**ASSIGNMENT**

**1. Overview**

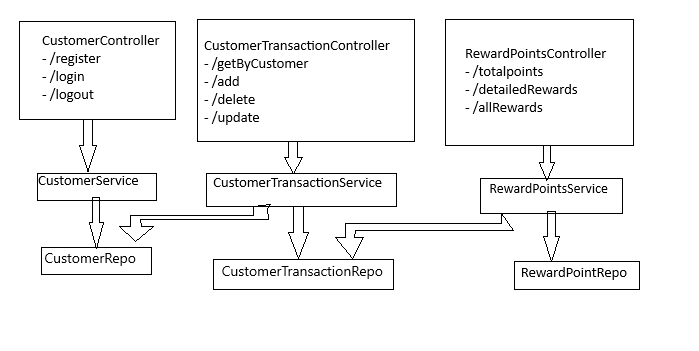
A retailer offers a rewards program to its customers, awarding points based on each recorded purchase.

**Purpose**: Application is used to calculate rewards points earned by customers on each transaction.

**Scope:** Customers are first registered before using the application. Transactions can be added, deleted and updated against the registered customer. Based on the amount of transaction rewards are calculated. A customer receives 2 points for every dollar spent over $100 in each transaction, plus 1 point for every dollar spent between $50 and $100 in each transaction. Total points earned by a customer and detailed points added based on year and month can be retrieved. Login and logout functionalities are included. Apis are accessible only on the basis of roles mentioned during registration.

**2. Architecture Diagram**

Spring Boot Backend Services



**3. Database Design**

PostgreSql is used for database connectivity

Tables - customer, customer\_transaction, reward\_point, roles

1. customer – Saves registered customers

Primary key: id

Fields: name, email, password

1. customer\_transaction – Saves transactions of customers

Primary key: id

Fields: amount, spentDetails, date

Foreign key: customer\_id

1. reward\_point – Saves reward points calculated during addition of transaction

Primary key: id

Fields: month, year, points

Foreign key: customer\_id, transaction\_id

1. Roles – role and corresponding id are populated at the start of application.

This table will be used during customer registration

**4. Component Breakdown**

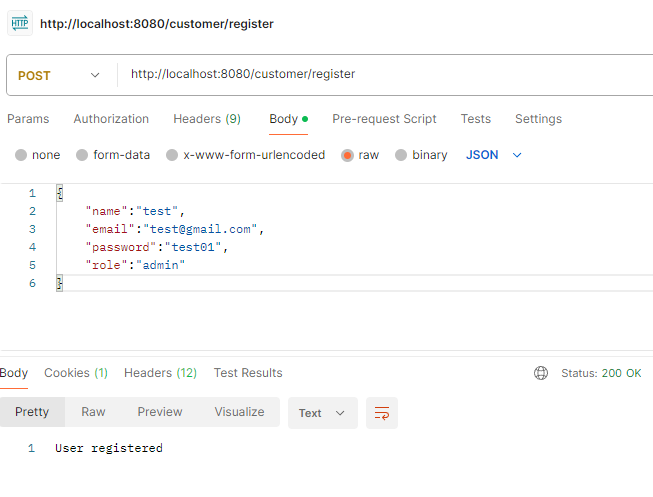
User authorized APIs - /transactions/getByCustomer, /rewardpoints/totalpoints, /rewardpoints/detailedRewards, /rewardpoints/allRewards

