Project Documentation

1. INTRODUCTION

ProjectTitle: Online Payments Fraud Detection using Machine Learning

Team Members:

Mannaru Prem Kumar

Anda Tharun

Chavva Tarun

Gaduputi Hima Varshini

2. PROJECT OVERVIEW

Purpose

The purpose of this project is to detect fraudulent online payment transactions using Machine Learning. It analyzes transaction details such as amount, balance, and transaction type to predict whether a transaction is safe or fraudulent.

Features

- Fraud detection using ML model
- Real-time transaction prediction
- User-friendly web interface
- Secure backend processing
- Accurate classification
- Easy deployment

3. ARCHITECTURE

The Online Payment Fraud Detection System is built using a **three-tier architecture** that includes **Frontend**, **Backend**, and **Machine Learning Model**.

A. Frontend Architecture

The frontend provides a user-friendly interface where users enter transaction details.

Technologies Used

- HTML
- CSS
- JavaScript
- Bootstrap

Functions

- Collect transaction details from users
- Send data to backend via HTTP request
- Display prediction result (Fraud or Safe)

Workflow

1. User opens the website
2. Enters transaction details
3. Clicks submit
4. Data is sent to backend

5. Result is shown

B. Backend Architecture

The backend processes user input and interacts with the ML model.

Technologies Used

- Python
- Flask

Functions

- Receives transaction data from frontend
- Converts data into required ML format
- Sends data to trained model
- Returns prediction result

C. Machine Learning Model

The ML model is trained using historical fraud data.

Technologies Used

- Python
- Scikit-Learn
- Pandas
- NumPy

Functions

- Reads transaction features
- Predicts Fraud or Safe
- Returns classification

Model Files

- model.pkl – Trained ML model
- encoder.pkl – Encodes transaction types

4. SETUP INSTRUCTION

Follow these steps to run the project.

Step 1: Install Software

Install:

- Python 3.8+
- Git
- VS Code

Step 2: Clone the Project

```
git clone https://github.com/Prem-0427/online-fraud-detection.git
cd online-fraud-detection
```

Step 3: Create Virtual Environment

```
python -m venv env
env\Scripts\activate
```

Step 4: Install Required Packages

pip install flask pandas numpy scikit-learn

Step 5: Train Model

python train_model.py

This creates:

- model.pkl
- encoder.pkl

Step 6: Run Application

python app.py

Open in browser:

http://127.0.0.1:5000

5. FOLDER STRUCTURE

```
online_fraud/
│
├── app.py                # Flask backend
├── train_model.py        # ML training script
├── fraud.csv             # Dataset
├── model.pkl             # Trained ML model
├── encoder.pkl           # Label encoder
├── requirements.txt      # Python dependencies
│
├── templates/            # Frontend HTML pages
│   ├── home.html
│   ├── predict.html
│   └── submit.html
│
├── static/               # CSS & JS files
│
└── env/                  # Virtual environment
```

6. RUNNING THE APPLICATION

Step 1: Train the Machine Learning Model

Before running the web application, train the model:

python train_model.py

This will generate:

- model.pkl
- encoder.pkl

These files store the trained fraud detection model and transaction type encoder.

Step 2: Start the Flask Server

Run the backend application:

python app.py

You will see output like:

Running on http://127.0.0.1:5000

7. API DOCUMENTATION

POST /predict

Request:

```
{
  "type": "TRANSFER",
  "amount": 500000,
  "oldbalanceOrg": 500000,
  "newbalanceOrig": 0,
  "oldbalanceDest": 0,
  "newbalanceDest": 500000
}
```

Response:

```
{
  "prediction": "Fraud"
}
```

8. AUTHENTICATION

The **Online Payment Fraud Detection System** is designed primarily as an analytical and prediction-based platform. In the current version, it does not require user login or registration to access the fraud detection functionality. However, authentication concepts are included in the system design for future expansion.

Current Authentication Approach

At present, the system allows open access to:

- Home page
- Fraud prediction page
- Transaction result page

This enables users to quickly test transactions without creating an account, making the system easy to use for demonstrations, research, and learning purposes.

