



Informatics Institute of Technology
5COSC010C Client Server Architecture

Group Coursework

Done by :-

Keshav Kumaresan (2014016) (w1582991)

Manuka Maduranga (2015313) (w1608496)

Sudam Dissanayake (2015096) (w1582963)

Khopithan Sathiyakeerthy (2015245) (w1608467)

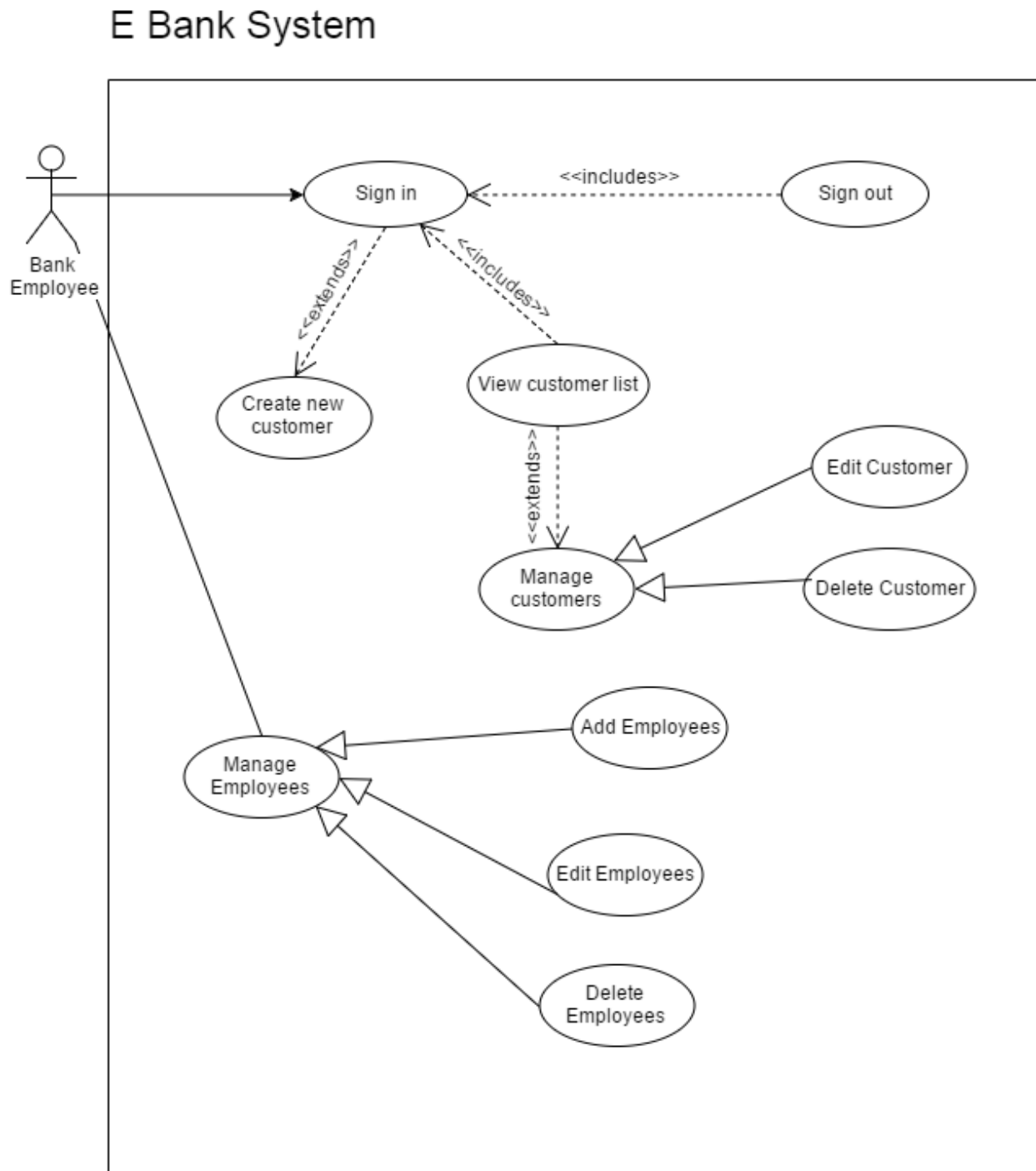
Sonal Muthukumarana (2015347) (w1608505)

Contents

UML Diagrams	3
Use case diagram	3
Sequence Diagrams.....	4
Web Services Code	8
Customer Web Service	12
Web Client Code.....	18
GUI Screenshots	28
Work Load Matrix	34

