717)

[3.3.1.2 4.1.2 Processing Layer (Machine Learning Core) 17](#_Toc208565718)

[3.3.1.2.1 Model Training Engine: 17](#_Toc208565719)

[3.3.1.2.2 Evaluation Engine: 17](#_Toc208565720)

[3.3.1.2.3 Model Repository: 17](#_Toc208565721)

[3.3.1.3 4.1.3 Application Layer 17](#_Toc208565722)

[3.3.1.3.1 Fraud Detection Module: 17](#_Toc208565723)

[3.3.1.3.2 Continuous Learning Module: 18](#_Toc208565724)

[3.3.1.4 4.1.4 Presentation Layer 18](#_Toc208565725)

[3.3.1.4.1 Visualization Dashboard: 18](#_Toc208565726)

[3.3.1.4.2 Reporting Module: 18](#_Toc208565727)

[3.3.1.5 4.1.5 Security and Control Layer 18](#_Toc208565728)

[3.3.1.5.1 Access Control: 18](#_Toc208565729)

[3.3.1.5.2 Data Protection: 18](#_Toc208565730)

[3.3.1.5.3 Audit Logs: 18](#_Toc208565731)

[3.3.1.6 4.2 High-Level Design Diagram 19](#_Toc208565732)

[3.3.2 UML Diagrams : 20](#_Toc208565733)

[3.4 Application of chosen Methodology 24](#_Toc208565734)

[3.4.1 Team organization 24](#_Toc208565735)

[3.4.1.1 Requirement Analysis: The project began with defining requirements based on the nature of the problem—detecting fraudulent transactions from a large financial dataset. Team members, such as business analysts and domain experts, contributed to identifying essential features, success criteria, and constraints. 25](#_Toc208565736)

[3.4.1.2 Data Collection and Preprocessing: Data scientists were responsible for collecting and cleaning the dataset (AIML Dataset.csv). Activities included checking for missing values, validating column structures, and understanding class distribution (fraudulent vs. non-fraudulent transactions). Collaboration at this stage ensured that the dataset was reliable and representative of real-world conditions. 25](#_Toc208565737)

[ Exploratory Data Analysis (EDA): 25](#_Toc208565738)

[ Model Development: 25](#_Toc208565739)

[ Testing and Validation: 25](#_Toc208565740)

[ Documentation and Reporting: 25](#_Toc208565741)

[ Deployment and Feedback: 25](#_Toc208565742)

[3.4.2 Workflow management 26](#_Toc208565743)

[3.4.2.1 2. Data Preprocessing 26](#_Toc208565744)

[3.4.2.2 3. Data Analysis 26](#_Toc208565745)

[3.4.2.3 4. Train-Test Split 27](#_Toc208565746)

[3.4.2.4 5. Model Training 27](#_Toc208565747)

[3.4.2.5 6. Evaluation 27](#_Toc208565748)

[3.4.2.6 7. Deployment 28](#_Toc208565749)

[3.5 Requirements specification (Product backlog and Sprint backlog) 28](#_Toc208565750)

[3.5.1 Product backlog: 28](#_Toc208565751)

[3.5.2 Sprint backlog: 30](#_Toc208565752)

[3.6 Test case document 31](#_Toc208565753)

[3.6.1 1. Introduction 31](#_Toc208565754)

[3.6.2 2. Test Case Structure 31](#_Toc208565755)

[3.6.3 3. Test Cases 31](#_Toc208565756)

[3.6.3.1 3.1 Data Preprocessing 31](#_Toc208565757)

[3.6.3.2 3.2 Model Training & Validation 32](#_Toc208565758)

[3.6.3.3 3.3 Fraud Detection (Functional Tests) 32](#_Toc208565759)

[3.6.3.4 3.4 Performance & Deployment 33](#_Toc208565760)

[3.6.4 4. Exit Criteria 34](#_Toc208565761)

[3.6.5 5. Conclusion 34](#_Toc208565762)

[3.7 Proposed Algorithms (core algorithm of your application) 34](#_Toc208565763)

[3.7.1 Step 1: Data Input 34](#_Toc208565764)

[3.7.2 Step 2: Data Preprocessing 34](#_Toc208565765)

[3.7.3 Step 3: Feature Engineering 35](#_Toc208565766)

[3.7.4 Step 4: Dataset Splitting 35](#_Toc208565767)

[3.7.5 Step 5: Model Training 35](#_Toc208565768)

[3.7.6 Step 6: Model Evaluation 35](#_Toc208565769)

[3.7.7 Step 7: Fraud Detection Decision Rule 35](#_Toc208565770)

[3.7.8 Step 8: Deployment & Monitoring 35](#_Toc208565771)

[3.8 Materials and technologies used 37](#_Toc208565772)

[3.8.1 Materials Used 37](#_Toc208565773)

[3.8.2 Technologies Used 37](#_Toc208565774)

[4 RESULTS AND DISCUSSIONS 38](#_Toc208565775)

[4.1 Screenshots of various application scenarios 38](#_Toc208565776)

[4.2 Screenshots of various API request/Response 45](#_Toc208565777)

[5 RECOMMENDATIONS AND CONCLUSION 47](#_Toc208565778)

# INTRODUCTION

## General Introduction:

In the modern digital economy, financial transactions are increasingly conducted through online platforms, mobile applications, and electronic banking systems. While these advancements have enhanced convenience and efficiency, they have also introduced vulnerabilities that fraudsters exploit for illicit gain. Fraudulent activities, such as identity theft, credit card fraud, and unauthorized transactions, pose significant threats to financial institutions, businesses, and individuals. Detecting such fraudulent patterns within massive volumes of transaction data requires robust, data-driven solutions. Machine learning has emerged as a powerful tool for analyzing complex datasets, identifying hidden patterns, and making accurate predictions in real-time, thereby offering effective means for fraud detection.

## Aim and Objectives:

The primary aim of this project is to design and implement a machine learning-based fraud detection system capable of analyzing transactional data and accurately distinguishing between legitimate and fraudulent activities. To achieve this aim, the project focuses on the following objectives:

* To conduct exploratory data analysis in order to identify patterns, correlations, and anomalies within financial transaction data.
* To preprocess and transform raw data into a suitable format for training machine learning models.
* To develop and evaluate classification models for detecting fraudulent activities.
* To compare the performance of different algorithms and select the most effective model for fraud detection.
* To provide a reliable framework that enhances security and minimizes financial risks

## Problem Statement:

The rise of digital financial transactions has created new opportunities for fraud, leading to substantial financial losses and reduced trust in digital payment systems. Traditional rule-based fraud detection methods are often rigid, slow to adapt, and insufficient for detecting complex or evolving fraudulent schemes. The sheer volume and complexity of transaction data further complicate the detection process. As a result, there is a critical need for advanced solutions that leverage data analysis and machine learning to automatically detect fraudulent activities with high accuracy and efficiency. This project seeks to address this problem by developing a fraud detection model that utilizes machine learning techniques to analyze transaction data, identify potential fraud, and improve decision-making in financial security systems.

# LITERATURE REVIEW

## Review of Concepts Related to the Project

Fraud detection is the process of identifying illegitimate transactions within large datasets of financial activities. It combines statistical analysis, pattern recognition, and machine learning to detect unusual behaviors that may indicate fraudulent activity. Core concepts relevant to this project include:

### Data Analysis and Feature Engineering:

### Critical for extracting meaningful insights from raw financial data, such as transaction type, account balances, and transaction amount.

### Machine Learning Models for Classification:

Algorithms such as Logistic Regression, Random Forest, and Gradient Boosting are widely applied in fraud detection due to their ability to learn complex relationships in data.

### Imbalanced Datasets:

Fraudulent cases are often much fewer than legitimate ones, requiring strategies like resampling or cost-sensitive learning.

## Review of Software Development Methodologies

Software development methodologies provide structured approaches for planning, designing, implementing, and testing software systems. The most common methodologies include:

### Waterfall Model:

A linear, sequential process with defined stages (requirements, design, implementation, testing, and deployment). While simple and structured, it lacks flexibility for iterative improvements.

### Agile Methodology:

Focuses on iterative development, continuous feedback, and adaptability. It is effective in environments where requirements change frequently.

### Spiral Model:

Combines iterative development with systematic risk analysis. It is suitable for large, high-risk projects.

### V-Model (Verification and Validation Model):

An extension of the Waterfall model where each development stage is directly associated with a testing phase. This ensures validation at every step, improving system reliability.

## Reason for Using V-Model Methodology

This project adopts the V-Model methodology because fraud detection systems demand high reliability, accuracy, and validation at every stage of development. Each step of the V-Model ensures that requirements are not only implemented but also tested against predefined goals. For example:

Data preprocessing is validated against data quality requirements.

Model training is tested against accuracy, precision, and recall metrics.

The Streamlit interface is validated for usability and real-time predictions.

By using the V-Model, the project ensures a systematic approach with continuous verification and validation, minimizing risks of errors in fraud prediction and guaranteeing that the system meets its intended objectives.

## Review of Related Literature

Several studies have explored the application of machine learning in fraud detection:

Bhattacharyya et al. (2011) investigated credit card fraud detection using Random Forests and highlighted the importance of handling imbalanced datasets.

