Bank Balance and Dispensing System

1. Introduction

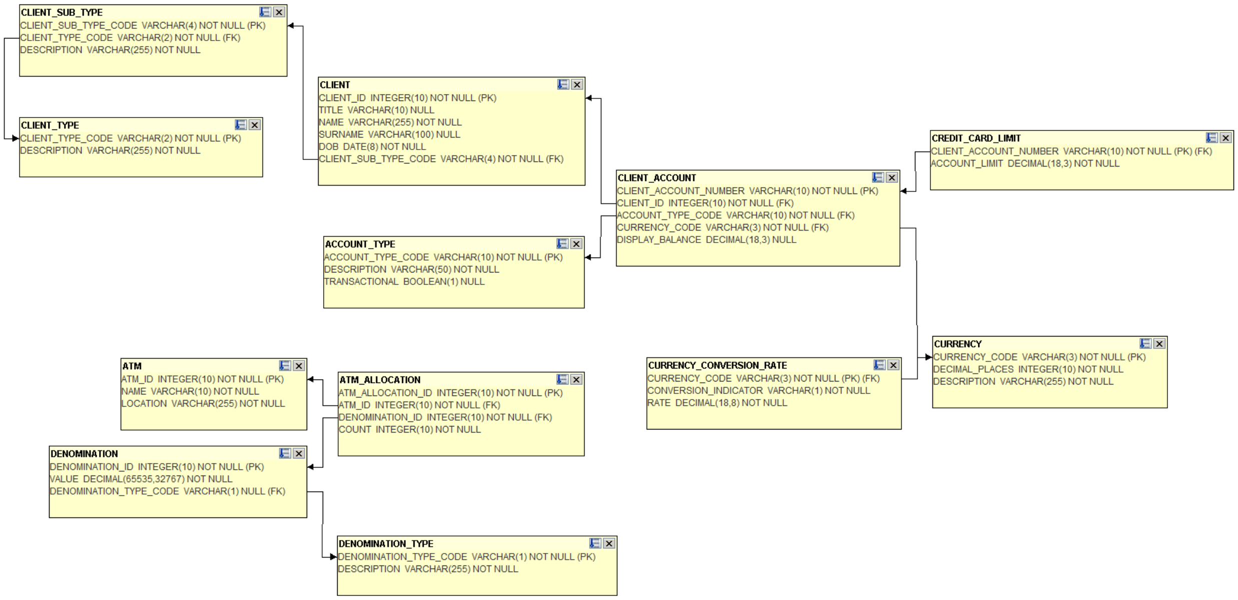
The goal of the Bank Balance and Dispensing System is to calculate and display the financial position to a client on an ATM screen. In addition, a client must also be able to make a request for a cash withdrawal.

2. Scope

The scope of the project is as follows:

In Scope	Out of Scope
Display transactional accounts with balances	Login / Authentication
Display currency accounts with converted Rand values	User interface
Withdraw (Calculation of notes to be dispensed)	Any points marked as out of scope
Reporting – Find the transactional account per client with the highest balance	
Reporting – Calculate aggregate financial position per client	
A readme document that outlines any special instructions that should be considered when reviewing the assignment. If standard ports and JDBC URLs are used and no custom tasks need to be executed, this can be omitted.	

3. ERD



CLIENT_SUB_TYPE
CLIENT_SUB_TYPE_CODE VARCHAR(4) NOT NULL (PK)
CLIENT_TYPE_CODE VARCHAR(2) NOT NULL (FK)
DESCRIPTION VARCHAR(255) NOT NULL

CLIENT_TYPE
CLIENT_TYPE_CODE VARCHAR(2) NOT NULL (PK)
DESCRIPTION VARCHAR(255) NOT NULL

CLIENT
CLIENT_ID INTEGER(10) NOT NULL (PK)
TITLE VARCHAR(10) NULL
NAME VARCHAR(255) NOT NULL
SURNAME VARCHAR(100) NULL
DOB DATE(8) NOT NULL
CLIENT_SUB_TYPE_CODE VARCHAR(4) NOT NULL (FK)

