Technical Design Document

# Project Title

Smarter Reconciliation and Anomaly Detection System

# Overview

This technical document outlines a detailed architecture and implementation plan for an advanced anomaly detection and reconciliation system. Leveraging artificial intelligence (AI), specifically machine learning and large language models (LLMs), the solution automates the identification, categorization, and resolution of reconciliation anomalies, improving operational efficiency and reducing manual effort.

# System Architecture

Architecture Diagram:  
[ Data Sources (CSV/Excel) ]  
 ↓  
 [ Data Ingestion ]  
 ↓  
[ Preprocessing and Feature Engineering ]  
 ↓  
[ Anomaly Detection Model (Isolation Forest) ]  
 ↓  
[ LLM Integration (OpenAI GPT-3.5 Turbo) ]  
 ↓  
[ Data Persistence and Caching (Pickle, CSV, Shelve) ]  
 ↓  
[ API Layer (FastAPI Endpoints) ]  
 ↓  
[ Notification System (SMTP Email) ]  
 ↓  
[ Logging, Monitoring, and Exception Handling ]

# High-Level Components

- Data Ingestion  
- Preprocessing and Feature Engineering  
- Anomaly Detection Model  
- Large Language Model (LLM) Integration  
- Data Persistence and Caching  
- API Layer (FastAPI)  
- Notification System (Email)  
- Logging, Monitoring, and Exception Handling

# Detailed Description of Components

1. Data Ingestion  
- Data Sources: CSV/Excel  
- Ingestion Method: pandas chunked loading  
- Error Handling: File validation  
  
2. Preprocessing and Feature Engineering  
- Techniques: Date parsing, temporal features, rolling computations, label encoding  
- Outputs: Structured numeric dataset  
  
3. Anomaly Detection Model  
- Algorithm: Isolation Forest  
- Output: Binary anomaly flags  
  
4. Large Language Model (LLM) Integration  
- Model: OpenAI GPT-3.5 Turbo  
- Functionalities: Categorization, resolution summaries  
- Caching Strategy: Shelve database  
  
5. Data Persistence and Caching  
- Persistent Storage: Pickle, CSV, Shelve  
  
6. API Layer  
- Framework: FastAPI  
- Endpoints: `/detect\_anomaly/`, `/health`  
  
7. Notification System  
- SMTP-based email alerts  
- Attach CSV anomaly reports  
  
8. Logging, Monitoring, and Exception Handling  
- Structured JSON logging  
- Fault tolerance mechanisms

# Configuration Management

YAML (`config.yaml`) manages parameters: file paths, model parameters, API keys, email configurations.

# Security Considerations

Secure handling of sensitive information, secure SMTP, comprehensive logging.

# Deployment and Infrastructure

Docker containerization, cloud deployment (AWS, Azure, GCP), infrastructure automation.

# Exception Handling and Fault Tolerance

Comprehensive exception handling, retry mechanisms, detailed logging.

# Scalability and Performance

Horizontal scalability, batch processing, concurrent processing, caching.

# Future Enhancements

Real-time dashboards, enhanced anomaly detection, model explainability tools.