Ngai et al. (2011) provided a comprehensive review of data mining techniques for fraud detection, categorizing them into classification, clustering, and hybrid approaches, highlighting their strengths and limitations.

Dal Pozzolo et al. (2015) emphasized the importance of using metrics beyond accuracy, such as ROC-AUC and precision-recall, when dealing with imbalanced fraud detection datasets.

Abdallah et al. (2016) analyzed various fraud detection frameworks and concluded that hybrid models integrating statistical and machine learning methods provide the highest performance.

Fiore et al. (2019) explored the application of deep learning for fraud detection and demonstrated that neural networks can uncover hidden non-linear relationships in large transaction datasets.

Zhang et al. (2020) examined graph-based fraud detection systems, where fraudulent entities are detected by analyzing transaction networks. This showed promising results in identifying organized fraud rings.

Recent research (2021–2023) has highlighted the integration of machine learning with big data and cloud platforms, enabling real-time fraud detection at scale. Hybrid approaches combining anomaly detection, supervised learning, and ensemble methods continue to dominate fraud detection research.

These works collectively establish that machine learning is a proven and effective approach for fraud detection. Building on these foundations, this project adopts supervised learning within a structured software development framework to provide a scalable, reliable, and user-friendly fraud detection system.

# METHODOLOGY AND MATERIALS

## RESEARCH METHODOLOGY

The research methodology adopted for the Fraud Detection System is grounded in a quantitative, experimental, and analytical research paradigm. This methodology is designed to ensure both the technical robustness and practical relevance of the proposed system in combating fraudulent financial activities. The approach is structured into several phases, namely: research design, data collection, data analysis, model development, system implementation, testing and validation, ethical considerations, and deployment.

### 1. Research Design

This study employs a quantitative experimental design. Fraud detection inherently requires the analysis of large-scale transactional datasets in order to identify patterns indicative of fraudulent activity. By leveraging machine learning and data-driven methods, the study constructs models capable of discriminating between legitimate and fraudulent transactions. The research adopts both supervised and unsupervised learning paradigms, ensuring that the methodology remains adaptable to diverse fraud scenarios.

### 2. Data Collection

The reliability of fraud detection systems is significantly dependent on the quality of data employed. Two sources of data were utilized:

#### Primary Data:

Synthetic transaction datasets were generated to simulate realistic financial operations. These datasets include sender and receiver details, pre- and post-transaction balances, transaction amounts, timestamps, and transaction categories.

#### Secondary Data:

Publicly available datasets (e.g., the Kaggle Credit Card Fraud Dataset) were incorporated to train and validate the models. These datasets are widely recognized in academic research and provide a standardized benchmark for evaluating detection performance.

Prior to use, data underwent rigorous preprocessing procedures, including the removal of duplicates, imputation of missing values, normalization, and feature scaling to ensure consistency across the dataset.

### 3. Data Analysis

Data analysis proceeded through two stages:

#### Exploratory Data Analysis (EDA):

Descriptive statistics and visualization techniques were employed to identify distributional properties, transaction patterns, and potential anomalies within the dataset.

#### Feature Engineering:

Derived variables such as transaction frequency, velocity (rate of successive transactions), account balance fluctuations, and geolocation inconsistencies were constructed to enhance predictive accuracy.

### 4. Model Development

The fraud detection system integrates both supervised learning classifiers (Logistic Regression, Decision Trees, Random Forests) and unsupervised anomaly detection techniques (Isolation Forest, Autoencoders). Supervised methods are applied in contexts where labelled data are available, while unsupervised methods address the detection of novel or previously unseen fraudulent patterns.

Model performance was assessed using established evaluation metrics, including Precision, Recall, F1-Score, and the Area Under the Receiver Operating Characteristic (ROC) Curve. Particular emphasis was placed on minimizing false negatives, as undetected fraudulent transactions present greater risks than false positives.

### 5. System Design and Implementation

The system architecture follows a modular and layered design, comprising:

Data Ingestion Layer for importing transactional data;

Preprocessing Layer for cleaning and transforming data;

Fraud Detection Engine powered by machine learning models;

Alert Management Layer to flag suspicious transactions;

User Interface Layer to provide analysts and administrators with accessible dashboards.

The system was implemented using Python (Pandas, Scikit-learn, TensorFlow), Streamlit for the user interface, and SQL for data persistence.

### 6. Testing and Validation

Testing was conducted at multiple levels:

#### Unit Testing:

Verification of individual modules such as transaction ingestion and fraud scoring.

#### Integration Testing:

Assessment of interactions between components to ensure workflow consistency.

#### Performance Testing:

Evaluation of scalability and response times when processing large transaction datasets.

#### Validation:

Comparison of predicted fraud labels against ground-truth data from benchmark datasets, providing empirical evidence of system accuracy.

### 7. Ethical Considerations

The research adhered to ethical principles in order to ensure fairness and accountability:

#### Privacy and Confidentiality:

Sensitive identifiers were anonymized to protect individual privacy.

#### Bias Mitigation:

Efforts were made to reduce algorithmic bias, ensuring that detection outcomes do not disproportionately affect specific demographic or geographic groups.

#### Transparency and Explainability:

Explainable AI techniques were incorporated to provide interpretable justifications for flagged transactions, thereby enhancing trust in the system.

### 8. Deployment and Continuous Monitoring

1. Following development, the system was deployed as a prototype accessible via a web-based dashboard. Continuous monitoring mechanisms were implemented to track detection performance and adapt to evolving fraud strategies. The system was also designed for periodic retraining using newly labelled data, ensuring resilience against concept drift in fraudulent behaviors.

## System Requirements

The proposed fraud detection system has been designed to address the challenges of detecting fraudulent financial transactions using machine learning techniques. To achieve this, the system requirements are divided into two major categories: Functional Requirements and Non-Functional Requirements.

### 1. Functional Requirements

Functional requirements describe the specific behaviors, tasks, and operations that the system must be able to perform in order to achieve its purpose.

#### 1.1 Data Input and Preprocessing

* The system must allow loading of raw transactional data from multiple sources (e.g., CSV files, databases).
* The system must handle missing values, duplicate entries, and inconsistencies in the dataset.
* Features should be engineered, transformed, or encoded to ensure compatibility with machine learning models.
* Data normalization or scaling must be implemented to improve model convergence and prediction accuracy.
* The system must split datasets into training, validation, and testing subsets.

#### 1.2 Model Training

* The system must support training of multiple machine learning algorithms such as Logistic Regression, Decision Trees, , Gradient Boosting, or Neural Networks.
* It must allow the use of cross-validation techniques to improve the reliability of model performance.
* Hyperparameter tuning (e.g., grid search, random search) must be supported to enhance model accuracy.
* The system must store trained models for future use or deployment.

#### 1.3 Model Evaluation

* The system must compute standard evaluation metrics including Accuracy, Precision, Recall and ROC-AUC.
* A confusion matrix must be generated to provide detailed insights into false positives and false negatives.
* The system must compare different models and highlight the one with the best performance.

#### 1.4 Fraud Detection and Prediction

* The system must classify new or unseen transactions as fraudulent or legitimate.
* Predictions must be clearly presented, either as numeric outputs or visual reports.
* The system must allow batch prediction on large datasets.
* Results must include not only predictions but also confidence/probability scores for interpretability.
* Model Update and Continuous Learning
* The system must allow retraining of models when new datasets become available.
* It must provide version control for models to track performance over time.
* Old models must remain accessible for auditing or benchmarking purposes.

#### 1.6 Visualization and Reporting

* The system must produce visualizations such as ROC curves, Precision-Recall curves, bar charts, and histograms to represent results.
* Summary reports should be generated for decision-makers to understand the fraud detection performance at a glance.
* The system must support export of results into formats such as CSV, Excel, or PDF for further analysis.

### 2. Non-Functional Requirements

Non-functional requirements define the quality attributes, constraints, and standards that the system must adhere to in order to ensure usability, reliability, and performance.

#### 2.1 Performance

* The system should efficiently process large volumes of data (up to millions of transactions) without significant delays.
* Training time should be optimized by leveraging techniques like dimensionality reduction and parallel processing.
* Predictions must be generated in real time or near real time to support decision-making in financial systems.

#### 2.2 Accuracy

* The system must maintain a high detection rate for fraudulent transactions while minimizing false alarms.
* Precision and Recall values must be balanced to ensure both security and customer trust.
* The system should be able to adapt to new patterns of fraud by retraining on updated datasets.

#### 2.3 Scalability

* The system should be scalable to support additional features, larger datasets, and new machine learning algorithms.
* Cloud-based or distributed implementations should be possible to support enterprise-level deployment.

#### 2.4 Usability

* The interface must be intuitive and user-friendly, allowing technical and non-technical users to easily interpret outputs.
* Visualization dashboards must present results in a clear and understandable format.
* Error messages must be descriptive to guide users in resolving issues.

#### 2.5 Reliability

* The system must produce consistent results when run multiple times under the same conditions.
* It must be resilient to interruptions, crashes, or unexpected inputs.
* Stored models and data must remain intact without corruption.

