

**BANKING SYSTEM BACKEND DEVELOPMENT**

**Project Overview**

This Java Spring Boot-based system is a microservices architecture for managing client data, bank accounts, loans, transactions, and notifications. The API Gateway routes requests to various services like LoginService, ClientService, AccountService, TransactionService, and NotificationService, each handling specific tasks such as user authentication, client and account management, transaction processing, and client notifications. The system uses FeignClient for inter-service communication and likely persists data in a relational database, making it ideal for a financial application.

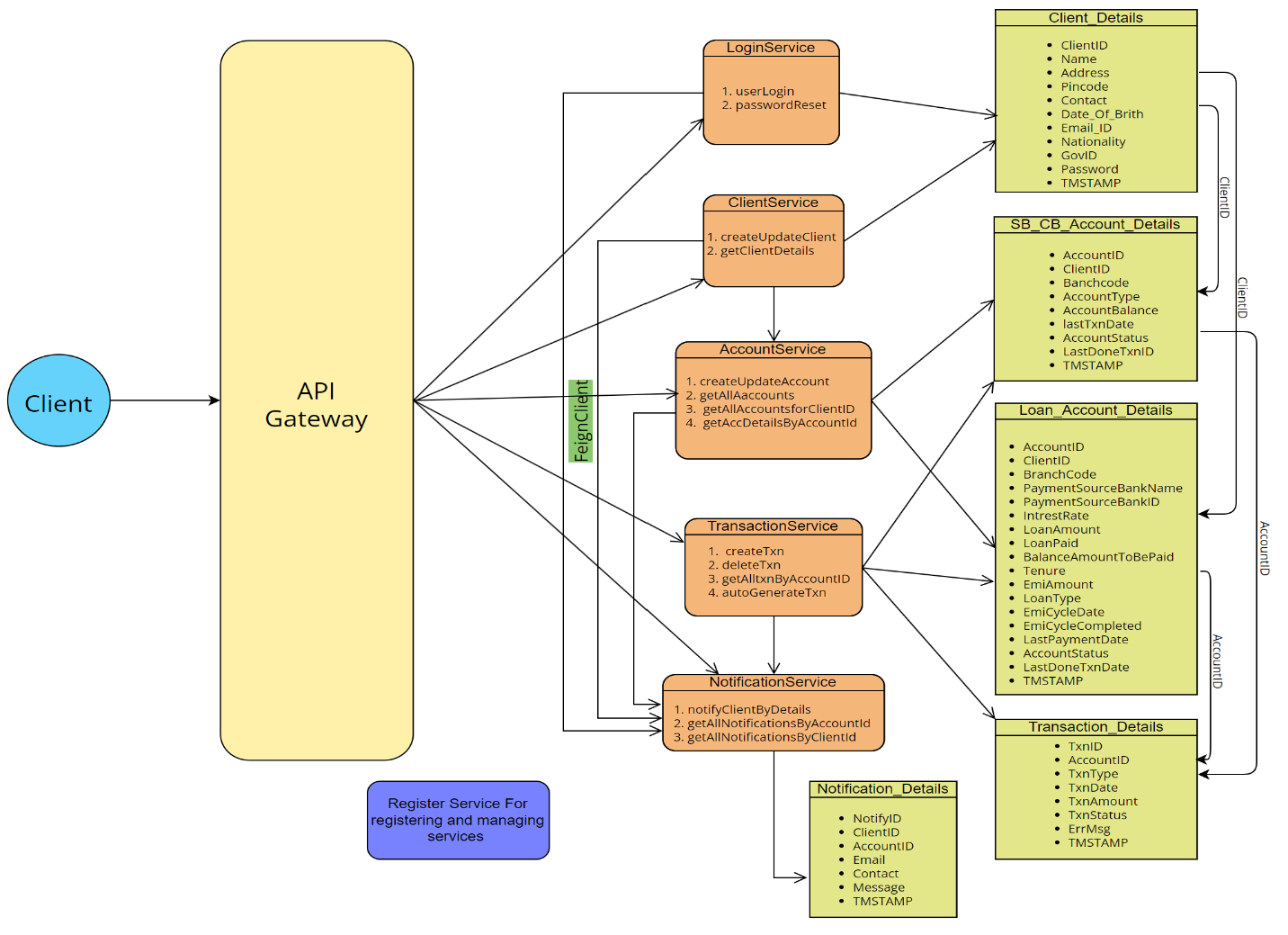
**Developer Details**

* Designed & Developed by Shubham Soman Samanta
* Email ID: [shubhamsamanta9302@gmail.com](mailto:shubhamsamanta9302@gmail.com)
* Contact: +91 8108442967
* Github: https://github.com/shubhamsamanta1

**Contents:**

* **Phase 1**: Initial Design Overview
* **Phase 2:** Implementing LoginService microservice
* **Phase 3:** Implementing ClientService microservice
* **Phase 6:** Implementing NotificationService microservice

**Phase 1**: Initial Design Overview

The diagram outlines a Java Spring Boot-based microservices architecture for managing client data, bank accounts, loans, transactions, and notifications within a financial application.