CREDIT_CARD_LIMIT
CLIENT_ACCOUNT_NUMBER VARCHAR(10) NOT NULL (PK) (FK)
ACCOUNT_LIMIT DECIMAL(18,3) NOT NULL

CLIENT_ACCOUNT
CLIENT_ACCOUNT_NUMBER VARCHAR(10) NOT NULL (PK)
CLIENT_ID INTEGER(10) NOT NULL (FK)
ACCOUNT_TYPE_CODE VARCHAR(10) NOT NULL (FK)
CURRENCY_CODE VARCHAR(3) NOT NULL (FK)
DISPLAY_BALANCE DECIMAL(18,3) NULL

ACCOUNT_TYPE
ACCOUNT_TYPE_CODE VARCHAR(10) NOT NULL (PK)
DESCRIPTION VARCHAR(50) NOT NULL
TRANSACTIONAL BOOLEAN(1) NULL

CURRENCY
CURRENCY_CODE VARCHAR(3) NOT NULL (PK)
DECIMAL_PLACES INTEGER(10) NOT NULL
DESCRIPTION VARCHAR(255) NOT NULL

ATM
ATM_ID INTEGER(10) NOT NULL (PK)
NAME VARCHAR(10) NOT NULL
LOCATION VARCHAR(255) NOT NULL

ATM_ALLOCATION
ATM_ALLOCATION_ID INTEGER(10) NOT NULL (PK)
ATM_ID INTEGER(10) NOT NULL (FK)
DENOMINATION_ID INTEGER(10) NOT NULL (FK)
COUNT INTEGER(10) NOT NULL

CURRENCY_CONVERSION_RATE
CURRENCY_CODE VARCHAR(3) NOT NULL (PK) (FK)
CONVERSION_INDICATOR VARCHAR(1) NOT NULL
RATE DECIMAL(18,8) NOT NULL

DENOMINATION
DENOMINATION_ID INTEGER(10) NOT NULL (PK)
VALUE DECIMAL(65535,32767) NOT NULL
DENOMINATION_TYPE_CODE VARCHAR(1) NULL (FK)

DENOMINATION_TYPE
DENOMINATION_TYPE_CODE VARCHAR(1) NOT NULL (PK)
DESCRIPTION VARCHAR(255) NOT NULL

4. System Requirements

Develop a spring boot application that provides REST services to produce the requested outcomes.

4.1 Dependency Management and Build Requirement

Dependency management and building of the application should be facilitated via Maven

4.2 Libraries and Frameworks

Spring boot should be used as a base for the application, the version is not important, with H2 as a data store. The scripts for the H2 database and data are provided at the end of this document.

You are free to use additional libraries but all components should build and start via Maven or a standard Java run command.

5. Requirements

5.1 Use Case Summary

5.2 Use Cases

5.2.1 Display transactional accounts with balances

Use case	Display transactional balances
Goal in context	To view available balances
Level	User
Primary Actor	Bank client at ATM terminal
Precondition	Client successfully logged on to ATM terminal with bank card
Success end condition	Client is able to view all transactional accounts with the available balances on each account in descending order with the highest balance displaying first and the lowest balance displaying last. All cheque accounts have an overdraft limit of R10000.
Basic flow of events	1. Client logs in to ATM terminal using the card and pin pad and selects the “View Transactional Balances” option (out of scope). 2. The ATM terminal sends a request to the server with the client id and a timestamp (**out of scope)** 3. The server receives and interprets the request 4. The server determines the current display balances, sorts the details and returns the response to the ATM terminal 5. The ATM terminal receives and interprets the response from the server (out of scope) 6. The ATM terminal displays the balances to the user **(out of scope)**
Extension	1. The client does not have any qualifying accounts The system displays an error message
Error Message	No accounts to display

Context Root	/discovery-atm/
REST Endpoint	/queryTransactionalBalances
Method	GET
Request Parameters	clientId

Response Body	{
	"client" : {
	"id" : ?:Long,
	"title" : ?:String,
	"name" : ?:String,
	"surname" : ?:String
	},
	"accounts" :
	[
	{
	"accountNumber" : ?: Long,
	"typeCode": ?:String,
	"accountTypeDescription" : ?:String,
	"currencyCode" : ?:String,
	"conversionRate" : ?:BigDecimal(19,3),
	"balance" : ?:BigDecimal(19,3),
	"zarBalance" : ?:BigDecimal(19,3),
	"accountLimit" : ?:BigDecimal(19,3)
	}
	]
	}

