**University of Westminster**

**Department of computing**

**Module No :** **5COSCO1OC.2**

**Module name : Client Server Architecture**

**Course work type : Group course work**

|  |  |  |
| --- | --- | --- |
| ***Student name*** | ***Student ID*** | ***UOW ID*** |
| **Maleesha Dilnath Wijeratne** | **2015355** | **W1610087** |
| **Isuri Anuradha** | **2015183** | **W1582947** |
| **Manul Radula Singhe** | **2014254** | **W1583074** |
| **Nuwan Herath** | **2015327** | **W1608501** |

**Acknowledgement**

We would like to express my greatest appreciation to all those who gave me the possibility to complete

this implementation and report.

A special gratitude, we give to our module leader, Mr. Nalaka Dissanayaka whose encouragement

Helped us lot to coordinate this assignment.

**Table of content**

1. Acknowledgement …………………………………………………………………………………………02
2. Table of Content …………………………………………………………………………………………03
3. Figure List …………………………………………………………………………………………………….04
4. Introduction ………………………………………………………………………………………………….05
5. UML Diagrams ………………………………………………………………………………………………06
   1. Use case Diagram and Description …………………………………………………………..07
   2. Sequence Diagram 01 ……………………………………………………………………………..09
   3. Sequence Diagram 02 …………………………………………………………………………….10
   4. Sequence Diagram 03 …………………………………………………………………………….11
   5. Sequence Diagram 04 …………………………………………………………………………….12
6. Implementation ……………………………………………………………………………………………..14
   1. Web Service …………………………………………………………………………………………...15
   2. Web Client ………………………………………………………………………………………………22
7. Screen Shots (GUI) ………………………………………………………………………………………..28
8. Conclusion ……………………………………………………………………………………………………..31
9. Work Load Matrix ………………………………………………………………………………………….32

**List of figures**

1. Use case Diagram ……………………………………………………………………………………………07
2. Sequence diagram for login …………………………………………………………………………….09
3. Sequence diagram for employee operations ……………………………………………………10
4. Sequence diagram for customer list ………………………………………………………………..11
5. Sequence diagram for customer operations ……………………………………………………..12
6. Screenshots – Login and Select Pages ………………………………………………………………29
7. Screenshots – Customer List and Customer Details pages …………………………………30
8. Screenshots - Employee Page ………………………………………………………………………….31

