Detailed Technical Design for eBanking App

This document outlines the scope and the design consideration of the project which also acts as documentation

Document version: 1.0

Document revisions

Date	Version	Description	Author
2023-03-04	0.1	Initial draft	Hishara Silva
2023-03-06	1.0	Baseline version	Hishara Silva

Definitions, Acronyms, and Abbreviations

Term	Description
API	Application Programming Interface
RESTful	Representational state transfer

Table of contents

1. Overview

The prime focus of the project is to come up with a RESTful API in order to facilitate the accounts and transactions lookup

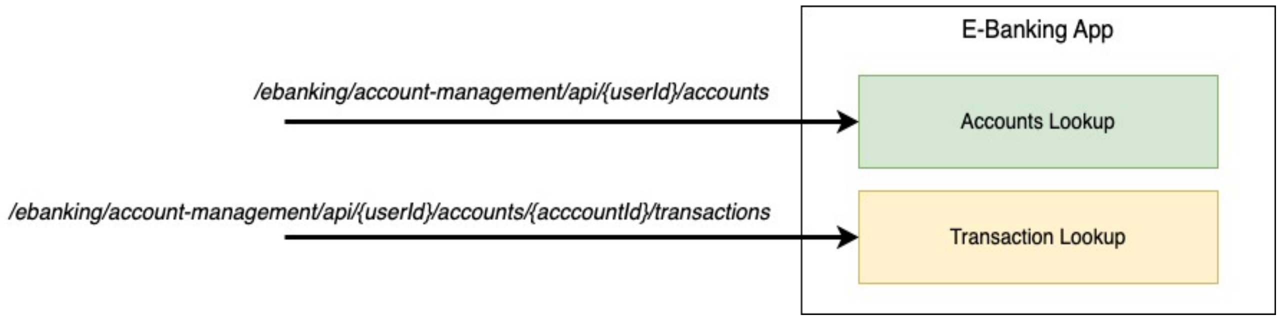
E-Banking App
/ebanking/account-management/api/{userId}/accounts
Accounts Lookup
/ebanking/account-management/api/{userId}/accounts/{acccountId}/transactions
Transaction Lookup

Image 1: API layer

Accounts Lookup	<host>/ebanking/account-management/api/{userId}/accounts
Transactions Lookup	<host>/ebanking/account-management/api/{userId}/accounts/{acccountId}/transactions

2. Target Architecture

2.1 Strategy

The following diagram illustrates the contribution to the overall architecture

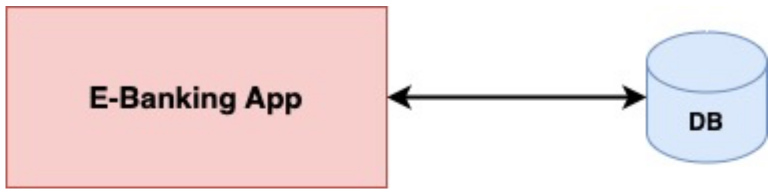

Image 2: Highlevel Overview

The proposed solution will be a simple Springboot Application comprising of two APIs. The Application will be talking to an H2 in-memory database for data persistence and retrievals

3. Design View

3.1 Flow of sequence

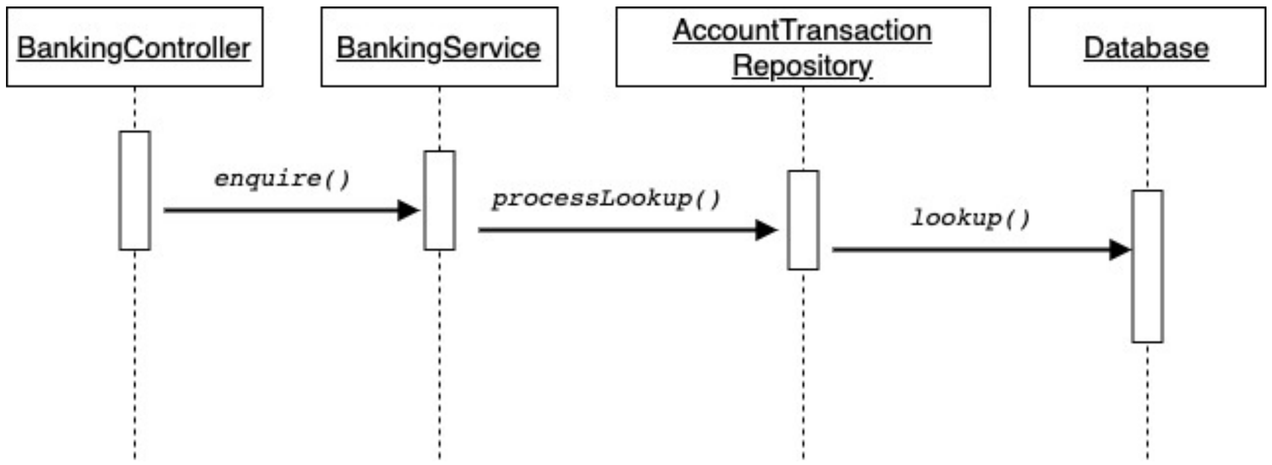

Image 3: Application flow

The controller will accept the request and flow through the service layer to the database where the data will be fetch from. Upon return, the entity will be transformed to the into the response DTOs and will be accordingly sent back to the users

3.2 Modular view

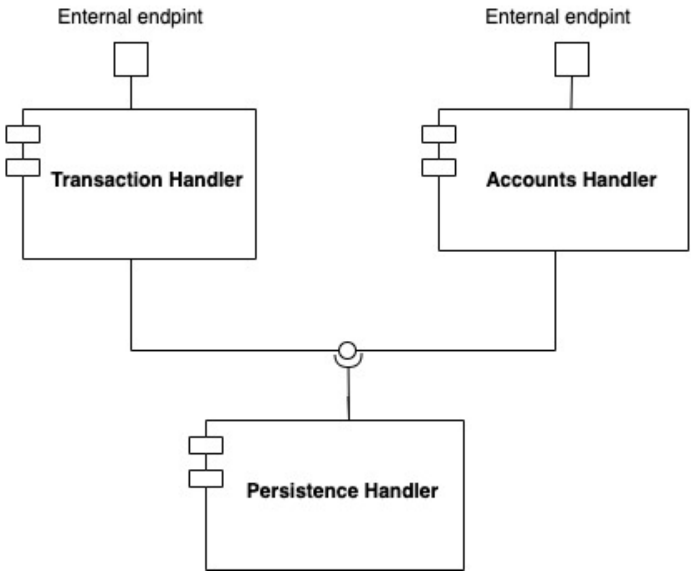

Image 4: Modularization

At high-level, the application will mainly compose two modules – accounts handler & transaction handler to facilitate the user requests. These two modules mainly interact with the persistence layer which is the main responsibly of the application

3.3 Data Store

The data store will be storing the information about accounts, transactions and users

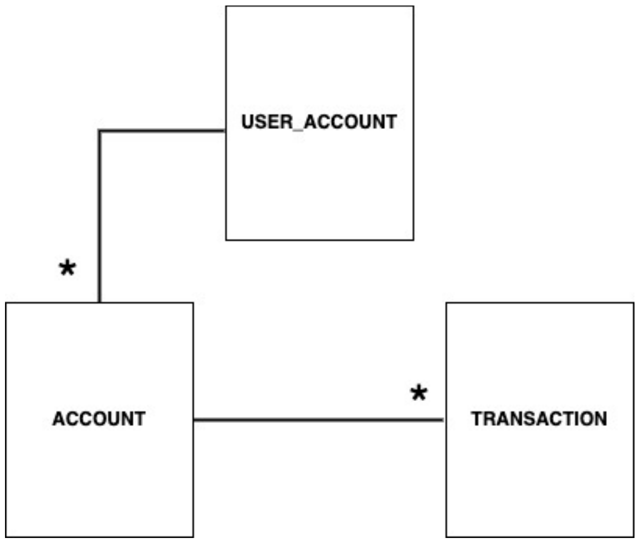

Image 5: Entity representation

3.4 API Design

The main two APIs of the application will be Accounts lookup and Transactions lookup:

API Definition:

Accounts Lookup	<host>/ebanking/account-management/api/{userId}/accounts
Transactions Lookup	<host>/ebanking/account-management/api/{userId}/accounts/{acccountId}/transactions
HTTP Verb	GET
Content type	application/json
Response codes	HTTP 200 HTTP 400 HTTP 404 HTTP 500
Error response	`{ "errorCode":"ERROR_1_001", "errorMessage":"No accounts found for user: 3000006", "timestamp":"2023-03-05T21:11:25.952645" }`

Sample Response Payloads

Accounts Lookup

```
[{
  "accountNumber":1000001,
  "accountName":"AUSavingsTest111",
  "currency":"AUD",
  "accountType":"Savings",
  "balanceDate":"2023-01-18",
  "openingAvailableBalance":55000
}]
```

Transactions Lookup

```
[{
   "accountNumber":1000003,
   "accountName":"TZSavingsTest333",
   "currency":"TZD",
   "valueDate":"2023-03-02",
   "debitAmount":"”,
   "creditAmount":150,
   "transactionType":"Credit",
   "transactionNarrative":""
}]
```

Refer the Swagger UI at: http://localhost:8080/ebanking/swagger-ui/ for more information about the API specification when the application is up and running

4. Key Architectural Goals

4.1 Technology Selection

Tool/Technology	Usage	Comments
SpringBoot	Java web application FW	
Java 11	Development language	
H2 DB	Database	In memory database is selected due to the simplicity
Docker	Containerization	

4.2 Error handling

Following error code convention will be followed for exception to easily identification of system faults

ERROR_a_00b

where;
a = component (1 – Account Handler, 2 – Transaction Handler)
b = error type, where;

