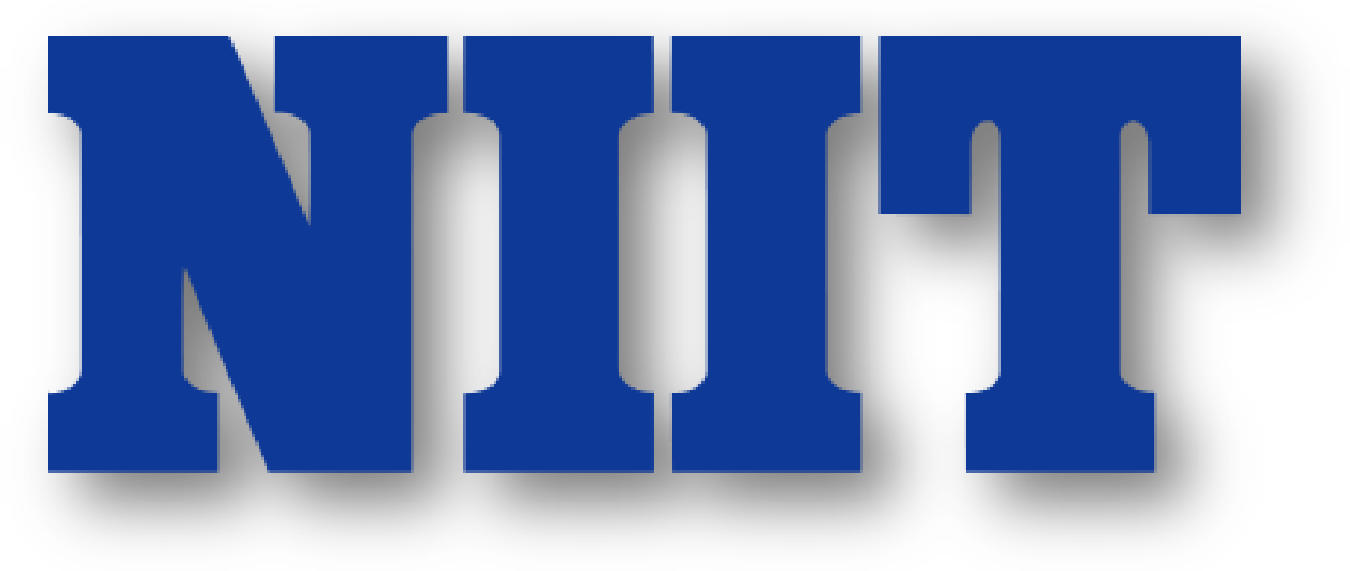
PROJECT ON

**Professionet Consultancy Services (PCS) Application**

DEVELOPED By –

KAMISHETTY ABHILASH



**Professionet Consultancy Services (PCS) Application**

Batch Code: S210192

Start Date : 10th December 2020

End Date : 15th January 2021

Name of the

Coordinator : Mrs. Lopamudra Bera

Name of the

Developer : KAMISHETTY ABHILASH

Date of

Submission : 15th January 2021

**Professionet Consultancy Services (PCS) Application**

CERTIFICATE

This is to certify that this report, titled Professionate Consultancy Services (PCS) Application embodies the original work done by

KAMISHETTY ABHILASH in partial fulfillment of their course requirement at NIIT.

CO-ORDINATOR:

Acknowledgement

*We have benefitted a lot from the feedback and suggestions given to us by Mrs. Lopamudra Bera and other faculty members.*

Abstract

This project is an application for the Professionate Consultancy Services (PCS), that provides the client to freely be able to manage, store and maintain their company records with a good user interface.

Most of the operation will be done on the user interface designed by awt and the database managed by sql and java.

Configuration

Hardware

Processor : Intel i3

RAM : 2GB(minimum)

Speed : 1.5GHz

Secondary

Storage : 10GB

Software :

Java Version : JDK 1.8

Database

Management: MySQL

IDE : Eclipse

OS : Windows 10.