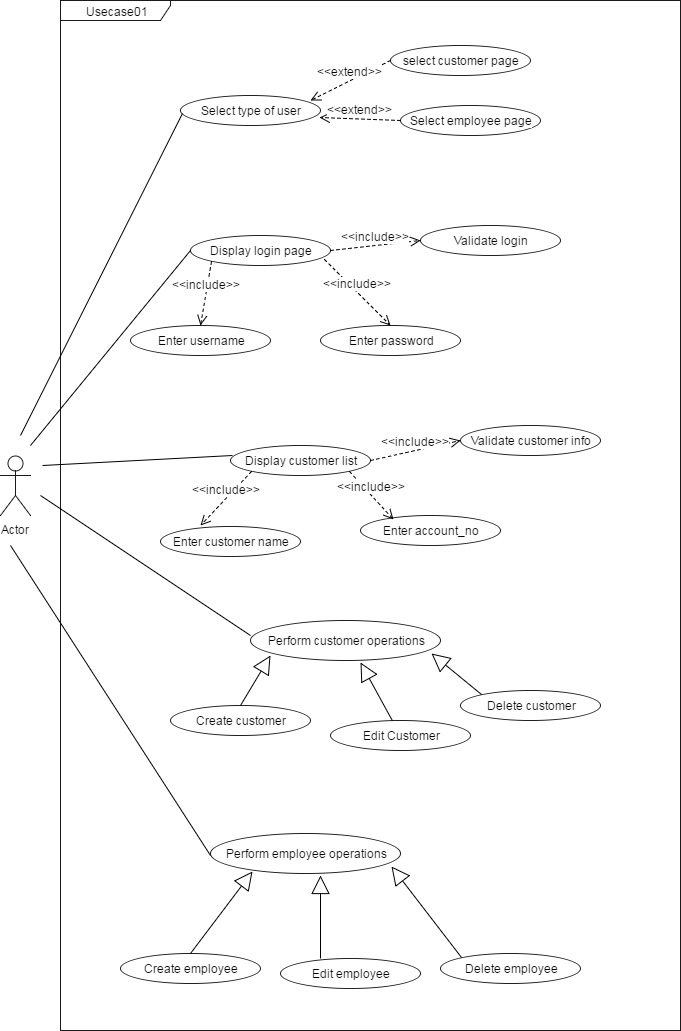
**04. Introduction**

This system will use web services to connect among the branches. There are two web services namely employee service and customer service. The customer service manages customers’ accounts performing operations like displaying customer list and performing customer’s account.

The employee service manages bank employees’ data and provides access to the customer service through the following operations like managing bank employees’ data and login/logout bank employees. Data of bank employees and customers is stored in storage, for example in a database or in file.

**05.UML Diagrams**

**5.1. Use case Diagram**



User

**Use case Description**

1. **Use case**: 001
2. **Use case name**: Configure bank account
3. **Priority :** High
4. **Actors:** User or admin
5. **Per conditions :**  Network connection must be alive
   * + - IsNetworkAlive== True
       - System should be functional
6. **Include use cases :**

* Enter username to login
* Enter password to login
* Validate username and password before login to system
* Enter customer name and account number before display customer list
* Validate name and account number before display customer list.

1. **Extend use cases:**

* User can select employee page to continue
* User can select customer page to continue

1. **Triggering event** :

* User has to select what type of operation to perform in customer page
* User has to select what type of operation to perform in employee page

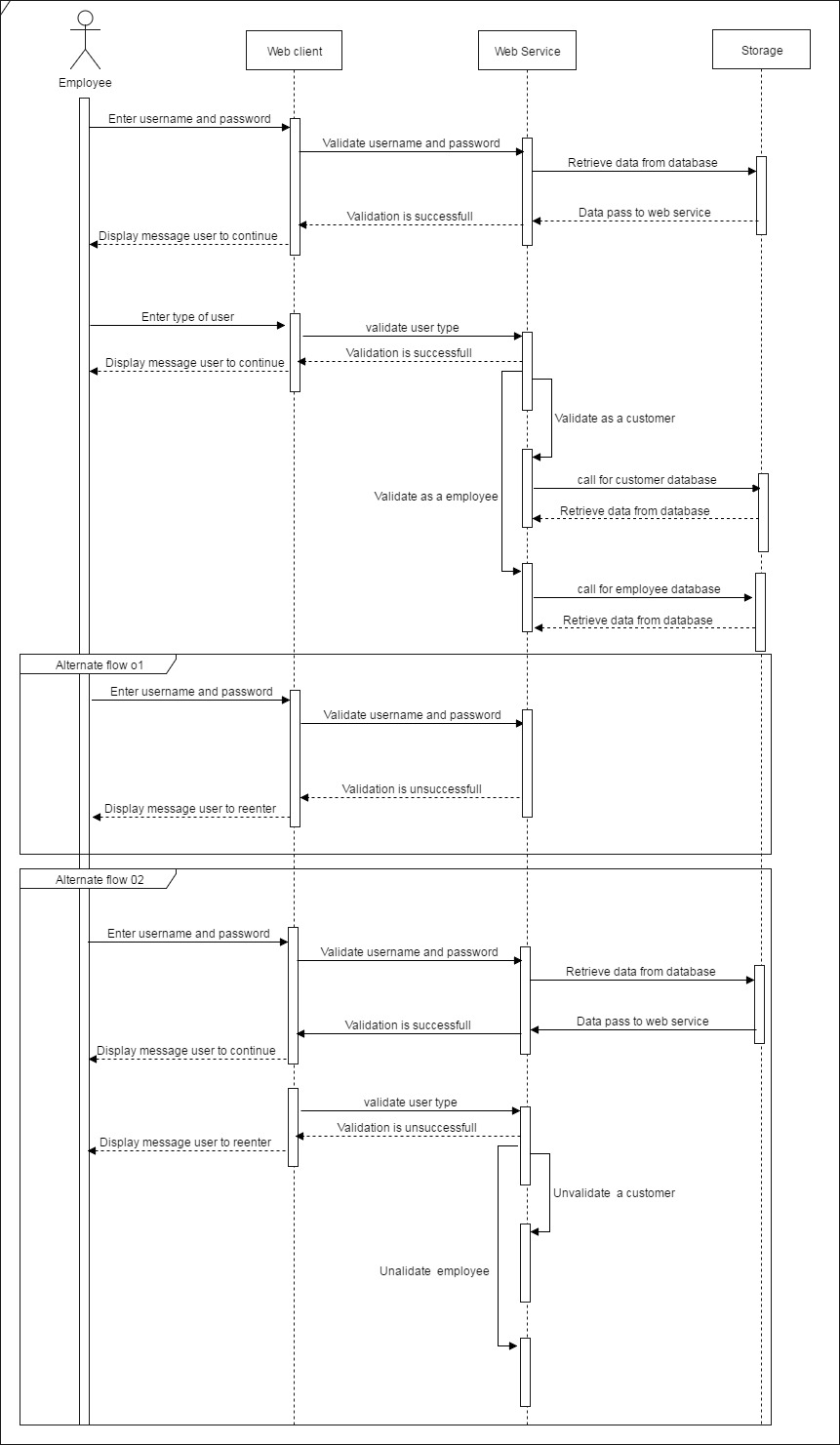
1. **Main flow** : ( Happy Day Scenario)

