**Banking Microservices Project**

**1. Introduction**

The **Banking Microservices Project** is a distributed system built using **Spring Boot Microservices architecture**.  
The application simulates a simple banking system with modules for:

## 2.Microservices Overview

* **api-gateway\_bank** : Serves as a single entry point, routing client requests to services.
* **discovery-server\_bank** : Enables service discovery using Eureka.
* **config-server\_bank** : Provides centralized configuration management for all services.
* **account-service\_bank** : Manages account operations like create, update, and view.
* **customer-service\_bank** : Handles customer details, registration, and profile updates.
* **auth-service\_bank** : Provides authentication and JWT-based authorization.
* **transaction-service\_bank** : Processes money transfers, deposits, and withdrawals.
* **statement-service\_bank** : Generates account statements and transaction history.
* **notification-service\_bank** : Sends transaction alerts via email/SMS.
* **audit-service\_bank** : Logs important events and actions for auditing.

## 3. API Endpoints

### 🔹 Authentication Service (Port: 9001)

* **POST /auth/login**
  + Request: { "username": "testuser", "password": "password" }
  + Response: JWT Token
* **GET /auth/validate**
  + Header: Authorization: Bearer <token>
  + Response: "Valid for testuser"

### 🔹 Customer Service (Port: 9002)

* **POST /customers**
  + Create a new customer.
  + Example Request:
  + {
  + "name": "priya",
  + "email": "priya@gmail.com",
  + "phone": "9876543210",
  + "kycStatus": "VERIFIED"
  + }
* **GET /customers/{id}**
  + Retrieve customer details by ID.
* **PUT /customers/{id}**
  + Update customer details.
* **DELETE /customers/{id}**
  + Delete customer by ID.
  + Response: "Customer deleted successfully"

### 🔹 Transaction Service (Port: 9004)

* **POST /api/transactions/transfer**
  + Transfer money between accounts.
  + Example Request:
  + {
  + "fromAccountId": 1,
  + "toAccountId": 2,
  + "amount": 100
  + }
  + Example Response:
  + {
  + "txId": "e570f20d-096e-4f38-961e-cd30ee8ed14e",
  + "status": "SUCCESS",
  + "message": "Transfer completed successfully"
  + }

#### 🔹 Authentication Service (Port: 9001)

**POST /auth/login**  
Request:

{ "username": "testuser", "password": "password" }

Response: JWT Token

**GET /auth/validate**  
Header: Authorization: Bearer <token>  
Response: Valid for testuser

#### 🔹 Customer Service (Port: 9002)

**POST /customers** – Create a new customer.  
Example Request:

{

"name": "John Doe",

"email": "john@gmail.com",

"phone": "9876543210",

"kycStatus": "VERIFIED"

}

**GET /customers/{id}** – Retrieve customer details.  
**PUT /customers/{id}** – Update customer details.  
**DELETE /customers/{id}** – Delete customer by ID.

#### 🔹 Transaction Service (Port: 9004)

**POST /api/transactions/transfer** – Transfer money between accounts.  
Example Request:

{

"fromAccountId": 1,

"toAccountId": 2,

"amount": 100

}

Example Response:

{

"txId": "e570f20d-096e-4f38-961e-cd30ee8ed14e",

"status": "SUCCESS",

"message": "Transfer completed successfully"

}

**Postman Testing:**  
**Error! Filename not specified.**

#### 🔹 Notification Service (Port: 9006)

**POST /api/notifications/send** – Send notifications for transactions.

Example Request:

{

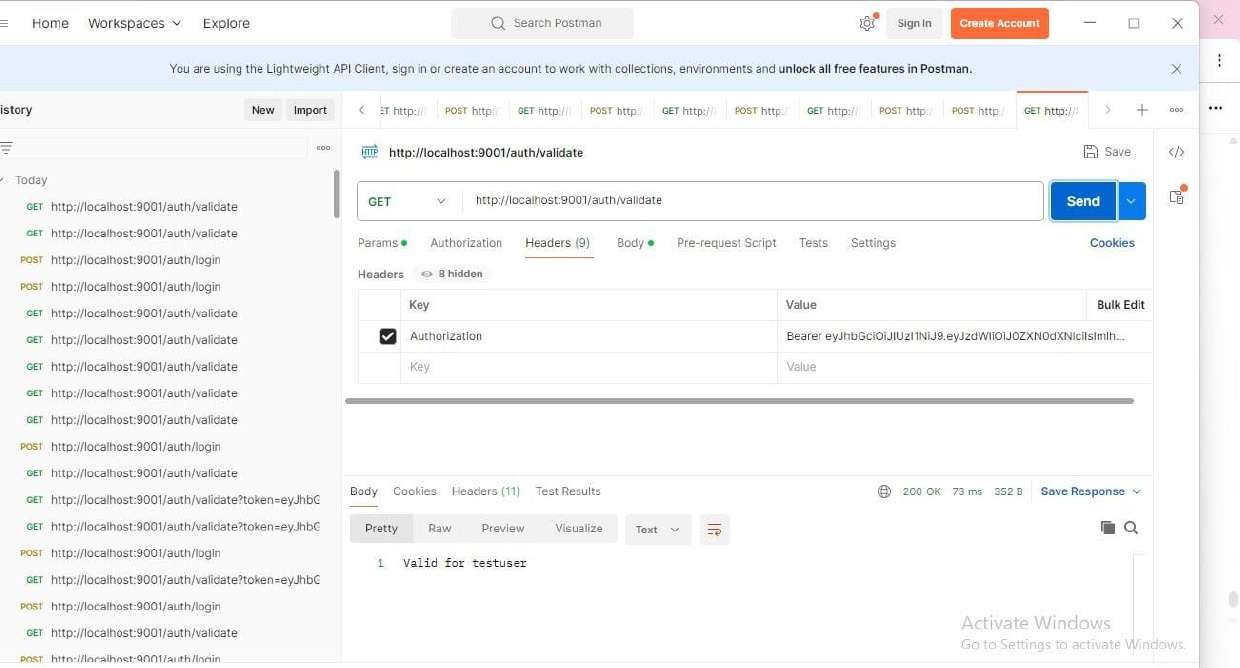
"to": "user@example.com",

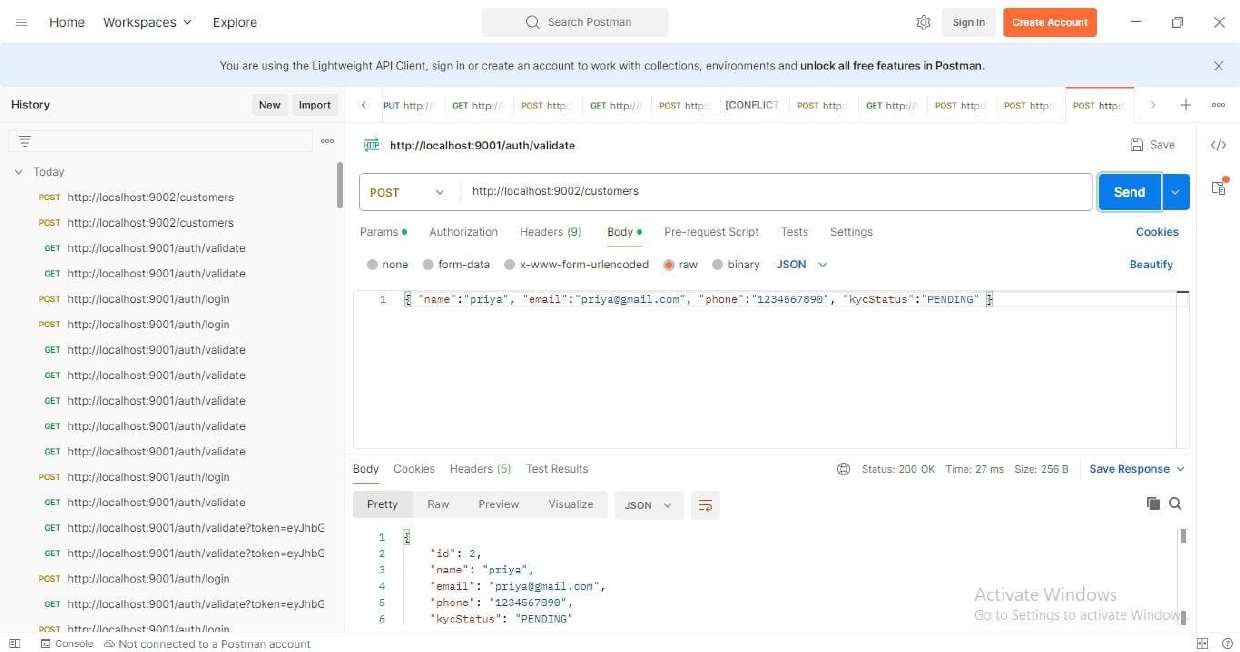
"message": "Test notification",

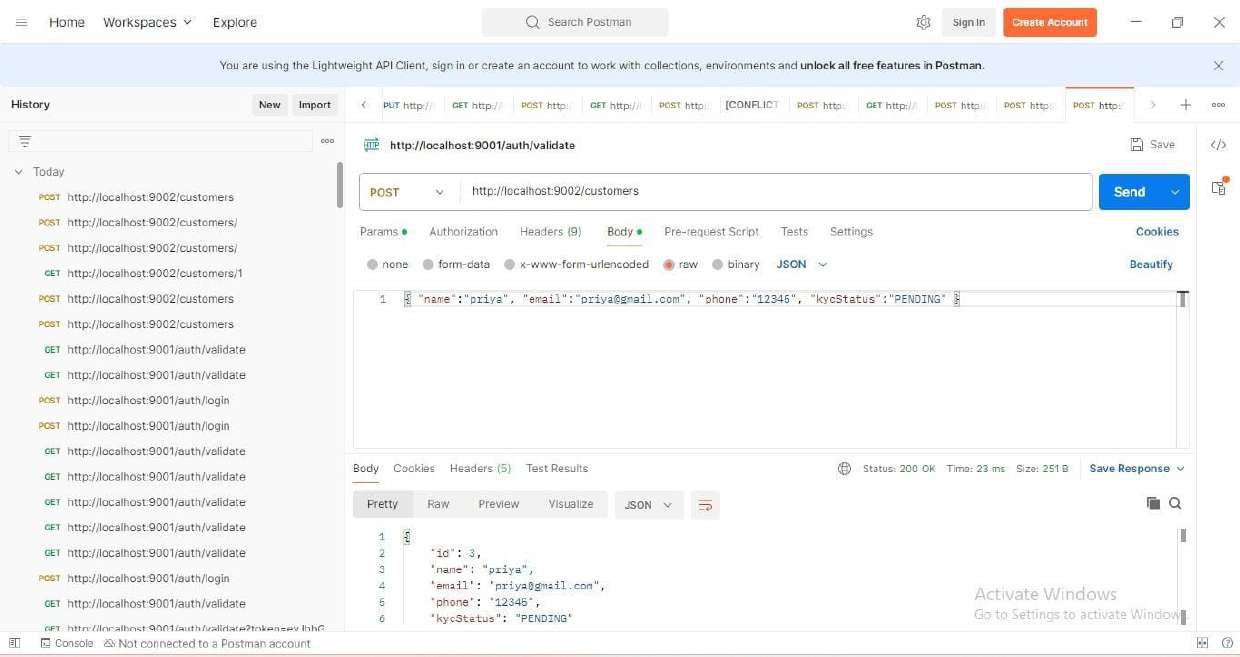
"type": "LOG"

