Informatics Institute of Technology

Department of Computing

(B.Eng.) in Software Engineering

Module: 4COSC010C.2 Programming Principles 02

Module Leader: Mr. Guhanathan Poravi

Assignment 02

Date of Submission	: 8th April 2019
Student ID	: 2018194
Student UoW ID	: w1714943
Student First Name	: Galgamuge
Student Surname	: Fernando

Introduction

This is about a "computer consultancy firm" software which the manager in that company can use to manage the records of its Customers, Contracts, Employees and Roles.

This software is for the use of the manager and only the manager has its login details, and he is the one who is authorized to do any changes to the records.

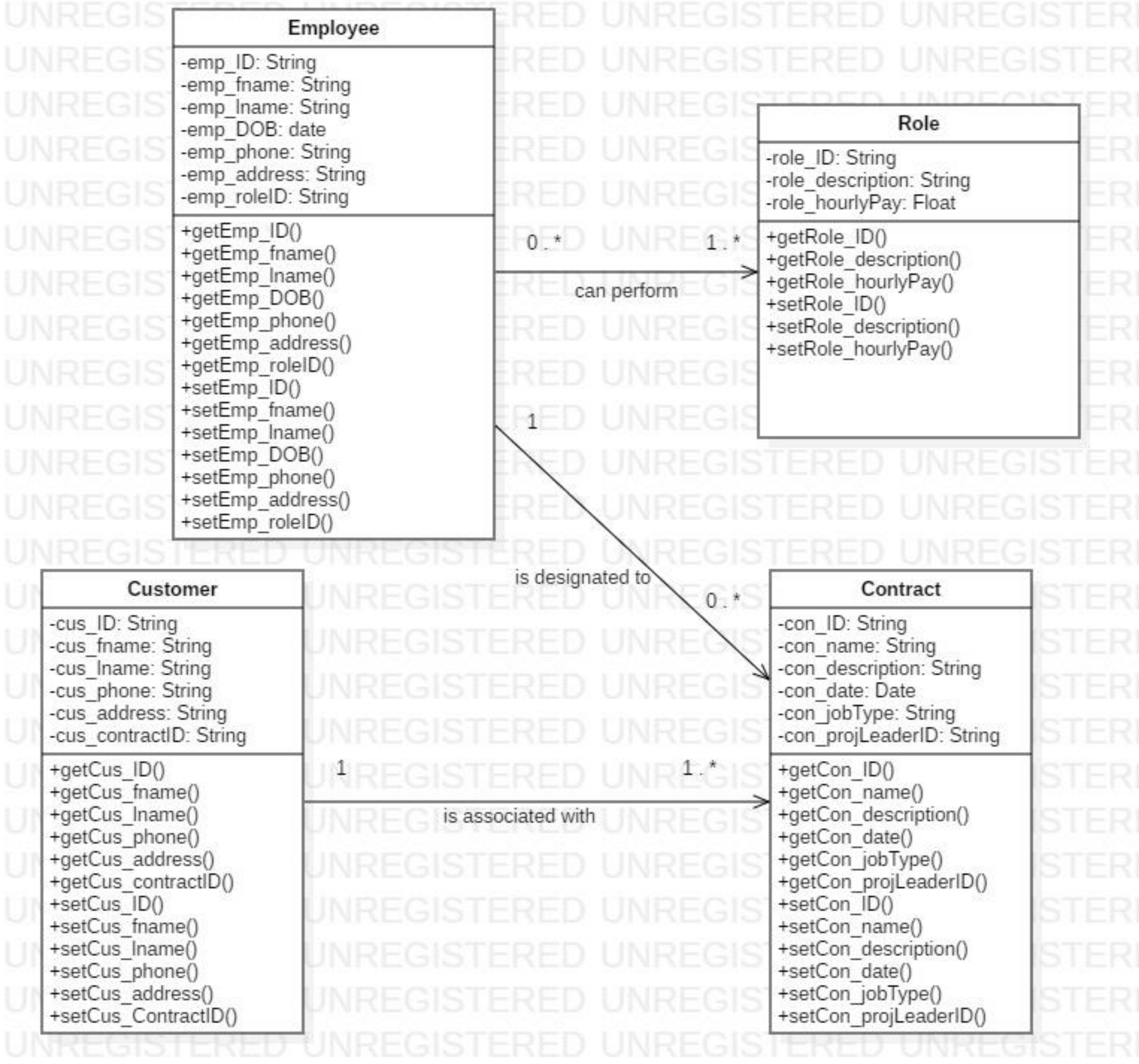
Analysis

1. Functional Requirements
    a. Add Customers, Contracts and Employees (I assumed roles cannot be added as the company has already distinguished the roles)
    b. View Customers, Contracts, Employees and Roles.
    c. Update Customers, Contracts, Employees and Roles.
    d. Delete Customers, Contracts and Employees. (I assumed the present roles are permanent and is highly needed to run the company so cannot be deleted)
2. Non-Functional Requirements
    a. Ability to change login Credentials

