**RISE KRISHNA SAI GANDHI GROUP OF INSTITUTIONS**

Valluru-523272

(Affiliated to JNTU, KAKINADA)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SMARTINTERNZ VIRTUAL INTERNSHIP PROGRAM-2025**

# TITLE: FRAUDSHIELD: Advanced Online Payment Fraud Detection Using Machine Learning

**Submitted by**

**Team Id: LTVIP2026TMIDS61558**

**Team leader: Ullaganti Nagur Vali** (228B1A0555)

**Team member:** Manjula Rakesh (228B1A0547)

**Team member:** Gunja Sai Kalyan (228B1A0538)

**Team member:** Kadhiri Sai Manoj Kumar (228B1A0541)

**Date Of Submission:** 16-02-26

# CERTIFICATE

This is to certify that the project report titled "**Fraud Shield: Advanced Online Payment Fraud Detection Using Machine Learning"** is a bona fide work carried out by the following students:

- **Ullaganti Nagur Vali** (228B1A0555)
- Manjula Rakesh (228B1A0547)
- Gunja Sai Kalyan (228B1A0538)
- Kadhiri Sai Manoj Kumar (228B1A0541**)**

of **RISE KRISHNA SAI GANDHI GROUP OF INSTITUTIONS,** in partial fulfillment of the requirements for the SmartInternz Virtual Internship Program-2026, during the period from 01-12-2025 to 16-02-2026

The work embodied in this project report has not been submitted to any other institution for the award of any degree or diploma.

**Endorsements:**

**Project Guide:** Mr.P.Suresh

**Head of Department:** Mr.Ch.Hari Krishna

**Principal:** Dr.K.V.Subrahmanyam

# TABLE OF CONTENTS

# 1. INTRODUCTION

Online transactions have increased rapidly due to digital banking, UPI, credit cards, and e-commerce platforms. While this growth improves convenience, it also increases the risk of **fraudulent activities** such as unauthorized transfers, fake payments, and account misuse. Financial fraud leads to huge economic losses for both individuals and institutions.

Traditional fraud detection methods rely on manual monitoring or fixed rule-based systems, which are slow and often fail to detect new fraud patterns. Machine Learning enables intelligent systems that can learn from historical transaction data and automatically classify whether a transaction is **fraudulent or safe**.

This project builds a web-based fraud detection system using Machine Learning and Flask, allowing users to input transaction details and instantly receive a prediction.

## 1.1 Project Overview

The **Online Payment Fraud Detection System** is a Machine Learning web application that predicts whether a financial transaction is fraudulent or legitimate.

The system uses classification algorithms such as **Random Forest** to analyze transaction features like:

- Transaction Step
- Transaction Type
- Amount
- Old Balance
- New Balance

The trained model is saved as a `.pkl` file and integrated with a Flask web interface where users can submit inputs and receive real-time fraud predictions

## 1.2 Purpose

The main purpose of this project is:

- Detect fraudulent transactions instantly
- Assist financial institutions in reducing losses
- Provide users with early warnings
- Demonstrate practical Machine Learning deployment
- Improve cybersecurity awareness

# 2. IDEATION PHASE

The ideation phase focused on identifying the growing issue of online payment fraud and understanding its impact on users and financial institutions. We analyzed possible solutions and selected Machine Learning as the most effective approach for automatic fraud detection. The final idea was to develop a Flask-based web application integrated with a trained Random Forest model to predict fraudulent transactions in real time.

## 2.1 Problem Statement

In Online payment systems are vulnerable to fraud due to large transaction volumes and evolving attack methods. Manual detection is inefficient and inaccurate.
**Problem:** How can we automatically detect fraudulent transactions using Machine Learning?