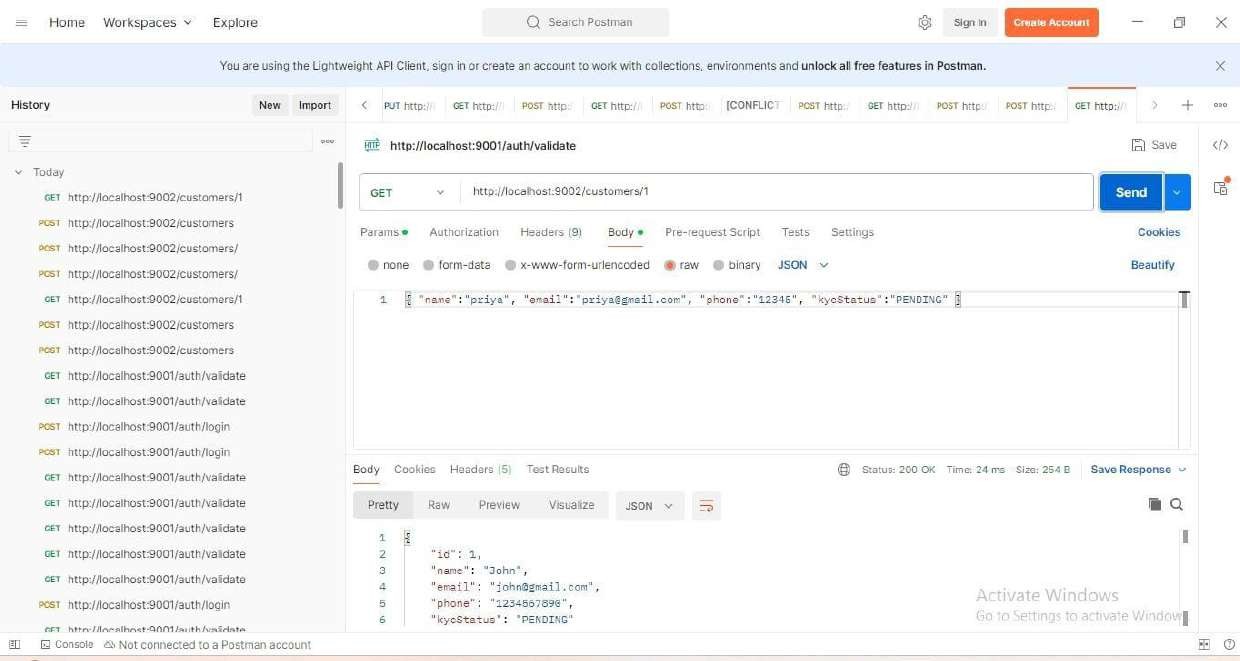
}

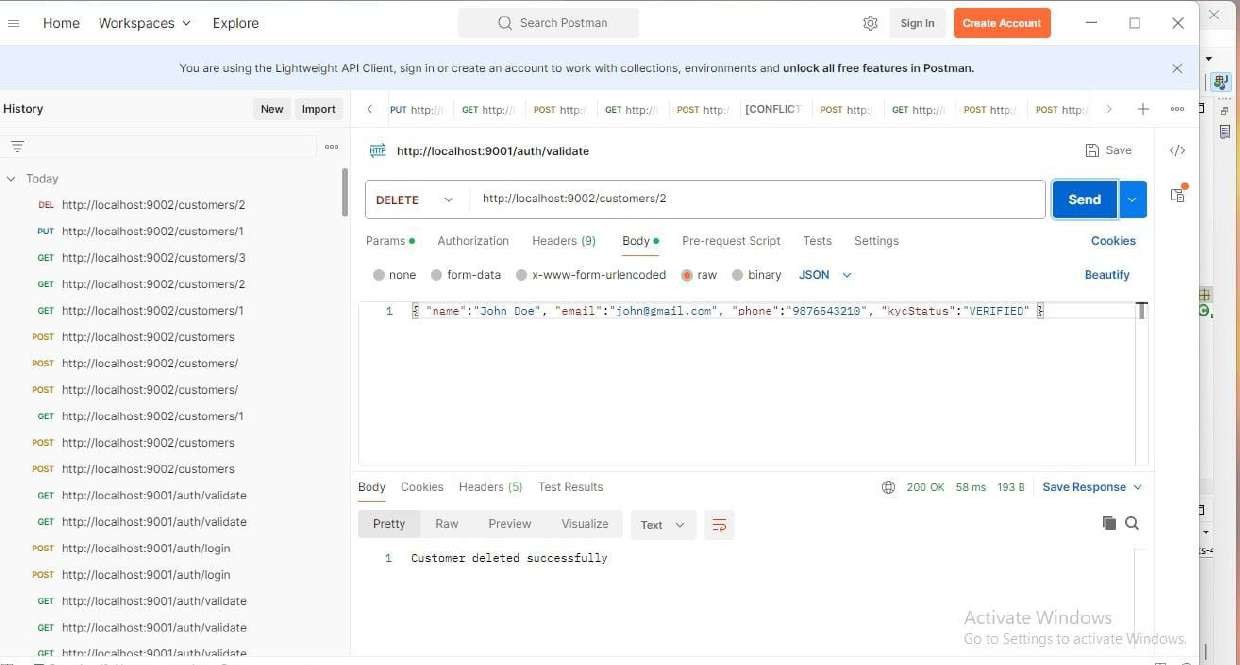
Response: Notification sent successfully!

