**Personal Finance Management System**

**Overview:**

Develop a Spring Boot REST API for managing personal finances, allowing users to track their income, expenses, and budget goals. Implement a one-to-many relationship between users and transactions, enabling users to view their transaction history and financial summaries.

**Functional Requirements:**

**User Entity:**

Define a User entity with the following attributes:

* id: int (user ID)
* username: String (username of the user)
* email: String (email address of the user)

Implement a one-to-many mapping with the Transaction entity.

**Transaction Entity:**

Create a Transaction entity with the following attributes:

* id: int (transaction ID)
* type: String (income or expense)
* amount: double (monetary amount of the transaction)
* description: String (description of the transaction)

Implement a many-to-one mapping with the User entity.

**Controller, Service, and Repository:**

* Controller: Develop a Controller class named "FinanceController" to handle API requests related to users and transactions.
* Service: Implement a Service class named "FinanceService" to handle business logic for managing user finances.
* Repository: Create Repository interfaces named "UserRepository" and "TransactionRepository" to interact with the database using Spring Data JPA.

**Endpoints:**

Create RESTful API endpoints for managing user finances and transactions.

**1.User Endpoints:**

* POST - "/users" --> Create a new user account.
* GET - "/users/{id}" --> Retrieve details of a specific user by ID, including transaction history.
* PUT - "/users/{id}" --> Update details of a specific user by ID.
* DELETE - "/users/{id}" --> Delete a user account by ID.

**2.Transaction Endpoints:**

* POST - "/transactions" --> Create a new transaction for a specific user.
* GET - "/transactions/{id}" --> Retrieve details of a specific transaction by ID.
* GET - "/transactions/user/{userId}" --> Retrieve all transactions associated with a specific user.
* PUT - "/transactions/{id}" --> Update details of a specific transaction by ID.
* DELETE - "/transactions/{id}" --> Delete a specific transaction by ID.

**Additional Notes:**

* **I**mplement one-to-many mapping between the User and Transaction entities to establish the relationship.
* Ensure cascading behavior for save and delete operations to maintain data consistency.
* Configure database connection details in the application.properties file.
* API Endpoint: localhost:8080