WEB Services and API

Group Z – Navjot, Valentin, Soffyan and Ryan

Introduction

This purpose of this document is to define online banking problem domain and proposed solution. The document contains a description of the project, Entity Relationship Diagram and Security concerns related to the project.

An Online banking requirements are a customer should be able to open an account (or multiple) and should be able do transactions (lodgement, transfer, withdrawal) and should be able to check the current balance.

We have named the project NSVBank, and the aim of the project is to design, develop and document Web API and to develop a Client that make use of the API as to allow the customer to use Online Banking.

The Web API will allow the customer to create an account with the bank (example: current or savings or both) and allow the customer to take advantage of the online banking services like checking current balance, withdraw, transfer and lodge money.

We will be developing the project using Jersey to develop a RESTFul Web API that will seamlessly expose data from the database which our Client will leverage and do http calls like GET, POST and DELETE and provide online banking functionality.

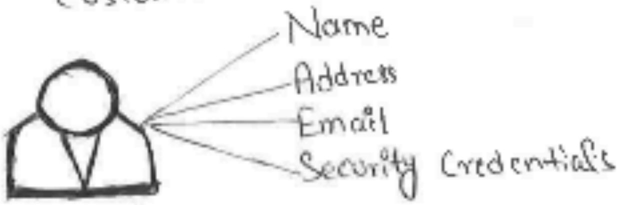
Motivation for the Project

In, today day and age "Online Banking" has become a must, people use it to buy things online, check their current balance or make transfers etc. Our project implements the functionality that a typical online banking system have these days.

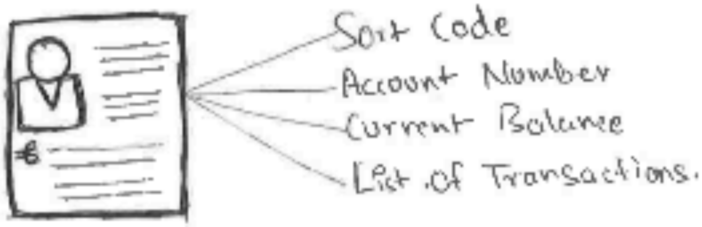
Solution Architecture Design

1.- Bank is an finincial Institution

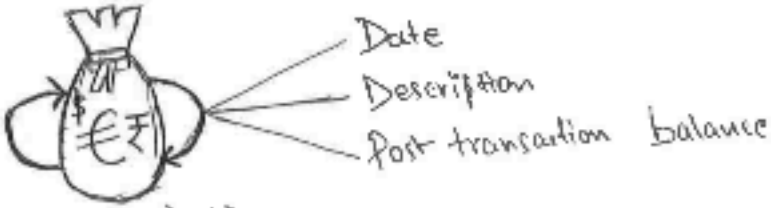
2. Bank has A Customer.



3. A Customer can have 1 or more accounts.

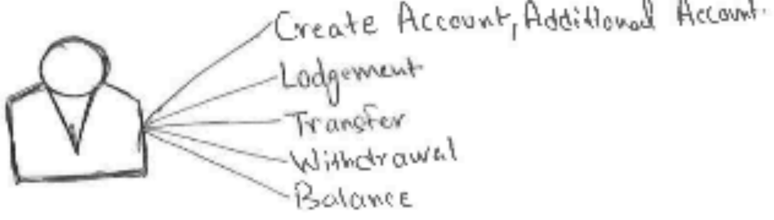


4- Customer can do a Transaction with an Account.
Credit or Debit



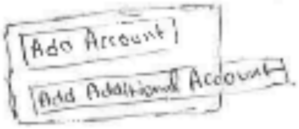
Credit or Debit

5- Customer will be able to do the following-



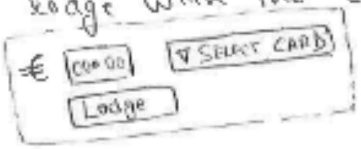
- Create

Customers Should be able to Create an account with the bank, and a customer who has an account Should be able to add additional accounts.
Ex- Current & Saving Account.



