### **Exercise: Notification Service**

*In modern applications,* ***notifications*** *are essential for keeping users informed and engaged. Whether it's a* ***confirmation email, a password reset SMS, or a push notification about an important update****, every platform relies on a* ***robust notification system****. The* ***notification-svc*** *is designed to* ***centralize and manage any type of notifications*** *efficiently, ensuring users receive timely and relevant updates while maintaining flexibility and scalability.*

### **Description**

The **notification-svc** is a **flexible notification REST service** that supports several functionalities. This microservice allows **managing user notification preferences, sending notifications, retrieving history, resending failed messages, and clear notification history**.

## **API Contracts & Functionality Descriptions**

### **1. Manage Notification Preferences (Upsert Operation)**

**Description:** Users can enable/disable notifications. This follows the **"Upsert" pattern.**

💡 **What is Upsert?**

* **Upsert = Update + Insert**
* **If a record exists**, it gets **updated**.
* **If it doesn’t exist**, a new record is **inserted**.
* This **avoids duplicates** and ensures data consistency.

**Endpoint:**

**POSTMAN: http://localhost:8081/api/v1/notifications/preferences**

**POST /api/v1/notifications/preferences**

**Request Body:**

{

"userId": "550e8400-e29b-41d4-a716-446655440000",

"isNotificationEnabled": true,

"type": "EMAIL",

"contactInfo": "user@example.com"

}

### 

### **2. Send a Notification**

Send a notification of a certain type. Email notifications are enough for the exercise, but still the functionality can be expanded in the future.

**Endpoint:**

**POST /api/v1/notifications**

**Request Body:**

{

"userId": "550e8400-e29b-41d4-a716-446655440000",

"type": "EMAIL",

"subject": "Monthly Payment",

"body": "Our platform charged you 20.00$ for your monthly subscription."

}

### **3. Get Notification History**

Fetch past notifications for a user, optionally filtered by type. Keep in mind a user may clear some of their notifications, so we must assure to return only non-deleted notifications. **But still keep that data in the database!**

**Endpoint:**

**GET /api/v1/notifications?user\_id=...&type=EMAIL**

**Response:**

[

{

"id": "9fd12196-74d9-45f7-8f0d-318ccea8362f",

"subject": "Welcome!",

"createdOn": "2024-02-23T11:00:00",

"status": "SUCCEEDED",

"type": "EMAIL"

},

{

"id": "9fd12196-74d9-45f7-8f0d-318ccea8362f",

"subject": "Money Transfer",

"createdOn": "2024-02-23T11:00:00",

"status": "SUCCEEDED",

"type": "EMAIL"

}

]

### **4. Resend Failed Notification**

Retries sending previously failed notifications for a user.  
**Endpoint:**

**PUT /api/v1/notifications/status?user\_id=...**

### **5. Clear Notification History**

Clear all notifications sent to that user, optionally filtered by type. You must assure the data is **NOT** deleted from the database, but still hidden from the user!   
**Endpoint:**

**DELETE /api/v1/notifications?user\_id=...&type=EMAIL**