INDEX

|  |  |  |
| --- | --- | --- |
| SNO | Title | Page No. |
|  | Introduction | 08 |
|  | Aim and Objective | 09 |
|  | Case Study | 10 |
|  | Project Requirements | 12 |
|  | Project Analysis | 13 |
|  | Implementation and Diagrams | 14 |
|  | Code | 17 |
|  | Sql code | 34 |
|  | Outputs | 36 |
|  | Challenges Faced | 37 |
|  | References | 38 |

Introduction

This project will help you to implement all the concepts. Strategies, techniques, and technologies its that you are laming in the current ten. In this project. you will integrate all those concepts and mate live project by following all the aspects of software engineering and project management.

You will implement various software engineering and project management concepts in each and every phase of project development. In addition. this project makes you go through the entire software project life cycle, which will help you to work in a real time application development environment.

AIM and Objective

The aim of this project is to design and develop a application for the

Professionet Consultancy Services (PCS).

Objective :

* To connect to the data base.
* Let the admin add update modify and maintain the database and the records.
* Provide a good user interface.
* Test the application.

Case Study

[2]Professionet Consultancy Services (PCS) is a business consultancy that has established itself as a renowned service provider of a wide range of business services to its clients. It was founded on November 25, 1992, by Dr. Monica Roberts, who is now the CEO of the company. PCS is among the top five business services consultancies in London.

The consultancy has its global presence in Asian, African, American, and European markets. It has multiple teams of expert Business Consultants which leverage the business potential of their clients by offering strategic business management services. Some of the prominent services provided by PCS are resource management, process optimization, IT administration, change management, project management, finance management, and risk management. PCS offers business solution to its clients to meet their business needs on a larger scale.

So there is a great significance of aligning PCS employees to their various clients' projects on the basis of skills of each PCS employeeand the project requirement. To meet the human resource requirement of their clients' projects, the consultancy follows conventional recruitment or skill mapping process, i.e., step-by-step consequent process, in which one phase initiates only after the previous phase is completed. PCS offers an offline platform for their employees to share their profiles to initiate internal job posting process with the expert Human Resource (HR) consultants. Face-to face meetings and interview rounds are conducted to align talent supply with business needs. Experienced professionals at PCS can directly consult with the HRs and land a suitable project within the same industry vertical or a different domain within PCS. There are over 22,000 new and 50,000 experienced PCS professionals providing their services to 150 clients aligned with the consultancy. The consultancy needs to maintain the information of every PCS employee focusing on their industry verticals and skills/competencies. All PCS employees should have registered profiles. The consultancy should be updated with the latest profiles of the PCS employees and the vacancies of different job roles on each project and industry vertical. Currently, all PCS employees are registered with PCS and are given a unique identification number. Profile validation is done by the HR experts and requirements are fulfilled by mapping skills and requirements manually. It is important to store employees' profile details, which include their current role, skills, and industry verticals, globally. All CVs/employees' profiles and staffing requirements are stored in a database and as hard copies so that the HR. experts of PCS can refer to them and filter manually.

Project Requirement

For Employers/Project Managers:

* Create an employer profile.
* Register the employer profile.
* Authenticate the employer profile through email ID and password
* Update the employer profile.
* Archive the employer profile.
* Suspend the employer profile.
* Subscribe to certain keywords.
* Search through keyword(s) for skills separated by comma, returning profile match percentage, contact details, IBU details, and supervisor name.
* For Employees:
* Create an employee profile.
* Register the employee profile.
* Authenticate the employee profile through email ID and password.
* Update the employee profile.
* Archive the employee profile.
* Suspend the employee profile.
* View the employee profile.
* For HRs:
* Profile Validation.

Project Analysis

10th December 2020 : Installed and make a database for the project in MySQL[zero errors]

17th December 2020 : Learned about the CRUD Program. In Eclipse.[zero errors]

24th December 2020: Implemented the CRUD Programs and review the performance till then to Co-Ordinator. [zero errors]

29th December 2020: Implemented to connect to the database to manipulate the records through eclipse java code. [zero error]

06th January 2021 : Learned to use the awt package and created some frames for the application.

10th January 2021: Tested the login and authentication and perform operation on the database using the UI.