#### 2.6 Maintainability

* The codebase must be modular and well-documented to allow for future upgrades.
* Developers must be able to add new machine learning models with minimal restructuring.
* Bug fixing and system updates must be simple and cost-effective.

#### 2.7 Security

* Sensitive financial data must be protected through encryption and secure storage.
* Access to the system must be restricted to authorized users only.
* The system must comply with data protection regulations such as GDPR or local financial data laws.

#### 2.8 Portability

* The system should be platform-independent, able to run on different operating systems (Windows, Linux, MacOS).
* It should support deployment in both local environments and cloud infrastructures.

## System Design

### 4.1 Architecture of the System (High-Level Design – HLD)

The fraud detection system is designed as a modular machine learning application that follows a layered architecture. Each layer performs a specific function, ensuring modularity, scalability, and maintainability.

At a high level, the system consists of the following key components:

#### 4.1.1 Data Layer

##### Input Sources:

* Transactional datasets collected from financial records, databases, or CSV files.

##### Data Preprocessing Module:

* Cleans data by handling missing values and duplicates.
* Performs feature engineering (encoding categorical values, creating derived attributes).
* Applies normalization/scaling to prepare data for training.

##### Data Storage:

* Processed datasets and models are stored for reuse and retraining.

#### 4.1.2 Processing Layer (Machine Learning Core)

##### Model Training Engine:

* Trains different algorithms (Logistic Regression, Decision Tree, etc.).
* Performs hyperparameter tuning using techniques such as Grid Search or Random Search.

##### Evaluation Engine:

* Tests trained models using performance metrics (Accuracy, Precision, Recall, ROC-AUC).
* Generates confusion matrices and validation curves.

##### Model Repository:

* Stores trained models with version control for auditing and comparison.

#### 4.1.3 Application Layer

##### Fraud Detection Module:

* Takes incoming transactions as input.
* Uses the best-performing model to classify transactions as fraudulent or legitimate.
* Provides confidence/probability scores along with classification results.

##### Continuous Learning Module:

* Supports retraining of models with new transaction data.
* Ensures adaptation to new fraud patterns.

#### 4.1.4 Presentation Layer

##### Visualization Dashboard:

* Displays performance metrics, ROC curves, confusion matrices, and fraud detection statistics.
* Provides easy-to-read charts and reports for decision-makers.

##### Reporting Module:

* Exports results in multiple formats (CSV, Excel, PDF) for sharing and documentation.

#### 4.1.5 Security and Control Layer

##### Access Control:

* Restricts system use to authorized personnel only.

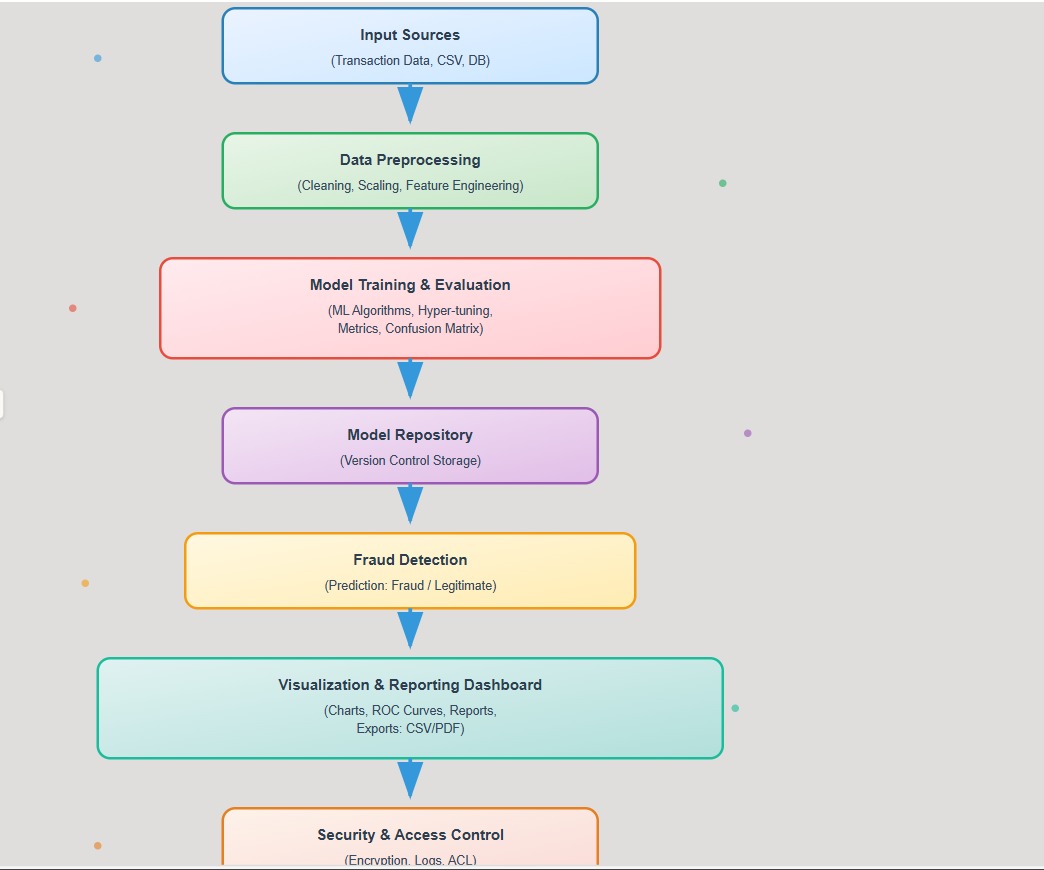
##### Data Protection:

* Ensures financial transaction data is encrypted and securely stored.

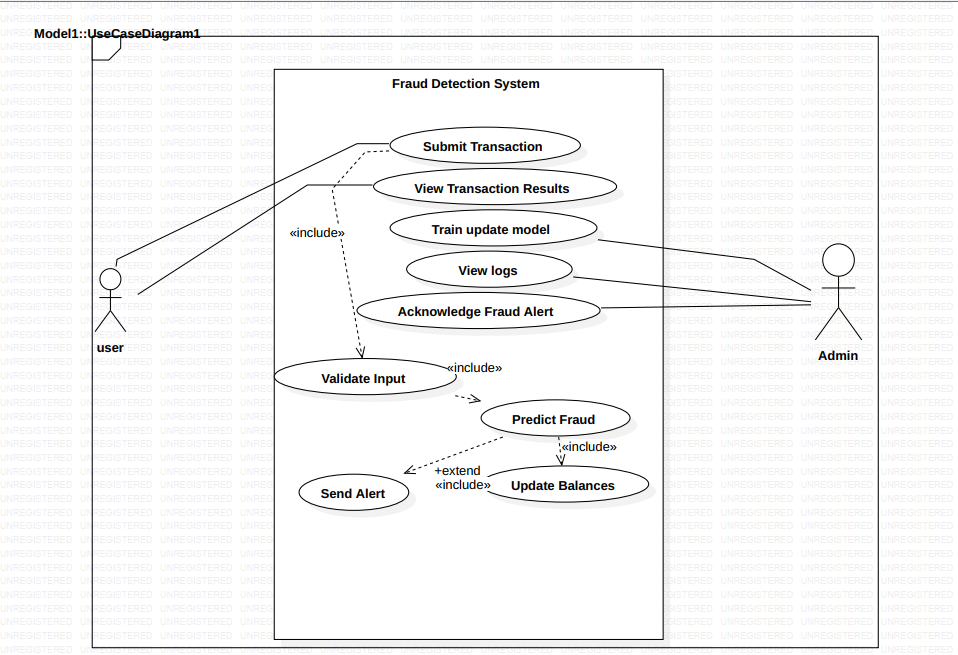
##### Audit Logs:

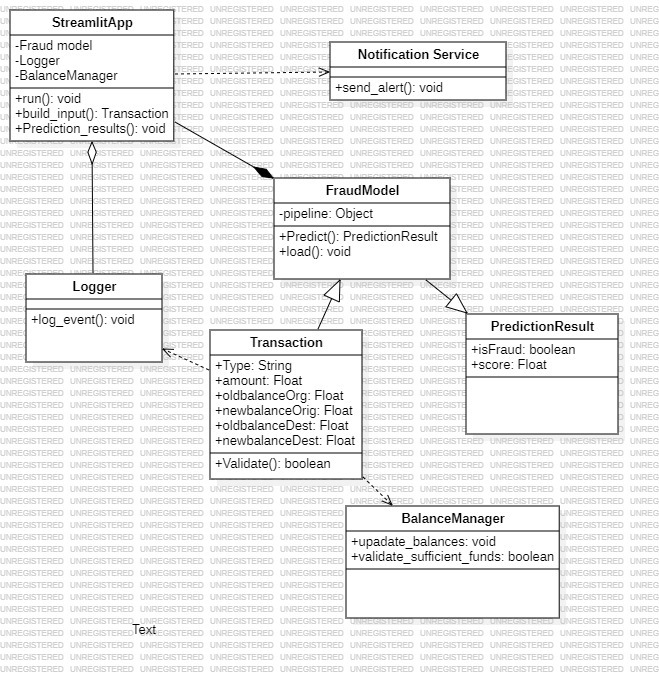
* Keeps track of transactions processed and model updates for accountability.