|  |  |
| --- | --- |
| System | Actor |
| 01.Display Login Page |  |
|  | 02.Enter username |
|  | 03.Enter password |
| 04.Validate login |  |
| 05. Display to select type |  |
|  | 06.Select customer page |
| 07.Display to enter username and account number |  |
|  | 08.Enter customer name and account number |
| 09.Validate the account number |  |
| 10.Display customer page to perform operations |  |
|  | 11. Select operation to be process |
| 12.Perform operation and Display |  |
|  | 13.Select button logout |
| 14.Logout from the system |  |

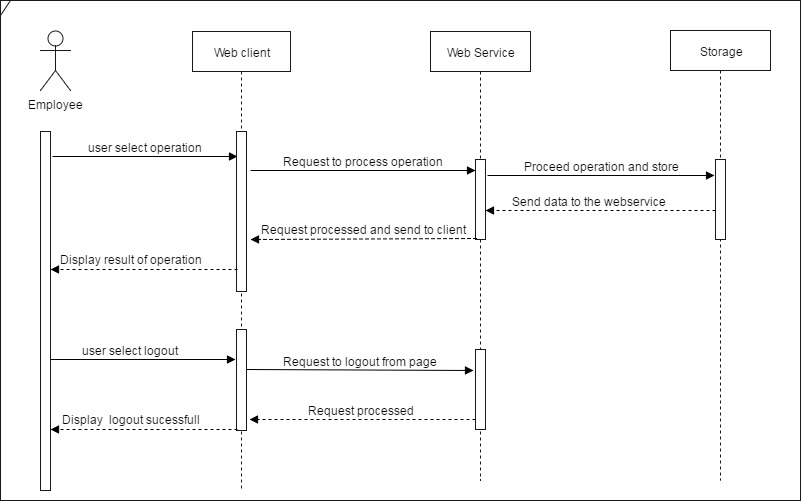
Alternate Scenario #1:

|  |  |
| --- | --- |
| System | Actor |
| 01.Display Login Page |  |
|  | 02.Enter username |
|  | 03.Enter password |
| 04.Validate login |  |
| 05. Display to select type |  |
|  | 06.Select employee page |
| 07.Display employee page to perform operations |  |
|  | 08. Select operation to be process |
| 09.Perform operation and Display |  |
|  | 10.Select button logout |
| 11.Logout from the system |  |

