Rewards Program API Documentation

**Index Page**

1. [Introduction](#introduction)
2. Architectural Components
3. Key Components
4. Rewards Calculation Logic
5. [High-Level Design (HLD)](#high-level-design-hld)
   * [System Architecture](#system-architecture)
   * [Components](#components)
   * [Database Design](#database-design)
6. [API Endpoints](#api-endpoints)
   * [Customer Endpoints](#customer-endpoints)
     + [Register Customer](#register-customer)
   * [Reward Endpoints](#reward-endpoints)
     + [Calculate Reward Points](#calculate-reward-points)
     + [Get Reward Report](#get-reward-report)
7. [Error Handling](#error-handling)
8. Security

**1. Introduction**

The Rewards Program Web API calculates reward points for customers based on their transaction amounts. Points are calculated based on a rewards system, where customers earn points for transactions over $50, with additional points for amounts over $100. The system provides APIs to retrieve total and monthly rewards for individual and multiple customers. The service is built using Spring Boot and adheres to SOLID principles.

**2. Architectural Components**

* **Spring Boot**: A Java-based framework used to build microservices with dependency injection, aspect-oriented programming, and REST APIs.
* **Spring Data JPA**: Used for interacting with the database and performing CRUD operations on customer transactions.
* **Spring Security**: Handles security aspects like CSRF, basic authentication (if enabled), and API authorization.
* **H2 Database**: An in-memory database used to store customer and transaction details.

**3. Key Components**

* **Customer Model**: Represents a customer in the system.
  + Fields: id, name, email, password
* **CustomerTransaction Model**: Represents a transaction made by a customer.
  + Fields: id, customerId, date, amount
* **RewardCalculator**: Contains the logic to calculate total and monthly reward points based on transactions.
* **RewardService**: The service that handles the business logic of fetching customer transactions and invoking the reward calculation.
* **Controller Layer**: Exposes APIs for customers and rewards calculation.
* **Repository Layer**: Interacts with the database to retrieve customer and transaction data.

**4. Rewards Calculation Logic**

The reward points are calculated as follows:

* For every dollar spent over $50, the customer earns 1 point.
* For every dollar spent over $100, the customer earns 2 points.

Example:

* Transaction of $120:
  + Points = (120 - 100) \* 2 + (100 - 50) \* 1 = 40 points.

5. High-Level Architecture Diagram

**High-Level Design (HLD)**

Reward  
calculator

Customer

Customer  
Transaction

- getMonthlyRewards  
- getTotalRewards

- getRewardPointsForCustomers

-id  
-CustomerId  
-date  
-amount

Id   
name   
email

Reward Service  
Logic Layer

Customer APIs  
REST Controller

Transaction APIs

**APIs Documentation**

**1. Get Total and Monthly Rewards for a Customer**

* **Endpoint**: /api/customers/rewards/{customerId}
* **Method**: GET
* **Description**: Fetches total reward points and monthly reward points for a specific customer based on their transaction history.
* **Request Parameters**:
  + customerId (Path Variable): The ID of the customer whose rewards are being requested.
* **Response**:

Json

{

"totalPoints": 150,

"monthlyPoints": {

"JANUARY": 50,

"FEBRUARY": 100

}

}

* **Error Responses**:
  + 404 Not Found: If the customer or their transactions are not found.

**2. Get Rewards for Multiple Customers**

* **Endpoint**: /api/customers/rewards/batch
* **Method**: GET
* **Description**: Retrieves total and monthly reward points for multiple customers.
* **Request Parameters**:
  + customerIds (Query Parameter): A list of customer IDs for whom the rewards are to be calculated (e.g., /api/customers/rewards/batch?customerIds=1,2,3).
* **Response**:

json

Copy code

{

"1": {

"totalPoints": 150,

"monthlyPoints": {

"JANUARY": 50,

"FEBRUARY": 100

}

},

"2": {

"totalPoints": 90,

"monthlyPoints": {

"MARCH": 90

}

}

}

* **Error Responses**:
  + 404 Not Found: If one or more customers are not found.
  + 400 Bad Request: If invalid customer IDs are passed.

**3. Get All Customers**

* **Endpoint**: /api/customers
* **Method**: GET
* **Description**: Fetches all customers present in the database.
* **Response**:

json

[

{

"id": 1,

"name": "John Doe",

"email": "john@example.com"

},

{

"id": 2,

"name": "Jane Doe",

"email": "jane@example.com"

}

]

**Error Handling**

* **404 Not Found**: This error is returned if the customer or their transactions are not found in the database.
* **400 Bad Request**: This error is returned when invalid parameters (e.g., empty or malformed customer IDs) are provided in the API requests.

**Security**

* **CSRF Protection**: Disabled for this API (for demonstration purposes, should be enabled in production).
* **Authentication and Authorization**: While authentication is not enabled in this version, it can be easily integrated using Spring Security.