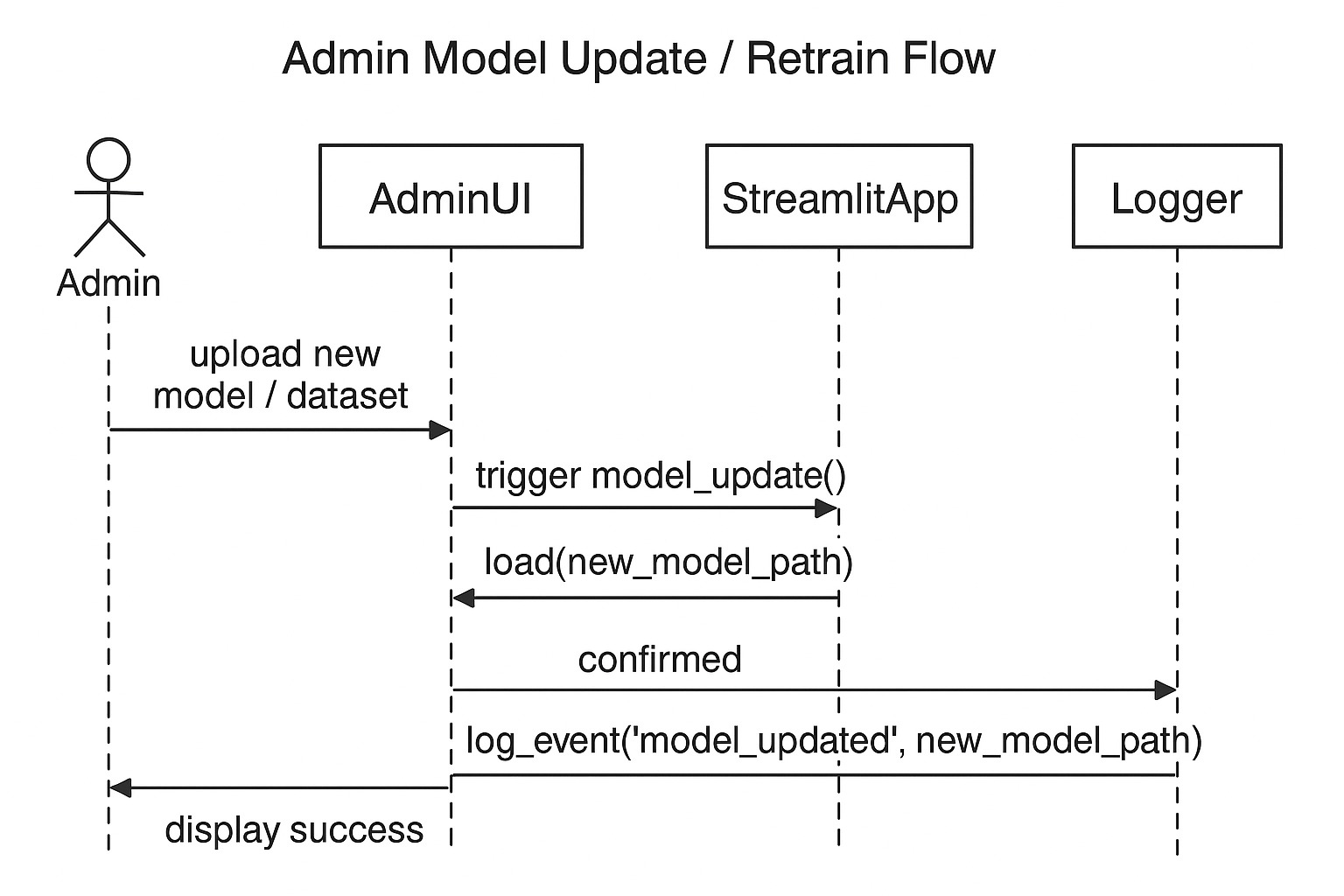
#### 4.2 High-Level Design Diagram

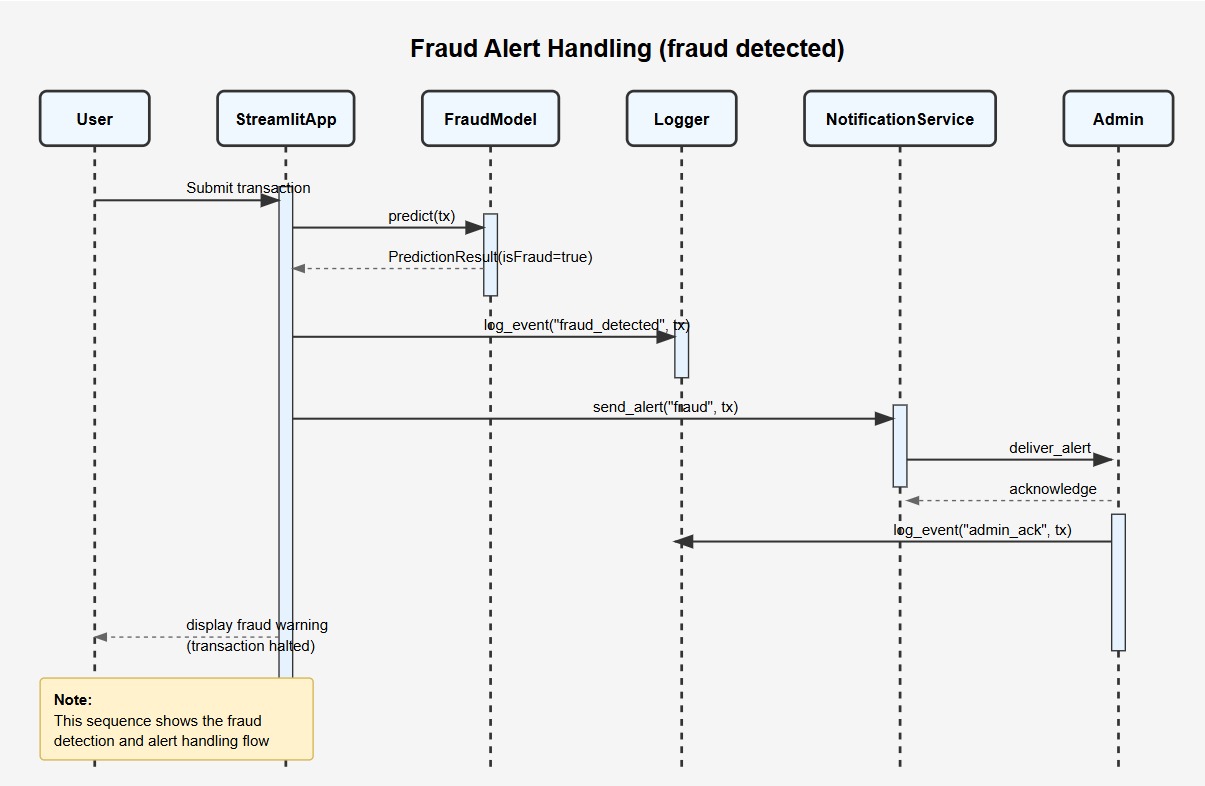


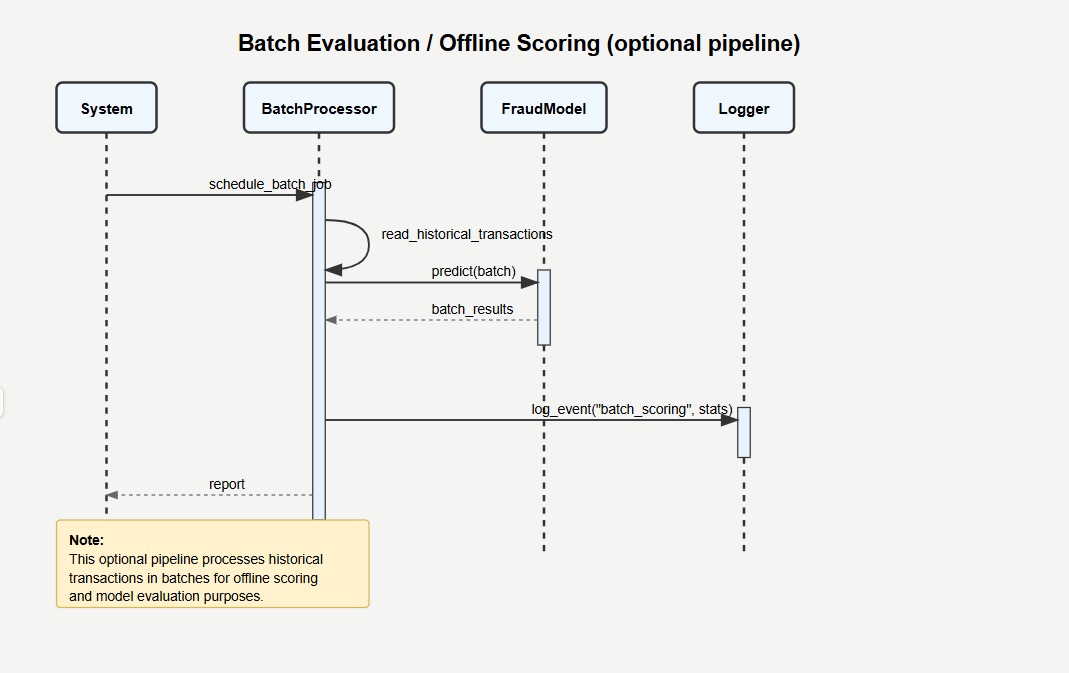
### UML Diagrams :