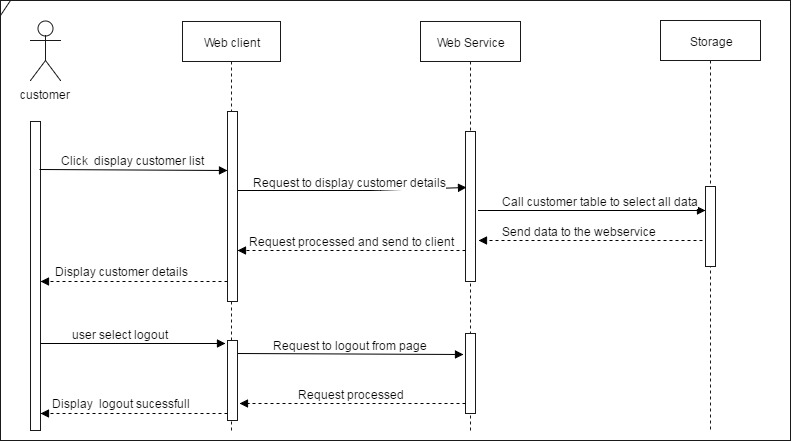
**5.2. Sequence Diagram for login**



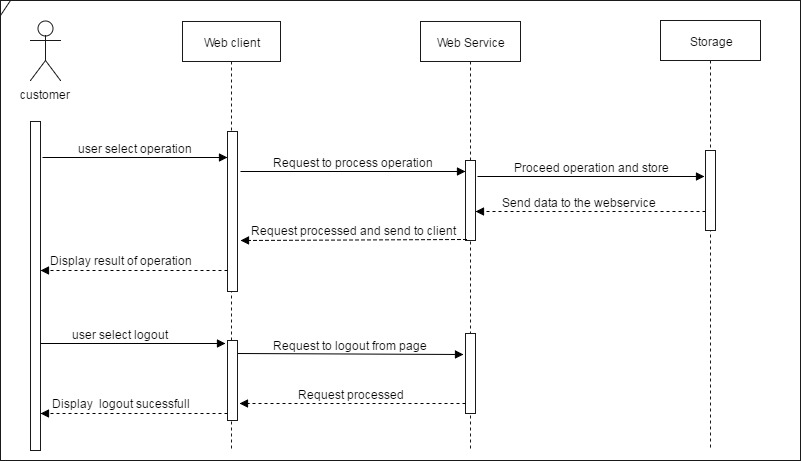
**5.3. Sequence diagram for employee operations**



**5.4. Sequence diagram for customer list**



**5.5. Sequence diagram for customer operations**



**06. Implementation**

**6.1. Web Service**

**Customer Web Service**

**@WebService(serviceName = "Customer\_service")**

**public class Customer\_service {**

**DatabaseConnectionTest db = new DatabaseConnectionTest();//creating a new DatabaseConnection objrct**

**Statement stmt;**

**@WebMethod(operationName = "create\_customer")**

**public boolean create\_customer(@WebParam(name = "name") String name, @WebParam(name = "birthday") String birthday,**

**@WebParam(name = "address") String address, @WebParam(name = "mobile") int mobile, @WebParam(name = "eMail") String eMail, @WebParam(name = "accountType") String accountType,**

**@WebParam(name = "accountNumber") String accountNumber, @WebParam(name = "sortCode") String sortCode, @WebParam(name = "balance") String balance, @WebParam(name = "card") String card) {**

**try {**

**PreparedStatement state = db.db\_Connect().prepareStatement("insert into customer values(?,?,?,?,?,?,?,?,?,?)");//insert data**