## 2.2 Empathy Map Canvas

Users feel anxious and worried about losing money due to unauthorized or fraudulent transactions. They often check bank statements frequently and rely on alerts to stay informed about suspicious activities. Their main expectation is a secure and reliable system that can instantly detect fraud and protect their financial information.

| **Section** | **Insight** |
|---|---|
| **Think & feel** | Fear of losing money, distrust in online payments |
| **Hear** | News about scams and fraud cases |
| **See** | Fake transactions, phishing messages |
| **Say & do** | Check bank statements, report suspicious activity |
| **Pain** | Financial loss, stress, wasted time |
| **Gain** | Secure transactions, instant alerts |

## 2.3 Brainstorming

During brainstorming, we explored different approaches such as rule-based detection, statistical analysis, and Machine Learning classification models. After comparing accuracy and efficiency, we selected the Random Forest algorithm for its strong performance in handling complex transaction patterns. Finally, we decided to integrate the trained model into a Flask-based web application to provide real-time fraud prediction.

**Ideas Explored**

- Rule-based detection
- Machine Learning classification
- Real-time alerts
- Web application interface

**Final Approach**

- Random Forest Classifier
- Flask Web App
- User input predict

# 3. REQUIREMENT ANALYSIS

The requirement analysis phase focused on identifying system needs, user expectations, and technical specifications for the fraud detection system. Functional requirements include accepting transaction inputs, processing data, and generating real-time fraud predictions. Non-functional requirements emphasize high accuracy, fast response time, security, and a user-friendly web interface.

## 3.1 Customer Journey Map

The user enters transaction details into the web application and submits the form to check whether the transaction is fraudulent or safe. The system processes the input using the trained model and instantly displays the prediction result (0 for safe, 1 for fraud) to help the user make informed decisions

| **Stage** | **Customer Actions** | **Pain Points** | **Opportunities** |
|---|---|---|---|
| Awareness | Learns about fraud risks | Lack of tools | Demo system |
| Input Submission | Enters transaction details | Confusion | Clear UI |
| Prediction Review | Views result | Unsure meaning | Show 0/1 clearly |
| Action | Decides safety | Doubt | Confidence through ML |

## 3.2 Solution Requirement

**Functional**

- Accept transaction inputs
- Load trained model
- Predict fraud
- Display result

**Non-Functional**

- Fast response (<2 sec)
- High accuracy
- Simple UI
- Secure handling

## 3.3 Data Flow Diagram (DFD)

User → Web Form → Flask Backend → ML Model (.pkl) → Prediction → Result Page.

## 3.4 Technology Stack

| **Layer** | **Technology Used** |
|---|---|
| Programming | Python |
| ML Libraries | scikit-learn, pandas, numpy |
| Web Framework | Flask |
| Front-End | HTML, CSS |
| Tools | HTML, CSS |
| Deployment | Localhost |

# 4. PROJECT DESIGN

Project design outlines the alignment between the identified problem and the selected solution, along with a structured representation of how the system is implemented. This phase ensures that the solution is both technically sound and aligned with the users' needs.

## 4.1 Problem Solution Fit

Fraud detection requires automated, fast, and accurate classification. Machine Learning models fit perfectly because they learn patterns from historical data.

The increasing risk of online payment fraud requires an intelligent system capable of detecting suspicious transactions accurately and quickly. By applying a Machine Learning model such as Random Forest to analyze transaction patterns, the proposed solution effectively addresses this problem by providing real-time fraud classification through a simple web interface

## 4.2 Proposed Solution

The proposed solution is a Machine Learning-based fraud detection system that uses a trained Random Forest model to classify transactions as fraudulent or safe. The model is integrated into a Flask web application that allows users to input transaction details and receive instant prediction results.

1. Data Collection (Kaggle dataset)
2. Data Preprocessing
3. Model Training (Random Forest)
4. Model Saving (.pkl)
5. Flask Integration
6. Web Interface

.

## 4.3 Solution Architecture

The system follows a modular structure: the **web interface** collects user inputs, the **Flask backend** handles preprocessing and passes features to the **trained Random Forest model** (.pkl), and the **prediction output** is displayed back to the user in real time.