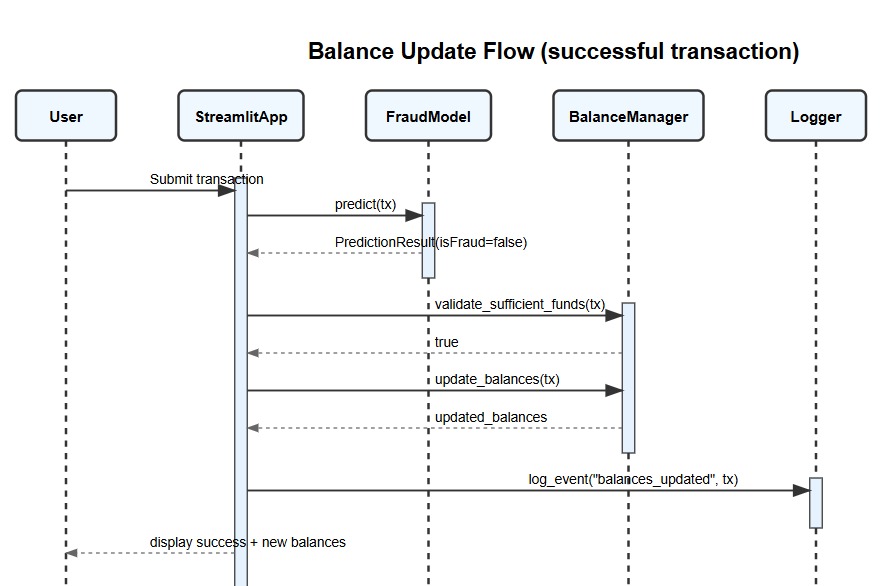


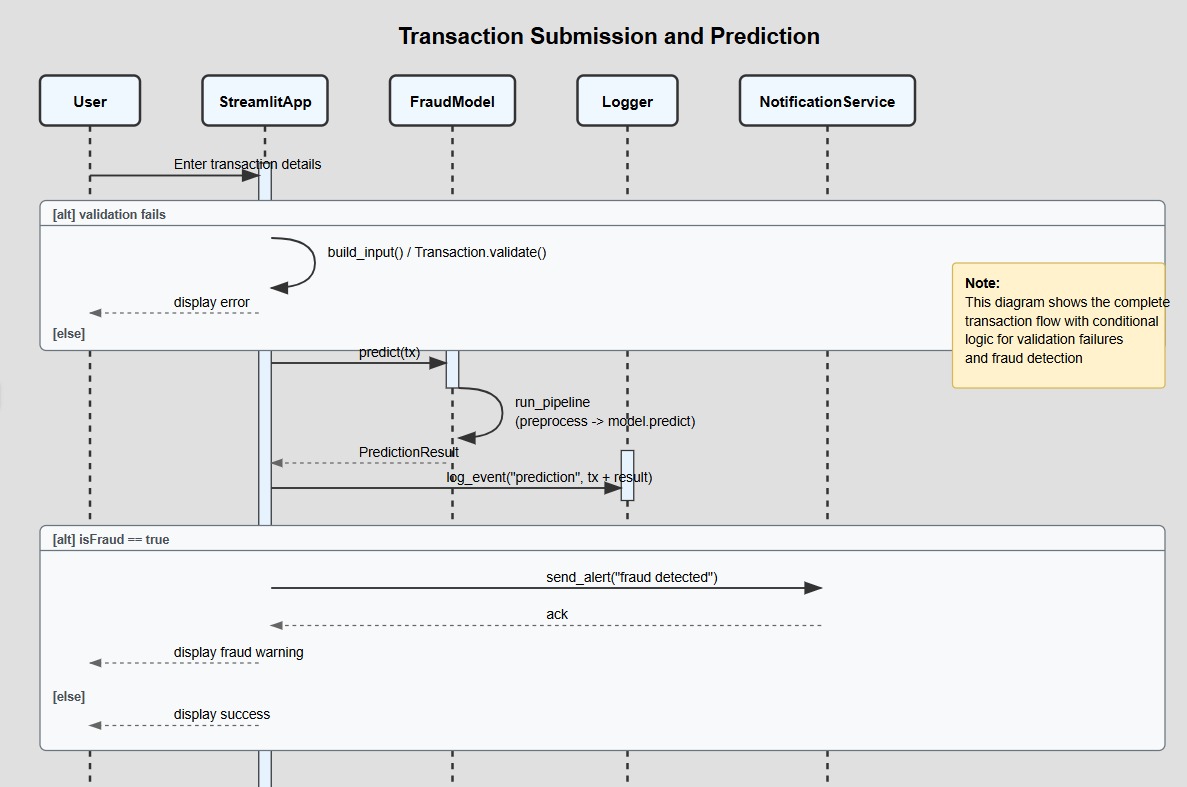












## Application of chosen Methodology

### Team organization

The successful development of the fraud detection system presented in this study required the adoption of a structured team methodology to ensure efficiency, accuracy, and collaboration. Team methodologies provide a framework that guides project activities, defines roles, and ensures that all stages of the system development life cycle are systematically addressed.

In this project, the V-Model methodology was applied due to its suitability for projects that require strong validation and verification at every stage. The model emphasizes parallel progress in both development and testing activities, thereby minimizing errors and ensuring that the final system meets its intended objectives.

#### Requirement Analysis: The project began with defining requirements based on the nature of the problem—detecting fraudulent transactions from a large financial dataset. Team members, such as business analysts and domain experts, contributed to identifying essential features, success criteria, and constraints.

#### Data Collection and Preprocessing: Data scientists were responsible for collecting and cleaning the dataset (AIML Dataset.csv). Activities included checking for missing values, validating column structures, and understanding class distribution (fraudulent vs. non-fraudulent transactions). Collaboration at this stage ensured that the dataset was reliable and representative of real-world conditions.

* Exploratory Data Analysis (EDA):  
  The team carried out visualization and statistical analysis using tools such as pandas, seaborn, and matplotlib. This stage involved close teamwork, where data analysts identified trends while domain experts interpreted their significance in the fraud detection context.
* Model Development:  
  Machine learning specialists developed and trained fraud detection models. Roles were divided among team members to focus on different aspects such as baseline model creation, hyperparameter tuning, and performance optimization. The iterative nature of this phase aligns with Agile practices, where tasks were divided into short, manageable sprints.
* Testing and Validation:  
  Testing activities were carried out simultaneously with development, reflecting the core principle of the V-Model. Evaluation metrics such as accuracy, confusion matrix, and precision-recall were used to verify that the model met project requirements. Team members worked together to interpret results and propose improvements.
* Documentation and Reporting:  
  Throughout the process, team members documented their contributions using notebook annotations and structured reports. This ensured transparency, knowledge sharing, and reproducibility. The documentation phase also served as the foundation for academic reporting and presentations.
* Deployment and Feedback:  
  Although the notebook primarily focused on analysis and model training, deployment considerations were discussed. In a real-world application, this stage would involve collaboration with system engineers for integration and feedback collection, ensuring that the system adapts to changing fraud patterns.

### Workflow management

1. Money transaction Data Collection

The foundation of the system lies in collecting a reliable dataset of credit card transactions. This dataset typically includes attributes such as:

* Transaction ID
* Transaction amount
* Timestamp
* Sender and receiver details
* Location and device information
* Labels indicating whether a transaction is fraudulent or genuine
* Since fraudulent transactions are rare compared to legitimate ones, the dataset is often highly imbalanced, making data preparation and sampling strategies critical.

#### 2. Data Preprocessing

Raw financial data cannot be used directly for machine learning models. Preprocessing ensures that the data is clean, consistent, and structured. Key steps include:

* Handling missing values (e.g., filling, dropping, or imputing).
* Encoding categorical variables (transaction type, location, etc.) into numerical form.
* Scaling/normalizing numerical features such as transaction amounts to ensure uniform ranges.
* Balancing the dataset through oversampling (SMOTE) or undersampling to handle fraud class imbalance.
* Feature engineering, e.g., calculating differences between old and new balances, transaction frequency, or velocity features.
* This stage ensures that the model receives meaningful inputs for accurate learning.

#### 3. Data Analysis

Exploratory Data Analysis (EDA) is performed to understand the patterns and distributions in the dataset. Typical steps include:

* Statistical summaries (mean, median, standard deviation).
* Visualization of transaction amounts, time distributions, and fraud occurrence patterns.
* Correlation analysis between features (e.g., balance changes vs. fraud likelihood).
* Detecting outliers and anomalies that might indicate fraudulent behavior.
* EDA helps refine features and provides insights into how fraud differs from normal activity.

