

Timo Digital Bank Case Study: Project Description

1. Introduction

This document provides a comprehensive overview of the “Timo Digital Bank Case Study” project, a robust simulation of a modern digital banking environment. Developed by ToGiaBaoKDL, this project serves as a complete solution for the case study, showcasing the design and implementation of a regulatory-compliant data platform. The entire system is built in strict adherence to Vietnamese banking regulations (specifically, 2345/QĐ-NHNN 2023), making it a valuable resource for understanding data engineering practices within a regulated financial context.

For direct access to the project repository, please visit:
<https://github.com/ToGiaBaoKDL/TimoBankCaseStudy>

2. Project Overview and Objectives

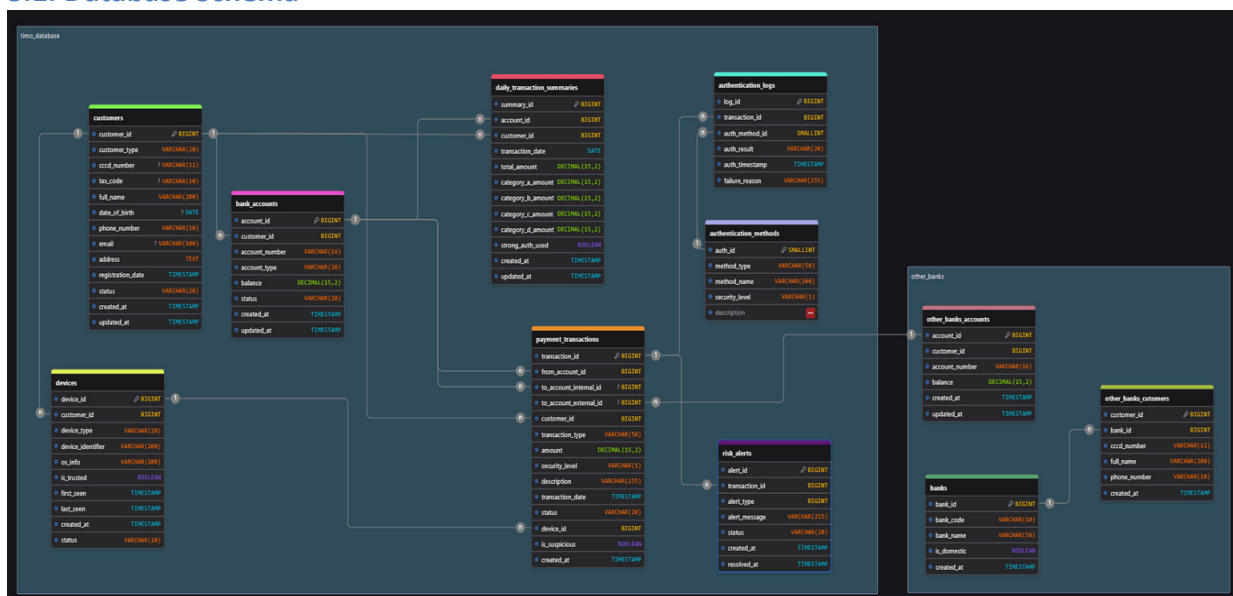
The core objective of the Timo Digital Bank Case Study is to simulate a functional digital banking ecosystem, focusing on data generation, quality, security, and analysis. The project addresses several critical aspects of a modern banking system:

- **Synthetic Data Generation:** Creation of realistic synthetic data for various banking entities, including customers, bank accounts, devices, and a wide range of transactions. This includes complex scenarios such as interbank transfers and e-wallet operations.
- **Data Quality Assurance:** Implementation of mechanisms to define and automatically enforce data quality standards, ensuring the integrity and reliability of the generated data.
- **Risk Monitoring and Fraud Detection:** Development of systems to monitor potential risks, detect fraudulent activities, and trigger security alerts in real-time. This is crucial for maintaining the security and trustworthiness of the banking platform.
- **Data Visualization and Analytics:** Provision of analytical dashboards and data visualization tools to offer insights into banking operations, customer behavior, and transaction patterns.
- **Automated Data Pipeline:** Orchestration and scheduling of data processes using Dagster, ensuring efficient and automated data flow from generation to analysis.

3. Technical Architecture and Components

The project’s architecture is designed for scalability, reliability, and compliance. It leverages a modern data stack, primarily built around PostgreSQL for data storage, Dagster for workflow orchestration, and Streamlit for interactive data visualization.

3.1. Database Schema



Database Schema

The system utilizes a PostgreSQL database with a meticulously designed schema to represent various banking entities and their relationships. Key tables include:

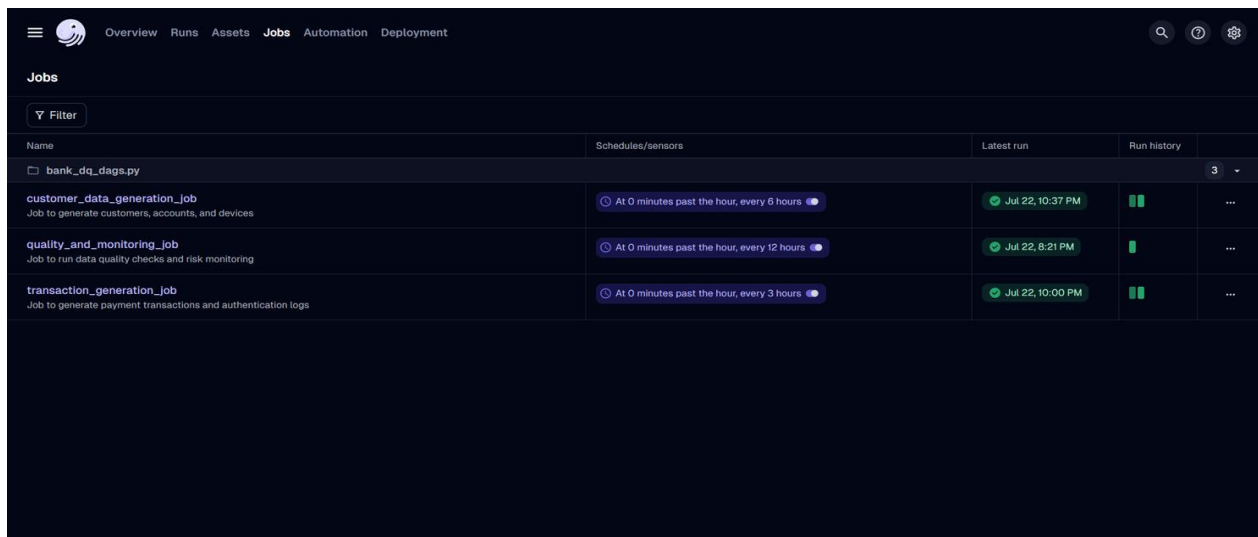
- **customers**: Stores comprehensive customer information, distinguishing between individuals and organizations, and including national IDs, tax codes, and addresses.
- **bank_accounts**: Manages customer bank accounts, detailing account types (savings, checking, e-wallet), balances, and status.
- **devices**: Records information about customer devices, including type, operating system, and trust status, crucial for device-based security.
- **authentication_methods**: Defines various authentication methods (e.g., OTP, biometric, digital signature) and their associated security levels.
- **payment_transactions**: Logs all payment transactions, transfers, and e-wallet operations, forming the basis for financial analysis and fraud detection.
- **risk_alerts**: Stores automatically generated alerts for anomalies such as high-value transactions, untrusted devices, or weak authentication attempts.

Crucially, the database schema incorporates CHECK Constraints for data validation and Triggers & Functions to automate processes like transaction classification (in accordance with SBV regulations) and daily transaction summary updates, which also trigger risk alerts.

3.2. Data Pipeline Components

The data pipeline is orchestrated using Dagster, ensuring a robust and automated flow of data through various stages:

- **Data Generation Scripts:** Python scripts (`src/generate_data_timo.py`, `src/generate_data_other_banks.py`) are responsible for generating diverse synthetic data that adheres to schema constraints and covers various banking scenarios.
- **Data Quality & Risk Monitoring:** Dedicated Python modules (`src/data_quality_standards.py`, `src/monitoring_audit.py`) perform automated data quality checks (e.g., nulls, duplicates, format validation) and monitor for financial risks, including high-value transactions without strong authentication or daily limit breaches. Detailed logs are maintained for audit purposes.
- **Dagster Orchestration:** The `dags_or_jobs/bank_dq_dags.py` file defines Dagster jobs for generating data, running quality checks, and executing risk monitoring. These jobs can be run manually or scheduled automatically, providing flexibility in data processing.



Name	Schedules/sensors	Latest run	Run history
bank_dq_dags.py			
customer_data_generation_job Job to generate customers, accounts, and devices	At 0 minutes past the hour, every 6 hours	Jul 22, 10:37 PM	3
quality_and_monitoring_job Job to run data quality checks and risk monitoring	At 0 minutes past the hour, every 12 hours	Jul 22, 8:21 PM	1
transaction_generation_job Job to generate payment transactions and authentication logs	At 0 minutes past the hour, every 3 hours	Jul 22, 10:00 PM	2

Dagster UI

- **Analytical Dashboard:** A Streamlit-based dashboard (`visualization/dashboard.py`) provides a user-friendly interface for visualizing key banking metrics. It displays overviews of customers, transactions, devices, and risk alerts, along with time-series charts, customer segmentation, and filterable data tables.

4. Setup and Usage

To set up and run the Timo Digital Bank Case Study, the following steps are generally required:

1. **Environment Setup:** Ensure Python 3.10+ and PostgreSQL are installed. Project dependencies are managed via `requirements.txt`.
2. **Database Initialization:** Create the PostgreSQL database and load the schema from `sql/schema.sql`.
3. **Data Generation and Checks:** Manually run Python scripts for data generation, quality checks, and risk monitoring, or leverage Dagster for automated execution.
4. **Dagster UI:** Start the Dagster UI to manage and monitor data pipeline jobs.

5. **Streamlit Dashboard:** Launch the Streamlit dashboard to access real-time analytical insights.

5. Conclusion

The Timo Digital Bank Case Study project is a well-structured and comprehensive demonstration of data engineering principles applied to the financial sector. Its focus on regulatory compliance, robust data quality, and automated risk monitoring makes it an excellent example of building secure and efficient data platforms for digital banking. The project's modular design and clear documentation facilitate understanding and potential extension, offering a valuable learning resource for aspiring data engineers and financial technology enthusiasts.