5.2.2 Display currency accounts with converted Rand values

Use case	Display currency account balances with converted Rand values
Goal in context	To view available balances of currency accounts with converted Rand values
Level	User
Primary Actor	Bank client at ATM terminal
Precondition	Client successfully logged on to ATM terminal with bank card
Success end condition	Client is able to view all transactional accounts with the available balances on each account in ascending order with the highest balance in Rand displaying first and the lowest balance displaying last
Basic flow of events	1. Client logs in to ATM terminal using the card and pin pad and selects the “View Currency Account Balances” option **(out of scope)**. 2. The ATM terminal sends a request to the server with the client id and a time stamp **(out of scope)** 3. The server determines the current display balances a. Determine current balance b. Determine conversion rate c. Determine multiplication rate for display purposes d. Calculate Rand amount e. Sort results by Rand amount f. The service returns the result to the ATM terminal 4. The ATM terminal receives and interprets the response from the server **(out of scope)** 5. The ATM terminal displays the balances to the user **(out of scope)**
Extension	1. The client does not have any qualifying accounts The system displays an error message
Error Message	No accounts to display

Context Root	/discovery-atm/
REST Endpoint	/queryCcyBalances
Method	GET
Request Parameters	clientId

Response Body	{
	"client" : {
	"id" : ?:String,
	"title" : ?:String,
	"name" : ?:String,
	"surname" : ?:String
	},
	"accounts" :
	[
	{
	"accountNumber" : ?:Long,
	"typeCode": ?:String,
	"accountTypeDescription" : ?:String,
	"currencyCode" : ?:String,
	"conversionRate" : ?:BigDecimal(19,3),
	"balance" : ?:BigDecimal(19,3),
	"zarBalance" : ?:BigDecimal(19,3),
	"accountLimit" : ?:BigDecimal(19,3)
	}
	]
	}

5.2.3 Withdraw cash

Use case	Allow client to withdraw cash from an account
Goal in context	A client is allowed to withdraw money from a transactional account if he has a positive balance or is allowed to go into a negative balance if the account is a cheque account up to a maximum of R 10 000.
Level	User
Primary Actor	Bank client at ATM terminal
Precondition	Client successfully logged on to ATM terminal with bank card and has selected a transactional account to withdraw money from
Success end condition	The client account balance has been adjusted with the withdrawal amount and the ATM has been notified of the number of notes per denomination to dispense
Basic flow of events	1. Client logs in to ATM via using the card and pin pad and selects the “Withdraw” **(out of scope)** 2. The is presented with a selection of transactional accounts and selects the appropriate account to proceed to the next screen **(out of scope)** 3. The client enters the amount required and selects submit **(out of scope)**. 4. The ATM sends a request to the server with the ATM id, client id, account number, required amount, and a time stamp **(out of scope)** 5. The server receives and interprets the request from the ATM terminal 6. The server determines the number of notes to dispense per denomination a. Determine current balance

	b. Determine if the requested amount is available for withdrawal based on balance and limit rules c. Determine if the amount specified can be provided using only bank notes (no coins) d. Determine the number of each denomination to dispense using the minimum number of notes using the available notes in the ATM e. The system adjusts the clients account balance with the amount requested f. The service returns the number of notes per denomination to dispense, to the ATM terminal 7. The ATM terminal receives and interprets the response from the server **(out of scope)** 8. The ATM dispenses the cash **(out of scope)**
Extension	*1.* The client does not have enough available funds *The system displays an error message "Insufficient funds"* *2.* The ATM does not have enough cash to dispense the required amount *The system displays an error message "Amount not available, would you like to draw X" where X is the next available amount lower than the requested amount.* 3. The system does not find the ATM in the system or does not find an allocation for the specified ATM *The system displays an error message "ATM not registered or unfunded"*
Error Message	Insufficient funds Amount not available, would you like to draw X ATM not registered or unfunded Invalid client or account number

Context Root	/discovery-atm/
REST Endpoint	/withdraw
Method	GET
Request Parameters	clientId atmId accountNumber requiredAmount