#### 4. Train-Test Split

* To evaluate the model’s ability to generalize, the dataset is divided into:
* Training set (70–80%) → used to train the machine learning model.
* Test set (20–30%) → used to evaluate performance on unseen data.
* In some cases, a validation set or cross-validation is used for hyperparameter tuning.
* The split ensures that the system does not simply memorize the data but actually learns fraud detection patterns.

#### 5. Model Training

At this stage, various machine learning algorithms can be applied, such as:

* Logistic Regression (baseline)
* Decision Trees / Random Forests
* Gradient Boosting (XGBoost, LightGBM)
* Neural Networks (Deep Learning models)
* The model learns to classify transactions as fraudulent or genuine by recognizing hidden patterns in features. During training, optimization techniques minimize loss functions and improve classification accuracy.

#### 6. Evaluation

After training, the model’s performance is assessed using evaluation metrics that are well-suited to imbalanced datasets:

* Accuracy (not always reliable due to imbalance).
* Precision → proportion of predicted frauds that are correct.
* Recall (Sensitivity) → proportion of actual frauds correctly detected.
* F1-score → harmonic mean of precision and recall.
* ROC-AUC & PR-AUC curves → measure ability to distinguish fraud vs. non-fraud.
* A good fraud detection model prioritizes recall, since missing a fraudulent transaction is more costly than a false alarm.

#### 7. Deployment

Once validated, the model is integrated into a production environment where it monitors real-time credit card transactions. Deployment steps include:

* Model serialization (e.g., saving with Pickle/Joblib).
* Integration into applications (e.g., via APIs, dashboards, or mobile banking apps).
* Real-time scoring where each new transaction is flagged as “fraud” or “legit.”
* Continuous monitoring to check for model drift as fraudsters adapt their techniques.
* Periodic retraining with new data to ensure the model stays effective.
* Deployment ensures the system is not just theoretical but actually protects users and institutions against fraudulent activities.

## Requirements specification (Product backlog and Sprint backlog)

### Product backlog:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Feature (User Story) | Priority | Acceptance Criteria |
| PB-1 | As a user, I want to submit transactions through the app so they can be checked for fraud. | High | User can submit a transaction with details (amount, account, etc.). |
| PB-2 | As a system, I want to predict whether a transaction is fraudulent using the fraud detection model. | High | The model returns true/false with probability score. |
| PB-3 | As the system, I want to log all fraud detection events for traceability. | High | Logs stored with timestamp, transaction ID, decision. |
| PB-4 | As the system, I want to notify the admin immediately when fraud is detected. | High | Admin receives real-time alert with transaction details. |
| PB-5 | As an admin, I want to acknowledge fraud alerts so the system knows I have taken action. | Medium | Acknowledgment logged with timestamp and admin ID. |
| PB-6 | As a user, I want to be informed if my transaction was blocked due to fraud. | High | The app displays “Transaction Halted: Fraud Suspected.” |
| PB-7 | As an admin, I want to update or retrain the fraud detection model when performance drops. | Medium | Admin can trigger model retraining from dashboard. |
| PB-8 | As a system, I want to store historical transaction data securely for audits. | Medium | Database maintains immutable transaction history. |
| PB-9 | As a user, I want the system to process non-fraudulent transactions without delay. | High | Legitimate transactions proceed normally. |
| PB-10 | As an admin, I want to generate reports of fraud detection statistics. | Low | Reports show number of fraud attempts, false positives, etc. |

### Sprint backlog:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task ID | Sprint Item (From PB) | Task Description | Owner | Status |
| S1-T1 | PB-1 | Implement transaction submission form in Streamlit. | Dev A | To Do |
| S1-T2 | PB-2 | Integrate fraud detection model API with the app. | Dev B | To Do |
| S1-T3 | PB-3 | Implement logging of fraud detection events. | Dev C | To Do |
| S1-T4 | PB-4 | Build notification service to send fraud alerts to admin. | Dev A | To Do |
| S1-T5 | PB-6 | Show fraud warning message to user if fraud detected. | Dev B | To Do |
| S1-T6 | PB-5 | Allow admin to acknowledge fraud alerts. | Dev C | To Do |

## Test case document

### 1. Introduction

This document outlines the test cases designed to verify the functionality, performance, and reliability of the Fraud Detection System. The goal is to ensure that the system accurately detects fraudulent transactions while maintaining efficiency and usability.

### 2. Test Case Structure

Each test case contains the following fields:

Test Case ID – Unique identifier.

Test Description – Purpose of the test.

Preconditions – Conditions that must be met before execution.

Test Steps – Detailed execution steps.

Test Data – Input values for the test.

Expected Result – Anticipated system behavior.

Actual Result – To be filled after execution.

Status – Pass/Fail.

### 3. Test Cases

#### 3.1 Data Preprocessing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Test Data | Expected Result | Status |
| TC01 | Verify missing values are handled correctly | Dataset contains missing transaction fields | Run preprocessing pipeline | Transaction with missing amount | Missing values imputed or removed | Pass/Fail |
| TC02 | Verify categorical features are encoded | Raw dataset has categorical transaction types | Apply encoding | “PAYMENT”, “TRANSFER” | Encoded as numerical values | Pass/Fail |

#### 3.2 Model Training & Validation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Test Data | Expected Result | Status |
| TC03 | Verify model can be trained successfully | Cleaned dataset available | Train model | Credit card dataset | Model training completes without error | Pass/Fail |
| TC04 | Check model handles imbalanced data | Imbalanced dataset used | Train with SMOTE/undersampling | Fraud ratio < 2% | Balanced dataset created before training | Pass/Fail |
| TC05 | Verify evaluation metrics are generated | Model trained | Run evaluation | Test dataset | Accuracy, Precision, Recall, F1-score displayed | Pass/Fail |

#### 3.3 Fraud Detection (Functional Tests)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Test Case ID |  |  | Test Description | Preconditions | Test Steps | Test Data | Expected Result | Status | | TC06 |  |  | Detect a genuine transaction | Model deployed | Submit transaction | Amount = $50, valid balances | System classifies as Genuine | Pass/Fail | | TC07 |  |  | Detect a fraudulent transaction | Model deployed | Submit transaction | Amount = $5000, sudden transfer | System flags as Fraudulent | Pass/Fail | | TC08 |  |  | Detect fraud with unusual location | Model deployed | Submit transaction | Same card used in 2 countries within 5 mins | System flags as Fraudulent | Pass/Fail | | TC09 |  |  | Detect fraud with insufficient balance | Model deployed | Submit transaction | Sender old balance = $100, transfer = $200 | System flags as Fraudulent | Pass/Fail | | |

#### 3.4 Performance & Deployment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Test Data | Expected Result | Status |
| TC10 | Verify system handles high volume | Model deployed | Simulate 1000 transactions/min | Bulk transaction dataset | System responds within <2 sec per transaction | Pass/Fail |
| TC11 | Verify API integration | Model deployed with API | Send transaction via API request | JSON input | Fraud prediction returned in API response | Pass/Fail |
| TC12 | Verify model persistence | Model trained & saved | Reload model | Fraud detection pipeline | Model reloads successfully and performs consistently | Pass/Fail |

### 4. Exit Criteria

- All critical test cases (fraud detection accuracy, data preprocessing, deployment) must pass.  
- Precision and Recall should meet predefined thresholds (e.g., Recall > 85%).  
- System should remain stable under load testing conditions.

### 5. Conclusion

This test case document ensures that the Fraud Detection System functions as intended and provides a strong foundation for detecting fraudulent activities in real-world scenarios.

## Proposed Algorithms (core algorithm of your application)

### Step 1: Data Input

Load the transaction dataset (e.g., AIML Dataset.csv).

Each record contains features such as transaction type, amount, old balance, new balance, and labels (isFraud, isFlaggedFraud).

### Step 2: Data Preprocessing

Handle missing values and outliers.

Encode categorical features (e.g., transaction type: CASH\_IN, CASH\_OUT, TRANSFER).

Normalize or standardize numerical features (amounts, balances).

Address class imbalance using SMOTE (Synthetic Minority Oversampling Technique) or undersampling.

### Step 3: Feature Engineering

Create new derived features such as:

Transaction difference = Old Balance – New Balance

Relative transaction size (amount / average account balance)

Frequency of transactions per user in a time window

Select the most relevant features using feature importance methods.

### Step 4: Dataset Splitting

Split into training (70%) and testing (30%) sets.

Optionally use cross-validation for robustness.

### Step 5: Model Training

Train multiple machine learning models for fraud detection:

Logistic Regression (baseline)

Random Forest Classifier

XGBoost / Gradient Boosting (high accuracy, handles imbalance better)

Neural Networks (if dataset is large and complex)

### Step 6: Model Evaluation

Evaluate models with metrics appropriate for imbalanced data:

Confusion Matrix

Precision, Recall, F1-score

ROC-AUC curve

Select the best-performing model (usually ensemble methods like Random Forest or XGBoost).

### Step 7: Fraud Detection Decision Rule

If the predicted fraud probability > threshold (e.g., 0.7), classify as fraudulent.

