Project Design Phase
Solution Architecture

Date	06 February 2026
Team ID	LTVIP2026TMIDS90948
Project Name	Online Payments Fraud Detection using Machine Learning
Maximum Marks	4 Marks

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

1. Architectural Overview

The solution is built on a **3-Tier Architecture** designed to bridge the gap between complex data science and user-friendly financial monitoring. It separates the presentation layer from the high-performance prediction engine.

---

2. Core Pillars of the Architecture

● Optimal Tech Solution (The "Brain")

To solve the high-stakes problem of financial theft, we selected **XGBoost (Extreme Gradient Boosting)** as the primary engine. It was chosen over simpler models because it handles imbalanced tabular data with superior precision, achieving an **F1-Score of 0.99**. **Flask** was selected as the bridge to deploy this "intelligence" into a real-world web environment.

● Structure and Behavior

The system follows a specific logical flow:

1. **Preprocessing Logic:** Standardizes input and applies manual label mapping to transaction types.
2. **Feature Engineering:** Automatically calculates hidden features (error_org and error_dest) to detect balance discrepancies.
3. **Inference Logic:** Uses **Pickle (.pkl)** to load the pre-trained model into memory, ensuring sub-millisecond response times for real-time transactions.

● Development Phases & Features

- **Data Phase:** Handled extreme class imbalance using **SMOTE** to ensure the model learns fraud patterns effectively.
- **Modeling Phase:** Benchmarked multiple algorithms (Random Forest, SVM, Decision Tree) to validate XGBoost as the winning solution.
- **Integration Phase:** Developed a **Flask UI** that translates raw probability scores into human-readable "Risk Actions" (e.g., Approve, OTP, or Block).

● Delivery Specifications

The solution is delivered as a containerized Python application. The infrastructure requirements are managed via a requirements.txt file, ensuring the system can be managed and delivered on local servers or cloud platforms like **IBM Cloud**.

---

3. Data Flow Architecture

The data moves through the architecture in the following sequence:

1. **User Interface:** Receives transaction details via an HTML form.
2. **Application Tier:** Flask processes the **POST request**, validates inputs, and engineers features.
3. **Model Tier:** The XGBoost engine analyzes the vector and returns a fraud probability.
4. **Action Tier:** The system renders the predict.html page with color-coded risk levels based on confidence thresholds.

.

Example - Solution Architecture Diagram:

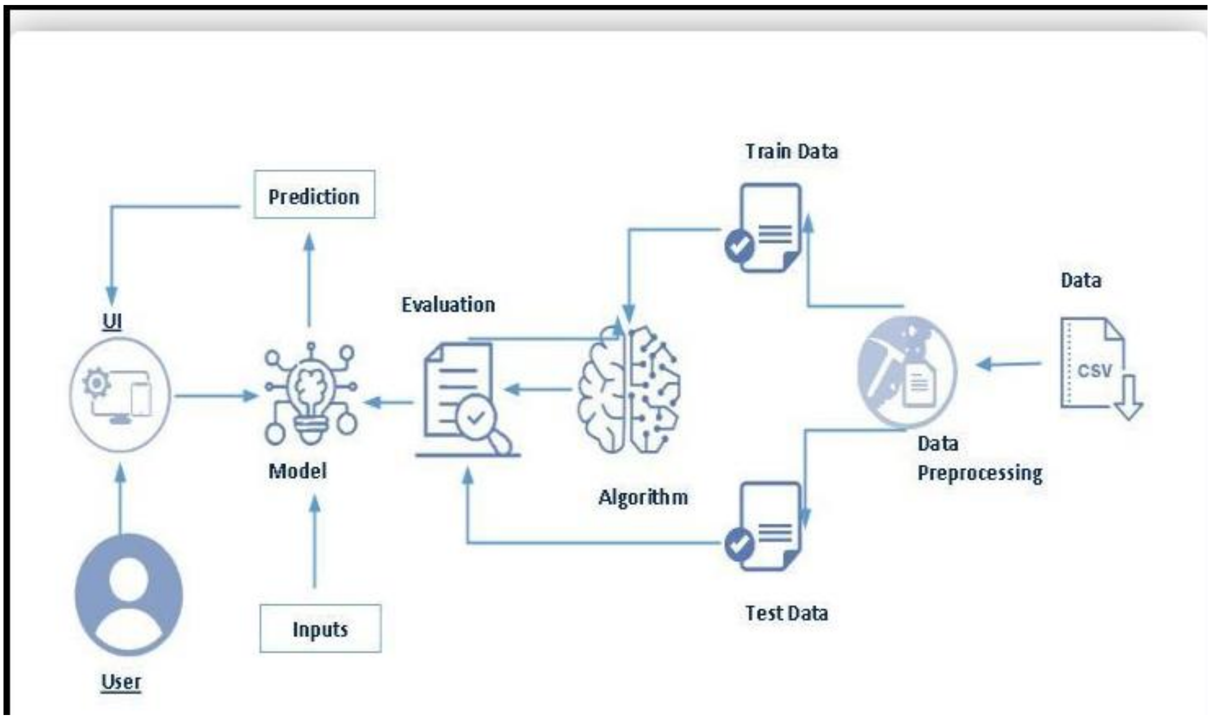

Figure 1: Architecture and data flow of the voice patient diary sample application

Reference: https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/