**state.setString(1, name);**

**state.setString(2, birthday);**

**state.setString(3, address);**

**state.setInt(4, mobile);**

**state.setString(5, eMail);**

**state.setString(6, accountType);**

**state.setString(7, accountNumber);**

**state.setString(8, sortCode);**

**state.setString(9, balance);**

**state.setString(10, card);**

**state.executeUpdate();**

**return true;**

**} catch (Exception e) {**

**return false;**

**}**

**}**

**@WebMethod(operationName = "display\_cus\_list")**

**public String[] display\_cus\_list() {**

**String[] cust;**

**try {**

**//try{**

**Statement state = db.db\_Connect().createStatement();//retrieves data**

**ResultSet rs = state.executeQuery("SELECT COUNT(\*) FROM customer");**

**rs.next();**

**int rowCount = rs.getInt(1);**

**System.out.println(rowCount);**

**String sql = "SELECT \* FROM customer";**

**ResultSet rs1 = state.executeQuery(sql);**

**rs1.next();**

**cust = new String[rowCount];**

**int x = 0;**

**while (x < rowCount) {**

**cust[x] = rs1.getString("name") + "-" + rs1.getString("accountNumber") + "-" + rs1.getString("birthday") + "-" + rs1.getString("address")**

**+ "-" + rs1.getString("mobile") + "-" + rs1.getString("email") + "-" + rs1.getString("accountType")**

**+ "-" + rs1.getString("sortCode") + "-" + rs1.getString("balance")**

**+ "-" + rs1.getString("card");**

**rs1.next();**

**x++;**

**}**

**return cust;**

**} catch (SQLException ex) {**

**Logger.getLogger(Customer\_service.class.getName()).log(Level.SEVERE, null, ex);**

**}**

**return null;**

**}**

**@WebMethod(operationName = "delete\_customer")**

**public boolean delete\_customer(@WebParam(name = "name") String name, @WebParam(name = "accountNumber") String accountNumber) {**

**try {**

**PreparedStatement state = db.db\_Connect().prepareStatement("delete from customer where name=? AND accountNumber=?"); //delete specific row**

**state.setString(1, name);**

**state.setString(2, accountNumber);**

**state.executeUpdate();**

**return true;**

**} catch (Exception e) {**

**return false;**

**}**

**}**

**@WebMethod(operationName = "update\_customer")**

**public boolean update\_customer(@WebParam(name = "name") String name, @WebParam(name = "birthday") String birthday,**

**@WebParam(name = "address") String address, @WebParam(name = "mobile") int mobile, @WebParam(name = "email") String mail,**

**@WebParam(name = "accountType") String accountType, @WebParam(name = "sortCode") String sortCode,**

**@WebParam(name = "balance") String balance, @WebParam(name = "card") String card, @WebParam(name = "accountNumber") String acNumber) {**

**try {**

**stmt = db.db\_Connect().createStatement();**

**String sql\_command = "Update customer set name=?, birthday=?, address=?, mobile=?, email=?, accountType=?,sortCode=?,balance=?,card=? WHERE accountNumber=?";**

**PreparedStatement pst = db.db\_Connect().prepareStatement(sql\_command);**

**pst.setString(1, name);**

**pst.setString(2, birthday);**

**pst.setString(3, address);**

**pst.setInt(4, mobile);**

**pst.setString(5, mail);**

**pst.setString(6, accountType);**

**pst.setString(7, sortCode);**

**pst.setString(8, balance);**

**pst.setString(9, card);**

**pst.setString(10, acNumber);**

**int updateResult = pst.executeUpdate();**

**if (updateResult == 1) {**

**} else {**

**}**

**// return false;**

**} catch (Exception e) {**

**return false;**

**}**

**return true;**

**}**

**}**

**Customer Web Service**

**@WebService(serviceName = "employee\_ser")**

**public class employee\_ser {**

**DatabaseConnectionTest db = new DatabaseConnectionTest();//creating a new DatabaseConnection objrct**

**Statement stmt;**

**@WebMethod(operationName = "login")**

**public boolean login(@WebParam(name = "username") String username, @WebParam(name = "password") String password) throws InvalidInputException {**

**try {**

**PreparedStatement pst = db.db\_Connect().prepareStatement("Select \* from employee where username=? and password=?");**

