Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core implementation details for **FinSight - Real-Time Financial Analytics Engine**. FinSight is a backend system for processing, analyzing, and visualizing large-scale financial transactions, leveraging Java Streams, multithreading, Spring Boot, PostgreSQL, and Dockerized microservices. The system emphasizes high-performance analytics, robust API security (JWT), and advanced error handling.

---

1. System Components

Component	Description / Responsibility
API Gateway	Routes requests, handles authentication (JWT), rate limiting
Analytics Service	Processes & analyzes transactions in real-time
Data Ingestion Service	Receives, validates, and queues incoming transactions
Visualization Service	Serves analytics data for visualization
PostgreSQL Database	Stores transactions, analytics results, user data
Auth Service	Manages user authentication and JWT issuance

---

2. Class/Interface Overview

Class/Interface	Responsibility / Key Methods	Relationships
`TransactionController`	REST endpoints for transaction ops	Uses `AnalyticsService`
`AnalyticsService`	Core analytics logic, stream processing	Uses `TransactionRepository`
`TransactionRepository`	DB access for transactions	Extends `JpaRepository`
`JwtAuthFilter`	JWT validation for API requests	Used by API Gateway
`ErrorHandler`	Centralized error handling	Global exception handler
`Transaction` (Entity)	Transaction data model	Mapped to DB
`User` (Entity)	User data model	Mapped to DB

Key Methods (Sample):

```
class AnalyticsService {
    List<AnalyticsResult> analyzeTransactions(Stream<Transaction> txStream);
    void processTransaction(Transaction tx);
}
```

```
class TransactionController {
    ResponseEntity<?> postTransaction(TransactionDTO dto);
    ResponseEntity<List<AnalyticsResult>> getAnalytics();
}
```

---

3. Data Structure Overview

Entity	Fields (Type)
Transaction	id (UUID), userId (UUID), amount (BigDecimal), timestamp (Instant), type (String), status (String)
User	id (UUID), username (String), passwordHash (String), roles (Set)
AnalyticsResult	metric (String), value (BigDecimal), period (Instant)

---

4. Algorithms / Logic

Transaction Processing & Analytics (Pseudocode):

```
// Ingest transaction
if (validate(tx)) {
    saveToDB(tx);
    analyticsService.processTransaction(tx);
}

// Real-time analytics (using Java Streams & multithreading)
Stream<Transaction> txStream = getRecentTransactions();
List<AnalyticsResult> results = txStream
    .parallel()
    .filter(tx -> tx.status == "COMPLETED")
    .collect(groupingBy(...), summarizing...);
```

JWT Authentication Flow:

1. Extract JWT from request header
2. Validate signature & expiry
3. Set user context or reject request

---

5. Error Handling

Scenario	Handling Approach
Invalid JWT / Unauthorized	Return 401 Unauthorized, log attempt
Invalid Transaction Data	Return 400 Bad Request, validation error message
DB Connection Failure	Return 503 Service Unavailable, retry logic

Analytics Processing Exception	Log error, return 500 Internal Server Error
Resource Not Found	Return 404 Not Found

End of Document