Welcome to MayBank Assessment

Table of Contents

Application Description

For stating project just run this command in root of the project directory (where pom.xml exist):

```
mvn spring-boot:run
```

If you haven't, the maven wrapper is available and you can use it instance of maven:
for linux

```
./mvnw spring-boot:run
```

for windows:

```
mvnw.cmd spring-boot:run
```

note: for **./mvnw spring-boot:run** on linux, make sure you make it executable with this command:

```
chmod +x mvnw
```

after running successfully you can open this address in browser and test APIs with using swagger ui:

```
http://localhost:8080
```

For database I use H2 (in memory) and you can access it from this address:

```
http://localhost:8080/h2-console/
```

and with this credentials:

- JDBC URL: jdbc:h2:mem:maybank
- User Name: maybank
- Password:
  - **Note:** The password must be blank;

For security handling, I used JWT and user after getting token could access resources; There is a default user (admin/admin) for get token and test APIs that you can find sign in API at this document.

This application designed by following OOP and SOLID principles, like:

- **Encapsulation**: variables are kept private and defined public accessor methods for access them.
- **Abstraction**: Created Interface and use them for tasks;
- **Inheritance**: Sharing common fields of entities in abstract classes and inherited to subclass;
- **Polymorphism**: Used in BaseEntity to help creating GenericService and AbstractServiceImpl
- **Single Responsibility**: Each service has only its responsibility; eg. saving transaction happened in TransactionService, so batch processing happened in this class too;
- **Open/Close**: TransactionServiceImpl extend AbstractServiceImpl (Open for extending) and add batch processing to it (close for modifying);
- **Liskov Substitution**: TransactionServiceImpl could replaced by AbstractServiceImpl and handle tasks of this classes;
- **Interface Segregation**: I didn't use this feature, because I didn't need it, but in repositories I used JpaRepository from Spring that cover this principle;

- **Dependency Inversion**: Controller using services interfaces and services using repository interfaces instance of their implementions.

Also, in this application I used some design pattern by using spring boot, that I mention some of them:

- **Inversion of Control** (**Dependency Injection**): Using Spring for injecting implementation for decoupling the execution of a task from its implementation. Used at: Injecting *TransactionService* and *TransactionServiceImpl.*
- **MVC**: consist of a data model, presentation information, and control information; Used for separating *Controllers*, *Services* and *Models.*
- **DAO**: Separating the data persistence logic in a separate layer; Used at *TransactionRepository*.
- **Singleton**: Defining a bean with *singleton* scope means the container creates a single instance of that bean; Used at: CustomUserDetailsService, JwtAuthenticationFilter and etc.
- **Factory Method**: Encapsulate object creation logic; Used at: creating logger;