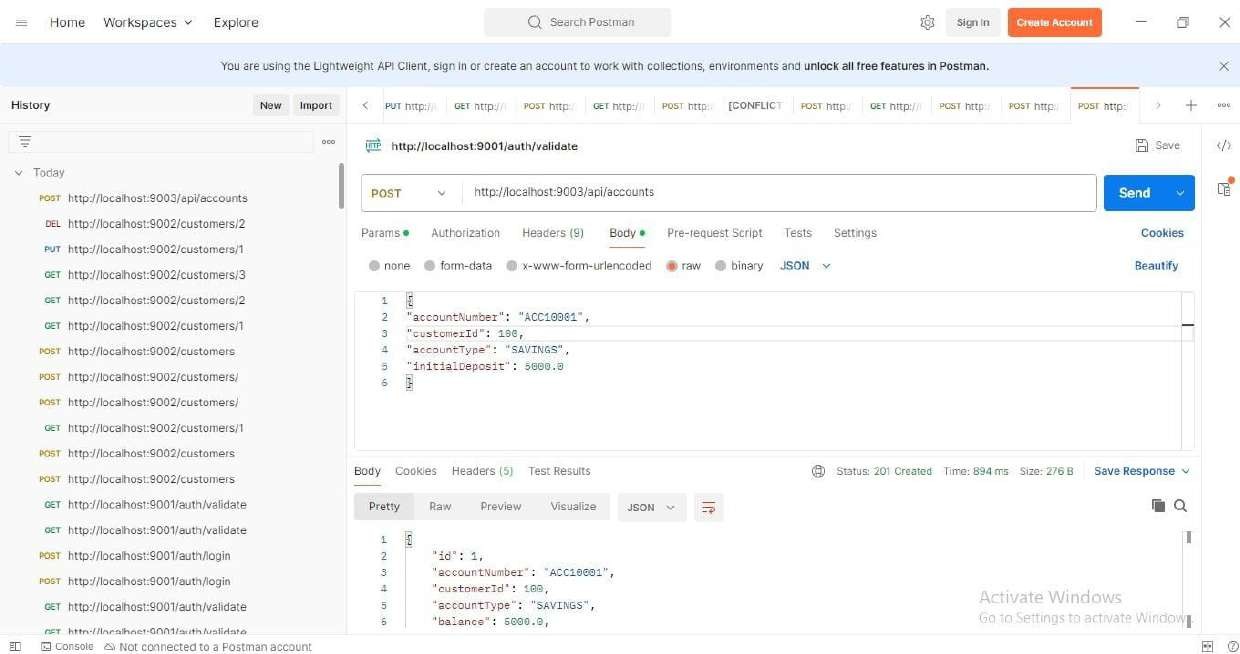


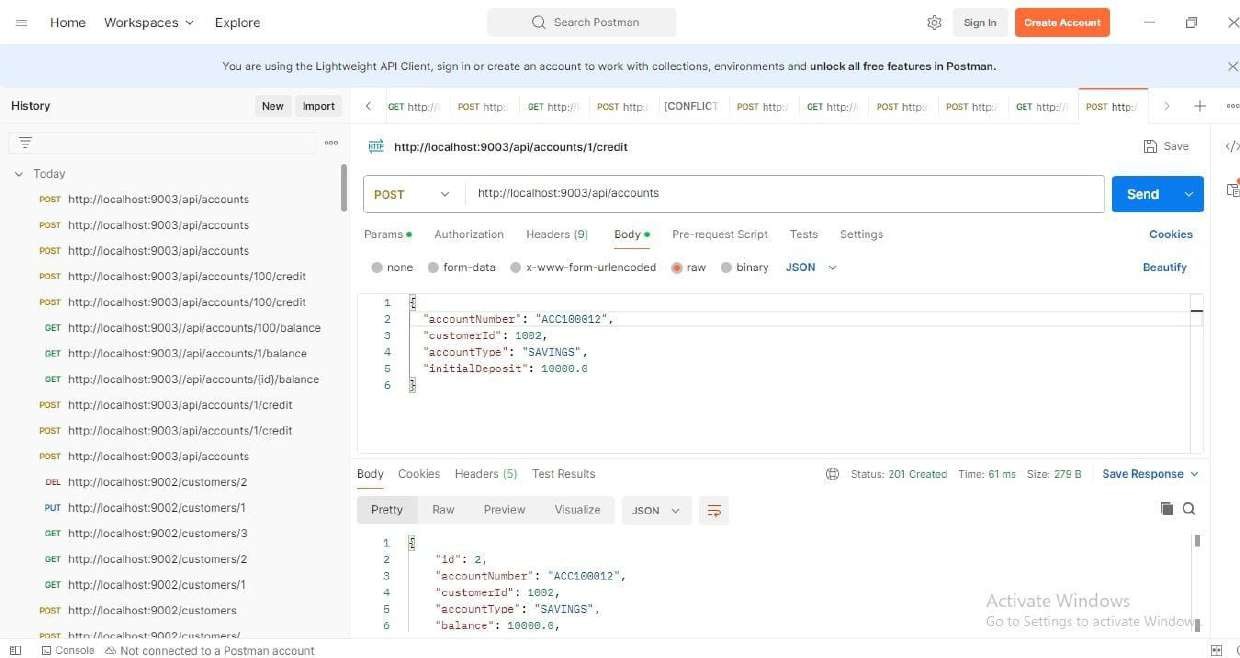


