**Author: Alagu  
  
Banking Microservices Project Documentation**

**1. Overview**

This project is a **Banking Application** built using **Spring Boot Microservices** with the following components:

* **Identity Service (Auth Service)** → manages user accounts & authentication (JWT-based).
* **Transaction Service** → handles deposits, credits, debits, and balance management for users.
* **Statement Service** → retrieves transaction history by calling the Transaction Service.
* **API Gateway** → single entry point to all services; validates JWTs and routes requests.
* **Service Registry (Eureka Server)** → for service discovery and dynamic load balancing.
* **Config Server** → centralizes configuration for all services.

All services communicate via **Spring Cloud**, and JWT is used for authentication and authorization.

**2. Architecture Diagram**

+------------------+

| Config Server |

+------------------+

|

v

+------------------+ +------------------+

| Identity Service | | Transaction Svc |

+------------------+ +------------------+

| |

v v

Generates JWT Handles Txns

| |

+---------+---------+

|

+------------------+

| Statement Svc |

+------------------+

|

v

+-------------------+

| API Gateway |

| Auth Filter + JWT |

+-------------------+

|

v

+-------------------+

| Eureka Registry |

+-------------------+

**3. Services**

**3.1 Identity Service**

* **Purpose**: Manages user registration, login, and JWT token generation.
* **Entity**: UserCredential (id, name, email, password).
* **Key Endpoints**:
  + POST /auth/register → register a new user.
  + POST /auth/login → authenticate user and return JWT.
  + GET /auth/validate?token=xxx → validate JWT (optional, if Gateway delegates validation).
* **JWT Secret**: Must match with API Gateway’s JwtUtil.SECRET.

**3.2 Transaction Service**

* **Purpose**: Maintains account balance and transaction history for users.
* **Entity**: Transaction (id, userId, amount, balance, type, timestamp).
* **Transaction Types**: CREDIT, DEBIT, DEPOSIT.
* **Key Endpoints (exposed via API Gateway)**:
  + POST /transactions/deposit/{userId}?amount=1000
  + POST /transactions/credit/{userId}?amount=500
  + POST /transactions/debit/{userId}?amount=200
  + GET /transactions/balance/{userId}
  + GET /transactions/history/{userId}
* **Balance Calculation**: Latest balance is taken from the last transaction of the user.

**3.3 Statement Service**

* **Purpose**: Provides transaction history for users by calling Transaction Service.
* **Uses**: WebClient (from WebFlux).
* **Key Endpoint**:
  + GET /statements/{userId} → returns user’s transaction history in **descending order**.

Example Response:

[

{

"id": 10,

"userId": 3,

"amount": 500.0,

"balance": 1500.0,

"type": "CREDIT",

"timestamp": "2025-09-18T10:22:15.123"

},

{

"id": 9,

"userId": 3,

"amount": 200.0,

"balance": 1000.0,

"type": "DEBIT",

"timestamp": "2025-09-18T09:45:02.567"

}

]

**3.4 API Gateway**

* **Purpose**: Entry point for all clients. Handles routing and JWT validation.
* **Port**: 9191
* **Key Config** (application.yml):
* spring:
* application:
* name: API-GATEWAY
* cloud:
* gateway:
* routes:
* - id: IDENTITY-SERVICE
* uri: lb://IDENTITY-SERVICE
* predicates:
* - Path=/auth/\*\*
* - id: TRANSACTION-SERVICE
* uri: lb://TRANSACTION-SERVICE
* predicates:
* - Path=/transactions/\*\*
* filters:
* - AuthenticationFilter
* - id: STATEMENT-SERVICE
* uri: lb://STATEMENT-SERVICE
* predicates:
* - Path=/statements/\*\*
* filters:
* - AuthenticationFilter
* **AuthenticationFilter**:
  + Extracts JWT from Authorization: Bearer <token>.
  + Validates token using JwtUtil.
  + Rejects unauthorized requests with 500 Unauthorized Access.

**3.5 Eureka Server**

* **Purpose**: Service Registry for service discovery.
* **Port**: 8761
* **Dashboard**: http://localhost:8761
* All microservices register with it automatically.

**3.6 Config Server**

* **Purpose**: Centralized config for all microservices.
* **Setup**:
  + Configurations stored in Git or local repo.
  + Example bootstrap.yml in each service:
  + spring:
  + application:
  + name: transaction-service
  + cloud:
  + config:
  + uri: http://localhost:8888

**4. End-to-End Flow**

1. **Register/Login User** → Identity Service generates JWT.
2. **User calls Transaction API** (e.g., /transactions/deposit/3?amount=1000) via API Gateway.
   * API Gateway intercepts request.
   * AuthenticationFilter validates JWT.
   * If valid → request forwarded to Transaction Service.
3. **Transaction Service processes the transaction** and updates balance.
4. **Statement Service** can query all transactions for a user by calling Transaction Service with WebClient.

**5. Tech Stack**

* **Spring Boot 3.2.9**
* **Spring Cloud 2023.0.3 (Leyton)**
* **Spring Security + JWT**
* **Spring Cloud Config, Eureka, Gateway**
* **JPA + PostgreSQL/MySQL**
* **WebFlux (WebClient)**
* **Lombok**
* **Maven**

**6. Running the Project**

1. **Start Config Server**
2. mvn spring-boot:run -pl config-server
3. **Start Eureka Server**
4. mvn spring-boot:run -pl service-registry
5. **Start Identity Service**
6. mvn spring-boot:run -pl identity-service
7. **Start Transaction Service**
8. mvn spring-boot:run -pl transaction-service
9. **Start Statement Service**
10. mvn spring-boot:run -pl statement-service
11. **Start API Gateway**
12. mvn spring-boot:run -pl api-gateway

**7. Testing with cURL**

* **Login to get JWT**:
* curl -X POST http://localhost:9191/auth/login -d '{"email":"test@mail.com","password":"123"}' \
* -H "Content-Type: application/json"
* **Deposit money**:
* curl -X POST "http://localhost:9191/transactions/deposit/3?amount=1000" \
* -H "Authorization: Bearer <jwt-token>"
* **Check balance**:
* curl "http://localhost:9191/transactions/balance/3" \
* -H "Authorization: Bearer <jwt-token>"
* **Get statement**:
* curl "http://localhost:9191/statements/3" \
* -H "Authorization: Bearer <jwt-token>"

**8. Known Issues & Fixes**

* **500 Unauthorized Access** → Ensure JWT is sent in header and secrets match between Identity Service & Gateway.
* **MacOS Netty warning** → Add netty-resolver-dns-native-macos dependency.
* **Cloud version mismatch** → Use Spring Boot 3.2.x with Spring Cloud 2023.0.x.