**Web Services and API Development**

**4th Year Final Project Document**

**Group N**

Sarah Yun Tiong – x14111187, Azeez Yusuff – x14443758  
Dawid Trojanowski – x14321651, Ben Skerritt – x14110717

# Introduction

The final project which requires us to design and develop a bank system that will have a fully functioning backend that is connected to a persistent database and a frontend client that will allow the operations through a functioning GUI. The bank system should allow for a customer to have accounts and transactions and the various interactions between these entities. To begin we will first draw up an entity diagram that will map out the architecture of the system. From the entity diagram we will design up the API entry points. The entry points will provide us with an understanding of how we should go about the development of the system.

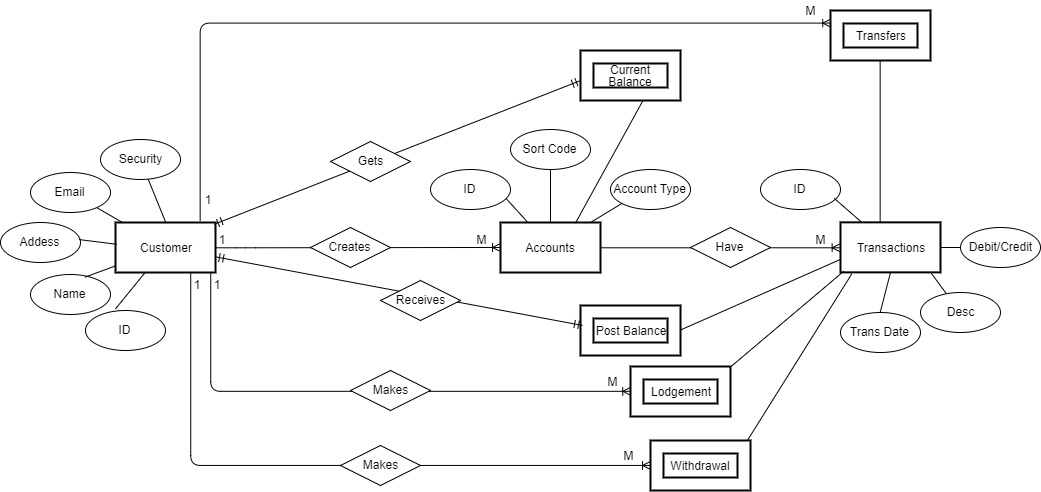
The bank system will be a RESTful web service application that will implement java JPA, xml and html files. All messages to the service will be in JSON or XML format and will use CRUD operations. Depending on the entry points, the system will use POST, GET, PUT and DELETE methods to enable the service requests connect to the client and the persistent MYSQL database.

The RESTful service must be designed in a way that is attractive and intuitive and uses action word verbs in methods when naming the entry points to the resource nouns.

Implementing a separation of concerns methodology, the code for the system will be comprised of a model, service and resource for each of the three entities. The customer class will be the model and allow the creation of a customer and will have an id, name, address, email and security credentials. Instead of a customer service and resource we will have a bank resource that will handle the POST, GET etc. methods and a bank service that will handle the logical operational functionality to the RESTful service. The accounts and transaction entities will also implement the separation of concerns and have model, service and resource.

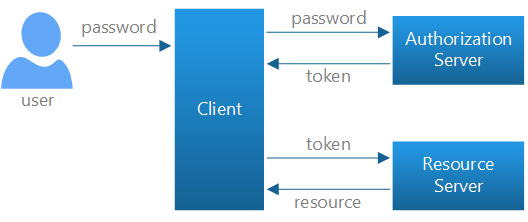
We will initially design the API addresses to navigate to five html pages that will allow for the creation of a new customer. The creation of accounts for that customer. The lodgement, withdrawal and transfer of funds. And finally the retrieval of a balance on an account. A full list of the entry points can be seen below.

# Entity-Relationship Diagram



# Bank API Security

Security is one of the most important and obvious things that is needed to be built into the web service. During the development, the security needs to be strict but at the same time the Web API needs to be easy to use. One of the ways that you can secure the accounts is to create a token-based authentication, token-based authentication is generating a random token and giving it to the user as also then allowing the user to reset this token at any point to retrieve stolen or lost account. The token will be passed through the POST method or a HTTP header. For an example, a user can send the token via SHA-1 as a POST or as a simple header like this: “Authorization key: 32vcx412416fd4213cb5643vzd3242314”, choosing this type of token is much more secure than a short numeric identifier that can be brute forced by tools. Here is an example of how it looks:



As with tokens or any other type of authentication in regards to accounts, all that needs to be saved in the database. SQL injections is one of the big things, preventing the access to the database is crucial right now, it’s really important to stop the attackers from playing around. The most common vulnerabilities that occur and make your data accessible by attacks using the SQL injection attacks include:

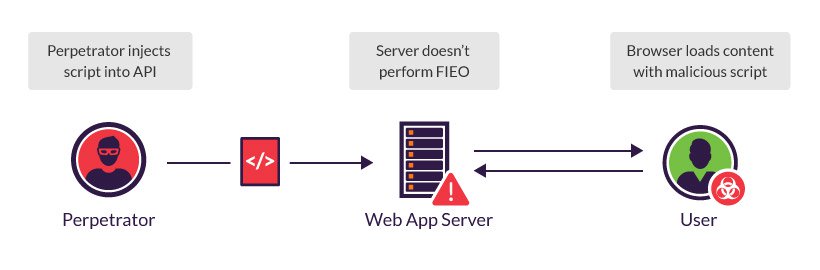
• A weak validation of the input bars, login, register or even the search bar.

• Wrong construction of the SQL statements, leaving the parameters unsafe.

• Also one of the main ones, over-privileged accounts with access to the database.

SQL injections are one of the most basic and most common attacks that occur now. It’s not difficulty to secure yourself from them but it’s not easy at the same time.

On the other hand, SQL injections are as much common as XSS also known as cross-site-scripting, it uses a script that is trying to change function of other scripts and retrieve the data from the database, cross-site-scripting is harder than SQL injections, in SQL injections you can just use some random tool and that will do the job, as for XSS you need a knowledge to do so. Securing the Web API from XSS is challenging, to secure the Web API from SQL and XSS injections, you need to secure the web API if it requires a parser or a processer since these type of web API’s are most vulnerable to attack. To protect the web API all inputs, have to be filtrated and validated correctly in order to keep the web API safe.



# API Entry Points

|  |  |
| --- | --- |
| API Name | createCustomer |
| Description | This allows a customer to be added to the database |
| URI | http://localhost:49000/api/customers/save |
| HTTP verb | POST |
| Parameters | {name} (String, required)  {address} (String, required)  {email} (String, required)  {securityCredentials} (int, required) |
| Resource contents | Customer, JSON format |
| Pre-conditions | Bank Service should be able to accept new Customer |
| Post-conditions | Customer with specific id is returned |

|  |  |
| --- | --- |
| API Name | updateCustomerDetails |
| Description | This allows a customer’s details to be updated |
| URI | http://localhost: 49000/api/customers/{id} |
| HTTP verb | PUT |
| Parameters | {id} (int, required)  {name} (String, required)  {address} (String, required)  {email} (String, required)  {securityCredentials} (int, required) |
| Resource contents | Returns customers updated details, JSON format |
| Pre-conditions | The customer must exist |
| Post-conditions | The customer details are updated. |

|  |  |
| --- | --- |
| API Name | readAllCustomers |
| Description | This allows the retrieval of all customer |
| URI | http://localhost: 49000/api/customers |
| HTTP verb | GET |
| Parameters | {Id} (int, required) |
| Resource contents | Customer, JSON format |
| Pre-conditions | The customer must exist in the system |
| Post-conditions | Returns customer details (customer id, name, address, email) |