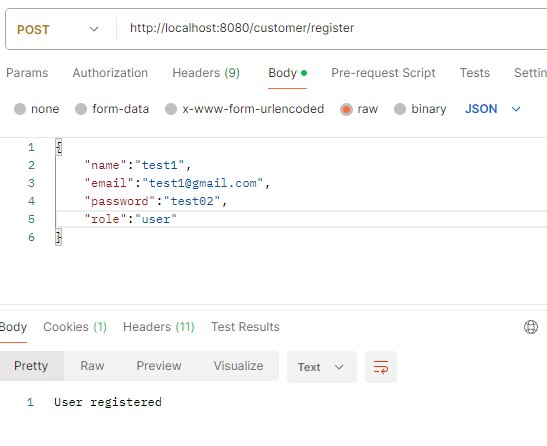
Admin authorized APIs - all

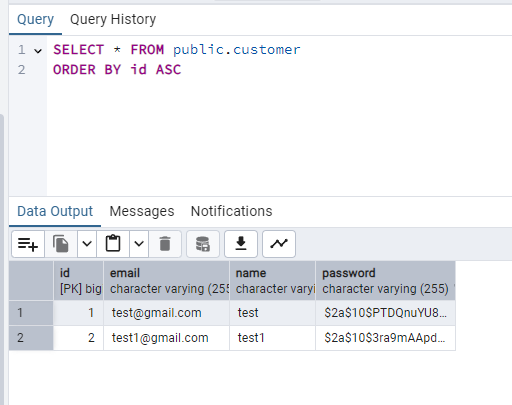
**1. /customer/register**

Used to register customers. Name, email-id, password and role have to be passed in the payload for successful registration. Password is encoded and stored. APIs are accessible based on roles.

Input fields - Name, email-id, password and role



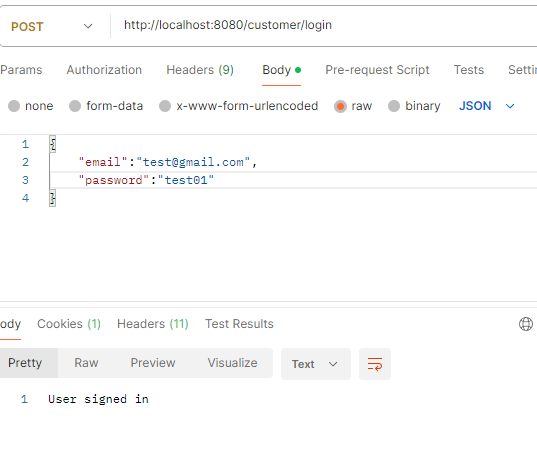




**2. /customer/login**

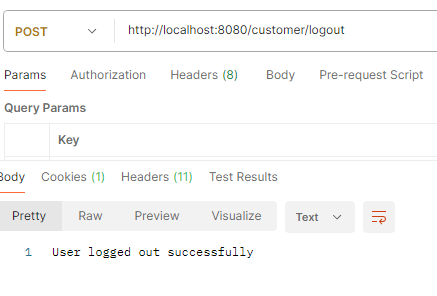
Used to login customers. User is authenticated using Email-id and password.