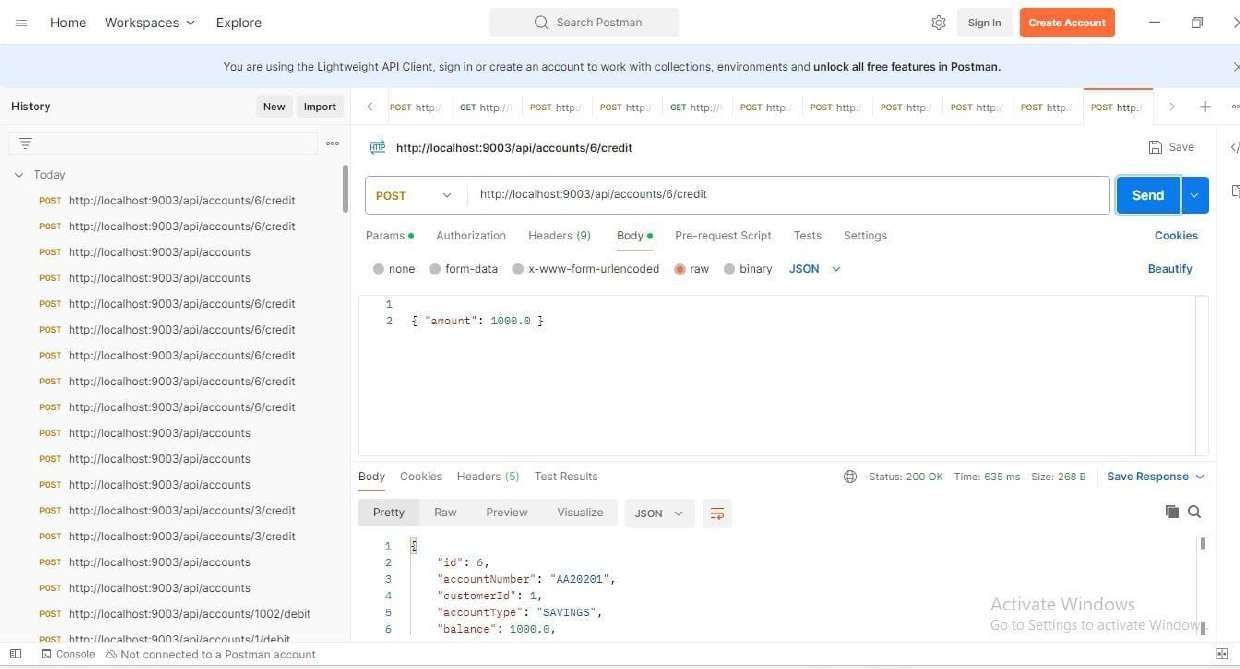


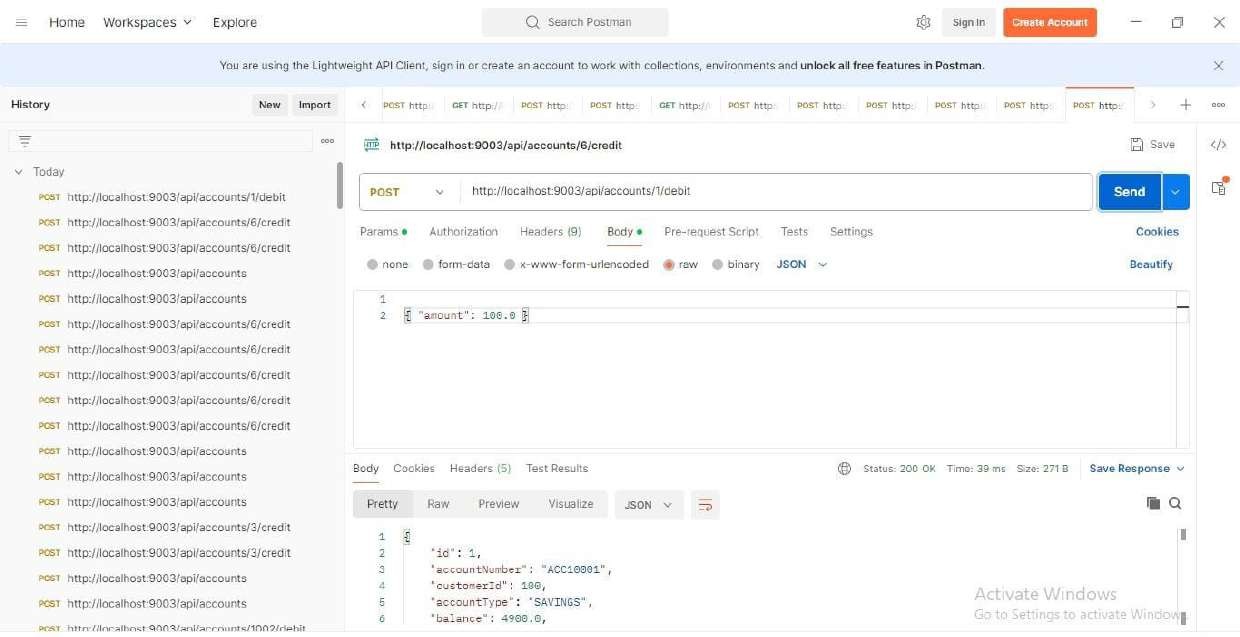


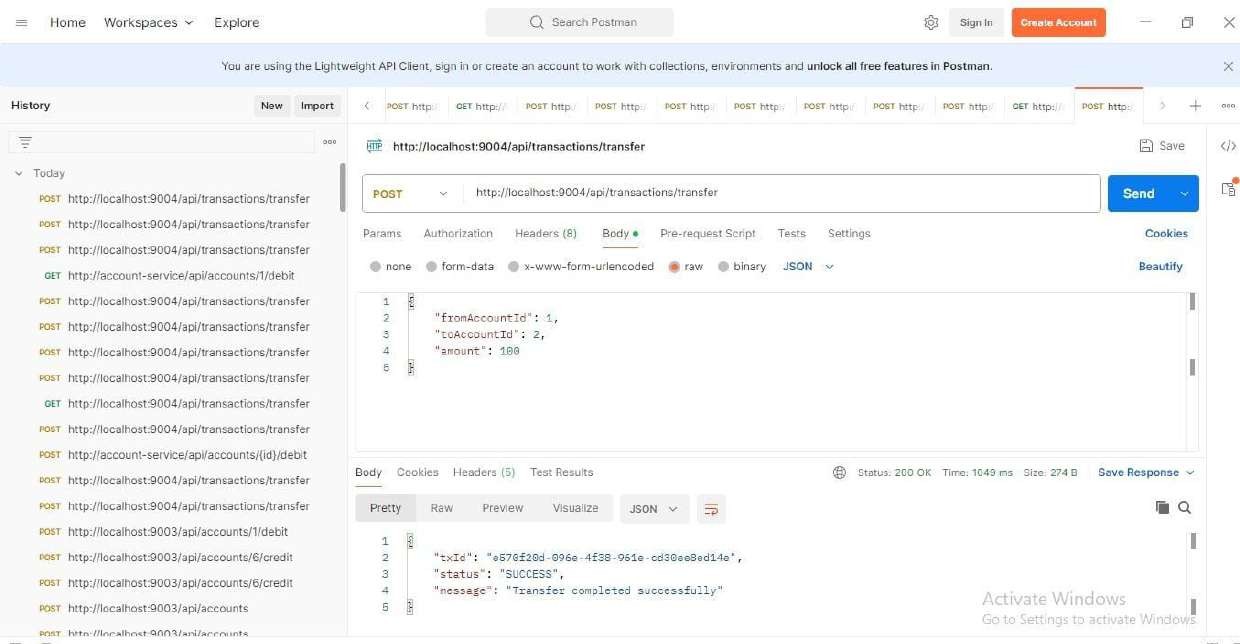