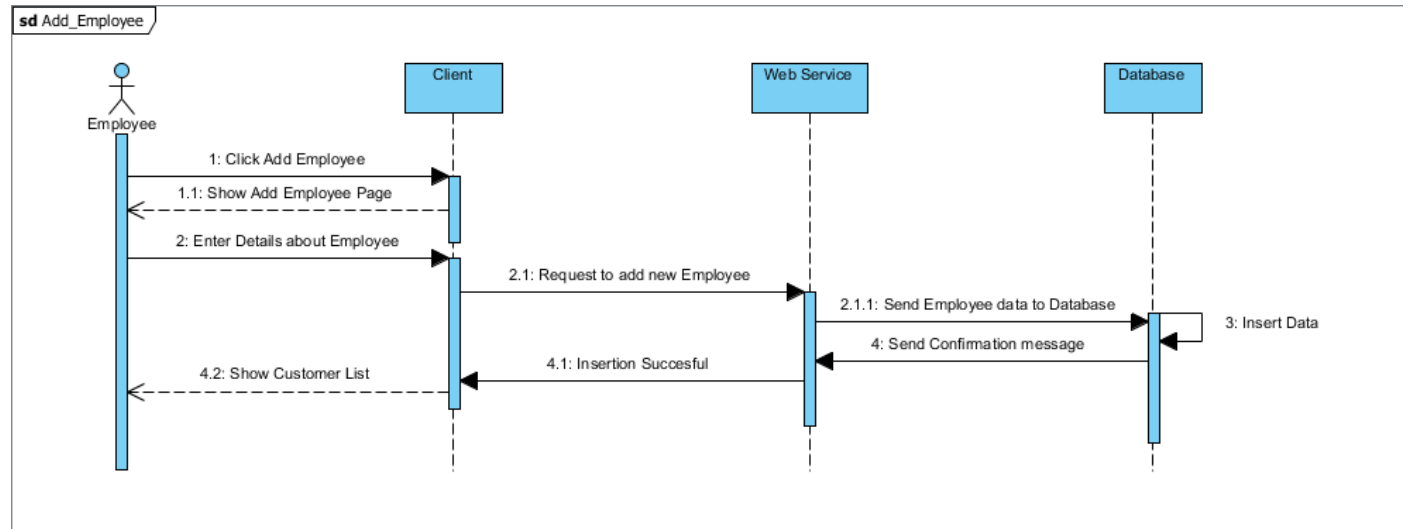
UML Diagrams

Use case diagram

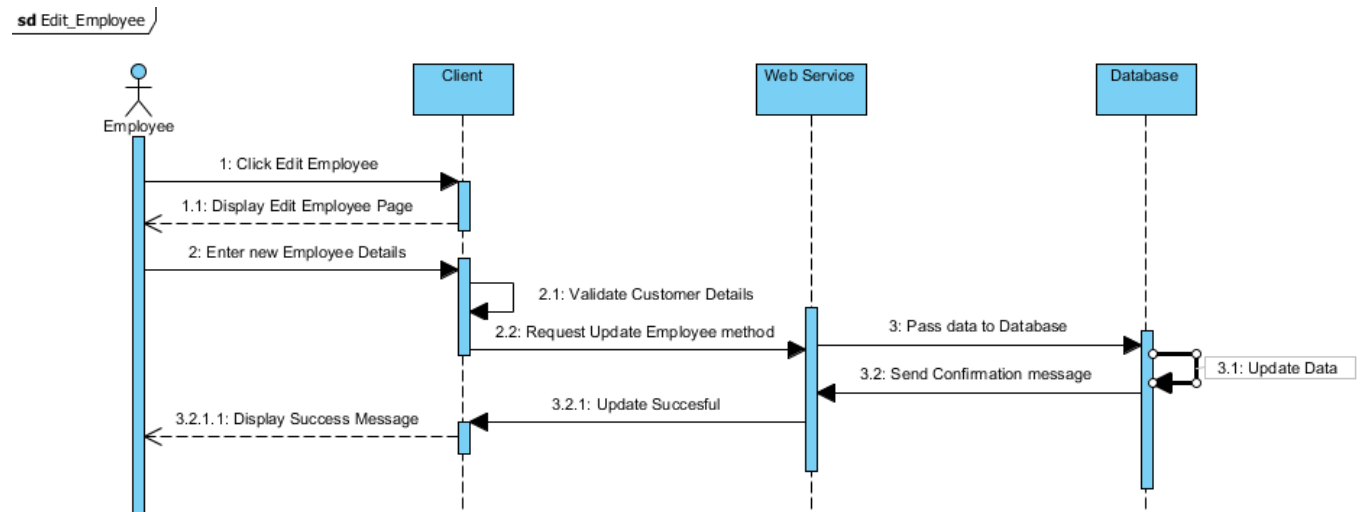


Sequence Diagrams

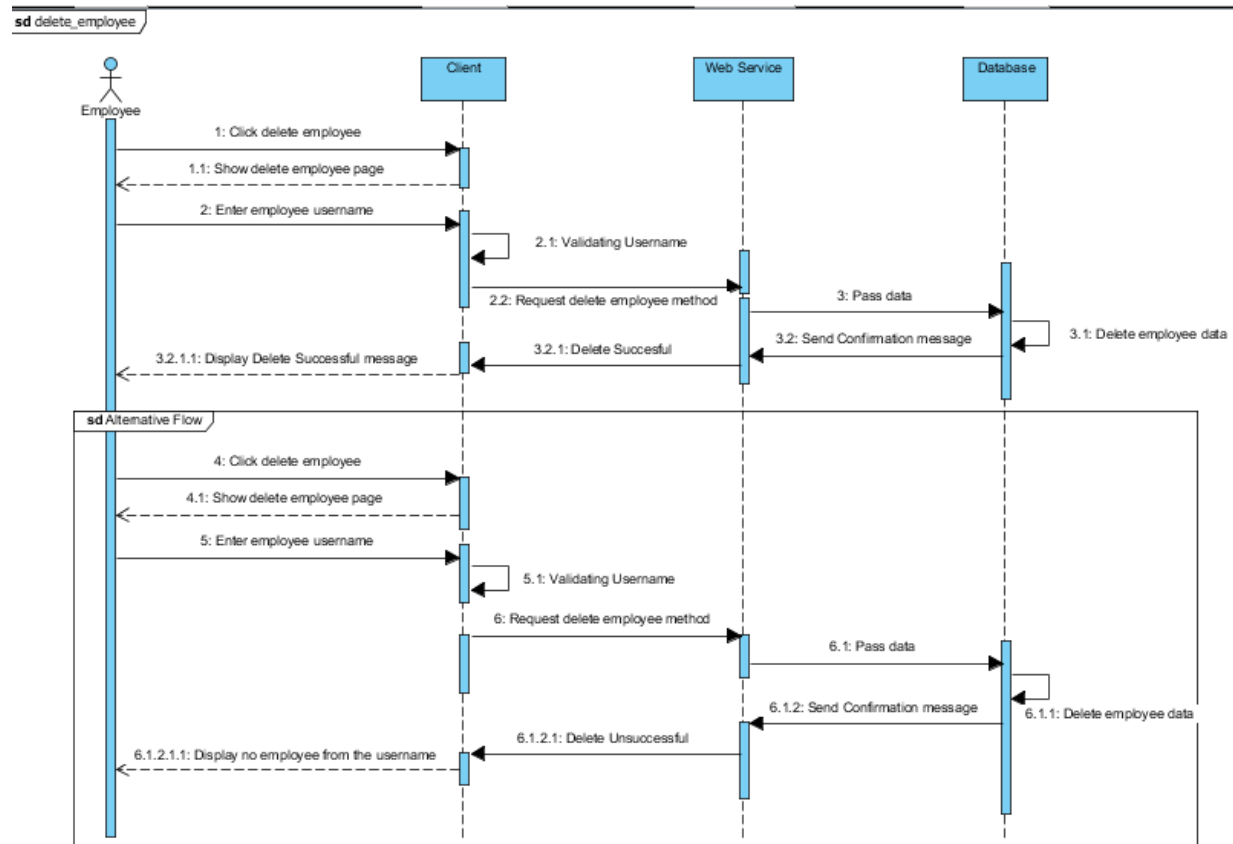
Adding an Employee



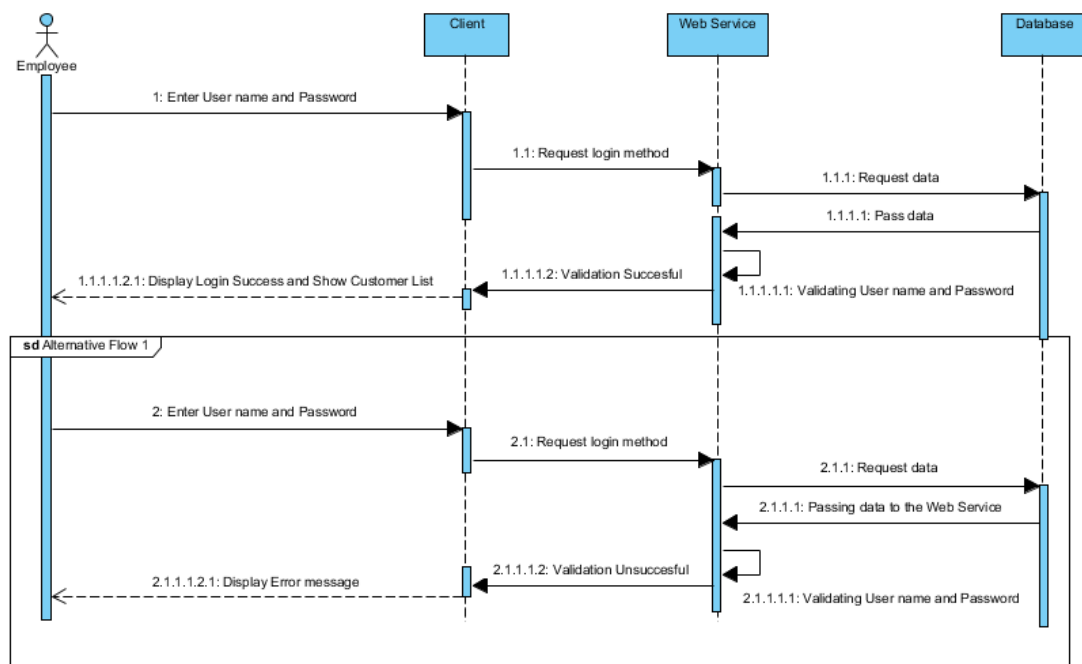
Editing an Employee



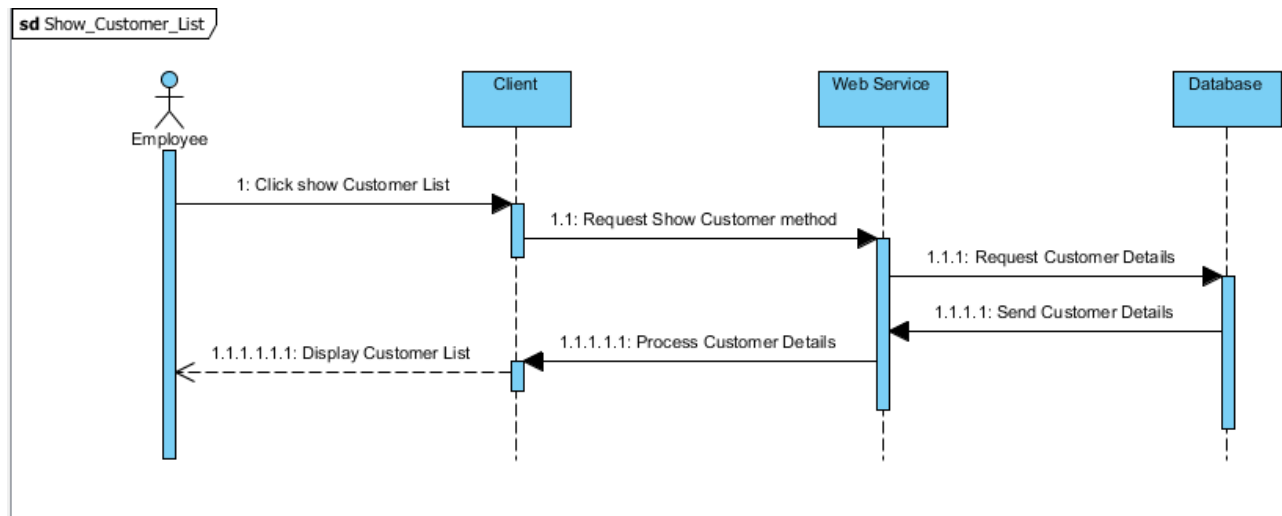
Deleting an Employee



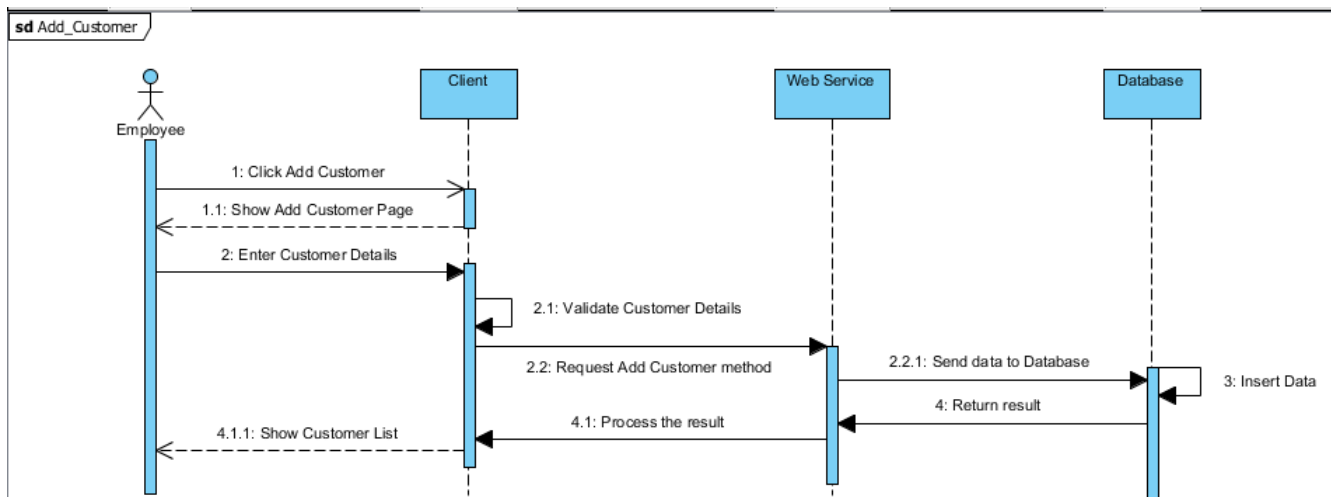
Logging In



Displaying Customer List

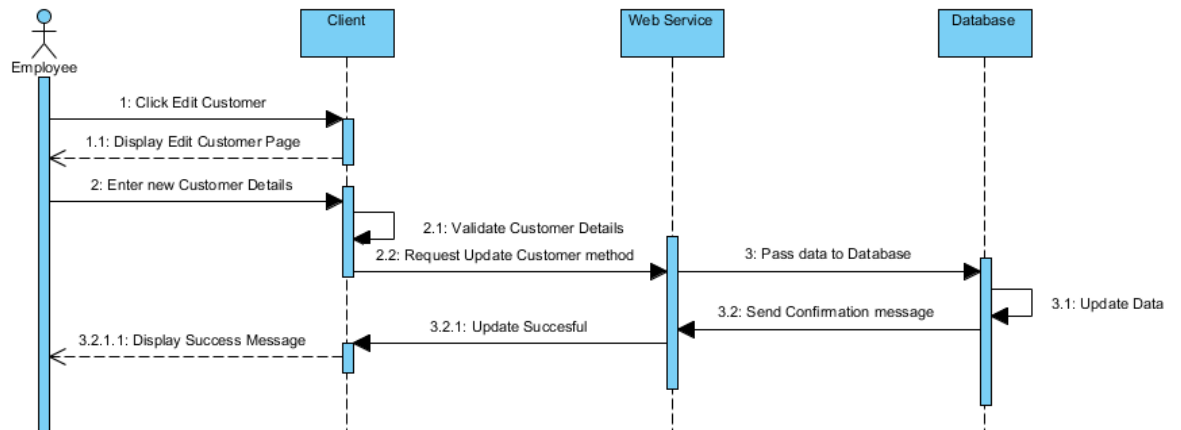


Add Customer



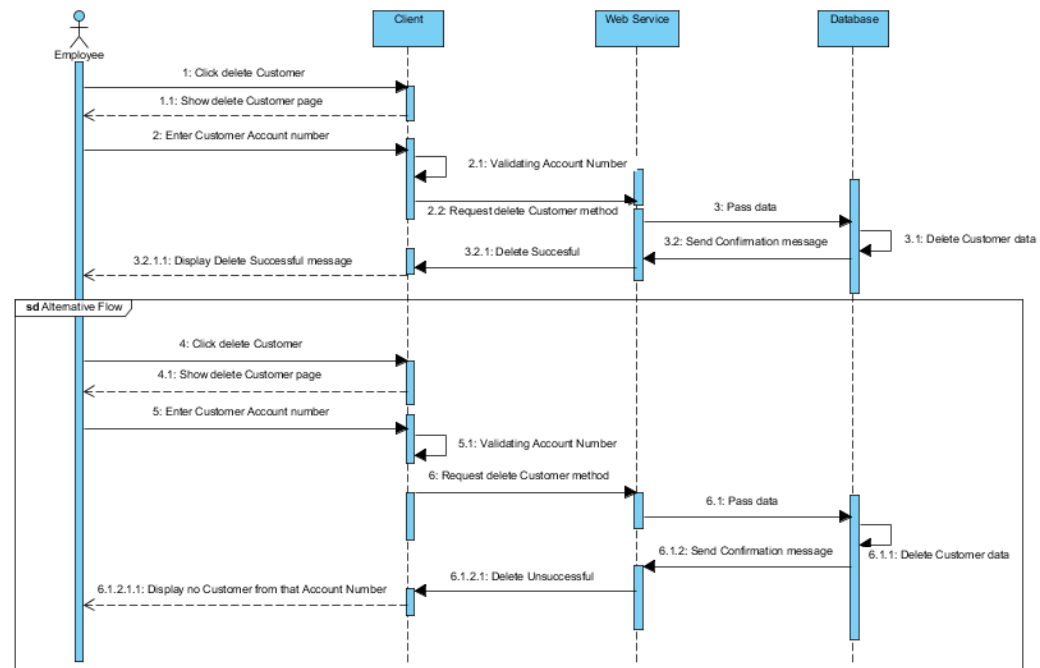
Edit Customer

sd update_customer



Delete Customer

sd delete_customer



Web Services Code

Employee Web Service

```
package login_ws;

import static com.sun.corba.se.impl.util.Utility.printStackTrace;
import javax.xml.ws.WebService;
import javax.xml.ws.WebMethod;
import javax.xml.ws.WebParam;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Manuka
 */
@WebService(serviceName = "Login")
public class Login {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "login_employee")
    public String login(@WebParam(name = "UserName") String UserName, @WebParam(name = "Password") String Password) {
        String result = "Pending";
        try {
            //TODO write your implementation code here:
            Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
            PreparedStatement stmt= con.prepareStatement("SELECT * FROM EMPLOYEES WHERE UserName = ? AND Password = ?");
            stmt.setString(1,UserName);
            stmt.setString(2, Password);
            ResultSet results = stmt.executeQuery();
            if(results.next()){
                result = "Login Success";
                con.close();

            } else {
                result = "Login Unsuccess";
                con.close();
            }

        } catch (SQLException ex) {
            Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
        }
        return result;
    }
}
```



```

@WebMethod(operationName = "delete_employee")
public String delete_employee(@WebParam(name = "userName") String userName) {
    String result = "Unsuccesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("DELETE FROM EMPLOYEES WHERE USERNAME = ?");
        stmt.setString(1, userName);
        int i = stmt.executeUpdate();
        result = "Delete Succesful";
        con.close();
    } catch (SQLException e) {
        printStackTrace();
    }
    return result;
}

@WebMethod(operationName = "show_employees")
public List<Employees> showEmployees() {
    List<Employees> employeeList = new ArrayList();
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("SELECT * FROM Employees");
        ResultSet results = stmt.executeQuery();
        while (results.next()) {
            String name = results.getString("Name");
            String position = results.getString("Position");
            String username = results.getString("UserName");
            String password = results.getString("Password");
            Employees employee = new Employees(name, position, username, password);
            employeeList.add(employee);