Diagrams

Class Diagram

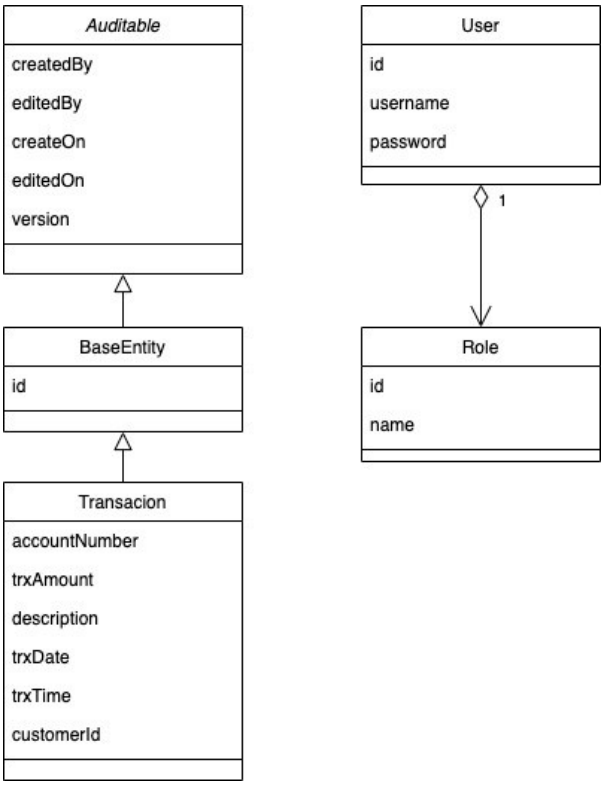

Figure 1: Class Diagram

Sign in Activity Diagram

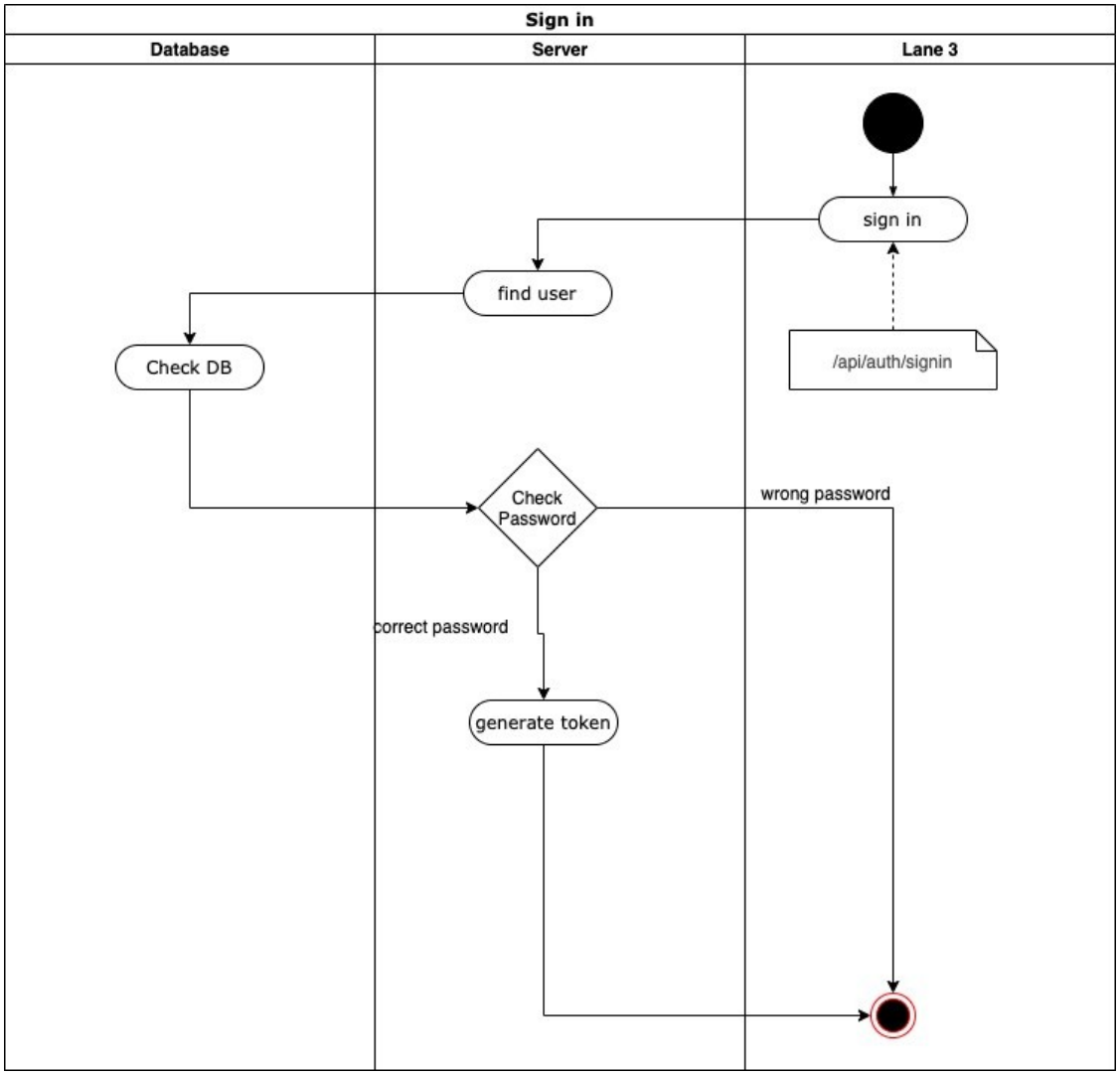

Figure 2: Sign in Activity Diagram

Sign up Activity Diagram

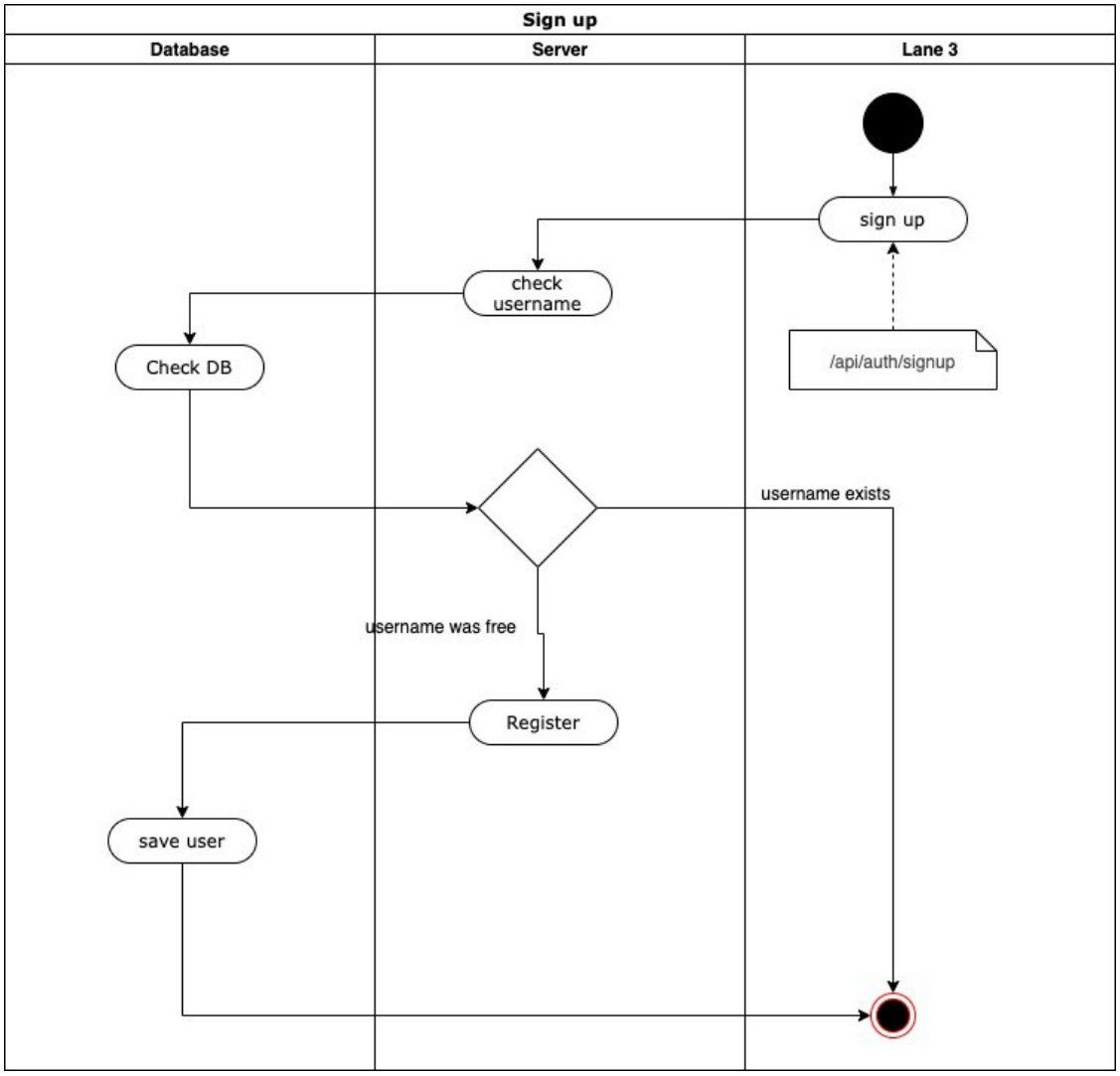

Figure 3: Sign up Activity Diagram

File Processing Activity Diagram

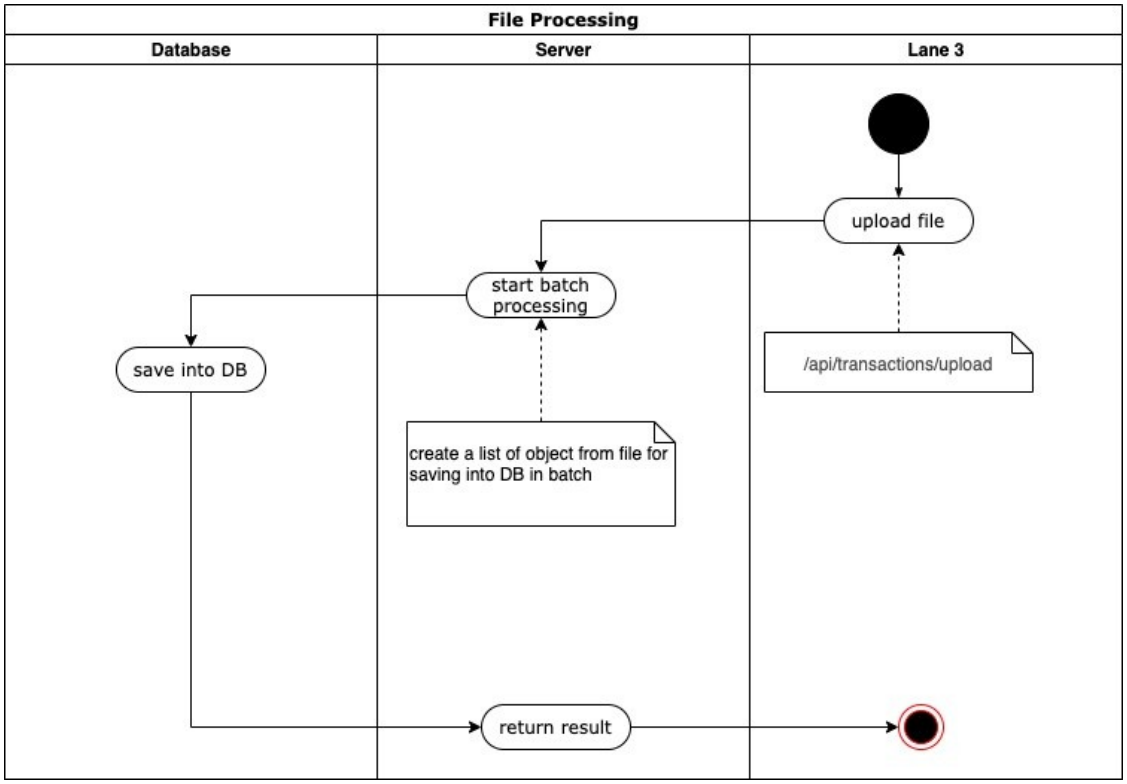

Figure 4: File Processing Activity Diagram

API Documentation

1. Sign up

Request:

```
curl --location --request POST 'localhost:8080/api/auth/signup' \
--header 'Content-Type: application/json' \
--data-raw '{
"username": "user",
"password": "user"
}'
```

Response:

Empty body with 201 header status number

2. Sign in

Request:

```
curl --location --request POST 'localhost:8080/api/auth/signin' \
--header 'Content-Type: application/json' \
--data-raw '{
"username": "admin",
"password": "admin"
}'
```

Response:

```
{