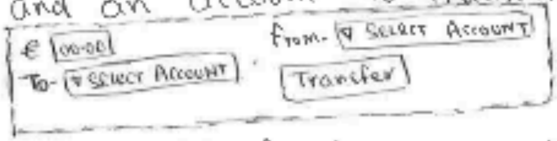
- Lodgement

For the Lodgement, a bank customer can Specify the amount to lodge with the credit card that will be debited.



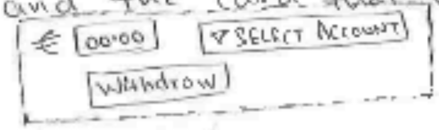
- Tranfer.

for the Transfer, the bank customer can Specify the amount to transfer and an account to transfer to.



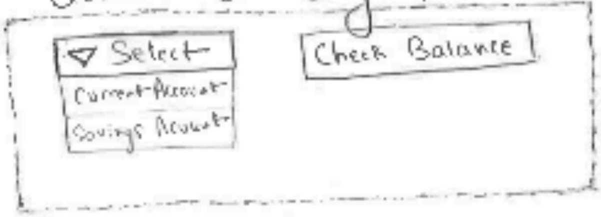
- Withdrawal

The bank customer Can Specify the amount to Withdraw and the card that will be created.

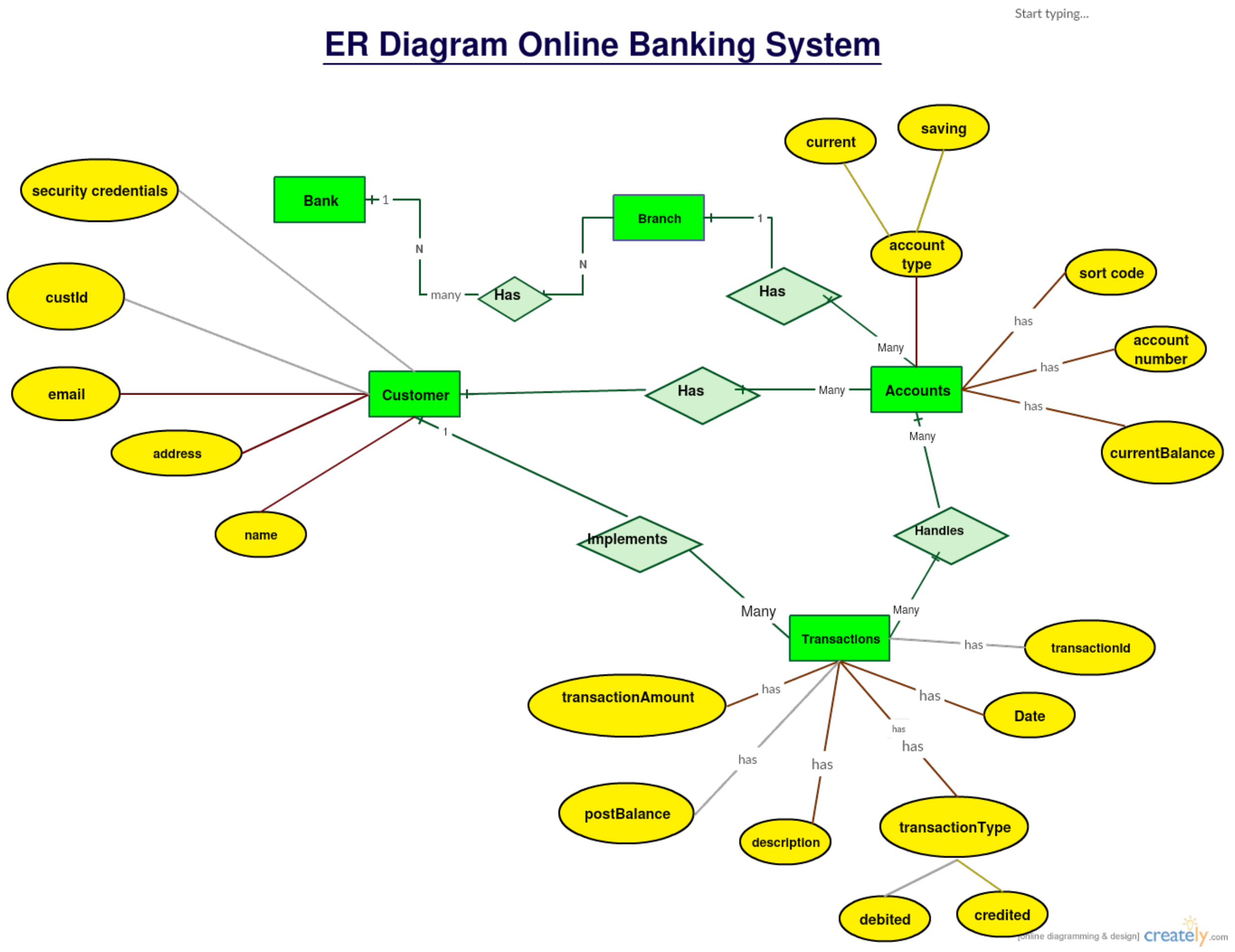


- Balance

The Customer can request a balance on any account at any time.



ERD Diagram



Security Concerns

Developing applications that must support different clients has determined the developers to use the restful APIs that became almost the standard.

In Stateless, the server does not store any state about the client session. The client must comewith all relevant information about the session and to be stored into it. The server is able to serve any client at any time. The server come up with resource state but the client come with application state, including session state.

In Statelessness, every HTTP request take place in isolation state. When a client makes a HTTP request, it has stored in all information required by the server in order to fulfil that request. One big concern of restful services is the that they are based on the http protocol that is insecure by definition.

To solve this lack of security, http has been secured by using SSL (HTTPS), to protect the data that is in transition. However, this implementation, often does not offer enough protection on data.

It must be properly built and configured allowing the use of services in a proper way. There is need of use of session authentication.

There are a many ways of standardised or custom methods to secure an authentication. But mostly used that we were decided to use is implementation of API key authentication.

The secure authentication method for HTTP request, has implemented as standardized methods follows:

- Basic HTTP Authentication
- OAuth Authentication
- Digest Authentication

Also it includes non-standardized methods as:

- Credentials in the request
