**Technical Specification Document**

Project Name : MBB-SPRING-BOOT-CRUD

Revision : Jan 10th, 2023

Author : Mohd Hanif Ab Rahman

**Overview**

MBB-SPRING-BOOT-CRUD is a simple banking system which serves as Backend and Database which then translated into Rest API for front-facing application usage.

Build Platform : Java, Spring Boot, Mongodb, Maven, RestAPI

Repository : https://github.com/hanep/mbb-springboot-crud

**Product Requirement**

1. Customer Management – Create, Read and Update Customer.
2. Account Management – Create, Read and Update Account.
3. Cash Management – Create, Read and Update Deposit/Withdrawal.

**Local Setup**

1. Clone the folder from <https://github.com/hanep/mbb-springboot-crud>.
2. Perform the following command as below in your terminal.

mvn spring-boot:run

This will run Spring Boot application with Maven command

1. You can use Postman to check on API queries and requests.

**Product Endpoints**

**Customer Endpoints (/customer)**

1. **GET /customer** - Get list of customers

**Example Output:**

[

{

"id": "63bc340e0bce2044d3129585",

"name": "Hanif"

},

{

"id": "63bc34adee39af2eb0a0f901",

"name": "Muzaffar"

},

{

"id": "63bceb4e586810316488dd03",

"name": "Noah"

}

]

1. **POST /customer** - Create new customer

**Example Input:**

{

"name" : "Hanif"

}

**Example Output:**

{

"id": "63bceb4e586810316488dd03",

"name": "Hanif"

}

Note : The id is auto generated by MongoDb upon creating a new collection

1. **GET /customer/{id}** - Get specific customer information based in its Id

**Example Output**

Using Id in 63bceb4e586810316488dd03 as {id} in URL params:

{

"id": "63bceb4e586810316488dd03",

"name": "Hanif"

}

**Account Endpoints (/account)**

1. **GET /account** - Get list of accounts

**Example Output:**

[

{

"id": "63bc3547ee39af2eb0a0f902",

"accountNumber": "1312312",

"accountType": "Saving",

"accountStatus": "active",

"accountCustomerId": "1231241451"

},

{

"id": "63bcef0d586810316488dd04",

"accountNumber": "1312382",

"accountType": "Current",

"accountStatus": "closed",

"accountCustomerId": "1231241451"

}

]

1. **POST /account** - Create a new account into collection

**Example Input:**

{

"id": "63bc3547ee39af2eb0a0f902",

"accountNumber": "1312312",

"accountType": "Saving",

"accountStatus": "active",

"accountCustomerId": "1231241451"

}

**Example Output**

Status code 200 successful creation:

{

"id": "63bcefd2586810316488dd05",

"accountNumber": "1312312",

"accountType": "Saving",

"accountStatus": "active",

"accountCustomerId": "1231241451"

}

**Cash Endpoints (/cash)**

1. **GET /cash** - Get list of cash transactions either withdrawals or deposits.

**Example Output:**

[

{

"id": "63bc34650bce2044d3129586",

"transactionType": "deposit",

"transactionAmount": 100,

"accountId": "1231231241"

},

{

"id": "63bc34650bce2044d3129586",

"transactionType": "deposit",

"transactionAmount": 1000,

"accountId": "1231231241"

},

{

"id": "63bc34660bce2044d3129587",

"transactionType": "deposit",

"transactionAmount": 1000,

"accountId": "1231231241"

},

{

"id": "63bc34670bce2044d3129588",

"transactionType": "deposit",

"transactionAmount": 1000,

"accountId": "1231231241"

},

{

"id": "63bc347eee39af2eb0a0f8fd",

"transactionType": "deposit",

"transactionAmount": 1000,

"accountId": "1231231241"

},

{

"id": "63bc348fee39af2eb0a0f8fe",

"transactionType": "withdraw",

"transactionAmount": 1000,

"accountId": "1231231241"

},

{

"id": "63bc3496ee39af2eb0a0f8ff",

"transactionType": "withdraw",

"transactionAmount": 1000.0,

"accountId": "1231231241"

},

{

"id": "63bc349bee39af2eb0a0f900",

"transactionType": "withdraw",

"transactionAmount": 1000.08,

"accountId": "1231231241"

},

{

"id": "63bc368bee39af2eb0a0f903",

"transactionType": " withdraw ",

"transactionAmount": 10,

"accountId": "1231231241"

},

{

"id": "63bc3690ee39af2eb0a0f904",

"transactionType": " withdraw ",

"transactionAmount": 10.000001,

"accountId": "1231231241"

},

{

"id": "63bc3696ee39af2eb0a0f905",

"transactionType": "1312312",

"transactionAmount": 10.000001203103121,

"accountId": "1231231241"

},

{

"id": "63bc381947723f4f8ed107b9",

"transactionType": "1312312",

"transactionAmount": 10.000001203103120311,

"accountId": "1231231241"

},

{

"id": "63bc39a5586810316488dd00",

"transactionType": "1312312",

"transactionAmount": 10.000001203103120311,

"accountId": "active"

},

{

"id": "63bc39ac586810316488dd01",

"transactionType": "1312312",

"transactionAmount": 10.000001203103120311,

"accountId": "closed"

}

]

1. **POST /cash** - Create a new cash transaction into collection

**Example Input:**

{

"transactionType": "withdraw",

"transactionAmount": 10.000001,

"accountId": "123456789"

}

**Example Output** (Status code 200 successful creation):

{

"id": "63bcf137586810316488dd07",

"transactionType": "withdraw",

"transactionAmount": 10.000001,

"accountId": "123456789"

}

**Deployment and Infrastructure**

The deployment process and infrastructure setup is housed with in the CDK under the ‘infrastructure’ folder.

For now this infrastructure section is still under development.