**Full API Documentation**

This section contains all the endpoints for your microservices. The base URLs are assumed to be http://localhost:8080 (User), http://localhost:8080 (Account), http://localhost:8082/86 (Transaction), and http://localhost:8090 (Ledger).

**1. User Service Endpoints**

These are the core endpoints for user management.

**🟢 Create a New User**

* **Method:** POST
* **URL:** http://localhost:8080/users
* **Description:** Creates a new user entry in the database.
* **Request Body:**

JSON

{

"name": "Jane Doe",

"email": "jane.doe@example.com"

}

**🔵 Get User by ID**

* **Method:** GET
* **URL:** http://localhost:8080/users/{id}
* **Description:** Retrieves a single user by their ID.

**🟠 Update an Existing User**

* **Method:** PUT
* **URL:** http://localhost:8080/users/{id}
* **Description:** Updates the details of a user.
* **Request Body:**

JSON

{

"name": "Jane Doe Jr.",

"email": "jane.doejr@example.com"

}

**🔴 Delete a User**

* **Method:** DELETE
* **URL:** http://localhost:8080/users/{id}
* **Description:** Deletes a user from the database.

**2. Account Service Endpoints**

These endpoints are for managing accounts and initiating transactions.

**🟢 Create an Account for a User**

* **Method:** POST
* **URL:** http://localhost:8080/accounts/user/{userId}
* **Description:** Creates a new account linked to a specific user.
* **Request Body:**

JSON

{

"accountType":"savings",

"balance": 0.00

}

**🔵 Get Account by ID**

* **Method:** GET
* **URL:** http://localhost:8080/accounts/{id}
* **Description:** Retrieves a single account by its ID.

**🔵 Get Accounts by User ID**

* **Method:** GET
* **URL:** http://localhost:8080/accounts/user/{userId}
* **Description:** Retrieves all accounts associated with a user.

**🟢 Handle a Transaction**

* **Method:** POST
* **URL:** http://localhost:8080/accounts/{accountId}/transact?amount={amount}&type={type}
* **Description:** This is the primary endpoint for a user to perform a transaction (deposit or withdrawal).

**3. Transaction Service Endpoints**

The TransactionController provides an internal API for other services and a public API for viewing transaction history.

**🟢 Process a Transaction (Internal API)**

* **Method:** POST
* **URL:** http://localhost:8082/transactions/process
* **Description:** **This is an internal API that should only be called by the Account Service.**
* **Request Body:**

JSON

{

"accNo": 1234,

"userId": 5678,

"userName": "Jane Doe",

"userBalance": 1000.00,

"type": "deposit",

"amount": 500.00

}

**🔵 Get Transactions by User ID**

* **Method:** GET
* **URL:** http://localhost:8082/transactions/user/{userId}
* **Description:** Retrieves all transaction history for a specific user.

**4. Ledger Service Endpoints**

The LedgerController is responsible for ledger entries and provides several access functions.

**🟢 Add a Ledger Entry (Manual)**

* **Method:** POST
* **URL:** http://localhost:8090/ledger/add
* **Description:** Manually adds a ledger entry.
* **Request Body:**

JSON

{

"accNo": 1234,

"userId": 5678,

"userName": "Jane Doe",

"userBalance": 1000.00,

"type": "deposit",

"amount": 500.00,

"timestamp": "2025-09-24T12:00:00.000+00:00"

}

**🔵 Get Ledger by User ID**

* **Method:** GET
* **URL:** http://localhost:8090/ledger/user/{userId}
* **Description:** Retrieves all ledger entries for a specific user.

**🔵 Get All Ledgers**

* **Method:** GET
* **URL:** http://localhost:8090/ledger/all
* **Description:** Retrieves all ledger entries in the database.

**Full Process Workflow Example**

Here is a step-by-step example of how to execute a full transaction from the client's perspective.

1. **Create a User:** POST http://localhost:8080/users Body: {"name": "Jane Doe", "email": "jane.doe@example.com"} *The response will give you the userId.*
2. **Create an Account:** POST http://localhost:8080/accounts/user/{userId} Body: {"balance": 1000.00} *The response will give you the accountId.*
3. **Perform a Deposit:** POST http://localhost:8080/accounts/{accountId}/transact?amount=500.00&type=deposit *The AccountService will call the TransactionService, and the balance will be updated.*
4. **Verify the Transaction:** GET http://localhost:8082/transactions/user/{userId} *This will show you the transaction record in the Transaction Service's database.*