11th January 2021: Completion of Documentation of PCSApplication.

15th January 2021: Submission of the Project PCSApplication.

Implementation and Diagrams

[3]Java has been one of the most popular programming language for many years. Java is Object Oriented. However it is not considered as pure object oriented as it provides support for primitive data types (like int, char, etc).The Java codes are first compiled into byte code (machine independent code). Then the byte code is run on **J**ava **V**irtual **M**achine (JVM) regardless of the underlying architecture. Java syntax is similar to C/C++. But Java does not provide low level programming functionalities like pointers. Also, Java codes are always written in the form of classes and objects.

Java is used in all kind of applications like Mobile Applications (Android isJava based), desktop applications, web applications, client server applications, enterprise applications and many more.

There are current two sets of Java APIs for graphics programming: AWT (Abstract Windowing Toolkit), Swing AWT API was introduced in JDK 1.0. Most of the AWT components have become obsolete and should be replaced by newer Swing components.Swing API, a much more comprehensive set of

graphics libraries that enhances the AWT, was introduced as part of Java Foundation Classes (JFC) after the release of JDK 1.1. JFC consists of Swing,

Java2D,

1. Accessibility, Internationalization, and Pluggable Look-and-Feel Support APIs. JFC has been integrated into core Java since JDK 1.2.

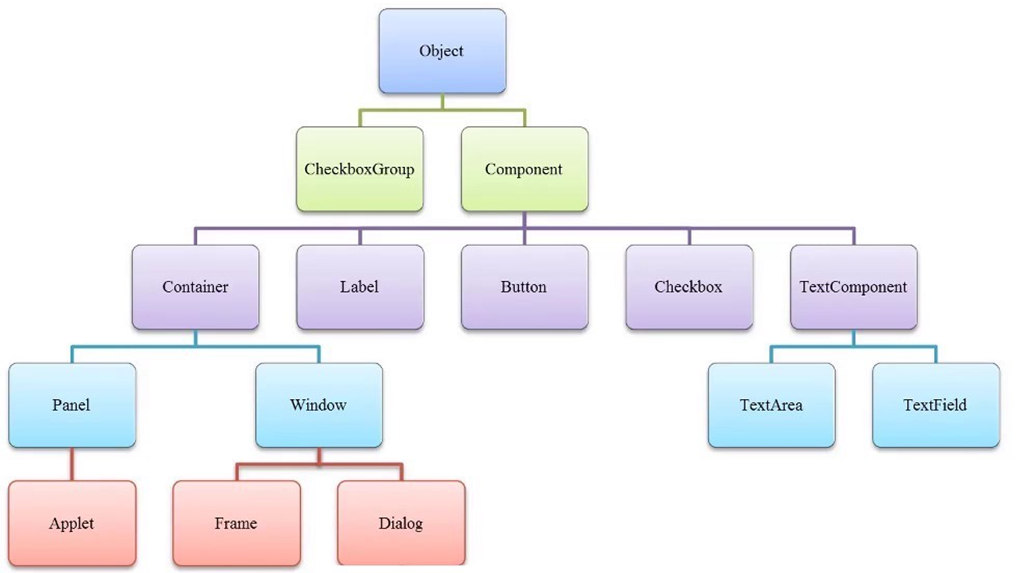
AWT

AWT is huge! It consists of 12 packages of 370 classes (Swing is even bigger, with 18 packages of 737 classes as of JDK 8). Fortunately, only 2 packages - java.awt and java.awt.event - are commonly-used.

1. The java.awt package contains the *core* AWT graphics classes:
   * GUI Component classes, such as Button, TextField, and Label.
   * GUI Container classes, such as Frame and Panel.
   * Layout managers, such as FlowLayout, BorderLayout and GridLayout.
   * Custom graphics classes, such as Graphics, Color and Font.
2. The java.awt.event package supports event handling:
   * Event classes, such as ActionEvent, MouseEvent, KeyEvent and WindowEvent,
   * Event Listener Interfaces, such as ActionListener, MouseListener, MouseMotionListener, KeyListener and WindowListener,
   * Event Listener Adapter classes, such as MouseAdapter, KeyAdapter, and WindowAdapter.

