**A**

**MINI PROJECT REPORT ON**

**“MULTI BANKING TRANSACTION SYSTEM”**

*Submitted in partial fulfilment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

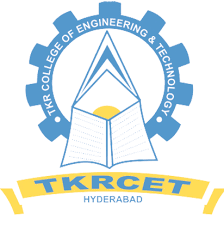
**COMPUTER SCIENCE & ENGINEERING**

|  |  |
| --- | --- |
|  | **By** |
| **PRASHANTH PASUNURI** | **17K91A05G5** |
| **MITTAPALLY JASWANTH** | **17K91A05D9** |
| **KANNEBOINA BHARATH KUMAR** | **18K95A0519** |
|  |  |

**UNDER THE GUIDANCE OF**

**Mrs. D LAKSHMI PRASANNA**

**Assistant Professor, CSE Dept**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TKR COLLEGE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

##### (Approved By AICTE. Affiliated to JNTUH. Accredited By NBA Accredited by NAAC with 'A' Grade" Recognized under 2(f) and 12(B) of UGC Act 1956")

##### Medbowli,Meerpet,Saroornagar, Hyderabad-500097.

(2017 – 2021)

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

##### (Approved By AICTE. Affiliated to JNTUH. Accredited By NBA Accredited by NAAC with 'A' Grade" Recognized under 2(f) and 12(B) of UGC Act 1956")

##### Medbowli,Meerpet,Saroornagar, Hyderabad-500097.

(2017 – 2021)

**CERTIFICATE**

This is to certify that the Project report titled **“MULTI BANKING TRANSACTION SYSTEM”** that is being submitted by **PRASHANTH PASUNURI (17K91A05G5), MITTAPALLY JASWANTH (17K91A05D9), KANNEBOINA BHARATH KUMAR (18K95A0519)** in partial fulfilment of the requirements for the awardof the degree of ***Bachelor of Technology in Computer Science & Engineering*** is a record of bonafide work carried out by them.

The result of investigation enclosed in this report have been verified and found satisfactory.

**Mrs.D LAKSHMI PRASANNA Dr. A . SURESH RAO**

**Assistant Professor, CSE HOD & professor, CSE**

**INTERNAL GUIDE**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success.

We extend our deep sense of gratitude to Principal, **Dr. D. V. RAVI SHANKAR**, TKR College of Engineering and Technology, Meerpet, for permitting us to undertakethis project.

We are indebted to **Dr. A. SURESH RAO,** HOD, Computer Science & engineering, TKR College of Engineering and Technology, Meerpet for his support throughout our Project.

We are indebted to **Mrs. D. LAXMI PRASANNA**, Associate Professor, Computer Science & engineering, TKR College of Engineering and Technology, Meerpet, for her support and guidance throughout our Project.

Finally, we express thanks to all those who have helped us in successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement.

|  |  |
| --- | --- |
| **P.PRASHANTH** | **17K91A05G5** |
| **M.JASHWANTH** | **17K91A05D9** |
| **K.BHARATH KUMAR** | **18K95A0519** |
|  |  |

**ABSTRACT**

**The Multi Banking Transaction System Interface is targeted to the future banking solution for the users who is having multiple bank accounts in multiple banks. This interface integrates all existing banks and provides business solutions for both retail and corporate.**

**This system acts as a standard interface between the clients and all the banks. By using this portal any client who maintain accounts in various banks can directly log on to Multi Banking System Interface and make any kind of transactions. In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.**

**LIST OF CONTENTS**

**S.NO PARTICULAR PAGE.NO**

1. INTRODUCTION 1

2. LITERATURE SURVEY 2

3. SYSTEM ANALYSIS 3

4. SYSTEM DESIGN 7

5. SYSTEM IMPLEMENTATION 18

6. TESTING 26

7. SCREENSHOTS 29

CONCLUSION AND FUTURE SCOPE 43

REFERENCES 44

**LIST OF FIGURES**

**FIGURE.NO FIGURE NAME PAGE.NO**

3.3 SDLC (umbrella) 4

3.4.1 System architecture 5

3.4.2 Architecture flow 6

3.4.3 URL patterns 6

4.2.1 Context level DFD 8

4.2.2 Level 0 DFD for admin 8

4.2.3 Level 0 DFD for customer 8

4.2.4 Level 0 DFD for bank admin 9

4.2.5 Level 1 DFD for chosen bank 9

4.2.6 Level 1 DFD for banker login 9

4.2.7 Level 2 DFD for Money Transfer 9

4.2.8 Level 2 DFD for Transaction Reports 10

4.3.1 Use Case Diagrams for Administrator 11

4.3.2 Use Case Diagram for Bank Administrator 11

4.3.3 Use Case Diagram for A/C Holder 12

4.3.4 Class Diagram 12

4.3.5 Sequence Diagram for Admin 13

4.3.6 Sequence Diagram for Bank Admin 13

4.3.7 Sequence Diagram for Customer 14

4.3.8 State Diagram for Admin 14

4.3.9 State Diagram for Customer 15

4.3.10 State Diagram for Bank Admin 15

4.4 ER diagram 15

5.1 Compiling and interpreting java source code 19

**LIST OF SCREENSHOTS**

**FIGURE .NO NAME OF THE FIGURE PAGE. NO**

7.1 Main page 29

7.2 Admin login page 29

7.3 Page displayed after admin login 30

7.4 New user request details 30

7.5 Page displayed when admin accepts user request 31

7.6 Customer registration page 31

7.7 User main page 32

7.8 New user registration page 32

7.9 User login page 33

7.10 Page displayed when user enter wrong details 33

7.11 Customer login page 34

7.12 Selecting the bank 34

7.13 New user creating the account page 35

7.14 Page displayed after user entering all details and

Clicking Submit button 35

7.15 User login page with choose bank 36

7.16 Page displayed after user login successfully 36

7.17 Total list of customers in selected bank page 37

7.18 List of accounts available in selected bank page 37

7.19 Transactions page 38

7.20 Transactions pending page 38

7.21 Customers page 39

7.22 Account information page 39

7.23 Transfer amount page 40

7.24 After clicking transfer link, it asks the account details 40

7.25 Transactions reports page 41

7.26 Accepted transactions page 41

7.27 Rejected transactions page 42

7.28 Pending transactions page 42

**LIST OF TABLES**

**TABLE.NO NAME OF TABLE PAGE. NO**

2.1 literature survey 2

**INDEX**

**S.NO PARTICULAR PAGE.NO**

**1. INTRODUCTION 1**

1.1 OBJECTIVE 1

1.2 PURPOSE OF PROJECT 1

1.3 EXSTING SYSTEM 1

1.4 PROPOSED SYSTEM 1

**2. LITERATURE SURVEY 2**

**3. SYSTEM ANALYSIS 3**

3.1 THE STUDY OF THE SYSTEM 3

3.2 INPUT & OUTPUT REPRESENTATION 3

3.3 PROCESS MODEL USED WITH JUSTIFICATION 4

3.4 SYSTEM ARCHITECTURE 5

**4. SYSTEM DESIGN 7**

4.1 SYSTEM REQUIREMENTS 7

4.2 DATA FLOW DIAGRAMS 8

4.3 UML DIAGRAMS 10

4.4 ER DIAGRAMS 15

4.5 MODULES 16

**5. SYSTEM IMPLEMENTATION 18**

5.1 ABOUT LANGUAGE 18

5.2 SAMPLE CODE 22

**6. TESTING 26**

6.1 INTRODUCTION OF TESTING 26

6.2 TYPES OF TESTING 26

6.3 TEST CASES 27

6.4 TEST RESULTS 28

**7. SCREENSHOTS 29**

**CONCLUSION AND FUTURE SCOPE 43**

**REFERENCES 44**

1. **INTRODUCTION**
   1. **OBJECTIVE**

The ‘Multi Banking System’ Interface is targeted to the future banking solution for the users who have multiple bank accounts in different banks. This interface integrates all existing banks and provides business solutions for both retail and corporate. System Involves

* This interface integrates all existing banks and provides business solutions for both retailers and corporate.
* This system acts as a standard interface between the clients and the banks
* Users who have accounts in various banks can login here and can make any kind of transactions.
* In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.
  1. **PURPOSE OF THE PROJECT**

Its purpose is to create a common portal for multiple banks. So users can login here and can. Access any of the available banks and can do required transactions.

* 1. **EXISTING SYSTEM & DISADVANTAGES**

Currently we are having lot of banks in the market and any person can do transactions of any individual bank either manually or in online. But no one can do all banks transactions in a single portal or in single bank. This is the main disadvantage in existing system to avoid this problem we are introducing “multi banking system”.

* 1. **PROPOSED SYSTEM & ITS ADVANTAGES**

The Multi Banking System Interface is targeted to the future banking solution for the users who is having multiple bank accounts in multiple banks. This interface integrates all existing banks and provides business solutions for both retail and corporate. This system acts as a standard interface between the clients and all the banks, By using this portal any client who maintain accounts in various banks can directly log on to Multi Banking System Interface and make any kind of transactions. In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.

**2. LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | PAPER | AUTHOR | YEAR | REMARKS |
| 1 | An architecture for integrated multi banking solution | G.Sree Rekha,  V.K Agarval | 2011 | How to remove security threats |
| 2 | MVC design pattern for the multi framework distributed applications using xml, spring and structs framework | Praveen Gupta | 2010 | How to use J2EE and multi  framework and MVC architecture |
| 3 | Architecture of E-commerce systems based on J2EE and MVC pattern | Yanfang Wang,Chunyan Guo,Lei song | 2009 | Architecture for E –banking |
| 4 | A model for securing E-banking authentication process | Antonio san Martino,Xavier perramon | 2008 | Authentication for secured banking |

Table 2.1 literature survey

**3. SYSTEM ANALYSIS**

**3.1 STUDY OF THE SYSTEM**

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. The GUI’S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The ‘administrative user interface’ concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. These interfaces help the administrators with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The ‘operational or generic user interface’ helps the end users of the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information in a customized manner as per the included flexibilities.

**3.2 INPUT & OUTPUT REPRESENTETION**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**OUTPUT DESIGN:**

In general are:

* External Outputs whose destination is outside the organization.
* Internal Outputs whose destination is with in organization and they are the User’s main interface with the computer. Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs
* Operational outputs whose use is purely with in the computer department.
* Interface outputs, which involve the user in communicating directly with the system.

**3.3 PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. Team formation
3. Project Specification Preparation

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

Assesment

Training

Integration & system testing

Delivary/Installation

ACCEPTANCE TEST

Requirements Gathering

Fig 3.3 SDLC (Umbrella model)

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**3.4 SYSTEM ARCHITECTURE**

**Context Diagram:**

Fig 3.4.1 System architecture

**Architecture flow:**

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tier architecture.

**SERVER**

**USER**

**Data base**

**REQUEST**

**RESPONSE**

Fig 3.4.2 Architecture flow

**URL Pattern:**

**Presentation Layer**

**SERVLETS AT THE SERVER SIDE**

**URL Request sent through the browser**

**Response sent from the servlet**

**DATABASE**

**Verifying or updating the database** **through a statement**

**Reply from the database according to the statement**

Fig3.4.3 URL pattern

URL pattern represents how the requests are flowing through one layer to another layer and how the responses are getting by other layers to presentation layer through server in architecture diagram.

**4. SYSTEM DESIGN**

**INTRODUCTION:**

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

**4.1 SYSTEM REQUIREMENTS**

**SOFTWARE REQUIREMENTS**:

Operating System : Windows

Technology : Java/j2ee (JDBC, Servlets, JSP)

Web Technologies : Html, JavaScript, CSS

Web Server : Tomcat

Database : MY SQL

Software’s : J2SDK1.5, Tomcat 5.5, Oracle 9i

**HARDWARE REQUIREMENTS**:

Hardware : Pentium based systems with a minimum of P4

RAM : 256MB (minimum)

**Additional Tools:**

HTML Designing : Dream weaver Tool

Development Tool kit : My Eclipse

**4.2 DATA FLOW DIAGRAMS**

Data flow diagram will act as a graphical representation of the system in terms of interaction between the system, external entities, and process and how data stored in certain location.

**DFD DIAGRAMS:**

**Context Level DFD**

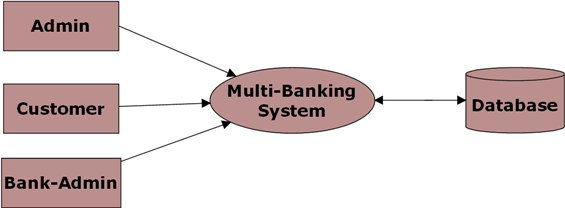
****

Fig 4.2.1 context level DFD

**Level 0 DFD for Admin:**

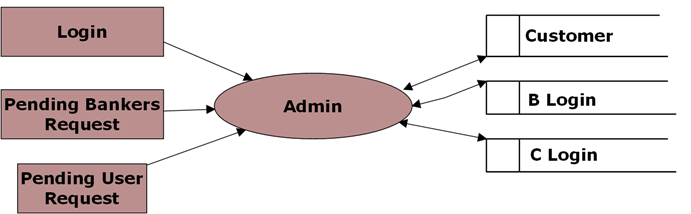
****

Fig 4.2.2 level 0 DFD for admin

**Level 0 DFD for Customer:**

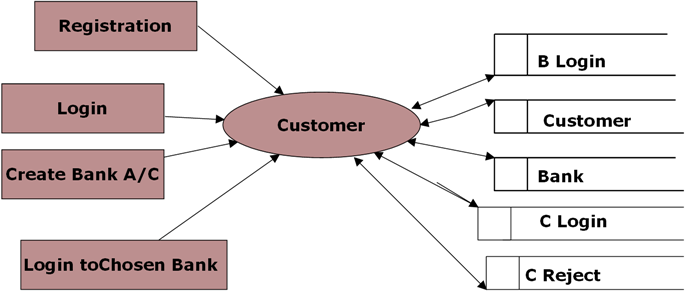


Fig 4.2.3 level 0 DFD for customer

**Level 0 DFD for Bank Admin:**

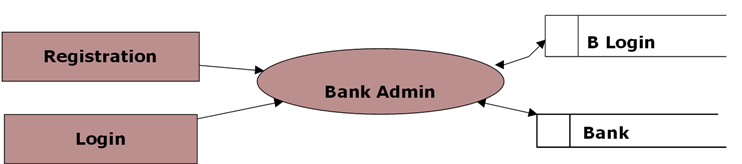
****

Fig 4.2.4 level 0 DFD for bank admin

**Level 1 DFD for Chosen Bank:**

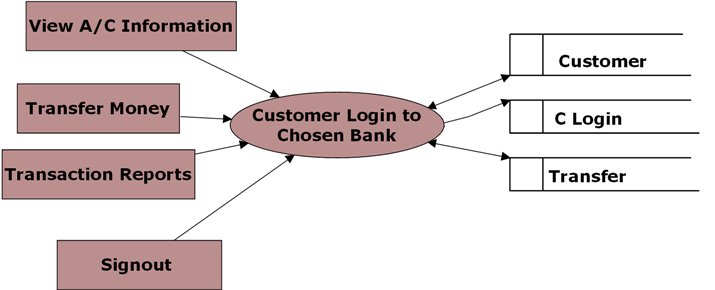
****

Fig 4.2.5 level 1 DFD for chosen bank

**Level 1 DFD for Banker Login:**

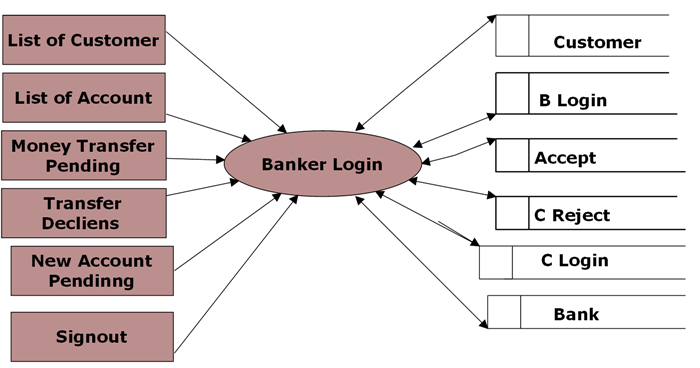
****

Fig 4.2.6 level 1 DFD for banker login

**Level 2 for Money Transfer:**

****

Fig 4.2.7 level 2 DFD for money transfer

**Level 2 DFD for Transaction Reports:**

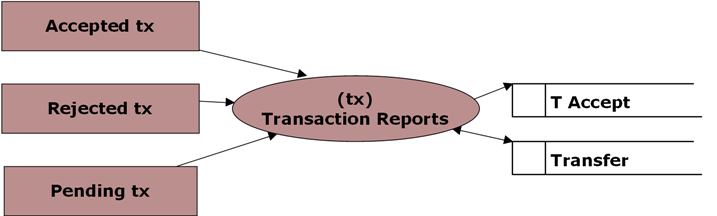
****

Fig 4.2.8 level 2 DFD for transaction reports

**4.3 UML DIAGRAMS**

**Unified Modeling Language:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* User Model View
  + 1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
* Structural model view
  + 1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioural as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**UML DIAGRAMS**

**Use Case Diagram for Administrator:**

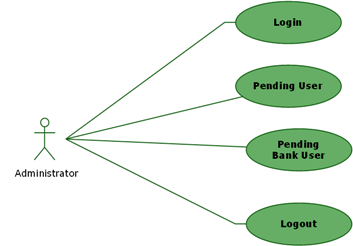


Fig 4.3.1 Use case diagram for administrator

**Use Case Diagram for Bank Administrator**:

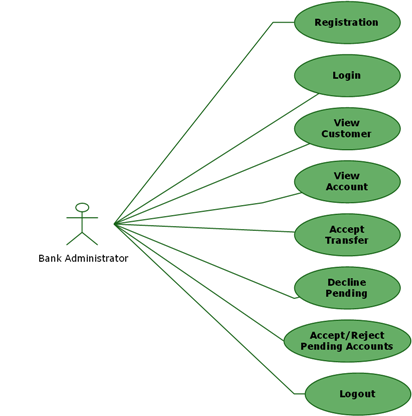


Fig 4.3.2 use case diagram for bank administrator

**Use Case Diagram for A/C Holder:**

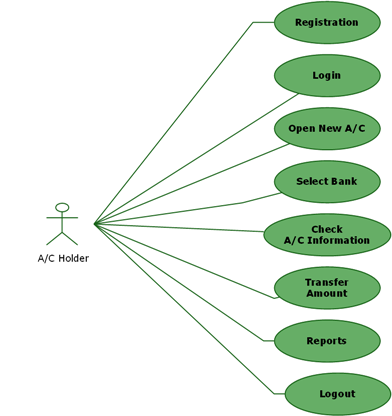
****

Fig 4.3.3 use case diagram for account holder

**Class Diagram:**

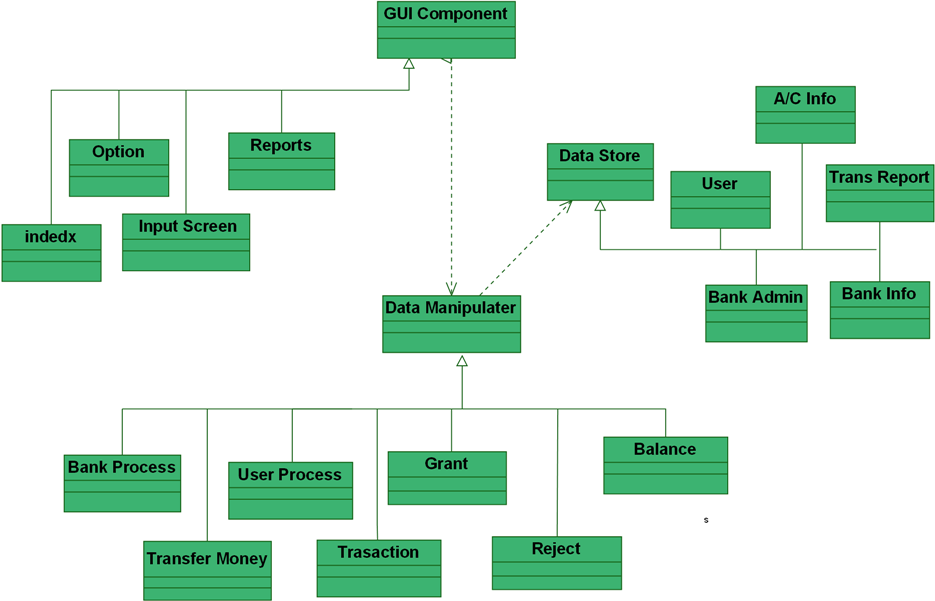
****

Fig 4.3.4 class diagram

**Sequence Diagram for Admin:**

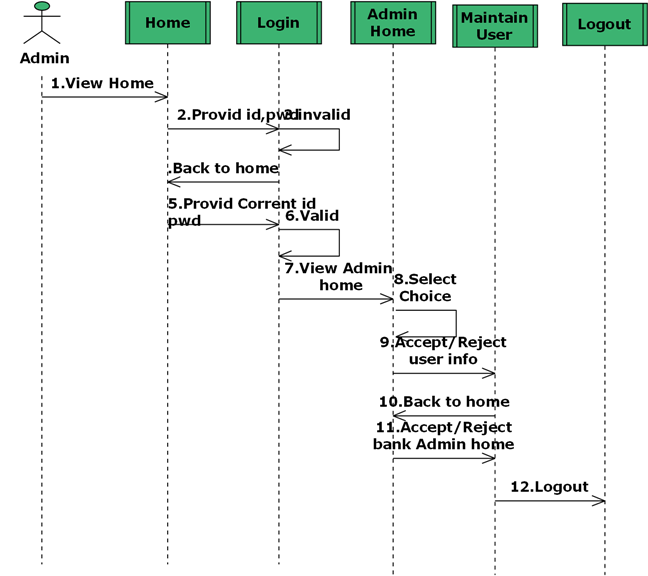
****

Fig4.3.5 sequence diagram for admin

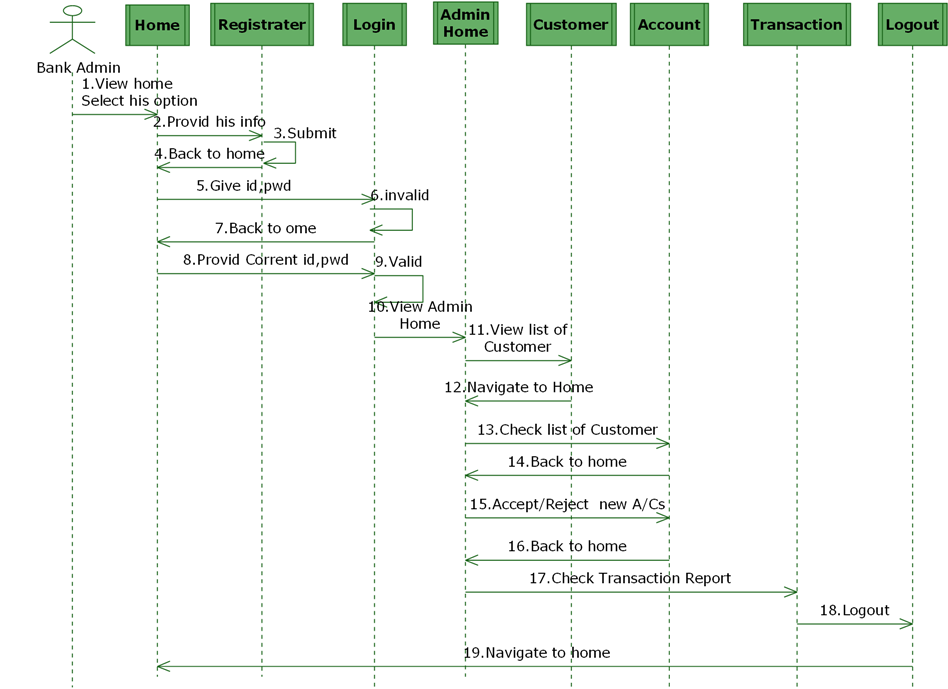
**Sequence Diagram for Bank Admin: **

Fig4.3.6 sequence diagram for bank admin

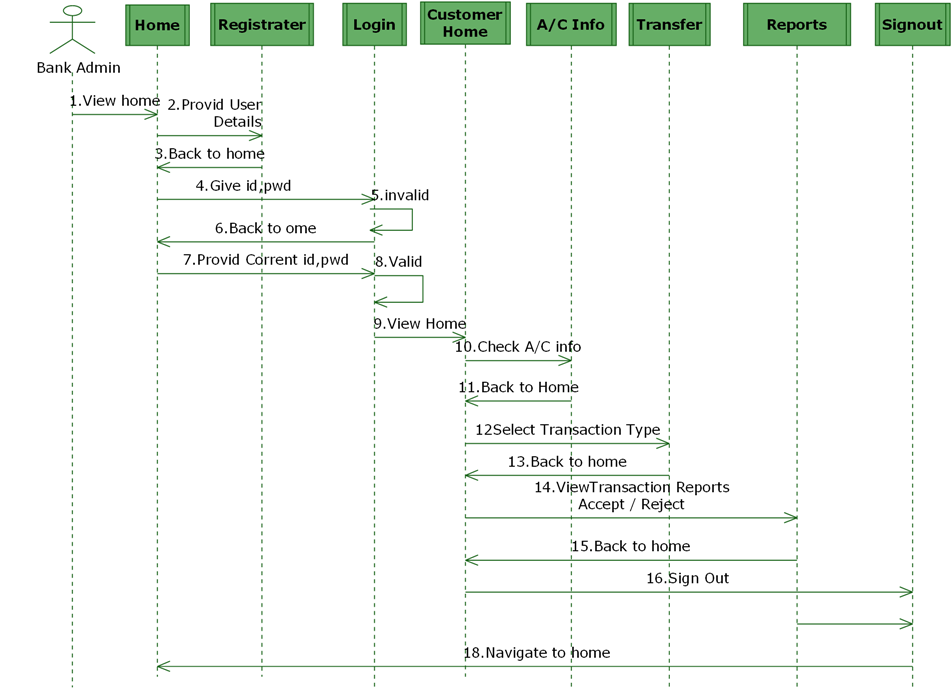
**Sequence Diagram for Customer: **

Fig4.3.7 sequence diagram for customer

**State Diagram for Admin**

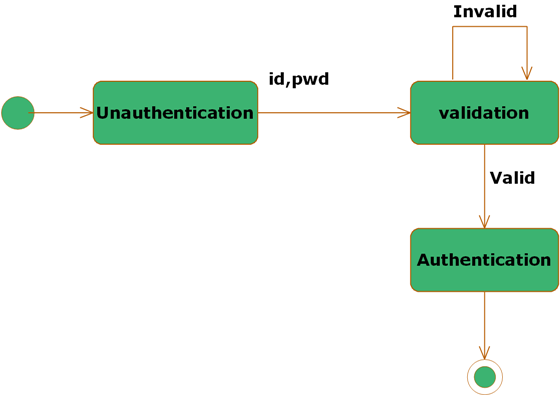
****

Fig4.3.8 state diagram for admin

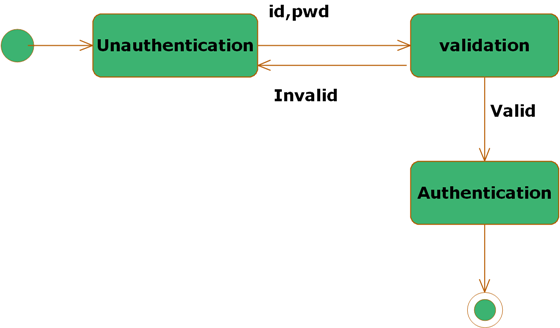
**State Diagram for Customer **

Fig 4.3.9 state diagram for customer

**State Diagram for Bank Admin:**

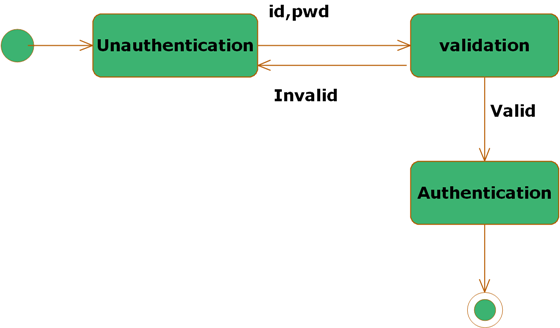
****

Fig 4.3.10 state diagram for bank admin

**4.4. ER DIAGRAMS**

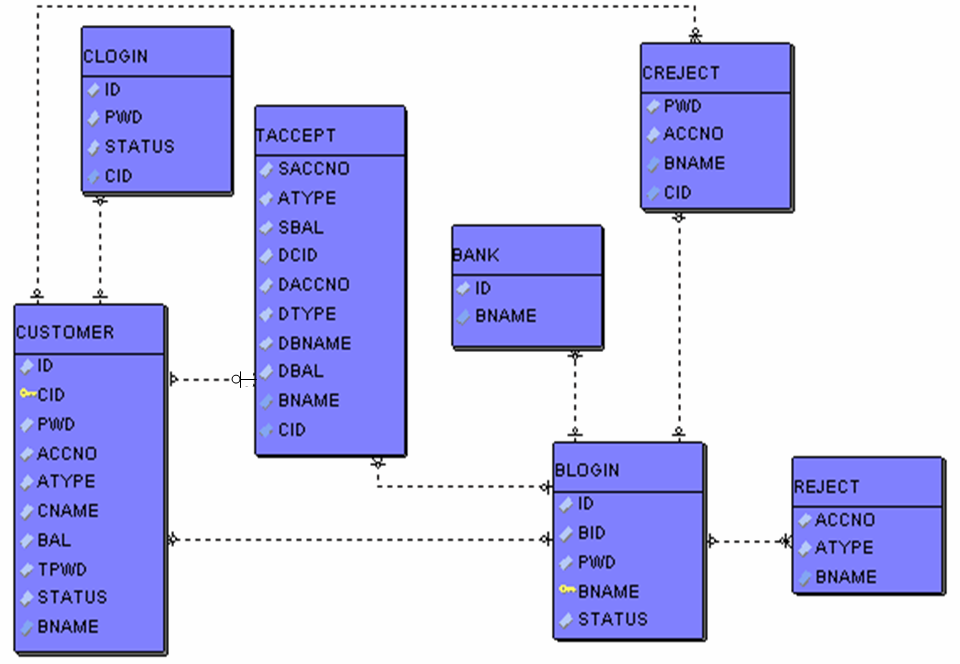


Fig4.4 ER diagram

**4.5 MODULES**

**1. Admin Module**

**2. Customer Module**

**3. Bank Admin Module**

**4. Reports Module**

**1. Admin Module:**

The admin module will be used by the administrator of this portal, admin can accept or reject the requests from the bankers, and also admin can accept or reject the requests from the users. The requests are in the form of bank registration, customer registration. This module is having following functionalities.

* **Pending Bankers Requests:** By using this functionality Administrator can give access permeations to all bankers who are registered in this portal.
* **Pending User Requests:** By using this functionality Administrator can give access permeations to all users who are registered in this portal.

**2. Customer Module:**

This module describes all about customers, by using this module any customer can do some operations like create a new account, view the account information, Transfer amount from one account to other account and customer can also see the Transaction Reports. This module consists following functionalities.

* **Create New Account:** By using this functionality user can create a new account in any bank by selecting bank name option.
* **View Account Information:** By using this functionality user view all his account details, this can be viewed by users who are having account in any bank.
* **Transfer Amount:** By using this functionality user can transfer money from his account to other accounts of same bank or other banks.
* **Transaction Reports:** By using this functionality user can get all his transaction reports like accepted transactions, rejected transactions and pending transactions.

**3. Bank Admin Module:**

This module deals with all transactions of bank management. By using this module bank staff can view all details of customers, they can go for any transactions of their customers and also they can give access permeations to all customers of that bank. This module consists following functionalities.

* **List of Customers:** By using this functionality Bank admin can get their entire customers list and their details.
* **List of Accounts:** By using this functionality Bank admin can get their entire customers list based on selected account type like saving account, current account etc.
* **Transfer Pending:** By using this functionality Bank admin can maintain money transfer details of customers.
* **Transfer Declines:** By using this functionality Bank admin can maintain money transfer rejected customer details.
* **New Accounts Pending:** By using this functionality Bank admin can maintain entire user details who are requesting for new account in that bank.

**4. Reports Module:**

In this module administrator will get different types of reports regarding customers like Number of customers of this portal and no. of banks registered in this portal. This module is controlled by administrator only.

**5. SYSTEM IMPLEMENTATION**

**5.1 ABOUT LANGUAGE**

**INTRODUCTION TO JAVA**

**About Java**

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral)language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control
* Finally Java is to Internet Programming where c was to System Programming.

**Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

**Java Architecture**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

**Compilation of code**

When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

**Compiling and interpreting java source code**

**Source code**

**Pc compiler**

**Macintosh compiler**

**SPARC Compiler**

**Java Byte code**

**Platform independent**

**Java interpreter**

**Java interpretermacintosh**

**)))**

**Java interpreter(SPARC)**

Fig 5.1 compiling and interpreting java source code

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

**SERVLETS/JSP**

**INTRODUCTION**

A Servlet is a generic server extension. a Java class that can be loaded Dynamically to expand the functionality of a server. Servlets are commonly used with web servers. Where they can take the place CGI scripts.A servlet is similar to proprietary server extension, except that it runs inside a Java Virtual Machine (JVM) on the server, so it is safe and portableServlets operate solely within the domain of the server. Unlike CGI and Fast CGI, which use multiple processes to handle separate program or separate requests, separate threads within web server process handle all servlets. This means that servlets are all efficient and scalable.

For example an HTTP servlet can be used to generate dynamic HTML content when you use servlets to do dynamic content you get the following advantages:

* They’re faster and cleaner then CGI scripts
* They use a standard API( the servlet API)
* They provide all the advantages of Java (run on a variety of servers without needing to be rewritten.

**ADVANTAGES OF THE SERVLET API**

One of the great advantages of the servlet API is protocol independent. It assumes nothing about:

* The protocol being used to transmit on the net
* How it is loaded
* The server environment it will be running in
* These quantities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the servlet API as well These include:
* It’s extensible-you can inherit all your functionality from the base classes made available to you
* Its simple small, and easy to use.

**THE SERVLET LIFE CYCLE**

The Servlet life cycle is one of the most exciting features of Servlets.This life cycle is a powerful hybrid of the life cycles used in CGI programming and lower-level NSAPI and ISAPI programming.Servlet life cycle is highly flexible Servers java significant leeway in how they choose to support servlets. The only hard and fast rule is that a servlet engine must confer to the following life cycle contact:

* Create and initialize the servlets
* Handle zero or more service from clients
* Destroy the servlet and then garbage collects it.

The most common and most sensible life cycle implementations for HTTP servlets are Single java virtual machine and astatine persistence.

**JDBC**

One can write a single program using the JDBC API and the JDBC is a Java API for executing SQL Statements(As a point of interest JDBC is trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java Programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API. Using JDBC, it is easy to send SQL statements to virtually program will be able to send SQL .statements to the appropriate database. The Combination of Java and JDBC lets a programmer writes it once and run it anywhere.

**JDBC-ODBC Bridge**

If possible use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge(that is, the Bridge native library, the ODBC driver manager library, library, the ODBC driver library, and the database client library).

**WHAT IS The JDBC-ODBE Bridge?**

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is joint development of Intervolve and Java Software.

**HTML**

Hypertext Mark up Language (HTML), the languages of the world wide web (WWW), allows users to produces web pages that included text, graphics and pointer to other web pages (Hyperlinks).HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Mark up Language),but Specialized to hypertext and adapted to the Web. The idea behind Hypertext one point to another point. We can navigate through the information based on out interest and preference. A mark up language is simply a series of items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

Html can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.HTML provides tags(special codes) to make the document look attractive.HTML provides are not case-sensitive. Using graphics, fonts, different sizes, colour, etc.Can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**JAVA SCRIPT**

JavaScript is a compact, object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page. And Livewire enables you to create server-based applications similar to common gateway interface (cgi) programs.

**5.2 SAMPLE CODE**

create table bank

(id number,

bname varchar2(100)

);

create table blogin

(id number,

bid varchar2(100),

pwd varchar2(100),

bname varchar2(100),

status number

);

create table clogin

(id number,

cid varchar2(100),

pwd varchar2(100),

status number

);

create table creject

(cid varchar2(100),

pwd varchar2(100),

accno varchar2(100),

bname varchar2(100)

);

create table reject

(cid varchar2(100),

accno varchar2(100),

atype varchar2(100),

bname varchar2(100)

);

create table taccept

(scid varchar2(100),

saccno varchar2(100),

atype varchar2(100),

sbname varchar2(100),

sbal number,

dcid varchar2(100),

daccno varchar2(100),

dtype varchar2(100),

dbname varchar2(100),

dbal number

);

create table transfer

(id varchar2(100),

saccno varchar2(100),

daccno varchar2(100),

amt number,

atype varchar2(100),

dtype varchar2(100),

tpwd varchar2(100),

sbank varchar2(100),

dbank varchar2(100)

);

create table customer

( id varchar2(100),

cid varchar2(100),

pwd varchar2(100),

accno varchar2(100),

atype varchar2(100),

cname varchar2(100),

bname varchar2(100),

bal number,

tpwd varchar2(100),

status number

);

**6. TESTING**

**6.1 INTRODUCTION TO TESTING**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**6.2 TYPES OF TESTING**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**UNIT TESTING**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two Strategies

**BLACK BOX TESTING**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.
* In this testing only the output is checked for correctness.
* The logical flow of the data is not checked.

**WHITE BOX TESTING**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**INTEGRATING TESTING**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**SYSTEM TESTING**

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

**ACCEPTANCE TESTING**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

**6.3 TEST CASES**

Testing can be done in two ways:

* Bottom up approach
* Top down approach

**BOTTOM UP APPROACH**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**TOP DOWN APPROACH**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**6.4 TEST RESULTS**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

**7. SCREENSHOTS**

**Main Page**

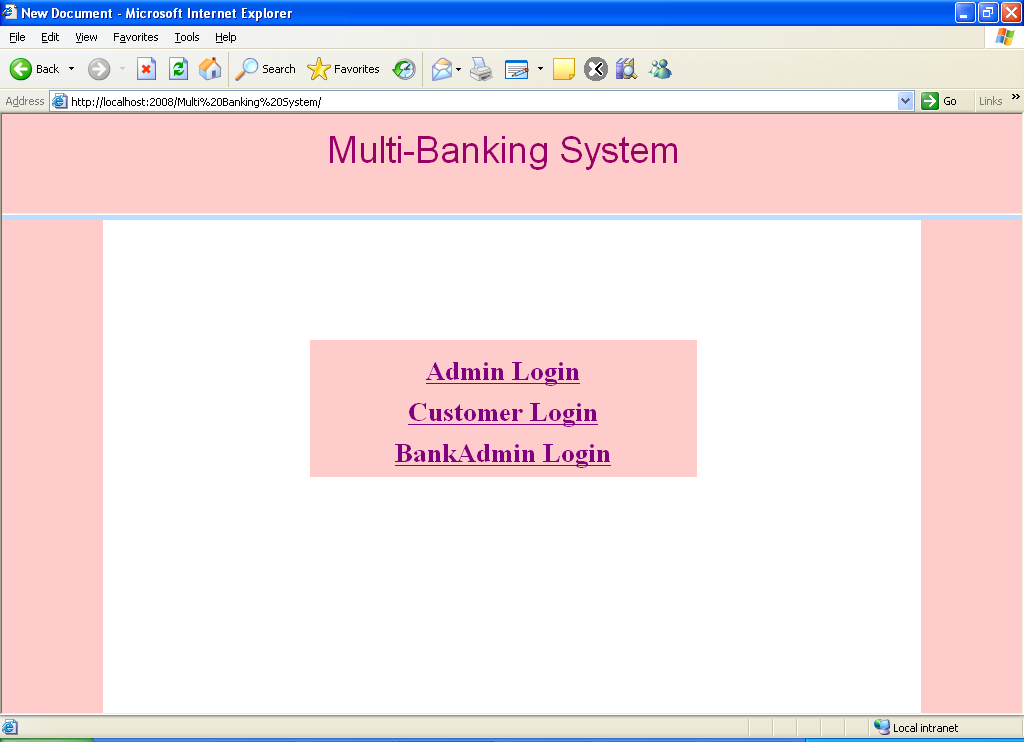
****

Fig 7.1 Main page

**Admin Login Page**

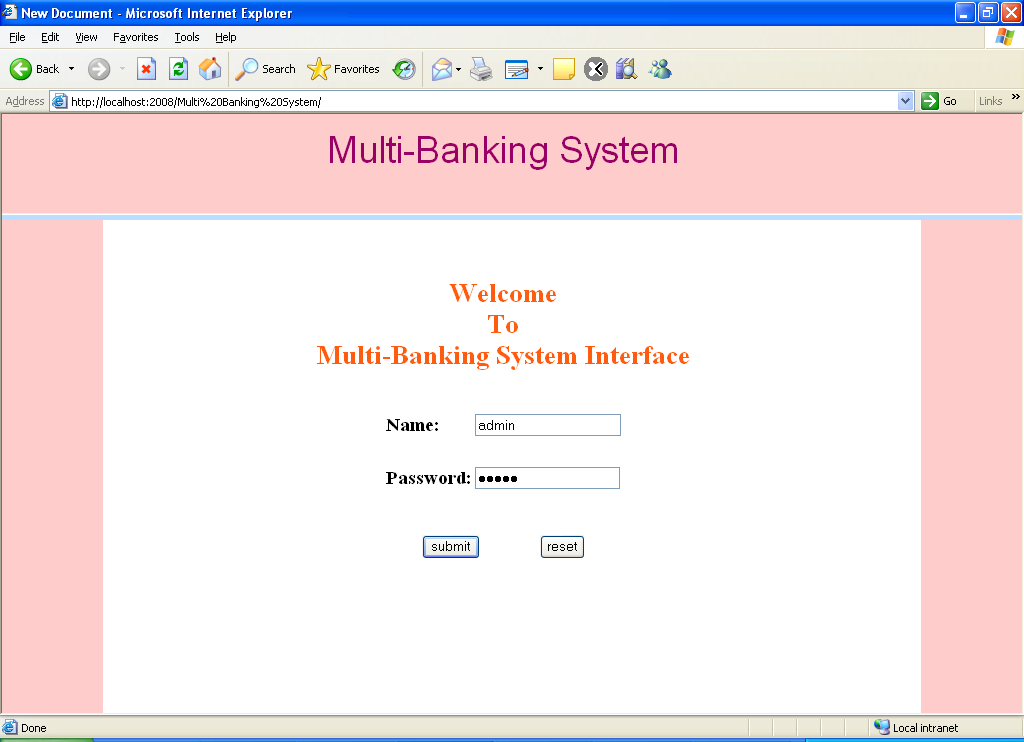
****

Fig 7.2 Admin page

**After admin login, this page was displays**

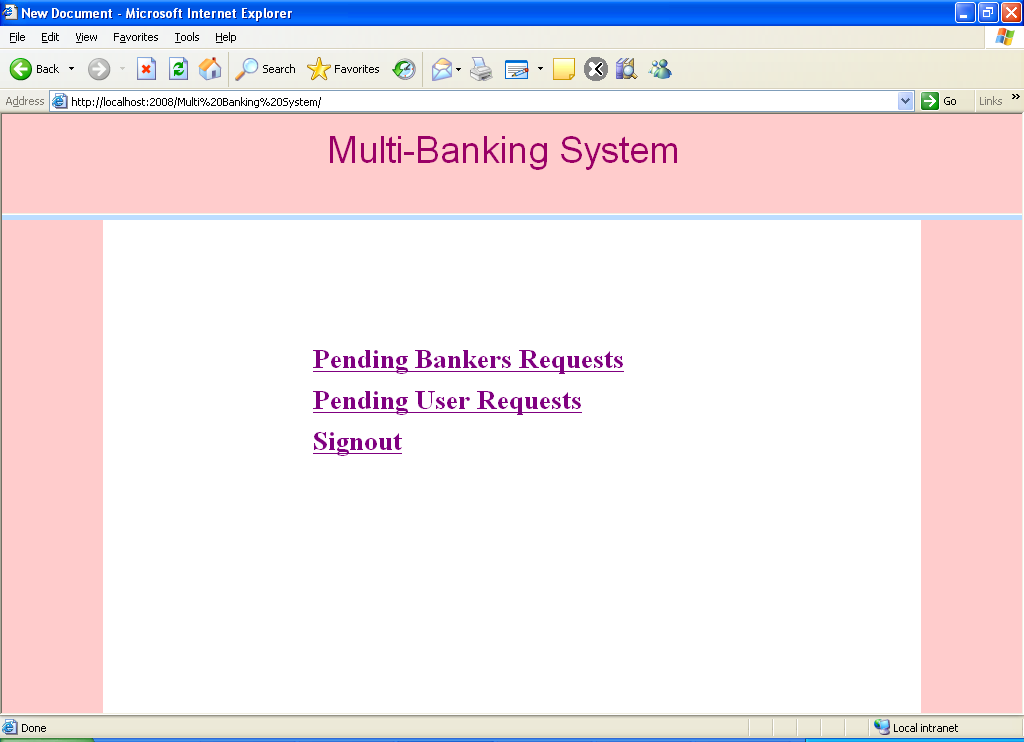
****

Fig 7.3 admin login page displayed successfully

**New User Request Details**

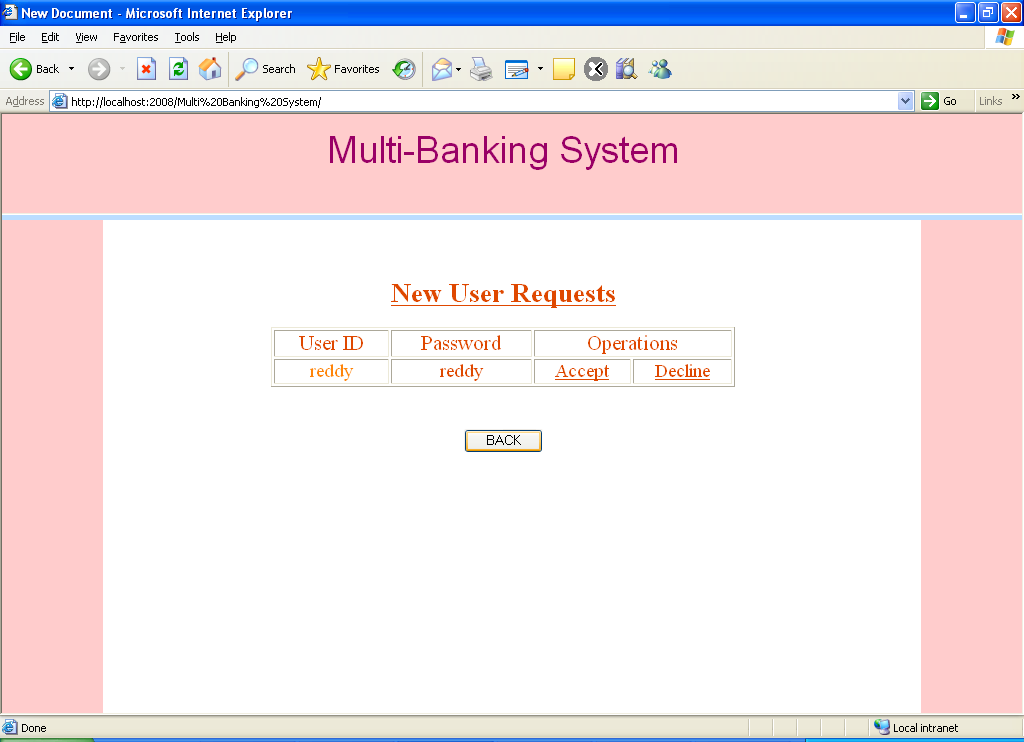
****

Fig 7.4 new user request details

**If the admin click accepts user then this page was displays**

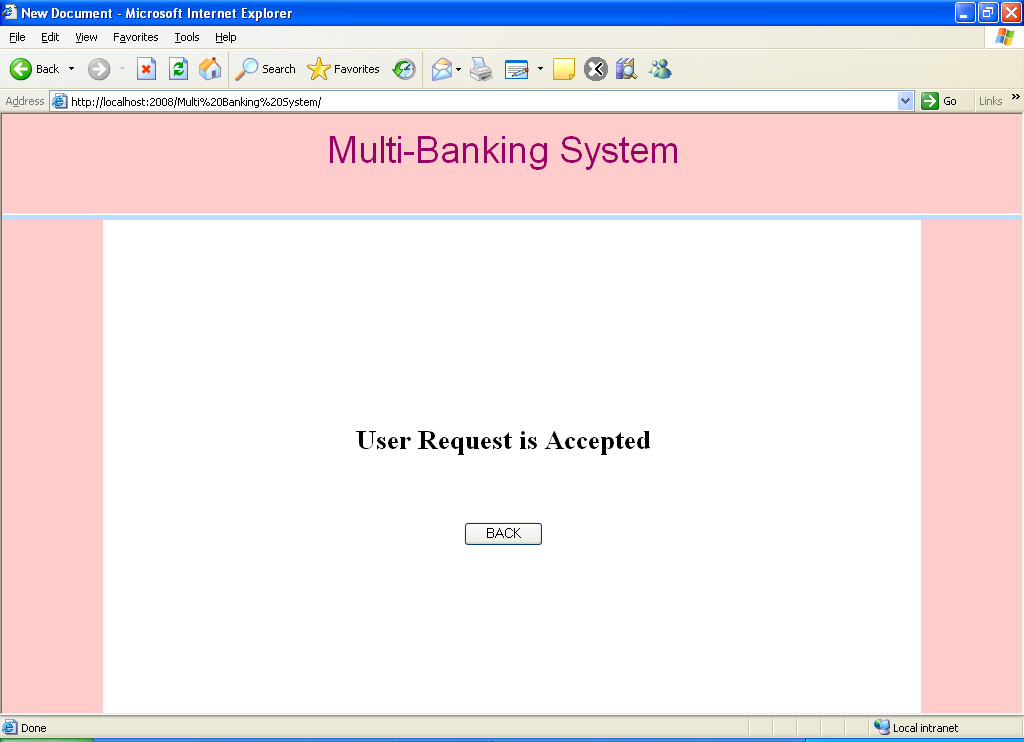
****

Fig 7.5 admin accepts user request

**Customer Registration Page**

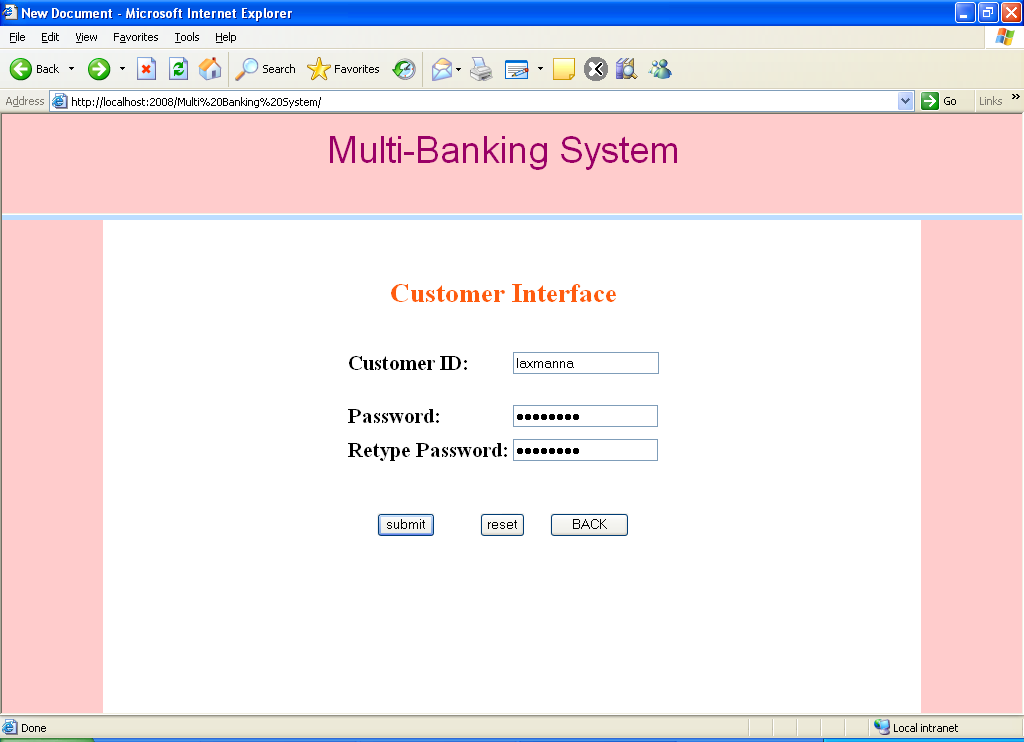
****

Fig 7.6 customer registration page

**User Main Page**

****

Fig 7.7 user main page

**New User Registration page**

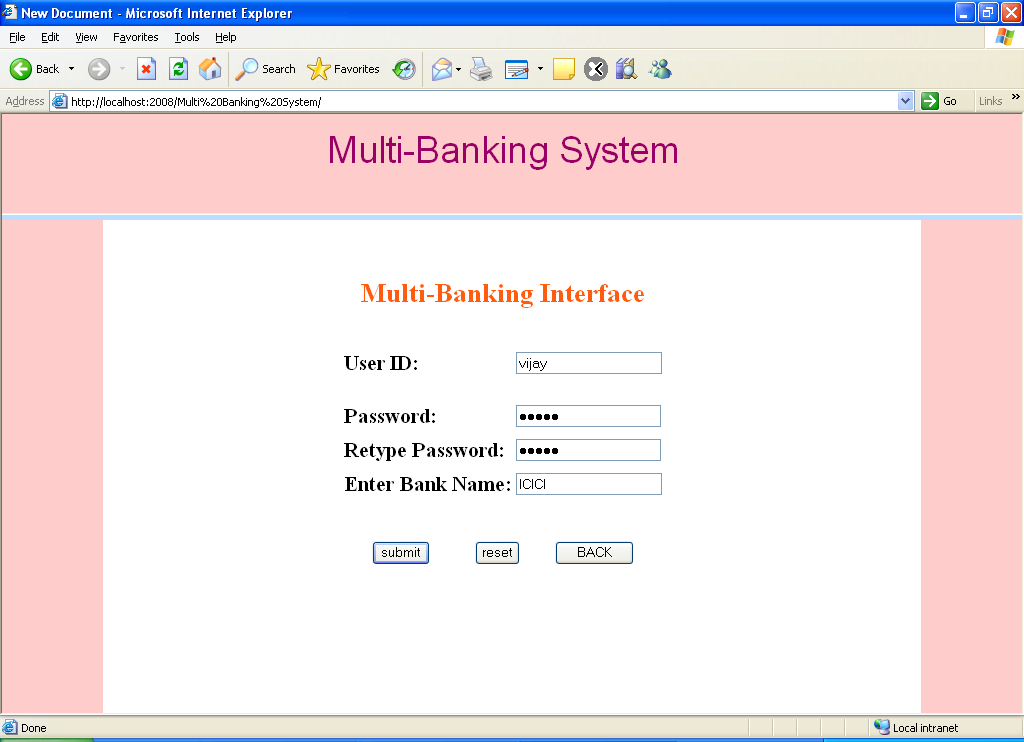
****

Fig 7.8 new user registration page

**User Login Page**

****

Fig 7.9 user login page

**If the user is enters wrong details then this page was displays**

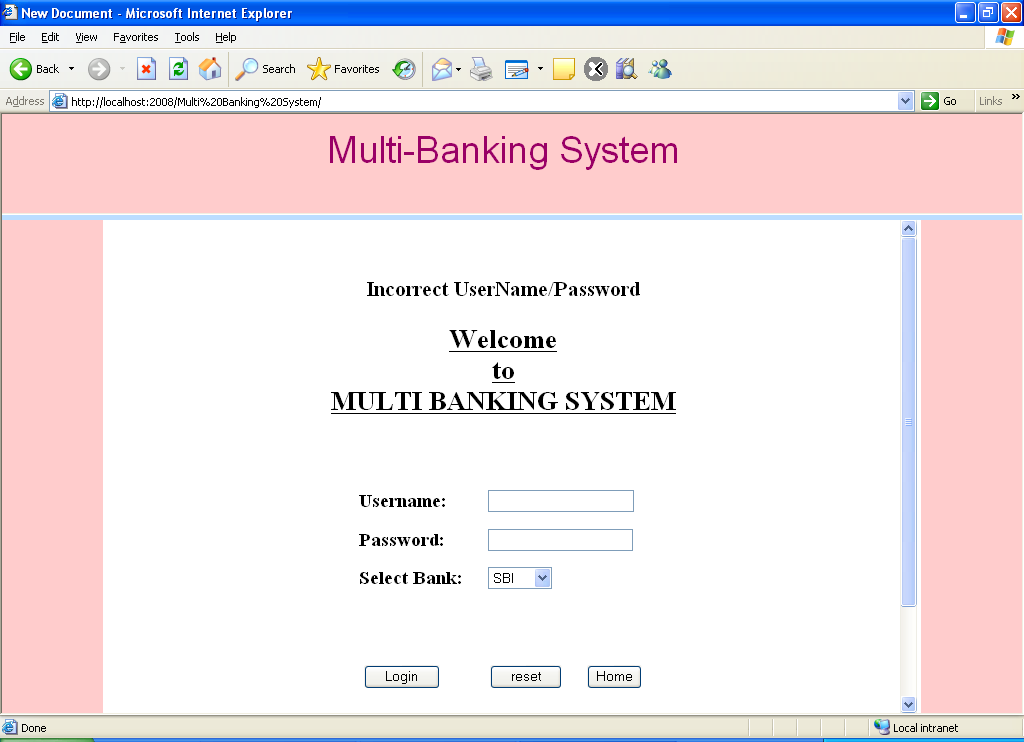
****

Fig 7.10 user entering wrong details

**Customer Login Page**

****

Fig 7.11 customer login page

**Choose the Bank that u processing page**

****

Fig 7.12 choosing the bank

**New User Create Account Page**

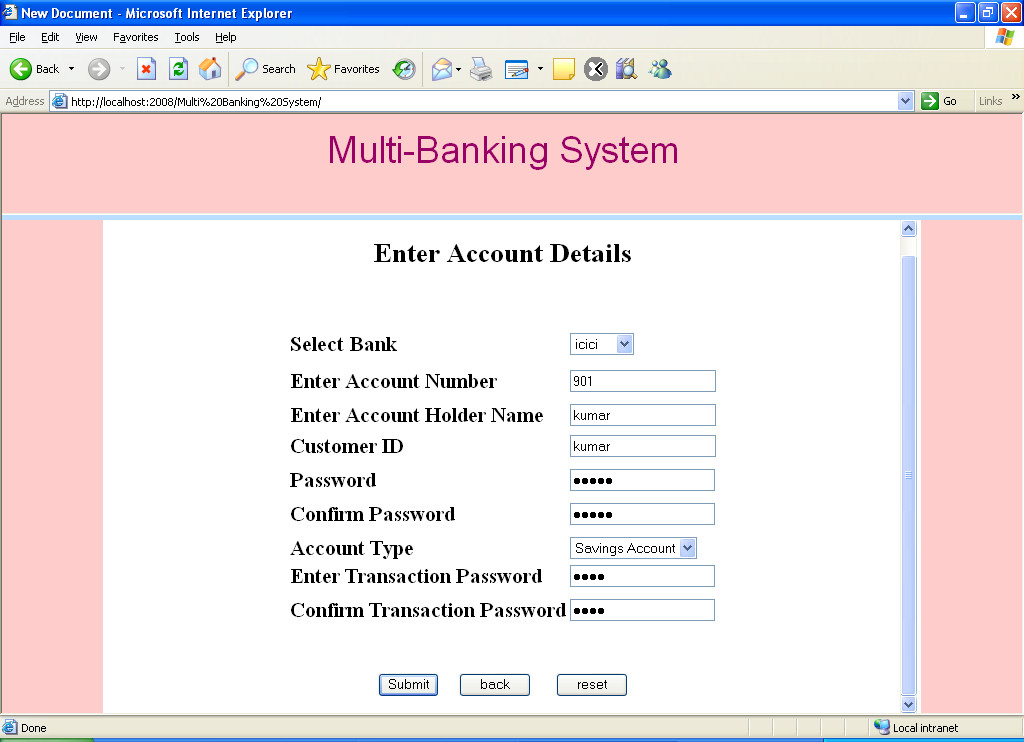
****

Fig 7.13 new user creating account page

**If User enter all the details and click submit button then this page was displays**

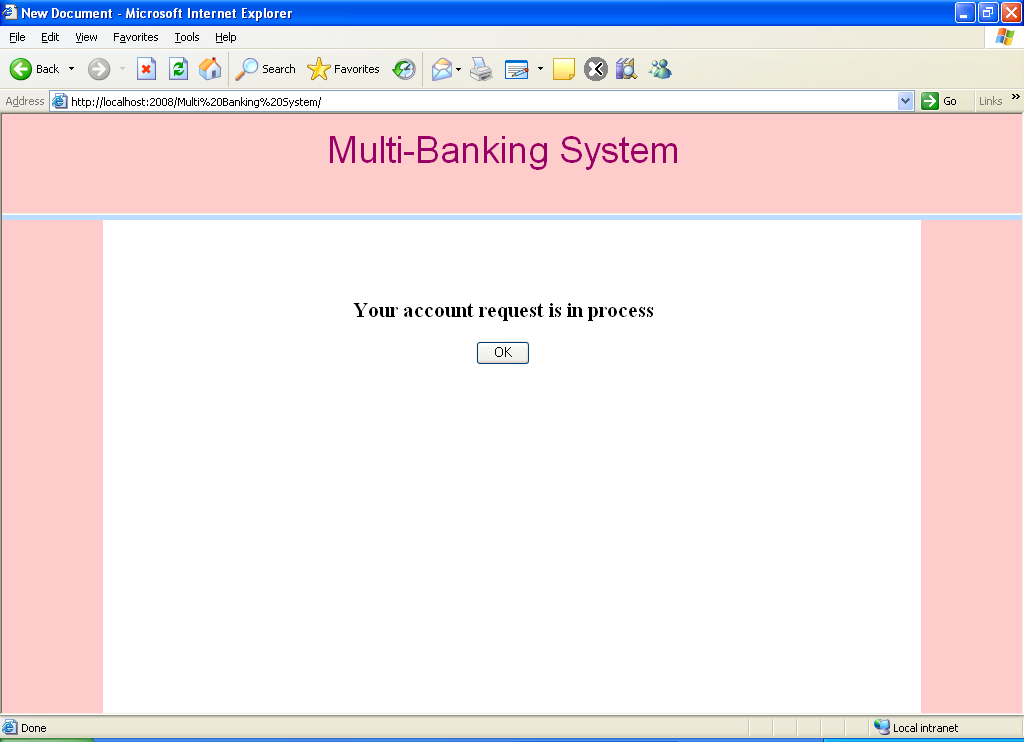
****

Fig 7.14 page displayed after entering user details and clicking submit button

**User Login Page with choose bank**

****

Fig 7.15 user login page to select bank

**If the user successfully login then this page was displays**

****

Fig 7.16 page displays after user login successfully

**Total List of customers in the selected bank page**

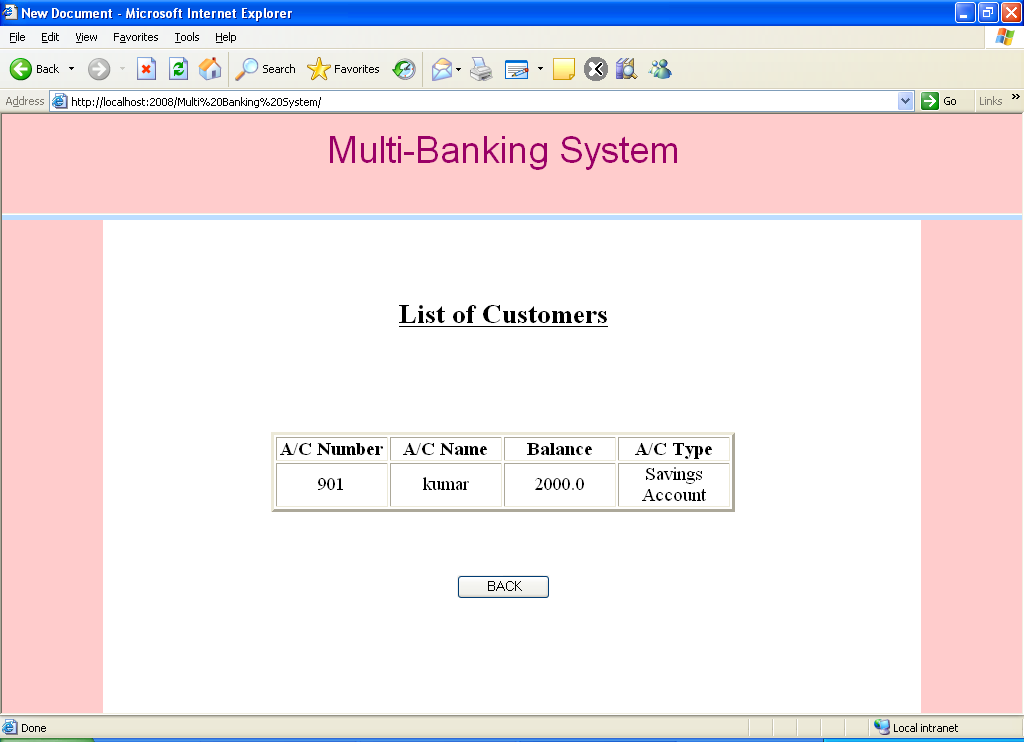
****

Fig 7.17 total list of customers in the selected bank

**List of Accounts that are available in that bank Page**

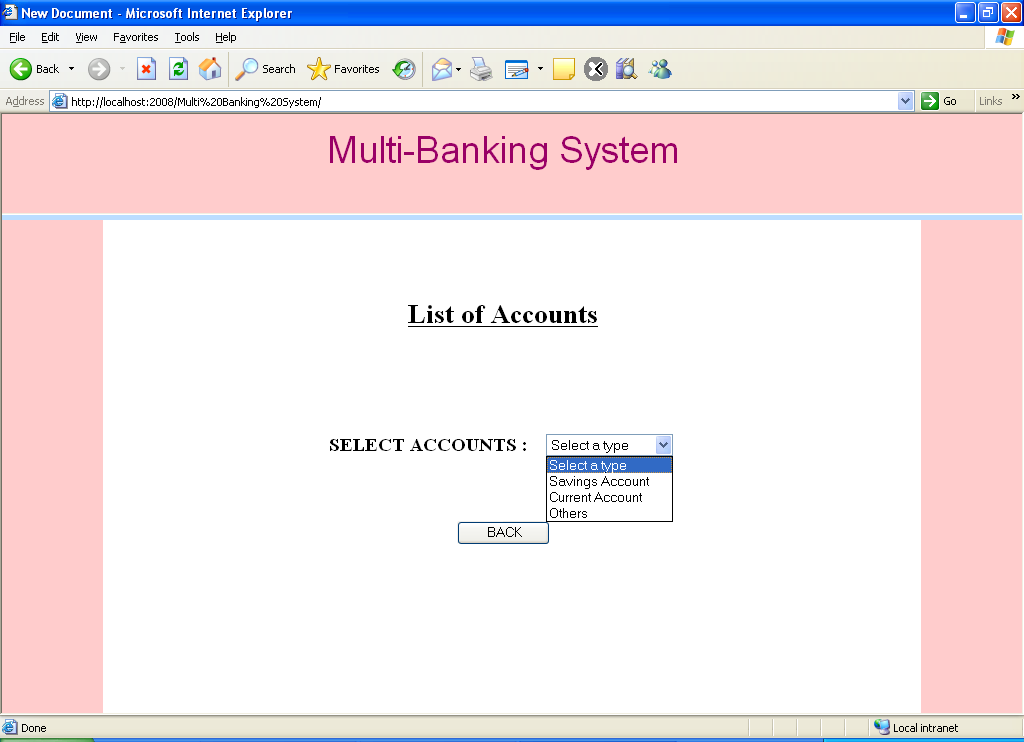
****

Fig 7.18 list of accounts available in the selected bank page

**Transactions Page**

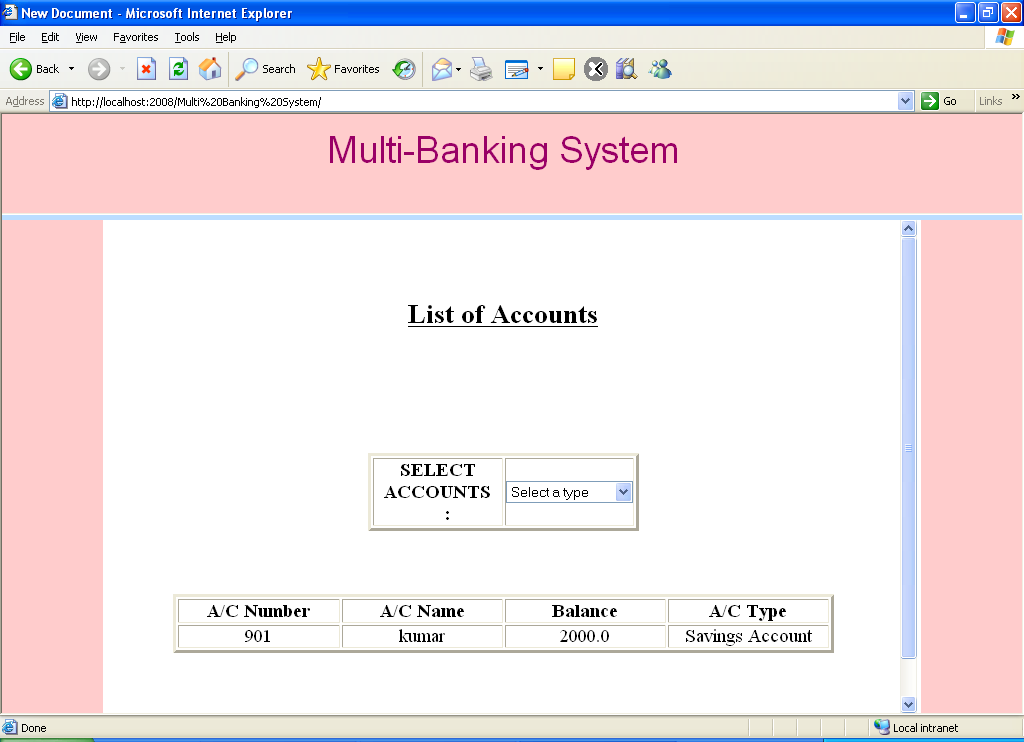
****

Fig 7.19 transactions page

**Transactions Pending Page**

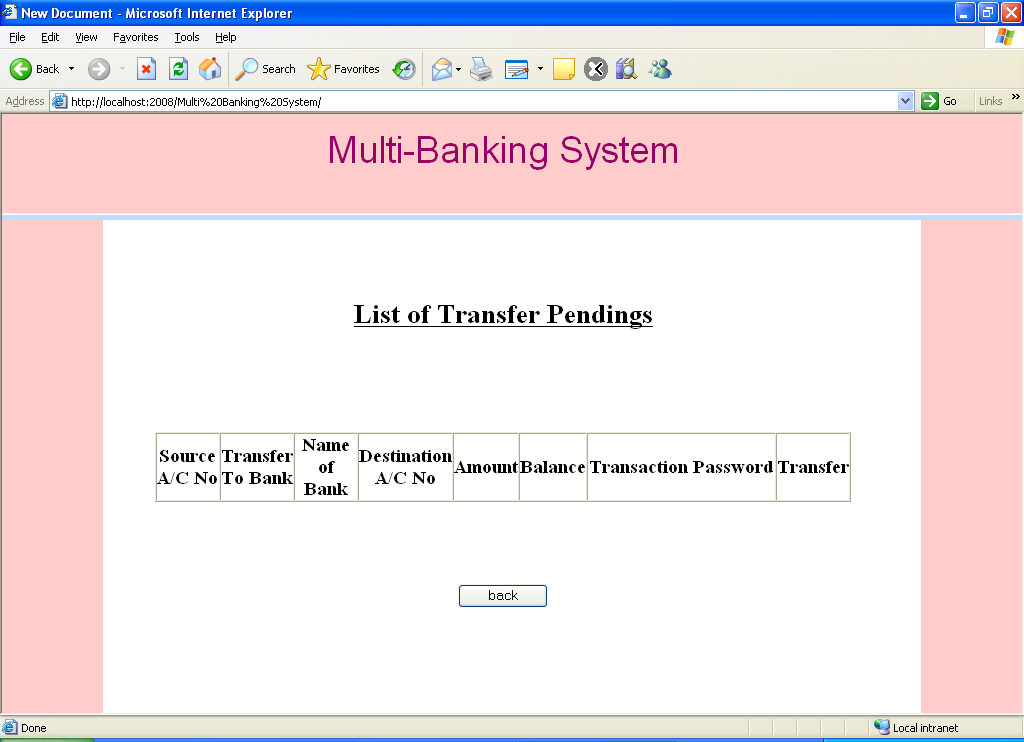
****

Fig 7.20 transactions pending page

**Customers Page**

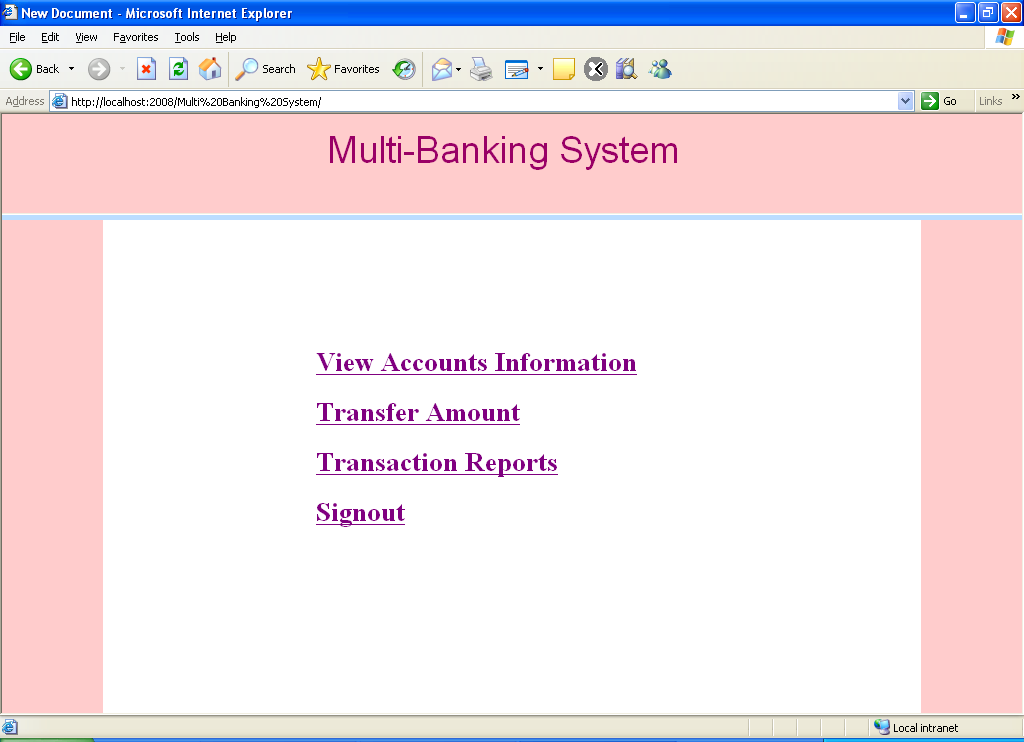
****

Fig 7.21 customer page

**Account Information Page**

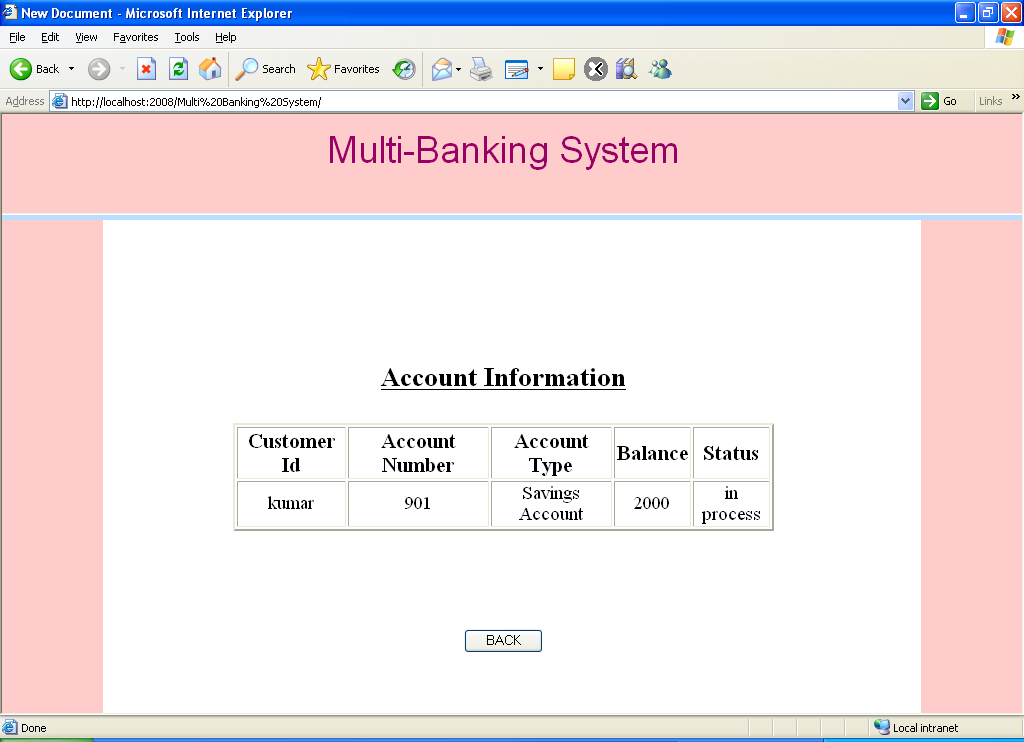
****

Fig 7.22 account information page

**Transfers Amount Page**

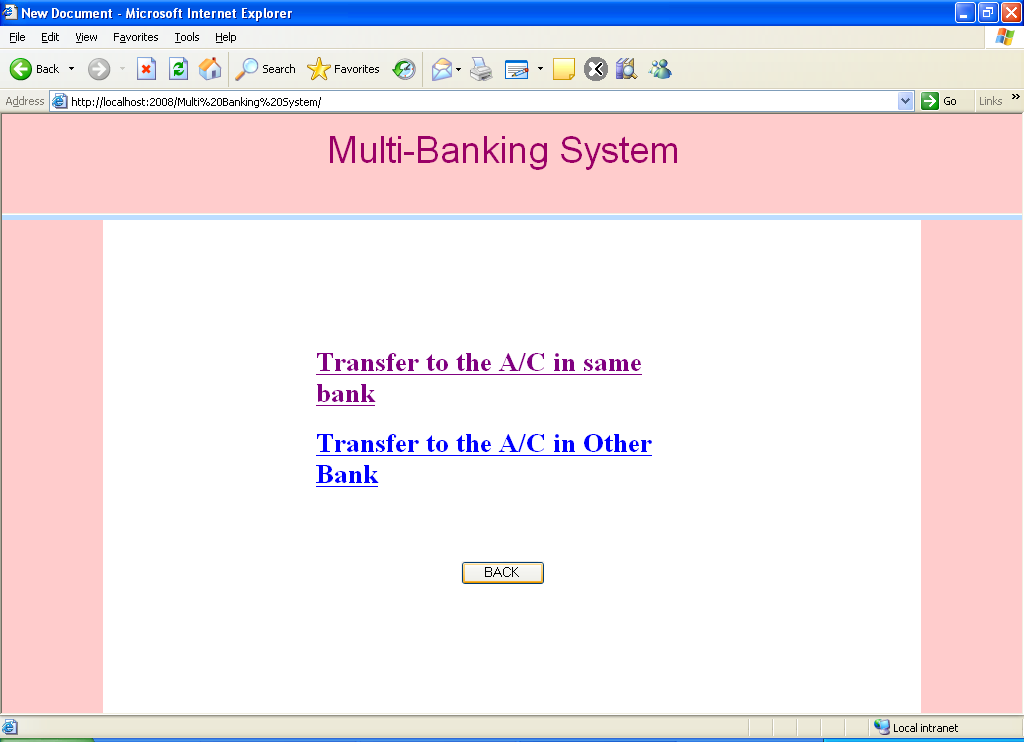
****

Fig 7.23 transfer amount page

**If You Click the Transfers link then it asks the Account details**

****

Fig 7.24 after clicking the transfer link, it ask you to enter the account details

**Transaction Reports Page**

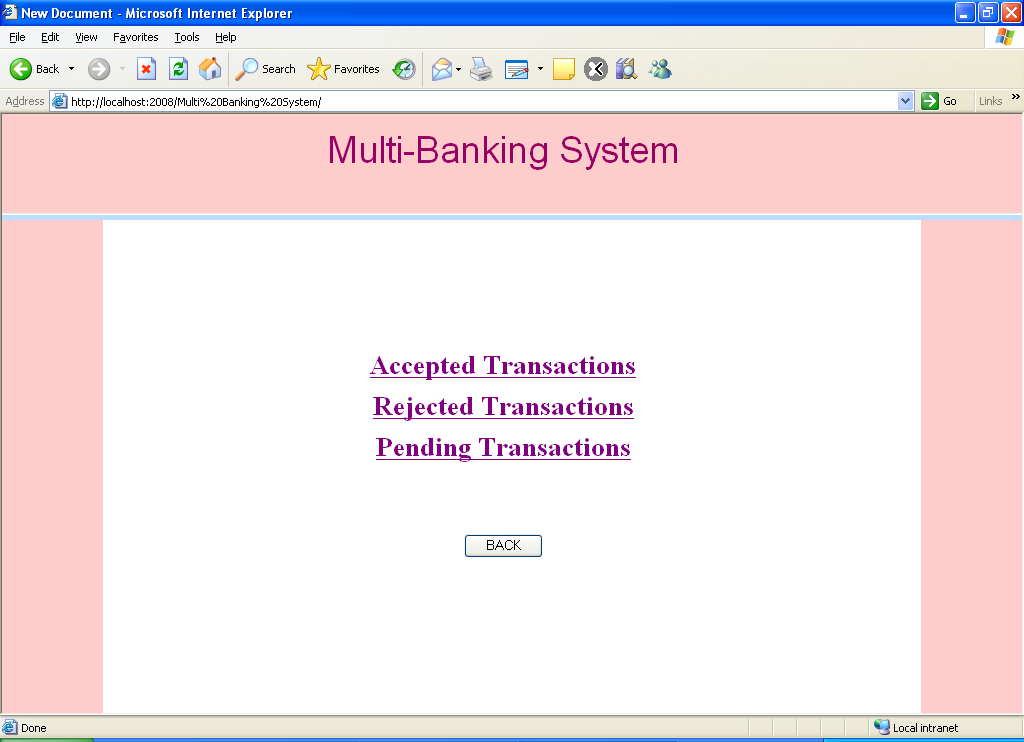
****

Fig 7.25 transaction reports page

**Accepted Transactions Page**

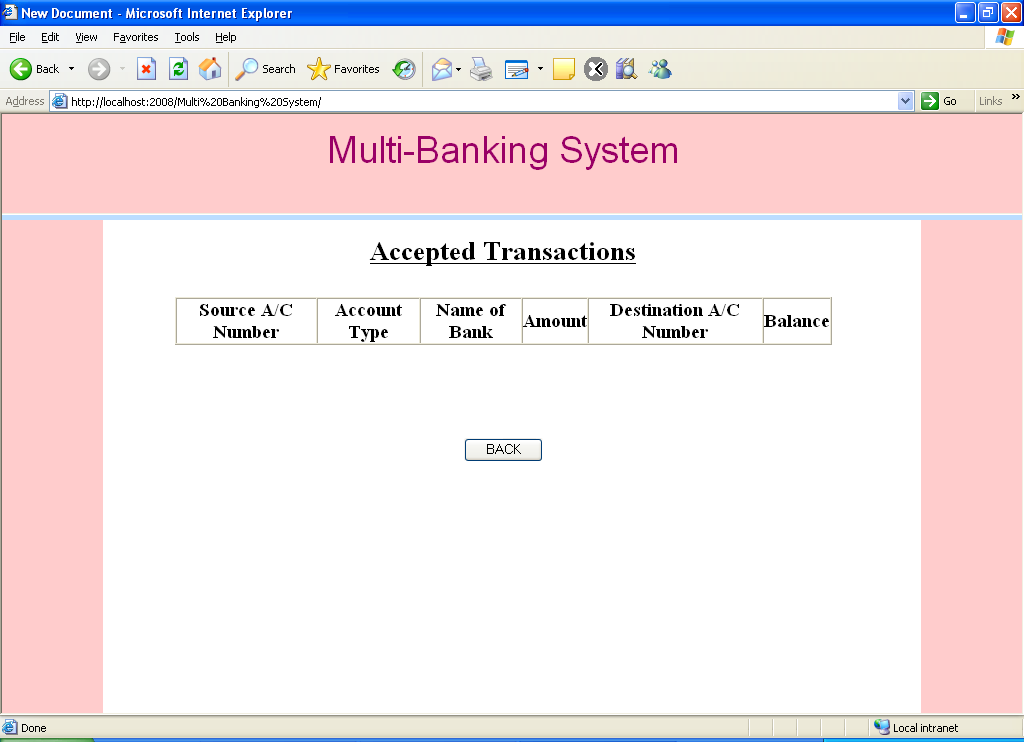
****

Fig 7.26 accepted transaction page

**Rejected Transactions Page**

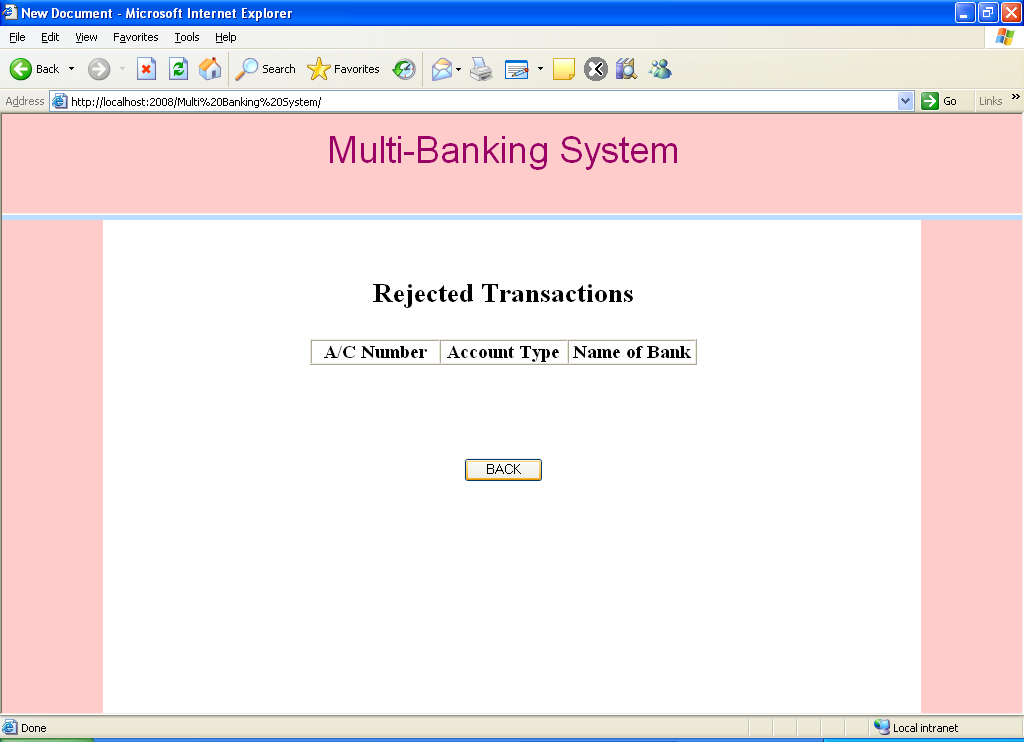
****

Fig 7.27 rejected transaction page

**Pending Transactions Page**

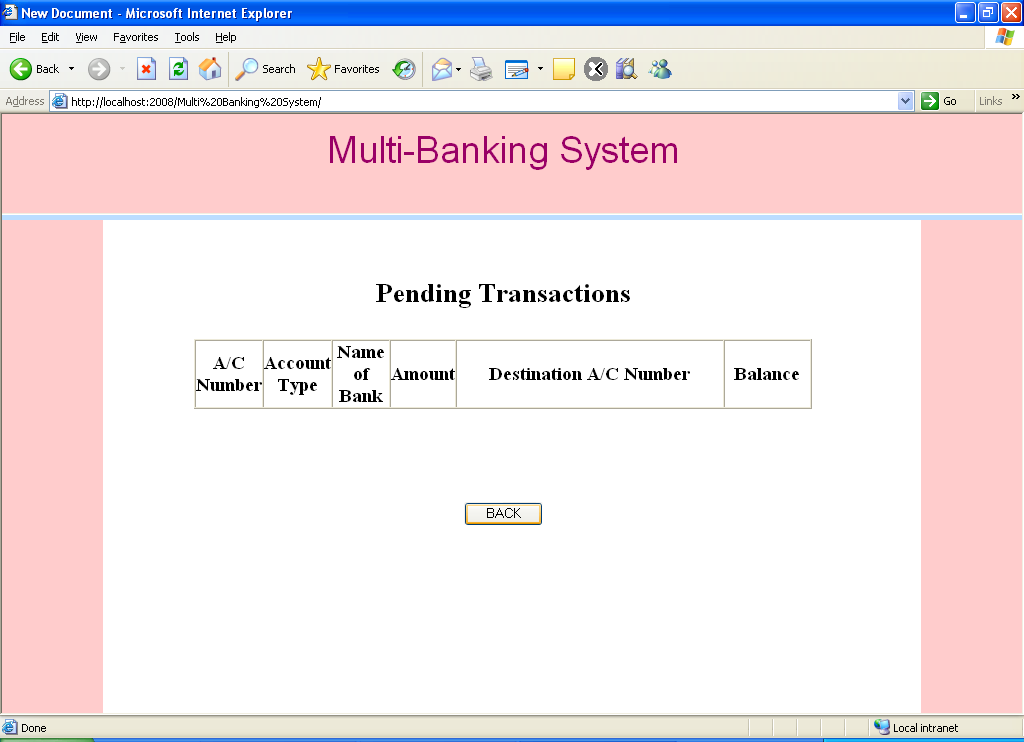
****

Fig 7.28 pending transaction page

**CONCLUSION & FUTURE SCOPE**

The application demonstrate the way to develop multi banking transaction system by using interactive web line by using JSP, servlet with more secure way to access. This means the application server easily deployable and accessible.

**REFERENCES**

# References for the Project Development Were Taken From the following Books and Web Sites.

**JAVA Technologies**

[1] JAVA Complete Reference

[2] Java Script Programming by Yehuda Shiran

[3] Mastering JAVA Security

[4] JAVA2 Networking by Pistoria

[5] JAVA Security by Scotl oaks

[6] Head First EJB Sierra Bates

[7] J2EE Professional by Shadab siddiqui

[8] JAVA server pages by Larne Pekowsley

[9] JAVA Server pages by Nick Todd

**HTML**

[1] HTML Black Book by Holzner

**JDBC**

[1] Java Database Programming with JDBC by Patel moss.

[2] Software Engineering by Roger Pressman.