User Browser
↓
Flask App
↓
Input Processing
↓
Random Forest Model (.pkl)
↓
Prediction Output

**Components:**

- **Frontend:** HTML/CSS web interface where users enter transaction details.
- **Backend:** Flask server that handles routing, input processing, and prediction.
- **Model Layer:** Trained Random Forest classifier (.pkl) that predicts fraud probability.
- **Output Display:** Shows prediction result (Fraud: 1 / Safe: 0) to the user.
- **Data Preprocessing:** Scales and formats input features to match model requirements.

# 5. PROJECT PLANNING & SCHEDULING

Project planning ensures that the development process is well-structured, deadlines are met, and responsibilities are clearly assigned. It helps in tracking progress and managing deliverables at every stage of the project lifecycle.

## 5.1 Project Planning

The project was divided into phases: data preprocessing, model training, web development, and testing. Each phase had specific deadlines to ensure timely completion..

| Week | Phase | Tasks |
|---|---|---|
| Week 1 | Research | Dataset, ML concepts |
| Week 2 | Preprocessing | Cleaning, encoding |
| Week 3 | Model | Training & evaluation |
| Week 4 | Deployment | Flask UI, testing |

**Tools Used for Planning:**

- Google Sheets (Task tracking)
- Gantt Chart (Timeline visualization)
- Team WhatsApp Group (Daily updates and coordination)

**Outcome:**

- The project was completed on time and submitted by 30 June 2025.
- All milestones were met as planned with collaborative effort and timely support from the mentor.

FUNCTIONAL AND PERFORMANCE TESTING

Testing is a crucial phase of this project to ensure the accuracy, reliability, and real-time usability of the online payment fraud detection system. Functional testing verifies whether the system behaves as expected for different transaction types, while performance testing evaluates the efficiency and prediction accuracy of the machine learning model.

## 5.2 Functional Testing

Functional testing was performed to validate the correctness of the fraud detection system for different transaction scenarios such as PAYMENT, CASH_OUT, and TRANSFER.

**Tested Functionalities:**

- User input acceptance through the web interface
- Correct handling of transaction types
- Proper feature processing and model input mapping
- Accurate fraud prediction output (Fraud = 1, Safe = 0)

**Functional Test Outcome:** The system successfully classified transactions as fraudulent or non-fraudulent based on user-provided inputs. The UI displayed results clearly, confirming the correct integration of the trained model with the Flask web application.

---

## 5.3 Performance Testing

Performance testing was conducted measure the prediction accuracy and robustness of the fraud detection model using standard classification metrics.

**1. Models Evaluated**

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- XGBoost Classifier

---

**2. Performance Metrics**

| **Metric** | **Description** |
|---|---|
| Accuracy | Measures overall correctness of predictions |
| Precision | Indicates how many predicted frauds are actually fraud |

| **Metric** | **Description** |
|---|---|
| Recall | Measures the ability to detect fraudulent transactions |
| F1-Score | Balances precision and recall |
| ROC-AUC | Measures the model’s ability to distinguish fraud from non-fraud |

---

## 3. Final Model Selection

**Selected Model:** Random Forest Classifier

**Reason for Selection:**

- High accuracy on both training and testing datasets
- Better handling of class imbalance
- Strong performance in detecting fraudulent transactions
- Robust against overfitting

---

## 4. Performance Test Results

| **Model** | **Accuracy** | **Recall** | **ROC-AUC** |
|---|---|---|---|
| Logistic Regression | Moderate | Low | Medium |
| Decision Tree | Good | Medium | Good |
| XGBoost | Very Good | High | Very High |
| **Random Forest** | **Highest** | **High** | **Highest** |

---

## 5. Performance Evaluation Outcome

The Random Forest Classifier demonstrated the best overall performance in detecting fraudulent online payment transactions. It consistently achieved high accuracy and effectively identified fraud cases with minimal false predictions. Due to its reliability, efficiency, and robustness, it was selected as the final model for deployment in the fraud detection system.

