

**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.

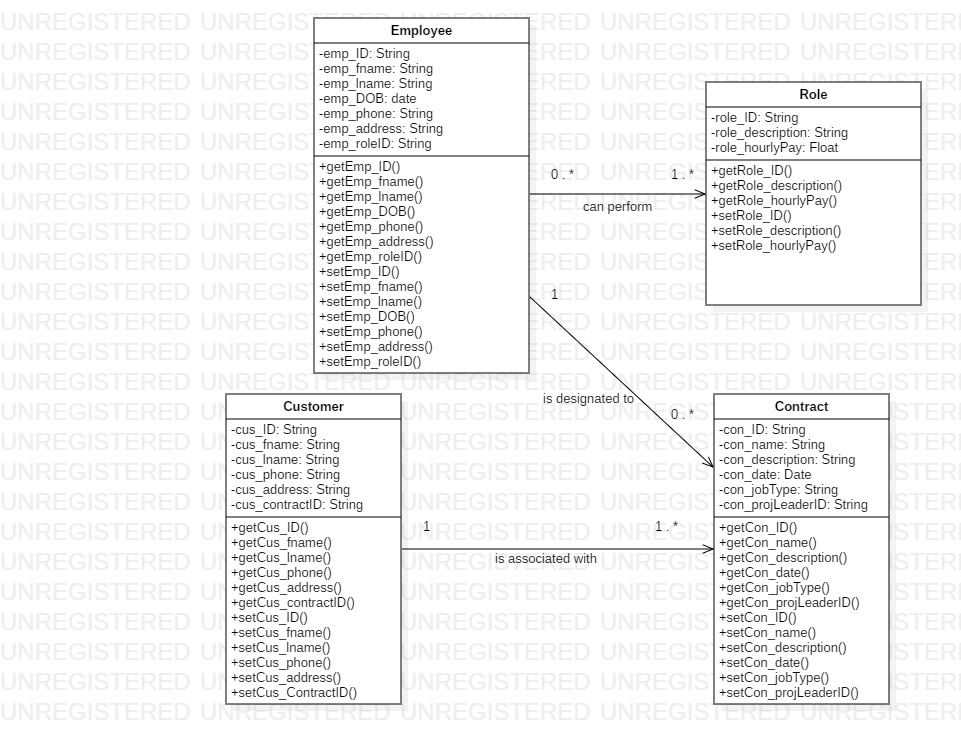
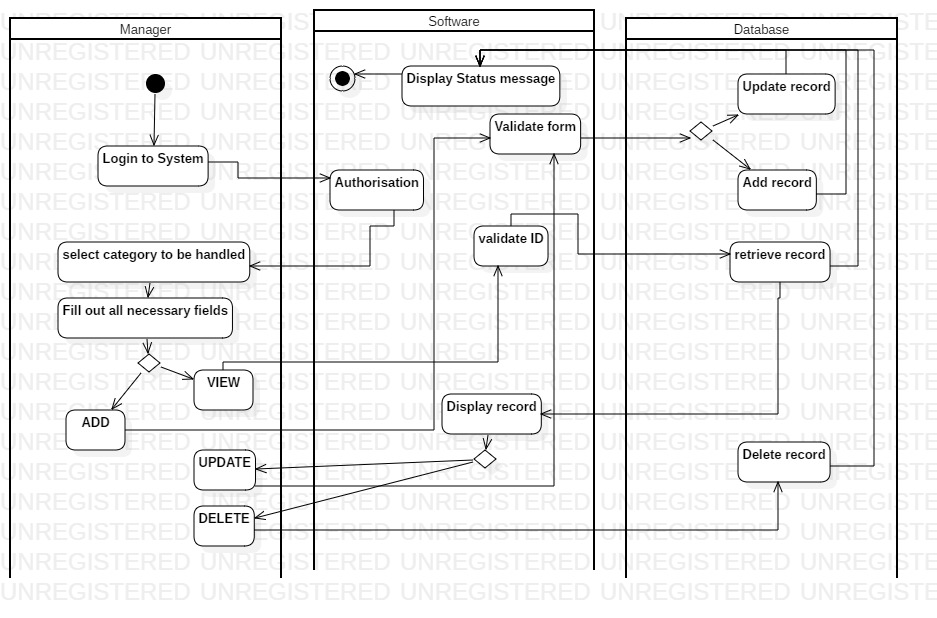
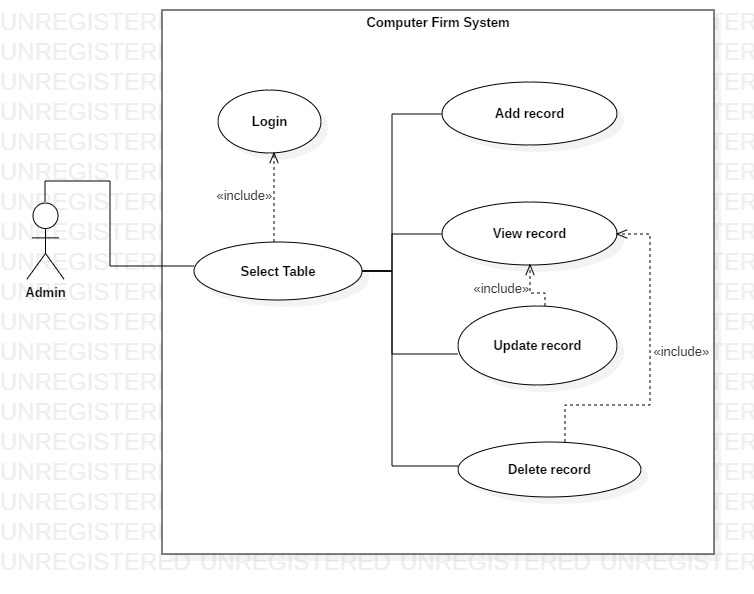
This software is for the use of the manager and only the manager has its login details, and he is the one who is authorized to do any changes to the records.

