**Banking Application**

**Name**: Ankit Chavan

**Project**: Banking Application API

Things we were learned …

**Project Details:**

The project aim was to develop RestFul web services in Spring Boot.

* Spring Boot
* ORM
* Database

Register customer

Create account for customer

Transaction -Deposit/Withdraw

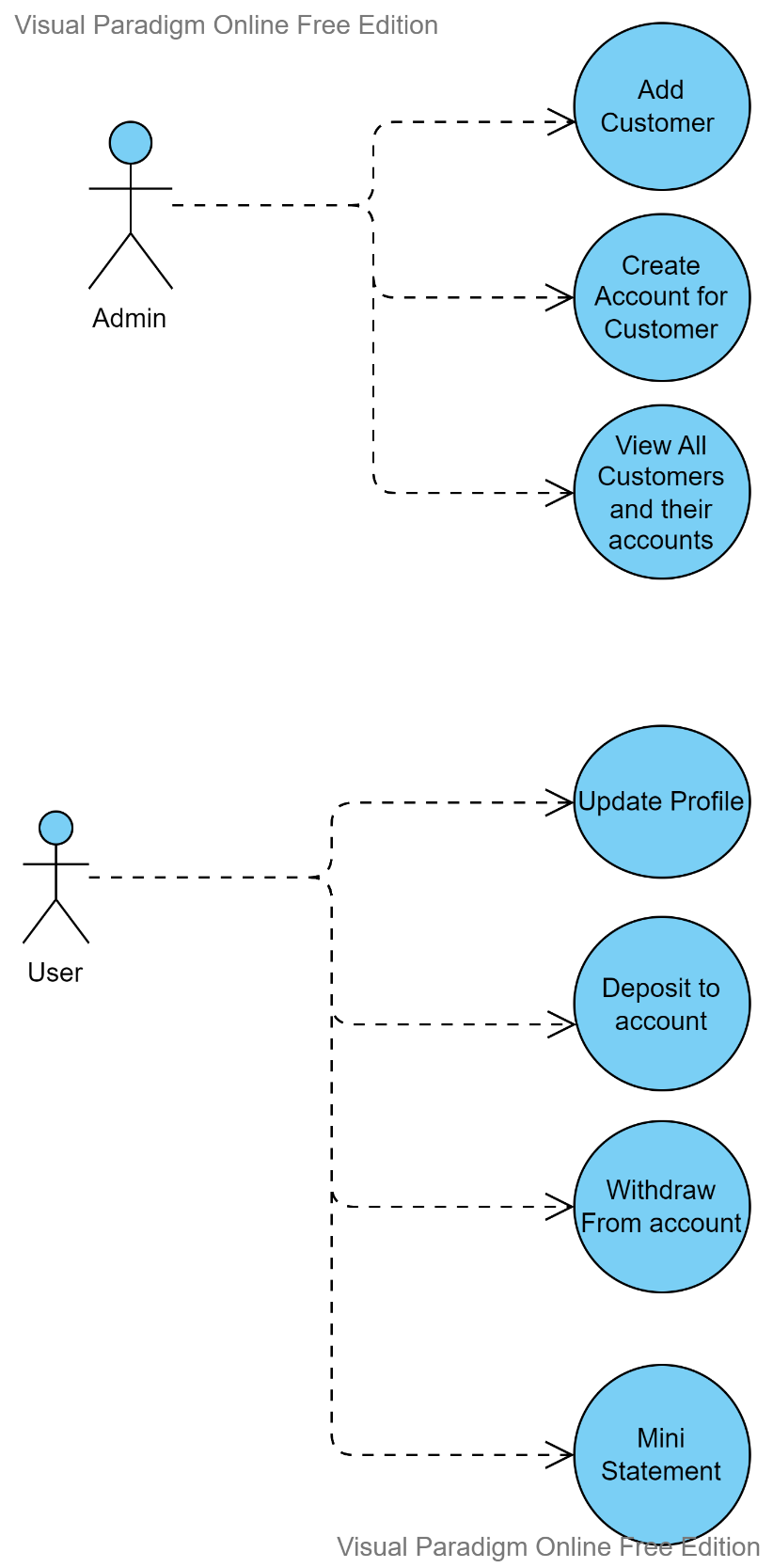
Get transaction details about account

Basic Flow:

-First create Customer with required fields

-

**Use Case Diagram :**

****

**Table Description:**

**Customer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Null Allowed** | **Key** | **Default** | **Extra** |
| customer\_id | Int | No | Primary Key | - | Autoincrement |
| first\_name | varchar(255) | Yes | - | - |  |
| last\_name | varchar(255) | Yes | - | - |  |
| Password | varchar(255) | Yes | - | - |  |

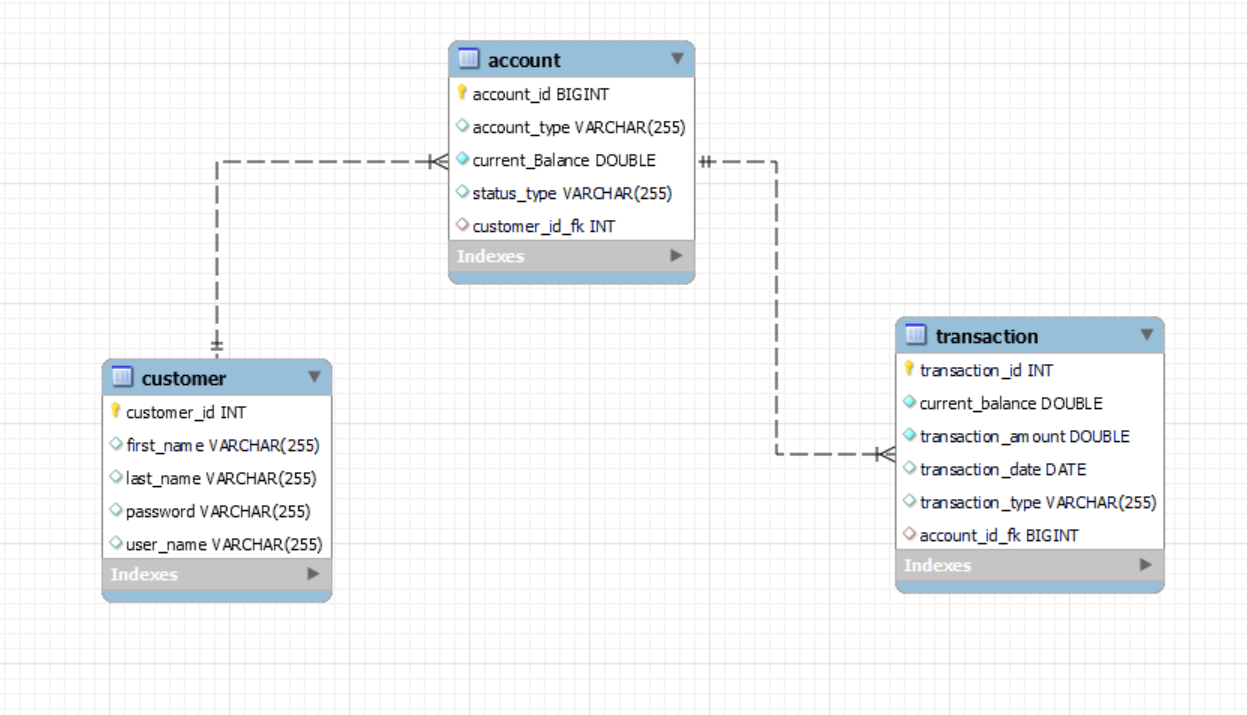
**Account**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Null Allowed** | **Key** | **Default** | **Extra** |
| account\_id | Bigint | No | Primary Key | - | Autoincrement |
| account\_type | varchar(255) | Yes |  | - |  |
| current\_blance | Double | No |  | - |  |
| status\_type | varchar(255) | Yes |  | - |  |
| customer\_id\_fk | Int | Yes | Foreign Key | - |  |