"accessToken":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2MzQ3MjE1LCJle
HAiOjE1OTY5NTIwMTV9.ow5heIjwuWmZ0RADP2_LDggHK7_UZvl9e-
XBHNGGSlCQbWhi24TtH96wCzQ0OjOqsbRGsP-MzhW_jklVAcPEng",
"tokenType": "Bearer"
}
```

Note: Token is valid for 604800000 ms (or 7 days)

3. Batch processing file

Request:

```
curl --location --request POST
'localhost:8080/api/transactions/upload' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2Mjk3MjkzLCJleH
AiOjE1OTY5MDIwOTN9.gPPJjr7rdLOy4asgn0fZgoE5GOFZftYRemiYsiIZxL92fxJ8
8LfKYeytPlh2iRz9e6A6AiINamvvY8MmyqOVmg' \
--form 'file=dataSource.txt'
```

Response:


```
{
      "createdBy": "admin",
      "editedBy": "admin",
      "createdOn": "2020-08-02T05:49:07.016+00:00",
      "editedOn": "2020-08-02T05:49:07.016+00:00",
      "version": 0,
      "id": 49,
      "accountNumber": "8872838283",
      "trxAmount": 123.00,
      "description": "FUND TRANSFER",
      "trxDate": "2019-09-12",
      "trxTime": "11:11:11",
      "customerId": 222
   },
...
   {
      "createdBy": "admin",
      "editedBy": "admin",
      "createdOn": "2020-08-02T05:49:07.032+00:00",
      "editedOn": "2020-08-02T05:49:07.032+00:00",
      "version": 0,
      "id": 95,
      "accountNumber": "6872838260",
      "trxAmount": 1923.00,
      "description": "FUND TRANSFER",
      "trxDate": "2019-09-11",
      "trxTime": "11:11:11",
      "customerId": 333
   }
]
```

4. Get Transaction Detail (Graphql)

Request:

```
curl --location --request POST 'http://localhost:8080/graphql' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2Mjk3MjkzLCJleH
AiOjE1OTY5MDIwOTN9.gPPJjr7rdLOy4asgn0fZgoE5GOFZftYRemiYsiIZxL92fxJ8
8LfKYeytPlh2iRz9e6A6AiINamvvY8MmyqOVmg' \
--header 'Content-Type: application/json' \
--data-raw '{"query":"query {\n\tgetTransactions(page: 0, size: 2)
{\n\t\tid\n accountNumber\n trxAmount\n description\n trxDate\n
trxTime\n customerId\n\t}\n}","variables":{}}'
```