**pst.setString(1, username);**

**pst.setString(2, password);**

**ResultSet rs = pst.executeQuery();**

**if (rs.next()) {**

**return true;**

**} else {**

**throw new InvalidInputException("Wrong input credentials!", username + "or" + password + "is incorrect");**

**}**

**// return false;**

**} catch (Exception e) {**

**return false;**

**}**

**@WebMethod(operationName = "create\_employee")**

**public boolean create\_employee(@WebParam(name = "name") String name, @WebParam(name = "position") String position, @WebParam(name = "username") String username, @WebParam(name = "Password") String Password, @WebParam(name = "repeatpassword") String repeatpassword) {**

**try {**

**PreparedStatement state = db.db\_Connect().prepareStatement("insert into employee values(?,?,?,?,?)");//insert data**

**state.setString(1, name);**

**state.setString(2, position);**

**state.setString(3, username);**

**state.setString(4, Password);**

**state.setString(5, repeatpassword);**

**state.executeUpdate();**

**return true;**

**} catch (Exception e) {**

**return false;**

**}**

**}**

**@WebMethod(operationName = "delete")**

**public boolean delete(@WebParam(name = "username") String username, @WebParam(name = "password") String password) {**

**try {**

**PreparedStatement state = db.db\_Connect().prepareStatement("DELETE from employee where username=? AND password=?"); //delete specific row**

**state.setString(1, username);**

**state.setString(2, password);**

**int rs = state.executeUpdate();**

**if (rs == 1) {**

**return true;**

**} else {**

**return false;**

**}**

**} catch (Exception e) {**

**return false;**

**}**

**}**

**@WebMethod(operationName = "update\_emp")**

**public boolean update\_emp(@WebParam(name = "name") String name, @WebParam(name = "position") String position,**

**@WebParam(name = "username") String username, @WebParam(name = "Password") String Password, @WebParam(name = "repeatpassword") String repeatpassword) {**

**try {**

**stmt = db.db\_Connect().createStatement();**

**String sql\_command = "Update employee set name=?, position=?, password=?, repeatpassword=? WHERE username=?";**

**PreparedStatement pst = db.db\_Connect().prepareStatement(sql\_command);**

**pst.setString(1, name);**

**pst.setString(2, position);**

**pst.setString(3, Password);**

**pst.setString(4, repeatpassword);**

**pst.setString(5, username);**

**int updateResult = pst.executeUpdate();**

**if (updateResult == 1) {**

**return true;**

**} else {**

**throw new InvalidInputException("Wrong input credentials!", username + "or" + Password + "is incorrect");**

**}**

**// return false;**

**} catch (Exception e) {**

**return false;**

**}**

**}**

**}**

**6.2. Web Client**

**Employee\_page Class**

private void addEmployeeBtnActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = createEmployee(lblName.getText(), lblPosition.getText(), lblUsername.getText(),

lblPass.getText(), lblrepeatPass.getText());

if (status) {

statuslbl.setForeground(Color.GREEN);

statuslbl.setText("Employee Added Successfully!");

} else {

statuslbl.setForeground(Color.red);

statuslbl.setText("FAILED!");

}

}

private void updateEmpBtnActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = updateEmp(lblName.getText(), lblPosition.getText(), lblUsername.getText(),

lblPass.getText(), lblrepeatPass.getText());

if (status) {

statuslbl.setForeground(Color.GREEN);

statuslbl.setText("Employee Updated Successfully!");

} else {

statuslbl.setForeground(Color.red);

statuslbl.setText("FAILED!");

}

}

private void lblNameActionPerformed(java.awt.event.ActionEvent evt) {

}

private void deleteEmpBtnActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = delete(lblUsername.getText(), lblPass.getText());

if (status) {

statuslbl.setForeground(Color.GREEN);

statuslbl.setText("Employee Deleted Successfully!");

} else {

statuslbl.setForeground(Color.red);

statuslbl.setText("FAILED!");

}

}

**Login\_page Class**

private void loginbtnActionPerformed(java.awt.event.ActionEvent evt) {

userName = usernameField.getText();

password = passwordField.getText();

try {

status = login(userName, password);