**1.1 API Gateway**

* Acts as the entry point, routing client requests to the appropriate microservices and handling tasks like authentication and load balancing.

**1.2 Microservices**

* LoginService:
  + Manages user authentication, including login and password reset.
* ClientService:
  + Handles client data operations, such as creating, updating, and retrieving client details.
* AccountService:
  + Manages bank and loan accounts, including account creation, updates, and retrievals.
* TransactionService:
  + Oversees transaction processing, including creation, deletion, and retrieval of transactions.
* NotificationService:
  + Manages client notifications, sending alerts and retrieving notification history.

**1.3. Data Model**

* Client\_Details:
  + Stores client information.
* SB\_CB\_Account\_Details:
  + Manages savings and current bank account details.
* Loan\_Account\_Details:
  + Tracks loan-related account information, including payments and balances.
* Transaction\_Details:
  + Logs transactions linked to accounts.
* Notification\_Details:
  + Captures notifications sent to clients.

**1.4. Inter-service Communication**

* Services communicate via REST APIs, with FeignClient used for HTTP requests between services.

**1.5. Database and Persistence**

* The system likely uses a relational database, with each microservice managing its data model entities.

This architecture provides a modular, scalable solution for managing clients and their financial interactions, making it ideal for financial institutions.

**Phase 2**: Implementing LoginService microservice.

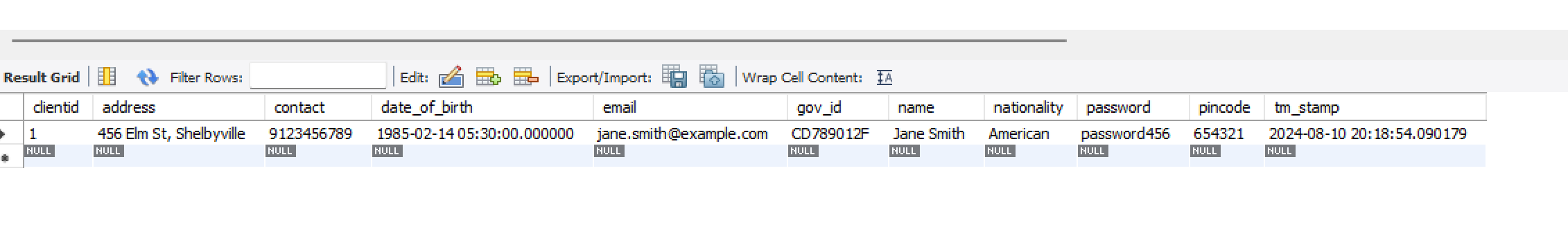
The `LoginService` class handles user login and password reset operations with integrated notifications via Feign (`CallNotify`).

* **userLogin(ClientDetail clientDetail)**
  + Checks client credentials.
  + Sends a "login success" notification if credentials are correct, or an "incorrect password" notification if they are wrong.
  + Returns a status message indicating the login result.
* **passwordReset(ClientDetail clientDetail)**
  + Updates the client’s password and sends a "password update success" notification.
  + The method is transactional to ensure atomicity of the operation.

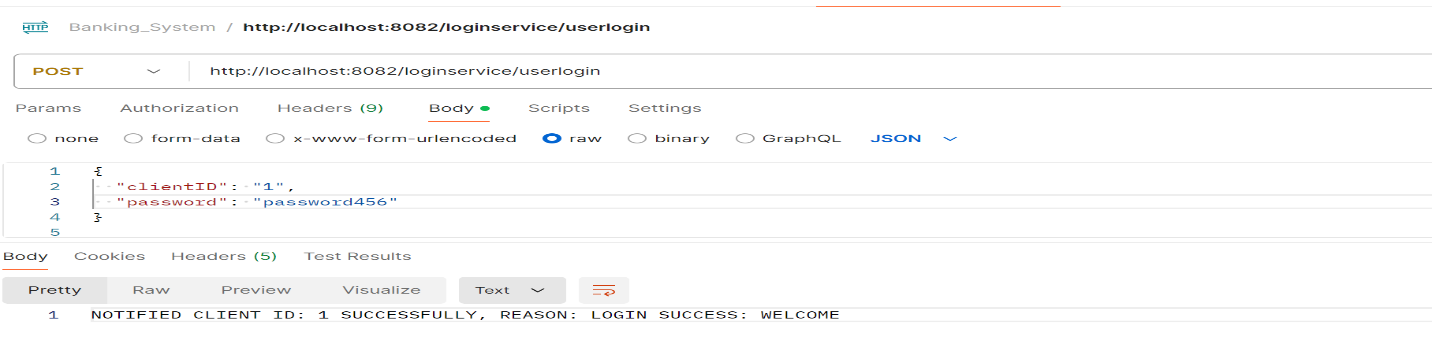
This service ensures reliable user authentication and password management, with real-time notifications for key actions.

Please Find the below Unit test cases -

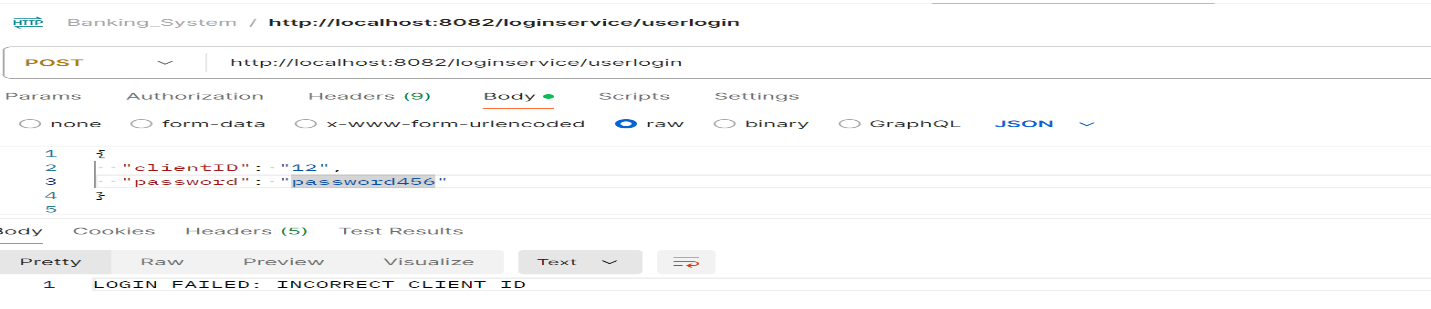
1. Current data in DB -



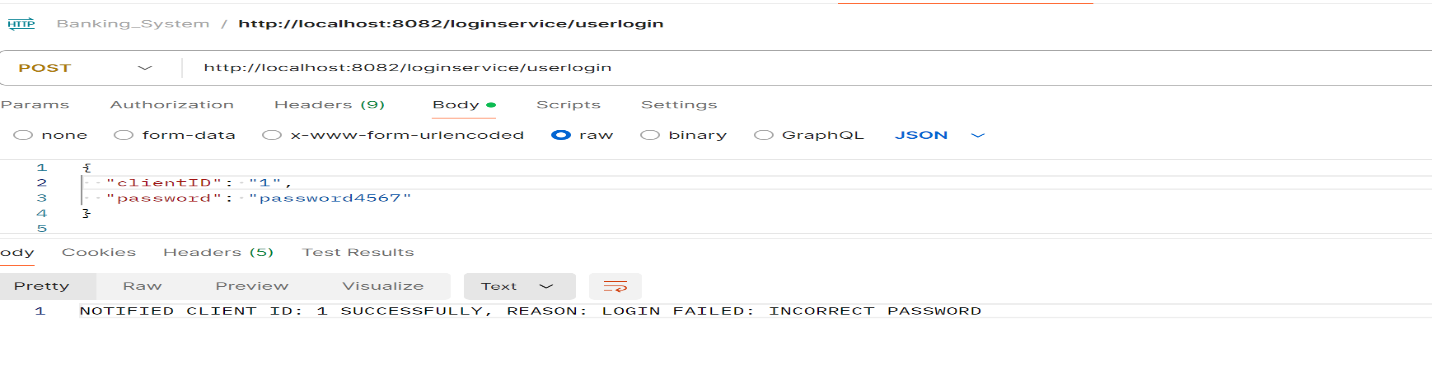
1. When client id and password both matches -

 Result - NOTIFIED CLIENT ID: 1 SUCCESSFULLY, REASON: LOGIN SUCCESS: WELCOME

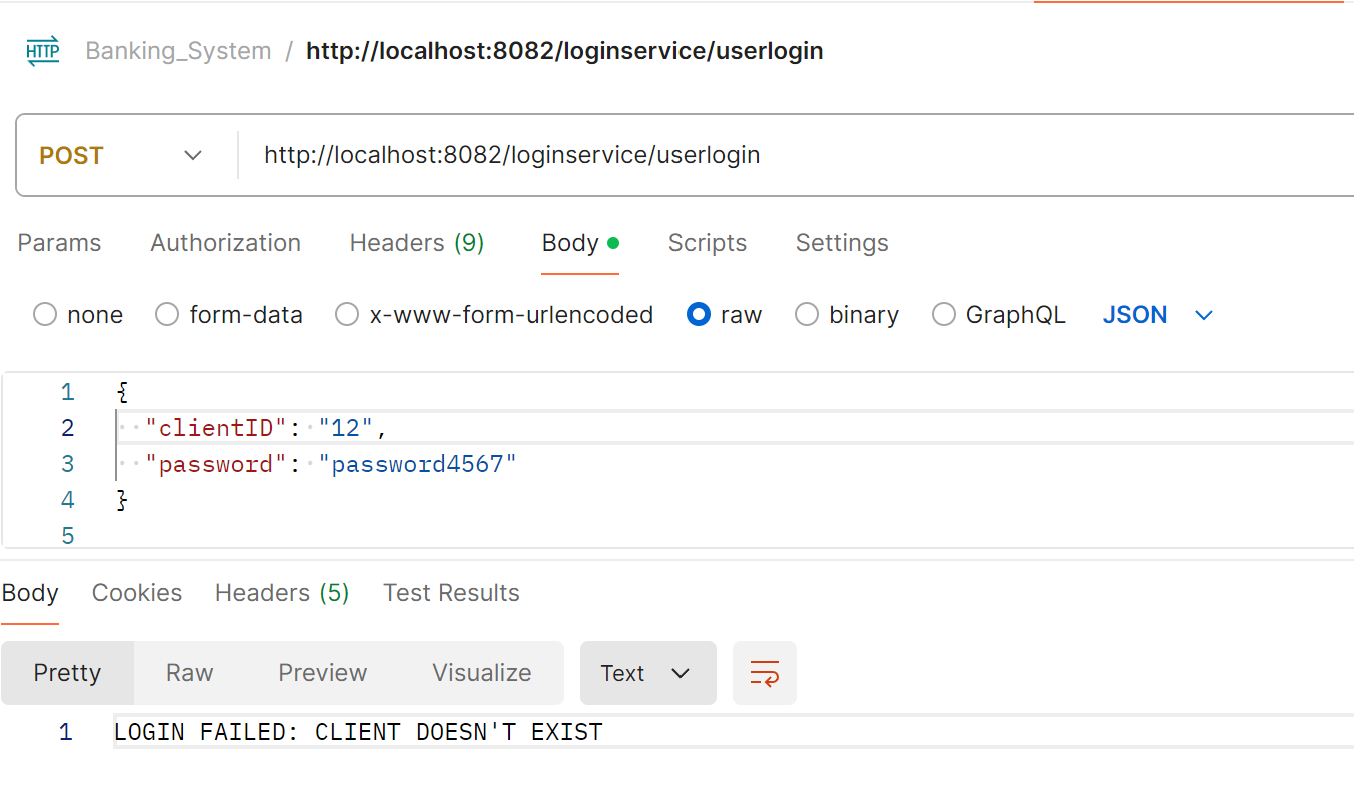
1. When client is incorrect and password is correct -

 Result - LOGIN FAILED: INCORRECT CLIENT ID

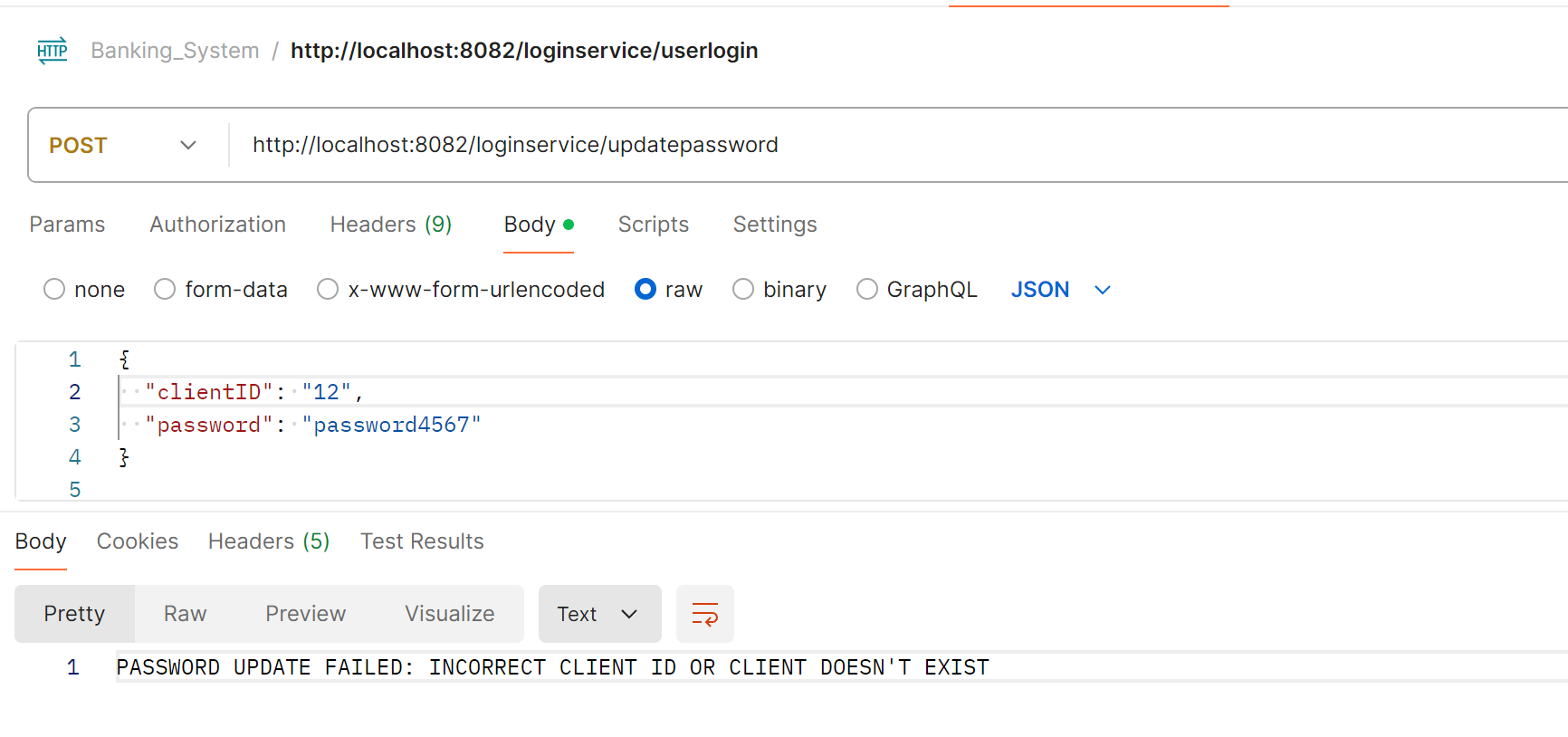
1. When client is correct and password is incorrect -

Result - NOTIFIED CLIENT ID: 1 SUCCESSFULLY, REASON: LOGIN FAILED: INCORRECT PASSWORD

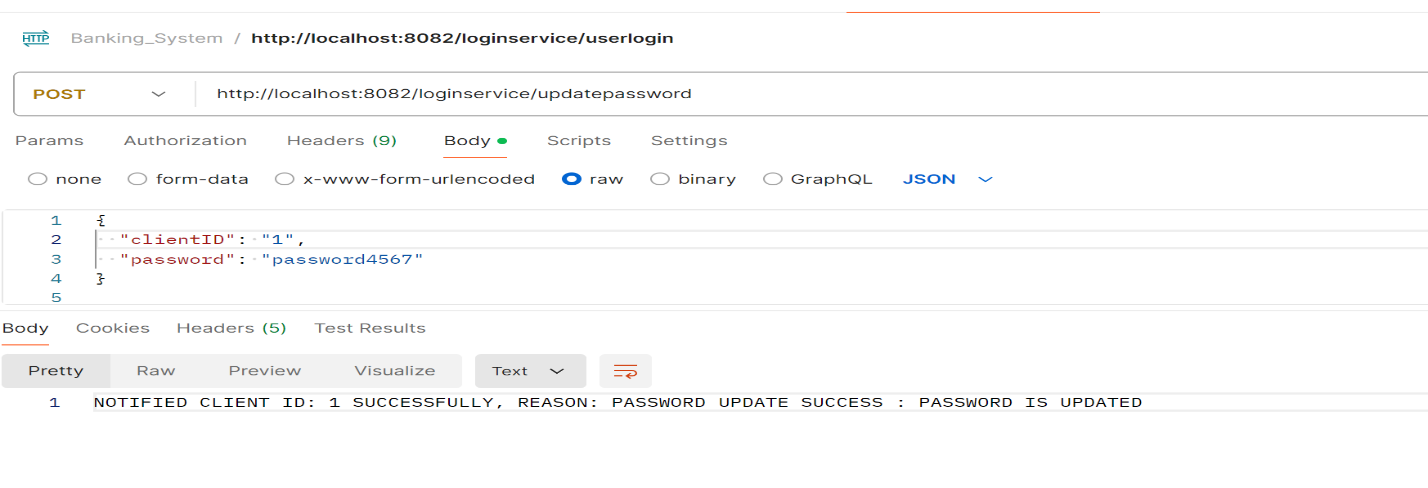
1. When client and password both are incorrect -

Result - LOGIN FAILED: CLIENT DOESN'T EXIST

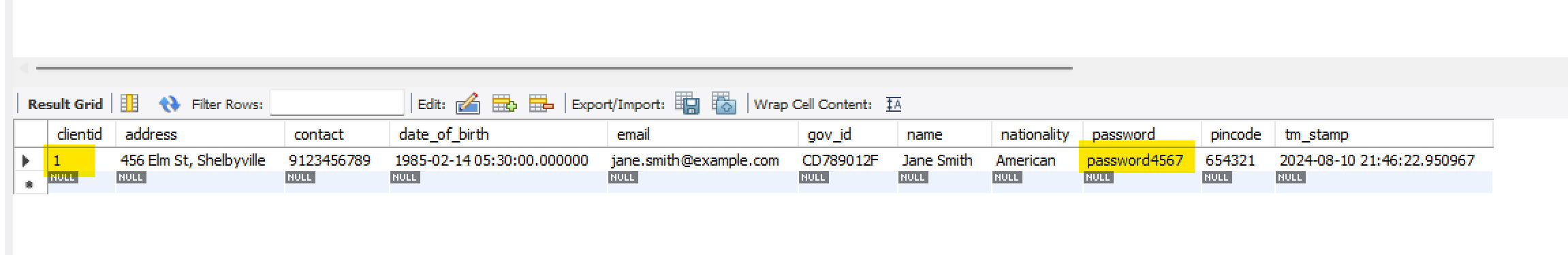
1. When trying to update password with incorrect client id -

Result - PASSWORD UPDATE FAILED: INCORRECT CLIENT ID OR CLIENT DOESN'T EXIST

1. When trying to update password with correct client id -

Result - NOTIFIED CLIENT ID: 1 SUCCESSFULLY, REASON: PASSWORD UPDATE SUCCESS: PASSWORD IS UPDATED

We can see password change reflecting on Database as well.



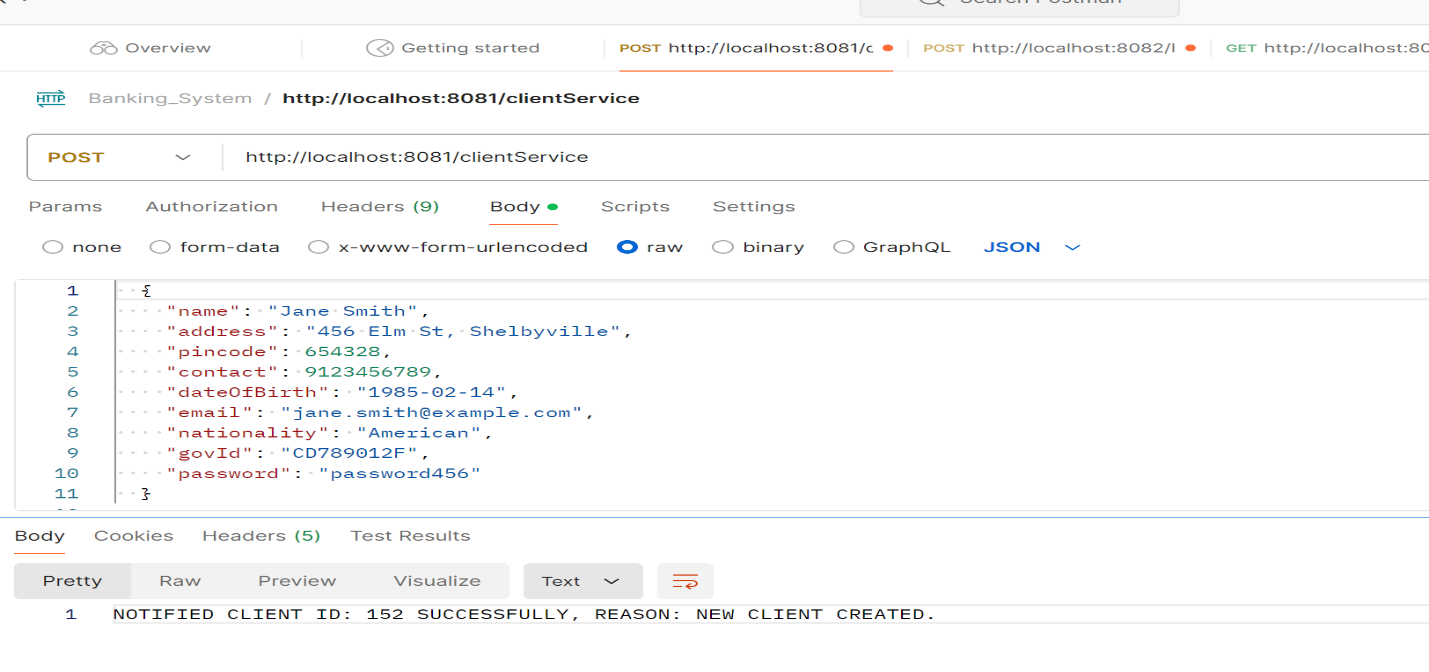
**Phase 3**: Implementing ClientService microservice.

The ClientDetailService class manages client data creation, updating, and retrieval, with integrated notifications via Feign (CallNotify).

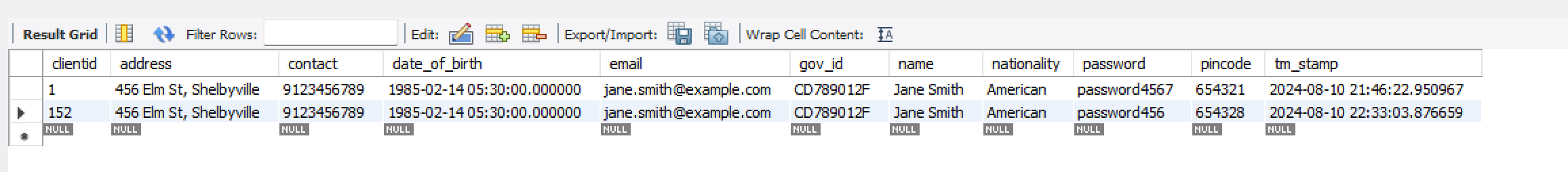
* **createUpdateClient(ClientDetail clientDetail)**:
  + If clientID is present, updates the existing client details; otherwise, creates a new client.
  + Saves the client data using clientDetailRepo.
  + Sends a notification with either "NEW CLIENT CREATED" or "CLIENT DETAILS UPDATED" based on the operation.
* **getClientDetails(Long clientID)**:
  + Retrieves client details by clientID.
  + Throws a RuntimeException if the client is not found.

This service ensures efficient client management with real-time notifications for client creation or updates.

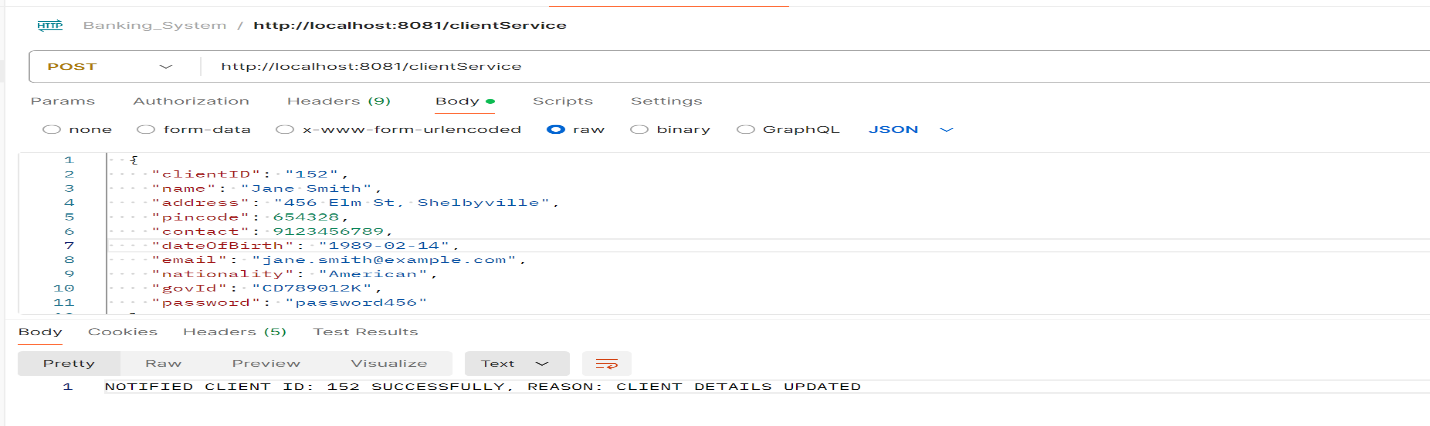
Please Find the below Unit test cases -

1. When we try to insert a new client detail record - Result - NOTIFIED CLIENT ID: 152 SUCCESSFULLY, REASON: NEW CLIENT CREATED.

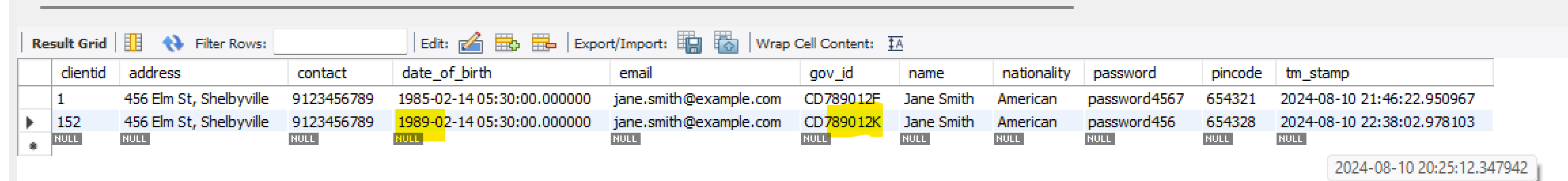
We can see new client detail entry reflecting on Database as well.



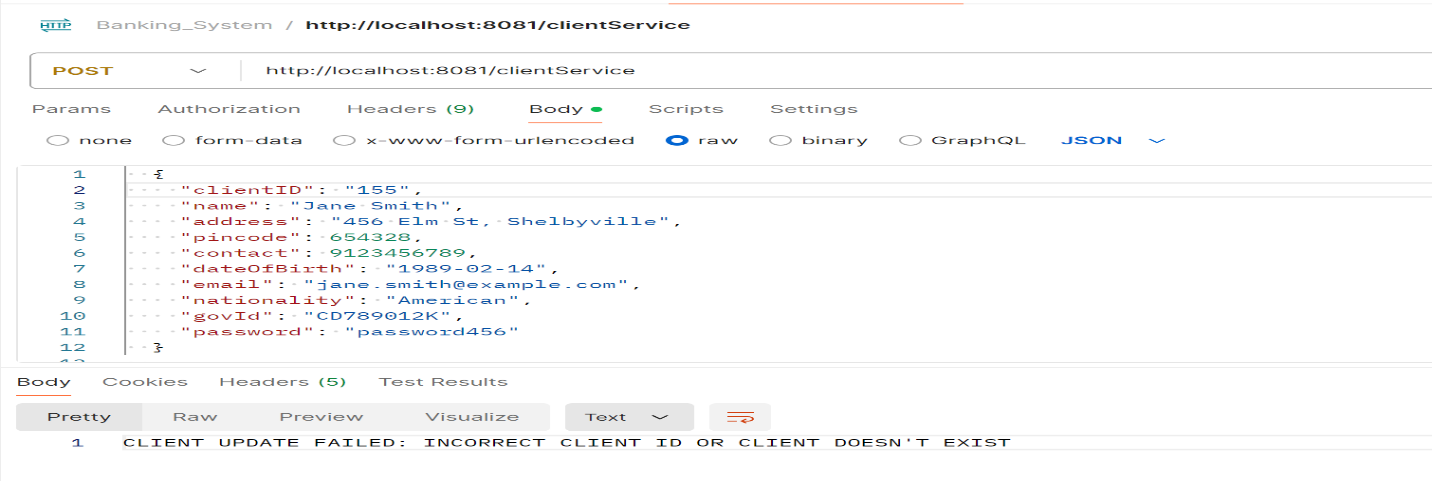
2. When we try to update an existing client detail record-

Result - NOTIFIED CLIENT ID: 152 SUCCESSFULLY, REASON: CLIENT DETAILS UPDATED.