        }
        con.close();
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }

    return employeeList;
}

@WebMethod(operationName = "add_employees")
public String add_employees(@WebParam(name = "name") String name, @WebParam(name = "position") String position, @WebParam(name = "username")
String username, @WebParam(name = "password") String password) {
    String result = "Not succesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("INSERT INTO EMPLOYEES (Name,Position,UserName>Password)"
+ "Values (?, ?, ?, ?)");
        stmt.setString(1, name);
        stmt.setString(2, position);
        stmt.setString(3, username);
        stmt.setString(4, password);

        int i = stmt.executeUpdate();
        result = "Insert Succesful";
        con.close();
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }

    return result;
}

```

```

@WebMethod(operationName = "update_employee")
public String update_employee(@WebParam(name = "name") String name, @WebParam(name = "position") String position,
    @WebParam(name = "Newusername") String newUsername, @WebParam(name = "password") String password,
    @WebParam(name = "Oldusername") String oldUsername) {
    String result = "Unsuccesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("UPDATE EMPLOYEES SET NAME = ?, POSITION = ?, USERNAME = ?, PASSWORD= ? WHERE USERNAME = ? ");
        stmt.setString(1, name);
        stmt.setString(2, position);
        stmt.setString(3, newUsername);
        stmt.setString(4, password);
        stmt.setString(5, oldUsername);
        int i = stmt.executeUpdate();
        result = "Update Succesful";
        con.close();
    } catch (SQLException e) {
        printStackTrace();
    }
    return result;
}
}

```

Employees Class

```
package login_ws;

/**
 *
 * @author Manuka
 */
public class Employees {
    private String name;
    private String position;
    private String username;
    private String password;

    Employees(String name, String position, String username, String password){
        this.name = name;
        this.position = position;
        this.username = username;
        this.password = password;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @param name the name to set
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * @return the position
     */
    public String getPosition() {
        return position;
    }

    /**
     * @param position the position to set
     */
    public void setPosition(String position) {
        this.position = position;
    }

    /**
     * @return the username
     */
    public String getUsername() {
        return username;
    }

    /**
     * @param username the username to set
     */
    public void setUsername(String username) {
        this.username = username;
    }

    /**
     * @return the password
     */
    public String getPassword() {
        return password;
    }

    /**
     * @param password the password to set
     */
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Customer Web Service