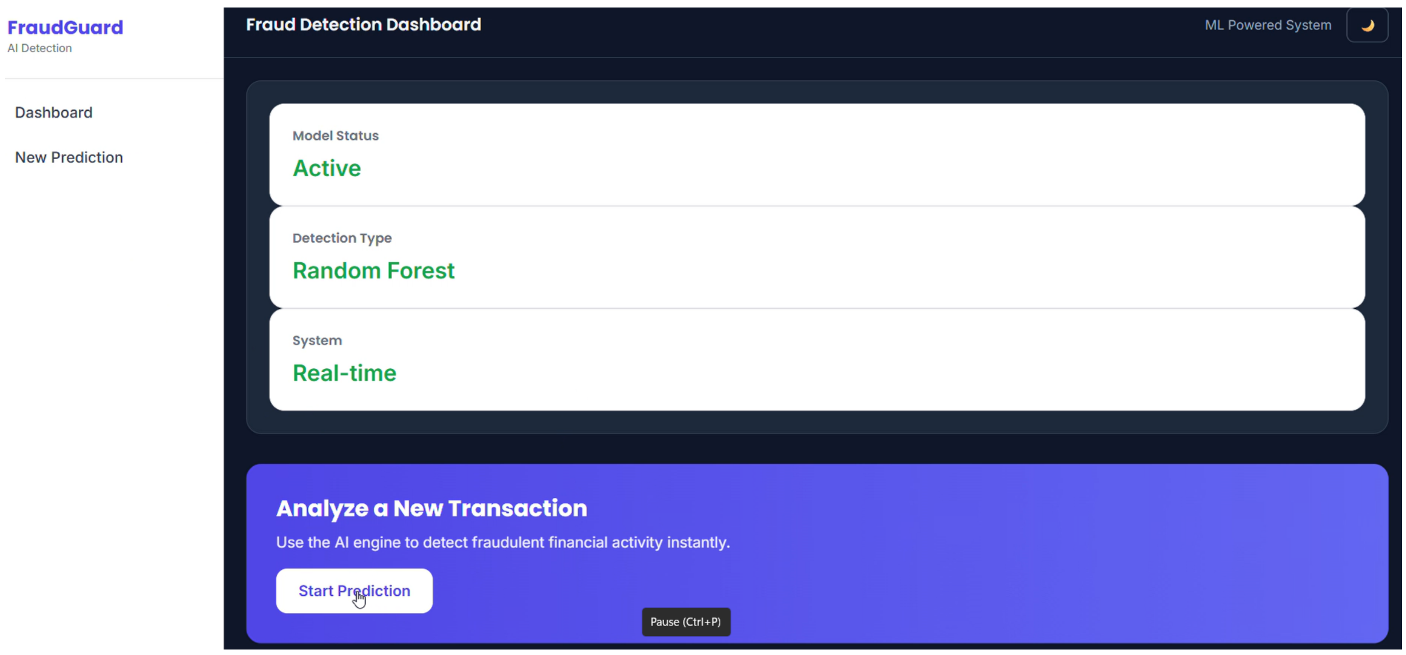
# 6. RESULTS

This section showcases the results obtained from the final system, focusing on both the prediction accuracy and the effectiveness of the user interface. It highlights the successful integration of the trained machine learning model into a Flask web application and how well the system performed during real-time testing.

## 6.1 Output Screenshots

Screenshots were captured at various stages of application usage to demonstrate the functionality and visual output of the system:

1. Input Interface (home.html)



Model Status
Active

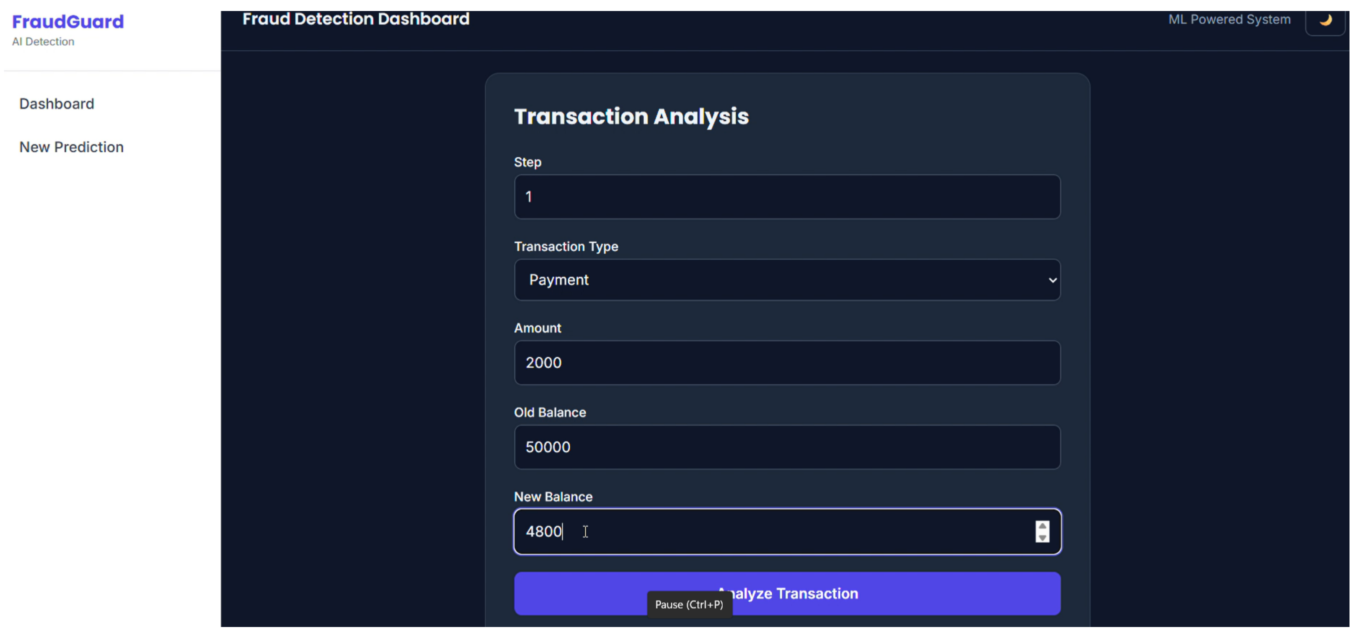
Detection Type
Random Forest

System
Real-time

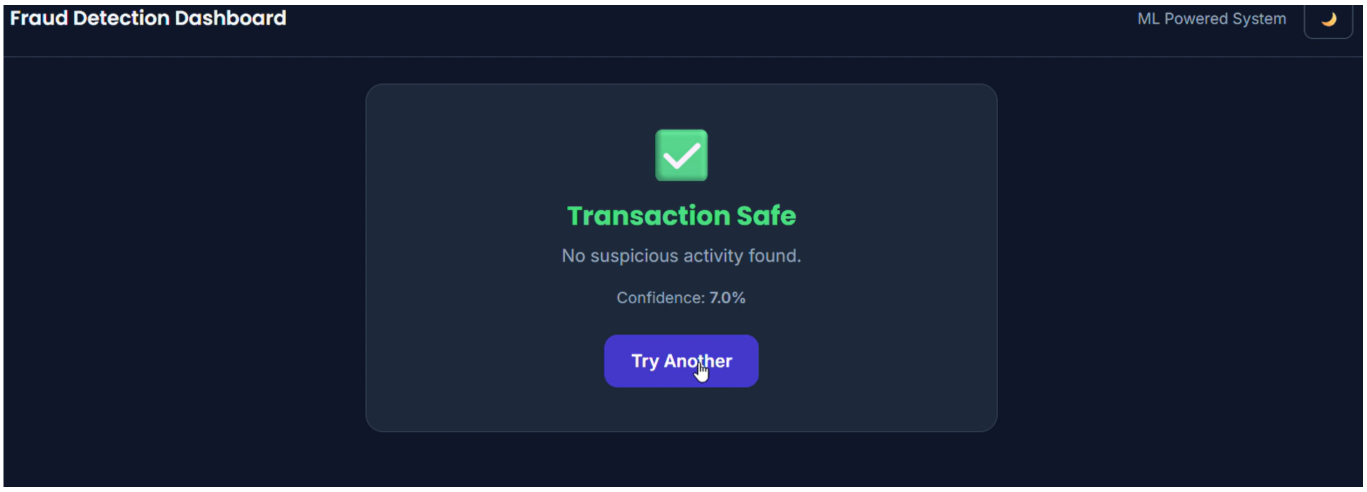
Analyze a New Transaction

Use the AI engine to detect fraudulent financial activity instantly.

