**Problem Statement — CRBT Management System with Secure Microservices**

**1️⃣ Business Context**

A telecom operator wants to provide **Caller Ring Back Tone (CRBT)** services to its customers.  
When someone calls a subscriber, instead of the standard ringing sound, a custom tone plays.  
The system should:

* Allow users to **browse, purchase, and activate CRBTs**
* Allow **admin** to manage CRBT content
* Secure all APIs using **JWT-based authentication**
* Follow a **microservices architecture** for scalability.

**2️⃣ Microservices to Build**

We will have **four main services**:

**1. Authentication Service**

* Manages user registration and login.
* Issues **JWT tokens** after successful login.
* Stores users and roles in MySQL using **Hibernate/JPA**.

Endpoints:

POST /auth/register → Register new user

POST /auth/login → Authenticate & return JWT

**2. CRBT Catalog Service**

* Manages CRBT songs (tone ID, title, artist, price, category).
* Admin can **add/update/delete** CRBTs.
* Users can **view** available CRBTs.

Endpoints:

GET /crbt/all → List all CRBTs

POST /crbt/add → Admin only

PUT /crbt/update/{id} → Admin only

DELETE /crbt/delete/{id} → Admin only

**3. CRBT Subscription Service**

* Allows users to **purchase and activate CRBTs**.
* Maintains mapping between **users** and their **active CRBTs**.
* Ensures only **authenticated** users can subscribe.
* Communicates with **Catalog Service** to verify CRBT exists.

Endpoints:

POST /subscription/activate/{toneId} → Purchase & activate tone

GET /subscription/my-tones → View user’s active CRBTs

DELETE /subscription/deactivate/{toneId} → Remove active CRBT

**4. API Gateway**

* Routes requests to respective services.
* Validates **JWT token** before forwarding.
* Handles **CORS, rate limiting, and logging**.

Example Routes:

/auth/\*\* → Authentication Service

/catalog/\*\* → CRBT Catalog Service

/subscription/\*\* → CRBT Subscription Service

**3️⃣ Tech Stack**

* **Java 17**
* **Spring Boot 3.x**
* **Spring Cloud Gateway**
* **Spring Security with JWT**
* **Hibernate + Spring Data JPA**
* **MySQL** (per service DB schema)
* **Maven**

**4️⃣ JWT Integration**

* **Authentication Service** issues JWT tokens.
* API Gateway uses a **JWT filter** to:
  + Extract Authorization: Bearer <token> header.
  + Validate token signature & expiration.
  + Add user info to request headers for downstream services.
* Downstream services **trust Gateway** but can also validate tokens for extra security.

**5️⃣ Database Design (per service)**

**Auth Service (auth\_db)**

| **user\_id** | **username** | **password\_hash** | **role** |
| --- | --- | --- | --- |
| 1 | avinash | (hash) | USER |
| 2 | admin | (hash) | ADMIN |

**CRBT Catalog Service (catalog\_db)**

| **tone\_id** | **title** | **artist** | **price** | **category** |
| --- | --- | --- | --- | --- |
| 101 | Love Tune | Arijit Singh | 20.00 | Romantic |
| 102 | Party Beat | Badshah | 15.00 | Pop |

**Subscription Service (subscription\_db)**

| **sub\_id** | **user\_id** | **tone\_id** | **activated\_on** |
| --- | --- | --- | --- |
| 501 | 1 | 101 | 2025-08-08 |

**6️⃣ Example Flow**

**Scenario:**

1. User registers → gets credentials.
2. User logs in → receives JWT.
3. User requests available CRBT list (Catalog Service via API Gateway).
4. User purchases CRBT (POST /subscription/activate/101).
5. CRBT Subscription Service records mapping in DB.

**7️⃣ Security Rules**

* **Auth Service** → open for registration & login.
* **Catalog Service**:
  + /all → public
  + /add, /update, /delete → only **ADMIN role**
* **Subscription Service** → only **authenticated users**.
* All requests pass through **API Gateway** with JWT validation.

**8️⃣ Expected Output Example**

**Register User:**

POST /auth/register

{

"username": "avinash",

"password": "12345"

}

**Login:**

POST /auth/login

{

"username": "avinash",

"password": "12345"

}

**Response:**

{

"token": "eyJhbGciOiJIUzI1NiIsInR..."

}

**Access Protected Endpoint:**

GET /subscription/my-tones

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR...