### **1. Domain Model:**

#### **Entities:**

* **Customer**:
  + Attributes: customerId, customerName, email, phone
  + Relationships: One-to-many with Transaction, one-to-many with RewardPoints
* **Transaction**:
  + Attributes: transactionId, customer, transactionDate, amount, description
  + Relationships: Many-to-one with Customer, one-to-many with RewardPoints
* **RewardPoints**:
  + Attributes: rewardId, customer, transaction, points, earnedDate
  + Relationships: Many-to-one with Customer, many-to-one with Transaction

#### **Repositories:**

* **CustomerRepository**: Handles CRUD operations for Customer entities.
* **TransactionRepository**: Handles CRUD operations for Transaction entities.
* **RewardPointsRepository**: Handles CRUD operations for RewardPoints entities and includes custom queries to fetch CustomerRewardPoints.

### **2. DTOs (Data Transfer Objects):**

* **TransactionDTO**: Used for transferring transaction-related data to/from the service layer.
* **TransactionRequestDTO**: DTO for creating new transactions.
* **TransactionResponseDTO**: DTO for returning transaction details and reward points earned after saving a transaction.

### **3. Services:**

* **CustomerService**: Manages operations related to customers, such as saving customers.
* **TransactionService**: Handles transaction-related operations, including saving transactions, calculating reward points, and managing transaction entities.
* **RewardPointsService**: Contains business logic for calculating reward points based on transaction amounts and saving RewardPoints entities.

### **4. Controllers :**

* **CustomerController**: Exposes endpoints to perform CRUD operations on customers.
* **TransactionController**: Provides endpoints for handling transactions, including creating new transactions, retrieving transaction details, and deleting transactions.
* **RewardPointsController**: Exposes endpoints to fetch customer reward points based on custom queries.

### **5. Database Schema:**

* **Tables**:
  + customer: Stores customer information.
  + transactions: Stores transaction details, including references to customer.
  + reward\_points: Stores reward points earned per transaction, referencing both customer and transactions.

### **6. Business Logic:**

* **Calculating Reward Points**: Implemented in RewardPointsService, calculates points based on transaction amounts and saves them to the database.
* **Transactional Operations**: Utilizes Spring @Transactional for managing transaction boundaries and ensuring data consistency.