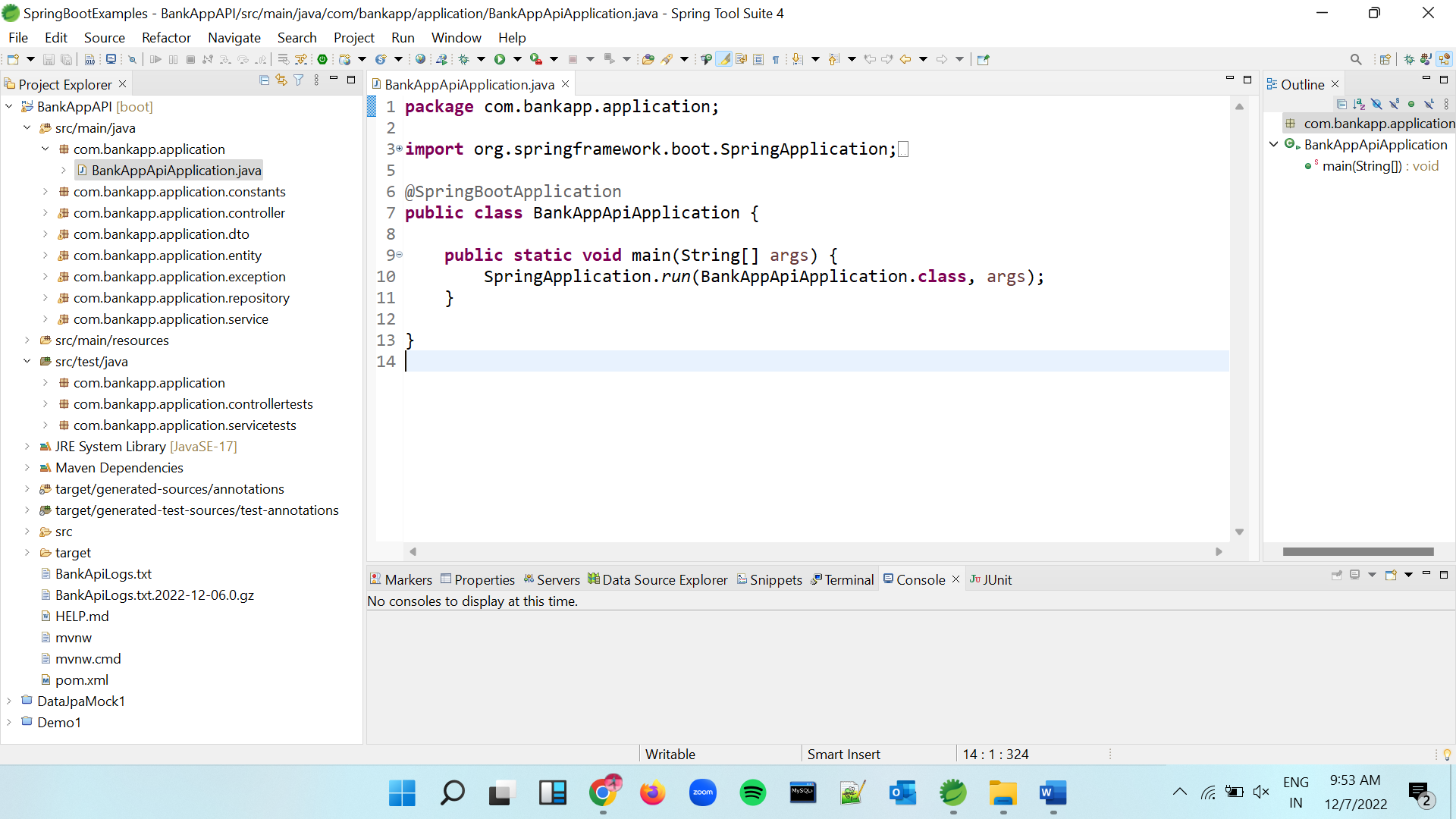
**Transaction**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Field** | | **Data Type** | | **Null Allowed** | **Key** | **Default** | | **Extra** |
| transaction\_id | Int | | No | | Primary Key | - | Autoincrement | |
| current\_balance | Double | | No | |  | - |  | |
| transaction\_amount | Double | | No | |  | - |  | |
| transaction\_date | Date | | Yes | |  | - |  | |
| transaction\_type | varchar(255) | | Yes | |  | - |  | |
| account\_id\_fk | Bigint | | Yes | | Foreign Key | - |  | |

**Entity Relationship Diagram:**



**Project Structure:**



**API Endpoints**

**Customer Controller:**

|  |  |  |
| --- | --- | --- |
| Method | API URL | Description |
| GET | api/customer/getAll | Returns all customers |
| GET | api/customer/get/{customerId} | Returns Customer for given Id |
| GET | api/customer/login | Process login for given login credentials |
| PUT | api/customer/update | Updates given Customer |
| POST | api/customer/post | Adds given Customer to DB |
| DELETE | api/customer/delete/{customerId} | Delete Customer from DB for given cutomer Id |

