

Software design document for internet banking system

Internet Banking System SDD



May 14, 2022

Version 1.0

Prepared B: Saad Hamdy

ITI SMART VILLAGE

Table of Contents

[1. Purpose of the document 2](#_Toc103695410)

[2. Summary of the project 2](#_Toc103695411)

[3. Table of functions 2](#_Toc103695412)

[3.1. Admin Add Account 2](#_Toc103695413)

[3.2. Admin Add Client 3](#_Toc103695414)

[3.3. Client Registration 4](#_Toc103695415)

[3.4. Delete Client 4](#_Toc103695416)

[3.5. Edit Client 5](#_Toc103695417)

[3.5.1. Edit Client Step One 5](#_Toc103695418)

[3.5.2. Edit Client Step Two 6](#_Toc103695419)

[3.6. Login 7](#_Toc103695420)

[3.7. Transfer History 7](#_Toc103695421)

[3.8. Transfer Money 8](#_Toc103695422)

# Purpose of the document

This document targets all team members in order to clarify the system and describe each function sequence and its inputs & outputs.

All functions mentioned in the document are specified to be front-end or back-end. For each function, functionality and Unique ids are also specified.

# Summary of the project

The internet banking system is relevant everywhere where banking exists. All banks will favor an online banking system over the traditional banking systems as it has more features and provides faster transfer methods.

# Table of functions

## Admin Add Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Add\_Account\_01 | Page\_Load | This function runs automatically when the page is reloaded initializing the database connection | No inputs (only the inputs created by the compiler) | The connection is established and the page is loaded |  |
| Add\_Account\_02 | AddAccountBtn\_Click | This function is run when the add account button is clicked | No inputs (only the inputs created by the compiler | The account added in the database if the data is valid based on the requirements | UserHasTwoAccounts |
| Add\_Account\_03 | UserHasTwoAccounts | This function checks if the user has already two accounts so he cannot have more than that. | Clientid(integer32) | Return a Boolean value that is true if than user has more than two accounts |  |

## Admin Add Client

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Add\_Client\_01 | Page\_Load | This function is the same as the one in Add\_Account\_01 | -- | -- |  |
| Add\_Client\_02 | AddClientBtn\_Click | This function is run when the Add client button is clicked | No inputs (only the inputs created by the compiler | The a client is added into the database if the data is valid based on the requirement | \*clientAlreadyExists  \*addClient  \*addPhoneNumber  \*addAccount |
| Add\_Client\_03 | addClient | This function is called to add the client if the data is valid | No inputs (only can use the class data) | The client is added into the function |  |
| Add\_Client\_04 | addAccount | This function take the account number from the data of the class and add the account in the database | No inputs (only can use the class data) | The Account is added into the database |  |
| Add\_Client\_05 | addPhoneNumber | This function adds the phone number from the field into the class and add it to the database | No inputs (only can use the class data) | The phone number is added into the database |  |
| Add\_Client\_06 | clientAlreadyExisits | The function takes the username inserted by the user and checks if it already exists in the database | clientUserName(string) | Boolean detects if the user exists or not in the database |  |

## Client Registration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Client\_Registration\_01 | Page\_Load | This function is the same as the one in Add\_Account\_01 | -- | -- |  |
| Client\_Registration\_02 | RegisterBtn\_Click | The function is called when the register button is clicked to add into the database | No inputs (only the inputs created by the compiler | The user is inserted into the database with all its realtions | \*UserNameisUnique  \*ConfirmPasswordEqualsPassword  \* addClient  \* addAccount  \*addClientPhoneNumber |
| Client\_Registration\_03 | UserNameisUniqe | The function is called to check if the username inserted by the client is unique | No inputs (only the data of the class) | Boolean that indicates if the username is unique or not |  |
| Client\_Registration\_04 | ConfirmPasswordEqualsPassword | The function is called when to check if the confirm password field is equal to the password field | No inputs only the class data | Boolean that indicates if the password is equal to confirm password |  |
| Client\_Registration\_05 | addClientPhoneNumebr | The function is called to add the phone number into the database | No inputs only the class data | The phone number is added into the database |  |
| Client\_Registration\_06 | addAccount | The function is called to add the account written by the user into the database | No inputs only the class data | The Account number is added into the database |  |

## Delete Client

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Delete\_Client\_01 | Page\_Load | This function is the same as Add\_Account\_01 | -- | -- |  |
| Delete\_Client\_02 | DeleteClientBtn\_Click | Checks if the client is already in the database and then delete the Client and his accounts | No inputs (only the inputs created by the compiler) | The Client and all his accounts are deleted from database | \* deleteAccount  \* deleteClient |
| Delete\_Client\_03 | deleteAccount | This function is created to delete all the accounts of the selected client from the database | No inputs (only the data created in the class) | The client accounts are deleted from the database |  |
| Delete\_Client\_04 | deleteClient | This function delete the Client record from the database | No inputs (only the data created in the class) | The client record is delete from the database |  |

## Edit Client

### Edit Client Step One

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Edit\_Client\_01 | Page\_Load | This function is the same as Add\_Account\_01 | -- | -- |  |
| Edit\_Client\_02 | FindClientDataBtn\_Click | Search is the user exists in the database and redirects to EditClientStepTwo | No inputs (only data class and data read by front-end) | Retrieve the client’s data if found | GetClientDetailsByUserName |
| Edit\_Client\_03 | GetClientDetailsByUserName | Get all the client details using his username written by the admin | No inputs (only data class and data read by front-end) | Retrieve all the client’s data | \*GetPhoneNumber  \*GetAccountNumber |
| Edit\_Client\_04 | GetPhoneNumber | Get all the client phone number | No inputs (only class data) | Retrieve the client phone number |  |
| Edit\_Client\_05 | GetAccountNumber | Get the client’s account number | No inputs (only class data) | Retrieve the Clients Account Number |  |