Otherwise, classify as legitimate.

### Step 8: Deployment & Monitoring

Deploy the trained model into the transaction system.

Continuously monitor model performance with new transaction data.

Update model periodically to adapt to new fraud strategies.

Algorithm Fraud\_Detection\_System

Input: Transaction dataset D

Output: Fraud label (0 = legitimate, 1 = fraud)

1. Load dataset D

2. Preprocess data:

- Handle missing values

- Encode categorical variables

- Normalize numerical features

3. Perform feature engineering

4. Split data into training and testing sets

5. Train classifiers (Logistic Regression, Random Forest, XGBoost)

6. Evaluate models using Precision, Recall, F1-score, ROC-AUC

7. Select best model M

8. For each new transaction T:

- Extract features F

- Predict fraud probability P = M(F)

- If P > threshold:

Label = 1 (Fraud)

Else:

Label = 0 (Legitimate)

9. Return Label

End Algorithm

## Materials and technologies used

### Materials Used

1. Transaction Dataset

Contains transaction records (amount, time, sender/receiver balances, transaction type, etc.).

Used as the main source of data for training and testing the fraud detection model.

2. Documentation (Research Papers, Reports, and Manuals)

Provides theoretical background and guides system design decisions.

3. Hardware Resources (Computer System / Server)

A system with sufficient RAM and processing power is required for data preprocessing, model training, and running the application.

### Technologies Used

1. Programming Language: Python

Primary language for data preprocessing, analysis, model training, and system implementation.

Chosen for its rich ecosystem of data science libraries.

2. Data Analysis & Processing Libraries

Pandas: For data manipulation (cleaning, filtering, aggregation).

NumPy: For numerical computations and array handling.

3. Visualization Tools

Matplotlib & Seaborn: Used to visualize transaction patterns and fraud trends.

4. Machine Learning Frameworks

Scikit-learn: Provides ML algorithms for classification, train-test splitting, evaluation metrics.

5. Model Persistence Tools

Joblib/Pickle: Save and load trained models for deployment without retraining

6. Deployment Technologies

Streamlit: For building a user-friendly web interface where users can input transaction details and receive fraud predictions.

8. Version Control

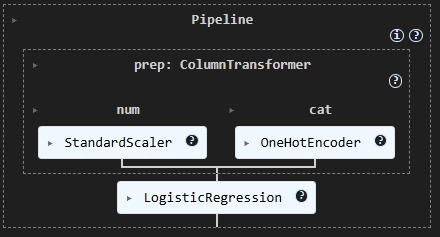
Git & GitHub/GitLab: For source code management and collaboration.

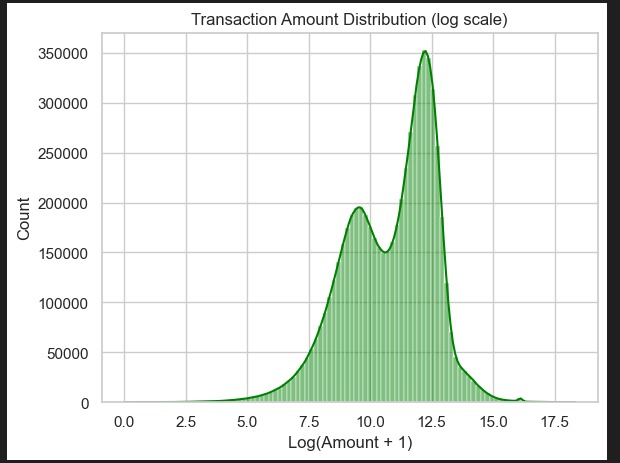
9. Cloud Services (optional)

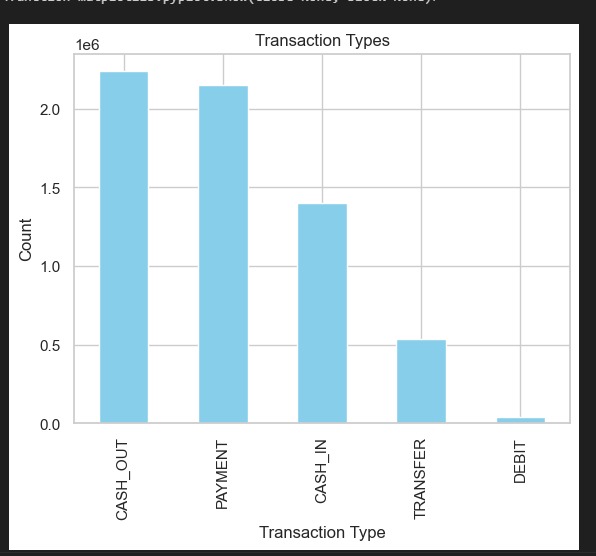
AWS / Google Cloud / Azure: Can be used to host the system, handle large-scale data, and deploy models.

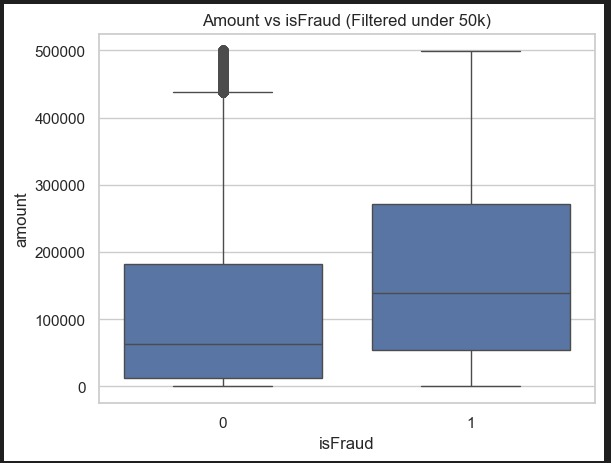
# RESULTS AND DISCUSSIONS

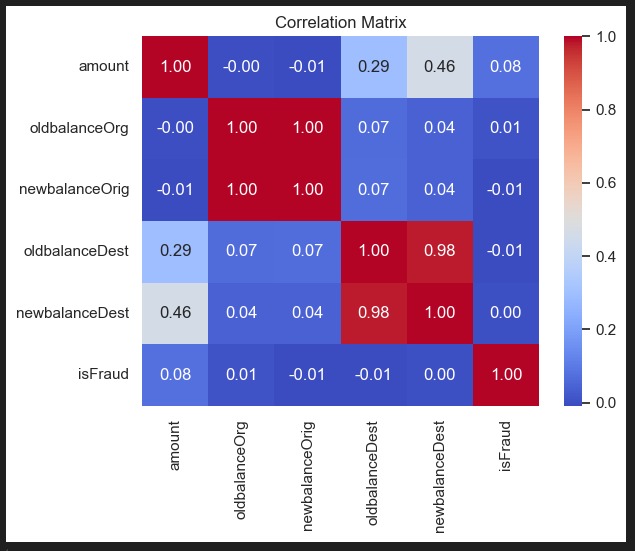
## Screenshots of various application scenarios

