Mini Project - Retail Banking

Design document

Zhipeng Zhou, Sen Li, Li Huang, Run Yan

# **1. Introduction**

This project is a Java Web Application which used to manage Personal Banking Services. In this project we are going to use HTML, CSS, JavaScript, JSP, JDBC and AJAX etc. to implement the following functions:

**Customer functions:**

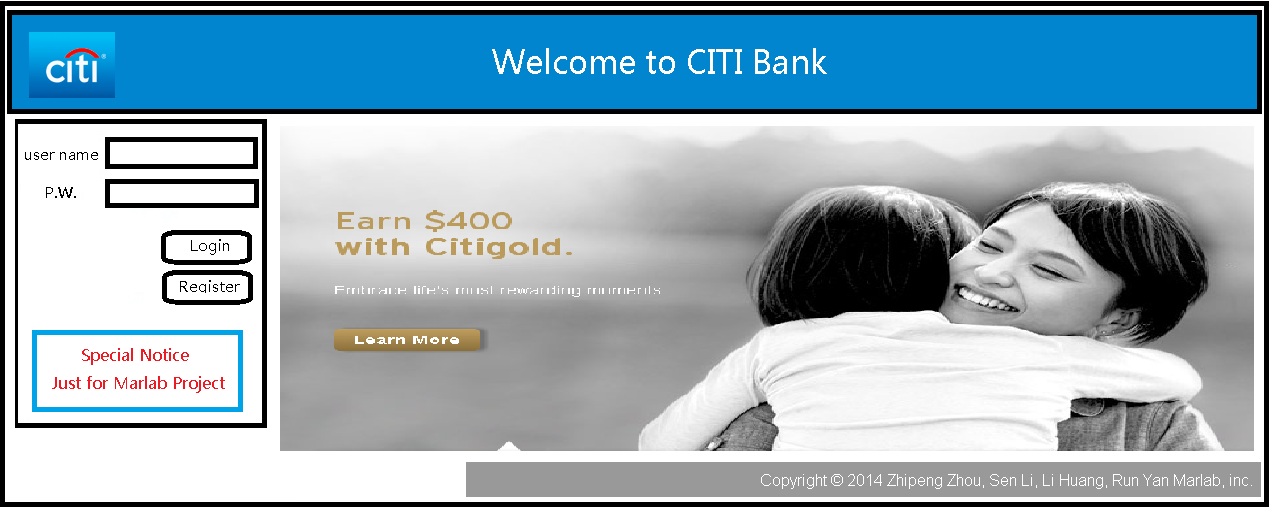
Open Account, Deposit Money, Transfer Money, View & edit customer information, View account information.

**Administrator functions:**

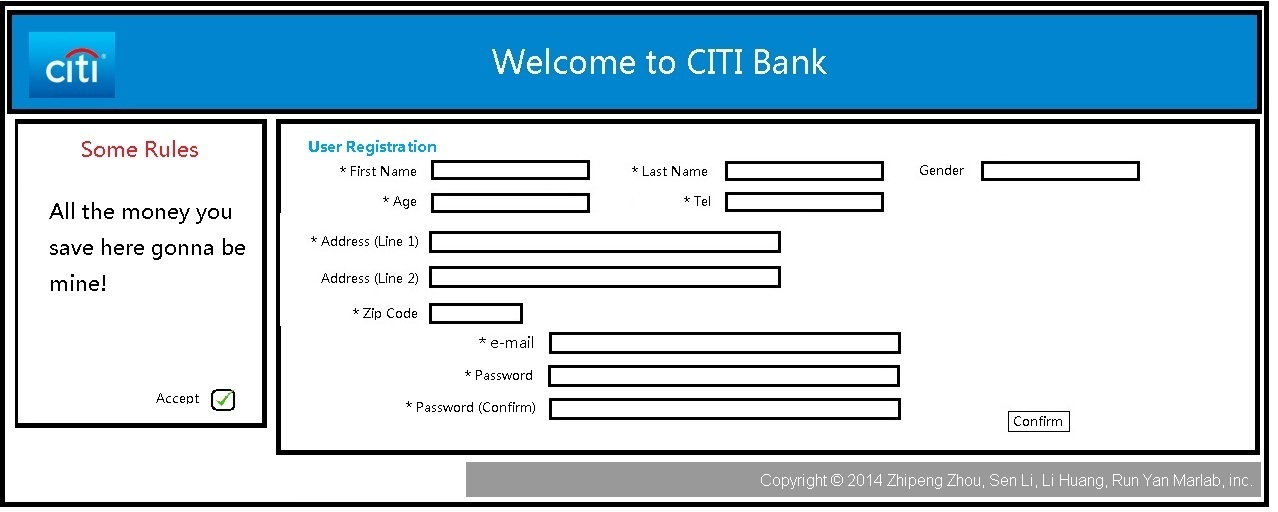
Display customers' information, reset customer password and delete customer, Frozen/Active account.

# **2. UI design**

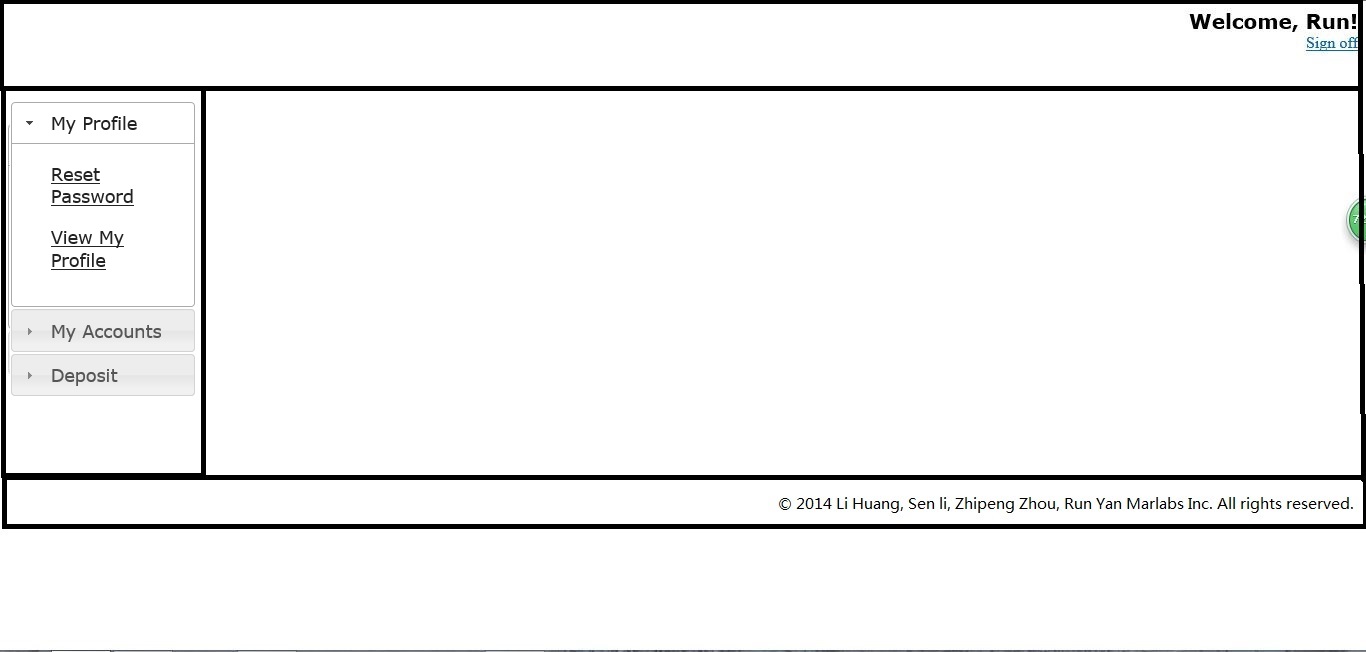
## 1.Login Interface



## 2. Register Interface



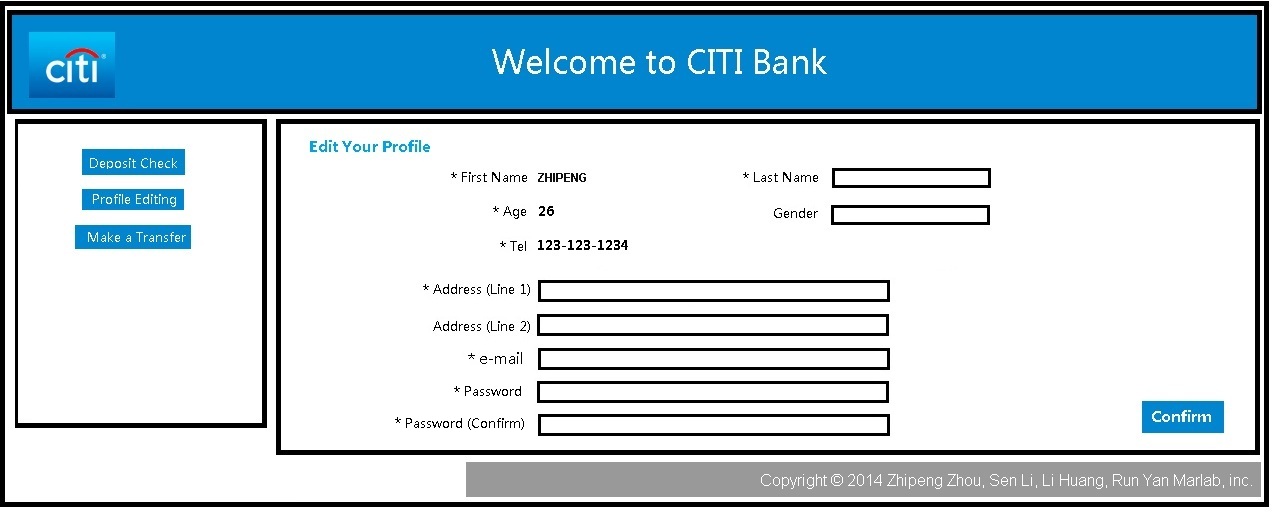
## 3. Customer Main Interface



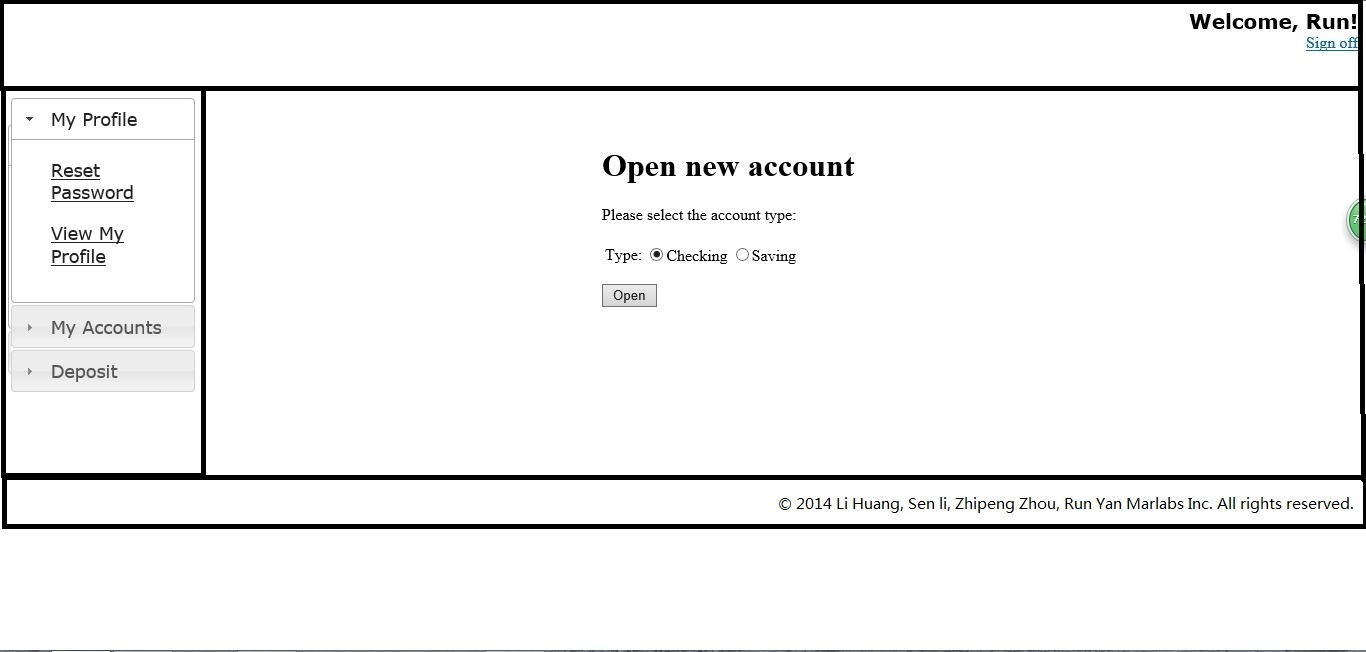
## 3.1 View customer information



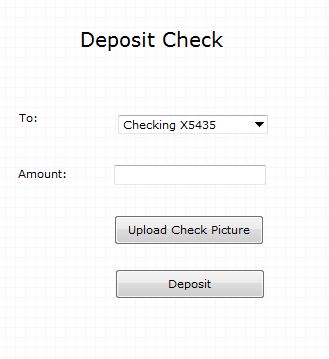
