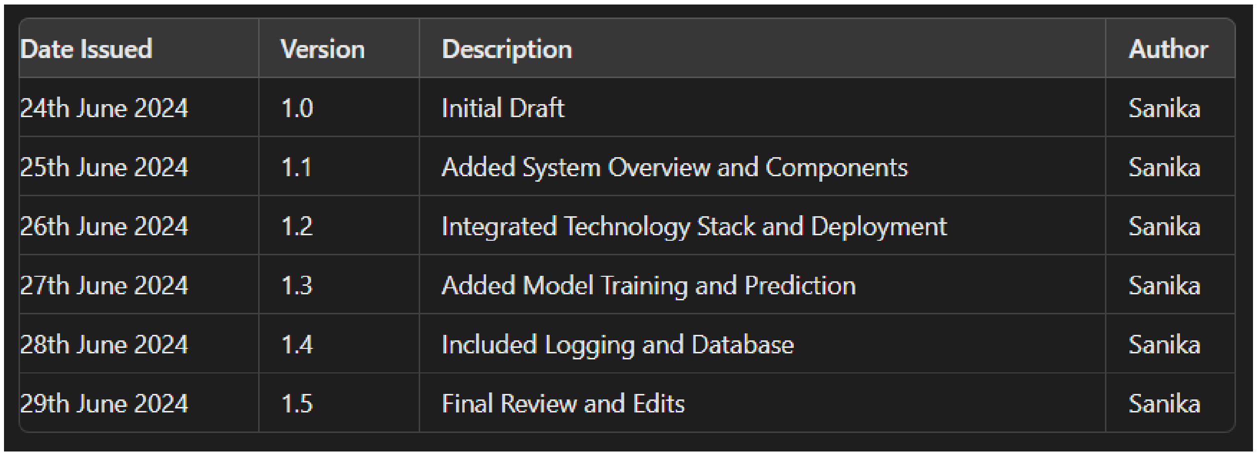
Architecture Document: Fraud Transaction Detection System

Document Version Control



Date Issued	Version	Description	Author
24th June 2024	1.0	Initial Draft	Sanika
25th June 2024	1.1	Added System Overview and Components	Sanika
26th June 2024	1.2	Integrated Technology Stack and Deployment	Sanika
27th June 2024	1.3	Added Model Training and Prediction	Sanika
28th June 2024	1.4	Included Logging and Database	Sanika
29th June 2024	1.5	Final Review and Edits	Sanika

Abstract

The Fraud Transaction Detection System aims to identify fraudulent transactions using machine learning models trained on historical transaction data. This document details the system architecture, components, data flow, and deployment strategy to achieve accurate fraud detection and minimize false positives.

1. Introduction

1.1 Purpose of Architecture Document

The purpose of this document is to provide a comprehensive overview of the Fraud Transaction Detection System's architecture. It outlines the system's functionality, interfaces, constraints, and the technology stack used for its implementation.

1.2 Scope

The system will focus on real-time detection of fraudulent transactions within financial datasets. It will utilize machine learning algorithms to analyze transaction patterns, detect anomalies, and predict fraudulent activities based on historical data.

1.3 Constraints

- Limited to analyzing structured financial transaction data.
- Initial deployment will be on AWS infrastructure.
- Integration with existing banking systems for data retrieval.

1.4 Risks

- Data privacy and security risks.
- Model accuracy and adaptability to new fraud patterns.
- Compliance with regulatory standards (e.g., GDPR, PCI-DSS).

1.5 Out of Scope

- Detailed integration with specific banking APIs.
- Real-time transaction blocking (system will only flag transactions as suspicious).

2. System Overview

The Fraud Transaction Detection System consists of the following main components:

- **Data Ingestion:** Retrieves transaction data from banking databases securely.
- **Preprocessing:** Cleans and prepares data for machine learning model training.
- **Model Training:** Trains fraud detection models using historical transaction data.
- **Real-time Prediction:** Applies trained models to incoming transactions for fraud detection.
- **Alerting:** Notifies stakeholders or systems about flagged fraudulent transactions.
- **Logging:** Records all system activities and model predictions for audit and debugging purposes.

3. Technology Stack

- **Backend:** Python, Flask
- **Database:** PostgreSQL for storing transactional data

- **Machine Learning:** Scikit-learn for model training, XGBoost for boosting models
- **Deployment:** AWS EC2 for hosting backend services, AWS S3 for data storage
- **Monitoring:** AWS CloudWatch for system metrics and logging

4. Model Training and Prediction Workflow

1. **Data Collection:** Fetch historical transaction data from PostgreSQL database.
2. **Data Preprocessing:** Clean data, handle missing values, and scale features.
3. **Feature Engineering:** Extract relevant features such as transaction amount, time of day, etc.
4. **Model Selection:** Train multiple machine learning models (e.g., Logistic Regression, XGBoost).
5. **Model Evaluation:** Evaluate models using metrics like precision, recall, and ROC-AUC score.
6. **Real-time Prediction:** Deploy best-performing model for real-time fraud prediction on incoming transactions.

5. Logging and Database

- **Logging:** Utilizes AWS CloudWatch for logging system activities and model predictions.
- **Database:** PostgreSQL database stores transactional data and model predictions for auditing and retraining purposes.

6. Deployment

- **Infrastructure:** Deployed on AWS EC2 instances for scalability and reliability.
- **Continuous Integration/Continuous Deployment (CI/CD):** Using Jenkins for automated deployment and testing.

Conclusion

This architecture document provides a structured overview of the Fraud Transaction Detection System, detailing its components, workflows, and technology stack. It serves as a guide for stakeholders, developers, and system administrators involved in the design, implementation, and maintenance of the system.