**Analysis**

1. Functional Requirements
   1. Add Customers, Contracts and Employees (I assumed roles cannot be added as the company has already distinguished the roles)
   2. View Customers, Contracts, Employees and Roles.
   3. Update Customers, Contracts, Employees and Roles.
   4. 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
   1. 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 |

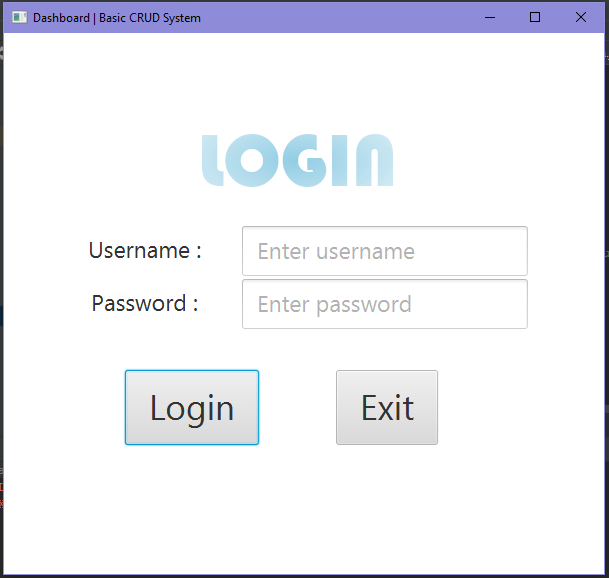
|  |  |
| --- | --- |
| Use case ID | ID005 |
| Use Case Name | Update Record |
| Use Case Description | The user can update an existing record by searching it by its ID |
| Actors | - |
| Pre-Conditions | User has to be logged in  Choose which section is to be updated  Choose which record needs to be 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 |