Response Body	{
	  "client" : {
	    "id" : ?:Long,
	    "title" : ?:String,
	    "name" : ?:String,
	    "surname" : ?:String
	  },
	  "account" :
	   {
	      "accountNumber" : ?:String,
	      "typeCode": ?:String,
	      "accountTypeDescription" : ?:String,
	      "currencyCode" : ?:String,
	      "conversionRate" : ?:BigDecimal(19,3),
	      "balance" : ?:BigDecimal(19,3),
	      "zarBalance" : ?:BigDecimal(19,3),
	      "accountLimit" : ?:BigDecimal(19,3)
	   }
	  "denomination" :
	  [
	    {
	      "denominationId" : ?:Long,
	      "denominationValue" : ?:BigDecimal(19,3),
	      "count" : ?:Integer
	    }
	  ]
	}

5.2.4 Reporting – Find the transactional account per client with the highest balance

Use case	Find the transactional account per client with the highest balance
Goal in context	A report needs to be generated that displays a list of all clients along with the account details of the client's account with the highest balance
Level	System
Primary Actor	System
Success end condition	Successful report generation
Basic flow of events	On a monthly basis a report is generated and emailed to support@thebank.com **(out of scope)**
Note	Create a SQL script that can be executed that can be executed against the DB that will cater for the criteria specified above and return the fields below: • Client Id • Client Surname • Client Account Number • Account Description • Display Balance

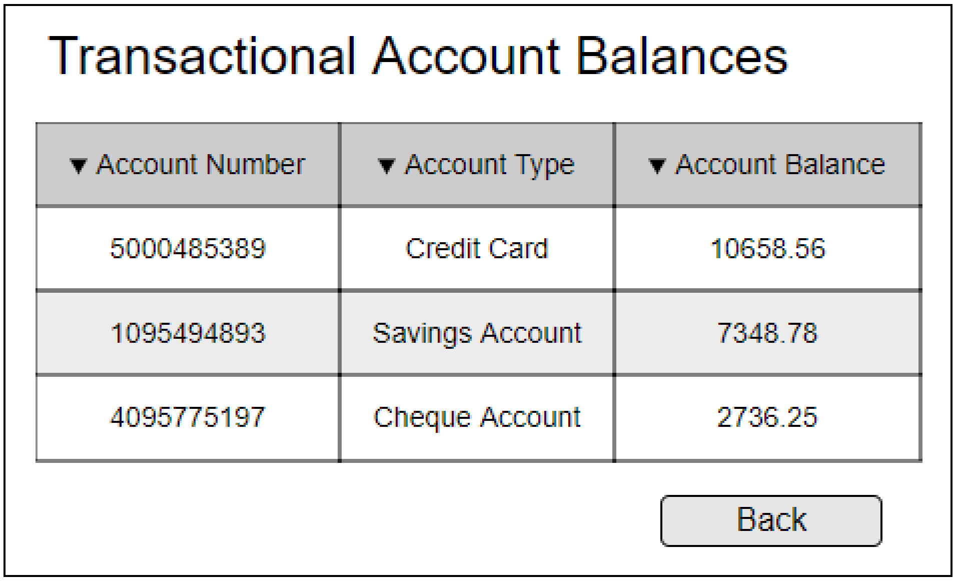
5.2.5 Reporting – Calculate aggregate financial position per client

Use case	Calculate aggregate financial position per client
Goal in context	A report needs to be generated that displays the net position per client based on all accounts held

Level	System
Primary Actor	System
Success end condition	Successful report generation
Basic flow of events	On a monthly basis a report is generated and emailed to support@thebank.com **(out of scope)**
Note	Create a SQL script that can be executed that can be executed against the DB that will cater for the criteria specified above and return the fields below: • Client (Client Title + Client Name + Client Surname) • Loan Balance (Aggregate of all loan amounts) • Transactional Balance (Aggregate of all transactional accounts) • Net Position (Net position across all accounts)

