**Banking project:**

**Background:**

Internet Banking System refers to systems that enable bank customers to Access accounts and general

Information on bank products and services through a personal computer or other intelligent device.

The chances and threats that the internet symbolizes is no longer news to the present day banking sector. No traditional bank would dare face investment analysts without an Internet strategy. The main intention behind the commencement of electronic banking services is to provide the customers with an alternative that is more responsive and with less expensive options. With options just a click away, customers have more control than ever. Their expectations are usability and real-time answers. They also want personal attention and highly customized products and services.

**Project Requirements (Customer provided description):**

**Stakeholder(s):**

* Bank
  + Administration user, all privileges
  + Internal general user, per department
* Customer
  + Customer w/account
  + Customer w/o account
* System Program
  + Interest algorithm(s) / process
  + Fiat currency exchange rates

**Min. functional requirement(s) (Customer provided description):**

1. Customer
   1. Can create an account
   2. Can deposit into his/her account
      1. Cash
      2. Check
   3. Can withdraw from his/her account
   4. Can close his/her account
   5. Perform search/query for transaction detail(s) by
      1. last “n” transactions
      2. specific date
      3. date range
   6. Transfer funds to another account in same bank
   7. Request a check book
   8. View his/her monthly statement by
      1. Statement-period (one-month)
   9. Make electronic funds transfer (EFT) at
      1. Their bank
      2. Other bank
2. INTRODUCTION
   1. Overview

This document provides the Software Requirements Specifications for “Bank\_Crypto” and is intended to detail the:

* + - Technical Architecture
    - Class Diagram(s)
    - Database Diagram(s)
    - Class Responsibilities Diagram(s)
    - Use Case Diagram(s)
  1. Scope

Servers: Wildfly Application Server

Database: MySQL

ORM: JDBC

UI Framework: Angular 2 & Thymeleaf

OOP language: Java 8

Project Object Model: Maven

* 1. Reference Material

Other system documentation for this system should include:

* + - System Requirement Specification
  1. Document Conventions

All diagrams, charts and tables are generated using standard UML guidelines.

1. SYSTEM OVERVIEW
   1. Overview

This web-application is an MVC project comprised of a 4-tier layered architecture which will be utilizing Maven to manage and compile the multi-module project into a single war file to deploy. The following are the layers of the application.

* + - UI-Layer
    - Web-Layer (Controllers/Servlet/Configurations)
    - Service-Layer
    - Data-Access-Layer (Dao)

In addition to, the project will include a subset of layers required to connect and process the data transferred amongst each layer.

* + - Models
    - Model-View
    - Utils/Common
  1. UI-Layer

This is the front-end client-side facing layer. Its purpose is to provide the end-user with a graphical interface within a web-browser e.g. Firefox, Chrome. Within the web-browser interface, the end-user will have the following views:

Stakeholder Type:

Customer:

* + - Login
    - Logout
    - Sign-Up / Register
    - Profile Home
    - Account Detail
    - Add Account
    - Profile Settings
    - Profile – Other Services

Administrator:

* + - Login
    - Logout
    - Home
    - Customer
    - Customers – Detail
    - Users
    - Users – Detail
    - Settings
    - Settings – Privileges
  1. Web-Layer (Controller/Servlet/Configuration)

The controller/Servlet is responsible for receiving HTTP requests and managing sessions from an end-user request and also forwarding that request to the appropriate layer, i.e. Service, where it is processed before forwarding back an HTTP response.

In the “bank-crypto” project, an HTTP request to a specified URL is forwarded to the responsible controller/servlet for that URL. Whereby, when the HTTP request is received, that controller forwards the process to the appropriate Service-Layer to validate the request.

The URLS for servlets begin with the root folder location as the default prefix, i.e. http://bank-crypto/ and the URL mappings for the controllers/servlets are as follows:

Customer URLS:

* “/”
  + Request/Response:
    - GET
  + View results:
    - Displays the landing page i.e. home.
* “/signup”
  + Request/Response:
    - GET
    - POST
  + View results:
    - Displays the signup page, for bank customer
* “/login”
  + Request/Response:
    - GET
    - POST
  + View results:
    - Displays the login page, for bank customer
* “/logout”
  + Request/Response:
    - GET
    - POST
  + View results:
    - Displays the logout page, for bank customer
* “/profile”
  + Request/Response:
    - GET
  + View results:
    - Displays the logged-in profile
      * for bank customer
    - Displays summary of account balances
      * Present/current balance
    - Displays sidebar-nav menu
      * Account Detail, submenu
        + Will display all active account(s)
      * Add Account
        + Can add new account by type

Checking

Savings

Credit card

Crypto-currency

* + - * Profile & Settings
        + Personal

TODO

* + - * + Preferences

TODO

* + - * Other Services
    - Displays profile avatar
      * Default profile OR uploaded image
    - Displays profile name
      * Bank customer
* “/profile/account-details”
  + Request/Response
    - GET
    - POST
  + View results:
    - Displays account reprehensive logo, e.g. checking, savings, company
    - Displays type of account + alias name
    - Displays account balance
    - Display todays date
    - Displays deposit button – functional
      * will present modal window w/ form
    - Displays withdraw button – functional w/form
      * will present modal window
    - Display table result of all transaction for the present account
    - Search feature, can narrow displayed results by “String parameter”
      * Typing in characters will filter results
    - Can organize table by column, click will sort table list
    - Displays transactions by:
      * Date and time of transaction
      * Transaction type
      * Recipient name
      * Wallet balance
        + Before transaction
        + Amount of transaction
        + Fees applied to transaction
        + After transaction
* “/profile/add\_account”
  + Request/Response
    - GET
    - POST
  + View results:
    - Display a representative form for the account to add
* “/profile/profile\_settings”
  + Request/Response
    - GET
    - POST
  + View results
    - Displays options to change profile information
      * TODO details
* “/profile/other\_services”
  + Request/Response
    - GET
    - POST
  + View results
    - Close account
      * Can delete account
        + For bank customer

Admin URLS:

TODO

* 1. Web-Layer (Controllers/Servlets/Configuration)

In addition, this layer will contain Configuration settings required to connect to a database, map servlets and act as the parent for the other dependency layers.

Configuration:

* SecurityConfiguration
* TemplateConfiguration
* Web.xml
* Application.properties
* DatasourceConfiguration
  1. Service-Business-Layer

This layer is responsible for validating any request received by the Controller. For example, when an HTTP request is received by the controller, it is routed/forwarded to the Service-layer, whereas for example – in a login request, the input of the user may be authenticated before forwarding the request to the Data-layer – preventing further processing if the credentials are invalid.

Furthermore, each business object (BO) model of the application will consist of a service implementation. This implementation will validate most requests that require data, thus preventing invalid and/or malicious data from entering the database.

* 1. Data-Access-Object-Layer (DAO)

The Data-Layer contains the data-manipulation-language (DML), data-definition-language (DDL), and data-control-language (DCL) logic required to execute queries for the database. Each CREATE, READ, UPDATE AND DELETE request to this layer attempts to perform on of the fore-mentioned operations.

Once the operation has been performed, the result of the operation is then returned back through the layers of the application which is accomplished by utilizing the objects from the model-layer.

* 1. Models-Layer

In this application, the model layer will consist of classes/objects so that the application can understand/translate different data –types received by each technology into one it can understand. In summary, this layer contains dummy objects that hold data, such as those received from an input form/and or those from the database.

* 1. Model-View-Layer

Content the user is viewing can require many different fields and, because of that, special objects to construct it, therefore requiring change to a default class or object to conform to the view. Therefore, rather than changing the default construction of the object to conform to the view requirement, this Model-View-Layer will be utilized to construct specialized views requirements through association.

* 1. Utilities/Common-Layer

Will consist of generic helper and common objects that may be utilized by many difference Objects from the other layers.

1. SYSTEM ARCHITECTURE
   1. Architectural Design



* 1. Decomposition Description

This section decomposes the Models of the software application and the general implementation of its members.



* + 1. Person

Type: Abstract class

The person is an abstract class that is the basic composition of a user.