```
package customer_ws;

import static com.sun.corba.se.impl.util.Utility.printStackTrace;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import login_ws.Login;

/**
 *
 * @author Manuka
 */
@WebService(serviceName = "Customer_ws")
public class Customer_ws {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "show_Customers")
    public List<Customers> showCustomers() {
        List<Customers> customerList = new ArrayList();
        try {
            //TODO write your implementation code here:
            Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
            PreparedStatement stmt = con.prepareStatement("SELECT * FROM Customers");
            ResultSet results = stmt.executeQuery();
            while (results.next()) {
                String name = results.getString("FULLNAME");
                String address = results.getString("ADDRESS");
                String birthDate = results.getString("DATEOFBIRTH");
                String mobile = results.getString("MOBILE");
                String email = results.getString("EMAIL");
                String accountType = results.getString("ACCOUNTTYPE");
                String accountNumber = results.getString("ACCOUNTNUMBER");
                String sortCode = results.getString("SORTCODE");
                String balance = results.getString("BALANCE");
                String card = results.getString("CARD");

                Customers customer = new Customers(name, balance, card, email, address, mobile, birthDate, accountType, accountNumber, sortCode);
                customerList.add(customer);
            }
            con.close();
        } catch (SQLException ex) {
            Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
        }

        return customerList;
    }
}
```

```

@WebMethod(operationName = "add_Customer")
public String Add_Customer(@WebParam(name = "name") String name, @WebParam(name = "address") String address,
    @WebParam(name = "mobile") String mobile, @WebParam(name = "dateOfBirth") String dateOfBirth,
    @WebParam(name = "email") String email, @WebParam(name = "accountType") String accountType,
    @WebParam(name = "accountNumber") String accountNumber, @WebParam(name = "balance") String balance,
    @WebParam(name = "sortCode") String sortCode, @WebParam(name = "card") String card) {
    String result = "Not succesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("INSERT INTO Customers (FullName,DateOfBirth,Address,"
            + "Mobile,Email,AccountType,AccountNumber,SortCode,Balance,Card)"
            + "Values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
        stmt.setString(1, name);
        stmt.setString(3, address);
        stmt.setString(4, mobile);
        stmt.setString(2, dateOfBirth);
        stmt.setString(5, email);
        stmt.setString(6, accountType);
        stmt.setString(7, accountNumber);
        stmt.setString(9, balance);
        stmt.setString(8, sortCode);
        stmt.setString(10, card);

        int i = stmt.executeUpdate();
        result = "Insert Succesful";
        con.close();
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }

    return result;
}

@WebMethod(operationName = "delete_Customer")
public String delete_Customer(@WebParam(name = "Account_number") String Account_number) {
    String result = "Unsuccesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("DELETE FROM CUSTOMERS WHERE ACCOUNTNUMBER = ?");
        stmt.setString(1, Account_number);
        int i = stmt.executeUpdate();
        result = "Delete Succesful";
        con.close();
    } catch (SQLException e) {
        printStackTrace();
    }

    return result;
}

```

```

@WebMethod(operationName = "edit_Customer")
public String edit_customer(@WebParam(name = "name") String name,
    @WebParam(name = "address") String address,
    @WebParam(name = "dateOfBirth") String dateOfBirth,
    @WebParam(name = "mobile") String mobile, @WebParam(name = "email") String email,
    @WebParam(name = "accountNumber") String accountNumber) {
    String result = "Unsuccesful";
    try {
        //TODO write your implementation code here:
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/cs_cw_db", "cw", "123");
        PreparedStatement stmt = con.prepareStatement("UPDATE CUSTOMERS SET FULLNAME = ?,"
            + "ADDRESS = ?,MOBILE = ?, DATEOFBIRTH = ?, EMAIL = ? WHERE ACCOUNTNUMBER = ? ");
        stmt.setString(1, name);
        stmt.setString(2, address);
        stmt.setString(3, dateOfBirth);
        stmt.setString(4, mobile);
        stmt.setString(5, email);
        stmt.setString(6, accountNumber);
        int i = stmt.executeUpdate();
        result = "Update Succesful";
        con.close();
    } catch (SQLException e) {
        printStackTrace();
    }
    return result;
}
}

```

Customers Class

```
public class Customers {

    private String name;
    private String address;
    private String mobile;
    private String birthDate;
    private String accountType;
    private String accountNumber;
    private String sortCode;
    private String balance;
    private String email;
    private String card;

    Customers(String name, String balance, String card, String email, String address,
        String mobile, String birthDate, String accountType, String accountNumber, String sortCode) {
        this.name = name;
        this.address = address;
        this.mobile = mobile;
        this.birthDate = birthDate;
        this.accountType = accountType;
        this.accountNumber = accountNumber;
        this.sortCode = sortCode;
        this.email = email;
        this.balance = balance;
        this.card = card;
    }

    /**
     * @return the balance
     */
    public String getBalance() {
        return balance;
    }

    /**
     * @param balance the balance to set
     */
    public void setBalance(String balance) {
        this.balance = balance;
    }

    public String getEmail() {
        return email;
    }

    /**
     * @param email the email to set
     */
    public void setEmail(String email) {
        this.email = email;
    }

    /**
     * @return the card
     */
    public String getCard() {
        return card;
    }

    /**
```

```

/**
 * @param card the card to set
 */
public void setCard(String card) {
    this.card = card;
}

/**
 * @return the name
 */
public String getName() {
    return name;
}

/**
 * @param name the name to set
 */
public void setName(String name) {
    this.name = name;
}

/**
 * @return the address
 */
public String getAddress() {
    return address;
}

/**
 * @param address the address to set
 */
public void setAddress(String address) {
    this.address = address;
}

/**
 * @return the mobile
 */
public String getMobile() {
    return mobile;
}

/**
 * @param mobile the mobile to set
 */
public void setMobile(String mobile) {
    this.mobile = mobile;
}

/**
 * @return the birtDate
 */
public String getBirtDate() {
    return birtDate;
}

/**
 * @param birtDate the birtDate to set
 */
public void setBirtDate(String birtDate) {
    this.birtDate = birtDate;
}

/**
 * @return the accountType
 */
public String getAccountType() {
    return accountType;
}

/**
 * @param accountType the accountType to set
 */
public void setAccountType(String accountType) {
    this.accountType = accountType;
}

```



```
/**
 * @return the accountNumber
 */
public String getAccountNumber() {
    return accountNumber;
}

/**
 * @param accountNumber the accountNumber to set
 */
public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

/**
 * @return the sortCode
 */
public String getSortCode() {
    return sortCode;
}

/**
 * @param sortCode the sortCode to set
 */
public void setSortCode(String sortCode) {
    this.sortCode = sortCode;
}
}
```

Web Client Code

Index Page

```
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new LoginPage().setVisible(true);  
}
```

```
private void btnEmployeeActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmployeePage().setVisible(true);  
}
```

Employee Page

```
private void btnCreateActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmpRegister().setVisible(true);  
}
```

```
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new Index().setVisible(true);  
}
```

```
private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmpEdit().setVisible(true);  
}
```

```
private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmpDelete().setVisible(true);  
}
```

Edit Employee Page

```
private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {  
    String name = txtName.getText();  
    String position = txtPosition.getText();  
    String oldUName = txtOldUsrName.getText();  
    String newUName = txtNewUsrName.getText();  
    String pWord = txtPWord.getText();  
  
    if (!newUName.equals("") || !pWord.equals("") || !name.equals("") || !position.equals("") || !oldUName.equals("")) {  
        String result = updateEmployee(name, position, newUName, pWord, oldUName);  
        if (result.equalsIgnoreCase("Update Successful")) {  
            showMessageDialog(null, "Employee Successfully Edited");  
            this.setVisible(false);  
            new EmployeePage().setVisible(true);  
        } else {  
            showMessageDialog(null, "Insrtion Failed, Check the old username");  
        }  
    } else {  
        lblErrorMessage.setText("Please fill all the fields!!");  
    }  
}  
  
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmployeePage().setVisible(true);  
}  
  
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {  
    txtName.setText("");  
    txtPWord.setText("");  
    txtOldUsrName.setText("");  
    txtNewUsrName.setText("");  
    txtPosition.setText("");  
    lblErrorMessage.setText("");  
}  
  
private static String updateEmployee(java.lang.String name, java.lang.String position, java.lang.String newusername,  
    java.lang.String password, java.lang.String oldusername) {  
    login_ws.Login_Service service = new login_ws.Login_Service();  
    login_ws.Login port = service.getLoginPort();  
    return port.updateEmployee(name, position, newusername, password, oldusername);  
}
```

Delete Employee Page

```
private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {  
    String uName = txtUsrName.getText();  
    if (!uName.equals("")) {  
        String result = deleteEmployee(uName);  
        if (result.equalsIgnoreCase("Delete Succesful")) {  
            showMessageDialog(null, "Employee Successfully Deleted");  
            this.setVisible(false);  
            new EmployeePage().setVisible(true);  
        } else {  
            showMessageDialog(null, "Deletion Failed, Check username");  
        }  
    } else {  
        lblErrorMessage.setText("Please fill all the fields!!");  
    }  
}
```

```
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmployeePage().setVisible(true);  
}
```