- API Key authentication

The first four outlined methods are designed to be used by human (human authentication), in browser. The last one REST APIs are designed to be used by machine authentication (API key requirements).

- The API Key has some properties such as:
- It is a type of long random string (32 characters)
- An identifier (for storage and unique identification)
- It is transmitted with request
- It must be known to the client
- Can be validated by server
- It must be unique to a device or software
- It is bound to a user if is necessary

The token is stored into client device looks like a password, but not to be restricted to human memory capabilities. This token is transmitted with every request. In this way the request can be authenticated all the time during requests.

Because the API has the same features as a password this should not not be stored in the password file or in the clear-text in database. A hash key must be stored with same features as a password hash that serves as identifier.

As I outlined above, a token can be a string but at the same time, it must meet some specific requirements.

acteristic of it that is a big power of API key over passwords that are chosen by human, is that it can be random and very long length (32 characters).

This tokens must not be stored in a clear-text anywhere on the server. To store it you must implement password storage standards; it must be hashed with hashing algorithms special designed for this purpose.

An example of it is BCrypt algorithm that has 2 inputs the password and salt. The password can be truncated at 56 or 72 bit using a good random salt.

A token of 32 characters has 10 characters for the password and 22 characters for the salt. This hash must always be unique. We use collision detection to generate a new token.

References:

Studied Security Concerns API Key Authentication here (Valentin):
http://eclipsesource.com/blogs/2016/04/15/api-key-authentication-in-a-rest-api-with-jax-rs/

Solution Architecture Design:
https://github.com/Virksaabnavjot/NSVBank/blob/master/documentation/Solution%20Architecture%20Design%20by%20Navjot%20Singh%20Virk.pdf