Fields:

* firstName
* middleName
* lastName
  + 1. Customer extends Person

Type: Concrete class

The customer class composes of members that would represent a user of “bank\_crypto”.

Fields:

* account

\*\* notes for creating classes \*\*

Customer properties – things that makes up a customer of bank\_crypto?

firstName

middleName

lastName

profile\_image

gender

date of birth

Location properties – things that represent the location of a customer

street address

city

state

zip code

Unique identifiers – things that can be unique about an individual

social security number

phone number

email address

contact method

hair color

salary

race

salary

employer

county of residence

Checking account – things that make up a checking account

account number

account balance

check number id

routing number

checking account owner

bank

denomination

Savings account – things that make up a savings account

account number

account balance

routing number

savings account owner

bank

Account – things that make up a financial account

isOpen

isClosed

Owner

Issuer

account number

balance

currency-type

originating country

Cash – things that make up cash

exchange value

denomination

issue country

amount

Check – things that make up a check

account number

routing number

bank issuer

check number

payable to

check owner

check amount

Crypto-currency – things that make up crypto-currency

account number < wallet address

account balance

exchange rate

value

name of currency

Transaction – things that make up a transaction

pay amount

pay date

payment type, e.g. cash, check, etc.

recipient

payer bank

recipient bank

recipient account number

payer account number

Deposit – things that make up a deposit

An abstract class, which will represent a super class for deposit method types.

- amount : double

- total : double

+ makeDeposit(double amount) : void

CashDeposit – things that make up a cash deposit

A concrete class, is representative of actions and fields relative to depositing cash.

- denominationCount : int = 0;

- usdDenomination : UsdDenomination

+ makeDeposit(double amount) : void

UsDenomination – the denomination of cash bills, for USD

- penny : int = 0;

- nickel : int = 0;

- dime : int = 0;

- quarter : int = 0;

- fiftyCent : int = 0;

- oneDollar : int = 0;

- fiveDollar : int = 0;

- tenDollar : int = 0;

- twentyDollar : int = 0;

- fiftyDollar : int = 0;

- oneHundredDollar : int = 0;