Response:


```
{
"data": {
"getTransactions": [
{
"id": "1",
"accountNumber": "8872838283",
"trxAmount": 123.00,
"description": "FUND TRANSFER",
"trxDate": "2019-09-12",
"trxTime": "11:11:11",
"customerId": 222
},
{
"id": "2",
"accountNumber": "8872838283",
"trxAmount": 1123.00,
"description": "ATM WITHDRWAL",
"trxDate": "2019-09-11",
"trxTime": "11:11:11",
"customerId": 222
}
]
}
}
```

5. Get Transaction by Customer ID

Request:

**curl --location --request POST 'http://localhost:8080/graphql' **

```
--header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2Mjk3MjkzLCJleH
AiOjE1OTY5MDIwOTN9.gPPJjr7rdLOy4asgn0fZgoE5GOFZftYRemiYsiIZxL92fxJ8
8LfKYeytPlh2iRz9e6A6AiINamvvY8MmyqOVmg' \
--header 'Content-Type: application/json' \
--data-raw '{"query":"query {\n\
tgetTransactionsByCustomerId(customerId: 333, page: 0, size: 2) {\
n\t\tid\n accountNumber\n trxAmount\n description\n trxDate\n
trxTime\n customerId\n\t}\n}","variables":{}}'
```

Response:


```
{
"data": {
"getTransactionsByCustomerId": [
{
"id": "32",
"accountNumber": "6872838260",
"trxAmount": 1.00,
"description": "BILL PAYMENT",
"trxDate": "2019-09-11",
"trxTime": "11:11:11",
"customerId": 333
},
{
"id": "33",
"accountNumber": "6872838260",
"trxAmount": 1223.00,
"description": "BILL PAYMENT",
"trxDate": "2019-09-12",
"trxTime": "11:11:11",
"customerId": 333
}
]
}
}
```

6. Get Transaction by Account Number

Request:

```
curl --location --request POST 'http://localhost:8080/graphql' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2Mjk3MjkzLCJleH
AiOjE1OTY5MDIwOTN9.gPPJjr7rdLOy4asgn0fZgoE5GOFZftYRemiYsiIZxL92fxJ8
8LfKYeytPlh2iRz9e6A6AiINamvvY8MmyqOVmg' \
--header 'Content-Type: application/json' \
--data-raw '{"query":"query {\n\
tgetTransactionsByAccountNumber(accountNumber: \"6872838260\",
page: 0, size: 2) {\n\t\tid\n accountNumber\n trxAmount\n
description\n trxDate\n trxTime\n customerId\n\t}\n}","variables":
{}}'
```

Response:


```
{
"data": {
"getTransactionsByAccountNumber": [
{
"id": "32",
"accountNumber": "6872838260",
"trxAmount": 1.00,
"description": "BILL PAYMENT",
"trxDate": "2019-09-11",
"trxTime": "11:11:11",
"customerId": 333
},
{
"id": "33",
"accountNumber": "6872838260",
"trxAmount": 1223.00,
"description": "BILL PAYMENT",
"trxDate": "2019-09-12",
"trxTime": "11:11:11",
"customerId": 333
}
]
}
}
```

7. Get Transaction by Description

Request:

**curl --location --request POST 'http://localhost:8080/graphql' **

```
--header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMDAwIiwiaWF0IjoxNTk2Mjk3MjkzLCJleH
AiOjE1OTY5MDIwOTN9.gPPJjr7rdLOy4asgn0fZgoE5GOFZftYRemiYsiIZxL92fxJ8
8LfKYeytPlh2iRz9e6A6AiINamvvY8MmyqOVmg' \
--header 'Content-Type: application/json' \
--data-raw '{"query":"query {\n\
tgetTransactionsByDescription(description: \"FUND TRANSFER\", page:
0, size: 2) {\n\t\tid\n accountNumber\n trxAmount\n description\n
trxDate\n trxTime\n customerId\n\t}\n}","variables":{}}'
```

Response:


```
{
"data": {
"getTransactionsByDescription": [
{
"id": "1",
"accountNumber": "8872838283",
"trxAmount": 123.00,
"description": "FUND TRANSFER",
"trxDate": "2019-09-12",
"trxTime": "11:11:11",
"customerId": 222
},
{
"id": "3",
"accountNumber": "8872838283",
"trxAmount": 1223.00,
"description": "FUND TRANSFER",
"trxDate": "2019-10-11",
"trxTime": "11:11:11",
"customerId": 222
}
]
}
}
```