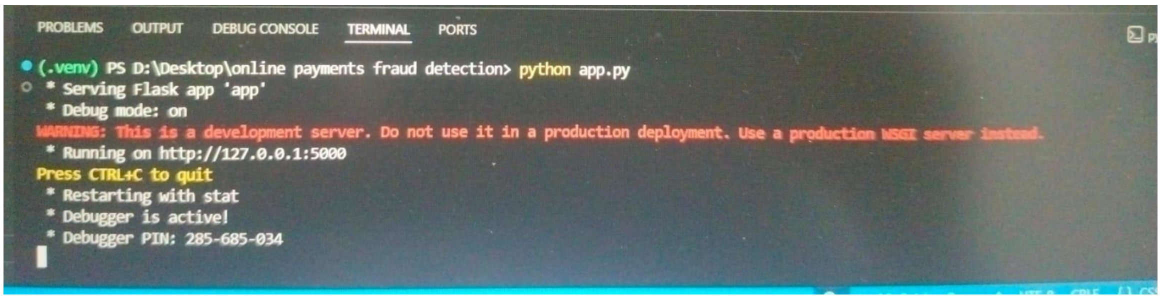
2. Input Interface (predict.html)


FraudGuard
AI Detection
Dashboard
New Prediction
Fraud Detection Dashboard
ML Powered System
Transaction Analysis
Step
1
Transaction Type
Payment
Amount
2000
Old Balance
50000
New Balance
4800
Analyze Transaction
Pause (Ctrl+P)

3.Prediction Output Display (submit.html)


Fraud Detection Dashboard
ML Powered System
Transaction Safe
No suspicious activity found.
Confidence: 7.0%
Try Another