Planned Authentication Design

In future versions, the system can support authentication using:

1. User Accounts

Users will be able to:

- Register using email and password
- Log in securely
- View their transaction history
- Track fraud reports

2. Password Security

Passwords will be:

- Encrypted using hashing algorithms
- Stored securely in the database
- Never saved in plain text

3. Session Management

After login:

- A secure session will be created
- The user will stay logged in until logout
- Sessions will expire automatically after inactivity

4. Role-Based Access (Future Scope)

The system can include:

- **Admin users** to monitor fraud reports
- **Regular users** to submit and verify transactions

Security Measures

The platform is designed to support:

- HTTPS communication
- Input validation to prevent injection attacks
- Secure cookies and session handling

9. USER INTERFACE

The user interface of the Online Payment Fraud Detection System is designed to be simple, attractive, and user-friendly.

Main Pages

1. **Home Page**
    - Displays project title and purpose
    - Includes a "Start Fraud Detection" button
    - Shows information about online payment fraud
2. **Fraud Prediction Page**
    - Input form for transaction details such as:
        - Step
        - Type
        - Amount
        - Old Balance (Sender)
        - New Balance (Sender)
        - Old Balance (Receiver)
        - New Balance (Receiver)
    - A "Check Fraud" button to submit data
3. **Result Page**
    - Displays whether the transaction is **Fraudulent** or **Safe**
    - Uses colors and animations to highlight results

- Shows prediction confidence

The UI is responsive and visually appealing with gradient backgrounds, animations, and clear form layouts.

10. TESTING

Testing ensures that the system is accurate and reliable.

Types of Testing Performed

1. Model Testing

- The trained machine learning model was tested using:
  - Training data
  - Testing data
- Accuracy and prediction results were checked for:
  - Fraud transactions
  - Normal transactions

2. Form Validation Testing

- Input fields were tested for:
  - Empty values
  - Incorrect data types
  - Invalid numbers

3. Backend Testing

- Flask routes were tested using:
  - Manual browser testing
  - Sample transaction inputs