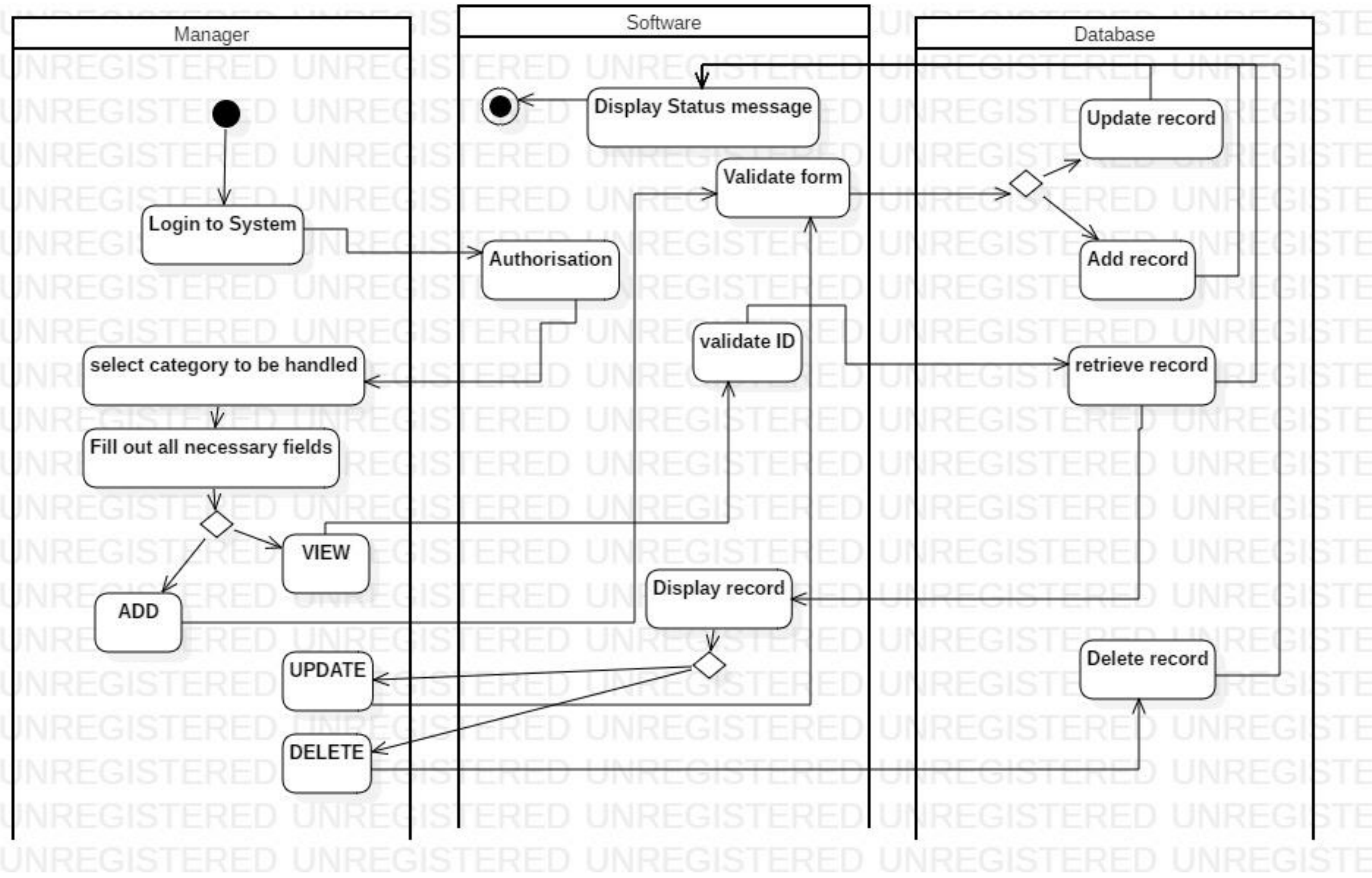
Design

1. Class Diagram
2. Activity Diagram
3. Use Case Diagram + Description

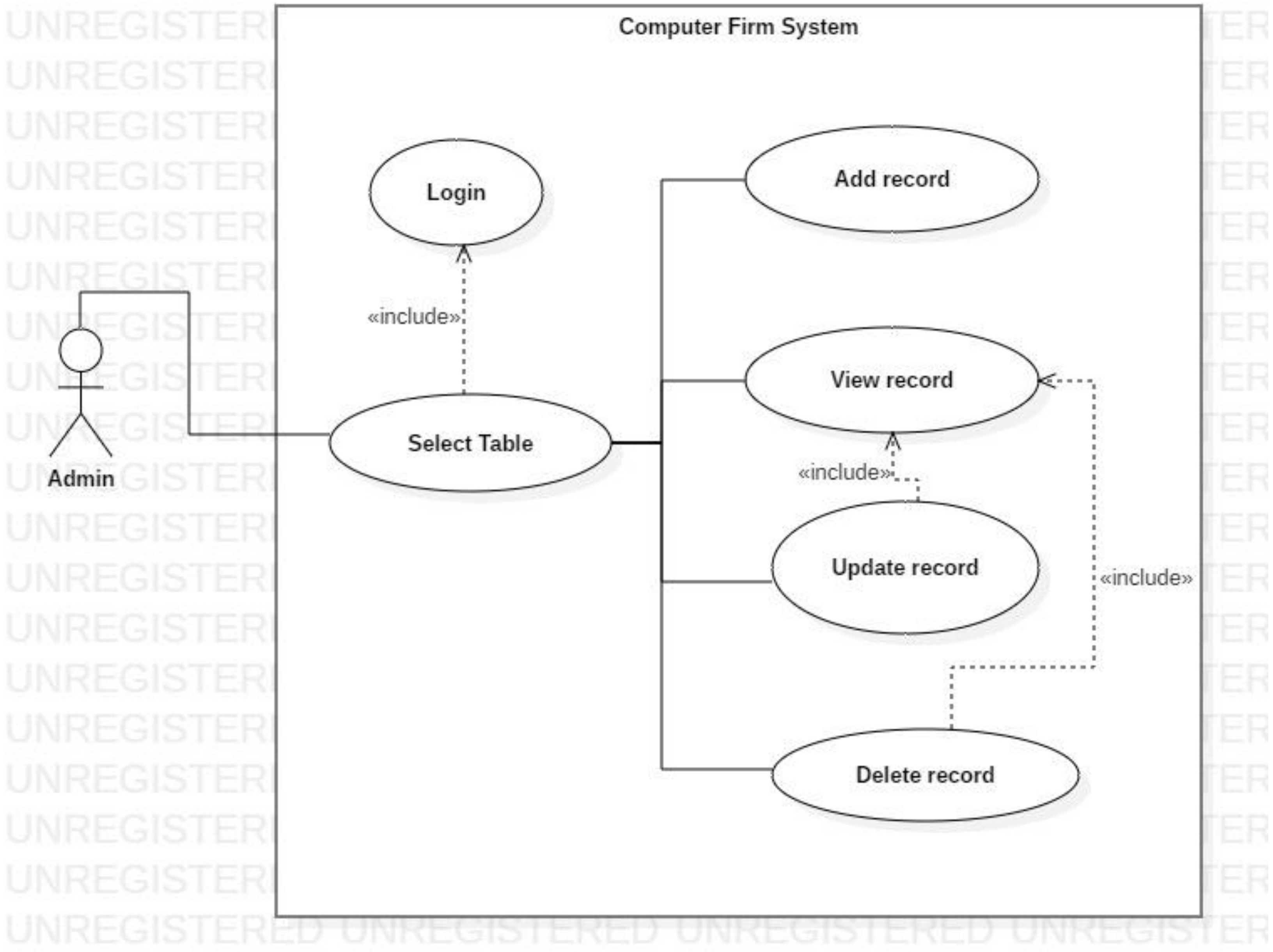
- Class Diagram



- Activity Diagram



- Use Case Diagram



- Use Case Descriptions

Use case ID	ID001
Use Case Name	Login
Use Case Description	The manager authorizes himself/herself in order to have access into the system
Actors	-
Pre-Conditions	-
Post Conditions	User will be directed to a welcome page and then the user can select which section (customer, contract, employee or role) he/she wants to interact with.
Path	
Primary Path	Enter user name Enter password If username and password match the user has access to the system
Alternate Path	-
Exception Path	-
Assumptions	The user cannot login to the system without proper Authorization

Use case ID	ID002
Use Case Name	Select Table
Use Case Description	The user can select which section he wants to interact with (customer, contract, employee or role) by clicking on the button, related to a section, on the menu bar.
Actors	Manager
Pre-Conditions	Login into the System
Post Conditions	The user can select which section he wants to interact with.
Path	
Primary Path	Click on the buttons to go into each of the 4 sections one by one in any order
Alternate Path	-
Exception Path	-
Assumptions	The user has independent selection to which section he could access.

Use case ID	ID003
Use Case Name	Add Record
Use Case Description	Each section (except role), has the ability of adding new records to its relevant entity
Actors	
Pre-Conditions	The user should be logged in The user should select a section from customer, employee or role
Post Conditions	The user can add, delete, view or update any other records The user can switch between each section
Path	
Primary Path	Fill out all the fields and with valid ID's Click "ADD"
Alternate Path	-
Exception Path	Keep a field empty and try clicking "ADD" Fill all fields but with invalid ID format Fill all fields but with an already existing ID
Assumptions	The data is validated before it is added to the database

Use case ID	ID004
Use Case Name	View Record
Use Case Description	The user can view and existing record in the database
Actors	-
Pre-Conditions	User should be logged in The user should select a section from customer, contract, employee or role.
Post Conditions	The user can add, delete, view or update any other records The user can switch between each section
Path	
Primary Path	Enter a record ID related to that particular section Click "VIEW"
Alternate Path	-
Exception Path	Enter an invalid ID and click "VIEW" Keep the ID field empty and click "VIEW"
Assumptions	The records are searched by its unique ID

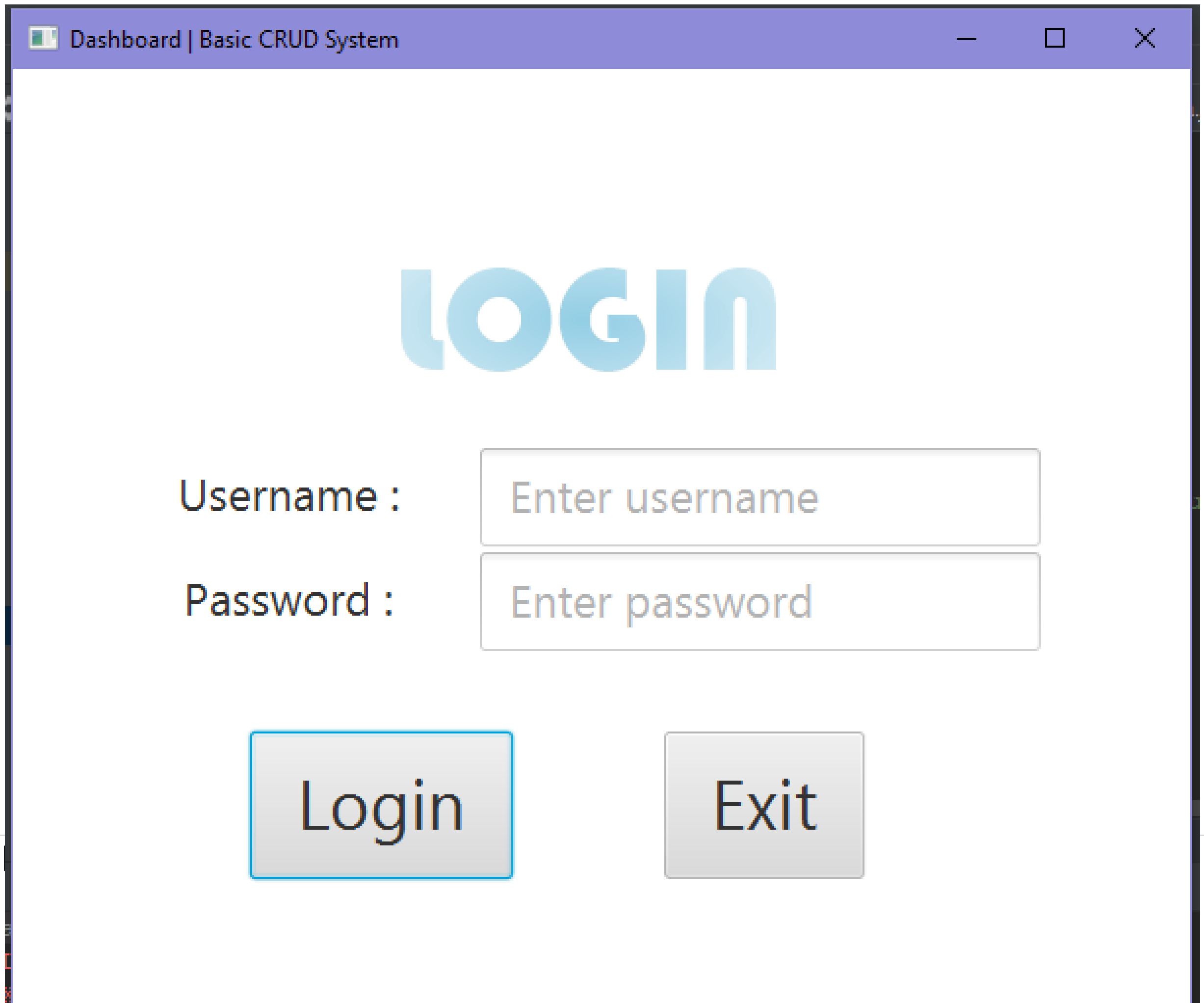
Use case ID	ID005
Use Case Name	Update Record
Use Case Description	The user can update an existing record by searching it by its ID
Actors	-
Pre-Conditions	User has to be logged in Choose which section is to be updated Choose which record needs to be updated
Post Conditions	The user can add, delete, view or update any other records The user can switch between each section
Path	
Primary Path	Enter a record ID View the record Do necessary changes to fields Click “Update”
Alternate Path	-
Exception Path	You cannot update a record without viewing it first
Assumptions	The records must be viewed in order to be updated

Use case ID	ID006
Use Case Name	Delete Record
Use Case Description	The user can delete an existing record by referring it through its ID
Actors	-
Pre-Conditions	User has to be logged in Choose which section is to be interacted by the user Choose which record needs to be deleted
Post Conditions	The user can add, delete, view or update any other records The user can switch between each section
Path	
Primary Path	Enter a record ID Click “delete”
Alternate Path	If you want to make sure you are deleting the correct record you can view the record before deleting, after verifying you can delete it.
Exception Path	you cannot delete a record without its ID
Assumptions	You can delete a record by only entering its ID

