AN INTERNET BANKING SYSTEM

Computer Science

Honours Documentation 2012

Author: Mohamed Hassan Ali

Student Number: 3270152

Supervisor: Mr. Michael Norman

Department of Computer Science

UNIVERSITY of the WESTERN CAPE

A mini-thesis submitted in partial fulfillment of the requirements for the degree of

B.Sc. Honours.

ABSTRACT

The adoption of Electronic Banking by commercial enterprises has been in existence since the mid 90s, much greater in number due to lower operating costs associated with it. Electronic banking has initially been in the form of automatic teller machines and telephone transactions.

More recently, it has been transformed by the Internet, a new delivery channel for banking services that benefits both customers and banks.

Internet banking system services can include: Open an account, Balance enquiry, Request for Cheque book, Beneficiary payments (EFT), Viewing monthly.

Furthermore, customer's application for electronic banking facilities is expanding as the cost savings on transactions over the Internet are significant.

PLAGIARISM DECLARATION

I, Mohamed Hassan Ali, certify that this project is my own work. I understand what plagiarism is and I have used quotations and references to fully acknowledge all the words and ideas of others, which we have used in our project. I have not copied anyone else's project. I have also not permitted anyone to copy my project.

Signature: _ _ _ _ _ _ _ _ _ _ _ _

ACKNOWLEDGEMENTS

First and foremost I am ever grateful to my Allah to whom I owe my life. I would also like to thank my parents for giving me the opportunity to study at the university of western cape.

I would also like to thanks my classmates specially Chisha Malama for his support, without him I wouldn’t be where I am today.

I am wholeheartedly grateful to my supervisor Mr. Michael Norman for guiding me to reach my initial milestones in the first semester.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMES

CSS-Cascading Style Sheets

EFT-Electronic Funds Transfer

HTML-Hypertext Mark-up Language

Internet Banking System- A system allowing individuals to perform banking activities at home, via the internet.

MYSQL-is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

RAD-Requirements Analysis Document

SSL-Secure Socket Layer

PHP-Hypertext Pre-processor

URD-User Requirements Document

Chapter 1

USER'S REQUIREMENTS DOCUMENT

1.1 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.

1.2 Problem Statement

We got user requirements from some Computer Science students, Zukilla Roro, Micheal Motlhabi, Yasser Buchana, and friends Allen Mwangonde, Ismail, from which we formulated the document analysis in February 2012.

Internet banking identifies a particular set of technological solutions for the development and the distribution of financial services, which rely upon the open architecture of the Internet. With the implementation of internet banking system, it maintain a direct relationship with the end users via the web and are able to provide a personal characterization to the interface, by offering additional customized services.

1.3 Scope of the Study

The scope of this project is limited to the activities of the operations unit of the banking system which includes opening of Account, Deposit of funds, Electronic funds transfer, Cheque balance and Monthly statement.

In the figure below, is the use-case diagram of the Internet banking system that the customer can expect all those functions with the bank manager acceptance.

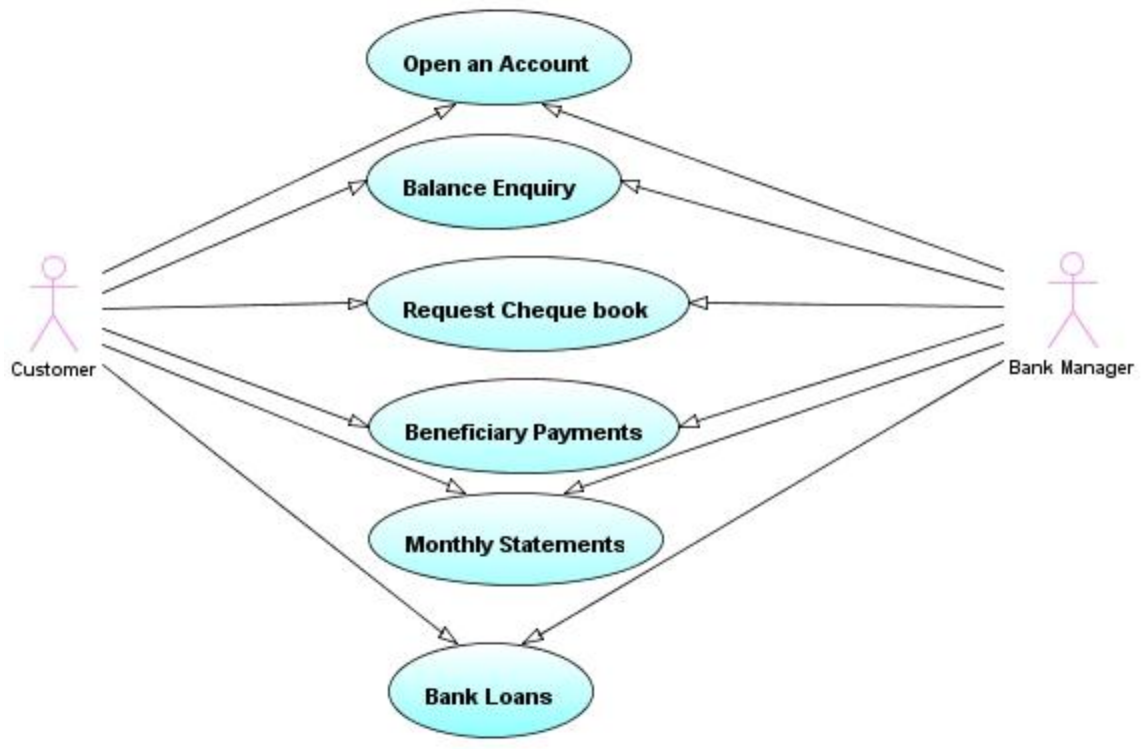


Figure-1 Use-case Diagram of showing user requirements

1.4 Limitations of the Internet Banking System

- **Problems of security:** Various sites are not properly locked at to ensure whether the customer's money is safe in cyber world or not.
- **Wrong assumption:** Many people are afraid using Internet Banking because of the assumption that it is more expensive than the traditional method of dealing with bank transactions. They still prefer going to bank to perform transactions.
- **Lack of awareness:** Another great hindrance is lack of awareness because effective and wide media efforts in publishing Internet Banking need to be emphasized.

Chapter 2

REQUIREMENTS ANALYSIS DOCUMENT

2.1 Functional Requirements

- Customer can request details of the last 'n' number of transactions he has performed on any account.
- Customer can make a funds transfer to another account in the same bank.
- Customer can request for cheque book
- Customer can view his monthly statement. She/he can also take print out of the same.
- Customer can make EFT's to accounts at their and other banks.
- The system is providing balance enquiry facility.

2.2 Non-functional Requirements

Those requirements which are not the functionalities of a system but are the characteristics of a system are called the non-functionalities.

- Secure access of confidential data. SSL can be used.
- 24X7 availability
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extensions.

2.3 Class Diagram

The below class diagram shows that the customer can have more than one account and that relationship goes to one-many relationship.

The transaction functions always depends on the web service, which means it's a web based.

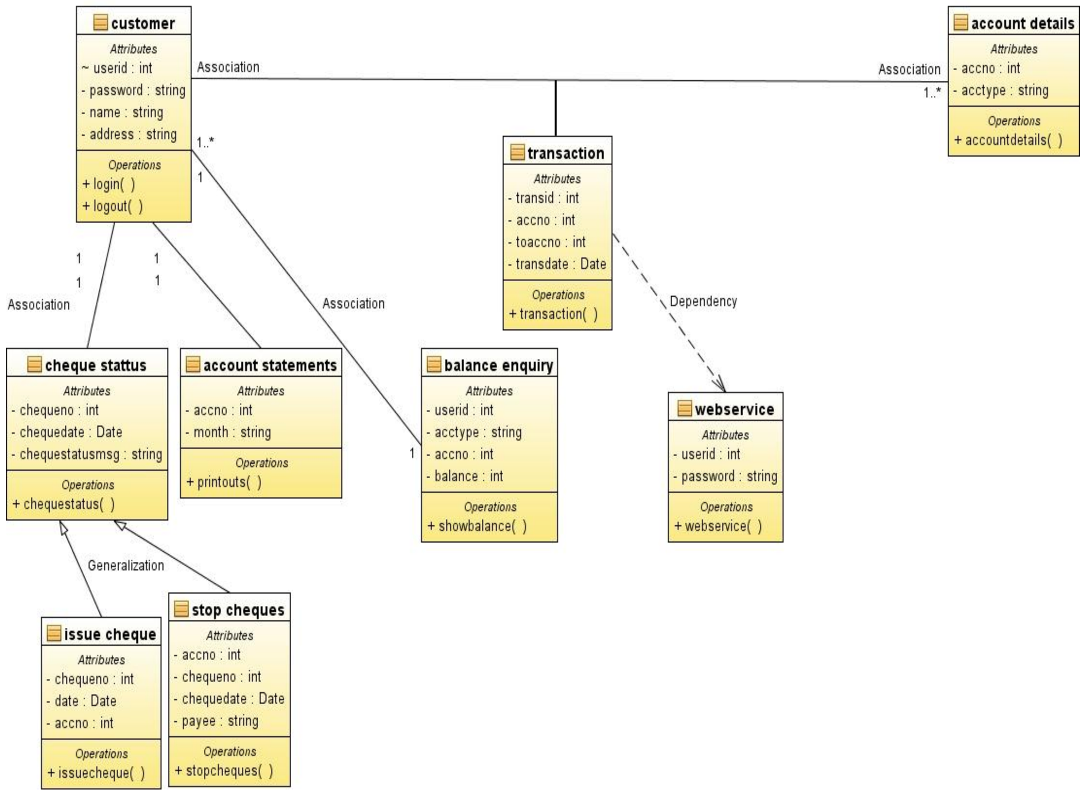


Figure-2 Class Diagram of Internet banking system

2.4 System Requirements

Software Requirements	**Hardware Requirements**
Operating System: Windows or linux or MAC	Processor: any
User Interface: HTML, CSS	Hard Disk: 10 GB minimum
Programming Language: PHP	RAM: 256MB or more
Database: MYSQL	Any Screen

Table 1: System Requirements

Chapter 3

USER INTERFACE SPECIFICATION

The purpose of this document is to provide a detailed specification of the Internet Banking System user interface. These requirements will detail the outwardly observable behavior of the program. The user interface provides the means for the user, to interact with the program. This User Interface Specification is intended to convey the general idea for the user interface design and the operational concept for the software. This document will be updated with additional detail as our analysis and design activities progress.

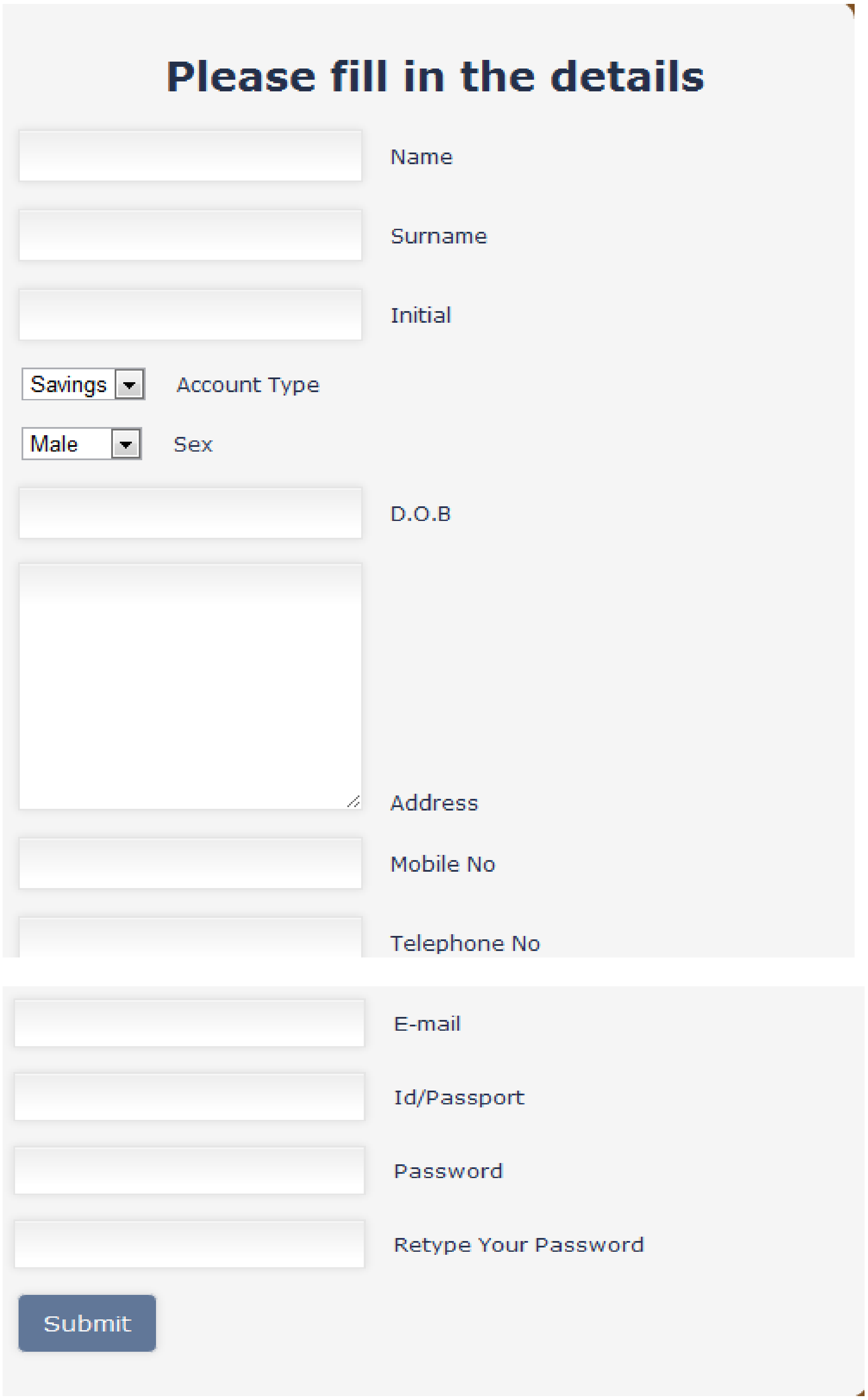
Section 2.5 gives a description of the complete user interface, section 2.6 shows what the user interface looks like to the user, section 2.7 tells how the interface behaves and section 2.8 tells how the user interacts with the system.

3.1 Description of the complete user interface

The User Interface Specification (UIS) consists of one main graphical user interface (GUI), which consists with different operations enlisted in the options.

3.2 What the user interface looks like to the user

The customer must first register then only the customer can open a new account in the system. He/She must fill all the details in the below form, as well as choose a password in order to login after the registeration.



Please fill in the details

	Name
	Surname
	Initial
Savings	Account Type
Male	Sex
	D.O.B
	Address
	Mobile No
	Telephone No
	E-mail
	Id/Passport
	Password
	Retype Your Password

Figure-3 Registeration form and opening a new account

The Login page consists of two text boxes, namely Account No and Password, and a login command button allowing the customer to log into the system. The login page helps the customers to login as a user who visualizes and analyze data contained in the database.

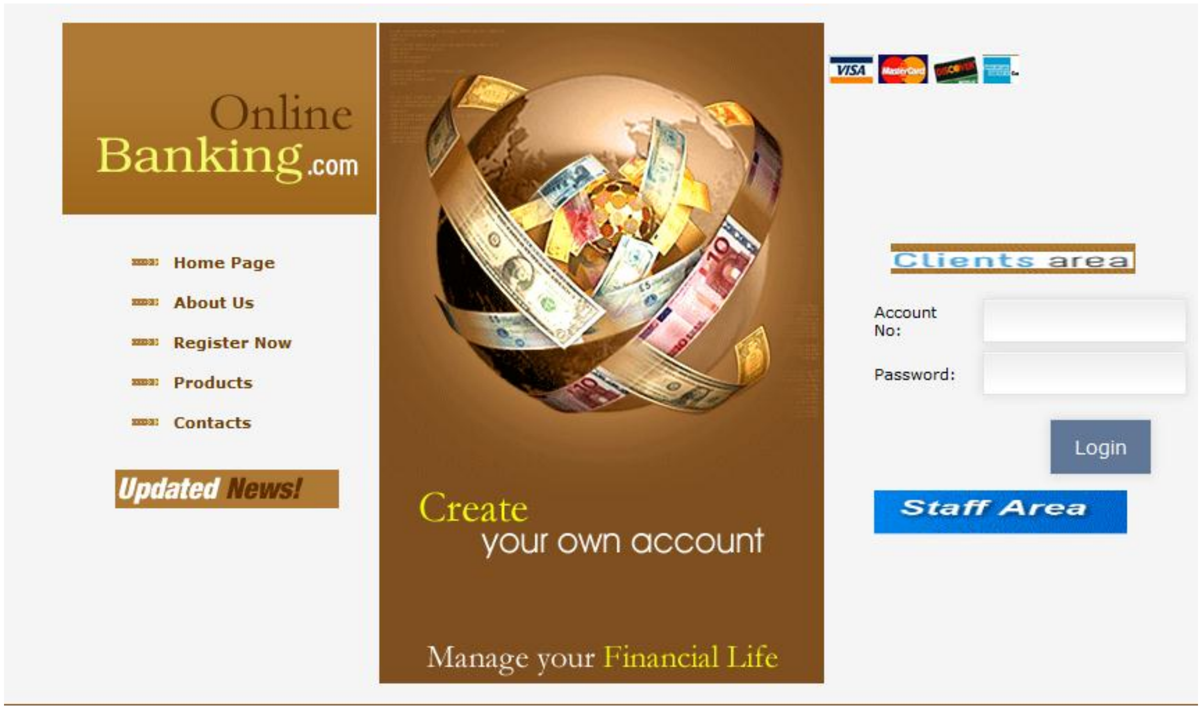
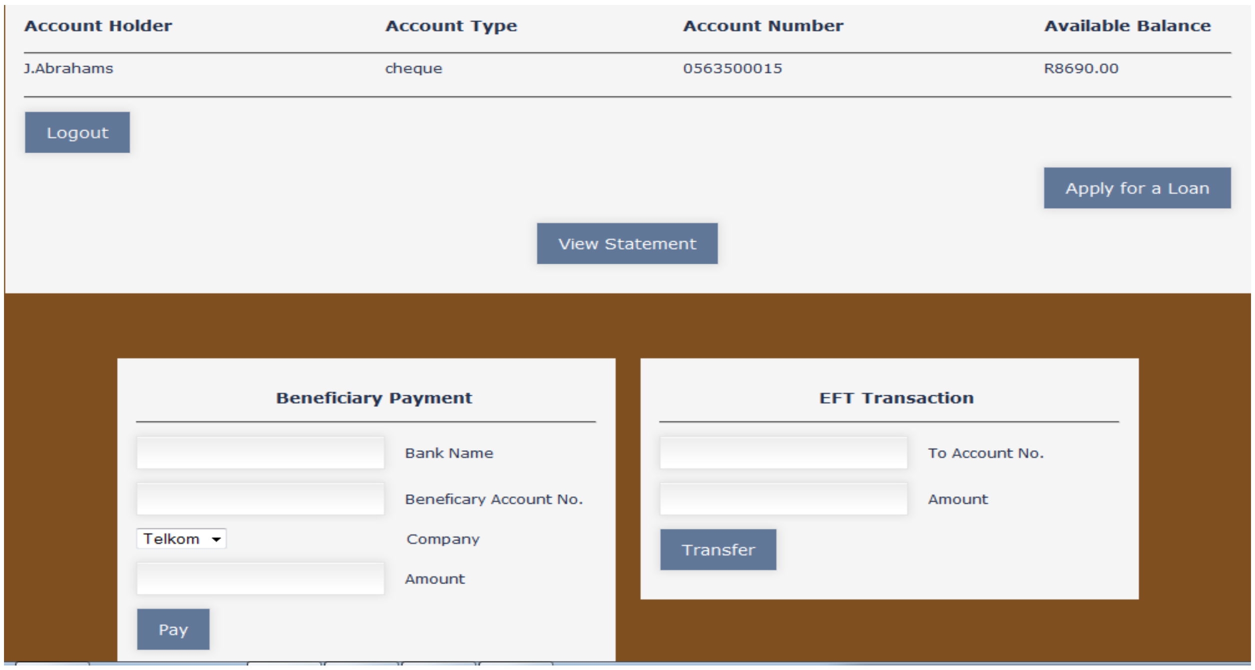


Figure-4 Home Page

Once logged on, the customer is ready to perform the transactions.



Account Holder	Account Type	Account Number	Available Balance
J.Abrahams	cheque	0563500015	R8690.00

Figure-5 Transaction Page

3.3 How the user interface behaves

The system verifies customers input in the field of account no and password, and displays an error message if the customer enters incorrect information. Thus, if the customer provides an appropriate data, then he will be allowed to logged in.

3.4 How the user interacts with the system

The sequence diagram shows how the customer can open an account as well as how to register the internet banking system in order to login the system. When the customer submit all the details in the form then the system automatically gives an account and sends to the database.

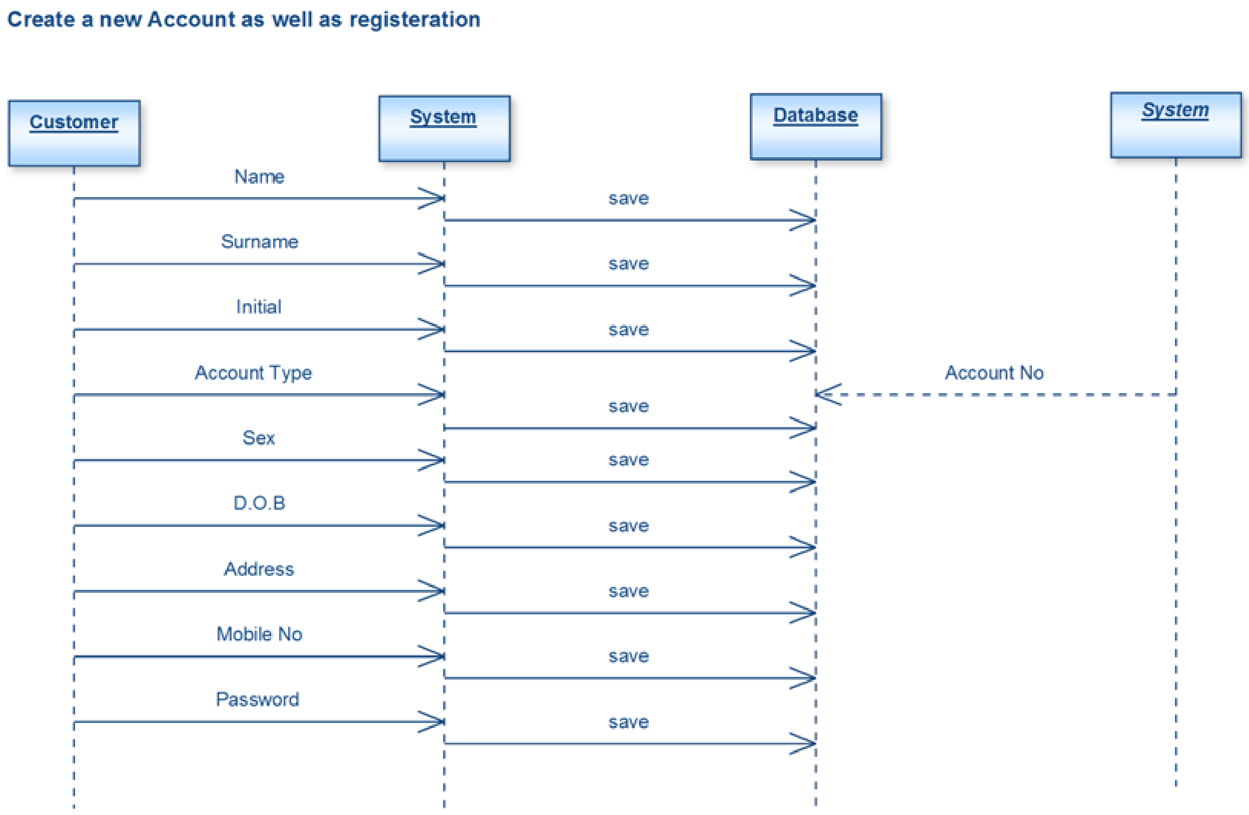


Figure-6 Create a new account

Then the login process is shown below, the customer enters a valid account number and password then the system checks if it is correct input or not, if it is correct then it allows to access for the transactions, if it is not correct it will remain the home page.

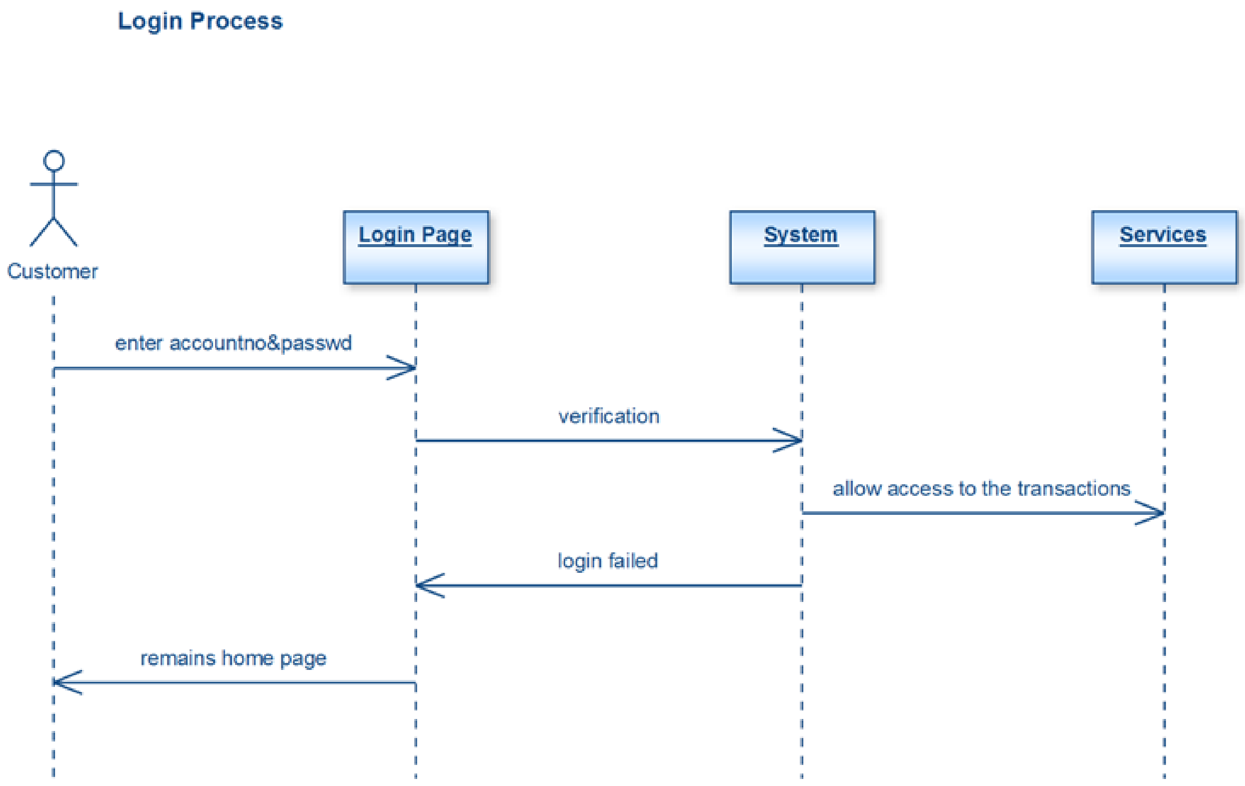


Figure-7 Login Process

3.5 How the admin interface looks like

The administrator should login to perform the transactions like to approve or disapprove a loan and so on

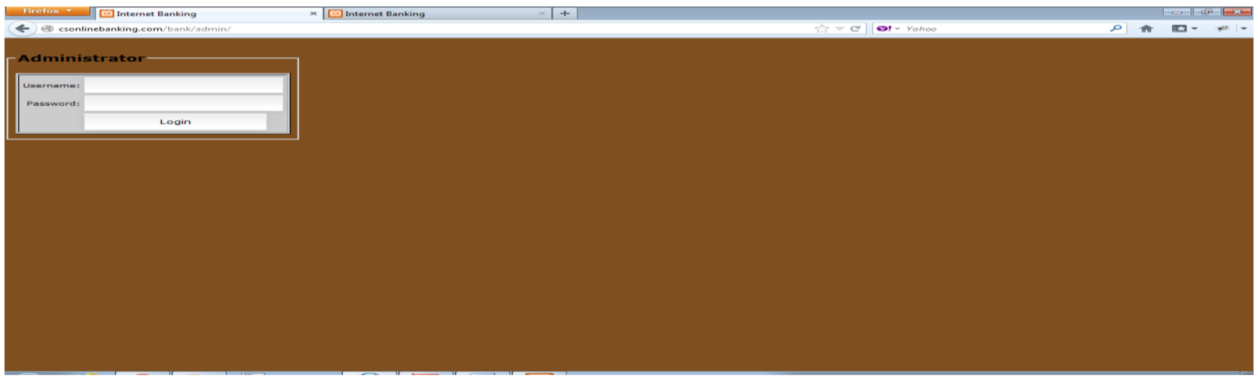


Figure-8 AdminPage

C h a p t e r 4

HIGH LEVEL DESIGN (OBJECT ORIENTED ANALYSIS)

This chapter presents the object oriented view of the system, analysis of the high level design and describes the objects needed to implement the system. Each one of these objects is described and documented, and a data dictionary providing details of each object is provided.

4.1 Data Dictionary

Table Name: **LOGIN**

Description: This table is used to store Login details.

Key	Field Name	Data Type	Length	Nullable
PK	ACCOUNTNO	VARCHAR	12	NO
	PASSWORD	VARCHAR	45	NO

Table 2: Login Table

Table Name: **CLIENTS**

Description: This table is used to store customer details.

Key	Field Name	Data Type	Length	Nullable
	NAME	VARCHAR	45	NO
	SURNAME	VARCHAR	45	NO
	INITIAL	VARCHAR	10	NO
	ACCOUNTTYPE	VARCHAR	45	NO
	SEX	VARCHAR	6	NO
	D.O.B	DATE		NO
	ADDRESS	VARCHAR	200	NO
	MOBILENO	VARCHAR	10	NO
	TELEPHONENO	VARCHAR	10	NO
	EMAIL	VARCHAR	45	NO
PK	ID_PASSPORT	VARCHAR	45	YES

Table 3: Clients Table

Table Name: **ACCOUNT**

Description: This table is used to store account details.

Key	Field Name	Data Type	Length	Nullabe
FK	ACCOUNTNO	VARCHAR	12	NO
	ACCOUNTTYPE	VARCHAR	45	NO
	ACCOUNTHOLDER	VARCHAR	45	NO
	DATEOPENED	DATE		NO
	BRANCHCODE	INT	5	NO
	DATEAPPROVED	DATE		NO
	ACCOUNTBALANCE	DECIMAL		NO
	APPROVED	VARCHAR	6	NO
	DISAPPROVED	VARCHAR	6	NO

Table 4: Account Table

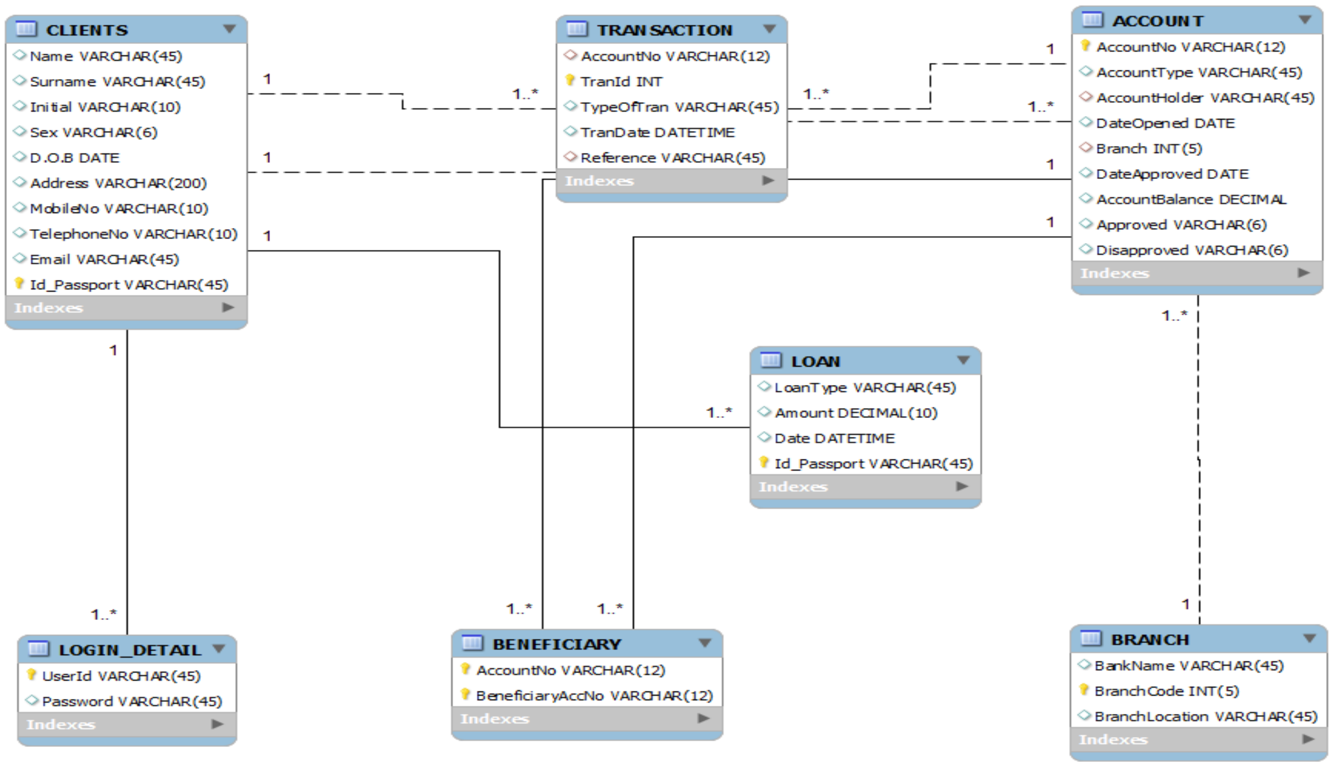
Table Name: **TRANSACTION**

Description: This table is used to store the transaction details.

Key	Field Name	Data Type	Length	Nullable
FK	ACCOUNTNO	VARCHAR	12	NO
	TRANSACTIONID	INT		NO
	TYPEOFTRANSACTION	VARCHAR	45	NO
	TRANSACTIONDATE	DATETIME		NO
	REFERENCE	VARCHAR	45	NO

Table 5: Transaction Table

4.2 Entity-Relationship Model



Chapter 5

LOW LEVEL DESIGN (OBJECT ORIENTED DESIGN)

This chapter presents the object oriented design of the system, analysis of the low level design and provides details for the object oriented analysis of the system.

5.1 Event Diagram

The diagram below indicates the customer connects to the internet to perform all the transactions after he logged in successfully then the information will receive the server to maintain the requirements, and it will send a copy of the data to the database and vice versa.

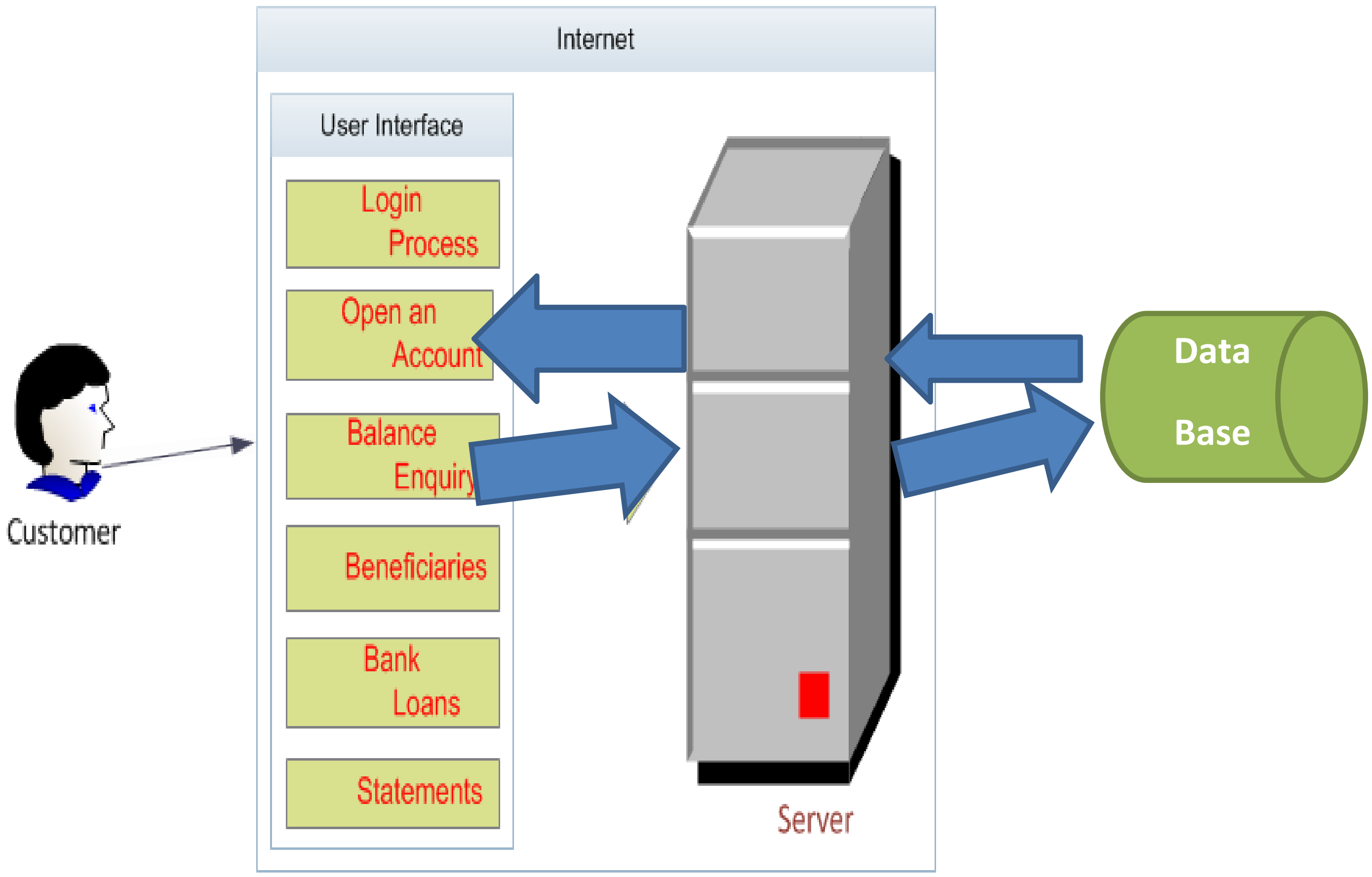


Figure-9 Event Diagram

5.2 Algrothmic Description

Registeration and opening new account:

Function register()

```
{
GetCustomer_information(name,surname….)
Valid =CheckInformation()
If  (Valid) then {
        Accountnumber= Generate_AccountNum()
        Insert(Accountnumber,name,…)
         Display (success)
}
Else
{
        Display_error (message)
}
```

Login_process()

```
Get_CustomerAuthentification(Accountnumber && password)
If (Accountnumber&&password=correct) then
{
        Display (transactions)
}
Elseif (Accountno&&password=wrong) then
{
        Display (Account Number or password are mismatched)
}
Else
{
        Display (Register now)
}
```

Viewing_Balance()

```
Login_process()
Display (AccountBalance)
```

Beneficiary()

```
Beneficiary_process()
If (AccountBalance=sufficient)
{
        Make (payment)
        Display (Update_Account_balance)
}
Else
{
        Display (insufficient)
}
```

Chapter 6

IMPLEMENTATION

In the previous chapter, an overview of the implementation of how the project looks like was given.

In this chapter, the tools I have used are: PHP, MySQL, XAMPP as a server, and https for the security of the site.

PHP: Is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

It's an open source for web development. So I have got some php files that are related to the database.

MySQL: Is the most popular open-source database system. The data in MySQL is stored in database objects called tables.

A table is a collection of related data entries and it consists of columns and rows.

XAMPP: Is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

SSL: Is the standard security technology for establishing an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browsers remain private and integral.

To be able to create an SSL connection a web server requires an SSL Certificate. When you choose to activate SSL on your web server you will be prompted to complete a number of questions about the identity of your website and your company. Your web server then creates two cryptographic keys - a Private Key and a Public Key.

The Public Key does not need to be secret and is placed into a Certificate Signing Request (CSR) - a data file also containing your details. You should then submit the CSR. During the SSL Certificate application process, the Certification Authority will validate your details and issue an SSL Certificate containing your details and allowing you to use SSL.

Your web server will match your issued SSL Certificate to your Private Key. Your web server will then be able to establish an encrypted link between the website and your customer's web browser.

Code Documentation

The source given below shows the documented code of the beneficiary transaction of the system.

```
<?php                                        // to start php file

        $accountno=$_GET['accountNo'];

        $benaccountno=$_GET['benaccountno'];

        $amount=$_GET['amount'];
```

I initialized the accountno, benaccountno, and amount as an accountNo, benaccountno, and amount.

This is how to connect the database as shown below:

```
        $connect = mysql_connect("localhost","root","") or die ("Couldn't connect!");

        mysql_select_db("banking") or die("Couldn't find db");
```

banking: is the database of my system.

“”: means I did not have any password.

This query is for selecting or reading the account number from an account table.

```
        $query = "SELECT * FROM account WHERE AccountNo='" . $accountno;

        $query .="'";
```

And now it checks if the available balance is less than the amount you want to pay the beneficiary.

```
        if ($balance > $amount)

        {

        $balance=$balance-$amount;


        $sql= "UPDATE account SET AccountBalance='" .$balance . "' WHERE ";

        $sql.= "AccountNo='" .$accountno ."'";
        $result = mysql_query($sql);
```

For this condition is to copy the beneficiary payment to the transaction table.

```
        if ($result > 0){
       $sql2 = "INSERT into transaction
(AccountNo,TypeOfTransaction,BeneficiaryAccountNo,Amount)";
        $sql2. = "VALUES ('";
        $sql2 .= $accountno;
        $sql2 .= "','";
        $sql2 .= "','EFT Transfer','";
        $sql2 .= $benaccountno;
        $sql2 .= "','";
        $sql2 .= $amount;
        $sql2 .= "')";
        $result = mysql_query($sql2);


        ?>                              //to close php file
```

Chapter 6

TESTING

The previous chapter focused on the implementation of the application. It gave a detailed documentation of the code used and explained how each part works to make the application usable, functional and how each component contributes to the project. The testing chapter will discuss the usability and the functionality the system and then evaluate the results. The process of testing is documented for both the user interface and the system.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under the test. [6]

USABILITY TESTING

We did basic usability testing with 7 participants from computer science and some information systems students.

Most of the participants said it was easy and after seeing the demo they did not need spend much time learning it. We used some Specific questions using likert scale and open ended questions and for each function/feature, it was asked how the user experienced performing that task.

Task 1: Registering as well as opening a new account

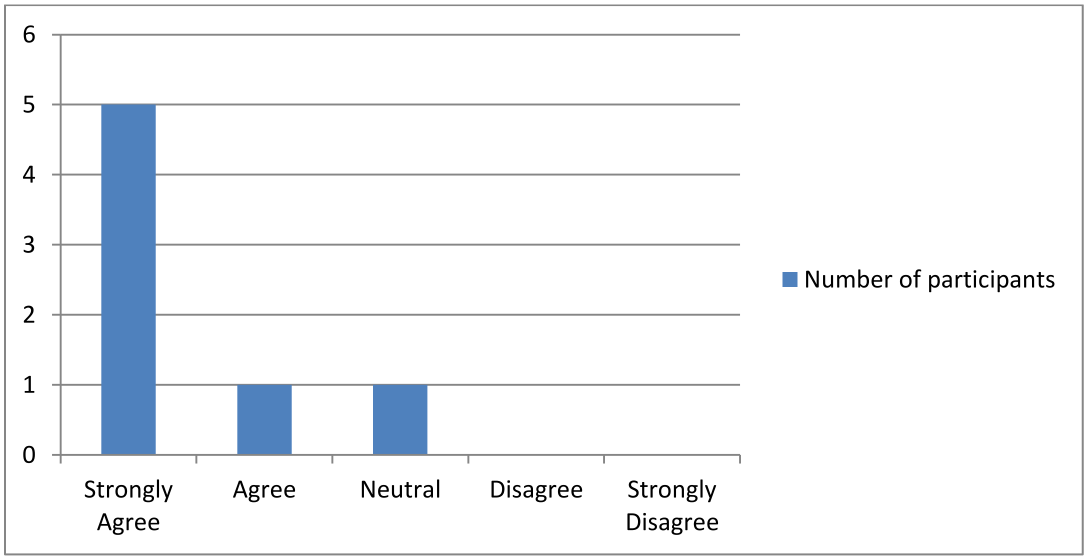


Figure-10 The graph shows how the participants were rated for that specific task.

From the results obtained here we found that the above mentioned task is very easy to understand and easy to learn as shown in figure 10.

Task 2: Balance Enquiry

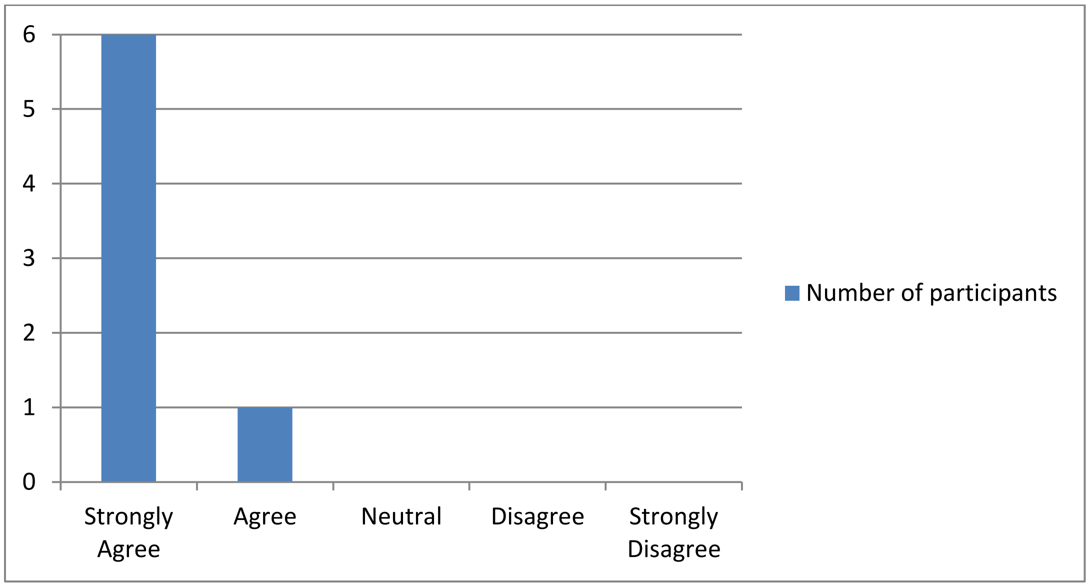


Figure-11 A graph shows how the participants were rated for the balance enquiry.

From the results obtained here we found that the above mentioned task is very easy to understand and easy to learn as shown in figure 11.

Task 3: Inter-account transfer (EFT)

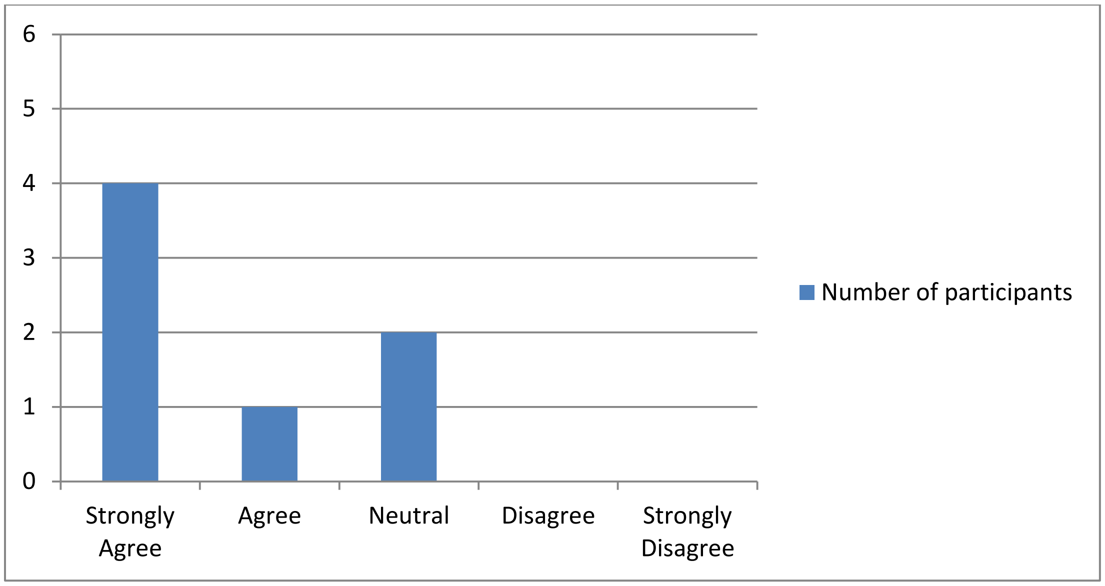


Figure-12 This graph shows how the participants were rated for the EFT transaction.

From the results obtained here we found that four participants were strongly agreed that it was very easy to use or to understand while only one participant had just agree and other two participants said it is neutral.

Task 4: Beneficiary Payments

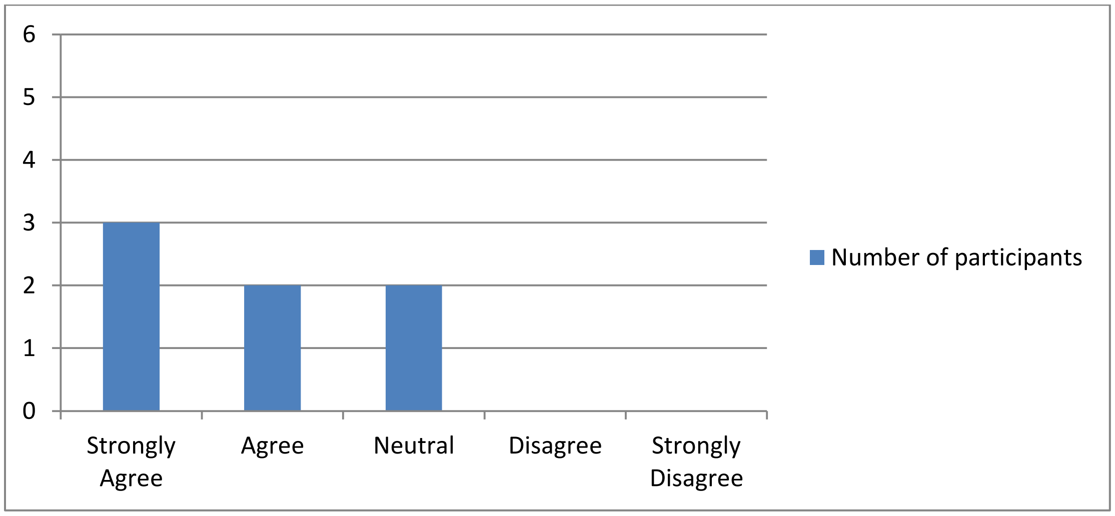


Figure-13 The graph shows how the participants were rated for the beneficiary payment function.

From the results obtained here we found that three participants were strongly agreed that it was very easy to use or to understand while two participant had just agree and other two participants rated as a neutral.

Task 5: Monthly Statement

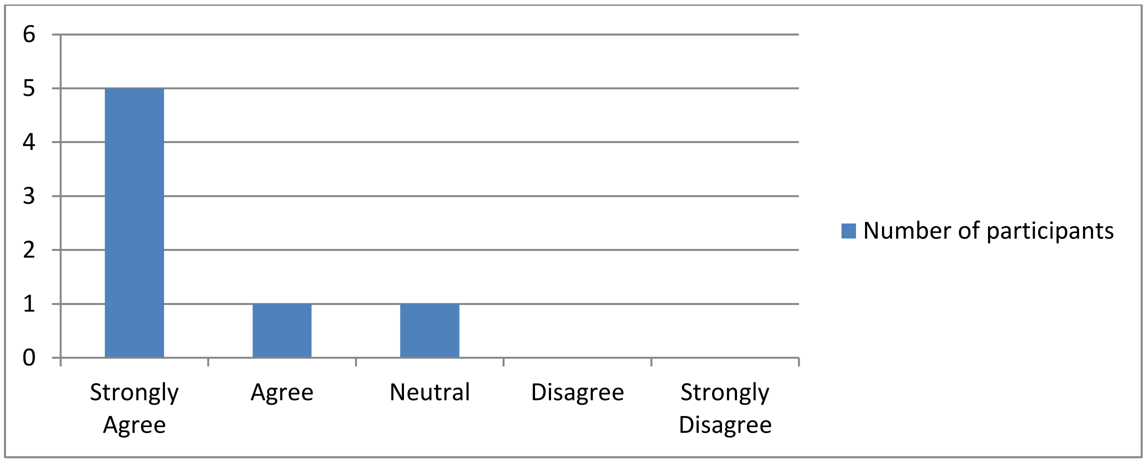


Figure-14 The graph shows how the participants were rated for the monthly statement feature.

From the results obtained here we found that five participants were strongly agreed that it was very easy to use or to understand while one participant had just agree and the other participant rated as a neutral.

Task 6: Apply for a Loan

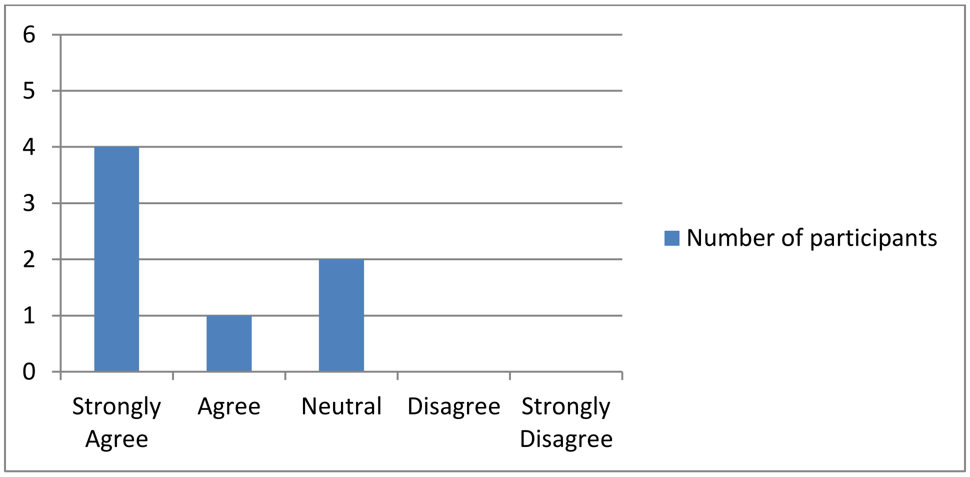


Figure-15 The graph shows how the participants were rated for the bank loan feature.

From the results obtained here we found that four participants were strongly agreed that it was very easy to use or to understand while one participant had just agreed and the other two participants rated as a neutral.

FUNCTIONAL TESTING

To test the system its functional features, I used a tool called AppPerfect WebTest, which is suitable to test the functional testing for the web applications. It provides support for recording web browser events and then replay them automatically and gives you a report [4].

Results

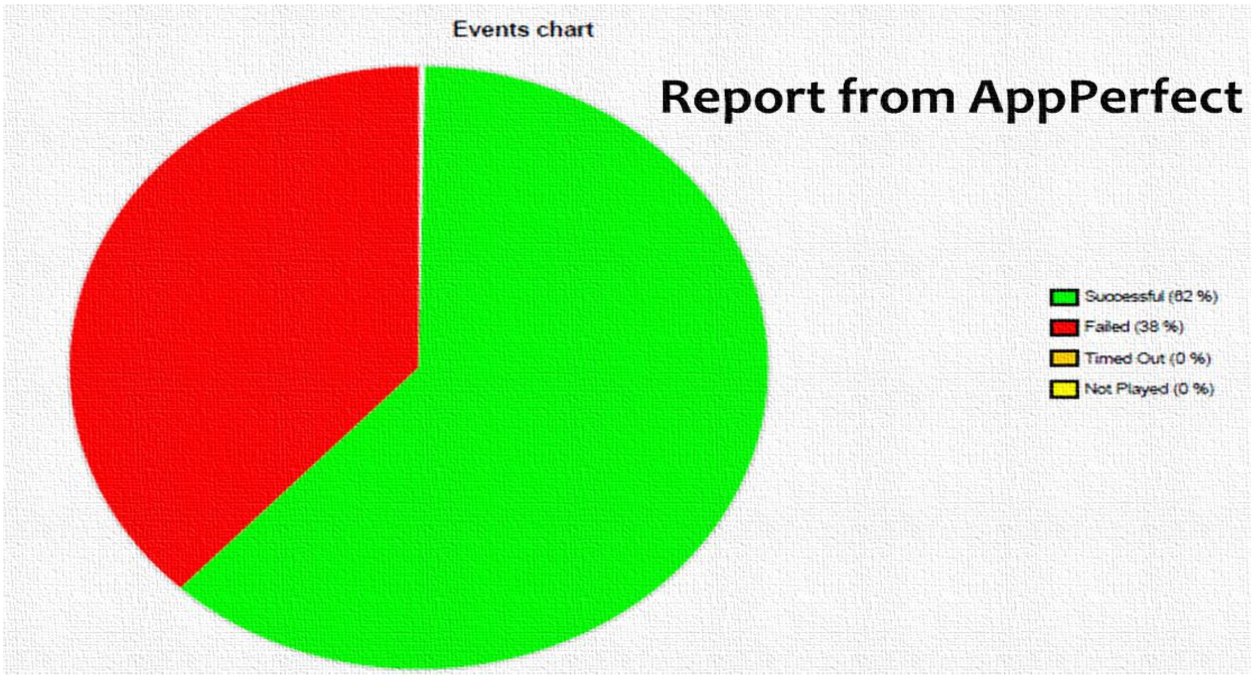
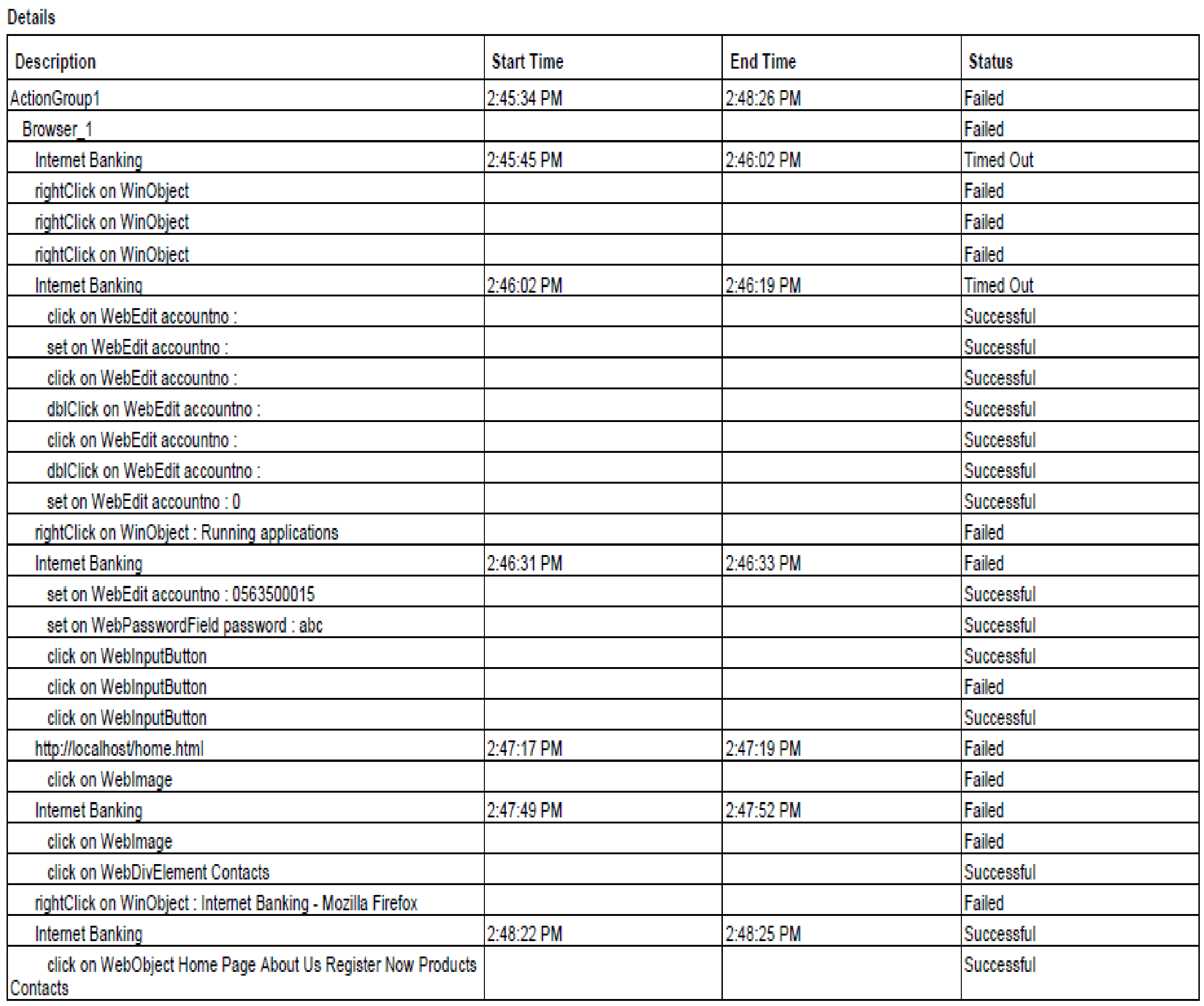


Figure-16 This report shows the functional testing system.

In above events chart clearly shows that the functional testing have 62% were successful while 38% were failed because of the untrusted ssl certificates.

Reports



Details

Description	Start Time	End Time	Status
ActionGroup1	2:45:34 PM	2:48:26 PM	Failed
Browser_1			Failed
Internet Banking	2:45:45 PM	2:46:02 PM	Timed Out
rightClick on WinObject			Failed
rightClick on WinObject			Failed
rightClick on WinObject			Failed
Internet Banking	2:46:02 PM	2:46:19 PM	Timed Out
click on WebEdit accountno :			Successful
set on WebEdit accountno :			Successful
click on WebEdit accountno :			Successful
dblClick on WebEdit accountno :			Successful
click on WebEdit accountno :			Successful
dblClick on WebEdit accountno :			Successful
set on WebEdit accountno : 0			Successful
rightClick on WinObject : Running applications			Failed
Internet Banking	2:46:31 PM	2:46:33 PM	Failed
set on WebEdit accountno : 0563500015			Successful
set on WebPasswordField password : abc			Successful
click on WebInputButton			Successful
click on WebInputButton			Failed
click on WebInputButton			Successful
http://localhost/home.html	2:47:17 PM	2:47:19 PM	Failed
click on WebImage			Failed
Internet Banking	2:47:49 PM	2:47:52 PM	Failed
click on WebImage			Failed
click on WebDivElement Contacts			Successful
rightClick on WinObject : Internet Banking - Mozilla Firefox			Failed
Internet Banking	2:48:22 PM	2:48:25 PM	Successful
click on WebObject Home Page About Us Register Now Products Contacts			Successful

Figure-17 The details of the results performed during the functional testing.

Chapter 7

USER GUIDE

Getting Started

The first page you get is the home page which contains several functions/features for the Internet Banking Application. In order to login you fill in only two fields which are account number you have and the password you selected when you registered.

Task 1: Register as well as open a new account

- Click the register now on the home page at left side
- Fill all the fields with your password
- Submit after you are done

Task 2: Balance Enquiry

- Login from home page with your account number and your password
- View Your balance at the top right

Task 3: Electronic Fund Transfer (EFT)

- Type the account number you want to transfer an amount
- Put the amount you want to transfer
- Click **Transfe**r to send the amount

Task 4: Beneficiary Payments

- Type the bank name of the beneficiary payee
- Enter the beneficiary account number you want to pay it
- Select the company.
- Put an amount you would like to pay it
- Click **pay** to finish

Task 5: Bank Statement

- Select the account from which you want a statement
- To view the record of the transaction, click **view statement** on top of the transaction page.

Task 6: Apply for a Loan

- Click **Apply for a Loan** to request for a bank loan
- Fill all the fields completely
- Click **Apply** to submit

APPENDIX

This code is for transaction page , it contains html, php, and javascript.

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link rel="stylesheet" href="style.css" type="text/css" media="all" />

<link href="css.css" rel="stylesheet" type="text/css" />

            <link rel="stylesheet" href="css/validationEngine.jquery.css" type="text/css" media="screen"
title="no title" charset="utf-8" />


                        <link rel="stylesheet" href="css/template.css" type="text/css" media="screen" title="no
title" charset="utf-8" />

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3/jquery.min.js"
type="text/javascript"></script>


                        <script src="js/jquery.validationEngine-en.js" type="text/javascript"></script>

                        <script src="js/jquery.validationEngine.js" type="text/javascript"></script>

<title>Internet Banking</title>

<link href="css.css" rel="stylesheet" type="text/css" />

<script type="text/javascript">


function postTodatabase(){
        if (window.XMLHttpRequest)

                {// code for IE7+, Firefox, Chrome, Opera, Safari

                xmlhttp=new XMLHttpRequest();

        }

        else

        {// code for IE6, IE5

                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

        }


        var fsend = "clientname=";


        var tosend = document.getElementById('name').value;


        fsend += tosend;


    fsend += "&surname=";

        tosend = document.getElementById('surname').value;

        fsend += tosend;


        fsend += "&initial=";

        tosend = document.getElementById('initial').value;

        fsend += tosend;


        fsend += "&id_passport=";
```

```
tosend = document.getElementById('id_passport').value;

fsend += tosend;


/*

fsend += "&mobileno=";

tosend = document.getElementById('mobileno').value;

fsend += tosend;


fsend += "&telephoneno=";

tosend = document.getElementById('telephoneno').value;

fsend += tosend;


fsend += "&address=";

tosend = document.getElementById('address').value;

fsend += tosend;


fsend += "&email=";

tosend = document.getElementById('email').value;

fsend += tosend;


fsend += "&accounttype=";

tosend = document.getElementById('accounttype').value;

fsend += tosend;


fsend += "&sex=";
```

```
            tosend = document.getElementById('sex').value;
            fsend += tosend;

            fsend += "&dob=";
            tosend = document.getElementById('dob').value;
            fsend += tosend;

            fsend += "&password=";
            tosend = document.getElementById('password').value;
            fsend += tosend;*/


 xmlhttp.open("GET","register.php?"+fsend,false);

 xmlhttp.send();

 document.getElementById("maincont").innerHTML=xmlhttp.responseText;

}

function getBalance(acctNo){
if (window.XMLHttpRequest)
                        {// code for IE7+, Firefox, Chrome, Opera, Safari
                        xmlhttp=new XMLHttpRequest();
            }
```

```
        else

        {// code for IE6, IE5

                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

        }


        var fsend = "accNo=";


        var tosend = acctNo;


        fsend += tosend;


        xmlhttp.open("GET","getbalance.php?"+fsend,false);


        xmlhttp.send();


        document.getElementById("detailst").innerHTML=xmlhttp.responseText;



}


function logout(){

        if (window.XMLHttpRequest)

                {// code for IE7+, Firefox, Chrome, Opera, Safari

                xmlhttp=new XMLHttpRequest();

        }
        else

        {// code for IE6, IE5

                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

        }

        xmlhttp.open("GET","logout.php?",false);


        xmlhttp.send();


        window.location = "home.html";

}


function getCookie(c_name)

{

var i,x,y,ARRcookies=document.cookie.split(";");

for (i=0;i<ARRcookies.length;i++)

{

  x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));

  y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);

  x=x.replace(/^\s+|\s+$/g,"");

  if (x==c_name)

    {

    return unescape(y);

    }

  }

}
```

```
function checkLogin(){
   var accountNo =getCookie("accountNo");
  if (accountNo!=null && accountNo!="")
  {
              getBalance(accountNo);
  }
else
  {
             alert("Please login in-order to perform a transaction");
             window.location = "home.html";
  }
}
/*
$(document).ready(function() {



                                    // SUCCESS AJAX CALL, replace "success: false," by:     success : function() {
callSuccessFunction() },
                                    $("#form1").validationEngine({
                                                ajaxSubmit: true,
                                                            ajaxSubmitFile: "ajaxSubmit.php",
                                                            ajaxSubmitMessage: "Thank you, We will contact you soon !",
                                                success :  false,
                                                failure : function() {}
                                    })
```

```
                });*/

function beneficiarypayment(){

        if (window.XMLHttpRequest)
                {// code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
        }
        else
        {// code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
        var accountNo =getCookie("accountNo");

        var fsend = "benaccountno=";
        var tosend = document.getElementById('benaccountno').value;
        fsend += tosend;

        fsend += "&amount=";
        tosend = document.getElementById('amount').value;
        fsend += tosend;
```

```
            fsend += "&accountNo=";

            tosend = accountNo;

            fsend += tosend;



            xmlhttp.open("GET","beneficiary.php?"+fsend,false);


            xmlhttp.send();


            //document.getElementById("detailst").innerHTML=xmlhttp.responseText;

            var resp=xmlhttp.responseText;


            if(resp == "paid"){

            alert('Your Transaction has been Successfully Completed.');


                        getBalance(accountNo);


            }else{

                        alert("You current balance is less than the amount you want to pay");

            }

}




            function getBalance(acctNo){

if (window.XMLHttpRequest)
```

```
        {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }

    var fsend = "accNo=";

    var tosend = acctNo;

    fsend += tosend;

    xmlhttp.open("GET","getbalance.php?"+fsend,false);

    xmlhttp.send();

    document.getElementById("detailst").innerHTML=xmlhttp.responseText;


}
```

```
function transferamount(){

        if (window.XMLHttpRequest)
                {// code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
        }
        else
        {// code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
        var accountNo =getCookie("accountNo");

        var fsend = "benaccountno=";
        var tosend = document.getElementById('benaccountnos').value;
        fsend += tosend;

        fsend += "&company=";
        tosend = document.getElementById('company').value;
        fsend += tosend;

        fsend += "&amount=";
        tosend = document.getElementById('amounts').value;
        fsend += tosend;

        fsend += "&accountNo=";
        tosend = accountNo;

        fsend += tosend;


        xmlhttp.open("GET","test.php?"+fsend,false);


        xmlhttp.send();


        //document.getElementById("detailst").innerHTML=xmlhttp.responseText;

        var resp=xmlhttp.responseText;


        if(resp == "done"){

        alert('You have paid the Beneficiary.');


                getBalance(accountNo);


        }else{

                alert("You current balance is less than the amount you want to pay");

        }

}



</script>



</head>
```

```
<body onload="checkLogin()">

<div id="maincont">

<div id="wrapperdetail">

  <div id="form-divdetail">

    <form class="form" id="form1" action='register.php' method='GET'>

            <h1 align="center"><label >DETAILS</label></h1>

            <div>

            <h3>

            <label>

            <span class="acctholder">Account Holder</span>

            <span class="acctype">Account Type</span>

            <span class="accNo">Account Number</span>

            <span class="balance">Available Balance</span>

            </label>

            </h3>

    </div>

            <br/>

            <hr/>


            <label>

            <div id ="detailst">

            <span class="acctype"></span>

            <span class="accNo"></span>

            <span class="balance"></span>

            </div>
            </label>


            <br/>

            <br/>

            <hr/>

              <p class="submit">

      <input type="button" value="Logout" onclick="logout()"/>

    </p>

   </form>


  </div id="form-div">


  </div>

  </div>


  <div id="AASDFASDFASDF" class="container">


   <div align="right">

  <h2><a href="loan.php">Apply For A Loan</a></h2>

 </div>
```

```
<br />

<p>

            <div id="wrapper2">

  <div id="form-div2">

    <form class="form" id="form2">

            <h3 align="center"><label >EFT Transaction</label></h3>

            <hr />



              <p class="name">

       <input name="benaccountno" type="text"
class="validate[required,custom[onlyLetter],length[0,100]] text-input" id="benaccountno" />

                        <label for="name">To Account No.</label>

     </p>







              <p class="name">



       <input name="amount" type="text" class="validate[required,custom[onlyLetter],length[0,100]]
text-input" id="amount"  />

                        <label for="name">Amount</label>

     </p>



              <p class="submit">

       <input type="button" value="Transfer" onclick="beneficiarypayment()"/>

     </p>
  </form>


 </div >

 </div>


 <p>

       <div id="wrapper5">

 <div id="form-div5">

  <form class="form" id="form2">

       <h3 align="center"><label >Beneficiary Payment</label></h3>

       <hr />

        <p class="name">


    <input name="bankname" type="text" class="validate[required,custom[onlyLetter],length[0,100]]
text-input" id="name"  />

              <label for="name">Bank Name</label>

   </p>

        <p class="name">

    <input name="benaccountno" type="text"
class="validate[required,custom[onlyLetter],length[0,100]] text-input" id="benaccountnos" />

              <label for="name">Beneficary Account No.</label>

   </p>


        <p class="name">

              <select name="company" class="validate[required,custom[onlyLetter],length[0,100]]
text-input" id="company" />

    <option value="telkom">Telkom</option>
```

```
      <option value="edgars">Edgars</option>

                    <option value="UWC">UWC</option>

                    </select>

                    <label
for="benef">&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp
&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp

                    &nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbspCompany</label>

    </p>



           <p class="name">



      <input name="amount" type="text" class="validate[required,custom[onlyLetter],length[0,100]]
text-input" id="amounts"  />

                    <label for="name">Amount</label>

    </p>



           <p class="submit">

      <input type="button" value="Pay" onclick="transferamount()"/>

    </p>

   </form>



  </div >

  </div>





<div id="wrapper8">

  <div id="form-div8">
```

```
<form class="form" id="form2" action="statement3.php ">

       <h2 align="center"><label >View Statement</label></h2>



Enter your Acc number<br><br />

<input name="AccountPin" type="text" id="AccountPin" />

<p class="submit">

    <input type="submit" name="submit" value="View" />

   </p>

</form>

</div>

<p/>

</div>




</body>

</html>
```

QUESTINARIES

1) Have you successfully completed each task?

Yes ☐ No ☐

If NO, Please specify which task have been unsuccessfully completed and why:

____

____

2) What do you like most about the interface?

____

3) What do you dislike most about the interface?

____

4) What do you think about the security of the system?

____

5) Are there any task which can be improved?

____

Evaluator Name:______________________________________ Date:_______________

Comments/Suggestions:____

____

____

Bibliography

1. (n.d.). Retrieved 2012, from http://www.programmingportal.in/2010/05/onilne-banking-system-sequence-diagram.html
2. (n.d.). Retrieved 2012, from www.php.net
3. (n.d.). Retrieved 2012, from SSL: http://info.ssl.com/article.aspx?id=10241
4. (2012). Retrieved november 2012, from AppPerfect: www.appperfect.com
5. *likert scale.* (2012). Retrieved november 2012, from Wikipedia: http://en.wikipedia.org/wiki/Likert_scale

6. C, B. (1994). Relational Approach. *Conceptual database Design*.

7. *internet-banking-for-many-benefits.* (n.d.). Retrieved 2012, from http://smartbisplan.com/internet-banking-for-many-benefits/

8. *Limitiations-of-E-Banking.* (n.d.). Retrieved 2012, from www.scribd.com/doc/53669376/36

9. M., S. (1999). Adoption of Internet Banking . *The International Journal of Bank Marketing.*, 324-334.

10. *Synopsis-Internet-Banking.* (n.d.). Retrieved 2012, from http://www.scribd.com/pgailani/d/36880246-Synopsis-Internet-Banking

11. *Thread-insurance-on-internet-9437.* (n.d.). Retrieved 2012, from www.seminarprojects.com

12. Williams, B. (2008). Retrieved 2012, from databaseanswers: www.databaseanswers.org/data_models/online_banking/facts.htm