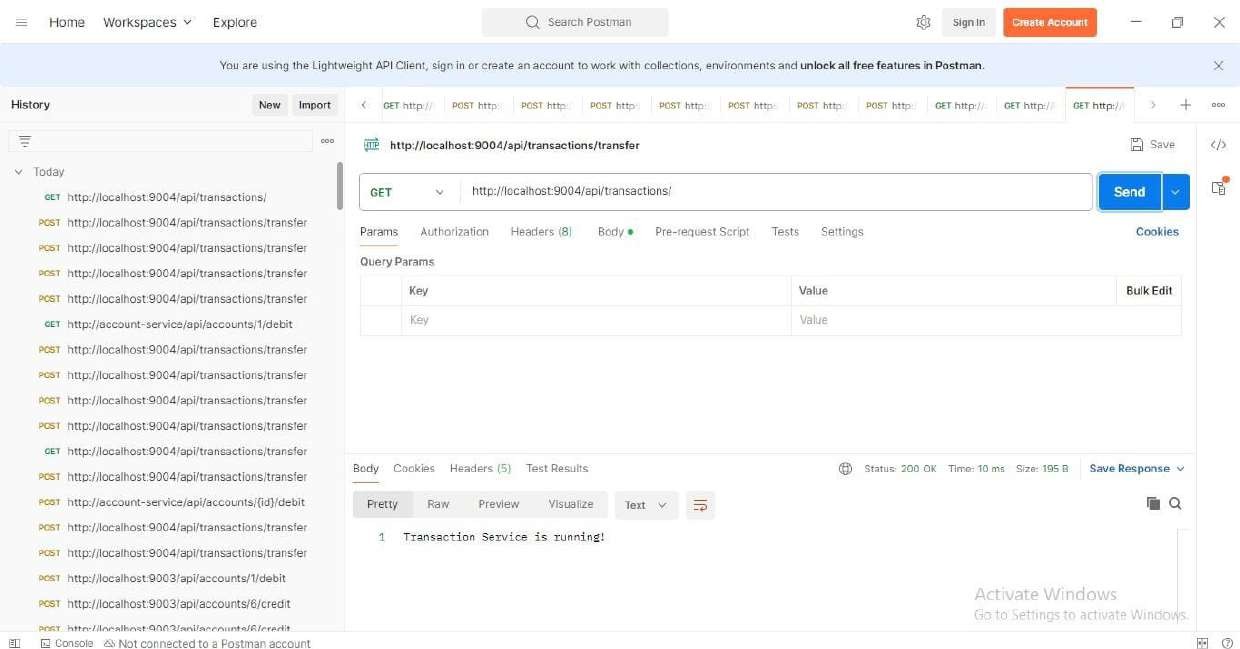


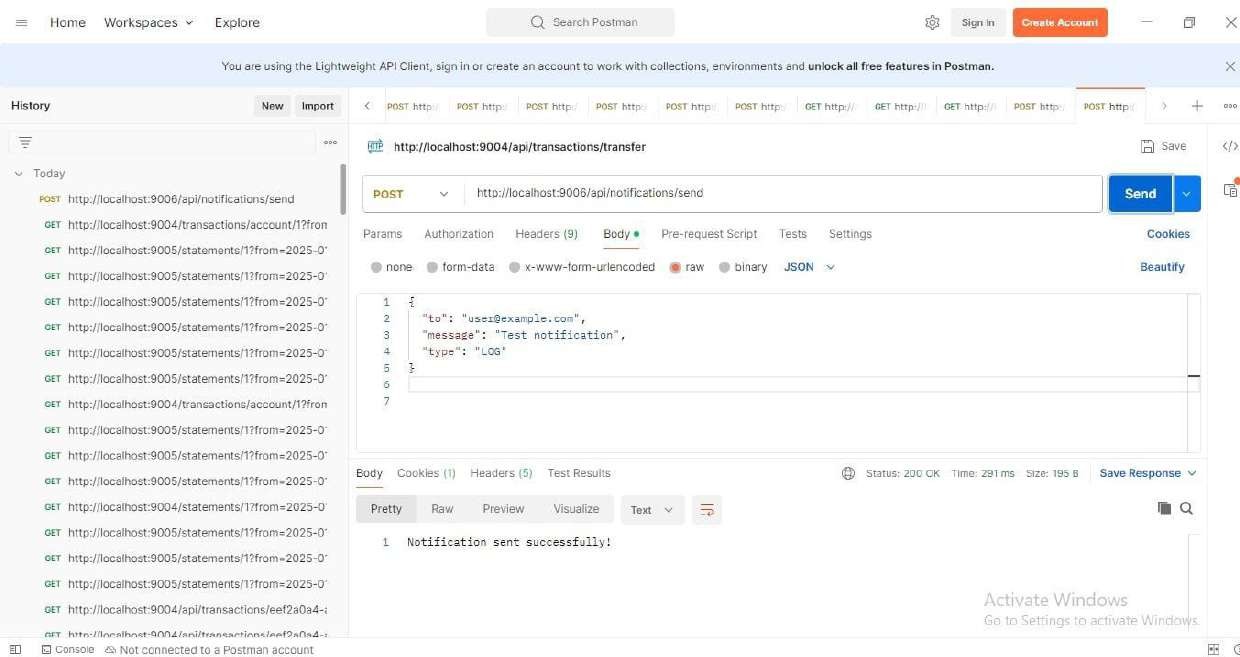