} catch (InvalidInputException\_Exception ex) {

Logger.getLogger(Login\_page.class.getName()).log(Level.SEVERE, null, ex);

}

if(status){

lblStaus.setForeground(Color.GREEN);

lblStaus.setText("Login Success!");

dispose();

new employee\_client.select\_type().setVisible(true);

}else{

lblStaus.setForeground(Color.RED);

lblStaus.setText("Login FAILED!");

}

}

**Customer\_page Class**

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = createCustomer(txtName.getText(), txtBday.getText(), txtaddress.getText(), Integer.parseInt(txtmob.getText()), txtMail.getText(), txtAcType.getText(), txtAC.getText(), txtSort.getText(), txtBalance.getText(), txtcard.getText());

if (status) {

lblStatus.setText("Customer Added Successfully!");

} else {

lblStatus.setText("FAILED!");

}

}

private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = updateCustomer(txtName.getText(), txtBday.getText(), txtaddress.getText(), Integer.parseInt(txtmob.getText()), txtMail.getText(), txtAcType.getText(), txtSort.getText(), txtBalance.getText(), txtcard.getText(), txtAC.getText());

if (status) {

lblStatus.setText("Customer Updated Successfully!");

} else {

lblStatus.setText("FAILED!");

}

}

private void txtNameActionPerformed(java.awt.event.ActionEvent evt) {

}

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {

boolean status = deleteCustomer(txtName.getText(), txtAC.getText());

if (status) {

lblStatus.setText("Customer Deleted Successfully!");

} else {

lblStatus.setText("FAILED!");

}

}

private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {

dispose();

new customerList\_page().setVisible(true);

}

**CustomerList\_page Class**

public customerList\_page() {

initComponents();

DefaultListModel dm = new DefaultListModel();

list = displayCusList();

System.out.println(list);

for (String var : list) {

String[] words = var.split("-");

dm.addElement(words[0]+" - "+words[1]);

}

jList1.setModel(dm);

jList1.addMouseListener((MouseListener) this);

}

private void btnAddCusActionPerformed(java.awt.event.ActionEvent evt) {

dispose();

new Customer\_page().setVisible(true);

}

private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {

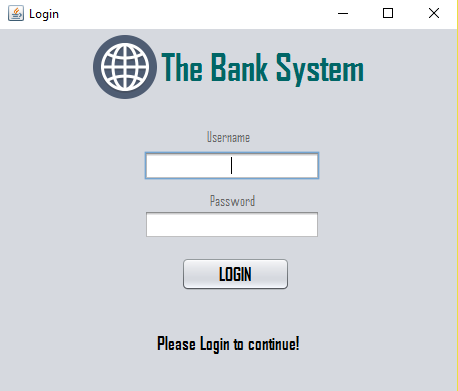
dispose();

new select\_type().setVisible(true);

