

**Banking System**

**Web services and API Development**



Lecturer: Ms. Julie Power

Jun Hsin Lim 16123107

Jessica Valeria 16118677

Table of Contents

[Table of Contents 2](#_Toc480830709)

[1.0 Introduction 3](#_Toc480830710)

[1.1 Problem Statement 3](#_Toc480830711)

[1.2 Proposed Solution 3](#_Toc480830712)

[1.3 Entity Relationship Diagram 4](#_Toc480830713)

[1.4 Security Concerns 5](#_Toc480830714)

[2.0 The RESTful API 6](#_Toc480830715)

[3.0 Bibliography 8](#_Toc480830716)

# 1.0 Introduction

## 1.1 Problem Statement

Create a typical consumer online banking application that provides for a customer, an account and a transaction. The functions that should be included in the application are:

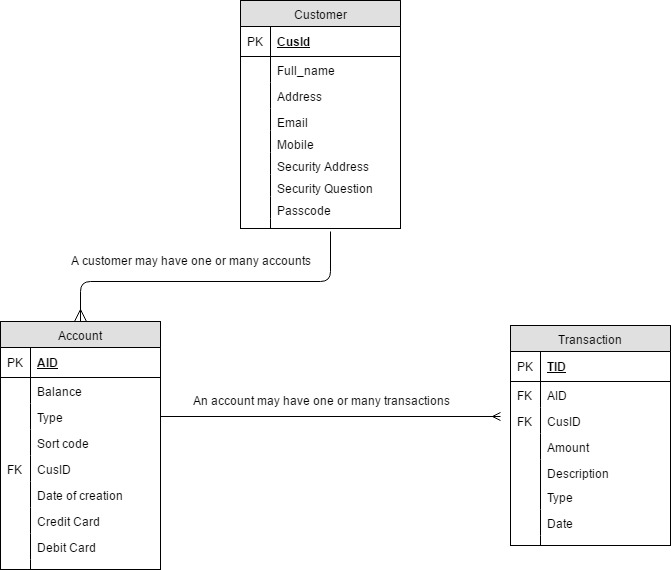
1. Customers should be able to **create** an *account* with the bank
2. Customers can specify the *amount* to **lodge** with the *account* that will be debited
3. Customers can specify the *amount* to **transfer** and the *account* to transfer from
4. Customers can specify the *amount* to **withdraw** and the *account* that will be credited
5. Customers can request to view their **balance** on any of their *account(s)* at any time

## 1.2 Proposed Solution

We will be using Netbeans to create a RESTful API to satisfy the problem statement. MySQL will be used to create the database required for the API and will be described in the Entity-Relationship diagram below. The database will be hosted in gearhost to be accessed through the cloud.

We will create a server side application that allows customers to interact with the API through GUIs.

## 1.3 Entity Relationship Diagram



**Customer –** Is a person or company with personal details, can hold one or more accounts and can make different transactions.

**Account –** holds information about type of account and balance after and before movements.

**Transaction –** describe what type of movement is the customer doing (withdraw, transfer, deposit), give us a description of the movement, the amount and date.

## 1.4 Security Concerns

As we are talking about a Banking system is a priority protect the user’s information, is very important develop a way to protect all personal information provided by users. To make this possible, this application is going to have an authentication system.

Normally the process of authentication consist of two phases:

* Identification: Which provides user identity to the security system (User ID). What the system is going to do is search through the databases to find the specific user. Once this user is located, will be able to be identified.
* Authentication is a process in which the credentials that the user provide previously are compared with a database of authorized users. If the credentials match, the user will be able to access into the system.

Once the user has been authenticated will be able to launch other process inside the system.

To make the authentication we can use different techniques such as:

**Password requirements,** this means enforcing the password with especial requirements such as:

* Minimum 6 Characters
* Must contain at least 1 lowercase letter (a-z)
* Must contain at least 1 lowercase letter (A-Z)
* Must contain at least 1 number (0-9)
* Must contain at least 1 special character (!,@,#,$,%,etc)

**Hashing passwords,** to make this possible, is necessary use bcrypt which is a platform file encryption utility based on the Blowfish symmetrick block cipher cryptographic algorithm and can be used in all supported operating systems. Passwords must have between 8 and 56 characters.