## 3.2 Edit customer information



## 3.3 New account



## 3.4 Deposit check



## 3.5 Account interface



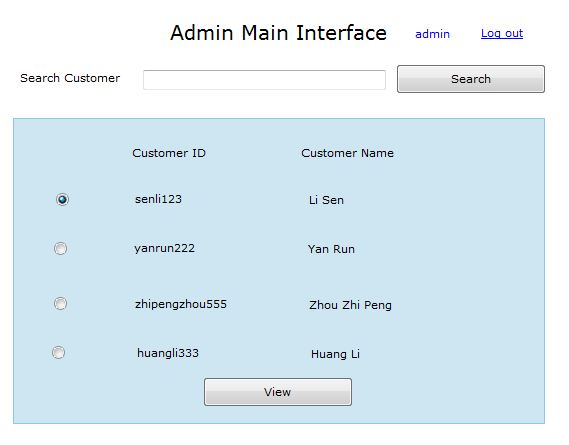
## 3.5.1 Transfer interface



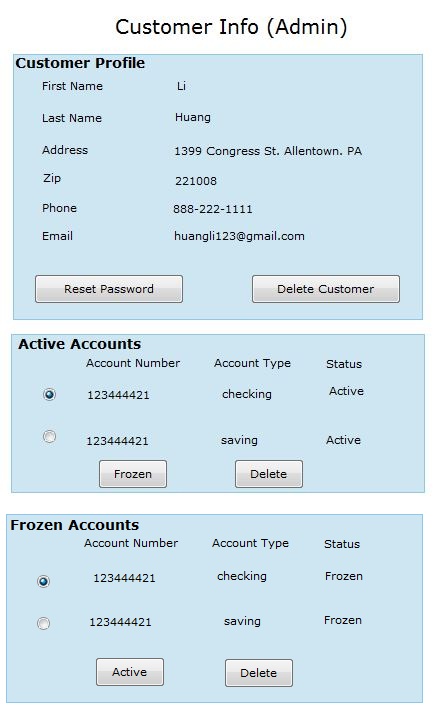
## 3.5.2 Statement interface



## 4. Admin main interface



## 4.1 Customer interface



# **3. Use case stories**

User:

1. Open Account.

Login Interface: Register --> Register Interface: input username, password, press "add" --> successful register --> enter Customer Info Interface: input personal info, press "update" --> goto Customer Main Interface: "new account" --> goto New Account Interface: select account type, "add". --> return to Customer Main Interface.