|  |  |
| --- | --- |
| API Name | readCustomerById |
| Description | This allows the retrieval of specific customer by id |
| URI | http://localhost: 49000/api/customers/{id} |
| HTTP verb | GET |
| Parameters | {Id} (int, required) |
| Resource contents | Customer, JSON format |
| Pre-conditions | The customer must exist in the system |
| Post-conditions | Returns specific customer details (customer id, name, address, email) |

|  |  |
| --- | --- |
| API Name | deleteCustomerById |
| Description | This allows a customer to be deleted |
| URI | http://localhost: 49000/api/customers/{id} |
| HTTP verb | DELETE |
| Parameters | {id}(int , URL Param, required) |
| Resource contents | Customer details, JSON format |
| Pre-conditions | The customer must exist |
| Post-conditions | The customer is deleted |

|  |  |
| --- | --- |
| API Name | createCustomerAccount |
| Description | This creates a new account for a customer and saves it to the database |
| URI | http://localhost: 49000/api/customers/{id}/accounts/saveAccount |
| HTTP verb | POST |
| Parameters | {int id} (int, URL Param, required)  {accountType} (String, required)  {sortCode} (String, required)  {accountCurrentBalance} (double, required) |
| Resource contents | Account, JSON format |
| Pre-conditions | A customer should already exist |
| Post-conditions | Account with specific id and type is returned |

|  |  |
| --- | --- |
| API Name | updateAccountDetailsById |
| Description | This allows a customer’s account type to be updated |
| URI | http://localhost: 49000/api/customers/{id}/accounts/{accountId} |
| HTTP verb | PUT |
| Parameters | {id} (int, URL Param, required)  {int accountId} (int, URL Param, required)  {Account Type} (String, required) |
| Resource contents | Account, JSON format |
| Pre-conditions | The account must exist in the system |
| Post-conditions | Returns updated account details (account id, customer name, account type) |

|  |  |
| --- | --- |
| API Name | readAllCustomerAccountDetails |
| Description | This allows a customer to return all accounts’ details |
| URI | http://localhost:49000/api/customers/{id}/accounts |
| HTTP verb | GET |
| Parameters | {id} (int, URL Param, required)  {accountType} (String, required)  {accountSortCode} (String, required)  {accountCurrentBalance} (double, required) |
| Resource contents | Account, JSON format |
| Pre-conditions | Customer with accounts should exists |
| Post-conditions | Returns all accounts’ details (account id, customer, account type, sortcode, account balance) |

|  |  |
| --- | --- |
| API Name | readCustomerAccountDetailsByAccountId |
| Description | This allows a customer to return specific account details |
| URI | http://localhost: 49000/api/customers/{id}/accounts /{accountId} |
| HTTP verb | GET |
| Parameters | {id} (int, URL Param, required)  {int accountId} (int, URL Param, required)  {accountType} (String, required)  {accountSortCode} (String, required)  {accountCurrentBalance} (double, required) |
| Resource contents | Account, JSON format |
| Pre-conditions | Customer with accounts should exists |
| Post-conditions | Returns specific account details (account id, customer, account type, sortcode, account balance) |

|  |  |
| --- | --- |
| API Name | deleteCustomerAccountById |
| Description | This allows a customer’s account to be deleted |
| URI | http://localhost: 49000/ api/customers/{id}/accounts/{accountId} |
| HTTP verb | DELETE |
| Parameters | {id} (int, URL Param, required)  {accountId} (int, URL Param, required) |
| Resource contents | The account details, JSON format |
| Pre-conditions | The customer’s account must exist |
| Post-conditions | The customer’s account is deleted |

|  |  |
| --- | --- |
| API Name | deleteAllCustomerAccounts |
| Description | This allows a customer’s account to be deleted |
| URI | http://localhost: 49000/api/customers/{id}/accounts |
| HTTP verb | DELETE |
| Parameters | {id} (int, URL Param, required) |
| Resource contents | Account, JSON format |
| Pre-conditions | The customer’s accounts must exist |
| Post-conditions | All customer’s accounts are deleted |