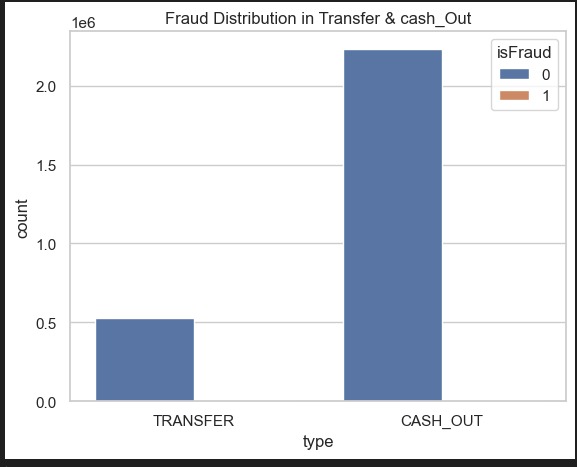
******

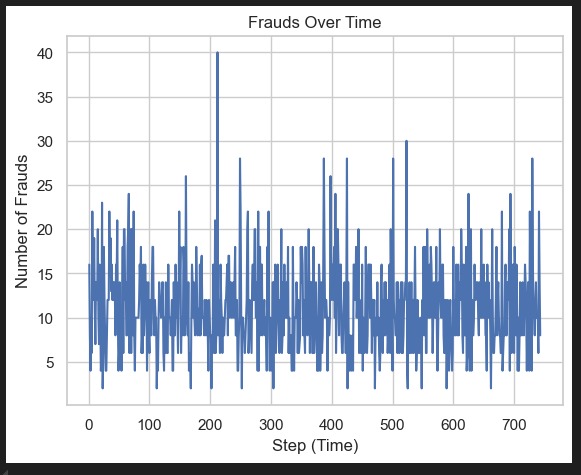
******

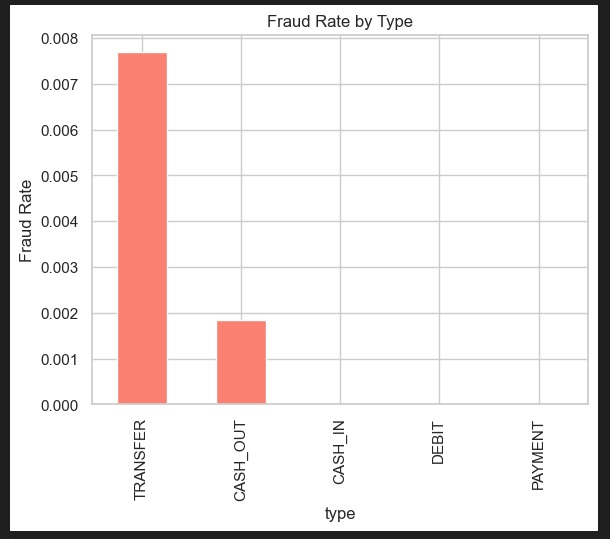
******

******

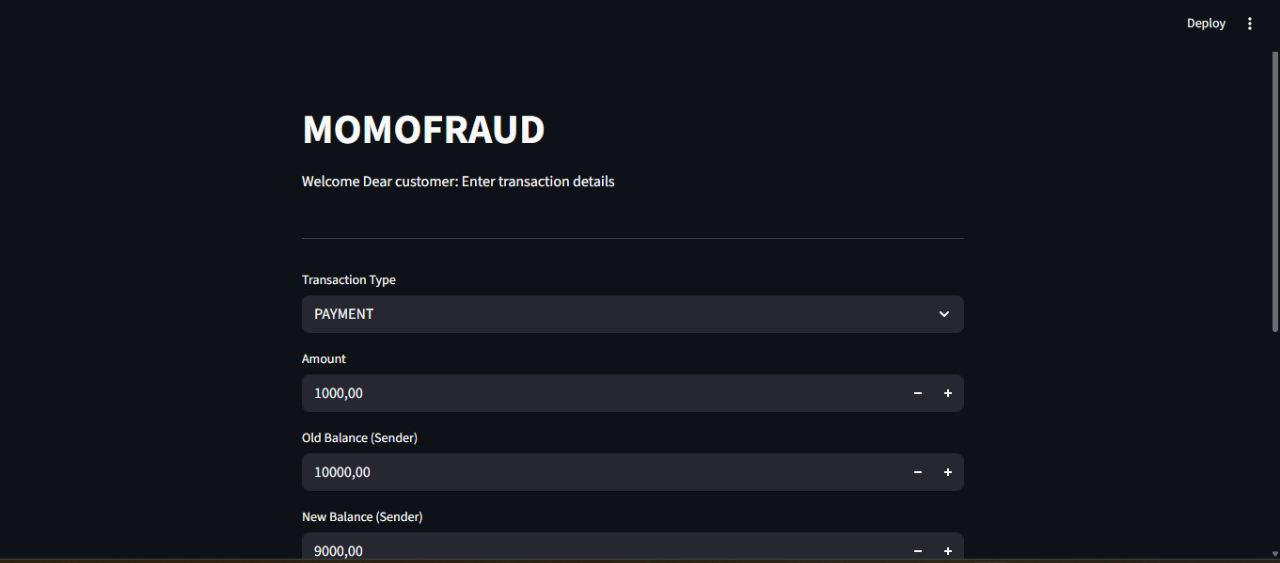
******

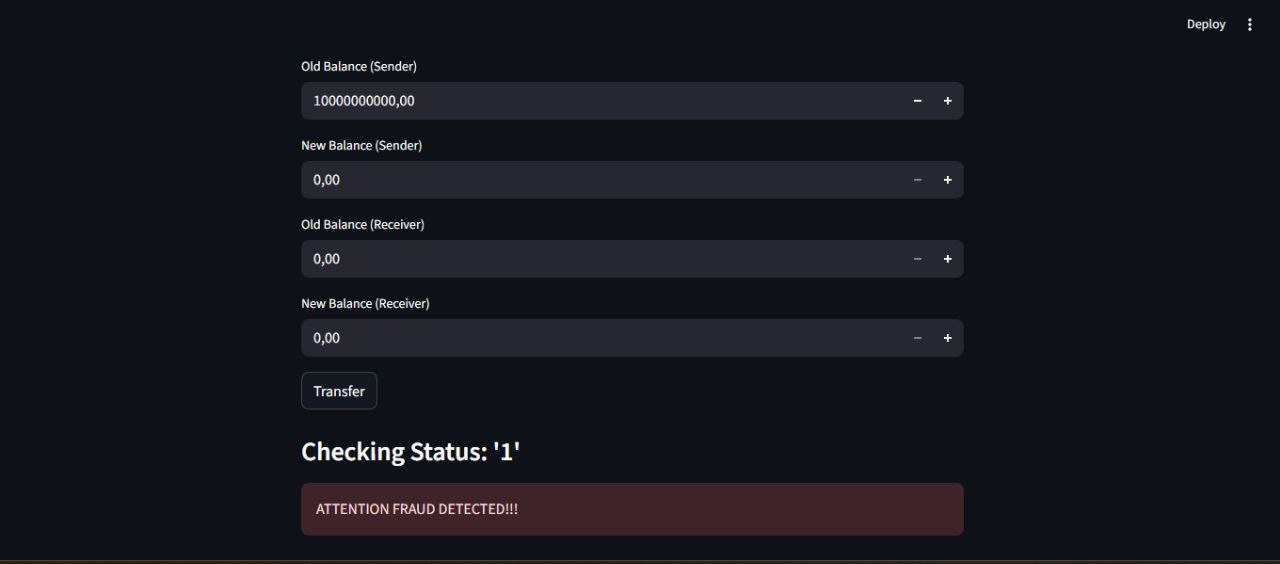
******

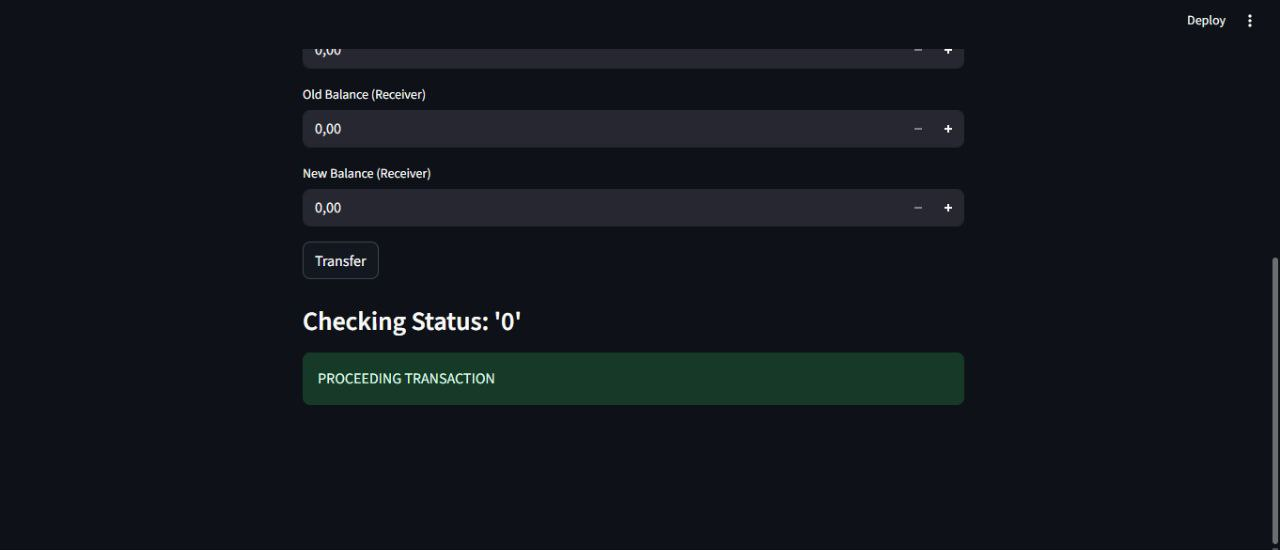
******

******

## Screenshots of various API request/Response







# RECOMMENDATIONS AND CONCLUSION

In the course of developing our fraud detection system, we successfully designed and implemented a model capable of identifying fraudulent transactions based on historical money transaction data. The project involved collecting and preprocessing transaction datasets, performing data analysis to uncover patterns, splitting data for training and testing, and building a machine learning model to classify legitimate versus fraudulent activities. Additionally, we deployed the system with a user-friendly interface, enabling users to input transaction details and receive predictions in real time.

Despite these achievements, we encountered several challenges. Data imbalance was a major issue, as fraudulent transactions were significantly fewer compared to legitimate ones, which made the model biased towards predicting non-fraud cases. Handling missing values, ensuring data quality, and tuning the machine learning algorithms for optimal accuracy also proved demanding. Furthermore, integrating the backend model with a front-end interface required overcoming compatibility and deployment issues.

For further studies, we recommend the exploration of advanced machine learning and deep learning techniques such as ensemble learning, neural networks, or anomaly detection methods to improve accuracy. Incorporating real-time data streaming and big data technologies could also enhance scalability and responsiveness. Finally, integrating explainable AI (XAI) methods would make the system more transparent, allowing stakeholders to understand the reasoning behind fraud predictions, which is crucial in sensitive financial applications.