PROJECT DESIGN PHASE DOCUMENT

Project Title: Online Fraud Payment Detection System

---

1. Introduction

The design phase of the project focuses on **how the system will work**, including architecture, data flow, and user interaction. This phase ensures that the system's structure is clearly defined before implementation.

The Online Fraud Payment Detection System predicts fraudulent transactions using a trained machine learning model deployed as a web application.

---

2. System Architecture

The system consists of four main modules:

1. **Data Module:**
    - Collects and preprocesses historical transaction data.
    - Handles missing values, normalization, and encoding categorical variables.
2. **Model Module:**
    - Trains a classification model (e.g., Random Forest / Logistic Regression).
    - Saves the trained model using Pickle for deployment.
3. **Web Application Module:**
    - Developed using Flask.
    - Provides user interface to input transaction details.
    - Displays real-time prediction results.
4. **Deployment Module:**
    - Hosted on a cloud platform (Render/Heroku/AWS).
    - Handles HTTP requests to /predict endpoint.

Architecture Diagram (Conceptual)

User Input --> Web Application --> Model Module --> Prediction --> Result Display

^ |

| v

Admin/Reports <---------- Data Module <------- Dataset

---

3. Data Flow Design

Step 1: User Input

- User enters transaction details through web form (e.g., amount, transaction type).

Step 2: Data Validation

- Inputs are checked for correctness and security.

Step 3: Prediction Request

- Validated input is sent to the trained ML model via /predict endpoint.

Step 4: Model Processing

- Model evaluates input and predicts fraud probability.

Step 5: Output Display

- Web interface displays result: "Fraudulent" or "Legitimate" transaction.

Step 6: Admin Reports (Optional)

- Aggregated data and statistics displayed on admin dashboard.

---

4. Module Design

4.1 Data Preprocessing Module

- Remove null values
- Encode categorical variables
- Scale features using StandardScaler
- Handle class imbalance with SMOTE/Resampling

4.2 Model Module

- Select appropriate algorithm (Random Forest, Logistic Regression, etc.)
- Train model with train-test split
- Evaluate model using accuracy, precision, recall, F1-score
- Save trained model as .pkl file

4.3 Web Application Module

- Flask-based interface
- Input form for transaction details
- Call /predict endpoint
- Display prediction results

4.4 Deployment Module

- Host the web application on Render/Heroku/AWS
- Enable HTTPS for secure communication
- Ensure scalability and reliability

---

5. Database Design

If storing historical transactions or user inputs:

Table Name	Columns	Description
transactions	id, amount, type, timestamp, user_id, label	Stores transaction data
users	user_id, name, email	Stores user details
reports	report_id, date, total, fraud_count	Stores aggregated reports

Note: If the system is only for prediction, a full database may not be needed; temporary in-memory storage can be used.

---

6. UML Diagrams

6.1 Use Case Diagram (Conceptual)

[User] --> (Enter Transaction Details)

(User) --> (View Prediction Result)

[Admin] --> (View Reports)

6.2 Sequence Diagram (Simplified)

User -> Web Application: Enter Transaction

Web Application -> Model: Send data for prediction

Model -> Web Application: Return result

Web Application -> User: Display prediction

6.3 Class Diagram (Simplified)

Class: Transaction

- id

- amount

- type

- user_id

- label

+ validate_input()

+ preprocess()

Class: Model

- trained_model

+ predict(input_data)

Class: WebApp

+ get_user_input()

+ display_result()

---

7. Interface Design

7.1 User Interface (UI)

- Clean input form for transaction details
- Submit button for prediction
- Display panel for results (Fraudulent / Legitimate)

7.2 Admin Interface (Optional)

- Reports dashboard
- Transaction statistics
- Graphs for visualization

---

8. Security Design

- Validate all user inputs
- Use HTTPS for communication
- Avoid storing sensitive transaction data in plaintext
- Role-based access (User vs Admin)

---

9. Design Considerations

- Modular design for easier maintenance
- Cloud deployment for accessibility and scalability
- Model retraining support for continuous improvement

---

10. Summary

The design phase outlines the system structure, modules, database, user interface, and security. This document serves as a blueprint for the implementation phase, ensuring a scalable, reliable, and efficient Online Fraud Payment Detection System.