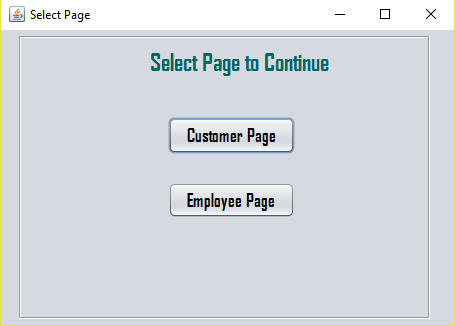
}

**07. Screen Shots**

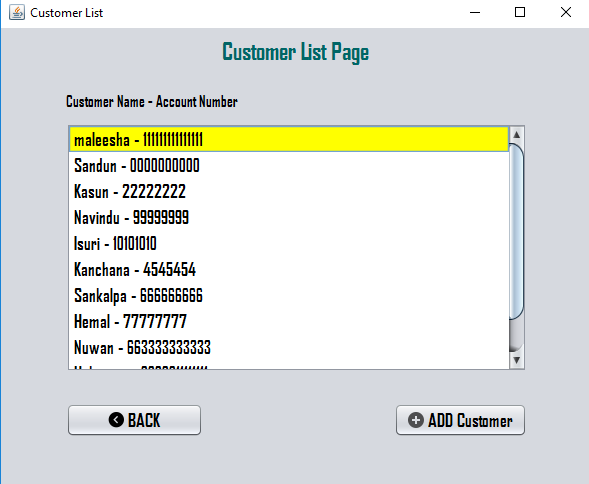
**Login Screen**



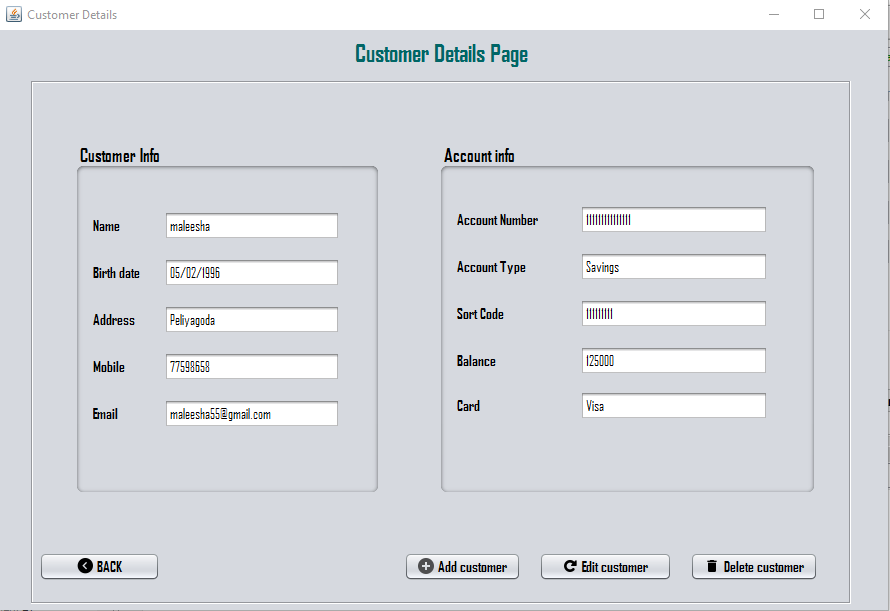
**Select Page**



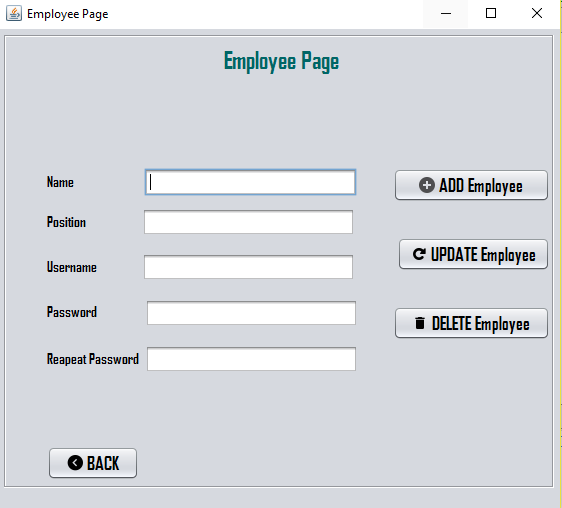
**Customer List Page**



**Customer Details Page**



**Employee Page**



**08. Conclusion**

By the completing this project we got knowledge about how web client and web service is integrating and also from java how easy to develop such a client application. We were able to complete the given task without any delay.

**09. Work Load Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Number** | **Student name** | **Student ID** | **UOW ID** |
| **Task 01** | **Nuwan Herath** | **2015327** | **W1608501** |
| **Task 02** | **Manul Radula Singhe** | **2014254** | **W1583074** |
| **Task 03** | **Isuri Anuradha** | **2015183** | **W1582947** |
| **Task 04** | **Maleesha Dilnath Wijeratne** | **2015355** | **W1610087** |

**Task 1. Design and implement a graphical user interface (GUI).**

**Task 2. Develop the UML use case diagrams.**

**Task 3. Design and implement the bank employee and the bank customer web services.**

**Task 4. Create a web client and integrate with web services.**