Input fields - email-id, password



**3. /customer/logout**

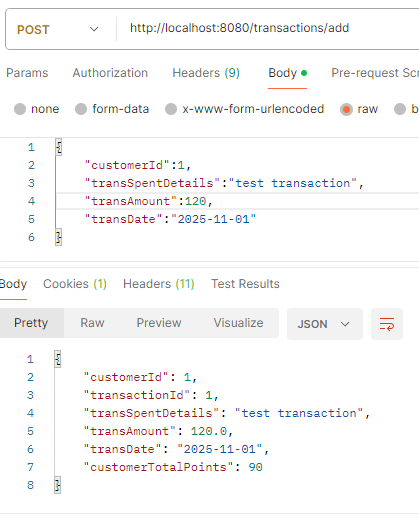
Used to logout the current user

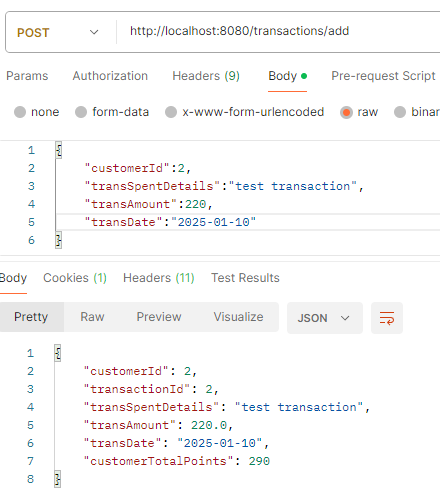


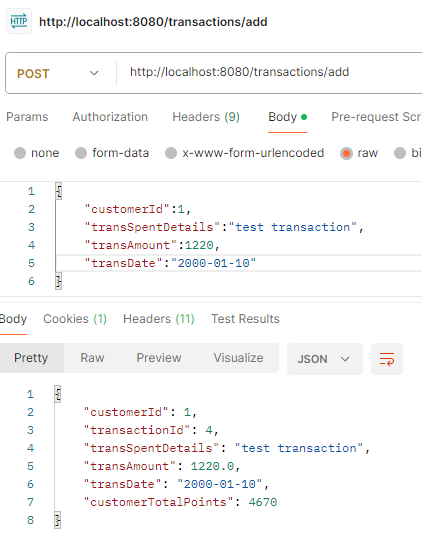
**4. /transactions/add**

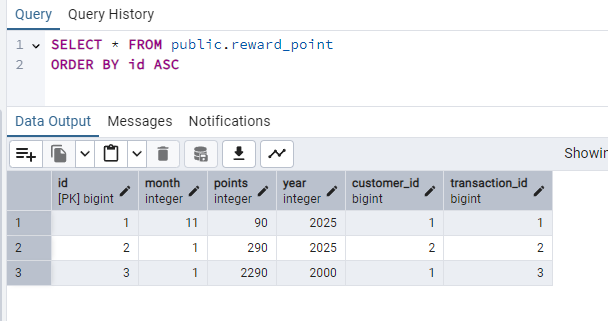
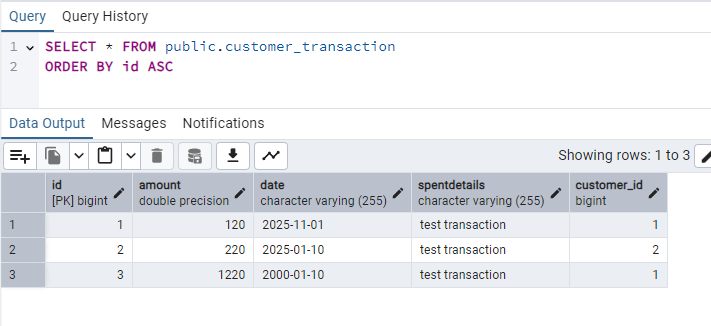
This API adds transaction to the transaction table for a customer. Reward points are calculated based on the data sent from payload and added in rewards point table.

Input fields - Customer id, transaction amount, transaction spent Details, transaction date





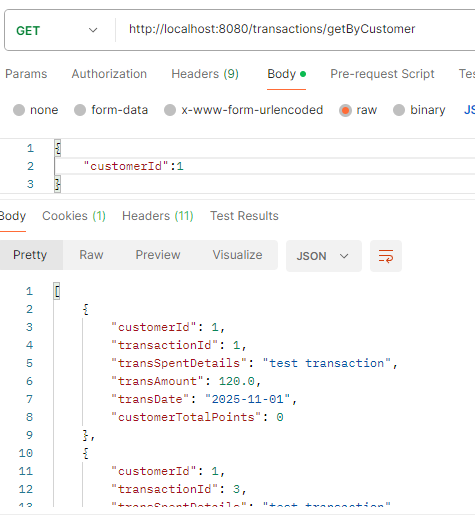




**5. /transactions/getByCustomer**

This API provides a list of transactions of a customer.