**Account Controller:**

|  |  |  |
| --- | --- | --- |
| Method | API URL | Description |
| GET | api/account/getAll | Get all Accounts available |
| GET | api/account/get/{accountId} | Get Account for specified account Id |
| GET | api/account/getDto/{accountId} | Get mapped Dto object |
| GET | api/account/getDtoWithoutTransactions/{accountId} | Get Account object without Transactions associated with it |
| GET | api/account/withdraw/{amount}/{accountId} | Use to withdraw from specified account Id  (Creates transaction for that account) |
| PUT | api/account/update | Update existing Account |
| POST | api/account/post | Adds given Account to DB |
| DELETE | api/account/delete/{accountId} | Delete Account from DB for given account Id |
| POST | api/account/deposit/{amount}/{accountId} | Use to deposit into specified account Id  (Creates transaction for that account) |

**Transaction Controller:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | | API URL | Description | |
| GET | api/transaction/getAll | | | Get all available transaction |
| GET | api/transaction/get/{transactionId} | | | Get Transaction for specified Id |
| GET | api/transaction/get/{accountId} | | | Get Transactions for given account Id |
| GET | api/transaction/getWithdraw/{accountId} | | | Get withdraw transactions for given accountId |
| GET | api/transaction/getDeposit/{accountId} | | | Get deposit transactions for given accountId |
| PUT | api/transaction/update | | | Update existing transaction |
| POST | api/transaction/post | | | Adds given Transaction to DB |
| DELETE | api/transaction/delete/{transactionId} | | | Delete transaction with transaction Id |

How to do basic operations:

1. First create simple customer object and hit URL <http://localhost:8091/api/customer/post> with following object with validations

{

"firstName": "Ankit",

 "lastName": "Chavan",

 "password": "Ankit123",

 "userName": [Ankit@gmail.com](mailto:Ankit@gmail.com)

}

1. You will get customer ID now you can create Account for that customer it can be either savings account or current account

{

    "customerId": 10,

    "firstName": "Akash",

    "lastName": "Jadhao",

    "password": "Akash123",

    "userName": "Ankit@gmail.com",

    "accounts": []

}

For that we have to create account object or provide fields which are required to build an account and using POST hit URL

http://localhost:8091/api/account/post

Eg.

{

     "currentBalance": 80999.0,

     "accountType": "CURRENT",

     "statusType": "DEACTIVE",

     "transactions": []

}

1. Now you can view Customer with created account using customer ID

For that hit URL

http://localhost:8091/api/customer/get/ {ID}

you will get customer Object with all its account

1. Now you can make transactions with this account for that you need to provide account Id and amount to be deposited or withdrawn from account (Keep in mind about minimum balance also minimum amount to be deposited and withdrawn are different for different account)

#Minimum Balance requirments

#For Savings Account

account.savings.minbalToBeMaintained=5000

account.savings.minAmountCanBeWithdrawn=1000

#50K

account.savings.maxAmountCanBeWithdrawn=50000

account.savings.minAmountCanBeDeposited=1000

account.savings.maxAmountCanBeDeposited=50000

#For Current Account

account.current.minbalToBeMaintained=-10000

account.current.minAmountCanBeWithdrawn=100

account.current.maxAmountCanBeWithdrawn=1000000

#10 lac

account.current.minAmountCanBeDeposited=100

account.current.maxAmountCanBeDeposited=1000000

You can change the requirements in application.properties file

Now look how can we make transactions for account

**URL:** http: /localhost:8091/api/account/deposit/{amount}/{accountId}

**Test URL:** <http://localhost:8091/api/account/deposit/50/677474750>

(invalid amount as we can’t deposit which is less than requirement)

**Test URL:** <http://localhost:8091/api/account/deposit/5000/677474750>

**(**this above URL has valid amount so that it will create transaction for that particular account and associate that transaction with that account no need to explicitly add transaction)

There’s different URL for withdraw:

**Test URL:**<http://localhost:8091/api/account/withdraw/5000/677474750>

1. We can also retrieve the transaction for account Id or we can mention which type of transaction you want in URL

**URL:** [https://api/transaction/getWithdraw/{accountId}](https://api/transaction/getWithdraw/%7baccountId%7d)

It will return all transactions which are for withdraw for this account ID