2. Deposit Money.

Login Interface: input username, password, "login" --> Customer Main Interface: "deposit check" --> Deposit Check Interface: choose account number , enter amount, upload check picture, "deposit". --> Customer Main Interface.

3. Transfer Money.

Login Interface: input username, password, "login" --> Customer Main Interface: choose account --> Account Main Interface:"transfer" -->Transfer Interface: enter Recipient account Number, Transfer Amount, and Description. "Make this transfer".

4. View & edit customer information.

Login Interface: input username, password, "login" --> Customer Main Interface: view profile --> Customer Info Interface: view & edit customer information. --> Customer Main Interface.

5. View account information.

Login Interface: input username, password, "login" --> Customer Main Interface: choose account --> Account Info Interface: view balance, "transaction history"--> Transaction History Interface: view history, generate and export .xls reports --> Account Info Interface.

------------------------------------------------------------------------------

Administrator:

1. Display customers' information.

Login Interface: input username, password, "login" --> Admin Main Interface: display all customers' list, "view" --> Customer Info(admin) Interface: view selected customer's profile and accounts information.

2. Customer Management.(reset customer password and delete customer)

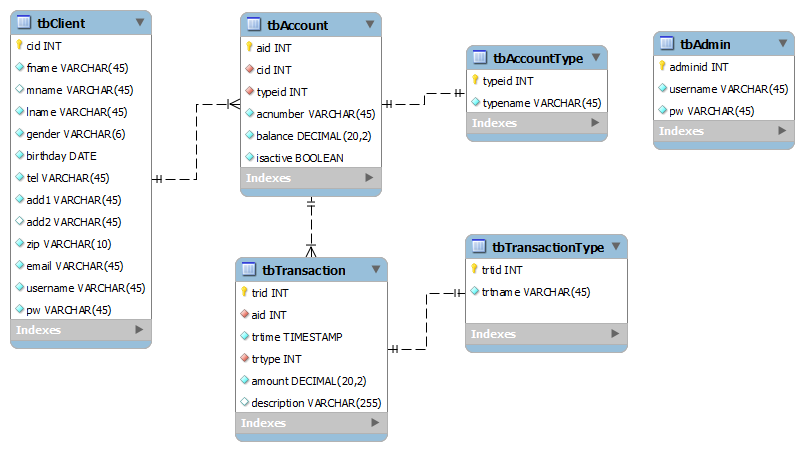
Login Interface: input username, password, "login" --> Admin Main Interface: display all customers' list, "view" --> Customer Info(admin) Interface: view selected customer's profile: "reset password" --> show reset success dialog --> Customer Info(admin) Interface.

“delete customer”.

3. Frozen/Active account

Login Interface: input username, password, "login" --> Admin Main Interface: display all customers' list, "view" --> Customer Info(admin) Interface: view accounts list: select an account: "frozen/active" --> Show frozen/active success dialog .

# 3. DB Design



# 4. JSP and Java Class files and their relations

==============================================================

**Root folder: webapps/root/bank**

==============================================================

JSP and Java Class files and their relations

==============================================================

Root folder: webapps/root/bank

1. Login

login.jsp ( ./src )

loginServlet.java ( .class ./build/classes )

Only jump to other pages. uses ClientDAO，

dao/ClientDao.java( Operate DB's ClientTable )

entity/Client.java (Entity class)

2. RegisterClient

register.jsp

registerServlet.java

ClientDao.java

new Client，Set its attributes from front page's textboxes

bool registerClient(Client)

check is this client already exist

insert a new Client row, save to tbCient table.

if (success)

{

out.println("alert(\"Success registered\")");

res.sendredirect("clientMain.jsp");

} else

{

show dialog(username has been exisits)

}

Client.java

3.Client Main Interface

clientMain.jsp

ViewMyProfile -- hypterlink to 3.1:View Client Information

Reset Password -- hypterlink to 3.2:Edit Client Information

View Account -- <a href="clientServlet">View Accounts</a>

Deposit -- hypterlink to 3.4 Deposit Check

clientServlet.java

doGet(){

AccountDao.java

getAccountsByClient(clientUsername)

return accounts = List<Account>

Account.java (Entity class)

if (accounts.size()>0)

{

str = ...// Generate accounts table (html)

out.println(str);

}

}

3.1 Client information

clientInfo.jsp

clientInfoServlet.java

ViewProfile

doGet()

clientDao.java

Client c= getClient(username) --username from session

//In front page these attributes are Tags, servlet set the values of the tags.

