Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **Enerlytics - Energy Consumption Analytics Service**. The service is a Spring Boot microservice for collecting, storing, and analyzing energy consumption data from smart meters, supporting asynchronous ingestion, secure analytics endpoints, and containerized deployment.

---

1. System Components

Component	Description	Key Responsibilities
API Layer	REST controllers	Expose ingestion & analytics endpoints
Ingestion Service	Async data processing	Validate, queue, and persist data
Analytics Service	Data analysis	Compute consumption stats, trends
Persistence Layer	PostgreSQL DB + JPA	Store/retrieve meter data
Security Module	JWT-based auth	Secure endpoints
Docker Container	Containerization	Deployment packaging

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`EnergyDataController`	REST API endpoints	`ingestData()`, `getAnalytics()`
`IngestionService`	Async data handling	`processDataAsync()`
`AnalyticsService`	Analytics logic	`calculateStats()`, `getTrends()`
`EnergyDataRepository`	JPA repository	`save()`, `findByMeterIdAndPeriod()`
`JwtAuthFilter`	JWT validation	`doFilterInternal()`
`EnergyData`	Entity/model	`id`, `meterId`, `timestamp`, `value`

Relationships:

- `EnergyDataController` uses `IngestionService` and `AnalyticsService`
- `IngestionService` uses `EnergyDataRepository`
- `AnalyticsService` uses `EnergyDataRepository`

---

3. Data Structure Overview

Entity/Model	Fields

`EnergyData`	`id: UUID`, `meterId: String`, `timestamp: Instant`, `value: Double`
`AnalyticsResult`	`meterId: String`, `period: String`, `total: Double`, `average: Double`

PostgreSQL Table:

```
CREATE TABLE energy_data (
  id UUID PRIMARY KEY,
  meter_id VARCHAR(64),
  timestamp TIMESTAMP,
  value DOUBLE PRECISION
);
```

---

4. Algorithms/Logic

Asynchronous Ingestion Flow:

```
@PostMapping("/ingest")
public ResponseEntity<?> ingestData(@RequestBody EnergyDataDto dto) {
    ingestionService.processDataAsync(dto);
    return ResponseEntity.accepted().build();
}
```

Analytics Calculation (Pseudocode):

```
For given meterId and period:
  Fetch all EnergyData records
  Compute total = sum(values)
  Compute average = total / count
  Return AnalyticsResult
```

---

5. Error Handling

Scenario	Handling Approach
Invalid JWT / Unauthorized	Return 401 Unauthorized
Invalid Input Data	Return 400 Bad Request
DB Connection Failure	Log error, return 503 Service Unavailable
Data Not Found	Return 404 Not Found
Async Processing Failure	Log error, return 202 Accepted

End of Document