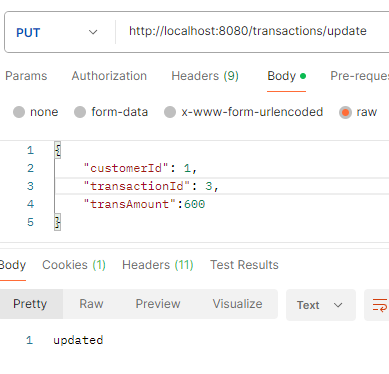
Input fields - Customer id

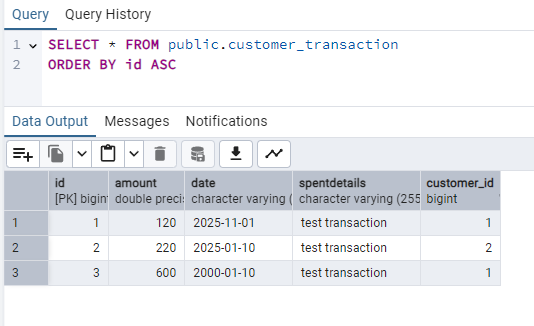


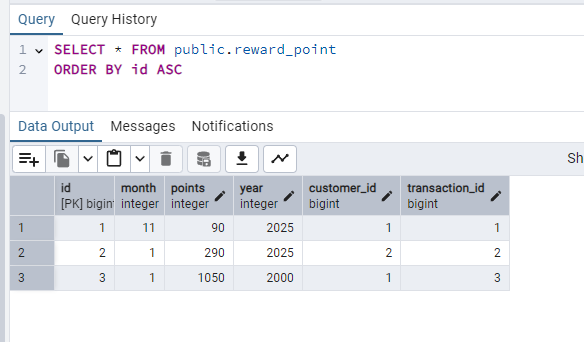
**6. /transactions/update**

Updates the transaction of a customer and hence reward points table will also be updated if needed

Input fields – Customer id, transaction id and fields to be updated



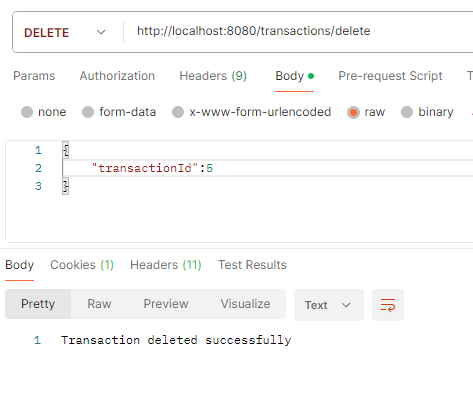




**7. /transactions/delete**

Deletes transaction of a customer and hence reward points for corresponding transaction

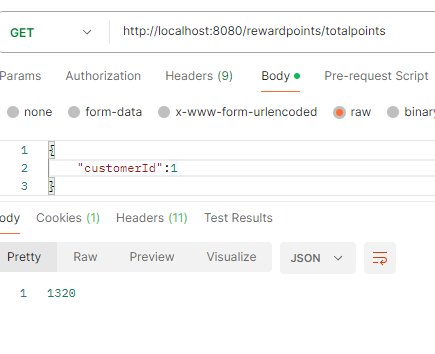
Input fields – transaction id



**8. /rewardpoints/totalpoints**

Provides total reward points of a customer

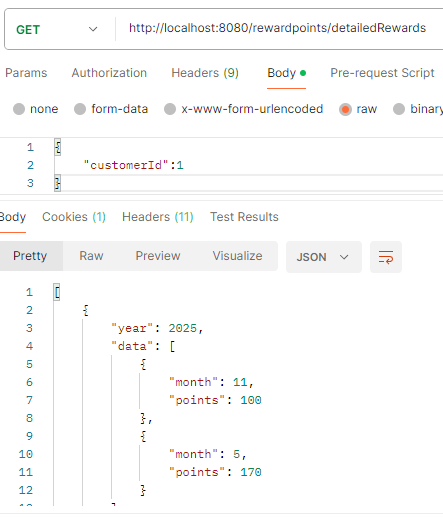
Input fields – Customer id



**9. /rewardpoints/detailedRewards**

Gives detailed (year and month wise) reward points data for a customer

Input fields – Customer id



[{"year":2025,"data":[{"month":11,"points":100},{"month":5,"points":170}]},{"year":2000,"data":[{"month":1,"points":1050}]}]

1. **Testing**

Used JUnit and Mockito for unit tests