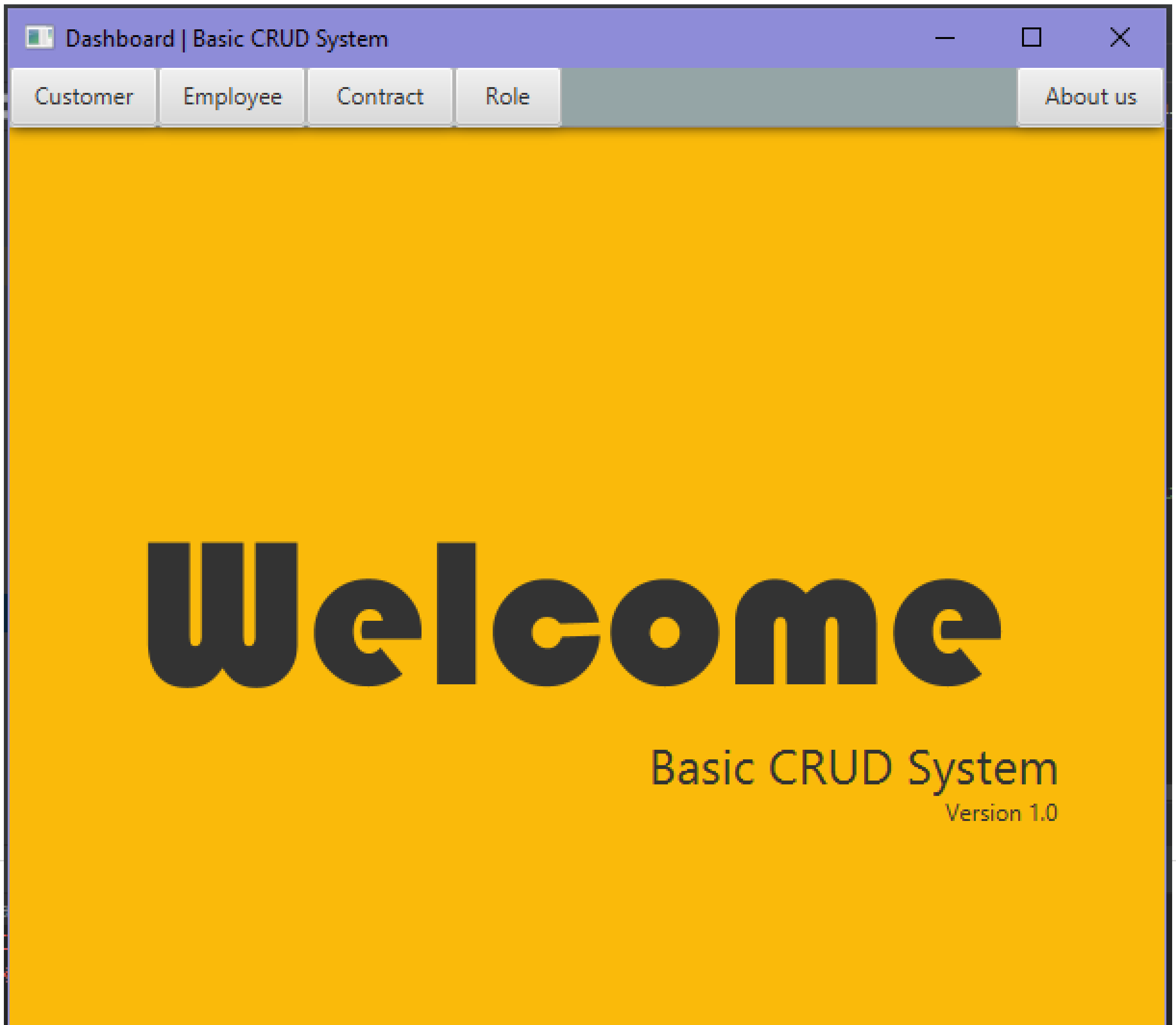
The UI of the Program

The UI has 8 sections in total,

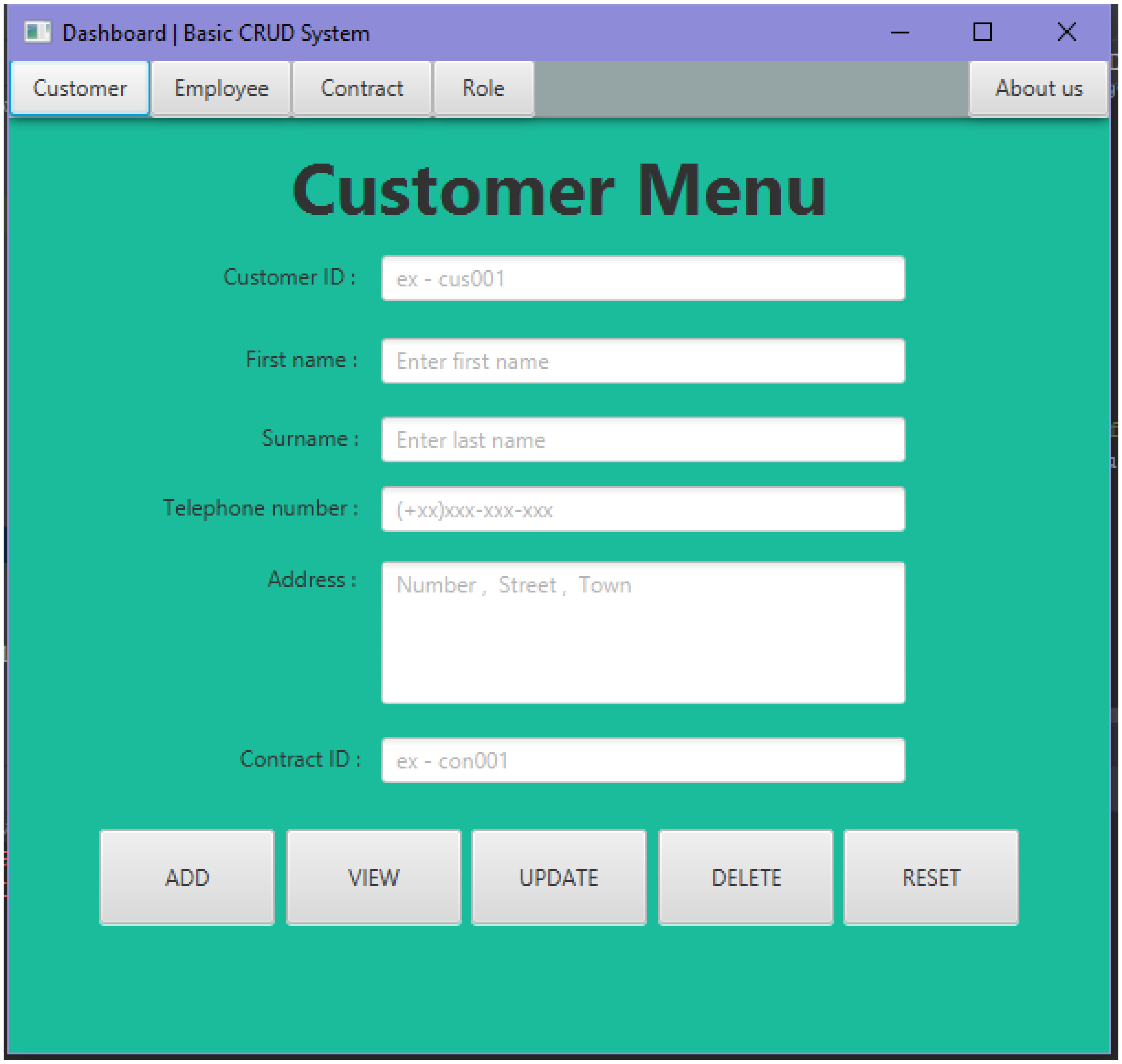
1. **Login section**



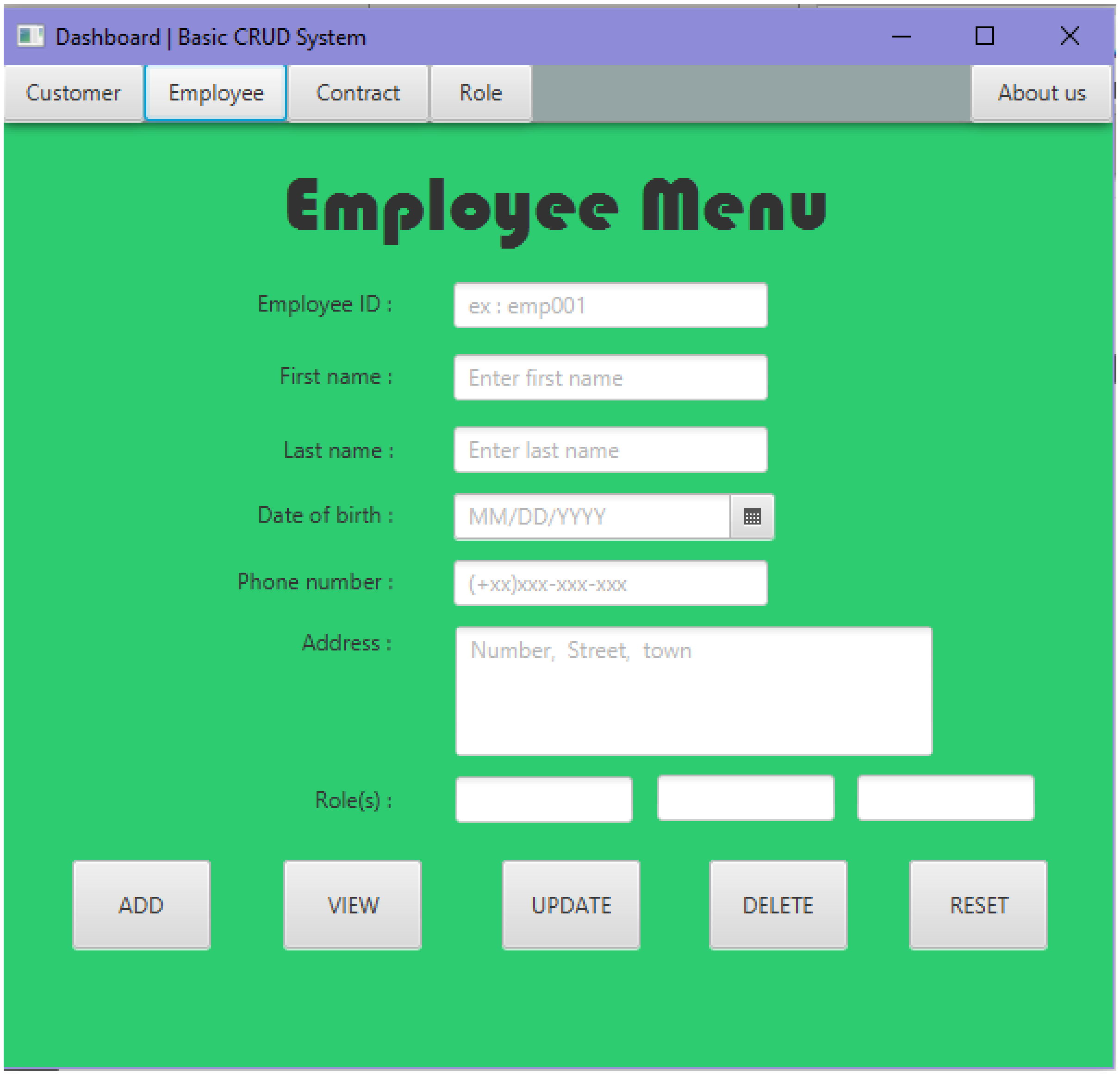
2. Welcome Screen Section



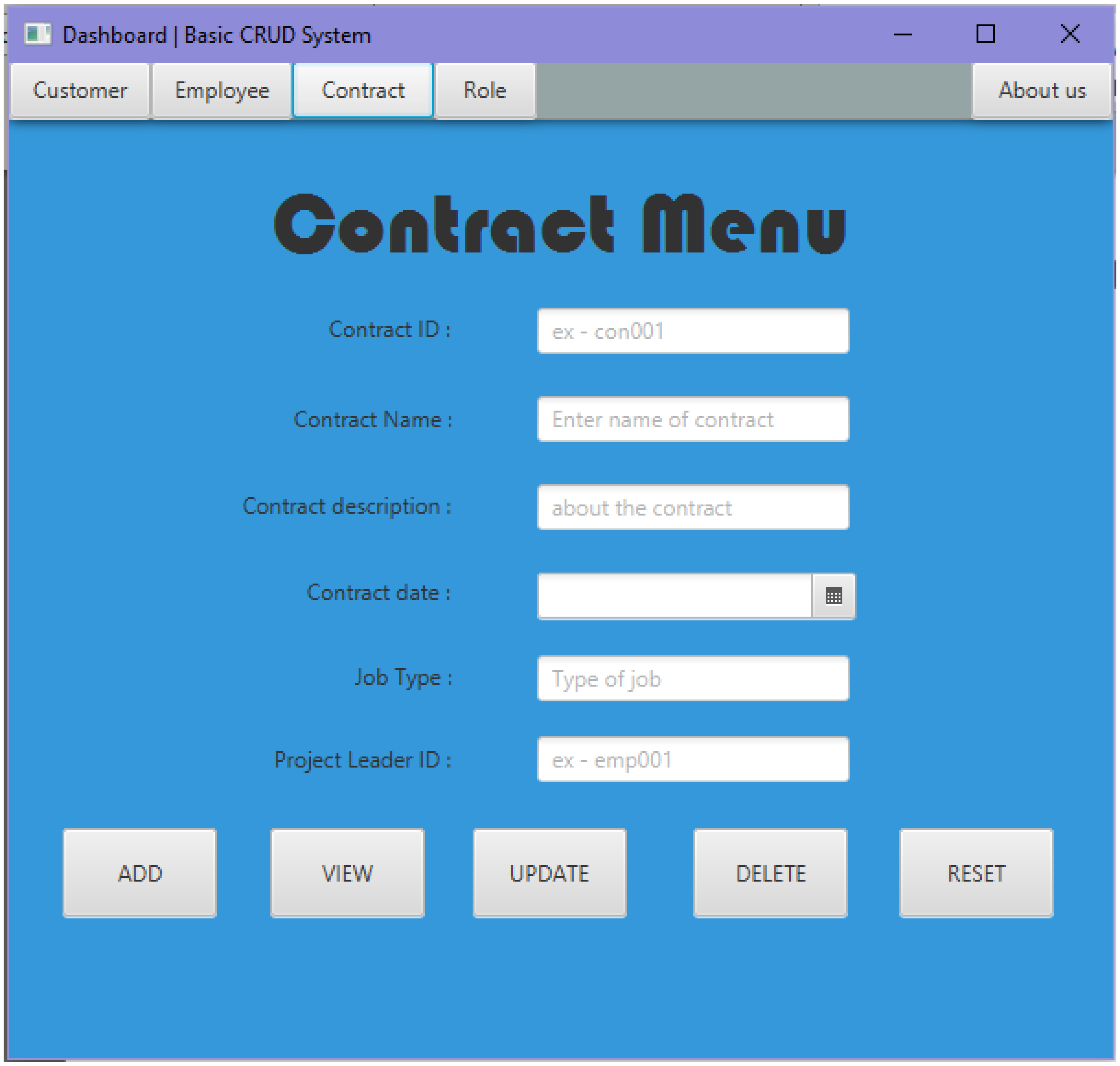
3. Customer Section



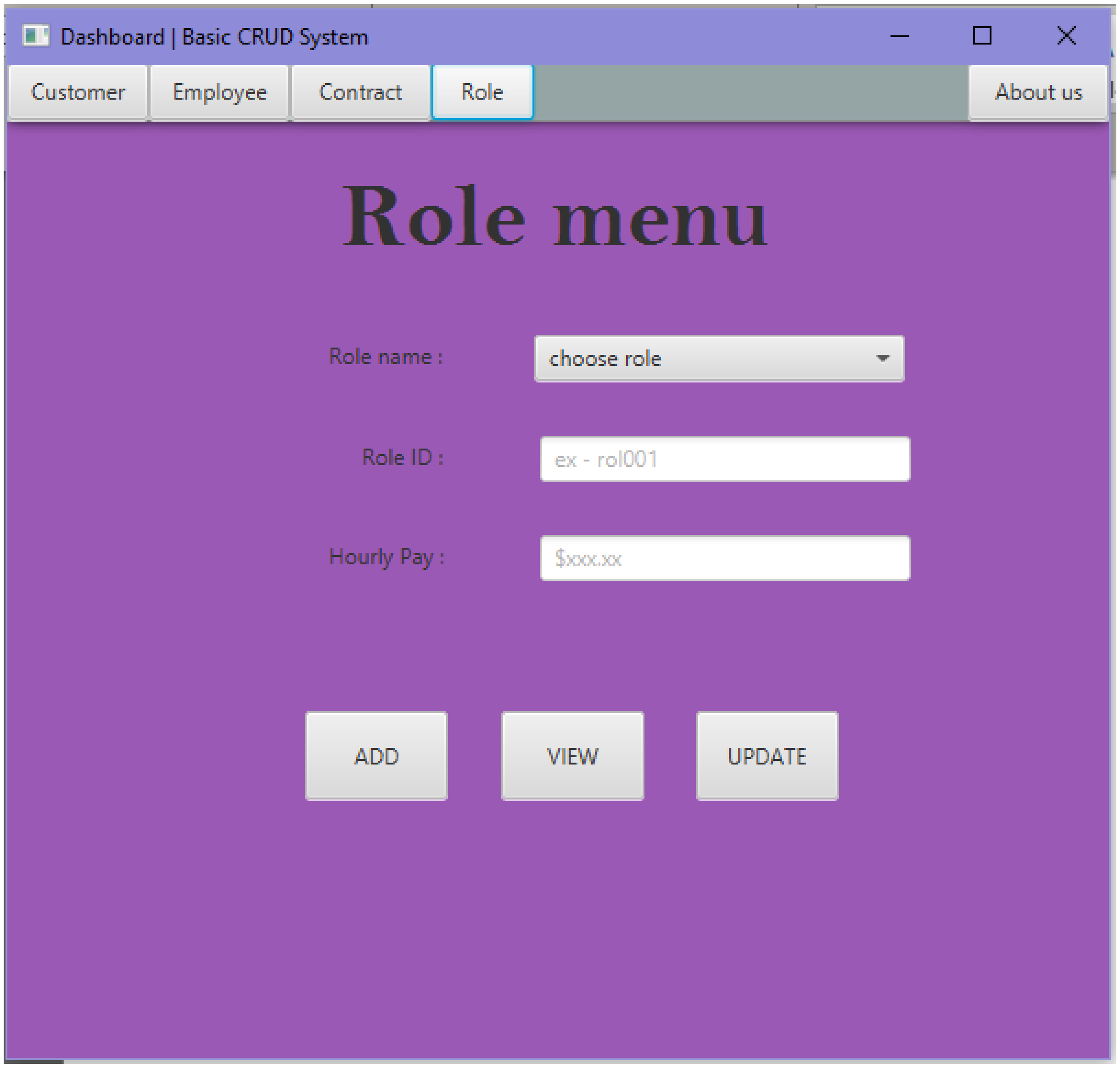
4. Employee Section