AWT provides a *platform-independent* and *device-independent* interface to develop graphic programs that runs on all platforms, including Windows, Mac OS X, and Unixes.





[4]Figure- AWT Implementation

Code

--Employee.java

**package** model;

**public** **class** Employee {

**private** **int** employeeid;

**private** String firstName;

**private** String lastName;

**private** String email;

**private** String userId;

**private** String password;

**private** String gender;

**private** String role;

**private** String active;

**private** String Skill;

**public** Employee() {

**super**();

}

**public** Employee(**int** employeeid, String firstName, String lastName, String email, String userId, String password,

String gender, String role, String active) {

**super**();

**this**.employeeid = employeeid;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.email = email;

**this**.userId = userId;

**this**.password = password;

**this**.gender = gender;

**this**.role = role;

**this**.active = active;

}

**public** **int** getEmployeeid() {

**return** employeeid;

}

**public** **void** setEmployeeid(**int** employeeid) {

**this**.employeeid = employeeid;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getUserId() {

**return** userId;

}

**public** **void** setUserId(String userId) {

**this**.userId = userId;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

**public** String getRole() {

**return** role;

}

**public** **void** setRole(String role) {

**this**.role = role;

}

**public** String getActive() {

**return** active;

}

**public** **void** setActive(String active) {

**this**.active = active;

}

**public** String getSkill() {

**return** Skill;

}

**public** **void** setSkill(String skill) {

Skill = skill;

}

@Override

**public** String toString() {

**return** "Employee [employeeid=" + employeeid + ", firstName=" + firstName + ", lastName=" + lastName + ", email="

+ email + ", userId=" + userId + ", password=" + password + ", gender=" + gender + ", role=" + role

+ ", active=" + active + "]";

}

**public** Employee(String firstName, String lastName, String email, String userId, String password, String gender,

String role, String active) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.email = email;

**this**.userId = userId;

**this**.password = password;

**this**.gender = gender;

**this**.role = role;

**this**.active = active;

}

}

package controller;

import java.util.List;

import dao.iEmployeeDao;

import daoImpl.iEmployeeDaoImp;

import model.Employee;

public class EmployeeController {

iEmployeeDao empDao=null;

public EmployeeController() {

empDao=new iEmployeeDaoImp();

}

public Employee checkLogin(String userId, String password) {

Employee emp = empDao.checkLogin(userId, password);

return emp;

}

public List<Employee> getAllEmployees(){

List<Employee> empList=empDao.getAllEmployees();

return empList;

}

public void addEmployee(Employee emp) {

empDao.addEmployee(emp);

}

public Employee getEmployeeById(int id) {

Employee emp=empDao.getEmployeeById(id);

return emp;

}

public void updateEmployee(Employee emp) {

empDao.updateEmployee(emp);

}

public void deleteEmployee(int id) {

empDao.deleteEmployee(id);

}

public void deactivateEmployee(Employee emp) {

empDao.deactivateEmployee(emp);

}

public void activateEmployee(Employee emp) {

empDao.activateEmployee(emp);

}

}package dao;

import java.util.\*;

import model.Employee;

public interface iEmployeeDao {

List<Employee> getAllEmployees();

void addEmployee(Employee emp);

Employee getEmployeeById(int id);

void updateEmployee(Employee emp);

void deactivateEmployee(Employee emp);

void activateEmployee(Employee emp);

void deleteEmployee(int id);

Employee checkLogin(String userId, String password);

}

**package** config;

**import** java.sql.\*;

**public** **class** JDBCConnection {

**static** String *url*="jdbc:mysql://localhost:3306/pcsdb";

**static** String *username*="root";

**static** String *password*="Niit@123";

**static** Connection *conn*=**null**;

**public** **static** Connection getDBConnection() {

**try** {

Class.*forName*("com.mysql.jdbc.Driver"); // Loading driver

*conn*=DriverManager.*getConnection*(*url*, *username*, *password*);