### Edit Client Step Two

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Edit\_Client\_06 | Page\_Load | This function runs automatically when the page is reloaded initializing the database connection and set all the class variables by the data passed from the EditClientStepOne | No inputs (only the inputs created by the compiler) | The connection is established and the page is loaded |  |
| Edit\_Client\_07 | EditClientBtn\_Click | Edit the client’s data based on the inserted data | No inputs (only the inputs created by the compiler) | The edits in the client’s data based on the inserted data | \*UserNameisUnique  \*ConfirmPasswordEqualsPassword  \* updateClient  \* updateAccount  \*updateClientPhoneNumber |
| Edit\_Client\_08 | UserNameisUnique | Checks if the Client’s new username is unique | No inputs (only data class) | Return a Boolean that checks if the username is unique |  |
| Edit\_Client\_09 | ConfirmPasswordEqualsPassword | Checks if the password and confirm password are equal | No inputs (only data class) | Return a Boolean that checks if the password and confirm password are equal |  |
| Edit\_Client\_10 | updateClient | Edits the client data record in the database based on the inputs by the Admin | No inputs (only class data) | Update the data record in the database |  |
| Edit\_Client\_11 | updateAccount | Update the account number as inserted by the admin | No inputs (only class data) | Update the account number of the client in the database |  |
| Edit\_Client\_12 | updateClientPhoneNumber | Update the phone number when updated by the admin | No inputs(only class data) | Update the client phone number in the database |  |

## Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Login\_01 | Page\_Load | This function is the same as Add\_Account\_01 | -- | -- |  |
| Login\_02 | LoginBtn\_Click | Checks if the data inserted is client’s data or admin’s data then login based on the date inserted | No inputs (only the inputs created by the compiler and class data set by the front end) | The end user is logged in as Client or as admin based on their data | \* LoginAsClient  \* LoginAsAdmin |
| Login\_03 | LoginAsClient | This function try to log in using the data inserted by the end user as client | username string  password  string | Boolean that detects if the login is done successfully and redirection to the client profile |  |
| Login\_04 | LoginAsAdmin | This function try to log in using the data inserted by the end user as Admin | username string  password  string | Boolean that detects if the login is done successfully and redirection to the Admin dashboard |  |

## Transfer History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Transfer\_History\_01 | Page\_Load | This function runs automatically when the page is reloaded initializing the database connection.  Get all the accounts associated with his name | No inputs (only the inputs created by the compiler) | The connection is established and the page is loaded  Data appear in table format. |  |
| Transfer\_History\_02 | addToTable | Retrieving the data from the transfer history and show it to the client in table format | AccountNumber string | Data of transfers appear in table form |  |

## Transfer Money

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Inputs | Outputs | Called Functions |
| Transfer\_Money\_01 | Page\_Load | This function is the same as the one in Add\_Account\_01 | -- | -- |  |
| Transfer\_Money\_02 | TransferBtn\_Click | This function adds a new transfer into the database after checking the required validations | No inputs (only data class) | The transfer is added into the database | \*checkIfAccountsnumberValid  \*isSourceAccountNumbreIsOwnedByClient  \*sourceAccountHasSufficentBalance  \*amountShouldOnlyBeUnderTwentyThousand  \* addTransfer |
| Transfer\_Money\_03 | checkIfAccountsnumberValid | Checks if the accounts both source and destination are in the database | SourceAccountNumber string  Destination Account Number string | Return Boolean checks if source and destination accounts are already in the database |  |
| Transfer\_Money\_04 | isSourceAccountNumbreIsOwnedByClient | Checks if the source account is owned by the client logged in | No inputs (only class data) | Return Boolean checks if the source account is owned by the client logged in |  |
| Transfer\_Money\_05 | sourceAccountHasSufficentBalance | Checks if the client’s source account number has sufficient balance to complete the transfer | sourceAccountNumber string | Return a Boolean that checks if the  Source account has sufficient balance to complete transfer | convertToEGP |
| Transfer\_Money\_06 | convertToEGP | Convert the amount of the transfer from USD To EGP to complete the transaction | Amount float | Return a float value represent the amount in EGP or return Nan in cases of not choosing a currency |  |
| Transfer\_Money\_07 | amountShouldOnlyBeUnderTwentyThousand | Checks if the amount inserted by the use is under 20000 EGP to complete the transfer | Amount float | Return a boolean that checks if the amount is under 20000EGP or not |  |
| Transfer\_Money\_08 | addTransfer | Adding the transfer into the database and updates all the records into the database | sourceAccountNumber string  DestinationAccountNumber string | Adding the transfer into the database and update the related records | \*subAmountFromBalance  \*updateSourceAccount  \*updateDestinationAccount |
| Transfer\_Money\_09 | updateSourceAccount | Updating the source account balance | sourceAccountNumber string | Update the source account balance from the transfer into account | subAmountFromBalance |
| Transfer\_Money\_10 | updateDestinationAccount | Updating the destination account’s balance | DestinationAccountNumber string | Update the destination account balance from the transfer into account |  |
| Transfer\_Money\_11 | subAmountFromBalance | Returns the float difference between the balance and amount | No inputs uses only class data | Float number represent the difference between the balance and amount |  |

**Revision Date**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Owner Name** | **Date** | **Status** | **Approved by** | **Version** |
| Saad Hamdy | 14/5/2022 | Approved | Mariam | 1.0 |
|  |  |  |  |  |
|  |  |  |  |  |