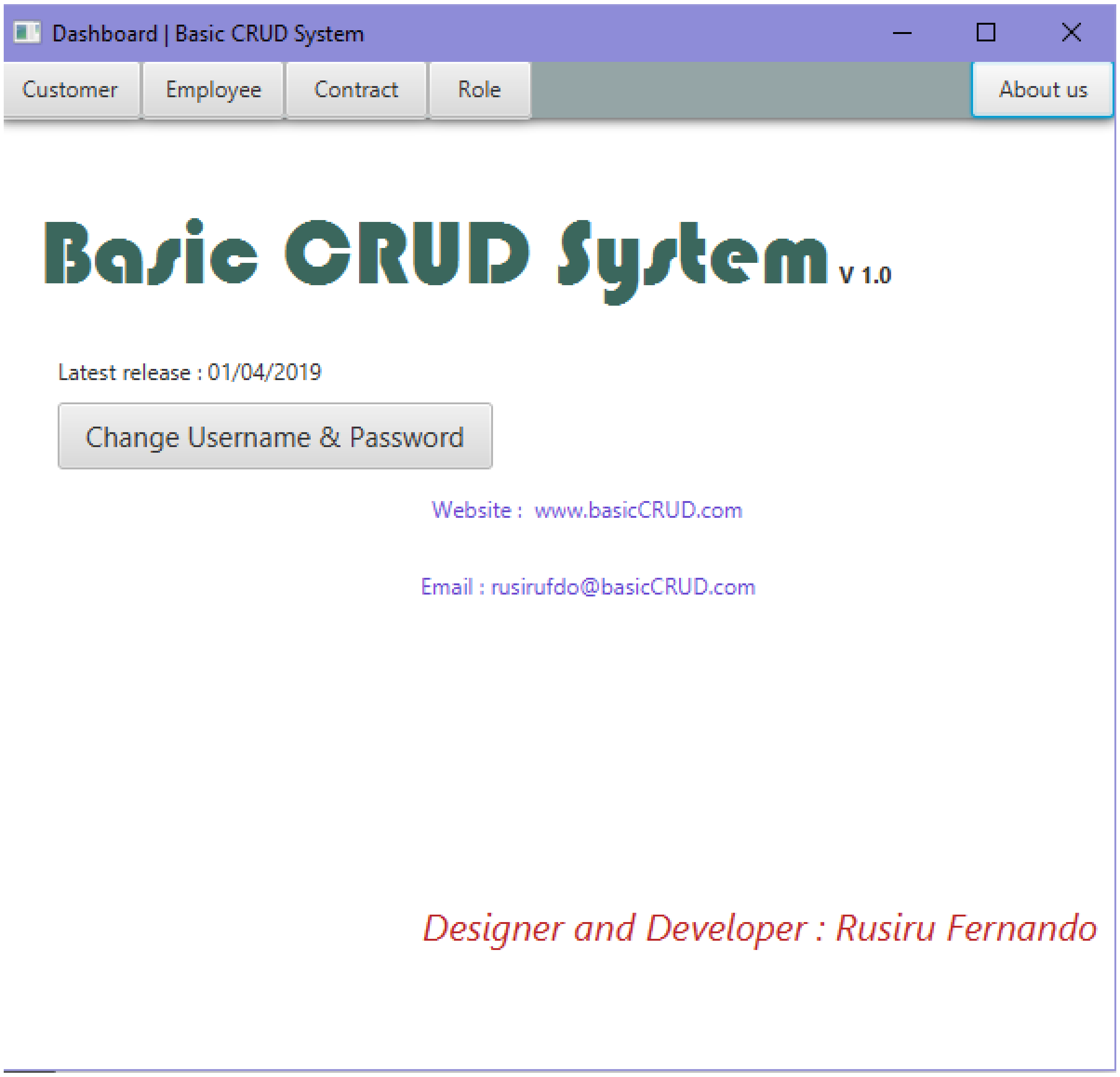
5. Contract Section



6. Role Section



7. About us section



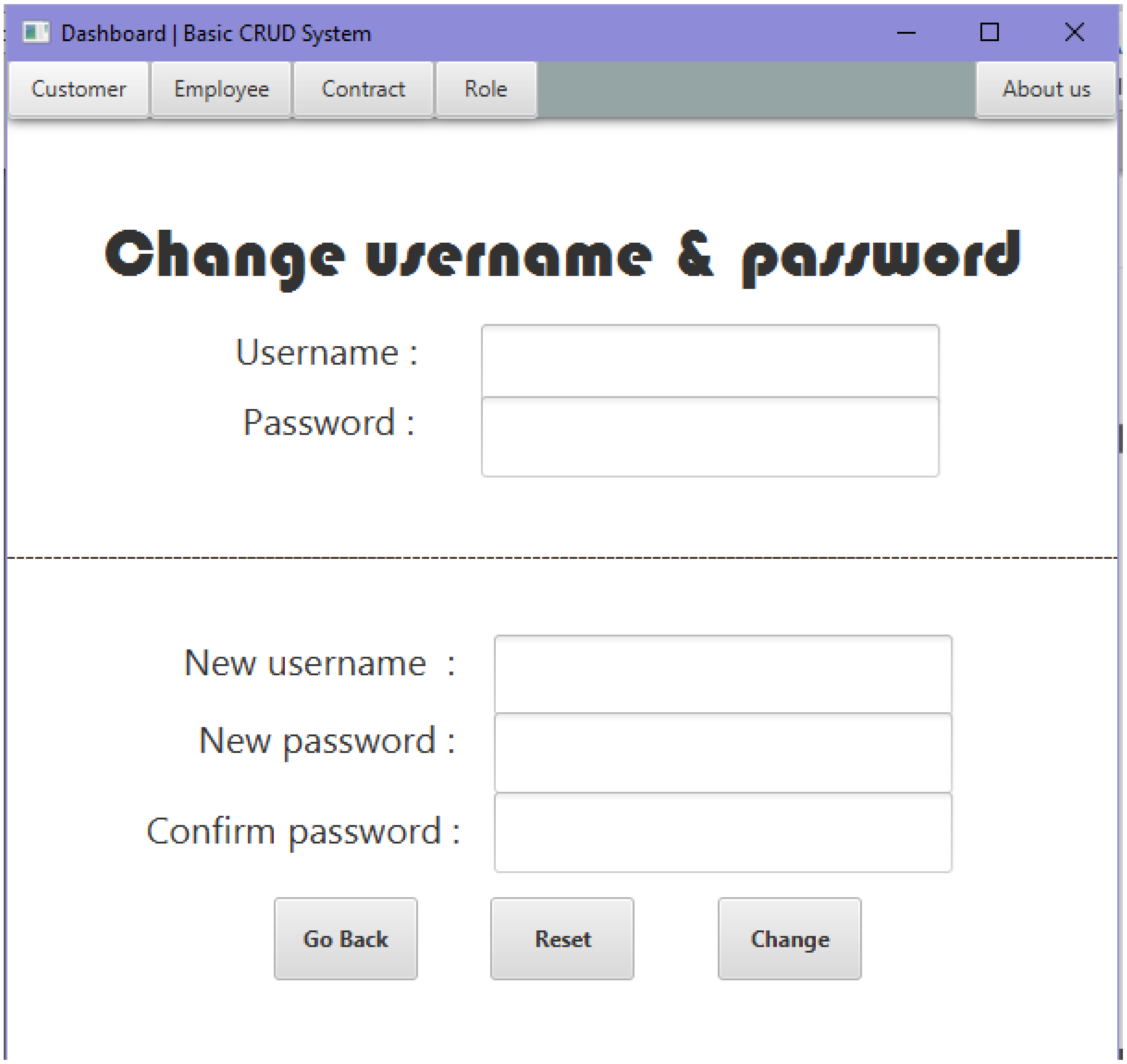
Latest release : 01/04/2019

Website : www.basicCRUD.com

Email : rusirufdo@basicCRUD.com

Designer and Developer : Rusiru Fernando

8. Change Credential section



- **I have implemented all these 8 sections in a single JavaFx Stage by the help of Panes**

Implementation

I have divided the whole program into 8 Java Classes,

1. Main.java

```java
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("Dashboard.fxml"));
        primaryStage.setTitle("Dashboard | Basic CRUD System");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);

    }

}
```