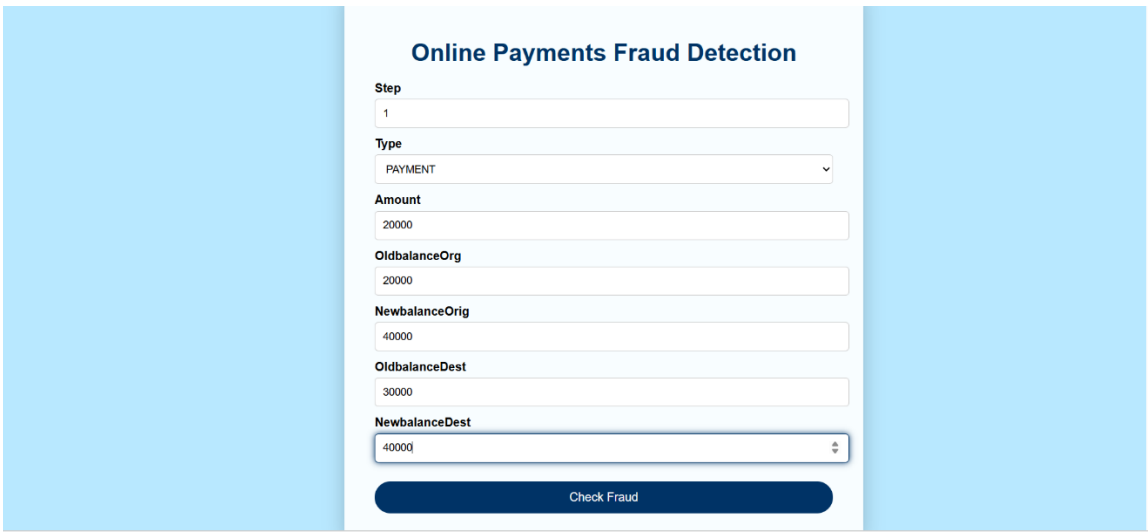
4. UI Testing

- Pages were checked on:
  - Desktop browsers
  - Different screen sizes

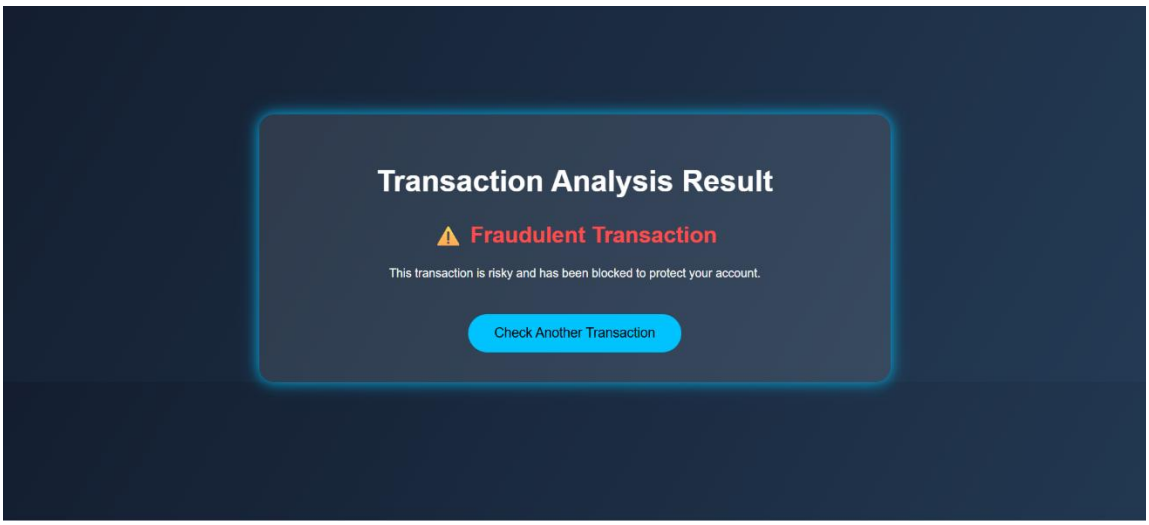
11. SCREENSHOTS OR DEMO

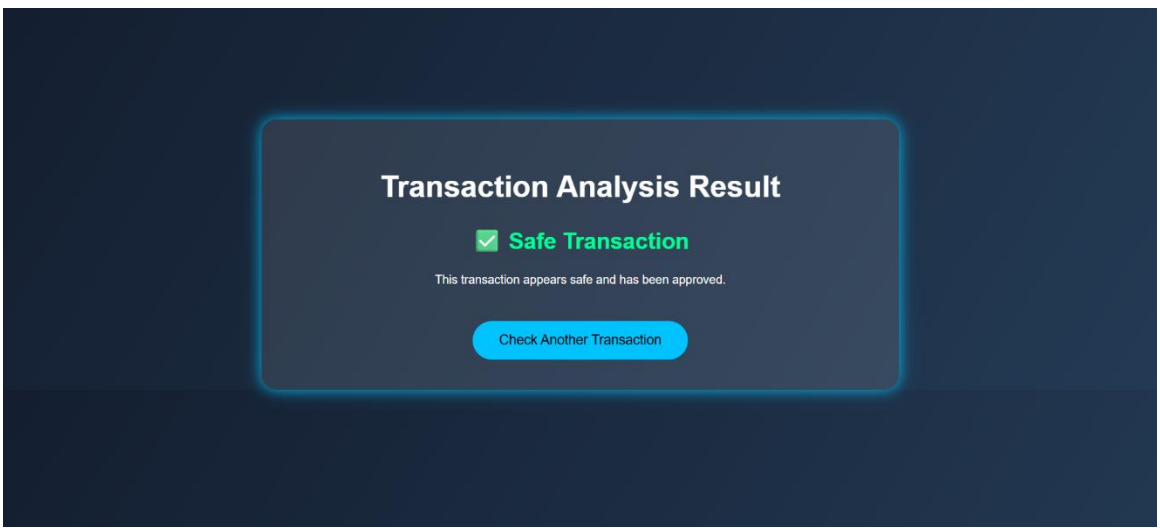
HomePage

Prediction Page



Result Page





12. KNOWN ISSUES

The system has some limitations:

1. **Dataset Dependency**
   - Predictions depend heavily on the training dataset.
   - If the data does not contain similar patterns, results may be inaccurate.
2. **No User Authentication**
   - Currently, there is no login system.
3. **False Positives**
   - Some safe transactions may be detected as fraud.
4. **Large Dataset Handling**
   - The large CSV file (fraud.csv) is difficult to upload and deploy on GitHub without Git LFS.
5. **Model Retraining**
   - The model must be retrained when new data is added.

13. FUTURE ENHANCEMENTS

The system can be improved in the following ways:

1. **User Login System**
   - Add authentication and user accounts.
2. **Real-Time Fraud Detection**
   - Integrate APIs for real-time transaction monitoring.
3. **More Accurate Models**
   - Use deep learning or ensemble models for better accuracy.
4. **Admin Dashboard**
   - Add analytics for fraud trends and reports.
5. **Mobile Application**
   - Develop Android and iOS apps for easy access.
6. **Cloud Deployment**
   - Deploy on AWS, Azure, or Google Cloud for high availability.

.