Web Services and API Development CA

**Consumer Online Banking API**

2017

**Kamil Lasecki x14100819**

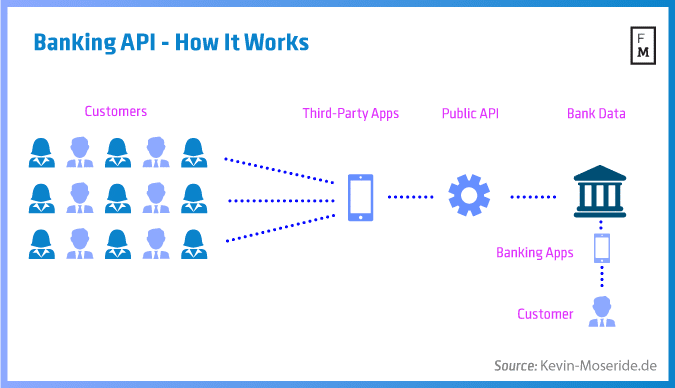
**Deniss Strods x14100398**

**Nuth Sirikitiwannakul x14101629**

# Introduction As a group we have been asked to create a problem domain online banking application for Web Services and API development project. This document will explain how important online banking is and how API will be fit to implement a typical consumer online banking application and also provide the prototype for the final project.

In particular we have been asked to create online banking for consumers; a typical consumer will be able to create a current account and a saving account. For lodgment and transfers, consumer will be able to choose the amount to transfer and to lodge in to an account and the card that will debit. Withdrawal can be chosen and the amount withdrawn and card that will be debited can be specified. Furthermore, consumer may request the balance on any account they have.

Online banking provides financial transaction services for banks at any time through the Internet via device such as computers and mobile phones. The service such as deposits, withdrawals, transfers or balance inquiries, payments for utilities, direct debit amendments, print out statements or view credit card transactions, etc. In the future, the service of online and electronic banking can be developed more and more to support the needs of users of online banking services. Now a day’s consumer using online services are increasing because online and electronic banking makes it faster and more convenient to conduct transactions and save resources. It is also cost saving for the banks as it requires less labour and fewer branches on its part.



Banking Application Program Interface is a third party that became a new competitor with the bank. It is a new financial service that is becoming popular at the moment, this banking API are working when the customers log into their information bank account and the customer will then be able to connect to the API of their choice. Then the open API technologies, in this case Banking API will then connect to various services for your daily transactions such as checking balance and all of the payment processing or income. The benefit of using the Banking API is that customers can access all the services faster, even

Instantly, saving the customer time and making life easier, transforming customer experience from lengthy waits in branches or on telephones.

In this document we going to discuss how we implement these banking online using RESTful JSON API in each point requested down below.

# The​ ​RESTful​ ​API

The​ ​API​ ​describes​ ​all​ ​the​ ​entry​ ​points​ ​below

API​ ​Name:​​ createCustomer

Description:​ This allows registering their infomation in the system.

URI:​ localhost:49000

HTTP​ ​verb:​ POST

Parameters:​ none

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​ None

Post-Conditions:​ ​a​ ​new​ ​record​ ​for​ ​the​ ​user with​ ​the​ ​specified​ ​user​ ​data.

API​ ​Name:​​ login

Description:​ This allow user to login to their account.

URI:​ localhost:49000

HTTP​ ​verb:​ POST

Parameters:​ none

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​ None

Post-Conditions:​ ​a​ ​new​ ​record​ ​for​ ​the​ ​user with​ ​the​ ​specified​ ​user​ ​data.

API​ ​Name:​​ destroySession

Description:​ This will allow the user to log out from their account.

URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​ none

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​ None

Post-Conditions:​ ​ User data will be log out

API​ ​Name:​​ getCustomer

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ setCustomer

Description:​ URI:​ localhost:49000

HTTP​ ​verb:​ POST

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ getAccounts

Description:​ URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ newAccount

Description:​ URI:​ localhost:49000

HTTP​ ​verb:​ POST

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ addMoney

Description:​ This will allow user to add money to their account.

URI:​ localhost:49000

HTTP​ ​verb:​ POST

Parameters:​ None

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ getAccount

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ getAccountBallance

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ sendMoney

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ PUT

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ deleteAccount

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ DELETE

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

API​ ​Name:​​ getAccountTransactions

Description:​

URI:​ localhost:49000

HTTP​ ​verb:​ GET

Parameters:​

​Resource​ ​contents:​ JSON or XML format (depends on HEADER set)

Pre-Conditions:​

Post-Conditions:​ ​

# Prototype

The​ ​API​ ​prototype,​ ​must​ ​be​ ​implemented​ ​according​ ​to​ ​the following​ ​requirements:   
a.​ ​​ ​​ ​​ ​​ ​An​ ​HTML + JavaScript​ ​or​ ​Mobile​ ​or​ ​Desktop​ ​client calling​ ​​ALL​​ ​portions​ ​of​ ​the​ ​API.​ ​E.g.,​ ​the​ ​client should​ ​check​ ​the​ ​account​ ​balance​ ​allowing​ ​a transfer​ ​or​ ​withdrawal.​ ​​ ​​(5​ ​Marks)   
b.​ ​​ ​​ ​​ ​​ ​A​ ​server​ ​developed​ ​in​ ​Java​ ​which​ ​implements​ ​​ALL portions​ ​of​ ​the​ ​API​ ​​using​ ​in-memory​ ​objects. Constraints​ ​should​ ​be​ ​implemented,​ ​balances should​ ​be​ ​updated​ ​and​ ​transactions​ ​should​ ​be remembered​ ​as​ ​the​ ​API​ ​is​ ​called.​ ​This​ ​should​ ​be achieved​ ​with​ ​a​ ​database​ ​layer​ ​(either​ ​in​ ​memory or​ ​disk).​ ​​(15​ ​Marks)

​ ​​ ​​ ​​ ​​ ​​ ​​ ​...OR...

A​ ​server​ ​developed​ ​in​ ​Java​ ​which​ ​implements​ ​​ALL portions​ ​of​ ​the​ ​API​ ​​using​ ​Java​ ​Persistence​ ​API (JPA)​ ​objects.​ ​Constraints​ ​should​ ​be​ ​implemented, balances​ ​should​ ​be​ ​updated​ ​and​ ​transactions should​ ​be​ ​remembered​ ​as​ ​the​ ​API​ ​is​ ​called.​ ​This should​ ​be​ ​achieved​ ​with​ ​a​ ​database​ ​layer​ ​(either memory​ ​or​ ​disk).​ ​​(35​ ​Marks)  
c.​ ​​ ​​ ​​ ​​ ​A​ ​server​ ​implementing​ ​​ALL​​ ​of​ ​the​ ​entry​ ​points listed​ ​above​ ​(Lodgment​ ​/​ ​Transfer​ ​/​ ​Withdrawal​ ​/ Balance)​ ​returning​ ​resource​ ​representations​ ​in XML​ ​and​ ​JSON.​ ​​(10​ ​Marks)

You​ ​should​ ​provide​ ​screenshots​ ​to​ ​demonstrate​ ​the operation​ ​of​ ​the​ ​Web​ ​Service.​ ​​(10​ ​Marks)   
 The​ ​prototype​ ​source​ ​code​ ​should​ ​also​ ​be​ ​included​ ​with the​ ​submission​ ​in​ ​a​ ​ZIP​ ​file