2. Dashboard.java

```java
3.  package sample;

    import java.net.URL;
    import java.util.ResourceBundle;
    import javafx.application.Platform;
    import javafx.collections.FXCollections;
    import javafx.collections.ObservableList;
    import javafx.event.ActionEvent;
    import javafx.fxml.FXML;
    import javafx.scene.control.*;
    import javafx.scene.control.Button;
    import javafx.scene.control.TextArea;
    import javafx.scene.control.TextField;
    import javafx.scene.layout.AnchorPane;
    import javafx.scene.layout.Pane;
    import javax.swing.*;

    public class Dashboard {

        @FXML private ResourceBundle resources;
        @FXML private URL location;
        @FXML private Pane customerPane;
        @FXML private Pane employeePane;
        @FXML private Pane contractPane;
        @FXML private Pane rolePane;
        @FXML private Pane credPane;
        @FXML private Pane aboutPane;
        @FXML private Pane loginPane;
        @FXML private Button menuCustomer;
        @FXML private Button menuEmployee;
        @FXML private Button menuContract;
        @FXML private Button menuRoles;
        @FXML private Button menuHelp;
        @FXML private TextField txtCusID;
        @FXML private TextArea txtCusAddress;
        @FXML private TextField txtCusFname;
        @FXML private TextField txtCusLname;
        @FXML private TextField txtCusNo;
        @FXML private TextField txtCusConID;
        @FXML private Button changeCred;
        @FXML private Button roleAdd;
        @FXML private Button roleView;
        @FXML private TextField txtRoleID;
        @FXML private TextField txtHourlyPay;
        @FXML private ComboBox roleCombo;
        @FXML private Button addEmpbtn;
        @FXML private Button viewEmpBtn;
        @FXML private Button upEmpbtn;
        @FXML private Button delEmpbtn;
        @FXML private Button resetEmpbtn;
        @FXML private TextField txtEmpID;
        @FXML private TextField txtEmpFname;
        @FXML private TextField txtEmpLname;
        @FXML private TextField txtEmpPhone;
        @FXML private TextArea txtEmpAddress;
        @FXML private TextField txtEmpRole1;
        @FXML private TextField txtEmpRole2;
        @FXML private TextField txtEmpRole3;
        @FXML private DatePicker txtEmpDOB;
        @FXML private Button conAddBtn;
        @FXML private Button conResetBtn;
        @FXML private Button conUpBtn;
        @FXML private Button conViewBtn;
        @FXML private Button conDelBtn;
        @FXML private TextField txtconID;
        @FXML private TextField txtconName;
        @FXML private TextField txtconDescription;
        @FXML private DatePicker txtconDate;
        @FXML private TextField txtconJobType;
        @FXML private TextField txtconProjLeaderID;
        @FXML private TextField txtusername;
        @FXML private PasswordField txtpassword;
        @FXML private Button exitbtn;
        @FXML private Button loginbtn;
        @FXML private AnchorPane loginAP;
        @FXML private Button resetCredbtn;
        @FXML private Button gobackbtn;



        ObservableList<String> roleList=
    FXCollections.observableArrayList("Hardware Technician","Programmer","Software
    Installer","Designer","Q/A Engineer" );





        @FXML
        void loadContractPane(ActionEvent event) {
            contractPane.toFront();
        }

        @FXML
        void loadCustomerPane(ActionEvent event) {
            customerPane.toFront();
        }

        @FXML
        void loadEmpPane(ActionEvent event) {
            employeePane.toFront();
        }

        @FXML
        void loadHelpPane(ActionEvent event) {
            aboutPane.toFront();
        }

        @FXML
        void loadRolePane(ActionEvent event) {
            rolePane.toFront();
            roleCombo.setItems(roleList);
        }


    //------------------------------LOGIN SECTION---------------------------------

        private String username="admin";
        private String password="admin123";

        @FXML
        void exitProgram(ActionEvent event) {
            Platform.exit();
            System.exit(0);
        }

        @FXML
        void loadScreen(ActionEvent event) {
            if
    ((txtusername.getText().equals(this.username))&&(txtpassword.getText().equals(t
    his.password))) {
                loginAP.toBack();
            }else{
                JOptionPane.showMessageDialog(null,"Username and password doesn't
    match");
                txtusername.setText("");
                txtpassword.setText("");
            }

        }

    //------------------------------ROLE SECTION-----------------------------------



        @FXML
        void roleViewdb(ActionEvent event) {
            try {
                Role obj = new Role();
                dbConnection connect = new dbConnection();
                boolean state;

                //obj.setRoleID(txtRoleID.getText());
                try {
                    obj.setRoleName(roleCombo.getValue().toString());
                    obj = connect.viewRole(obj);

                    txtRoleID.setText(obj.getRoleID());
                    txtHourlyPay.setText(Double.toString(obj.getPay()));

                    System.out.println(obj.getRoleName());
                } catch (Exception x) {
                    JOptionPane.showMessageDialog(null, "Please select a Role
    name!!!");
                }
            }
            catch (Exception x){
                JOptionPane.showMessageDialog(null,"Connection error!!!");
            }
            //System.out.println(obj.getRoleName())
            //obj.setPay(Double.parseDouble(txtHourlyPay.getText()));


        }
        @FXML
        public void upRole(){
            try {
                Role obj = new Role();
                dbConnection connect = new dbConnection();
                try {
                    obj.setRoleName(roleCombo.getValue().toString());
                    obj.setPay(Double.parseDouble(txtHourlyPay.getText()));
                    obj.setRoleID(txtRoleID.getText());
                    connect.updateRole(obj);
                    JOptionPane.showMessageDialog(null, "Update Successful!!!");
                } catch (Exception x) {
                    JOptionPane.showMessageDialog(null, "Please select a Role
    name!!!");
                }
            }catch (Exception x){
                JOptionPane.showMessageDialog(null,"Connection Error!!!");
            }
        }







    // ---------------------------CUSTOMER SECTION---------------------------------
    --------
        @FXML
        void addCus(ActionEvent event) {
            boolean state = false;

            dbConnection connect = new dbConnection();
            Customer obj = new Customer();

            Validation val = new Validation();
            if (val.checkID(txtCusID.getText(),"cus")) {

                if ((txtCusID.getLength() > 0) && (txtCusFname.getLength() > 0) &&
    (txtCusFname.getLength() > 0) && (txtCusNo.getLength() > 0) &&
    (txtCusAddress.getLength() > 0) && (txtCusConID.getLength() > 0)) {

                    obj.setCustomer_ID(txtCusID.getText());
                    obj.setCustomer_fname(txtCusFname.getText());
                    obj.setCustomer_lname(txtCusLname.getText());
                    obj.setCustomer_no(txtCusNo.getText());
                    obj.setCustomer_address(txtCusAddress.getText());
                    obj.setCustomer_contract(txtCusConID.getText());

                    state = connect.addCustomer(obj);

                    if (state) {
                        JOptionPane.showMessageDialog(null, "Data Entry
    Successful!!!");
                        txtCusID.setText("");
                        txtCusFname.setText("");
                        txtCusLname.setText("");
                        txtCusNo.setText("");
                        txtCusAddress.setText("");
                        txtCusConID.setText("");
                    } else {
                        JOptionPane.showMessageDialog(null, "Data Entry
    Aborted!!!");
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Please fill all
    fields!!!");
                }

            }else{
                if (txtCusID.getLength()==0){
                    JOptionPane.showMessageDialog(null,"Customer ID required !!!");
                }else {
                    JOptionPane.showMessageDialog(null, "Invalid Customer ID !!!");
                }
            }
        }



        @FXML
        void resetCus(ActionEvent event) {
            txtCusID.setText("");
            txtCusFname.setText("");
            txtCusLname.setText("");
            txtCusNo.setText("");
            txtCusAddress.setText("");
        }

        @FXML
        void upCus(ActionEvent event) {
            boolean state=false;
            dbConnection connect=new dbConnection();
            Customer obj=new Customer();
            Validation valid=new Validation();

            if(valid.checkID(txtCusID.getText(),"cus")) {

                if (txtCusID.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Please enter a Customer ID
    and View to Update!!!");
                } else {
                    try {
                        obj.setCustomer_ID(txtCusID.getText());
                        obj.setCustomer_ID(txtCusID.getText());
                        obj.setCustomer_fname(txtCusFname.getText());
                        obj.setCustomer_lname(txtCusLname.getText());
                        obj.setCustomer_no(txtCusNo.getText());
                        obj.setCustomer_address(txtCusAddress.getText());

                        state = connect.updateCustomer(obj);
                    } catch (Exception e) {
                        JOptionPane.showMessageDialog(null, "Connection Error!!!");
                    }
                }


                if (state) {
                    JOptionPane.showMessageDialog(null, "Record Update
    Successful!!!");
                    txtCusID.setText("");
                    txtCusFname.setText("");
                    txtCusLname.setText("");
                    txtCusNo.setText("");
                    txtCusAddress.setText("");
                    txtCusConID.setText("");
                } else {
                    JOptionPane.showMessageDialog(null, "Record update
    Aborted!!!");
                }
            }else{
                JOptionPane.showMessageDialog(null, "Invalid Customer ID!!!");
            }
        }

        @FXML
        void viewCus(ActionEvent event) {
            dbConnection connect = new dbConnection();
            Customer obj = new Customer();
            Validation valid=new Validation();

            if(valid.checkID(txtCusID.getText(),"cus"))
                if (txtCusID.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Please enter a Customer ID
    to View");
                } else {
                    obj.setCustomer_ID(txtCusID.getText());
                    try {
                        obj = connect.searchCustomer(obj);
                    }
                    catch (Exception e){
                        JOptionPane.showMessageDialog(null, "Such record doesn't
    exist!!!");
                    }
                    try {
                        txtCusID.setText(obj.getCustomer_ID());
                        txtCusFname.setText(obj.getCustomer_fname());
                        txtCusLname.setText(obj.getCustomer_lname());
                        txtCusNo.setText(obj.getCustomer_no());
                        txtCusAddress.setText(obj.getCustomer_address());
                        txtCusConID.setText(obj.getCustomer_contract());
                    }
                    catch (Exception l){
                        JOptionPane.showMessageDialog(null,"Connection error");
                    }
            }else
            {
                JOptionPane.showMessageDialog(null,"Invalid Customer ID!!!");
            }
```

```java
    }

    @FXML
    void delCus(ActionEvent event){
        boolean state=false;

        Validation valid=new Validation();
        if(valid.checkID(txtCusID.getText(),"cus")) {
            try {
                dbConnection connect = new dbConnection();
                Customer obj = new Customer();


                if (txtCusID.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Please enter a
Customer ID and View in order to delete it!!!");
                } else {
                    obj.setCustomer_ID(txtCusID.getText());
                    try {
                        state = connect.deleteCustomer(obj);
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Such record
doesn't exist!!!");
                    }
                }

            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "Connection Error!!!");
            }

            if (state) {
                JOptionPane.showMessageDialog(null, "Record Deletion
Successful!!!");
                txtCusID.setText("");
                txtCusFname.setText("");
                txtCusLname.setText("");
                txtCusNo.setText("");
                txtCusAddress.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Record Deletion
Aborted!!!");
            }
        }else{
            JOptionPane.showMessageDialog(null, "Invalid Customer ID!!!");
        }
    }
// ------------------------------EMPLOYEE SECTION-----------------------------
----------
    @FXML
    void addEmp(ActionEvent event) {
        boolean state = false;
            dbConnection connect = new dbConnection();
            Employee obj = new Employee();

            Validation val = new Validation();
            if (val.checkID(txtEmpID.getText(),"emp")) {

                if ( (txtEmpFname.getLength() > 0) && (txtEmpLname.getLength()
> 0) && (txtEmpAddress.getLength() > 0) && (txtEmpAddress.getLength() > 0) ) {

                    obj.setEmp_ID(txtEmpID.getText());
                    obj.setEmp_fname(txtEmpFname.getText());
                    obj.setEmp_lname(txtEmpLname.getText());
                    obj.setEmp_phone(txtEmpPhone.getText());
                    obj.setEmp_address(txtEmpAddress.getText());
                    try {
                        obj.setEmp_DOB(txtEmpDOB.getValue().toString());
                        try {
                            obj.setEmp_roleID_01(txtEmpRole1.getText());
                            obj.setEmp_roleID_02(txtEmpRole2.getText());
                            obj.setEmp_roleID_03(txtEmpRole3.getText());
                            state = true;
                            connect.addEmp(obj);
                        } catch (Exception js) {
                            JOptionPane.showMessageDialog(null, "Invalid Role
ID's");
                        }

                        if (state) {
                            JOptionPane.showMessageDialog(null, "Data Entry
Successful!!!");