‘b’ value	Description
1	Data not found
2	Payload related issue
3	System issues

Example: **ERROR_1_001 –** a data not found for Account handler

4.3 System Logging

Each message for a given request/transaction to be lodged against the correlation_id, so that all the logs for a given request/transaction can be easily filtered out in case of troubleshooting to figure out the failure point. The following format can be followed

INFO: component : operation : message
ERROR: component : operation : error_code : error message

where;

component = (AH - Transaction Handler, TH – Transaction Handler)
operation = 'Lookup'

Example:
INFO: AH | Lookup | Retrieving accounts data for user: user_id
ERROR: AH | Lookup | ERROR_1_001 | No accounts found for user: user_id

4.4 Pagination

Pagination is enabled for both the APIs, in order to utilize this feature, caller should pass the optional query parameter as follows:

Parameter	**Default value**	**Explanation**
pageNo	0	Page from which the results to the fetched
pageSize	10	No of elements per page

Usage

/ebanking/account-management/api/{userId}/accounts?pageNo=0&pageSize=10

/ebanking/account-management/api/{userId}/accounts/{acccountId}/transactions?pageNo=0&pageSize=10

4.5 Configurations

The following table will discuss some of the key configurations

Config	**Description**	**Possible/Default value**
spring.datasource.url	jdbc:h2:mem:<db_name>	N/A
username	Database username	sa
password	Database password	password

5. Deployment Strategy

The following diagram will illustrate the overall deployment of the application

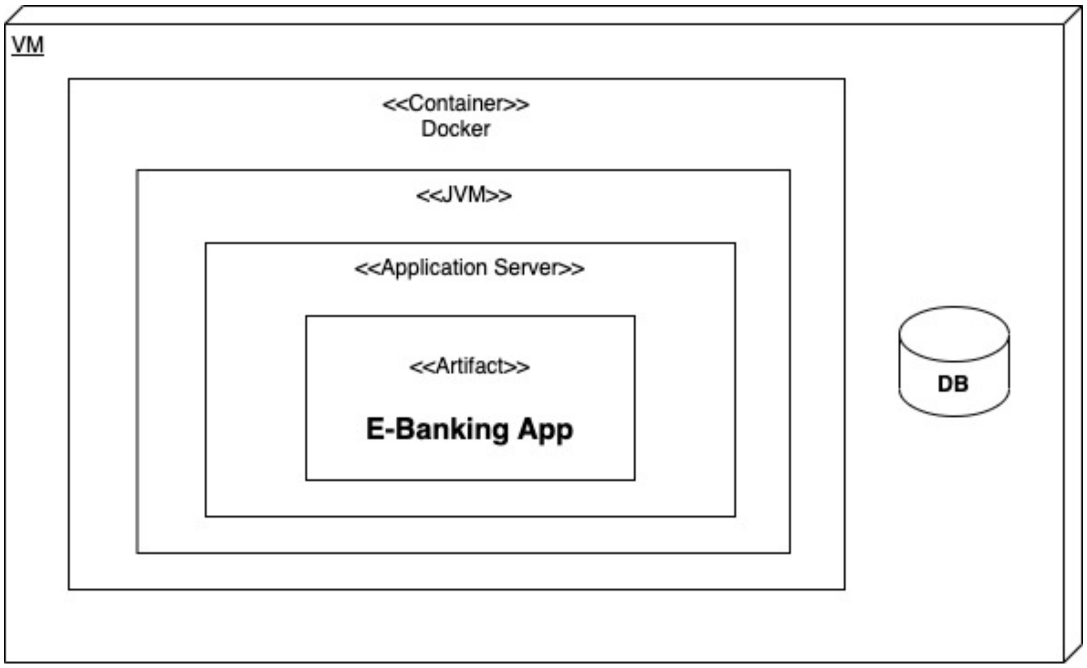

Image 6: Deployment diagram

Application will be dockerized and deployed in the VM

6. Future Enhancements

The following items are limitations in the current application which are to be enhanced in the future

6.1 Security

- API security is not implemented in the application
- user_id and account_id is passed in plain text in the path parameter in the URL which can be enhanced

6.2 Database

- The DB is kept in-memory for the simplicity, which can be enhanced to a PostgreSQL DB
- Database migration tool such as Flyway can be utilized
- AccountType and Currency can be metadata tables and 2nd level caching can be enabled

6.3 API Versioning

- API versioning can be introduced

7. User's Guide

In order to start the application in the local environment, please execute the following scripts accordingly:

Prerequisites: Java 11 installed

Navigate to the project root directory to find the following scripts

Users with Docker installed	docker_startup.sh
Users without Docker installed	startup.sh

Additional web UIs:

Component	**URL**	**Usage**
H2 console	http://localhost:8080/ebanking/h2-console/	Access database web console
Swagger UI	http://localhost:8080/ebanking/swagger-ui/	Swagger API doc

-- End of the document --