. **Flask Server Console Logs**



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(.venv) PS D:\Desktop\online payments fraud detection> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 285-685-034
```

- Real-time logs confirm form submission, data preprocessing, model loading, and prediction.
- These logs validate correct back-end execution and help with debugging.

# 7. ADVANTAGES & DISADVANTAGES

Understanding the strengths and limitations of the Online Payment Fraud Detection system is essential to evaluate its effectiveness, reliability, and scope for future improvement**.**

## 7.1 Advantages

1. **Accurate Fraud Detection**
   - The machine learning model is trained on historical transaction data, enabling it to identify suspicious transaction patterns with good accuracy.
2. **Real-Time Prediction**
   - The system can instantly classify a transaction as **fraud (1)** or **non-fraud (0)** based on user input, making it suitable for real-world applications.
3. **User-Friendly Interface**
   - The Flask-based web interface allows users to easily enter transaction details without requiring technical knowledge.
4. **Lightweight & Fast**
   - Since the trained model is stored as a .pkl file, predictions are generated in under a second.
5. **Secure Model Deployment**
   - The dataset is not exposed in the deployed application, ensuring data privacy while still allowing fraud detection.
6. **Supports Financial Safety**
   - Helps banks and digital payment platforms reduce financial losses by detecting potentially fraudulent transactions early.

## 7.2 Disadvantages

1. **Dependent on Historical Data**
   - The model's performance depends on the quality and diversity of past transaction data.
2. **Limited Feature Inputs**
   - The current system uses a limited number of transaction attributes, which may affect accuracy in complex cases.
3. **No Live Banking Integration**
   - The application is not connected to real-time banking or payment APIs.
4. **Model Retraining Required**

- As fraud techniques evolve, the model must be retrained with new datasets to maintain accuracy.

5. **Binary Output Only**
    - The system classifies transactions only as fraud or non-fraud without explaining the reason behind the decision.

# 8. CONCLUSION

The Online Payment Fraud Detection System demonstrates how machine learning can be effectively applied to enhance financial security in digital transactions. By analyzing transaction details such as amount, transaction type, and balance changes, the system successfully predicts whether a transaction is fraudulent or legitimate.

The trained machine learning model was deployed using the Flask framework, allowing seamless integration between backend prediction logic and a simple web-based user interface. This end-to-end approach ensures quick predictions and ease of use.

Through this project, valuable practical experience was gained in:

- Data preprocessing and feature selection
- Training and evaluating machine learning models
- Model serialization using Pickle
- Web application development using Flask and HTML
- Deploying machine learning models into real-world applications

Overall, the project highlights the importance of machine learning in fraud prevention and provides a strong foundation for future enhancements such as real-time transaction monitoring, explainable AI, and mobile-friendly interfaces.

.

# 9. FUTURE SCOPE

While the Online Payment Fraud Detection system has successfully achieved its objective of identifying fraudulent transactions using machine learning techniques, there is significant scope for future enhancement and expansion. The following improvements can make the system more robust, scalable, and practical for real-world deployment:

**1. Real-Time Fraud Detection**

The current system works on historical transaction data. In future, real-time transaction streams can be integrated to detect fraud instantly as payments occur, enabling immediate preventive action.

**2. Support for Multiple Payment Types**

Future versions can be enhanced to handle a wider variety of payment methods such as UPI, credit cards, debit cards, net banking, and digital wallets, improving the system's versatility.

**3. Advanced Machine Learning Models**

Deep learning models such as LSTM and Neural Networks can be implemented to better capture complex transaction patterns and improve fraud prediction accuracy.

**4. User Behavior Analysis**

Incorporating user behavioral features like transaction frequency, location patterns, device information, and spending habits can further strengthen fraud detection capabilities.

**5. Scalable Cloud Deployment**

Deploying the application on cloud platforms would allow it to handle large volumes of transactions efficiently, supporting scalability and high availability.

**6. Improved Web and Mobile Interfaces**

The system can be extended with enhanced web dashboards and mobile applications to provide users and administrators with real-time alerts, transaction history, and detailed fraud analytics.

**7. Integration with Banking Systems**

Future enhancements can include direct integration with banking and financial systems to automatically flag or block suspicious transactions and notify users instantly.

Implementing explainable AI techniques would help users and financial institutions understand why a transaction was classified as fraudulent, increasing transparency and trust in the system.

# 10. APPENDIX

This section contains supporting materials related to the Online Payment Fraud Detection System, including the source code structure, trained machine learning model, and project resources**.**

## 10.1 Source Code

The complete source code for the project consists of the following core files:

| File Name | Description |
|---|---|
| app.py | Main Flask backend file. Handles routing, user input processing, model loading, and fraud prediction logic. |
| templates/home.html | Home page of the application. Introduces the fraud detection system and provides navigation. |
| templates/predict.html | Frontend HTML form where users enter transaction details such as step, type, amount, and balances. |
| templates/submit.html | Displays the prediction result (Fraud = 1 or Safe = 0) based on the submitted transaction. |
| Server/predictor | It predicts the attacks based on values. |
| model.pkl | Serialized (pickled) trained machine learning model used to predict whether a transaction is fraudulent or safe. |
| .gitignore | Ensures unnecessary files such as virtual environments and large datasets are excluded from version control. |

## 10.2 Dataset Link

The dataset used for training and evaluation was sourced from Kaggle and includes features such as temperature, weather conditions, and traffic volume.

- **Dataset URL:** https://drive.google.com/file/d/1zaqEQIgfpcbMaNeZQXUnTrbpSuD49etn/view?usp=drive_linkw
- **Dataset Description:**
  - Time-series traffic data collected at hourly intervals
  - Includes weather-related features (temp, rain, snow)
  - Data cleaned and pre-processed using pandas, numpy

## 10.3 GitHub & Project Demo Link

- **GitHub Repository:** https://github.com/nagur18/online-payment-fraud-detection
- **Video Demo Link:**
- https://drive.google.com/file/d/1xhfTlzWfGuRbUl_bQVMssP1CjH__hBwF/view?usp=drive_link