                            txtEmpID.setText("");
                            txtEmpFname.setText("");
                            txtEmpLname.setText("");
                            txtEmpDOB.setValue(null);
                            txtEmpPhone.setText("");
                            txtEmpAddress.setText("");
                            txtEmpRole1.setText("");
                            txtEmpRole2.setText("");
                            txtEmpRole3.setText("");
                        } else {
                            JOptionPane.showMessageDialog(null, "Data Entry
Aborted!!!");
                        }
                    }catch (Exception x){
                        JOptionPane.showMessageDialog(null, "Please enter
DOB!!!");
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Please fill all
fields!!!");
                }

        }else{
            if (txtEmpID.getLength()==0){
                JOptionPane.showMessageDialog(null,"Employee ID required !!!");
            }else {
                JOptionPane.showMessageDialog(null, "Invalid Employee ID !!!");
            }
        }
    }

    @FXML
    void viewEmp(ActionEvent event) {
        dbConnection connect = new dbConnection();
        Employee obj = new Employee();
        Validation val = new Validation();
        if (val.checkID(txtEmpID.getText(),"emp")) {
            if (txtEmpID.getText().equals("")) {
                JOptionPane.showMessageDialog(null, "Please enter a Employee ID
to View");
            } else {
                obj.setEmp_ID(txtEmpID.getText());
                try {
                    obj = connect.viewEmp(obj);
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null, "Such record doesn't
exist!!!");
                }
                try {
                    txtEmpID.setText(obj.getEmp_ID());
                    txtEmpFname.setText(obj.getEmp_fname());
                    txtEmpLname.setText(obj.getEmp_lname());
                    txtEmpDOB.setValue(null);
                    txtEmpPhone.setText(obj.getEmp_phone());
                    txtEmpAddress.setText(obj.getEmp_address());
                    txtEmpRole1.setText(obj.getEmp_roleID_01());
                    txtEmpRole2.setText(obj.getEmp_roleID_02());
                    txtEmpRole3.setText(obj.getEmp_roleID_03());
                } catch (Exception l) {
                    JOptionPane.showMessageDialog(null, "Connection error");
                }
            }
        }else{
            JOptionPane.showMessageDialog(null, "Invalid Employee ID!!!");
        }
    }
    @FXML
    void upEmp(ActionEvent event) {

        Employee obj = new Employee();

        Validation val = new Validation();
        if (val.checkID(txtEmpID.getText(), "emp"))
            if (txtEmpID.getText().equals("")) {
                JOptionPane.showMessageDialog(null, "Please enter a Employee ID
and View to Update!!!");
            } else {
                try {
                    dbConnection connect = new dbConnection();
                    obj.setEmp_ID(txtEmpID.getText());
                    obj.setEmp_fname(txtEmpFname.getText());
                    obj.setEmp_lname(txtEmpLname.getText());
                    obj.setEmp_phone(txtEmpPhone.getText());
                    obj.setEmp_address(txtEmpAddress.getText());
                    try {
                        obj.setEmp_DOB(txtEmpDOB.getValue().toString());

                        try {
                            obj.setEmp_roleID_01(txtEmpRole1.getText());
                            obj.setEmp_roleID_02(txtEmpRole2.getText());
                            obj.setEmp_roleID_03(txtEmpRole3.getText());
                            connect.updateEmp(obj);
                            JOptionPane.showMessageDialog(null, "Update
Successful!!");
                        } catch (Exception js) {
                            JOptionPane.showMessageDialog(null, "Invalid Role
ID's");
                        }
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Please enter
DOB!!!");
                    }

                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null, "Connection Error!!!");
                }
            }else{
                JOptionPane.showMessageDialog(null,"Invalid Employee ID!!!");
        }

    }

    @FXML
    void delEmp(ActionEvent event) {
        Validation val = new Validation();
        if (val.checkID(txtEmpID.getText(),"emp")) {
            boolean state = false;
            try {
                dbConnection connect = new dbConnection();
                Employee obj = new Employee();


                if (txtEmpID.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Please enter a
Employee ID and View in order to delete it!!!");
                } else {
                    obj.setEmp_ID(txtEmpID.getText());
                    try {
                        state = connect.deleteEmp(obj);
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Such record
doesn't exist!!!");
                    }
                }

            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "Connection Error!!!");
            }

            if (state) {
                JOptionPane.showMessageDialog(null, "Record Deletion
Successful!!!");
                txtEmpID.setText("");
                txtEmpFname.setText("");
                txtEmpLname.setText("");
                txtEmpDOB.setValue(null);
                txtEmpPhone.setText("");
                txtEmpAddress.setText("");
                txtEmpRole1.setText("");
                txtEmpRole2.setText("");
                txtEmpRole3.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Record Deletion
Aborted!!!");
            }
        }else {
            JOptionPane.showMessageDialog(null, "Invalid Employee ID!!!");
        }
    }
    @FXML
    void resetEmp(ActionEvent event) {
        txtEmpID.setText("");
        txtEmpFname.setText("");
        txtEmpLname.setText("");
        txtEmpDOB.setValue(null);
        txtEmpPhone.setText("");
        txtEmpAddress.setText("");
        txtEmpRole1.setText("");
        txtEmpRole2.setText("");
        txtEmpRole3.setText("");
    }

//------------------------------------CONTRACT SECTION------------------------
-----------

    @FXML
    void conAdd(ActionEvent event) {
        boolean state = false;
        try {
            dbConnection connect = new dbConnection();
            Contract obj = new Contract();

            Validation val = new Validation();
            if (val.checkID(txtconID.getText(), "con")) {

                if ((txtconName.getLength() > 0) &&
(txtconDescription.getLength() > 0) && (txtconJobType.getLength() > 0) &&
(txtconProjLeaderID.getLength() > 0)) {

                    obj.setCon_ID(txtconID.getText());
                    obj.setCon_name(txtconName.getText());
                    obj.setCon_description(txtconDescription.getText());
                    obj.setCon_jobType(txtconJobType.getText());
                    try {
                        obj.setCon_date(txtconDate.getValue().toString());
                        try {
obj.setCon_projLeaderID(txtconProjLeaderID.getText());
                            state =connect.addCon(obj);

                        } catch (Exception js) {
                            JOptionPane.showMessageDialog(null, "Invalid
Project Leader ID's");
                        }

                        if (state) {
                            JOptionPane.showMessageDialog(null, "Data Entry
Successful!!!");
                            txtconID.setText("");
                            txtconName.setText("");
                            txtconDescription.setText("");
                            txtconDate.setValue(null);
                            txtconJobType.setText("");
                            txtconProjLeaderID.setText("");
                        } else {
                            JOptionPane.showMessageDialog(null, "Data Entry
Aborted!!!");
                        }
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Please enter
DOB!!!");
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Please fill all
fields!!!");
                }

            } else {
                if (txtEmpID.getLength() == 0) {
                    JOptionPane.showMessageDialog(null, "Contract ID required
!!!");
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid Contract ID
!!!");
                }
            }
        } catch (Exception x) {
            JOptionPane.showMessageDialog(null, "Connection Error!!!");
        }
    }

    @FXML
    void conDel(ActionEvent event) {
        Validation val = new Validation();
        if (val.checkID(txtconID.getText(),"con")) {
            boolean state = false;
            try {
                dbConnection connect = new dbConnection();
                Contract obj = new Contract();


                if (txtconID.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Please enter a
Contract ID and View in order to delete it!!!");
                } else {
                    obj.setCon_ID(txtconID.getText());
                    try {
                        state = connect.deleteCon(obj);
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Such record
doesn't exist!!!");
                    }
                }

            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "Connection Error!!!");
            }

            if (state) {
                JOptionPane.showMessageDialog(null, "Record Deletion
Successful!!!");
                txtconID.setText("");
                txtconName.setText("");
                txtconDescription.setText("");
                txtconDate.setValue(null);
                txtconJobType.setText("");
                txtconProjLeaderID.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Record Deletion
Aborted!!!");
            }
        }else {
            JOptionPane.showMessageDialog(null, "Invalid Contract ID!!!");
        }
    }

    @FXML
    void conReset(ActionEvent event) {
        txtconID.setText("");
        txtconName.setText("");
        txtconDescription.setText("");
        txtconDate.setValue(null);
        txtconJobType.setText("");
        txtconProjLeaderID.setText("");
    }
    @FXML
    void conUp(ActionEvent event) {
        try{
            boolean state=false;
            dbConnection connect = new dbConnection();
            Contract obj = new Contract();
            Validation val = new Validation();
            if ((txtconName.getLength() > 0) && (txtconDescription.getLength()
> 0) && (txtconJobType.getLength() > 0) && (txtconProjLeaderID.getLength() >
0)) {
                if (val.checkID(txtconID.getText(), "con")) {
                    obj.setCon_ID(txtconID.getText());
                    obj.setCon_name(txtconName.getText());
                    obj.setCon_description(txtconDescription.getText());
                    obj.setCon_jobType(txtconJobType.getText());
                    try {
                        obj.setCon_date(txtconDate.getValue().toString());
                        try {

obj.setCon_projLeaderID(txtconProjLeaderID.getText());
                            state =connect.upCon(obj);

                        } catch (Exception js) {
                            JOptionPane.showMessageDialog(null, "Invalid
Project Leader ID's");
                        }

                        if (state) {
                            JOptionPane.showMessageDialog(null, "Update
Successful!!!");
                            txtconID.setText("");
                            txtconName.setText("");
                            txtconDescription.setText("");
                            txtconDate.setValue(null);
                            txtconJobType.setText("");
                            txtconProjLeaderID.setText("");
                        } else {
                            JOptionPane.showMessageDialog(null, "Update
Aborted!!!");
                        }
                    } catch (Exception x) {
                        JOptionPane.showMessageDialog(null, "Please enter
DOB!!!");
                    }
                }else{
                    JOptionPane.showMessageDialog(null,"Incorrect Contract
ID!!!");
                }
            }else{
                JOptionPane.showMessageDialog(null,"Fill all the Fields");
            }

        }catch (Exception e){
            JOptionPane.showMessageDialog(null,"Connection ERROR!!");
        }
    }

    @FXML
    void conView(ActionEvent event) {
        try {
            dbConnection connect = new dbConnection();
            Contract obj = new Contract();
            Validation val = new Validation();
```

```
                if (val.checkID(txtconID.getText(), "con")) {
                    if (txtconID.getText().equals("")) {
                        JOptionPane.showMessageDialog(null, "Please enter a 
Contract ID to View");
                    } else {
                        obj.setCon_ID(txtconID.getText());
                        try {
                            obj = connect.viewCon(obj);
                        } catch (Exception e) {
                            JOptionPane.showMessageDialog(null, "Such record 
doesn't exist!!!");
                        }
                        try {
                            txtconID.setText(obj.getCon_ID());
                            txtconName.setText(obj.getCon_name());
                            txtconDescription.setText(obj.getCon_description());
                            txtconDate.setValue(null);
                            txtconJobType.setText(obj.getCon_jobType());
                            txtconProjLeaderID.setText(obj.getCon_projLeaderID());
                        } catch (Exception l) {
                            JOptionPane.showMessageDialog(null, "Connection 
error");
                        }
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid Contract ID!!!");
                }
            }catch (Exception x){
                JOptionPane.showMessageDialog(null,"Connection Error!!!");
            }
        }

//----------------------OPEN CRED CHANGE SECTION------------------------------
-----
    @FXML
    void changeLogin(ActionEvent event) {
        credPane.toFront();
    }

//----------------------CHANGE CREDENTIAL SECTION-----------------------------
-----
    @FXML
    void goback() {
        aboutPane.toFront();
    }

    @FXML
    void resetCred(ActionEvent event) {

    }

    @FXML
    void changeCred(ActionEvent event) {
        if 
((txtusername.getText().equals(this.username))&&(txtpassword.getText().equals(t
his.password))) {
            loginAP.toBack();
        }else{
            JOptionPane.showMessageDialog(null,"Username and password doesnt 
match");
            txtusername.setText("");
            txtpassword.setText("");
        }
        }
    }
```

