# Banking Case Study: Real-Time Transaction Processing

Submitted by: [Adarsh]

Date: [28-11-2024]

# Table of Contents

1. Introduction

2. System Architecture

3. Modules

3.1 User Module

3.2 Transaction Processing Module

3.3 Fraud Detection Module

4. Code Samples

5. Testing

6. Monitoring Integration

7. Dockerization

8. GitHub/GitLab Link

# 1. Introduction

This document provides a comprehensive solution for the Banking Case Study on Real-Time Transaction Processing. The project aims to modernize legacy systems by implementing a scalable, secure, and efficient system using Spring Boot microservices.

# 2. System Architecture

The architecture includes the following components:

- API Gateway

- Spring Boot Microservices

- Prometheus and Grafana for monitoring

- Dockerized infrastructure

The architecture diagram is provided below.

[Insert Architecture Diagram Here]

# 3. Modules

## 3.1 User Module

The User Module handles user registration, authentication (using JWT or OAuth2), and user management.

## 3.2 Transaction Processing Module

This module processes real-time transactions, ensuring data integrity and compliance with regulatory requirements.

## 3.3 Fraud Detection Module

The Fraud Detection Module analyzes transactions in real-time to identify and prevent fraudulent activities.

# 4. Code Samples

### Spring Boot Application Configuration

@SpringBootApplication  
public class BankingApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(BankingApplication.class, args);  
 }  
}

### REST API Endpoint Example

@RestController  
@RequestMapping("/api/transactions")  
public class TransactionController {  
 @PostMapping("/process")  
 public ResponseEntity<String> processTransaction(@RequestBody Transaction transaction) {  
 // Logic for processing transaction  
 return ResponseEntity.ok("Transaction Processed");  
 }  
}

### Dockerfile

FROM openjdk:17-jdk-slim  
COPY target/banking-app.jar app.jar  
ENTRYPOINT ["java", "-jar", "app.jar"]

# 5. Testing

Postman and ThunderClient were used to test the endpoints. Screenshots of the test results are included below.

# 6. Monitoring Integration

Prometheus and Grafana are used to monitor the application's metrics and performance.

# 7. Dockerization

The application is containerized using Docker. The Docker Compose file ensures easy deployment of all services.

# 8. GitHub/GitLab Link

[DataVisionary190824/Banking-Real-time-System](https://github.com/DataVisionary190824/Banking-Real-time-System)