|  |  |
| --- | --- |
| API Name | createALodgment |
| Description | This allows a customer to make a lodgment |
| URI | http://localhost:49000/api/customers/{id}/accounts/{accountType}/{accountId}/transactions/lodgment |
| HTTP verb | POST |
| Parameters | {customerId} (int, required)  {accountType} (String, required)  {accountId} (int, required)  {transactionType} (String, required) ( = Lodgment in this case) |
| Resource contents | Lodgment, JSON format |
| Pre-conditions | Specific customer with specific account type should exists |
| Post-conditions | Lodgment is saved to the database with a confirmation message returned |

|  |  |
| --- | --- |
| API Name | createATransfer |
| Description | This allows a customer to make a transaction between accounts |
| URI | http://localhost:49000/api/customers/{customerId}/accounts/{accountType}/{accountId}/transactions/transfer |
| HTTP verb | POST |
| Parameters | {customerId} (int, required)  {accountType} (String, required)  {accountId} (int, required)  {TransactionType} (String, required) ( = Transfer in this case) |
| Resource contents | Transfer, JSON format |
| Pre-conditions | Specific customer with at least 2 specific account types should exists |
| Post-conditions | Transfer is saved to the database with a confirmation message returned |

|  |  |
| --- | --- |
| API Name | createAWithdrawal |
| Description | This allows a customer to make a withdrawal |
| URI | http://localhost:49000/api/customers/{customerId}/accounts/{accountType}/{accountId}/transactions/withdrawal |
| HTTP verb | POST |
| Parameters | {customerId} (int, required)  {accountType} (String, required)  {accountId} (int, required)  {TransactionType} (String, required) ( = Withdrawal in this case) |
| Resource contents | Withdrawal, JSON format |
| Pre-conditions | Specific customer with specific account type should exists |
| Post-conditions | Withdrawal is saved to the database with a confirmation message returned |

|  |  |
| --- | --- |
| API Name | readAllTransactions |
| Description | This allows a customer to request all transaction details |
| URI | http://localhost:49000/api/customers/{id}/accounts/{accountType}/{accountId}/Transactions |
| HTTP verb | GET |
| Parameters | {id} (int, URL Param, required)  {accountType} (String, URL Param, required)  {accountId} (int, URL Param, required)  {transactionType} (String, required)  {transactionId} (int, required)  {transactionDateTime} (DateTime, required) |
| Resource contents | Transaction, JSON format |
| Pre-conditions | Specific customer with specific account types with transaction details should exist |
| Post-conditions | Returns all transactions with specific details from a customer account |

|  |  |
| --- | --- |
| API Name | readWithdrawalDetails |
| Description | This allows a customer to request only withdrawal transactions |
| URI | http://localhost:49000/api/customers/{id}/accounts/{accountType}/{accountId}/transactions/withdrawal/{transactionId} |
| HTTP verb | GET |
| Parameters | {id} (int, URL Param, required)  {accountType} (String, URL Param, required)  {accountId} (int, URL Param, required)  {transactionId} (int, URL Param, required)  {transactionType} (String, required) |
| Resource contents | Transaction, JSON format |
| Pre-conditions | Specific customer with specific account type with specific withdrawal transactions should exists |
| Post-conditions | Withdrawal transactions details are returned |

|  |  |
| --- | --- |
| API Name | readLodgementDetails |
| Description | This allows a customer to request only lodgment transactions |
| URI | http://localhost:49000/api/customers/{id}/accounts/{accountType}/{accountId}/transactions/lodgment/{transactionId} |
| HTTP verb | GET |
| Parameters | {id} (int, URL Param, required)  {accountType} (String, URL Param, required)  {accountId} (int, URL Param, required)  {transactionId} (int, URL Param, required)  {transactionType} (String, required) |
| Resource contents | Transaction, JSON format |
| Pre-conditions | Specific customer with specific account type with specific lodgment transactions should exists |
| Post-conditions | Lodgments transactions are returned |