```
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {  
    txtUsrName.setText("");  
    lblErrorMessage.setText("");  
}
```

```
private static String deleteEmployee(java.lang.String userName) {  
    login_ws.Login_Service service = new login_ws.Login_Service();  
    login_ws.Login port = service.getLoginPort();  
    return port.deleteEmployee(userName);  
}
```

Add Employee Page

```
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new EmployeePage().setVisible(true);  
}  
  
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {  
    txtName.setText("");  
    txtPWord.setText("");  
    txtUsrName.setText("");  
    txtPosition.setText("");  
    lblErrorMessage.setText("");  
}  
  
private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {  
    String name = txtName.getText();  
    String position = txtPosition.getText();  
    String uName = txtUsrName.getText();  
    String pWord = txtPWord.getText();  
  
    if (!uName.equals("") || !pWord.equals("") || !name.equals("") || !position.equals("")) {  
        String result = addEmployees(name, position, uName, pWord);  
        if (result.equalsIgnoreCase("Insert Successful")) {  
            showMessageDialog(null, "Registration Successful");  
            this.setVisible(false);  
            new LoginPage().setVisible(true);  
        } else {  
            showMessageDialog(null, "Registration Failed");  
        }  
    } else {  
        lblErrorMessage.setText("Please fill all the fields!!");  
    }  
}  
  
private static String addEmployees(java.lang.String name, java.lang.String position,  
    java.lang.String username, java.lang.String password) {  
    login_ws.Login_Service service = new login_ws.Login_Service();  
    login_ws.Login port = service.getLoginPort();  
    return port.addEmployees(name, position, username, password);  
}
```

Employee Login Page

```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  
    String uName = txtUserName.getText();  
    String pWord = txtPWord.getText();  
  
    if (loginEmployee(uName, pWord).equalsIgnoreCase("Login Success")) {  
        this.setVisible(false);  
        new Customer_List().setVisible(true);  
    } else {  
        showMessageDialog(null, "Incorrect Username/Password");  
    }  
}
```

```
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new Index().setVisible(true);  
}
```

```
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {  
    txtPWord.setText("");  
    txtUserName.setText("");  
    lblErrorMessage.setText("");  
}
```

```
private static String loginEmployee(java.lang.String userName, java.lang.String password) {  
    login_ws.Login_Service service = new login_ws.Login_Service();  
    login_ws.Login port = service.getLoginPort();  
    return port.loginEmployee(userName, password);  
}
```

Customer List Page

```
public Customer_List() {
    initComponents();
    getContentPane().setBackground(new Color(255, 255, 255));
    cust_list = showCustomers();
    displayCustomers();
}

public void displayCustomers() {
    custList.setLayout(new GridLayout(0, 3));
    JLabel lblName = new JLabel("Name");
    lblName.setForeground(Color.black);
    lblName.setFont(new Font("Roboto", Font.PLAIN, 18));
    lblName.setBorder(BorderFactory.createLineBorder(Color.white));
    JLabel lblAccNumber = new JLabel("Account Number");
    lblAccNumber.setForeground(Color.BLACK);
    lblAccNumber.setFont(new Font("Roboto", Font.PLAIN, 18));
    lblAccNumber.setBorder(BorderFactory.createLineBorder(Color.WHITE));
    JLabel lblManage = new JLabel("");
    lblManage.setBorder(BorderFactory.createLineBorder(Color.WHITE));
    custList.add(lblName);
    custList.add(lblAccNumber);
    custList.add(lblManage);
    JButton[] buttons = new JButton[cust_list.size()];
    JLabel[] nameLabels = new JLabel[cust_list.size()];
    JLabel[] accNumberLabels = new JLabel[cust_list.size()];
    for (int i = 0; i < cust_list.size(); i++) {
        buttons[i] = new JButton();
        nameLabels[i] = new JLabel(cust_list.get(i).getName());
        accNumberLabels[i] = new JLabel(cust_list.get(i).getAccountNumber());
        buttons[i].setText("Manage");
        buttons[i].setFont(new Font("Roboto", Font.PLAIN, 18));
        buttons[i].setBorder(BorderFactory.createLineBorder(Color.WHITE));
        nameLabels[i].setHorizontalAlignment(SwingConstants.CENTER);
        accNumberLabels[i].setHorizontalAlignment(SwingConstants.CENTER);
        Customers tempCust = cust_list.get(i);
        buttons[i].addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                new CustomerAccount(tempCust).setVisible(true);
            }
        });

        nameLabels[i].setForeground(Color.BLACK);
        nameLabels[i].setFont(new Font("Roboto", Font.PLAIN, 18));
        nameLabels[i].setBorder(BorderFactory.createLineBorder(Color.BLACK));
        accNumberLabels[i].setForeground(Color.BLACK);
        accNumberLabels[i].setFont(new Font("Roboto", Font.PLAIN, 18));
        accNumberLabels[i].setBorder(BorderFactory.createLineBorder(Color.BLACK));
        custList.add(nameLabels[i]);
        custList.add(accNumberLabels[i]);
        custList.add(buttons[i]);
    }
    add(custList);
    custList.revalidate();
    custList.repaint();
}
```

```
private void btnCreateCustActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new CustomerAccount().setVisible(true);  
}
```

```
private void btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
    this.setVisible(false);  
    new Index().setVisible(true);  
}
```

```
private static java.util.List<customer_ws.Customers> showCustomers() {  
    customer_ws.CustomerWs_Service service = new customer_ws.CustomerWs_Service();  
    customer_ws.CustomerWs port = service.getCustomerWsPort();  
    return port.showCustomers();  
}
```


Customer Account Page

```
public CustomerAccount() {
    initComponents();
    lblHeading.setText("Create New Customer");
    btnEdit.setVisible(false);
    btnDelete.setVisible(false);
}

CustomerAccount(Customers cust) {
    initComponents();
    lblHeading.setText("Manage Customer");
    btnAdd.setVisible(false);
    btnClear.setVisible(false);
    txtName.setText(cust.getName());
    txtDob.setText(cust.getBirtDate());
    txtAddress.setText(cust.getAddress());
    txtMobNo.setText(cust.getMobile());
    txtEmail.setText(cust.getEmail());
    txtAccType.setText(cust.getAccountType());
    txtAccNo.setText(cust.getAccountNumber());
    txtSortCode.setText(cust.getSortCode());
    txtBalance.setText(cust.getBalance());
    txtCard.setText(cust.getCard());

    txtAccNo.setEditable(false);
    txtAccType.setEditable(false);
    txtSortCode.setEditable(false);
    txtBalance.setEditable(false);
    txtCard.setEditable(false);
}

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    String name = txtName.getText();
    String dob = txtDob.getText();
    String address = txtAddress.getText();
    String mobileNo = txtMobNo.getText();
    String email = txtEmail.getText();
    String accType = txtAccType.getText();
    String accNo = txtAccNo.getText();
    String sortCode = txtSortCode.getText();
    String balance = txtBalance.getText();
    String card = txtCard.getText();

    if (!name.equals("") || !dob.equals("") || !address.equals("") || !mobileNo.equals("") || !email.equals("")
        || !accType.equals("") || !accNo.equals("") || !sortCode.equals("") || !balance.equals("") || !card.equals("")) {
        String result = addCustomer(name, address, mobileNo, dob, email, accType, accNo, balance, sortCode, card);
        if (result.equalsIgnoreCase("Insert Succesful")) {
            showMessageDialog(null, "Customer Successfully Registered");
            this.setVisible(false);
            new Customer_List().setVisible(true);
        } else {
            showMessageDialog(null, "Insertion Failed");
        }
    } else {
        lblErrorMessage.setText("Please fill all the fields!!");
    }
}
```

```

private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    new Customer_List().setVisible(true);
}

private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
    txtName.setText("");
    txtDob.setText("");
    txtAddress.setText("");
    txtMobNo.setText("");
    txtEmail.setText("");
    txtAccType.setText("");
    txtAccNo.setText("");
    txtSortCode.setText("");
    txtBalance.setText("");
    txtCard.setText("");
    lblErrorMessage.setText("");
}

private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {
    String name = txtName.getText();
    String dob = txtDob.getText();
    String address = txtAddress.getText();
    String mobileNo = txtMobNo.getText();
    String email = txtEmail.getText();
    String accNo = txtAccNo.getText();
    String result = editCustomer(name, address, dob, mobileNo, email, accNo);
    if (result.equalsIgnoreCase("Update Successful")) {
        showMessageDialog(null, "Customer Successfully Edited");
        this.setVisible(false);
        new Customer_List().setVisible(true);
    } else {
        showMessageDialog(null, "Editing Failed");
    }
}

