Banking System Simulator (Spring Boot Monolith with MongoDB)

Overview

In this project, We are working on a Banking Services Application — a Spring Boot–based monolithic backend system that simulates real-world banking operations.
The goal is to transform your Core Java Banking System into a Spring Boot Monolithic Application that exposes RESTful APIs for account management and transactions, and implementing using RESTful APIs to perform banking operations, manage data persistence with MongoDB, handle exception

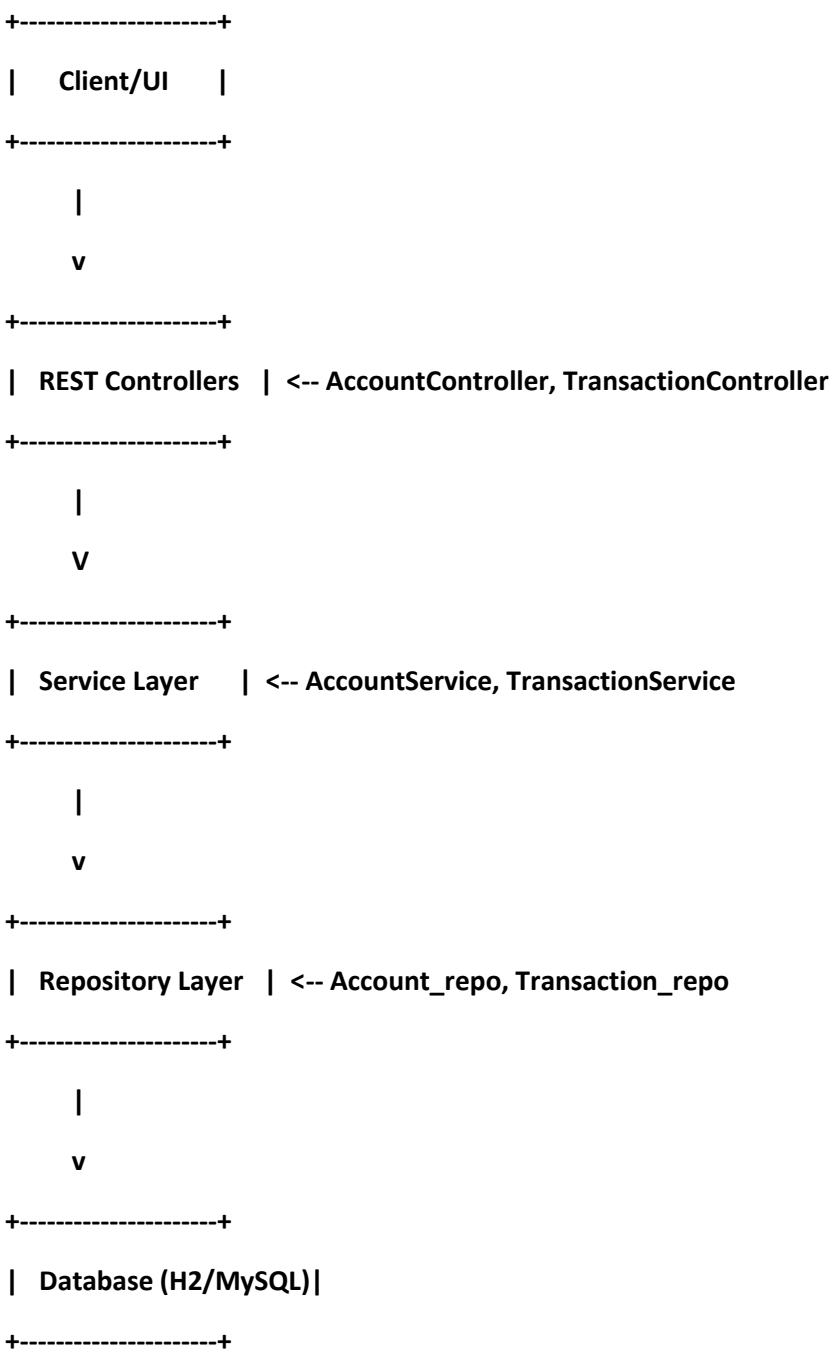
Project Overview

This project simulates a basic banking system where users can:

- Create, view, update, and delete bank accounts
- Perform transactions (deposit, withdraw, transfer)
- Track account balances and transaction history

It demonstrates Spring Boot dependency injection**,** MONGOdb repositories**,** and REST API design.

Architecture



Layer Explanation:

- **Controller Layer**: Handles incoming HTTP requests and sends responses.
- **Service Layer**: Contains business logic for banking operations.
- **Repository Layer**: Interface with the database using Spring Data JPA.

- **Database**: Stores account and transaction data.

Technologies Used

- Java 17+
- Spring Boot 4.0
- Spring Boot mongodb
- Maven
- Lombok
- REST API

API Documentation

Base URL: http://localhost:8080/api/accounts

Endpoints

METHOD	ENDPOINT	DESCRIPTION	REQUEST BODY/PARAMS	RESPONSE
POST	/api/accounts	Create a new bank accoun	Body: { "holderName": "John Doe" }	Account object
GET	/api/accounts/{accountNumber}	Get account details	Path variable → accountNumber	Account object
PUT	/api/accounts/{accountNumber}/deposit	Deposit money into an account	Query param → amount=500	Transaction object
PUT	/api/accounts/{accountNumber}/withdraw	Withdraw money from an account	Query param → amount=200	Transaction object
POST	/api/accounts/transfer	Transfer money between accounts	Body: { "sourceAccount": "...", "destinationAccount": "...", "amount": 500 }	Transaction object
GET	/api/accounts/{accountNumber}/transactions	Get all transactions for an account	Path variable → accountNumber	List of transactions
PUT	/api/accounts/{accountNumber}	Update account holder name	Query param → holderName=newName	Updated Account object
DELETE	/api/accounts/{accountNumber}	Delete/Deactivate an account	Path variable → accountNumber	204 No Content

Setup Instructions

1. Clone the repository
2. Configure the database
3. Build the project
4. Run the application
5. Access API