6. Screens

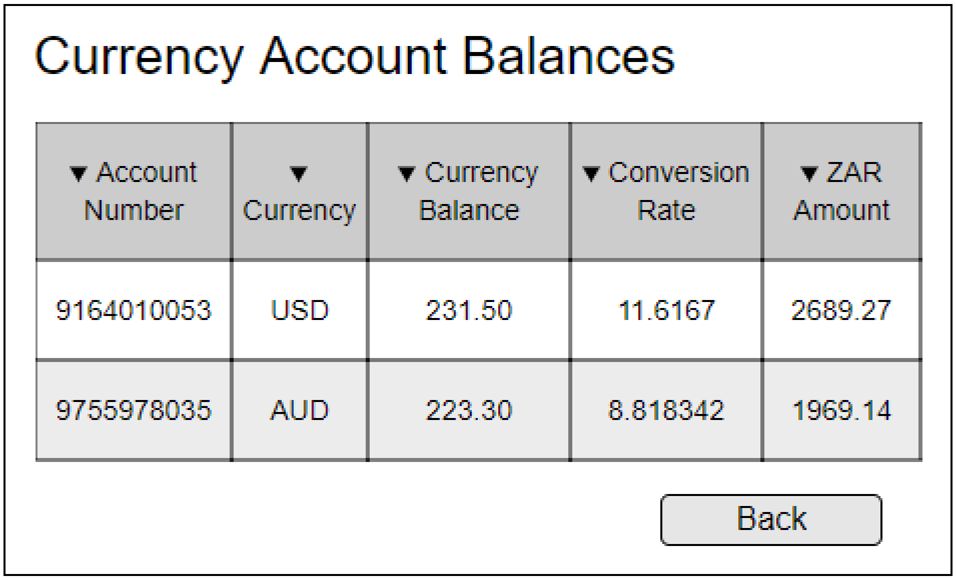
6.1.1 Display transactional accounts with balances



Transactional Account Balances

▼ Account Number	▼ Account Type	▼ Account Balance
5000485389	Credit Card	10658.56
1095494893	Savings Account	7348.78
4095775197	Cheque Account	2736.25

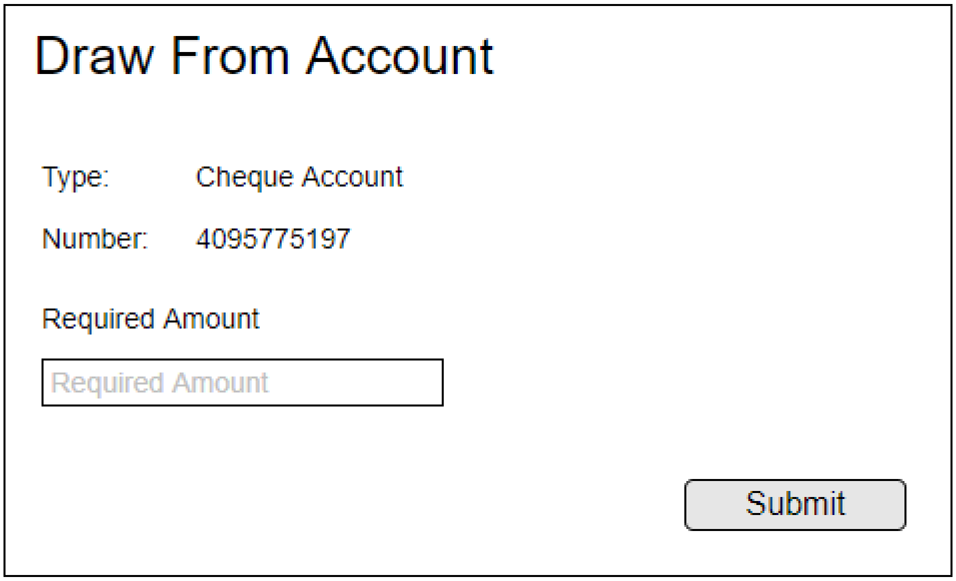
6.1.2 Display currency accounts with converted Rand values



Currency Account Balances

▼ Account Number	▼ Currency	▼ Currency Balance	▼ Conversion Rate	▼ ZAR Amount
9164010053	USD	231.50	11.6167	2689.27
9755978035	AUD	223.30	8.818342	1969.14

6.1.3 Withdraw cash



Draw From Account

Type: Cheque Account

Number: 4095775197

Required Amount

7. Additional Information

Scripts are provided for the creation of an H2 in memory DB and the service is to run as a Spring Boot application exposing a REST API.

Data is provided for reference purposes and may be enhanced at time of review.

schema.sql data.sql