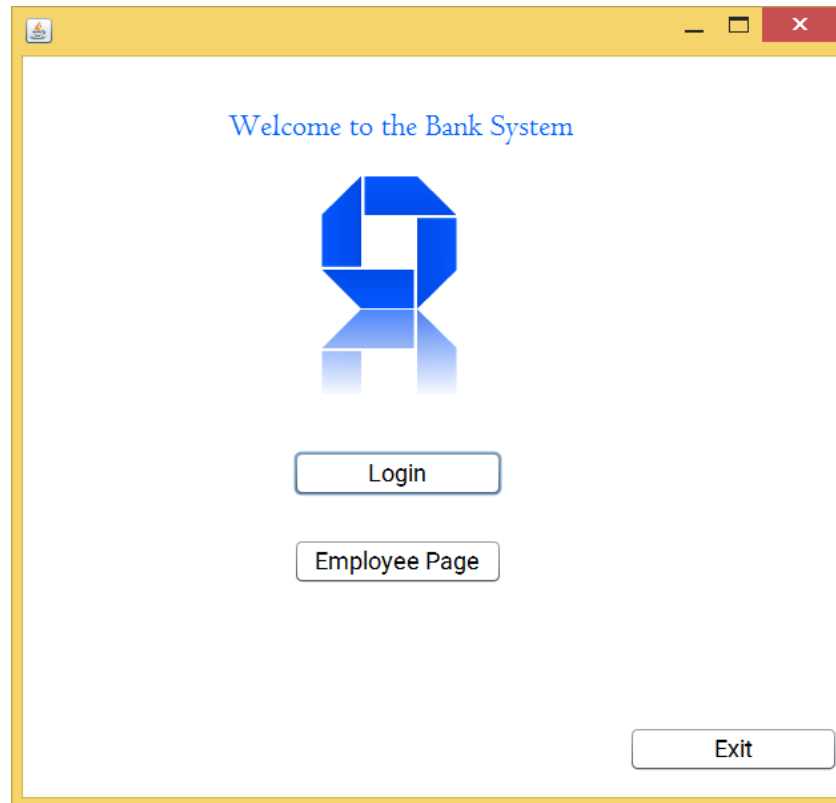
private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    String accNo = txtAccNo.getText();
    System.out.println(accNo);
    String result = deleteCustomer(accNo);
    if (result.equalsIgnoreCase("Delete Successful")) {
        showMessageDialog(null, "Customer Successfully Deleted");
        this.setVisible(false);
        new Customer_List().setVisible(true);
    } else {
        showMessageDialog(null, "Deletion Failed, Check the old username");
    }
}

```

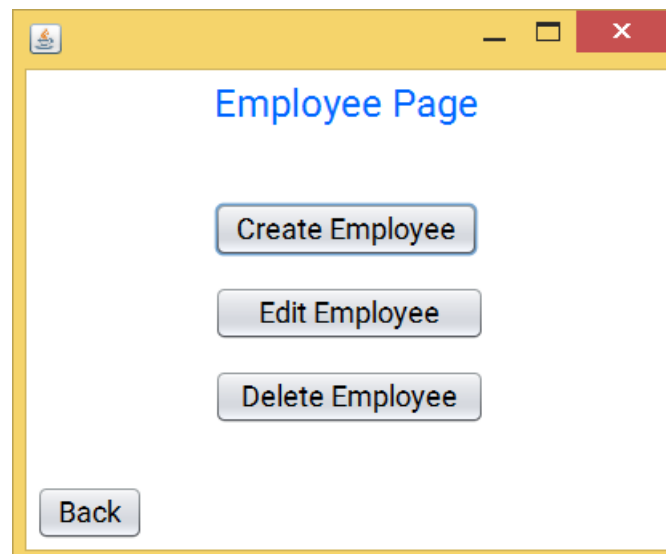
```
private static String addCustomer(java.lang.String name, java.lang.String address,  
    java.lang.String mobile, java.lang.String dateOfBirth, java.lang.String email,  
    java.lang.String accountType, java.lang.String accountNumber, java.lang.String balance,  
    java.lang.String sortCode, java.lang.String card) {  
    customer_ws.CustomerWs_Service service = new customer_ws.CustomerWs_Service();  
    customer_ws.CustomerWs port = service.getCustomerWsPort();  
    return port.addCustomer(name, address, mobile, dateOfBirth, email, accountType, accountNumber,  
        balance, sortCode, card);  
}  
  
private static String editCustomer(java.lang.String name, java.lang.String address,  
    java.lang.String dateOfBirth, java.lang.String mobile, java.lang.String email,  
    java.lang.String accountNumber) {  
    customer_ws.CustomerWs_Service service = new customer_ws.CustomerWs_Service();  
    customer_ws.CustomerWs port = service.getCustomerWsPort();  
    return port.editCustomer(name, address, dateOfBirth, mobile, email, accountNumber);  
}  
  