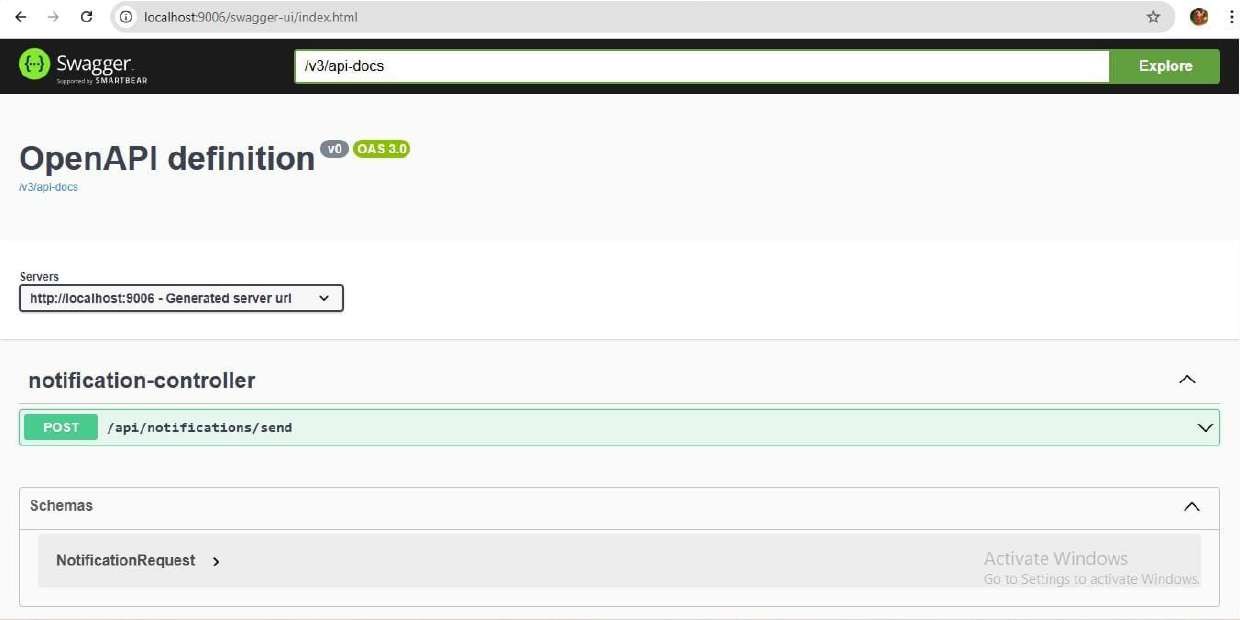


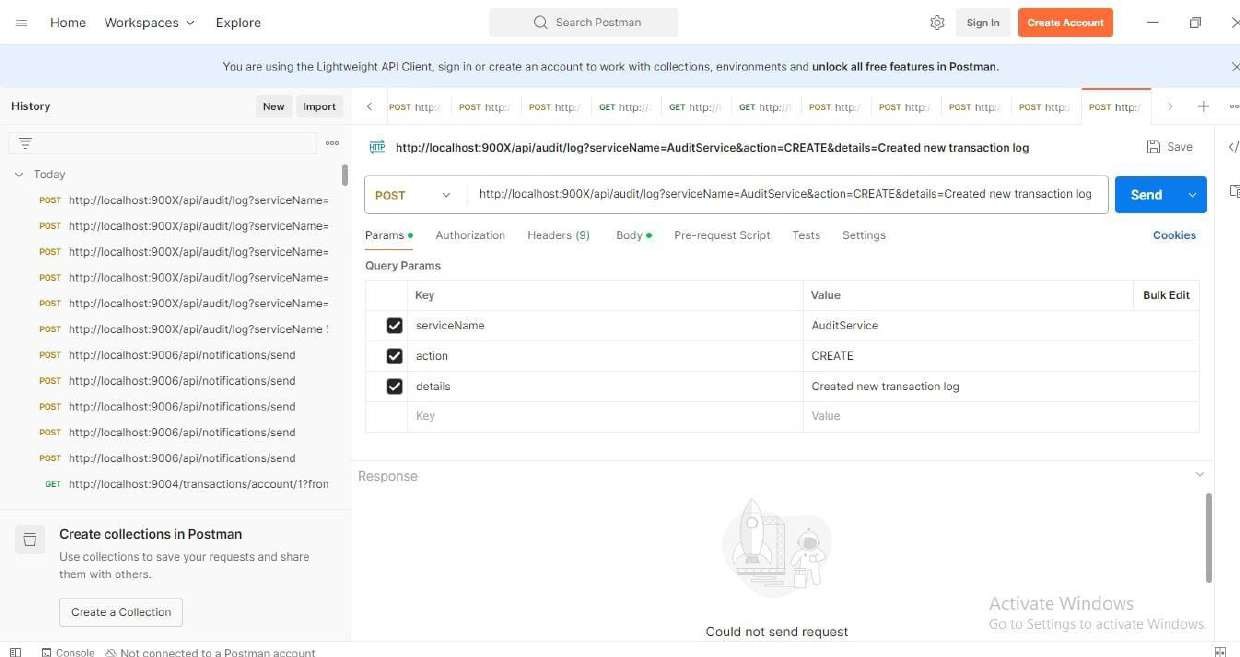


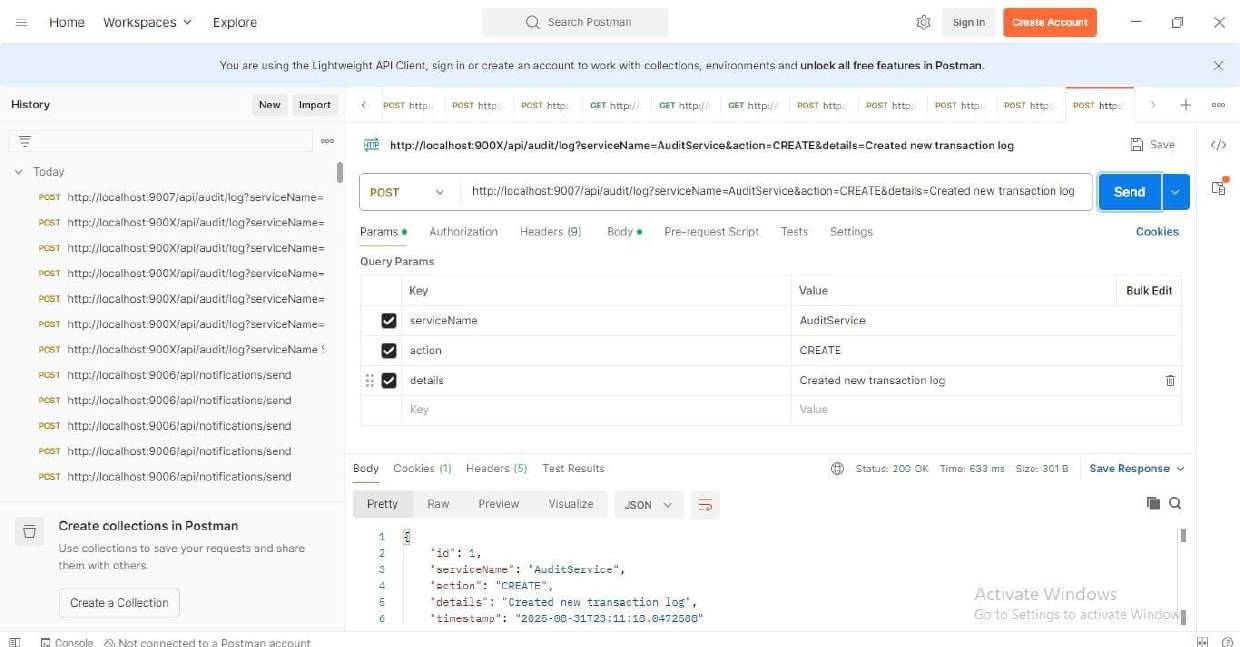












### 4. Database – MySQL Integration

Accounts table in MySQL after transactions:

