# CHAPTER 1

# INTRODUCTION

# OVERVIEW

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”.

Existing System proposed the user to go to Different banks or their respective sites in order to check balance, to do bill payments, do transaction. This leads to a lot of time wastage. People are really busy now-a-days who cannot afford such wastage of time. Multi-Banking System is a web-based project for number of banks who wants to make some revolution in the existing system. This web-application is providing the scope to provide reliable, scalable, low cost solutions to reach more users so that they can handle their multiple accounts very easily. This is a Web-application for its web-savvy users who expect 24 \* 7 information on demand over internet. This system is also very user friendly.

In the current scenario, there is a rat race in each and every professional field. It is true for the banking too. A multi-banking portal is a website dedicated for online information about different banking needs.

This multi-banking portal helps both the end user and bank management to keep track on the transaction. It is being developed in such a way that each record is stored separately without making any confusion. So, a multi-banking portal is the perfect online arena, where both the users and the banks find their goal in the pursuit of finding easy way of doing transaction and keeping track on different functional areas

**1.2 MULTI-BANKING SYSTEM**

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.

In other words, Multi-Banking is the process of personnel banking using electronic resources, in particular the internet. Banks and user have moved much of their process online so as to improve the speed by which users can take advantage of various functionalities. Using database technologies, and search engines, users can now find schemes of different bank and choose the one suits them. Using an online multi-banking system may potentially save the user time The internet, which reaches a large number of people and can get immediate feedback, has become the major source of online banking.

Internet has radically changed the banking function from the user and bank perspective. Conventional methods of banking processes are readily acknowledged as being time-consuming with high costs and limited geographic reach. However, banking through (WWW) provides global coverage and ease. Likewise, the speedy integration of the internet into banking processes is primarily recognized due to the internet's unrivalled communications capabilities, which enable managers for written communications through e-mails, blogs and portals.

**1.3 OBJECTIVE OF THE SYSTEM**

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.

* 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.

**1.4 JUSTIFICATION AND NEED FOR THE SYSTEM**

The system provides proper security and reduces the manual work. This application tries to eliminate the difficulties of the existing system and helps the user to reduce the workload and mental conflict. It is very user-friendly. Through this, the users can check their account, transaction and hence it becomes very easy for them. Multi-banking is so effective and is considered to be the most efficient way of banking for the following reasons:

• It is highly cost efficient and promises increased ROI

• Multi-Banking brings in an organized in PBP

• Easy and efficient way of banking

• Reduced complexity, reduced paper work and streamlined workflow

• Establishes efficient communication channel between users and banks

• Helps in establishing a relationship between the user and the manager

• Dependable database applications available to support banking process

• One cannot ignore the efficiency that internet brings in

**1.5 ADVANTAGES OF THE SYSTEM**

* In this system the database is maintained in centralized manner.
* The user can access the entire account information
* This system is very fast because of the centralized database and accessing database will be very easy, when compared to the existing system.
* The user can also ask any query to support in the website, if any.

**1.5.1 OTHER ADVANTAGES**

**Reach**

• As it is an online portal so it can be reached by every individual, moreover it depends how they use it.

**Speed**

• The speed of operation depends on network, though it works faster than offline mode and decreases paper work cost.

**Cost**

• Multi-Banking System is very cost effective and since it is done in planned way, it will be quite useful.

**Interaction**

• Online banking process provides interaction between user and bank. It is user friendly and also provides support face so user can ask their query also.

# CHAPTER 2

# LITERATURE SURVEY

**[1] TITLE: BANKING STRUCTURE AND PERFORMANCE OF INDIAN BANKING SYSTEM**

**AUTHOR: BHATIA (1978)**

In his study titled , “BANKING STRUCTURE AND PERFORMANCE ” a case study of the Indian banking system attempted to analyze the economic performance of Indian banking system as reflected by its output ,price and profitability during the period 1950-68.he found the profit of Indian banking system during the said period

.

**[2] TITLE: DEVELOPMENT RESPONSIBILITY AND PROFITABILITY OF BANKS**

**AUTHOR: KULKARNI (1979)**

In his study titled, “DEVELOPMENT RESPONSIBILITY AND PROFITABILITY OF BANKS” Stressed upon social responsibility of banking sector. He was the view that looking for profit maximization only was not true profitability of banks as social benefits arising out of banking operations cannot be ignored.

**[3] TITLE: SOCIAL PRIORITY INDEX OF PUBLIC SECTOR BANKS**

**AUTHOR: MARKAND (1979)**

In his book titled, “SOCIAL PRIORITY INDEX OF PUBLIC SECTOR BANKS” evaluated the performance of public sector banks. With the help of performance index consisting six quantitative indicators such as branch expansion, priority sector credit and wage cost, it concluded that the priority sector financing was essential and necessary.

# CHAPTER 3

# EXISTING SYSTEM

**3.1 EXISTING SYSTEM**

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. In the existing system project system ,there is no such system is present who provide such interface, a person can access to their bank account by online or offline but can’t access through a single interface.so a customer needs to remember the details of each bank.

**3.2 DISADVANTAGES**

No one can do all banks transactions in a single portal or in single bank. This is the main disadvantages in existing system to avoid this problem we are introducing “Multi banking system”.

# CHAPTER 4

# PROPOSED SYSTEM

**4.1 PROPOSED SYSTEM**

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. Proposed system will provide an integrated system to the client so that he/she can do banking from a single place. This system will also store and maintain details of each bank so that a person need not remember. The multi banking system project system can be used by an industry to maintain and to provide the salary of different workers from different account through a single system.

**4.2 BENEFITS**

* Quick , authenticated access to accounts via the desktop
* Easily scalable to grow with changing system requirement
* Enterprise wide access to information.

# CHAPTER 5

# SYSTEM SPECIFICATIONS

**5.1 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented. Thus, below are the hardware requirements for the project:

* + - Processor : x64 based processor
    - Ram: Minimum 2Gb

**5.2 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the systems do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. Thus, below are the software requirements for the project:

* + - Operating System : Windows
    - Programming Language : Java
    - Database : Oracle
    - Web browser: Chrome/Edge
    - Web requirements: JSP, Servlet
    - Scripting Language : HTML

# CHAPTER 6

# SYSTEM DESIGN

**6.1 SYSTEM DESIGN**

It 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.

**6.2 UML DIAGRAMS**

A UML system is represented by using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram.

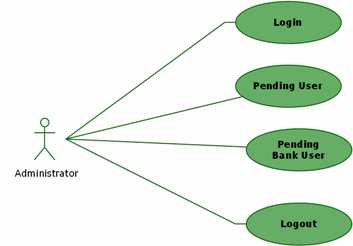


Figure 6.2.1 Use Case diagram for Administrator

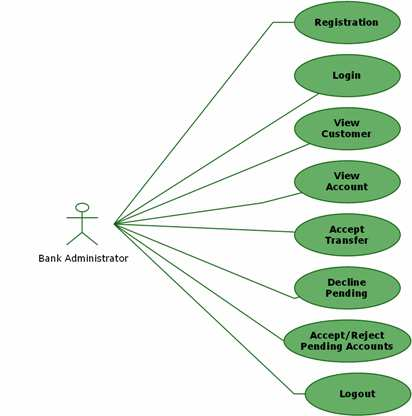


Figure 6.2.2 Use Case diagram for Bank Administrator

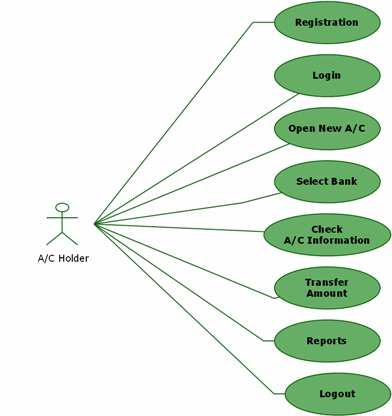


Figure 6.2.3 Use Case diagram for Account Holder

**6.3 DATA FLOW DIAGRAM**

Data Flow Diagram is a diagrammatic representation of data movement through a system- manual or automated- from inputs to outputs through processing.

The data flow diagram helps in analysis of the flow of data through a system and thus helps in identifying the system requirements.

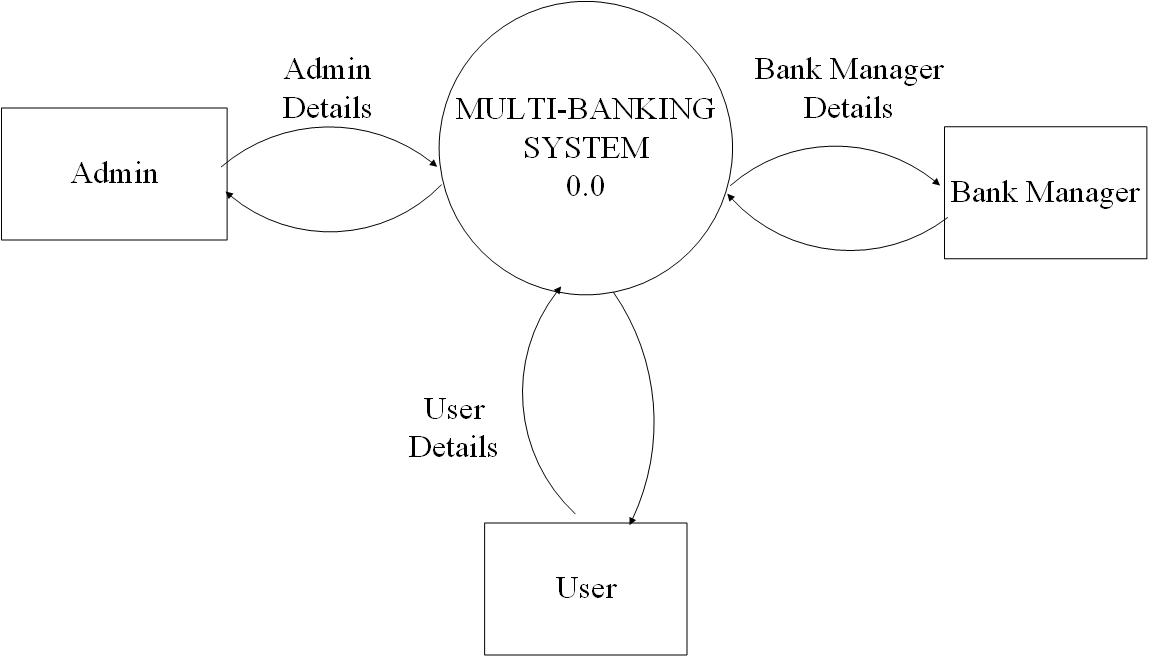


Figure 6.3.1 Data flow diagram level 0

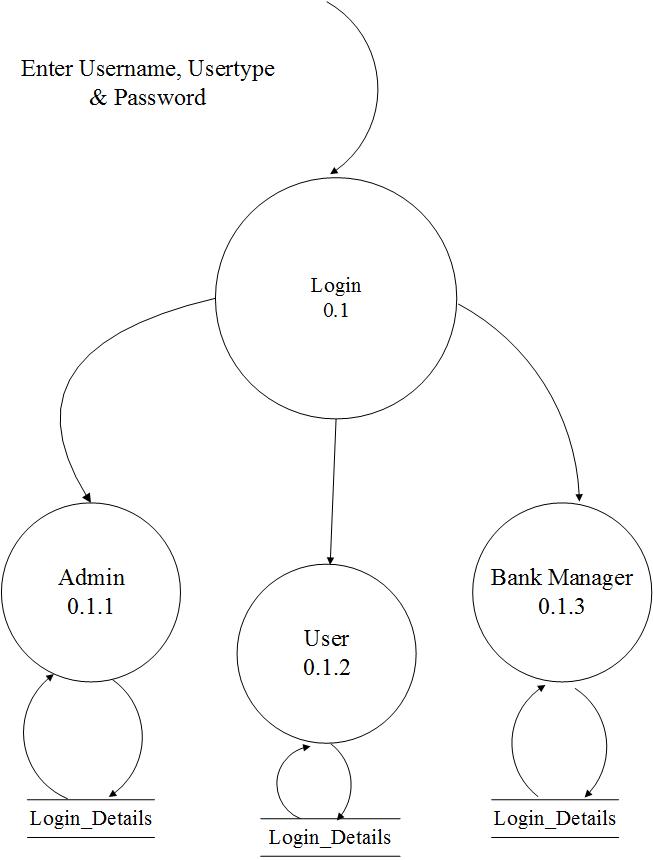


Figure 6.3.2 Data flow diagram level 1

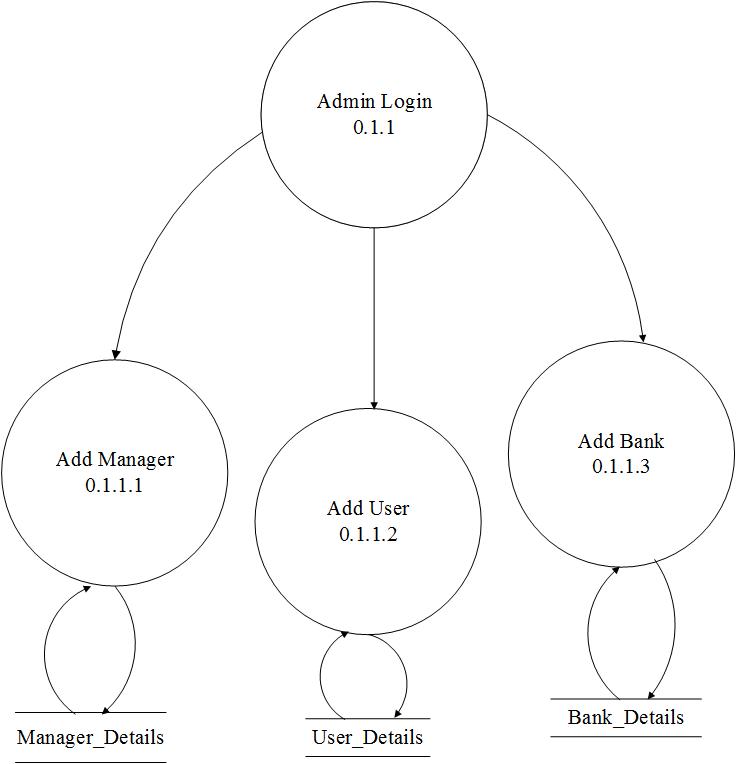


Figure 6.3.3 Data flow diagram level 2(Admin Module)

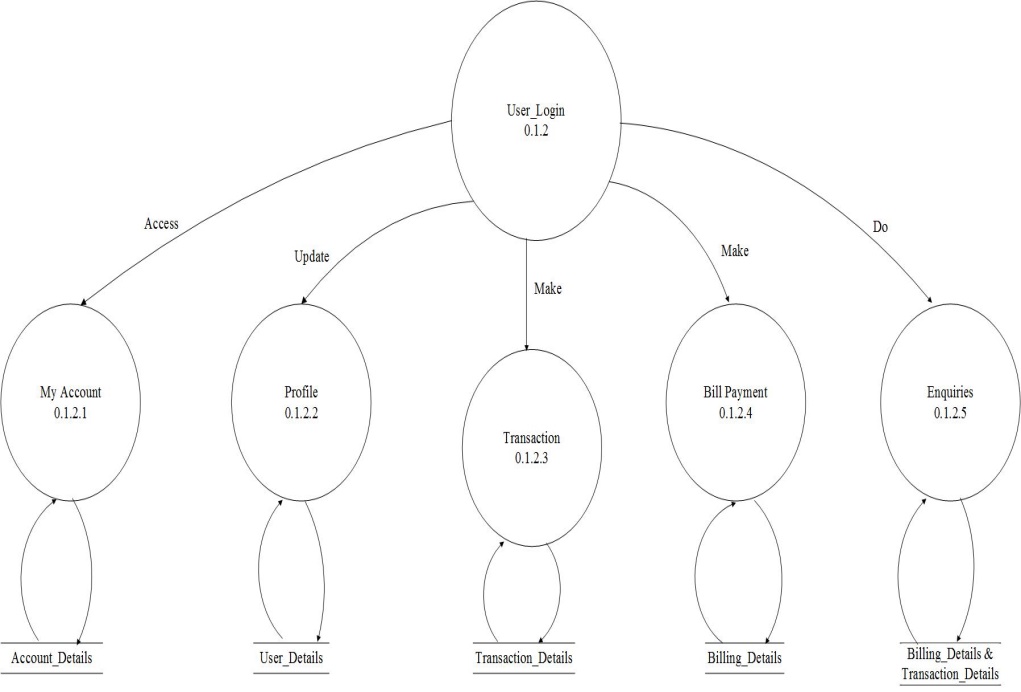


Figure 6.3.4 Data flow diagram level 2 (User Module)

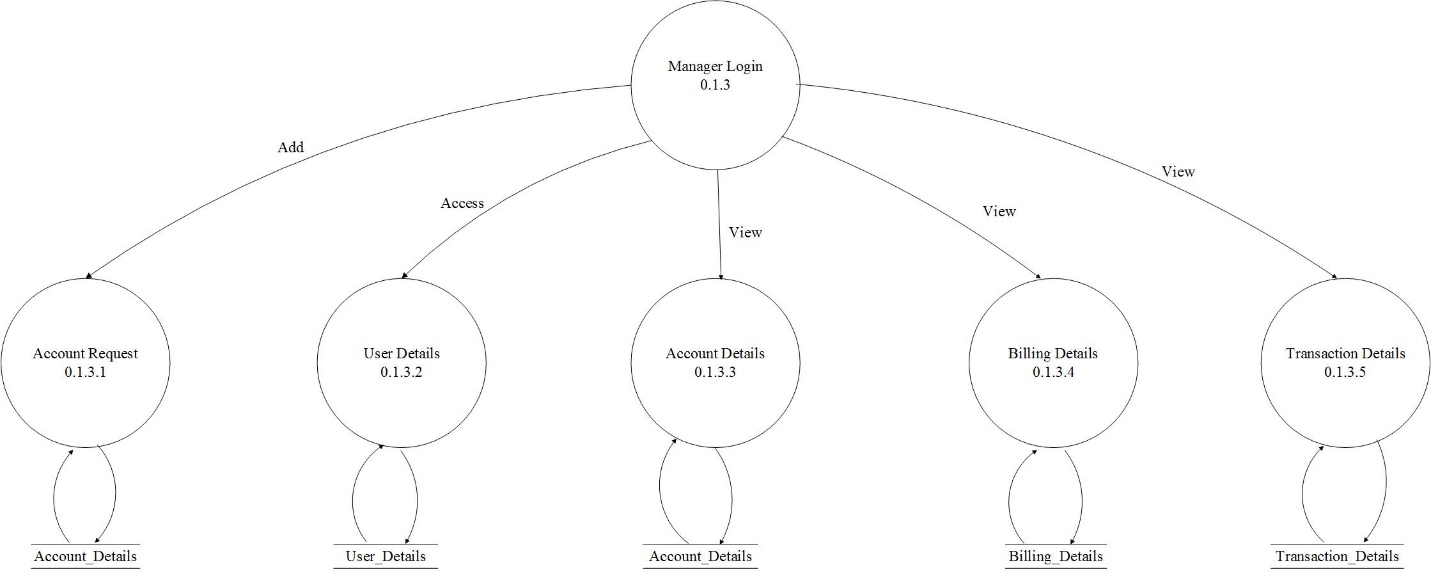


Figure 6.3.5 Data flow diagram level 2(Bank Manager Module)

# CHAPTER 7

# SYSTEM IMPLEMENTATIONS

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover, and evaluation of changeover methods.

**7.1 MODULES**

* Module 1: Admin Module
* Module 2: User Module
* Module 3: Bank Admin Module

**7.2 MODULE DESCRIPTIONS**

**7.2.1 ADMIN MODULE**

The admin module will be used by the administrator of this portal, admin can accept or reject the requests from the bank managers, and also admin can accept or reject the requests from the users. The requests are in the form of bank registration, customer registration. After accepting the request from bank managers, admin need to add the bank in the so that it is available for the user. This module is having following functionalities.

* **Pending User Requests**

By using this functionality admin give access permission to all users who are registered in this portal.

* **Pending Bank Manager Requests**

By using this functionality admin give access permission to all bank managers who are registered in this portal.

* **See Status**

By using this functionality admin see the status of user and manager in order to make change.

* **See list of banks**

By using this functionality admin can see all bank that are registered on the system

* **Add New Bank**

By using this functionality admin add the registered bank to the list to make it available for the user.

**7.2.2 USER MODULE**

This module describes all about users, a user can do operations like add a new account, view the account information, transfer amount from one account to other account, update personal details, make bill payment, see the transaction and billing details. This module consists following functionalities:-

* **Add New Account**

By using it user add a new account and make an account request to the manager.

* **Delete Account**

By using this functionality user delete the account, if he/she does not require in future.

* **View Account Information**

By using this functionality user view all his/her account details, this can be viewed by users who are having account in any bank.

* **Update Personal details**

By using this functionality user change their personal details, giving some new/updated data.

* **Change Password**

By using this functionality user change the login password and the transaction password in order to make his account more secured.

* **Transfer Amount**

By using this functionality user transfer money from his/her account to other accounts of same bank or other banks.

* **Bill Payment**

By using this functionality user make bill payment, by using the different bank accounts.

* **Transaction Enquires**

By using this functionality user is able to see the different transaction made. Each transaction is having a ‘transaction id’ along with the other details.

* **Billing Enquires**

By using this functionality user is able to see the different Bill payment made. Each transaction is having a ‘billing id’ along with the other details.

* **Product & Services and Contact details**

In any situation user can ask for help and product & services tab keep him/her updated about the different facilities of bank.

**7.2.3 BANK ADMIN MODULE:**

This module describe all about the bank manager. By using this module bank manager can view details of user, they can go for any transactions of their user and also they can give access permeations to all users of that bank.

This module consists following functionalities:-

* **User Details**

By using this functionality bank manager get the entire user list with its status and user id.

* **Accounts Details**

By using this functionality bank manager get the entire account list with the user detail of that account.

* **Transaction Details**

By using this functionality bank manager get the details of different transactions made by the user either to their accounts or by their accounts.

* **Billing Details**

By using this functionality bank manager get the details of different bill payment made by the user by their accounts.

* **New Account Request**

By using this functionality bank manager accept or reject account request, requested by the user.

# CHAPTER 8

# INPUT & OUTPUT DESIGN

**8.1 INPUT DESIGN**

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.

**8.1.1 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

**8.1.2 INPUT TYPES**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* + External inputs, which are prime inputs for the system.
  + Internal inputs, which are user communications with the system.
  + Operational, which are computer department’s communications to the system
  + Interactive, which are inputs entered during a dialogue.

**8.2 OUTPUT DESIGN**

It is in general are External Outputs whose destination is outside the organization & Internal Outputs whose destination is within organization, which is considered that 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.

# CHAPTER 9

# 9. PERFORMANCE ANALYSIS

A bank’s performance determinants are of fundamental importance to evaluate its profitability, but for what concerns the specific case of multi banking it is necessary to find out the micro and macro components useful to carry out an efficient analysis aimed at revealing the advantages or disadvantages of this innovative process.

The decision of a bank to undertake a particular strategy is doubtlessly determined by its comparative advantage in the cost structure and in the revenue model. Moreover, it is easily intuitive to understand how a bank’s strategy would be mostly influenced by the external economic development.

In light of our analysis and data gathered, it can be concluded that the difference in performance between single and multi-banking is so far seen as an add-on to the existing banking environment, and not as a business model able to efficiently challenge competitors; but they, however support major banking companies into driving more clients towards their services. This, as a consequence sets the stage for a process which provides with a competitive advantage and a significant boost in profitability. Doubtlessly, as IT infrastructures will further develop and more importance will be given to R&D and innovation, the structure of the multi-banking market would dramatically update.

# CHAPTER 10

# SYSTEM STUDY

# 

* 1. **FEASIBILITY STUDY**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* + Technical Feasibility
  + Economic Feasibility
  + Operation Feasibility
    1. **TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following whether there are necessary technologies that exist to do or whether the proposed equipment’s have the technical capacity to hold the data required to use the new system.

* + 1. **OPERATIONAL FEASIBILITY**

**User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details.

Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

**Reliability**

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

**Security**

The web server and database server should be protected from hacking, virus etc.

**Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc. these software will work both on Windows and Linux operating system. Hence portability problems will not arise.

**Availability**

This software will be available always.

**Maintainability**

The system uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses **Oracle10g**, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

**10.1.3 ECONOMIC FEASILITY**

The computerized system takes care of the present existing system’s data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

It should be built as a web based application with separate web server and database server. This is required as the activities are spread throughout the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

Open source software like TOMCAT, JAVA and Oracle is used to minimize the cost for the Customer. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

# 

# CHAPTER 11

# CONCLUSION

**11.1 CONCLUSION**

The project “Multi-Banking System” has been developed as per the requirement specification. This Project has several forms that are validated properly and provide facilities to the user to know more about their accounts and banks. Here giving the user id provide complete security and authentication to each department so that unauthenticated user could not hack the system. So it is a secured and better performance oriented project.

It has been developed properly and the complete system is thoroughly tested with the availability data and throughput reports which are prepared manually. This design is so flexible that any new modules can be incorporated easily. The concept and the technology used in the project make it robust and user friendly with 100% accuracy output. The cost of this system is also less and the deployment sector does not have to make any additional investment for implementing this project. So, it is economically feasible.

In this growing industry, it is very important to meet the growing requirements and to save time of the users. A multi-banking portal just fulfills the tough demand of the corporate world. It is one of the fastest growing industries. There are more and more opportunities to come for the portals in the time to come; there is no doubt about it.

**11.2 FUTURE ENHANCEMENTS**

In future, as the technology emerges it is possible to upgrade the system & based on the future security issues, security can be improved using emerging technologies.

**APPENDIX 1**

**1. SAMPLE OUTPUT**



Figure A1.1 Home Screen

****

Figure A1.2 Login Screen

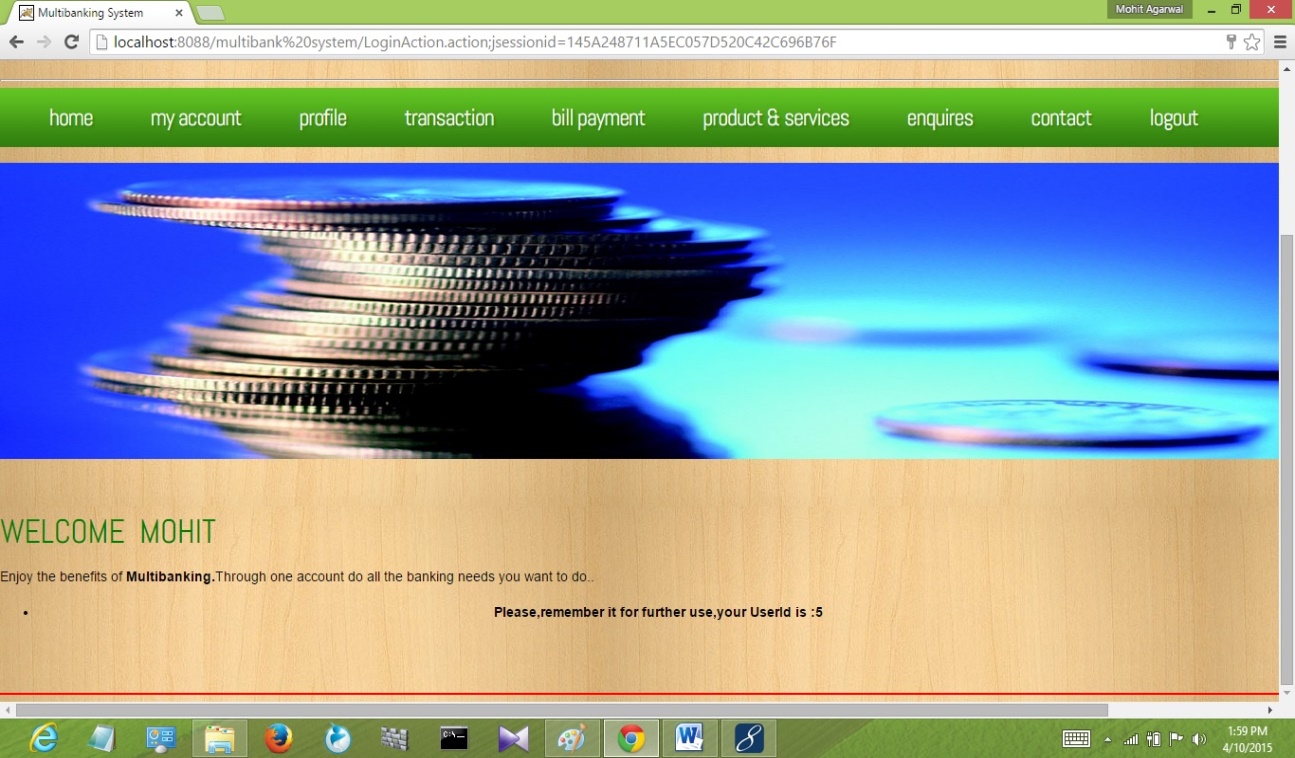
****

Figure A1.3 User Homepage Screen

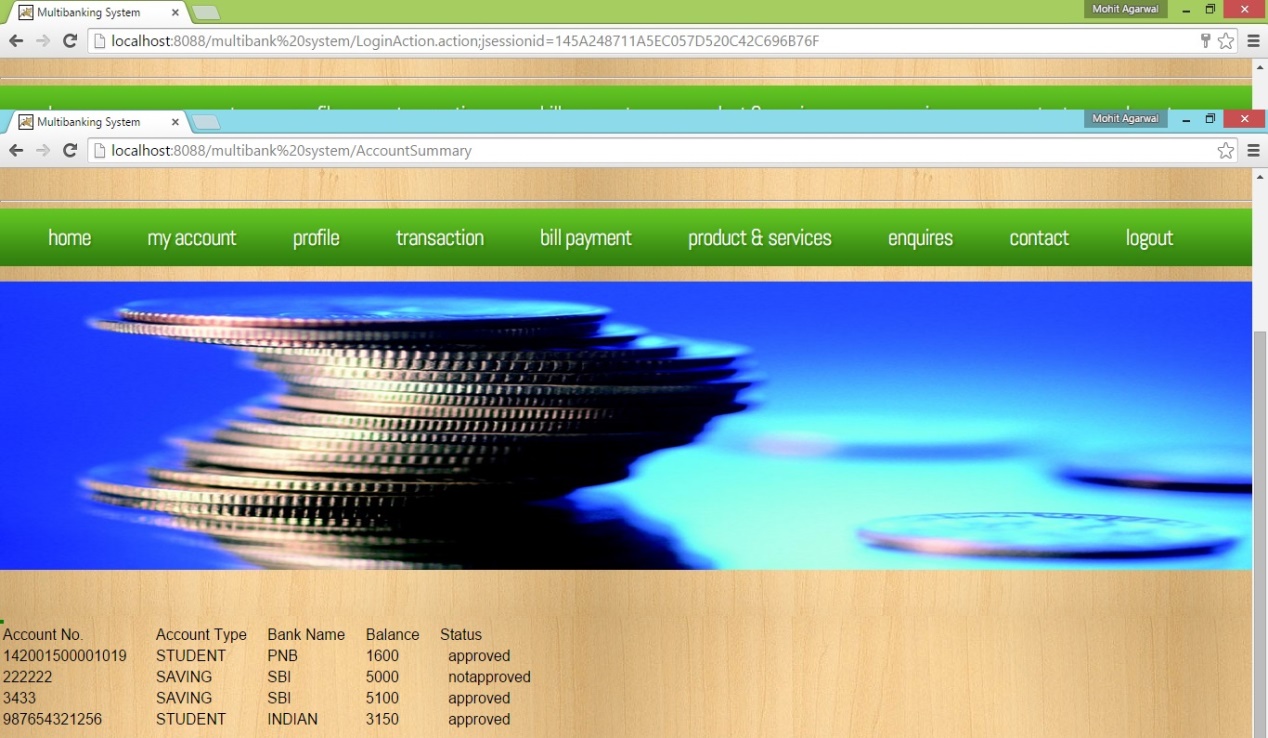
****

Figure A1.4 Account summary Screen

**2. SAMPLE CODE**

**Login Form:**

packagepkg;

importjava.sql.ResultSet;

importjava.sql.SQLException;

importjava.text.SimpleDateFormat;

importjava.util.\*;

importjava.io.PrintWriter;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

importjavax.servlet.http.HttpSession;

Dblogicdl=newDblogic();

public String LoginData()throwsSQLException,ClassNotFoundException

{

ResultSetrs=dl.RetLogin(username,password,usertype);

if(rs.next())

{

if(usertype.equals("mbadmin"))

{

return"mbadmin";

}

elseif(usertype.equals("user"))

{

String status=rs.getString("userstatus");

if(status.equals("approved"))

{

bank.s=(rs.getString(1)).toString();

String name=(rs.getString("USERNAME")).toUpperCase();

ResultSet rs1=DbConnect.getStatement().executeQuery("select \* from user\_details where loginid='"+s+"'");

while (rs1.next())

{

addActionMessage("Please,remember it for further use,yourUserId is :" + rs1.getString(1));

bank.s1=(rs1.getString(1)).toString();

}

setMsg(name);

return"user";

}

}

elseif(usertype.equals("bankmanager"))

{

String status=rs.getString("userstatus");

if(status.equals("approved"))

{

bank.m=(rs.getString(1)).toString();

String name=(rs.getString("USERNAME")).toUpperCase();

ResultSet rs2=DbConnect.getStatement().executeQuery("select \* from MANAGER\_DETAILS where loginid='"+m+"'");

while(rs2.next())

{

bank.m1=(rs2.getLong("MANAGERID"));

bank.m2=(rs2.getString("BANKNAME")).toString();

}

setMsg(name);

return"bankmanager";

}

}

else

{

setMsg("Waiting for Approval");

return"loginfail";

}

}

setMsg("Check Your Login Details.");

return"loginfail";

}

public String CreateUser()throws Exception

{

Map session=ActionContext.getContext().getSession();

String Uname=(String)session.get("Username");

String Bname=(String)session.get("Bankname");

Long Accno=(Long)session.get("Accno");

String Acctype=(String)session.get("Acctype");

String Passkey=(String)session.get("Passkey");

String Fname=(String)session.get("Fname");

String Lname=(String)session.get("Lname");

String Sex=(String)session.get("Sex");

String Address=(String)session.get("Address");

Long Areacode=(Long)session.get("Areacode");

String City=(String)session.get("City");

String State=(String)session.get("State");

String Country=(String)session.get("Country");

String Dob=(String)session.get("Dob");

String Email=(String)session.get("Email");

Long Mobile=(Long)session.get("Mobile");

String Securityquestion=(String)session.get("Securityquestion");

String Answer=(String)session.get("Answer");

inti=0;

i=dl.Registeruser(Uname,Bname,Accno,Acctype,Passkey,Fname,Lname,Sex,Address,Areacode,City,State,Country,Dob,Email,Mobile,Securityquestion,Answer,getPassword(),getTransactionpassword());

return"registersuccess";

}

public String ManagerData()throws Exception

{

Map session=ActionContext.getContext().getSession();

session.put("Username", getUsername());

session.put("Bankname", getBankname().toUpperCase());

return"managersuccess";

}

public String ManagerData1()throws Exception

{

Map session=ActionContext.getContext().getSession();

String Uname=(String)session.get("Username");

String Bname=(String)session.get("Bankname");

inti;

i=dl.RegisterManager(Uname,Bname,getFirstname(),getLastname(),getSex(),getAddress(),getAreacode(),getCity(),getState(),getCountry(),getDob(),getEmail(),getMobile(),getSecurityquestion(),getAnswer(),getPassword());

return"managersuccess";

}

}

req.setAttribute("ARR",arlst);

return"request"

}

public String UserApproved()throwsSQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.getContext().get(ServletActionContext.HTTP\_REQUEST);

Long id=Long.parseLong(request.getParameter("id"));

System.out.println(id);

inti=dl.userApproved(id);

if(i>0)

{

returnNewUserRequest();

}

returnnull;

}

public String UserReject()throwsSQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.getContext().get(ServletActionContext.HTTP\_REQUEST);

Long id=Long.parseLong(request.getParameter("id"));

System.out.println(id);

inti=dl.userReject(id);

if(i>0)

{

returnNewUserRequest();

}

returnnull;

}

public String ManagerApproved()throwsSQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.getContext().get(ServletActionContext.HTTP\_REQUEST);

Long id=Long.parseLong(request.getParameter("id"));

System.out.println(id);

inti=dl.managerApproved(id);

if(i>0)

{

returnNewManagerRequest();

}

returnnull;

}

public String ManagerReject()throwsSQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.getContext().get(ServletActionContext.HTTP\_REQUEST);

Long id=Long.parseLong(request.getParameter("id"));

System.out.println(id);

inti=dl.managerReject(id);

if(i>0)

{

returnNewManagerRequest();

}

returnnull;

}

public String ListOfUser()throwsSQLException,ClassNotFoundException

{

ResultSetrs=dl.userList();

ArrayList<bank>arlst=newArrayList<bank>();

while(rs.next())

{

bank b=new bank();

b.setUsername(rs.getString("USERNAME"));

//System.out.println(rs.getString("LOGINID"));

b.setUserstatus(rs.getString("USERSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

return"userlist";

}

public String ListOfManager()throwsSQLException,ClassNotFoundException

{

ResultSetrs=dl.managerList();

ArrayList<bank>arlst=newArrayList<bank>();

while(rs.next())

{

bank b=new bank();

b.setUsername(rs.getString("USERNAME"));

//System.out.println(rs.getString("LOGINID"));

b.setUserstatus(rs.getString("USERSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

return"managerlist";

}

public String AddNewAccount()throws Exception

{

inti=0;

Long uid=Long.parseLong(s1);

Long bid=getBank\_id();

ResultSetrs=DbConnect.getStatement().executeQuery("select \* from bank\_details where bank\_id="+bid+"");

if(rs.next())

{

String bname=rs.getString(2).toUpperCase();

i=dl.addNewAccount(getUsername(),bname,getAccountno(),getAccounttype(),getBankpasskey(),uid);

}

return"accountadded";

}

public String Accountsummary() throws Exception

{

Long uid=Long.parseLong(s1);

ResultSet rs1=DbConnect.getStatement().executeQuery("select \* from ACCOUNT\_DETAILS where userid="+uid+" ");

ArrayList<bank>arlst=newArrayList<bank>();

while(rs1.next())

{

bank b=new bank();

b.setAccountno(rs1.getLong("ACCOUNTNO"));

System.out.println(rs1.getLong("ACCOUNTNO"));

b.setAccounttype(rs1.getString("ACCOUNTTYPE"));

b.setBankname(rs1.getString("BANKNAME"));

b.setBalance(rs1.getLong("BALANCE"));

b.setAccountstatus(rs1.getString("ACCOUNTSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

return"showaccounts";

}

public String deleteAccount()throwsSQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.getContext().get(ServletActionContext.HTTP\_REQUEST);

Long id=Long.parseLong(request.getParameter("id"));

System.out.println(id);

inti=dl.accountDelete(id);

if(i>0)

{

returnDeleteAccounts();

}

returnnull;

}

public String PasswordChange() throws Exception

{

Long id1=Long.parseLong(s);

Long id2=Long.parseLong(s1);

inti=dl.passwordChange(getOldpassword(),getNewpassword(),id1,id2);

return"pswdchanged";

}

**DbLogic.java**

packagepkg;

importjava.sql.\*;

publicclassDblogic

{

Connection con=null;

Statement st=null;

ResultSetrs=null;

publicResultSetRetLogin(String username,Stringpassword,Stringusertype)throwsSQLException,ClassNotFoundException{

rs=DbConnect.getStatement().executeQuery("select \* from LOGIN\_DETAILS where username='"+username+"' and password='"+password+"' and usertype='"+usertype+"'");

returnrs;

}

publicintRegisteruser(String uname, String bname, Long accno,

String acctype, String passkey, String fname, String lname,

String sex, String address,

Long areacode, String city, String state, String country,

String dob, String email, Long mobile,Stringsecurityquestion,String answer,

String password, String transactionpassword) throwsSQLException

{

inti=0;

intj,k;

try

{

i=DbConnect.getStatement().executeUpdate("insert into LOGIN\_DETAILS values(LOGIN\_DETAILS\_SEQ.nextval,'"+uname+"','"+password+"','user','notapproved')");

j=DbConnect.getStatement().executeUpdate("insert into USER\_DETAILS values(USER\_DETAILS\_SEQ.nextval,'"+uname+"','"+fname+"','"+lname+"','"+sex+"','"+address+"',"+areacode+",'"+city+"','"+state+"','"+country+"','"+dob+"','"+email+"',"+mobile+",'"+securityquestion+"','"+answer+"','"+password+"','"+transactionpassword+"',LOGIN\_DETAILS\_SEQ.nextval-1)");

k=DbConnect.getStatement().executeUpdate("insert into ACCOUNT\_DETAILS values(ACCOUNT\_DETAILS\_SEQ.nextval,"+accno+",'"+acctype+"','"+bname+"',5000,'"+passkey+"','notapproved',USER\_DETAILS\_SEQ.nextval-1)");

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

returni;

}

publicintRegisterManager(String uname, String bname, String firstname,

String lastname, String sex, String address, Long areacode,

String city, String state, String country, String dob,

String email, Long mobile, String securityquestion, String answer,

String password) throwsSQLException

{

inti=0;

int j=0;

try

{

i=DbConnect.getStatement().executeUpdate("insert into LOGIN\_DETAILS values(LOGIN\_DETAILS\_SEQ.nextval,'"+uname+"','"+password+"','bankmanager','notapproved')");

j=DbConnect.getStatement().executeUpdate("insert into MANAGER\_DETAILS values(MANAGER\_DETAILS\_SEQ.nextval,'"+uname+"','"+firstname+"','"+lastname+"','"+sex+"','"+address+"',"+areacode+",'"+city+"','"+state+"','"+country+"','"+dob+"','"+email+"',"+mobile+",'"+securityquestion+"','"+answer+"','"+password+"',LOGIN\_DETAILS\_SEQ.nextval-1,'"+bname+"')");

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

returni+j;

}

}

publicResultSetViewprofile(Long id) throwsSQLException

{

rs=DbConnect.getStatement().executeQuery("select \* from USER\_DETAILS where userid="+id+"");

returnrs;

}

publicResultSettransactionAccount(Long accno, String acctype, String bname,

Long bal) throwsSQLException

{

rs=DbConnect.getStatement().executeQuery("select \* from ACCOUNT\_DETAILS where accountno="+accno+" and accounttype='"+acctype+"' and bankname='"+bname+"' and balance="+bal+"");

returnrs;

}

publicint bill(Long accno, Long b) throwsSQLException

{

inti=0;

i=DbConnect.getStatement().executeUpdate("update ACCOUNT\_DETAILS set balance="+b+" where accountno="+accno+"");

returni;

}

publicResultSetbillLogin(String transactionpassword, Long id) throwsSQLException,ClassNotFoundException

{

rs=DbConnect.getStatement().executeQuery("select \* from USER\_DETAILS where transactionpassword='"+transactionpassword+"' and userid="+id+"");

returnrs;

}

publicResultSetbillAccount(Long accno, String acctype, String bname,

Long bal) throwsSQLException

{

rs=DbConnect.getStatement().executeQuery("select \* from ACCOUNT\_DETAILS where accountno="+accno+" and accounttype='"+acctype+"' and bankname='"+bname+"' and balance="+bal+"");

returnrs;

}

publicResultSetbillEnquire(Long accno, String bname, Long uid, String x, String y) throwsSQLException,ClassNotFoundException

{

rs=DbConnect.getStatement().executeQuery("select bill\_id,billpayment\_date,billername,billno,billamount,accountno,balance from BILLING\_DETAILS where accountno="+accno+" and bankname='"+bname+"' and billpayment\_date BETWEEN '"+x+"' and '"+y+"' and userid="+uid+" order by bill\_id");

returnrs;

}

publicResultSettransactionEnquire(Long accno, String bname, Long uid,

String x, String y)throwsSQLException,ClassNotFoundException

{

rs=DbConnect.getStatement().executeQuery("select tran\_id,transaction\_date,r\_accountno,r\_bankname,amount,accountno,balance from TRANSACTION\_DETAILS where accountno="+accno+" and bankname='"+bname+"' and transaction\_date BETWEEN '"+x+"' and '"+y+"' and userid="+uid+" order by tran\_id");

returnrs;

}

**DbConnect.java**

packagepkg;

importjava.sql.Connection;

importjava.sql.DriverManager;

importjava.sql.Statement;

public final class DbConnect

{

public static Statement getStatement()

{

Connection con=null;

Statement st=null;

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","multibanking","multibanking");

st=con.createStatement();

}

catch(Exception e)

{

e.printStackTrace();

}

returnst;

}

}

**JSP PAGES**

**Login.jsp Page**

<%@pagelanguage="java"import="java.util.\*"pageEncoding="ISO-8859-1"%>

<%@tagliburi="/struts-tags"prefix="s"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPEHTMLPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<metaname="keywords"content=""/>

<metaname="description"content=""/>

<metahttp-equiv="content-type"content="text/html; charset=utf-8"/>

<title>Multibanking</title>

<linkhref="http://fonts.googleapis.com/css?family=Abel"rel="stylesheet"type="text/css"/>

<linkhref="style.css"rel="stylesheet"type="text/css"media="screen"/>

<scripttype="text/javascript"src="jquery-1.7.1.min.js"></script>

<scripttype="text/javascript"src="jquery.slidertron-1.0.js"></script>

<scripttype="text/javascript">

functionValidateForm(frm)

{

varpswd=frm.password.value;

varun=frm.username.value;

if(un.length==0)

{

alert("Username Must Not Be Blank...");

returnfalse;

}

if(pswd.length==0)

{

alert("Password Must Not Be Blank...");

returnfalse;

}

if(document.forms[0].usertype[0].checked==false&&document.forms[0].usertype[1].checked==false&&document.forms[0].usertype[2].checked==false)

{

alert("usertype cannot be null");

returnfalse;

}

returntrue;

}

</script>

</head>

<body><divid="wrapper">

<divid="header-wrapper">

<divid="header">

<divid="logo">

<h1>MULTIBANKING SYSTEM<br></h1>

<p><em><br></em></p>

</div>

</div>

</div>

<tablewidth="1500">

<tr><td><marqueescrollamount="7"align="left"><fontcolor="green"size="8"><b>Banks Linked:SBI,PNB,ICICI,INDIAN,IDBI</b></font></marquee></td></tr>

<s:formaction="LoginAction"onsubmit="return ValidateForm(this)">

<tr><td><s:textfieldname="username"label="User Name"/></td></tr>

<tr><td><s:passwordname="password"label="Password"/></td></tr>

<tr><td><s:radioname="usertype"label="User Type"list="{'user','mbadmin','bankmanager'}"/></td></tr>

<tr><tdcolspan="2"align="right"><s:submitvalue="Login"></s:submit></td></tr>

<tr><td><s:propertyvalue="msg"/></td></tr>

</s:form></table>

<br/>

&nbsp;&nbsp;<b><ahref="forgot password.jsp">forgot password?</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<ahref="signup.jsp">Register Now</a>

</b><br/><br/><hrcolor="red"width="1500"/>

</div>

</body>

</html>

MyAccounts.jsp Page

<%@pagelanguage="java"import="java.util.\*"pageEncoding="ISO-8859-1"%>

<%@pageimport="pkg.bank"%>

<%@tagliburi="/struts-tags"prefix="s"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPEHTMLPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<metaname="keywords"content=""/>

<metaname="description"content=""/>

<metahttp-equiv="content-type"content="text/html; charset=utf-8"/>

<title>Multibanking System</title>

<linkhref="http://fonts.googleapis.com/css?family=Abel"rel="stylesheet"type="text/css"/>

<linkhref="style.css"rel="stylesheet"type="text/css"media="screen"/>

<scripttype="text/javascript"src="jquery-1.7.1.min.js"></script>

<scripttype="text/javascript"src="jquery.slidertron-1.0.js"></script>

</head>

<body><divid="wrapper">

<divid="header-wrapper">

<divid="header">

<divid="logo">

<h1>MULTIBANKING SYSTEM</h1>

<p><br></p>

</div>

</div>

</div><hrwidth="1500"/>

<!-- end #header -->

<divid="menu">

<center><ul>

<liclass="current\_page\_item"><br></li>

<li><br></li>

<li><ahref="User Home.jsp"">Home</a></li>

<li><ahref="profile.jsp">Profile</a></li>

<li><ahref="Transaction Login.jsp">Transaction</a></li>

<li><ahref="Bill Login.jsp">Bill Payment</a></li>

<li><ahref="Product and Services.jsp">Product & Services</a></li>

<li><ahref="enquire type.jsp">Enquires</a></li>

<li><ahref="Contact.jsp">Contact</a></li>

<li><ahref="Logout">Logout</a></li>

</ul></center>

</div>

<!-- end #menu --><divid="slider">&nbsp;

<br/><divclass="viewer">

<divclass="reel">

<divclass="slide"><imgsrc="images/3.jpg"width="1500"height="300"alt=""/></div>

<divclass="slide"><imgsrc="images/img01.jpg"width="1500"height="300"alt=""/></div>

<divclass="slide"><imgsrc="images/1.jpg"width="1500"height="300"alt=""/></div>

</div>

</div>

</div>

<scripttype="text/javascript">

$('#slider').slidertron({

viewerSelector:'.viewer',

reelSelector:'.viewer .reel',

slidesSelector:'.viewer .reel .slide',

advanceDelay:3000,

speed:'slow'

});

</script>

<br/><br/><br/>

<center><ul>

<b><li><ahref="AccountSummary">Account Summary</a></li>

<li><ahref="ANA">Add New Account</a></li>

<li><ahref="DeleteAccount">Delete Account</a></li></b>

</ul></center>

<br/><br/><br/><hrcolor="red"width="1500"/>

</div>

</body>

</html>

# REFERENCES

[1] ["Banking Software Overview"](https://www.trustradius.com/banking#overview-2)*. TrustRadius*. Retrieved January 11, 2019*.*

[2] Discovering Your Financial Software Assistance www.dondosa.net/blogs/post/2021

[3] Finance – Page 565 | Angelico A. Groppelli, Ehsan Nikbakht – 2006

[4] ["The Advent of Modern Banking in India: 1720 to 1850s"](http://www.rbi.org.in/currency/museum/m-1720.html)*.*[Reserve Bank of India](https://en.wikipedia.org/wiki/Reserve_Bank_of_India). Retrieved 12 January 2015*.*

[5] McCarthy, Jim (1995). [Dynamics of Software Development](https://archive.org/details/dynamicsofsoftwa00mcca). Microsoft Press. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [1556158238](https://en.wikipedia.org/wiki/Special:BookSources/1556158238).

[6] ["Open Source Technology for Core Banking Solution & Software"](http://www.kimayainfotech.com/open-source-technology-for-core-banking-solution-&-software). kimayainfotech.com. Retrieved 20 March 2018.

[7] Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. (2002). ["Agile Software Development Methods: Review and Analysis"](https://web.archive.org/web/20110907061542/http:/agile.vtt.fi/publications.html). VTT Publications. 478. Archived from [the original](http://agile.vtt.fi/publications.html) on 7 September 2011. Retrieved 20 February 2012.

[8] [HTML5 Draft recommendation](https://www.whatwg.org/specs/web-apps/current-work/multipage/), changes to HTML and related APIs to ease authoring of web-based applications.

[9] [Web Applications](https://curlie.org/Computers/Internet/On_the_Web/Web_Applications) at [Curlie](https://en.wikipedia.org/wiki/Curlie" \o "Curlie)

[10] [Web Applications Working Group](http://www.w3.org/2008/webapps/) at the [World Wide Web Consortium](http://www.w3.org/) (W3C)