private static String deleteCustomer(java.lang.String accountNumber) {  
    customer_ws.CustomerWs_Service service = new customer_ws.CustomerWs_Service();  
    customer_ws.CustomerWs port = service.getCustomerWsPort();  
    return port.deleteCustomer(accountNumber);  
}
```

GUI Screenshots

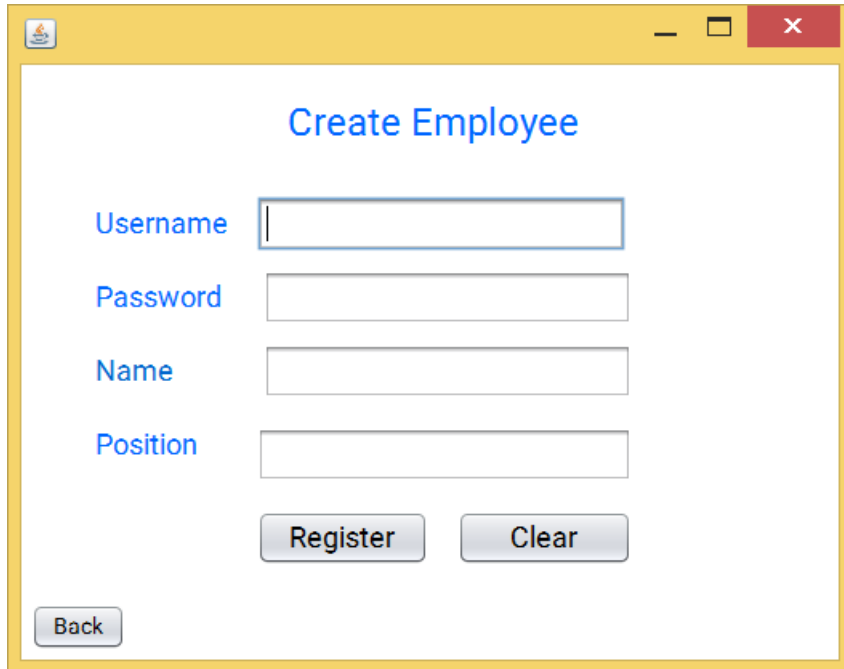
Index Page



Employee Page



Create Employee Page



A screenshot of a web application window titled "Create Employee". The window has a yellow border and standard Windows window controls (minimize, maximize, close) in the top right corner. The title "Create Employee" is centered at the top in blue. Below the title, there are four input fields with labels to their left: "Username", "Password", "Name", and "Position". Each label and its corresponding input field are highlighted with a blue border. At the bottom of the form, there are three buttons: "Register" and "Clear" are side-by-side, and "Back" is positioned below the "Register" button.

Create Employee

Username

Password

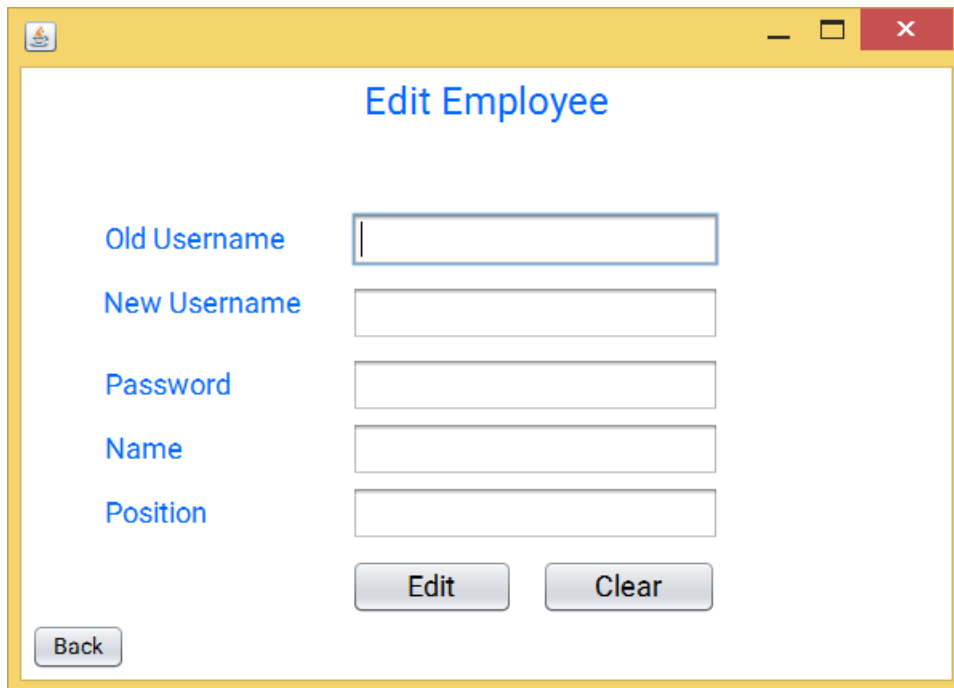
Name

Position

Register Clear

Back

Edit Employee Page



A screenshot of a web application window titled "Edit Employee". The window has a yellow border and standard Windows window controls (minimize, maximize, close) in the top right corner. The title "Edit Employee" is centered at the top in blue. Below the title, there are five input fields with labels to their left: "Old Username", "New Username", "Password", "Name", and "Position". Each label and its corresponding input field are highlighted with a blue border. At the bottom of the form, there are two buttons: "Edit" and "Clear". A "Back" button is located in the bottom left corner.

Edit Employee

Old Username

New Username

Password

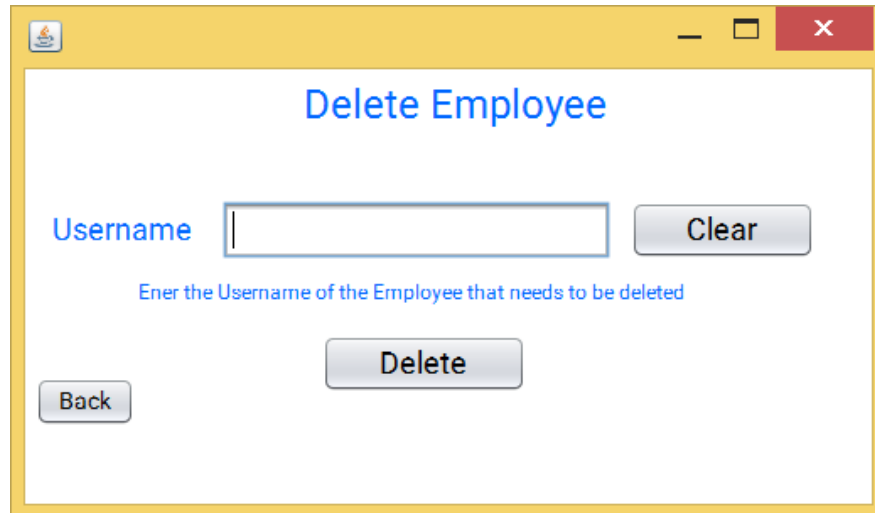
Name

Position

Edit Clear

Back

Delete Employee Page



A screenshot of a web application window titled "Delete Employee". The window has a yellow border and standard Windows window controls (minimize, maximize, close) in the top right corner. The main content area has a white background. At the top, the title "Delete Employee" is displayed in blue. Below the title, there is a label "Username" in blue, followed by a text input field. To the right of the input field is a "Clear" button. Below the input field, a blue instruction text reads "Enter the Username of the Employee that needs to be deleted". At the bottom left is a "Back" button, and at the bottom center is a "Delete" button.

Delete Employee

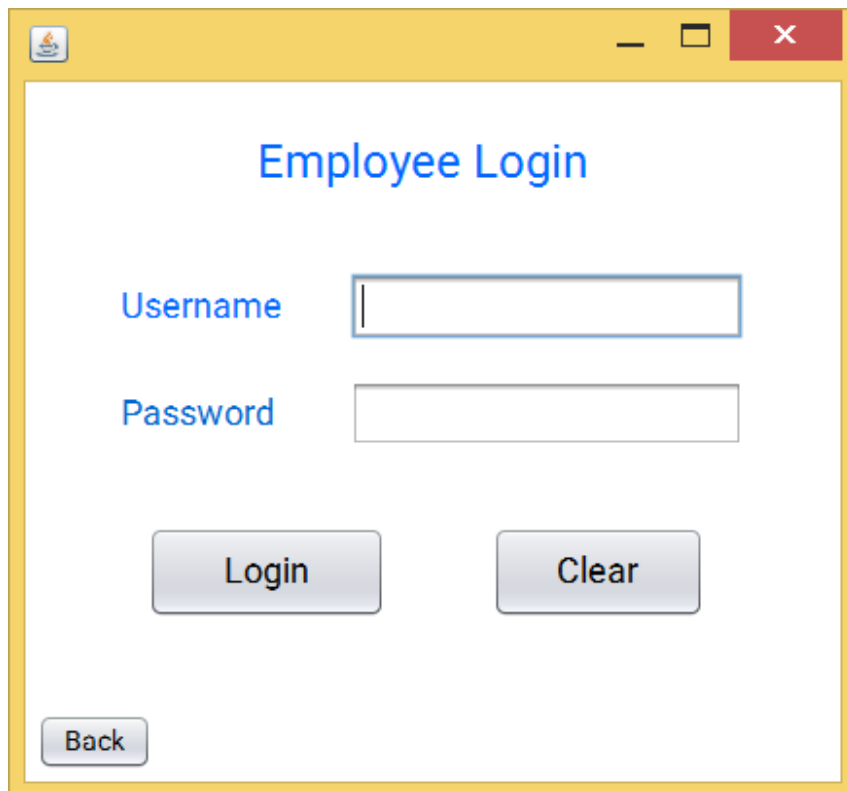
Username

Clear

Enter the Username of the Employee that needs to be deleted

Back Delete

Login Page



A screenshot of a web application window titled "Employee Login". The window has a yellow border and standard Windows window controls (minimize, maximize, close) in the top right corner. The main content area has a white background. At the top, the title "Employee Login" is displayed in blue. Below the title, there are two labels in blue: "Username" and "Password". Each label is followed by a text input field. Below the input fields, there are two buttons: "Login" and "Clear". At the bottom left is a "Back" button.

Employee Login

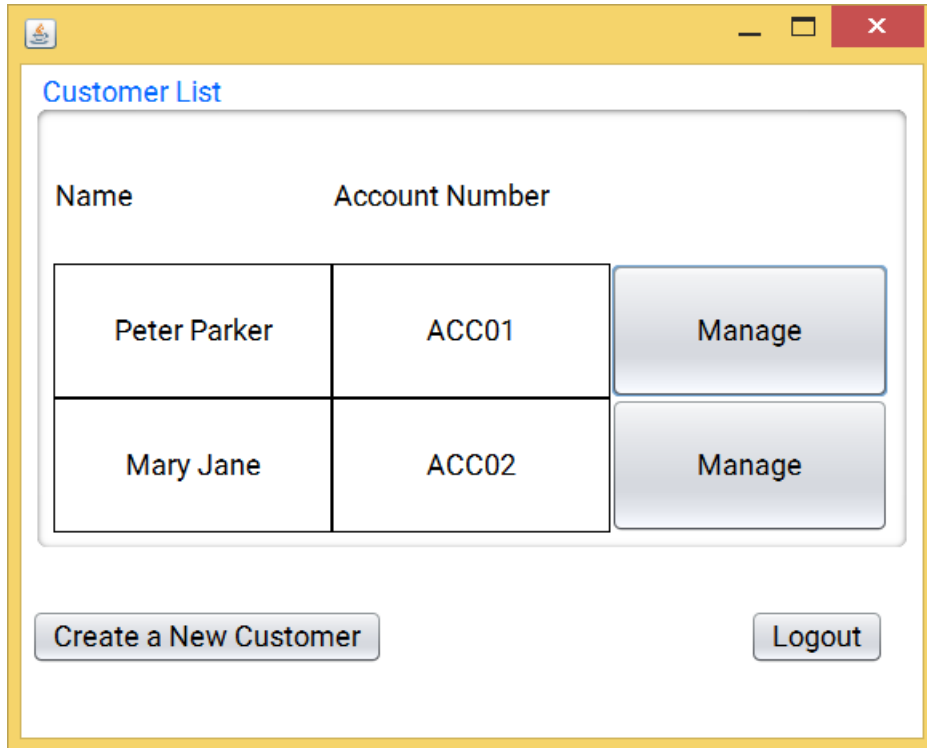
Username

Password

Login Clear

Back

Customer List Page

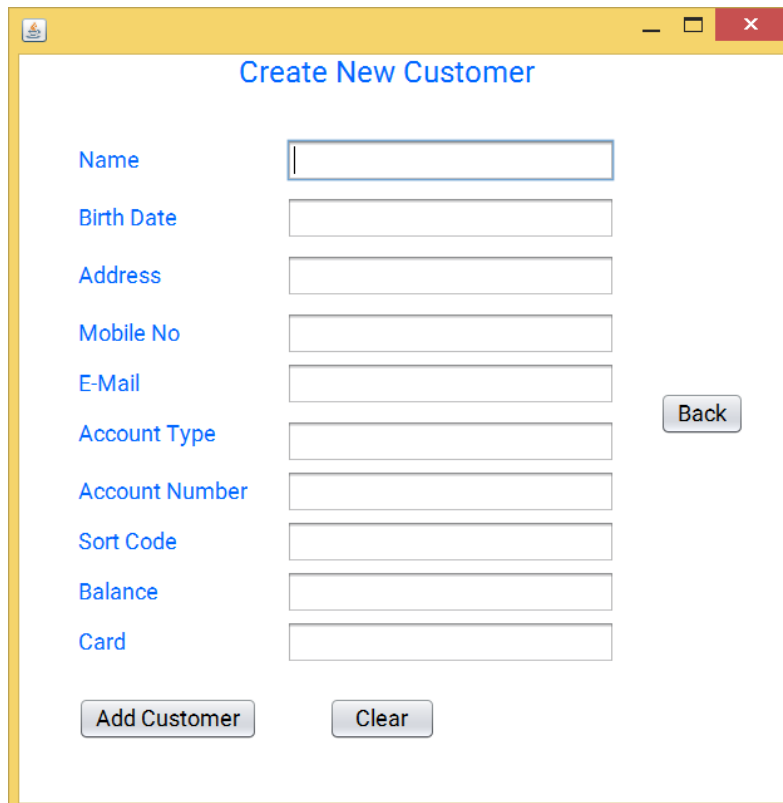