CheckDeposit – things that make up a check deposit

- check : Check

- checkTransaction : CheckTransaction

Check – things that make up a check

- checkNumber : int

- routingNumber : int

- accountNumber : int

- bankIssuer : Bank

CryptoDeposit – things that make up a crypto deposit

- cryptoDepositId : long

- cryptoType : CryptoType

- cryptoAmount : double

- cryptoDecimalPlaces : byte

CryptoType – identifies the type of crypto-currency

- cryptoTypeId : long

- cryptoTypeDescription : String

- isActive : Boolean

Withdraw – things that make up a withdrawal

CashWithdraw – things that make up a cash withdraw

CheckWithdraw – things that make up a check withdraw

CryptoCurrencyWithdraw – things that make up a crypto-currency-withdraw

Order – things that make up an order, services

CheckOrder – things that make up checkOrder

Credit card – things that make up a credit card

History – things that make up a history

Record – things that make up a record

Transaction – things that make up a transaction

CashTransaction – things that make up a cash-transaction

CheckTransaction – things that make up a check-transaction

CreditTransaction – things that make up a credit-transaction

CryptoTransaction – things that make up crypto-transaction

WithdrawTransaction – things that make up a withdraw-transaction

DepositTransaction – things that make up a deposit-transaction

TransactionRecord – things that make up a transaction-record

CreditCardRecord – things that make up a credit-card-record

CheckingAccountRecord – things that make up a checking-account-record

SavingsAccountRecord – things that make up a savings-account-record

DepositRecord – things that make up a deposit-record

WithdrawalRecord – things that make up a withdrawal-record

Profile – things that make up a profile

BankCustomerProfile – things that make up a bank-customer-profile

CheckingCustomerProfile – things that make up a checking-customer-profile

SavingsCustomerProfile – things that make up a savings-customer-profile

CryptoCustomerProfile – things that make up a crypto-customer-profile

Characteristic – things that make up a characteristic

Attribute – things that make up an attribute

AttributeCategory – things that identity/group attributes categories

CharacteristicCategory – things that identify/group characteristic categories

Charge – things that make up a charge

Fee – things that make up a fee

BankFee – things that make up a bank-fee

BankCharge – things that make up bank-charge

NetworkFee – things that make up a network-fee

NetworkCharge – things that make up network-charge

ServiceFee – things that make up a service-fee

ServiceCharge – things that make up a service-charge

CheckingFee – things that make up a checking-fee

CheckingCharge – things that make up a checking-charge

SavingsFee – things that make up a savings-fee

SavingsCharge – things that make up a savings-charge

CryptocurrencyFee – things that make up crypto-currency-fee

CryptocurrencyCharge – things that make up a crypto-currency-charge

CryptoCustomer – things that make up a Crypto-Customer

CryptoCustomerCharacteristic

CheckingCustomer

CheckingCustomerCharacteristic

SavingsCustomer

SavingsCustomerCharacteristic

Bank – things that make up/identify a bank

Branch – things that make up a bank branch

Favorite – things that a favorite is composed of

FavoriteCategory – things that a category can be grouped into

UML Class Responsibilities Diagrams (ref 4):

UML Class Diagrams (ref 5-8):

UML Sequence Diagrams (ref 13):

Use Case Diagrams (ref 15 – 20):

STAKEHOLDER (BANK CUSTOMER):

* UC-1000001 – Register for an online account to view bank account details
* UC-1000002 – Login to account
* UC-1000003 – Logout of account
* UC-1000004 – Close single account, $0.00 balances
* UC-1000005 – Close all accounts, $0.00 balances
* UC-1000006 – Add an account, checking + currently registered
* UC-1000007 – Add an account, savings + currently registered
* UC-1000008 – Add an account, crypto-currency + currently registered
* UC-1000009 – Add an account, credit card – type Visa
* UC-1000010 – Add an account, credit card – type Master-card
* UC-1000011 – Add an account, credit card – type Discover
* UC-1000101 – Deposit fiat-currency into account, USD + checking
* UC-1000102 – Deposit fiat-currency into account – check + electronic scan
* UC-1000103 – Deposit fiat-currency into account – ACH deposit
* UC-1000104 – Deposit fiat-currency into account – credit card
* UC-1000150 – Deposit crypto-currency into account, by purchase w/ fiat-currency
* UC-1000151 – Deposit crypto-currency into account, by purchase with credit card
* UC-1000200 – Withdraw from primary account
* UC-1000201 – Withdraw from secondary account
* UC-1000300 – Request/reorder check(s) for primary account
* UC-1000301 – Request an ATM card
* UC-1000302 – Request statement(s)
* UC-1000400 – View transaction history (current-statement period)
* UC-1000401 – View transaction history (current-statement period)
* UC-1000500 – View Transaction history (period-range)
* UC-1000600 – View Transaction history (by “n” transactions)
* UC-1000601 – View Transaction history (by “search name”)
* UC-1000700 – View total, count, of all transactions, i.e. purchases, withdrawals, credits.
* UC-1000701 – View total, count, of all transactions by type – purchases
* UC-1000702 – View total, count, of all transactions by type – withdrawals
* UC-1000703 – View total, count, of all transactions by type – credits
* UC-1000704 – View total, count, of all transactions by type – fees
* UC-1000800 – View monetary total of all transactions
* UC-1001200 – Transfer money electronically into account (same bank, internal incoming)
* UC-1001201 – Transfer money electronically out of account (same bank, internal outgoing)
* UC-1001210 – Transfer money electronically into account (different bank, external incoming)
* UC-1001211 – Transfer money electronically out of account (different bank, external outgoing)

STAKEHOLDER (BANK):

* UC-2000001 – Create a bank account for customer
* UC-2000101 – Charge bank customer an overdraft fee
* UC-2000102 – Charge bank customer a statement fee
* UC-2000103 – Charge bank customer a transaction fee
* UC-2000104 – Charge bank customer a money transfer fee
* UC-2000105 – Charge bank customer an ATM fee
* UC-2000106 – Charge bank customer monthly service charge fee