}

**catch**(ClassNotFoundException ex) {

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

}

**catch**(SQLException ex) {

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

}

**return** *conn*;

}

}

package daoImpl;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import config.JDBCConnection;

import dao.iEmployeeDao;

import model.Employee;

public class iEmployeeDaoImp implements iEmployeeDao {

Connection conn=null;

public iEmployeeDaoImp() {

conn=JDBCConnection.getDBConnection();

}

public Employee checkLogin(String userId,String password) {

Employee emp = new Employee();

try {

String query ="select \* from Employee where userId=? and password=?";

PreparedStatement pst = conn.prepareStatement("");

pst.setString(1, userId);

pst.setString(2, password);

ResultSet rst=pst.executeQuery();

if(rst!=null) {

if(rst.next()) {

emp.setEmployeeid(rst.getInt(1));

emp.setFirstName(rst.getString(2));

emp.setLastName(rst.getString(3));

emp.setEmail(rst.getString(3));

emp.setUserId(rst.getString(3));

emp.setPassword(rst.getString(3));

emp.setRole(rst.getString(3));

emp.setGender(rst.getString(3));

emp.setActive(rst.getString(3));

}

}

}catch(SQLException ex) {

System.out.println(ex);

}

return emp;

}

@Override

public List<Employee> getAllEmployees() {

List<Employee> empList=new ArrayList<Employee>();

try {

String query="select \* from Employee";

Statement stmt=conn.createStatement();

ResultSet rst=stmt.executeQuery(query);

if(rst!=null) {

while(rst.next()) {

Employee emp=new Employee();

emp.setEmployeeid(rst.getInt(1));

emp.setFirstName(rst.getString(2));

emp.setLastName(rst.getString(3));

emp.setEmail(rst.getString(4));

emp.setUserId(rst.getString(5));

emp.setPassword(rst.getString(6));

emp.setRole(rst.getString(7));

emp.setGender(rst.getString(8));

emp.setActive(rst.getString(9));

empList.add(emp);

}

}

}catch(SQLException ex) {

System.out.println(ex);

}

return empList;

}

@Override

public void addEmployee(Employee emp) {

try {

String query="insert into Employee(FirstName, LastName, Email, UserId, Password, Role, Gender, Active) values(?,?,?,?,?,?,?,?)";

PreparedStatement pst=conn.prepareStatement(query);

pst.setString(1,emp.getFirstName());

pst.setString(2,emp.getLastName());

pst.setString(3,emp.getEmail());

pst.setString(4,emp.getUserId());

pst.setString(5,emp.getPassword());

pst.setString(6,emp.getRole());

pst.setString(7,emp.getGender());

pst.setString(8,emp.getActive());

int i=pst.executeUpdate();

if(i==1) {

System.out.println("1 record inserted...");

}

else {

System.out.println("Insertion failed...");

}

}catch(SQLException ex) {

System.out.println(ex);

}

}

@Override

public Employee getEmployeeById(int id) {

Employee emp=new Employee();

try {

String query="select \* from Employee where EmployeeId=?";

PreparedStatement pst=conn.prepareStatement(query);

pst.setInt(1,id);

ResultSet rst=pst.executeQuery();

if(rst!=null) {

while(rst.next()) {

emp.setEmployeeid(rst.getInt(1));

emp.setFirstName(rst.getString(2));

emp.setLastName(rst.getString(3));

emp.setEmail(rst.getString(4));

emp.setUserId(rst.getString(5));

emp.setPassword(rst.getString(6));

emp.setRole(rst.getString(7));

emp.setGender(rst.getString(8));

emp.setActive(rst.getString(9));

}

}

}catch(SQLException ex) {

System.out.println(ex);

}

return emp;

}

@Override

public void updateEmployee(Employee emp) {

try {

String query = "UPDATE Employee SET password=?, email=? WHERE EmployeeId=?";

PreparedStatement statement = conn.prepareStatement(query);

statement.setString(1, emp.getPassword());

statement.setString(2, emp.getEmail());

statement.setInt(3,emp.getEmployeeid());

int rowsUpdated = statement.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("An existing user was updated successfully!");