# 2.0 The RESTful API

|  |  |
| --- | --- |
| Function: | Sign In |
| Description: | Allows user to sign in to their account |
| URL: | http://localhost:8080/BankingSystem/api/customer/login |
| HTTP: | GET |
| Parameters: | email (String, required), passcode (String, required) |
| Resource contents: | Accepts users input (email and passcode) and find the user’s details from the customer table, allowing user to login successfully |
| Pre-Conditions: | User must have registered as a customer in JJ’s Bank |
| Post-Conditions: | Allows user to do transaction functionalities, create a new account, check balances |

|  |  |
| --- | --- |
| Function: | Create Customer |
| Description: | Creates a new customer |
| URL: | http://localhost:8080/BankingSystem/api/customer |
| HTTP: | POST |
| Parameters: | customer (JSON object, required) |
| Resource contents: | Boolean status: Customer has been successfully created, a new bank account would also be created along the customer; New customer could not be created |
| Pre-Conditions: | User is not a pre-existing customer in the bank |
| Post-Conditions: | Allows user access to the bank’s functionalities |

|  |  |
| --- | --- |
| Function: | Create Account |
| Description: | Allows the addition of a new account into the system referencing the customer |
| URL: | http://localhost:8080/BankingSystem/api/bankAccount |
| HTTP: | POST |
| Parameters: | account (JSON object, required) |
| Resource contents: | Boolean status, create account or unable to create account  If user selects debit card, a debit card number will be generated  If user selects credit card, a credit card number will be generated |
| Pre-Conditions: | User must have registered in JJ’s Bank |
| Post-Conditions: | User will be able to perform transactions on their accounts |

|  |  |
| --- | --- |
| Function: | Withdraw (MakeTransaction.java) |
| Description: | Allows user to take money from their account. |
| URL: | http://localhost:8080/BankingSystem/api/transaction  http://localhost:8080/BankingSystem/api/bankAccount  http://localhost:8080/BankingSystem/api/bankAccount/customerAccounts |
| HTTP: | GET, POST and PUT |
| Parameters: | GET to retrieve the list of accounts for the customer: cusId (String, required)  POST to create transaction: transaction (JSON object, required)  PUT to update account balance: account (JSON Object, required, aId (Integer, required) |
| Resource contents: | Advises the user if the action was successfully done through a Boolean and if statement |
| Pre-Conditions: | User must be registered and have an account in the bank |
| Post-Conditions: | A new transaction will be created and the selected account’s balance will be reduced |

|  |  |
| --- | --- |
| Function: | Transfer (MakeTransaction.java) |
| Description: | Allows user to send money from their account |
| URL: | http://localhost:8080/BankingSystem/api/transaction  http://localhost:8080/BankingSystem/api/bankAccount  http://localhost:8080/BankingSystem/api/bankAccount/customerAccounts |
| HTTP: | GET, POST and PUT |
| Parameters: | GET to retrieve the list of accounts for the customer: cusId (String, required)  POST to create transaction: transaction (JSON object, required)  PUT to update account balance: account (JSON Object, required, aId (Integer, required) |
| Resource contents: | Advises the user if the action was successfully done through a Boolean and if statement |
| Pre-Conditions: | User must be registered and have an account in the bank |
| Post-Conditions: | A new transaction will be created and the selected account’s balance will be reduced |

|  |  |
| --- | --- |
| Function: | Deposit (MakeTransaction.java) |
| Description: | Allows user to make a deposit |
| URL: | http://localhost:8080/BankingSystem/api/transaction  http://localhost:8080/BankingSystem/api/bankAccount  http://localhost:8080/BankingSystem/api/bankAccount/customerAccounts |
| HTTP: | GET, POST and PUT |
| Parameters: | GET to retrieve the list of accounts for the customer: cusId (String, required)  POST to create transaction: transaction (JSON object, required)  PUT to update account balance: account (JSON Object, required, aId (Integer, required) |
| Resource contents: | Advises the user if the action was successfully done through a Boolean and if statement |
| Pre-Conditions: | User must be registered and have an account in the bank |
| Post-Conditions: | A new transaction will be created and the selected account’s balance will be increased |

|  |  |
| --- | --- |
| Function: | View Balance (Home.java) |
| Description: | Allows user see the balance in each of their accounts. |
| URL: | http://localhost:8080/BankingSystem/api/bankAccount/customerAccounts |
| HTTP: | GET |
| Parameters: | cusId (String, required) |
| Resource contents: | Creates an array of accountList which contains the lists of the accounts that belongs to the user. |
| Pre-Conditions: | User must have at least an account to be able to view their account balances |
| Post-Conditions: | All the bank accounts and their balances will be displayed on the Home |

**RESTConnection.java**

Location: com.mycompany.bankingclient

Content:

1. signIn method that relates to /customer/login
2. customerAccounts method that relates to /bankAccount/customerAccounts
3. sendPostRequest method that relates to all post requests in the server
4. sendPutRequest method that relates to all put requests in the server

# 3.0 Bibliography

* SearchSecurity. (2015). What is authentication? - Definition from WhatIs.com. [online] Available at: http://searchsecurity.techtarget.com/definition/authentication [Accessed 24 Apr. 2017].
* Stack Abuse. (2015). Implementing User Authentication the Right Way. [online] Available at: http://stackabuse.com/implementing-user-authentication-the-right-way/ [Accessed 24 Apr. 2017].
* Havighurst, R. (2007). Mechanics of User Identi?catlion and Authentication. 1st ed. Taylor & Francis Group, p.5.