We can see update on client detail reflecting on Database as well.

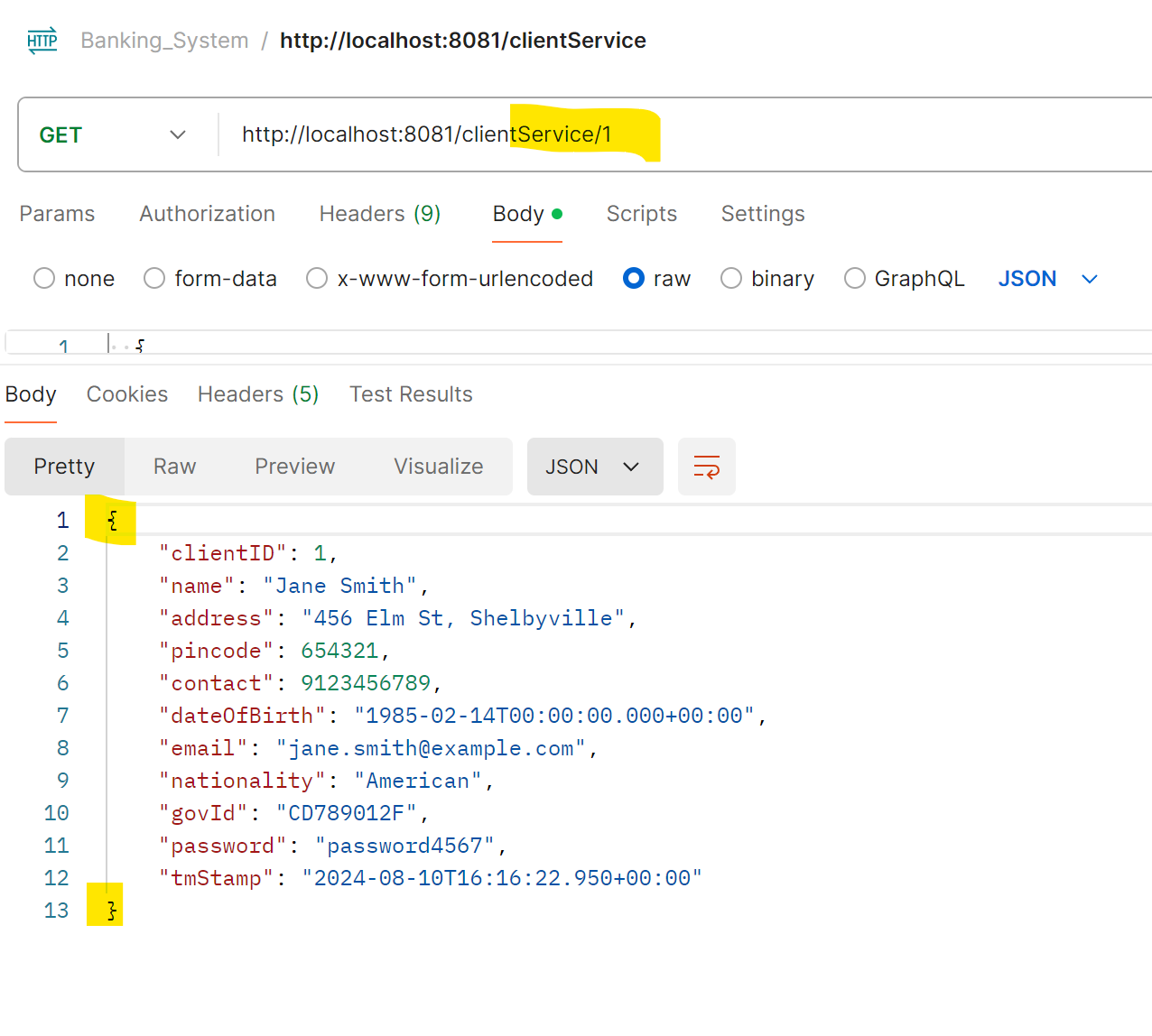


3. When we try to update an existing client detail record but the client id is incorrect (not present on database) -



Result - CLIENT UPDATE FAILED: INCORRECT CLIENT ID OR CLIENT DOESN'T EXIST.

4. When we try to fetch existing client detail record with client id from database -

Result - It fetches the client detail record based on client id passed in parameter from database.

**Phase 5**: Implementing NotificationService microservice.