}

else {

System.out.println("updation failed...");

}

}catch(SQLException ex) {

System.out.println(ex);

}

}

@Override

public void deactivateEmployee(Employee emp) {

try {

//creating PreparedStatement object by passing query string

PreparedStatement pst=conn.prepareStatement("update Employee set Active=? where EmployeeId=?");

pst.setString(1, "Deactive");

pst.setInt(2, emp.getEmployeeid());

int i=pst.executeUpdate();

if(i==1){

System.out.println("Employee deactivated...");

}

else {

System.out.println("updation failed...");

}

}

catch(SQLException ex) {

System.out.println(ex.getMessage());

}

}

@Override

public void activateEmployee(Employee emp) {

try {

//creating PreparedStatement object by passing query string

PreparedStatement pst=conn.prepareStatement("update Employee set Active=? where EmployeeId=?");

pst.setString(1, "Active");

pst.setInt(2, emp.getEmployeeid());

int i=pst.executeUpdate();

if(i==1){

System.out.println("Employee Activated...");

}

else {

System.out.println("updation failed...");

}

}

catch(SQLException ex) {

System.out.println(ex.getMessage());

}

}

@Override

public void deleteEmployee(int id) {

try {

//creating PreparedStatement object by passing query string

PreparedStatement pst=conn.prepareStatement("delete from Employee where EmployeeId=?");

pst.setInt(1, id);

int i=pst.executeUpdate();

if(i==1){

System.out.println("Employee deleted...");

}

else {

System.out.println("deletion failed...");

}

}

catch(SQLException ex) {

System.out.println(ex.getMessage());

}

}

}

package excDao;

import java.util.Iterator;

import java.util.List;

import java.io.\*;

import controller.EmployeeController;

import model.Employee;

public class EmployeeExec {

EmployeeController empController=null;

public EmployeeExec() {

empController=new EmployeeController();

}

public void getAllEmployees() {

List<Employee> empList=empController.getAllEmployees();

for(Employee emp:empList) {

System.out.println(emp);

}

}

public void getEmployeeById() throws NumberFormatException, IOException {

int id=0;

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter employee id whose record you want to view:");

id=Integer.parseInt(reader.readLine());

}

catch(IOException ex){

System.out.println(ex);

}

Employee emp=empController.getEmployeeById(id);

System.out.println(emp);

}

public void addEmployee() {

Employee emp=new Employee();

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter Employee Detail:");

System.out.println("First Name:");

emp.setFirstName(reader.readLine());

System.out.println("Last Name:");

emp.setLastName(reader.readLine());

System.out.println("Email:");

emp.setEmail(reader.readLine());

System.out.println("User Id:");

emp.setUserId(reader.readLine());

System.out.println("Password:");

emp.setPassword(reader.readLine());

System.out.println("Role:");

String role=reader.readLine();

emp.setRole(role);

System.out.println("Gender:");

emp.setGender(reader.readLine());

if(role.equals("HRA")) {

emp.setActive("Active");

}

else {

emp.setActive("Deactive");

}

}catch(IOException ex){

System.out.println(ex);

}

empController.addEmployee(emp);

}

public void updateEmployee() {

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

int id;

String email, password, confirmpassword;

System.out.println("Enter EmployeeId whose record you want to update:");

id=Integer.parseInt(reader.readLine());

Employee emp=empController.getEmployeeById(id);

System.out.println("Enter your new email:");

email=reader.readLine();

System.out.println("Enter your new password:");

password=reader.readLine();

System.out.println("Re-enter same password to confirm:");

confirmpassword=reader.readLine();

if(password.equals(confirmpassword)) {

emp.setPassword(password);

emp.setEmail(email);

empController.updateEmployee(emp);

}

else {

System.out.println("Sorry! you have entered different password!");

}

}

catch(IOException ex) {

System.out.println(ex.getMessage());

}

}

public void deactiveEmployee() {

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

int id;

System.out.println("Enter EmployeeId whose record you want to deactivate:");

id=Integer.parseInt(reader.readLine());

Employee emp=empController.getEmployeeById(id);

empController.deactivateEmployee(emp);