3.Validation.java (This class validates ID's)

```
package sample;



public class Validation {

    private boolean state;
    private String letters;
    private int numbers;


    public boolean checkID(String ID,String typeID){
        if(ID.length()>0) {
            this.letters = ID.substring(0, 3);
            try {
                this.numbers = Integer.parseInt(ID.substring(3));
                if (this.letters.equals(typeID)) {
                    this.state = true;
                }
            } catch (Exception e) {
                return false;
            }
        }else{
            return false;
        }

        return this.state;
    }

}
```

4. dbConnection.java

```
package sample;

import javafx.fxml.FXML;

import javax.swing.*;
import java.sql.*;

public class dbConnection {
    private Connection con;
    private Statement st;
    private ResultSet rs;

    public dbConnection() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/computer_consultancy_firm",
"root", "");
            st = con.createStatement();
        } catch (Exception ex) {
            System.out.println("Connection error...");
        }
    }

//----------------------CUSTOMER SECTION------------------------------------------------
-----------

    public boolean addCustomer(Customer obj){
        PreparedStatement pst=null;

        try{
            pst=con.prepareStatement("insert into customer values(?,?,?,?,?,?)");
            pst.setString(1,obj.getCustomer_ID());
            pst.setString(2,obj.getCustomer_fname());
            pst.setString(3,obj.getCustomer_lname());
            pst.setString(4,obj.getCustomer_no());
            pst.setString(5,obj.getCustomer_address());
            pst.setString(6,obj.getCustomer_contract());
            pst.executeUpdate();
            return true;
        }catch(Exception em){
            JOptionPane.showMessageDialog(null,"Customer ID already exists!!!");
            return false;
        }
    }

    public Customer searchCustomer(Customer obj){
       try{
           String query="SELECT * FROM customer ";
           rs=st.executeQuery(query);
           while (rs.next()){
               if (rs.getString("cus_ID").equals(obj.getCustomer_ID())) {
                   obj.setCustomer_fname(rs.getString("cus_fname"));
                   obj.setCustomer_lname(rs.getString("cus_lname"));
                   obj.setCustomer_no(rs.getString("cus_phone"));
                   obj.setCustomer_address(rs.getString("cus_address"));
                   obj.setCustomer_contract(rs.getString("cus_contractID"));
                   break;
               }
           }
           return obj;
        }catch (Exception ex){
            JOptionPane.showMessageDialog(null,"Such ID doesn't exist ");
            return null;
        }
    }

    public boolean updateCustomer(Customer obj){

            PreparedStatement pst=null;
            try{
                pst=con.prepareStatement("update customer set 
cus_fname=?,cus_lname=?,cus_phone=?,cus_address=?,cus_contractID=? where cus_ID=?");
                pst.setString(1,obj.getCustomer_fname());
                pst.setString(2,obj.getCustomer_lname());
                pst.setString(3,obj.getCustomer_no());
                pst.setString(4,obj.getCustomer_address());
                pst.setString(5,obj.getCustomer_contract());
                pst.setString(6,obj.getCustomer_ID());
                pst.executeUpdate();
                return true;
            }catch(Exception e){
                return false;
            }
    }

    public boolean deleteCustomer(Customer obj){
        PreparedStatement pst=null;
        try{
            pst=con.prepareStatement("delete from customer where cus_ID=?");
            pst.setString(1,obj.getCustomer_ID());
            pst.executeUpdate();
            return true;
        }catch(Exception e){
            return false;
        }
    }

    public Role viewRole(Role obj){
        try{
            String query="SELECT * FROM role ";
            rs=st.executeQuery(query);
            while (rs.next()){
                if (rs.getString("role_description").equals(obj.getRoleName())) {
                    obj.setRoleID(rs.getString("role_ID"));
                    obj.setPay(Double.parseDouble(rs.getString("role_hourlyPay")));
                    break;
                }
            }
            return obj;


        }catch (Exception ex){
            JOptionPane.showMessageDialog(null,"Connection Error ");
            return null;
        }
        }

        public void updateRole(Role obj){
            PreparedStatement pst=null;
            try{
                pst=con.prepareStatement("update role set role_hourlyPay=? where 
role_ID=?");
                pst.setDouble(1,obj.getPay());
                pst.setString(2,obj.getRoleID());


            }catch(Exception e){
                JOptionPane.showMessageDialog(null,"Connection error!!!");
            }
        }
//-----------------------------------EMPLOYEE SECTION--------------------------------
---
    public boolean addEmp(Employee obj){
        PreparedStatement pst=null;

        try{
            pst=con.prepareStatement("insert into employee
values(?,?,?,?,?,?,?,?,?)");
            pst.setString(1,obj.getEmp_ID());
            pst.setString(2,obj.getEmp_fname());
            pst.setString(3,obj.getEmp_lname());
            pst.setString(4,obj.getEmp_DOB());
            pst.setString(5,obj.getEmp_phone());
            pst.setString(6,obj.getEmp_address());
            pst.setString(7,obj.getEmp_roleID_01());
            pst.setString(8,obj.getEmp_roleID_02());
            pst.setString(9,obj.getEmp_roleID_03());
            pst.executeUpdate();
            return true;
        }catch(Exception em){
            JOptionPane.showMessageDialog(null,"Employee ID already exists!!!");
            return false;
        }
    }

    public Employee viewEmp(Employee obj){
        try{
            String query="SELECT * FROM employee ";
            rs=st.executeQuery(query);
            while (rs.next()){
                if (rs.getString("emp_ID").equals(obj.getEmp_ID())) {
                    obj.setEmp_fname(rs.getString("emp_fname"));
                    obj.setEmp_lname(rs.getString("emp_lname"));
                    obj.setEmp_DOB(null);
                    obj.setEmp_phone(rs.getString("emp_phone"));
                    obj.setEmp_address(rs.getString("emp_address"));
                    obj.setEmp_roleID_01(rs.getString("emp_roleID_01"));
                    obj.setEmp_roleID_02(rs.getString("emp_roleID_02"));
                    obj.setEmp_roleID_03(rs.getString("emp_roleID_03"));
                    break;
                }
            }
            return obj;


        }catch (Exception ex){
            JOptionPane.showMessageDialog(null,"Connection Error ");
            return null;
        }
    }

    public boolean updateEmp(Employee obj){
        PreparedStatement pst=null;
        try{
            pst=con.prepareStatement("update employee set
emp_fname=?,emp_lname=?,emp_DOB=?,emp_phone=?,emp_address=?,emp_roleID_01=?,emp_roleID
```

```
_02=?,emp_roleID_03=? where emp_ID=?");
            pst.setString(1,obj.getEmp_fname());
            pst.setString(2,obj.getEmp_lname());
            pst.setString(3,obj.getEmp_DOB());
            pst.setString(4,obj.getEmp_phone());
            pst.setString(5,obj.getEmp_address());
            pst.setString(6,obj.getEmp_roleID_01());
            pst.setString(7,obj.getEmp_roleID_02());
            pst.setString(8,obj.getEmp_roleID_03());
            pst.setString(9,obj.getEmp_ID());
            pst.executeUpdate();
            return true;
        }catch(Exception e){
            return false;
        }
    }

    public boolean deleteEmp(Employee obj){
        PreparedStatement pst=null;
        try{
            pst=con.prepareStatement("delete from employee where emp_ID=?");
            pst.setString(1,obj.getEmp_ID());
            pst.executeUpdate();
            return true;
        }catch(Exception e){
            return false;
        }
    }

//-----------------------------------CONTRACT SECTION-------------------------------
----

    public boolean addCon(Contract obj){
        PreparedStatement pst=null;

        try{
            pst=con.prepareStatement("insert into contract values(?,?,?,?,?,?)");
            pst.setString(1,obj.getCon_ID());
            pst.setString(2,obj.getCon_name());
            pst.setString(3,obj.getCon_description());
            pst.setString(4,obj.getCon_date());
            pst.setString(5,obj.getCon_jobType());
            pst.setString(6,obj.getCon_projLeaderID());
            pst.executeUpdate();
            return true;
        }catch(Exception em){
            JOptionPane.showMessageDialog(null,"Contract ID already exists!!!");
            return false;
        }
    }

    public Contract viewCon(Contract obj){
        try{
            String query="SELECT * FROM contract ";
            rs=st.executeQuery(query);
            while (rs.next()){
                if (rs.getString("con_ID").equals(obj.getCon_ID())) {
                    obj.setCon_name(rs.getString("con_name"));
                    obj.setCon_description(rs.getString("con_description"));
                    obj.setCon_date(null);
                    obj.setCon_jobType(rs.getString("con_jobType"));
                    obj.setCon_projLeaderID(rs.getString("con_projLeaderID"));
                    break;
                }
            }
            return obj;


        }catch (Exception ex){
            JOptionPane.showMessageDialog(null,"Connection Error ");
            return null;
        }
    }

    public boolean deleteCon(Contract obj){
        PreparedStatement pst=null;
        try{
            pst=con.prepareStatement("delete from contract where con_ID=?");
            pst.setString(1,obj.getCon_ID());
            pst.executeUpdate();
            return true;
        }catch(Exception e){
            return false;
        }
    }

    public boolean upCon(Contract obj){
        PreparedStatement pst=null;
        try{
            pst=con.prepareStatement("update contract set
con_name=?,con_description=?,con_date=?,con_jobType=?,con_projLeaderID=? where
con_ID=?");
            pst.setString(1,obj.getCon_name());
            pst.setString(2,obj.getCon_description());
            pst.setString(3,obj.getCon_date());
            pst.setString(4,obj.getCon_jobType());
            pst.setString(5,obj.getCon_projLeaderID());
            pst.setString(6,obj.getCon_ID());
            pst.executeUpdate();
            return true;
        }catch(Exception e){
            return false;
        }
    }


}
```

5. Customer.java

```
package sample;

public class Customer {
    private String customer_ID;
    private String customer_fname;
    private String customer_lname;
    private String customer_no;
    private String customer_address;
    private String customer_contract;

    public void setCustomer_ID(String cusID){
        this.customer_ID=cusID;
    }

    public String getCustomer_ID(){
        return customer_ID;
    }

    public void setCustomer_fname(String fname){
        this.customer_fname=fname;
    }

    public String getCustomer_fname(){
        return customer_fname;
    }
    public void setCustomer_lname(String lname){
        this.customer_lname=lname;
    }

    public String getCustomer_lname(){
        return customer_lname;
    }
    public void setCustomer_no(String no){
        this.customer_no=no;
    }

    public String getCustomer_no(){
        return customer_no;
    }
    public void setCustomer_address(String address){
        this.customer_address=address;
    }

    public String getCustomer_address(){
        return customer_address;
    }

    public void setCustomer_contract(String contract){
        this.customer_contract=contract;
    }

    public String getCustomer_contract(){
        return customer_contract;
    }
}
```

6. **Employee.java**

```
7. package sample;
```

```java
import java.util.Date;

public class Employee{
    private String emp_ID;
    private String emp_fname;
    private String emp_lname;
    private String emp_DOB;
    private String emp_phone;
    private String emp_address;
    private String emp_roleID_01;
    private String emp_roleID_02;
    private String emp_roleID_03;

    public void setEmp_ID(String ID){
        this.emp_ID=ID;
    }

    public void setEmp_fname(String fname){
        this.emp_fname=fname;
    }

    public void setEmp_lname(String lname){
        this.emp_lname=lname;
    }

    public void setEmp_DOB(String date){
        this.emp_DOB=date;
    }

    public void setEmp_phone(String no){
        this.emp_phone=no;
    }

    public void setEmp_address(String address){
        this.emp_address=address;
    }

    public void setEmp_roleID_01(String ID){
        this.emp_roleID_01=ID;
    }

    public void setEmp_roleID_02(String ID){
        this.emp_roleID_02=ID;
    }

    public void setEmp_roleID_03(String ID){
        this.emp_roleID_03=ID;
    }

    public String getEmp_ID(){
        return emp_ID;
    }

    public String getEmp_fname(){
        return emp_fname;
    }

    public String getEmp_lname(){
        return emp_lname;
    }

    public String getEmp_DOB(){
        return emp_DOB;
    }

    public String getEmp_phone(){
        return emp_phone;
    }

    public String getEmp_address(){
        return emp_address;
    }

    public String getEmp_roleID_01(){
        return this.emp_roleID_01;
    }

    public String getEmp_roleID_02(){
        return this.emp_roleID_02;
    }

    public String getEmp_roleID_03(){
        return this.emp_roleID_03;
    }
}
```

1. **Contract.java**

```
package sample;

public class Contract {


    private String con_ID;
    private String con_name;
    private String con_description;
    private String con_date;
    private String con_jobType;
    private String con_projLeaderID;



    public String getCon_ID() {
        return con_ID;
    }

    public String getCon_name() {
        return con_name;
    }

    public String getCon_description() {
        return con_description;
    }

    public String getCon_date() {
        return con_date;
    }

    public String getCon_jobType() {
        return con_jobType;
    }

    public String getCon_projLeaderID() {
        return con_projLeaderID;
    }

    public void setCon_ID(String ID){
        this.con_ID=ID;
    }

    public void setCon_name(String name){
        this.con_name=name;
    }

    public void setCon_description(String desc){
        this.con_description=desc;
    }

    public void setCon_date(String date){
        this.con_date=date;
    }

    public void setCon_jobType(String job){
        this.con_jobType=job;
    }

    public void setCon_projLeaderID(String ID){
        this.con_projLeaderID=ID;
    }



}
```

8. Role.java

```
9.  package sample;

    public class Role {
        private String roleName;
        private String roleID;
        private double pay;

        public void setRoleName(String name){
            this.roleName=name;
        }

        public void setRoleID(String ID){
            this.roleID=ID;
        }

        public void setPay(double pay){
            this.pay=pay;
        }

        public String getRoleName(){
            return roleName;
        }

        public String getRoleID(){
           return this.roleID;
        }
        public double getPay(){
            return this.pay;
        }
    }
```

Test Cases

- Login Screen

Test case	Result
Correct Username & Correct Password	Access Granted
Incorrect Username & Incorrect Password	Username and Password Mismatch Alert
Empty Username & Empty Password	Username and Password Mismatch Alert
Correct Username & Incorrect Password	Username and Password Mismatch Alert
Incorrect Username & Correct Password	Username and Password Mismatch Alert

```
@FXML
void loadScreen(ActionEvent event) {
    if
((txtusername.getText().equals(this.username))&&(txtpassword.getText().equals(this.pas
sword))) {
        loginAP.toBack();
    }else{
        JOptionPane.showMessageDialog(null,"Username and password doesn't match");
        txtusername.setText("");
        txtpassword.setText("");
    }

}
```

- Adding a record

Test case	Result
All fields completed with valid ID	Record is Added
Empty Fields with valid ID	Record addition aborted
All fields completed with Invalid ID	Record addition aborted
Invalid data types in fields	Record addition aborted

- Updating a record or Deleting a record or Viewing a record

Test Case	Result
Empty ID	ID is compulsory

An ID which doesn't exist in the database	Error "ID does not exist"
ID wrong format	Error ID wrong format
Empty Fields	All fields must be filled message

```
public boolean checkID(String ID,String typeID){
    if(ID.length()>0) {
        this.letters = ID.substring(0, 3);
        try {
            this.numbers = Integer.parseInt(ID.substring(3));
            if (this.letters.equals(typeID)) {
                this.state = true;
            }
        } catch (Exception e) {
            return false;
        }
    }else{
        return false;
    }

    return this.state;
}
```

Each entity has its own unique identification number which is a combination of 3 letters and 3 numbers.

- For Employee entity (emp001)
- For Customer entity (cus001)
- For Contract entity (con001)
- For Role entity (rol001)

Conclusions

Supportive resources : https://www.youtube.com/watch?v=BCqW5XwtJxY , for connecting mySQl DB

For inserting data into DB : https://www.youtube.com/watch?v=q8Z3CmruGzI

Tools used :

- Gluon : for scene building
- Intellij :as Java IDE
- StarUML : for class diagrams, Activity diagrams and Use case diagrams
- DB server : xampp (phpMyAdmin)