Reward Points Calculating System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Version no*** | ***Prepared by*** | ***Date*** | ***Reviewed & Approved by*** | ***Date*** |
| *1* | *Deep Aatekar* | 13-Jan-2025 |  |  |

## Table of Contents

1. [Introduction](#_1._Introduction)  
2. [System Overview](#_2._System_Overview)  
3. [Architecture Diagram](#_3._Architecture_Diagram)  
4. [Module Descriptions](#_4._Module_Descriptions)  
5. [Data Flow](#_5._Data_Flow)  
6. [Database Design](#_6._Database_Design)  
7[. API Specifications](#_7._API_Specifications)  
8. [Security](#_8._Security)  
9. [Non-Functional Requirements](#_9._Non-Functional_Requirements)  
10. [Assumptions and Constraints](#_10._Assumptions_and)

## 1. Introduction

### Objective

The Reward Points and Transaction Management System enables customers to earn reward points based on their transactions. It allows customers to view monthly and total reward points, while admins can manage transactions.

### Scope

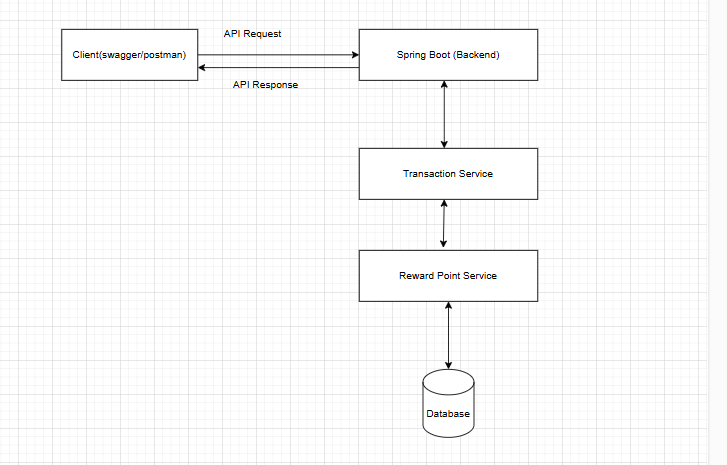
1. Customer registration and login.  
2. Transaction management.  
3. Reward points calculation and retrieval.  
4. Zone-based customer grouping.

## 2. System Overview

The system consists of the following:  
- Customer Management: Handles customer registration, login, and zone-based grouping.  
- Transaction Management: Allows adding, editing, deleting, and viewing transactions.  
- Reward Points Management: Calculates and stores total and monthly reward points.  
- Zone Management: Adds and retrieves zones.

## 3. Architecture Diagram

Include a simple architecture diagram showing the following components:  
- Frontend: Swagger/Postman for testing.  
- Backend: Spring Boot application.  
- Database: PostgreSQL for data storage.



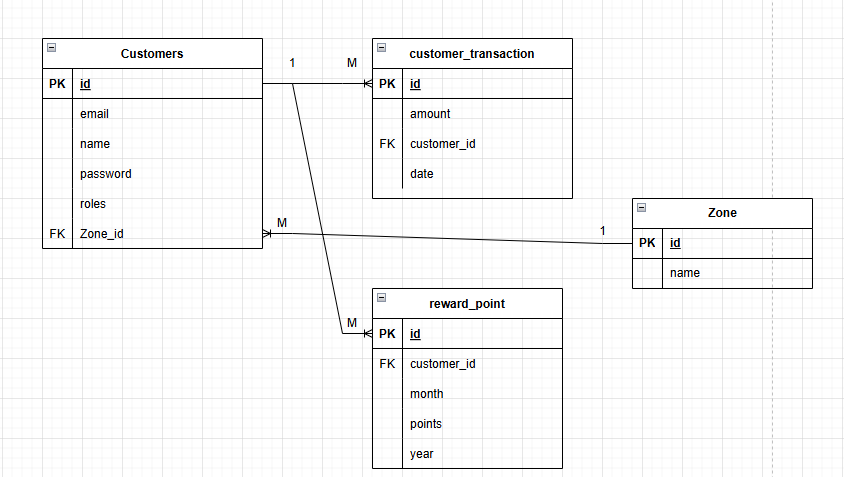
## 4. Module Descriptions

1. Customer Module and Responsibilities:  
- Register and login customers.  
- Group customers by zones.  
Key APIs:  
- POST /customers/register: Registers a new customer.  
- POST /customers/login: Logs in a customer.  
  
2. Transaction Module Responsibilities:  
- Manage transactions.  
- Calculate and save reward points.  
Key APIs:  
- POST /transactions/add: Adds a transaction and calculates reward points.  
- GET /transactions/{id}: Retrieves a specific transaction.  
  
 3. Reward Points Module Responsibilities:  
- Maintain reward point records.  
- Provide total and monthly reward points for customers.  
Key APIs:  
- GET /transactions/rewards/{customerId}: Retrieves total reward points for a customer.  
  
 4. Zone Module Responsibilities:  
- Manage zones and associate customers with zones.  
Key APIs:  
- POST /zones/add: Adds a new zone.  
- GET /zones/{id}: Retrieves zone details.

## 5. Data Flow

## 

## 6. Database Design



## 7. API Specifications

|  |  |  |  |
| --- | --- | --- | --- |
| **API Endpoint** | **HTTP Method** | **Description** | **Authentication** |
| /customers/register | POST | Register a new customer | No |
| /customers/login | POST | Login a customer | No |
| /transactions/add | POST | Add a transaction | Yes |
| /transactions/edit/{id} | PUT | Edit a transaction | Admin Only |
| /transactions/delete/{id} | DELETE | Delete a transaction | Admin Only |
| /zones/add | POST | Add a new zone | No |

## 8. Security

- Authentication:  
 - Basic Authentication.  
 - Role-based (ADMIN, USER).  
- Authorization:  
 - Role-specific access to APIs.  
- Data Security:  
 - Passwords are hashed using BCryptPasswordEncoder.

## 9. Non-Functional Requirements

- Performance:  
 - System should handle multiple transactions per second.  
- Scalability:  
 - Support future integration with web  
- Reliability:  
 - Data consistency ensured during reward calculation.

## 10. Assumptions and Constraints

- Customers have unique email addresses.  
- Zones are predefined and managed separately.