# Online Fraud Detection Web App Documentation

## 1. Overview

This Flask-based web application utilizes a pre-trained Random Forest Classifier to detect online fraudulent financial transactions. It takes transaction-related input and predicts whether the transaction is legitimate or fraudulent.

## 2. Setup Instructions

### 2.1 Requirements

Ensure Python is installed and then install the required packages using the following command:

pip install flask flask-cors numpy scikit-learn

### 2.2 Project Structure

Organize your files as shown below:

project/  
├── app.py # Main Flask application  
├── fraud\_model1.pkl # Trained ML model (Random Forest)  
├── templates/  
│ └── index.html # Frontend HTML page

### 2.3 Running the Application

Use the command below to start the Flask server:

python app.py

The application will be available at http://127.0.0.1:5000/.

## 3. Model Information

Model Type: Random Forest Classifier

Dataset: Online Payments Fraud Detection Dataset

Input Features:

- amount  
- oldbalanceOrg  
- newbalanceOrig  
- oldbalanceDest  
- newbalanceDest  
- type (categorical; one-hot encoded)

## 4. API Endpoints

### 4.1 GET `/`

Returns the homepage with the HTML form for user input.

### 4.2 POST `/predict`

Receives transaction data and returns prediction result in JSON format.

Request Fields:

- amount (float): Transaction amount  
- oldbalanceOrg (float): Sender's balance before transaction  
- newbalanceOrig (float): Sender's balance after transaction  
- oldbalanceDest (float): Receiver's balance before transaction  
- newbalanceDest (float): Receiver's balance after transaction  
- type (str): One of ['CASH\_IN', 'CASH\_OUT', 'DEBIT', 'TRANSFER', 'PAYMENT']

Example JSON Response:

{  
 "status": "success",  
 "result": "✅ Transaction Seems Legitimate.",  
 "confidence": "95.23%",  
 "processing\_time": "0.02s",  
 "transaction\_type": "TRANSFER",  
 "amount": 10000.0  
}

## 5. Feature Engineering

The 'type' feature is one-hot encoded using the following mapping:

{  
 "CASH\_IN": [0, 0, 0, 0],  
 "CASH\_OUT": [1, 0, 0, 0],  
 "DEBIT": [0, 1, 0, 0],  
 "TRANSFER": [0, 0, 1, 0],  
 "PAYMENT": [0, 0, 0, 1]  
}

## 6. Example Input

amount: 4500.0  
oldbalanceOrg: 10000.0  
newbalanceOrig: 5500.0  
oldbalanceDest: 3000.0  
newbalanceDest: 7500.0  
type: TRANSFER

## 7. Output Interpretation

If the prediction is 0: ✅ Transaction Seems Legitimate.

If the prediction is 1: ❌ Fraudulent Transaction Detected.

The model also returns its confidence in the prediction and the time taken to process the input.

## 8. Security & Deployment Notes

Use environment variables for sensitive configurations.

Deploy using Gunicorn + Nginx or Docker for better performance and scalability.

Sanitize user inputs thoroughly before processing.

## 9. Developer Info

Author: Peeyush Chauhan

Model: fraud\_model1.pkl

App Version: 1.0