Loyalty Points Management System

# 1. Overview

This Spring Boot application calculates and manages loyalty points for users based on their spending in Hotels, Casinos, and Restaurants.  
It integrates with legacy data stored in PostgreSQL tables and offers a structured REST API to retrieve, add, update, and delete loyalty points.  
This project is built using Java 8, Spring Boot, REST API, JPA, PostgreSQL, and tested using JUnit with Mockito.  
**Note**: Here it will read the legacy data as well from respected tables from DB and add it into loyalty table.

# 2. Technology Stack

- Java 8  
- Spring Boot  
- Spring Data JPA  
- REST APIs  
- PostgreSQL  
- JUnit & Mockito for Unit Testing

# 3. Project Structure

- controller: REST API layer  
- service: Business logic layer  
- repository: Database access layer  
- entity: JPA entities mapped to DB  
- dto: Request/response objects  
- test: Unit tests using Mockito

# 4. API Endpoints

## 4.1 Get Loyalty Summary by User

Endpoint: GET /api/loyalty/summary/{userId}

Description: Returns total points grouped by Hotel, Casino, and Restaurant for a user.

Sample Output:

{  
 "userName": "Yogesh",  
 "hotelPoints": 10000,  
 "casinoPoints": 20000,  
 "restaurantPoints": 2000,  
 "totalPoints": 32000  
}

## 4.2 Get Detailed Loyalty Points

Endpoint: GET /api/loyalty/details/{userId}

Description: Returns breakdown of points by source (Taj, Marriott, Olive, etc.) for a user.

Sample Output:

{  
 "userName": "Yogesh",  
 "hotelDetails": [  
 {"sourceName": "Taj", "points": 3000},  
 {"sourceName": "Marriott", "points": 7000}  
 ],  
 "casinoDetails": [  
 {"sourceName": "Bellagio", "points": 20000}  
 ],  
 "restaurantDetails": [  
 {"sourceName": "Olive", "points": 2000}  
 ]  
}

## 4.3 Add Loyalty Points

Endpoint: POST /api/loyalty/add

Description: Add points for a user based on amount spent and type of service.

Sample Input:

{  
 "userId": 1,  
 "type": "HOTEL",  
 "sourceName": "Hyatt",  
 "billAmount": 5000  
}

Response: 200 OK - Points added successfully.

## 4.4 Update Loyalty Points

Endpoint: PUT /api/loyalty/update/{id}

Description: Manually update the number of points for a record.

Sample Input:

{  
 "points": 8000  
}

Response: 200 OK - Points updated.

## 4.5 Delete Loyalty Points

Endpoint: DELETE /api/loyalty/delete/{id}

Description: Delete a loyalty record by ID.

Response: 200 OK - Points deleted.

**Summery**:

When a user makes a purchase or visit to a hotel, restaurant, or casino, the client app calls the POST /api/loyalty/add endpoint. The backend calculates points dynamically using a service layer and persists them in PostgreSQL. We use DTOs and service layers to isolate business logic from controller code. For data analysis or frontend display, APIs like /summary and /details provide both total and breakdown views. It’s a cleanly structured, testable, extensible system.