<Online banking>

Supplementary Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction.

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

# Non-functional Requirements

* The online banking system provides an improved and secured transaction system, which envisages the future expansion, and modification, which is necessary for a core sector like online banking operations.
* Source of stimulus: human
* Stimulus: the customer uses the application
* Environment: the server is functional
* Artifact: the application
* Response: details in the database are being modified
* Response measure: checking the user input for correctness

## Availability

The application will be available to every user that has internet connection and wishes to register. After registration, the account can operate for free.

## Performance

As transfers take place in no time, the performance of the application regarding transfer speed would be much higher than regular bank transactions.

## Security

The application provides a secure environment for the user as each account has a private username and password.

## Testability

Test cases considered:

1. Testing the functionality of customer operations:
   * Whether the user can see the transactions history
   * Whether the user can enquire his account balance
   * Whether the user can transfer money to other accounts
   * Whether the user can navigate through the home page and choose what he wants to do next
2. Testing the functionality of the register page
   * Whether the user can fill in the data
   * Whether when submitting the data the validators prevent entering wrong data
   * If the user submits valid data, whether it is registered

* This can be verified by the admin.

1. Testing the functionality of the login page
   * If the user fills in the password that corresponds to the password in the database tied to that user account
   * If the user fills in the username that corresponds to the username in the database tied to that user account
   * He can login into his account
2. Testing the functionality of the admin page
   1. Whether the admin can see all the transactions that took place over the webpage
   2. Whether the admin can register a new customer to the application based on request
   3. Whether the admin can delete a customer from the database in case he didn’t respect the policy or based on demand

## Usability

The application can be used by anyone who has access to internet.

# Design Constraints

This project will be implemented using Java language as it encapsulates concepts that will be needed in the application development. The used IDE will be Eclipse. The database used will be created using MySQL Server (back-end part). The building of the user interface(front-end) is created with Java Server Pages (JSP), which communicate with the server using Java Servlets.

I decided to use the **Factory design pattern** which is applied to the classes from the dataAccessLayer, which assure the connectivity with the data base. A factory pattern is one of the core design principles to create an object, allowing clients to create objects of a library in a way such that it doesn’t have tight coupling with the class hierarchy of the library.