}

catch(IOException ex) {

System.out.println(ex.getMessage());

}

}

public void activeEmployee() {

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

int id;

System.out.println("Enter EmployeeId whose record you want to activate:");

id=Integer.parseInt(reader.readLine());

Employee emp=empController.getEmployeeById(id);

empController.activateEmployee(emp);

}

catch(IOException ex) {

System.out.println(ex.getMessage());

}

}

public void deleteEmployee() {

try {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

int id;

System.out.println("Enter EmployeeId whose record you want to delete:");

id=Integer.parseInt(reader.readLine());

empController.deleteEmployee(id);

}

catch(IOException ex) {

System.out.println(ex.getMessage());

}

}

}package entry;

import java.util.Scanner;

import java.io.IOException;

import java.sql.SQLException;

import config.JDBCConnection;

import excDao.EmployeeExec;

public class TestApp {

public void testConnection() {

try{

if(JDBCConnection.getDBConnection().isClosed()) {

System.out.println("Connection is closed");

}

else {

System.out.println("Connection is opened");

}

}catch(SQLException e) {

e.printStackTrace();

}

}

public void processMenu() {

EmployeeExec obj=new EmployeeExec();

Scanner sc=new Scanner(System.in);

int option=0;

char ch='y';

do {

System.out.println("----------CRUD Operation-----------");

System.out.println("1. View all Employees");

System.out.println("2. View single Employee");

System.out.println("3. Add Employee");

System.out.println("4. Update Employee");

System.out.println("5. Delete Employee");

System.out.println("6. Deactivate Employee");

System.out.println("7. Activate Employee");

System.out.println("8. Quit");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Enter your choice:");

option=sc.nextInt();

switch(option) {

case 1:

obj.getAllEmployees();

break;

case 2:

obj.getEmployeeById();

break;

case 3:

obj.addEmployee();

break;

case 4:

obj.updateEmployee();

break;

case 5:

obj.deleteEmployee();

break;

case 6:

obj.deactiveEmployee();

break;

case 7:

obj.activeEmployee();

break;

case 8:

System.exit(0);

break;

default:

System.out.println("Wrong input!");

break;

}

System.out.println("Do you want to continue(y/n)?");

ch=sc.next().charAt(0);

}while(ch=='y' || ch=='Y');

}

public static void main(String[] args) {

TestApp test=new TestApp();

//test.testConnection();

test.processMenu();

}

}

package view;

import java.awt.Container;

import javax.swing.JFrame;

public class ActivateFrame extends JFrame{

Container container;

public ActivateFrame() {

container=getContentPane();

setLayoutManager();

setLocationAndSize();

addComponentsToContainer();

this.setTitle("ACTIVATE EMPLOYEE");

this.setVisible(true);

this.setBounds(10, 10, 500, 700);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setResizable(false); // fixed size frame

}

public void setLayoutManager() {

container.setLayout(null);

}

public void setLocationAndSize() {

}

public void addComponentsToContainer() {

}

public void callDispose() {

this.dispose();

}

}

package view;

import java.awt.Container;

import javax.swing.JFrame;

public class EmpHomeFrame extends JFrame{

Container container;

public EmpHomeFrame() {

container = getContentPane();

setLayoutManager();

setLocationAndSize();

addComponentsToContainer();

this.setTitle("EMP HOME");

this.setVisible(true);

this.setBounds(10,10,600,700);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setResizable(false);//fixed size frame

}

public void setLayoutManager() {

container.setLayout(null);

}

public void setLocationAndSize() {

}

public void addComponentsToContainer() {

}

public void callDispose() {

this.dispose();

}

}

**package** view;

**import** java.awt.Container;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** model.Employee;

**import** javax.swing.\*;

**import** controller.EmployeeController;

**public** **class** HraHomeFrame **extends** JFrame{

EmployeeController empController=**null**;

Container container;

JButton bLogout, bActivate, bViewAllEmployee, bAddSkill;

**public** HraHomeFrame() {

container=getContentPane();

bLogout=**new** JButton("LOGOUT");

empController=**new** EmployeeController();

bLogout.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

callDispose();