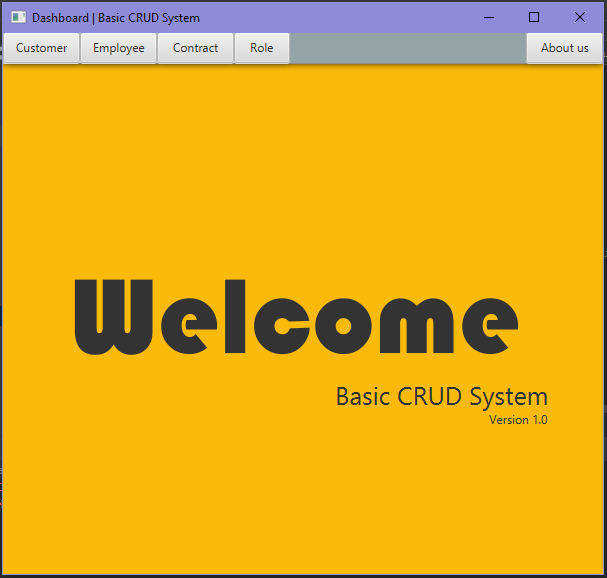
**The UI of the Program**

The UI has 8 sections in total,

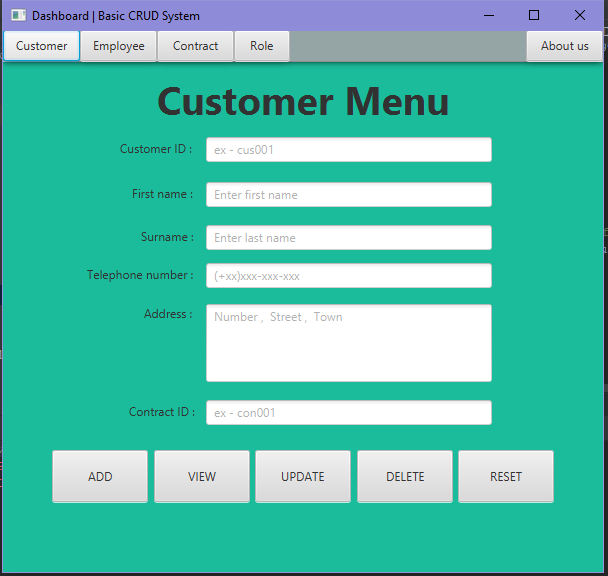
1. **Login section**



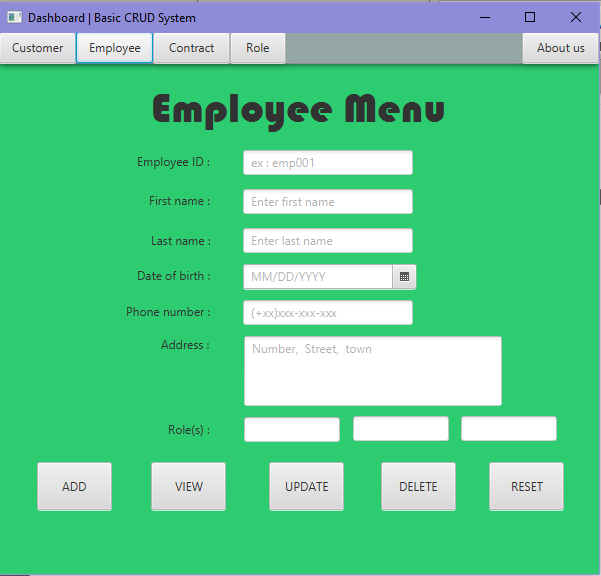
1. **Welcome Screen Section**



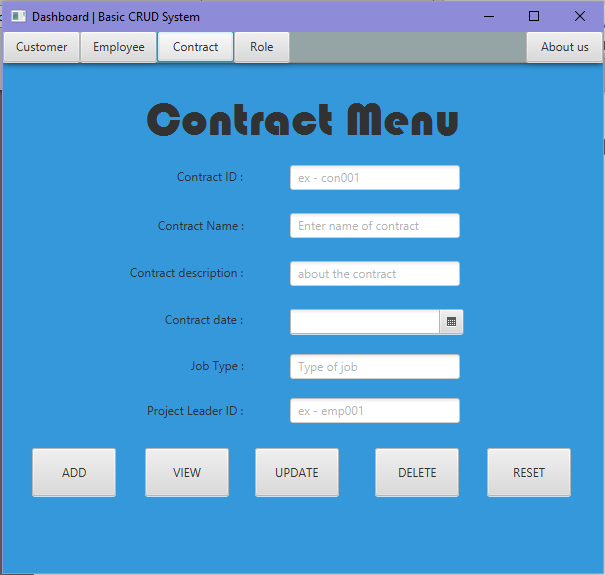
1. **Customer Section**



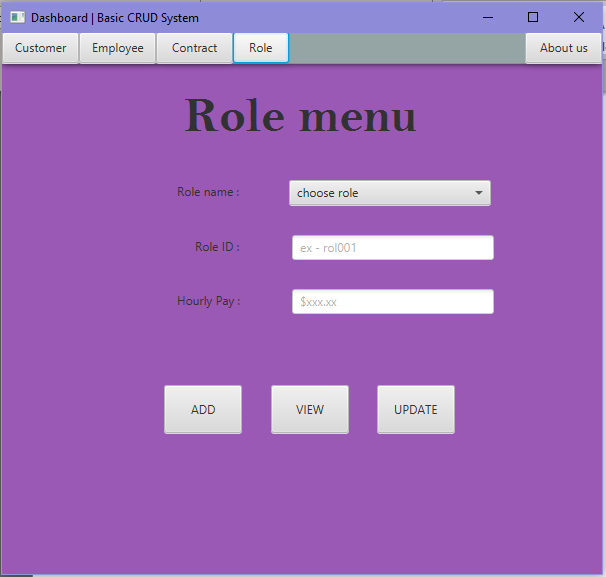
1. **Employee Section**



1. **Contract Section**



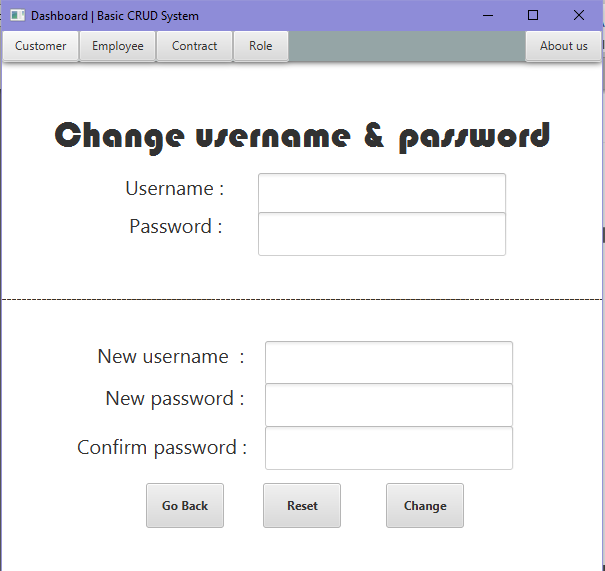
1. **Role Section**



1. **About us section**



1. **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**

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);  
  
 }  
  
}

1. **Dashboard.java**
2. 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(this.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!!!");  
    }  
     
    }  
     
    @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(this.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;  
 }  
  
}

1. **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;  
 }  
 }  
  
  
}

1. **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;  
 }  
}

1. **Employee.java**
2. package sample;  
     
   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;  
    }  
   }
3. **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;  
 }  
  
  
  
}

1. **Role.java**
2. 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.password))) {  
 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)