A screenshot of a web application window titled "Customer List". The window has a yellow title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top, the title "Customer List" is displayed in blue. Below the title, there are two columns: "Name" and "Account Number". Under these columns, there is a table with two rows. The first row contains "Peter Parker" and "ACC01". The second row contains "Mary Jane" and "ACC02". To the right of each row is a "Manage" button. At the bottom of the window, there are two buttons: "Create a New Customer" and "Logout".

Name	Account Number	
Peter Parker	ACC01	Manage
Mary Jane	ACC02	Manage

Create a New Customer Logout

Create Customer Page



A screenshot of a web application window titled "Create New Customer". The window has a yellow title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top, the title "Create New Customer" is displayed in blue. Below the title, there are several input fields for customer information: Name, Birth Date, Address, Mobile No, E-Mail, Account Type, Account Number, Sort Code, Balance, and Card. To the right of the input fields, there is a "Back" button. At the bottom of the window, there are two buttons: "Add Customer" and "Clear".

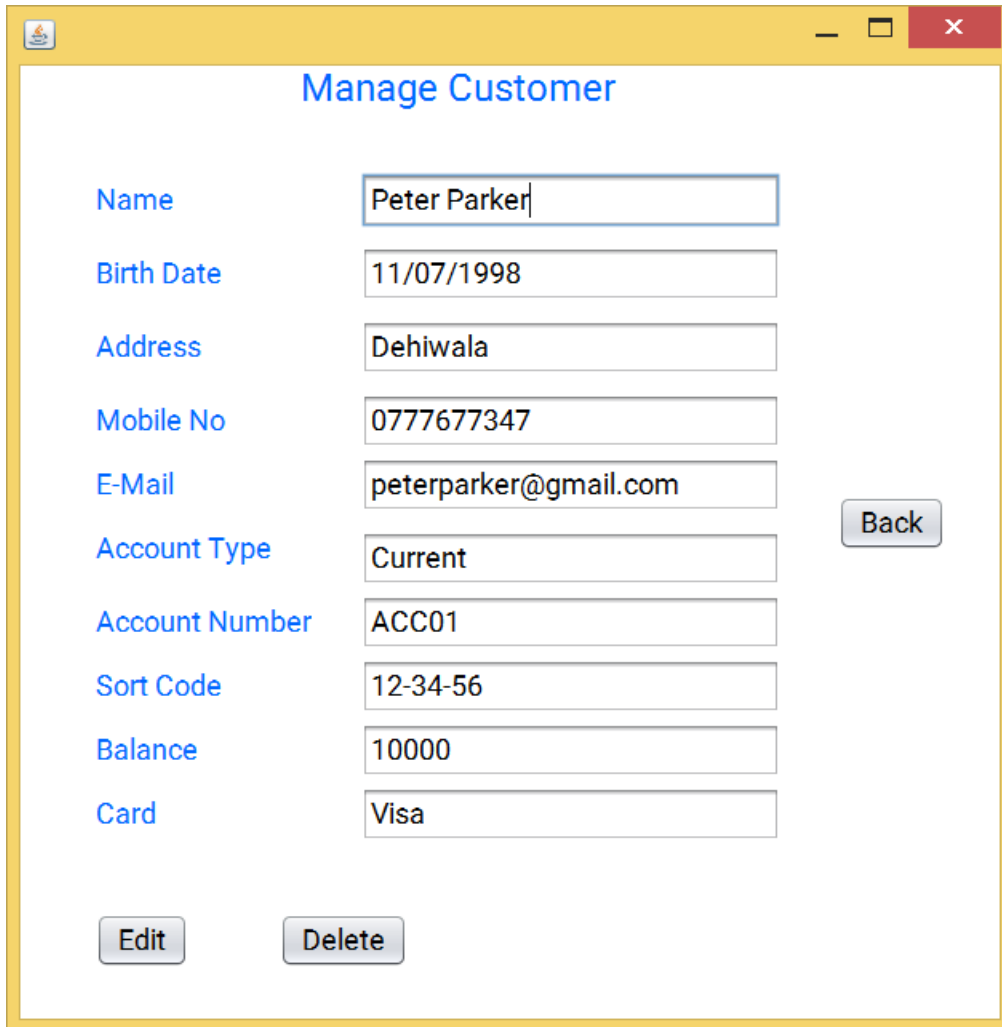
Create New Customer

Name Birth Date Address Mobile No E-Mail Account Type Account Number Sort Code Balance Card

Back

Add Customer Clear

Edit and Delete Customer Page



A screenshot of a web application window titled "Manage Customer". The window has a yellow border and standard window controls (minimize, maximize, close) in the top right corner. The title "Manage Customer" is centered at the top in blue. Below the title, there is a form with several input fields, each with a label to its left. The labels are in blue text. The input fields contain the following data: Name (Peter Parker), Birth Date (11/07/1998), Address (Dehiwala), Mobile No (0777677347), E-Mail (peterparker@gmail.com), Account Type (Current), Account Number (ACC01), Sort Code (12-34-56), Balance (10000), and Card (Visa). To the right of the input fields, there is a "Back" button. At the bottom left of the form, there are two buttons: "Edit" and "Delete".

Name	Peter Parker
Birth Date	11/07/1998
Address	Dehiwala
Mobile No	0777677347
E-Mail	peterparker@gmail.com
Account Type	Current
Account Number	ACC01
Sort Code	12-34-56
Balance	10000
Card	Visa

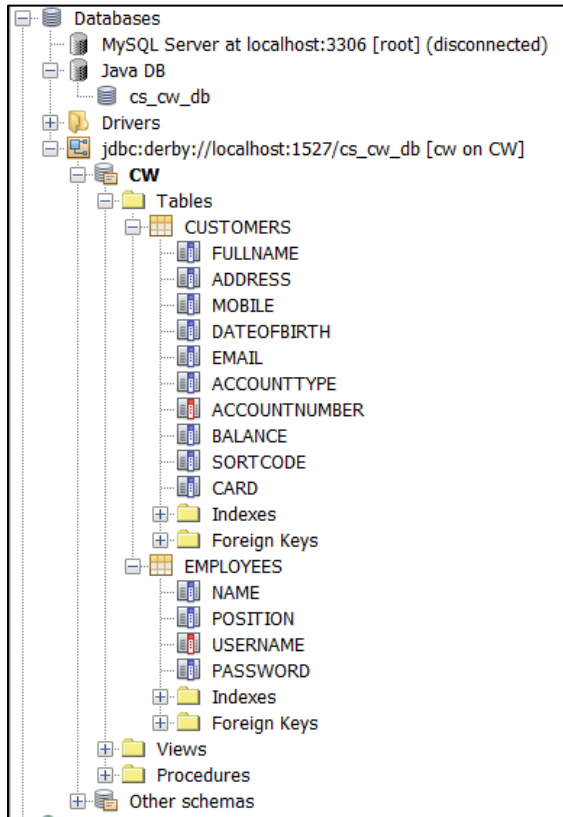
Back

Edit Delete

The above page is presented once the employee clicks on the “Manage” button beside the relevant customer.

Database Screenshot

We used the NetBeans Derby Database to store the Customer and Employee Data.



Customer Data

SELECT * FROM CW.CUSTOMER... ×										
Max. rows: 100 Fetched Rows: 2										
#	FULLNAME	ADDRESS	MOBILE	DATEOFBIRTH	EMAIL	ACCOUNTTYPE	ACCOUNTNUMBER	BALANCE	SORTCODE	CARD
1	Peter Parker	Dehiwala	0777677347	11/07/1998	peterparker@gmail.com	Current	ACC01	10000	12-34-56	Visa
2	Mary Jane	Harmers Avenue	0767450170	08/05/1996	maryjane@gmail.com	Current	ACC02	99999999	12-34-56	Visa

Employee Data

SELECT * FROM CW.EMPLOYEE... ×				
Max. rows: 100 Fetched Rows: 2				
#	NAME	POSITION	USERNAME	PASSWORD
1	John Doe	CEO	admin	123
2	Jane Doe	CFO	admin2	456

Work Load Matrix

Group Members

Name	IIT ID	UoW ID
Keshav Kumaresan	2014016	W1582991
Manuka Maduranga	2015313	W1608496
Sudam Dissanayake	2015096	W1582963
Khopithan Sathiyakeerthy	2015245	W1608467
Sonal Muthukumarana	2015347	W1608505

Tasks

Task 1 – Designing the Use Case Diagram

Task 2 – Designing the Sequence Diagrams

Task 3 – Designing the Graphical User Interface (GUI)

Task 4 – Designing and Implementing the Web Services

Task 5 –Integrate the Web Service with the Web client

Task done by each member

Name	Task 1	Task 2	Task 3	Task 4	Task 5
Keshav Kumaresan					✓
Manuka Maduranga				✓	
Sudam Dissanayake	✓				
Khopithan Sathiyakeerthy			✓		
Sonal Muthukumarana		✓			