**new** LoginFrame();

}

});

bActivate=**new** JButton("ACTIVATE USERS");

bActivate.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

callDispose();

**new** ActivateFrame();

}

});

bViewAllEmployee=**new** JButton("VIEW ALL EMPLOYEES");

bViewAllEmployee.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

callDispose();

empController.getAllEmployees();

}

});

bAddSkill=**new** JButton("ADD SKILL");

bAddSkill.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

callDispose();

BufferedReader reader=**new** BufferedReader(**new** InputStreamReader(System.***in***));

**try** {

Employee emp = **null**;

empController.addEmployee(emp);

emp.setSkill(reader.readLine());

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

});

setLayoutManager();

setLocationAndSize();

addComponentsToContainer();

**this**.setTitle("HRA HOME");

**this**.setVisible(**true**);

**this**.setBounds(10, 10, 500, 700);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.setResizable(**false**); // fixed size frame

}

**public** **void** setLayoutManager() {

container.setLayout(**null**);

}

**public** **void** setLocationAndSize() {

bActivate.setBounds(100, 100, 300, 30);

bViewAllEmployee.setBounds(100, 140, 300, 30);

bAddSkill.setBounds(100,180 , 300, 30);

bLogout.setBounds(100, 220, 300, 30);

}

**public** **void** addComponentsToContainer() {

container.add(bActivate);

container.add(bViewAllEmployee);

container.add(bAddSkill);

container.add(bLogout);

}

**public** **void** callDispose() {

**this**.dispose();

}

}

package entry;

import view.ActivateFrame;

import view.LoginFrame;

public class RunProject {

public static void main(String[] args) {

new LoginFrame();

//new ActivateFrame();

}

}

SQL CODE

use pcsdb;

Create table Employee(

EmployeeId int auto\_increment,

FirstName varchar(30) not null,

LastName varchar(30) not null,

Email varchar(30) not null,

UserId varchar(30) not null,

Password varchar(20) not null,

Role varchar(3) not null,

Gender varchar(10) not null,

Active varchar(10) not null,

primary key(EmployeeId)

);

select \* from Employee

Create table Skill(

SkillID int auto\_increment,

SkillName varchar(30) not null,

SkillDescription varchar(60) not null,

Active varchar(10) not null,

primary key(SkillID)

);

select \* from Job

Create table Job(

JobID int auto\_increment,

JobTitle varchar(30) not null,

JobDescription varchar(30) not null,

CompanyName varchar(30) not null,

Location varchar(30) not null,

KeySkill varchar(30) not null,

Salary int not null,

Experience int not null,

Active varchar(10) not null,

primary key(JobID)

);

create table EmpSkill(

ESID int auto\_increment,

EmployeeId int,

SkillId int,

Experience int,

primary key(ESID)

);

create table EmpJob(

EJID int auto\_increment,

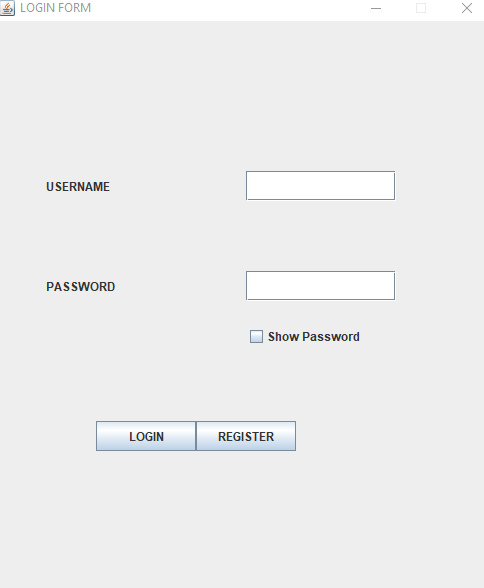
EmployeeID int,

primary key(EJID)

);

Output

Login :



Challenges

* The initial steps to connect the MySQL to the Eclipse together.

References

[1] NIIT Project Guide

[2] NIIT Project Guide

[3] Wikipedia

[4] Google Images