EditProfile -- url to 3.2 Edit Customer Info

3.2 Edit Client Information

editClientInfo.jsp

editClientInfoServlet.java

doGet()

clientDao.getClient(username) get Client Info

fill to front page's textboxes

Click "Confirm" button, Confirm link to editClientInfoServlet.java

doPost()

Get Client's attributes from front page

c= new Client(...)

clientDao.updateClientProfile(username, c)

if success

redirect to 3.1 View Customer Info: res.sendredirect("clientInfo.jsp");

3.3 Open New Account

openAccount.jsp

openAccountServlet.java

doPost

accountType= get account type from front page: checking / saving

AccountDao.java

Account a = openAccount(username, int accountTypeId)

Generate a Random accountNumber

if(a!=null)

MessageBox("Success")

jump to 3.clientInfo.jsp

3.4 Deposit check

depositCheck.jsp

depositCheckServlet.java

doGet()

get accounts list (dropdown list: account number(s) )

deposit(String accountNumber, int amount) AccountDao.deposit(),

TransactionDao.addTransaction()

put deposit amount to backend.

Show messagebox (Success,Deposit amount), jump to 3.5 Account Infomation

AccountDao.java

deposit(String accountNumber, int amount)

TransactionDao.java

addTransaction(String accountNumber, int trsactionType, double amount, String description)

3.5 Account Infomation

accountDetail.jsp

accountDetailServlet.java

doGet()

Display current account info.

AccountDao.java

getAccount(session.accountNumber)

Display all info.

makeTransfer(URL) -- jump to 3.5.1 Transfer Money

viewStatement(URL) -- jump 3.5.2 View Statement

3.5.1 Transfer Money

makeTransfer.jsp

makeTransferServlet.java

makeTransfer(String fromAccountNumber, String toAccountNumber, doulbe amount)

AccountDao.java

checkAccountExist(String accountNumber) -- check whether this account exist?

makeTransfer(String fromAccountNumber, String toAccountNumber, doulbe amount)

update tbAccount table, change balance - transfer money amount

TransactionDao.java

addTransaction(String accountNumber, int trsactionType, double amount, String description)

insert 2 records to Transaction table ( fromAccount - amount, toAccount + amount)

btnCancel -- jump to 3.5 Account Infomation

3.5.2 View Statement

viewStatement.jsp

viewStatementServlet.java

TransactionDao.java

getTransactionRecords(String accountNumber, date fromDate, date endDate)

return List<Transaction> from D.B.

exportReport(List<Transaction>, String filePath)

export a excel file

4 Admin Main

adminMain.jsp

adminMainServlet.java

doGet()

searchClient(String name)

if(name == null)

return List<table\_Of\_All\_Clients>;

else return List<Client> from D.B.

adminMainViewClientServlet.java

doGet()

view(String userName)

-- get userName from front page, jump to 4.1 with the arg userName

4.1 Client Info (Admin)

adminClientInfo.jsp

adminClientInfoServlet.java

deleteClient(String userName) // btn

TransactionDao.java

deleteByClient(userName)

remove(from table Transaction where username = 'userName');

AccountDao.java

deleteByClient(userName)

remove(from table Account where username = 'userName');

ClientDao.java

delete(userName)

remove(from table Client where username = 'userName');

clientDao.java

resetPassword(String userName) // btn

alert("reset succes");

Table tbClient.pw = default where username="username"; //reset password

AccountDao.java

frozen(String accountNumber)

editable = false;

activeAccount(String accountNumber)

editable = true;

deleteAccount(String accountNumber)

remove(from table Account where aid = 'accountNumber')