**PCS AUTOMATION**

Developed By

Name: ARPAN BHADRA Reg. No: R201190200105

# NIIT

### PCS AUTOMATION

Batch Code : S210167

Start Date : 12.11.2020

End Date : 12.12.2020

Name of the Coordinator : LOPAMUDRA BERA

Name of the Developer : ARPAN BHADRA

Date of Submission : 12.12.2020

**NIIT**

###### CERTIFICATE

This is to certify that this report, titled **PCS AUTOMATION** embodies the original work done by **Mr. ARPAN BHADRA**, in partial fulfilment of his course requirement at NIIT.

Coordinator : LOPAMUDRA BERA

### ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **LOPAMUDRA BERA** as well as to the education institute **NIIT** who gave me the golden opportunity to do this wonderful project on the topic **PROFESSIONET CONSULTANCY SERVICES**, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.  
Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

### ABSTRACT

This project manages the entire process of allocating project to existing employees as per his/her skillset working in that company

It’s also provides single window system to Employee, HR and Project Managers of a company to cater the skill specific requirements emerging in Projects.

### CONFIGURATION

Hardware:8GB RAM,2GB GRAPHICS CARD,1 TB HARD DISK, INTEL I3 8TH GENERATION PROCESSOR

Operating System: WINDOWS 10

Software: ECLIPSE 2019, MY SQL

### TABLE OF CONTENTS

Chapter 1- Introduction

1.0 Aim

1.1 Objectives

1.2 Case Study

Chapter 2- Project Requirement Specification

2.0 Literature Research

2.1 Statement of Requirements

2.2 Vision Document

2.3 Project Life Cycle Model

Chapter 3- Project Analysis

3.0 Project Plan

3.1 Risk Management Plan

3.2 Weekly Status Report

3.3 Storyboard

3.4 System Architecture

3.5 Business Process Model

3.6 Software Requirement Specification

3.7 High Level Use Case Diagrams

Chapter 4- Project Design

4.0 Low Level Use Case Diagrams

4.1 User Interface Design

4.2 Systems Input and Output Design

4.3 Database Structure

4.4 Entity Relationship Diagram

4.5 Data Model

4.6 Class Diagrams

4.7 Activity Diagrams

4.8 Sequence Diagrams

4.9 User-centered Interface Designs

4.10 Personas

4.11 Paper Prototypes

4.12 Web Interfaces

4.13 Change Request Form

Chapter 5- Project Implementation

5.0 Acceptance Plan

5.1 System Support Plan

Chapter 6- Project Testing

6.0 Test Plan

6.1 Test Cases

Chapter 7- Project Deployment

7.0 Training Plan

7.1 Component and Deployment Diagrams

7.2 Deployment Plan

7.3 Maintenance Request

7.4 Service Level Agreement

7.5 Maintenance Plan

Challenges

Observations

References

Appendix

### AIM

To create an application that operates via online recruiting website. Which stored the Records in Data Base and Also retrieving it.

### OBJECTIVES

Following are the objectives to be achieved through Skill Mapping Application

* User registration
* Skill map
* Job Postings
* Profile Validation
* Recruitment

These objectives are to be automated which were previously done manually.

### CASE STUDY

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.

PCS offers an offline platform for their employees to share their profiles to initiate internal job posting process with the expert Human Resources consultants.

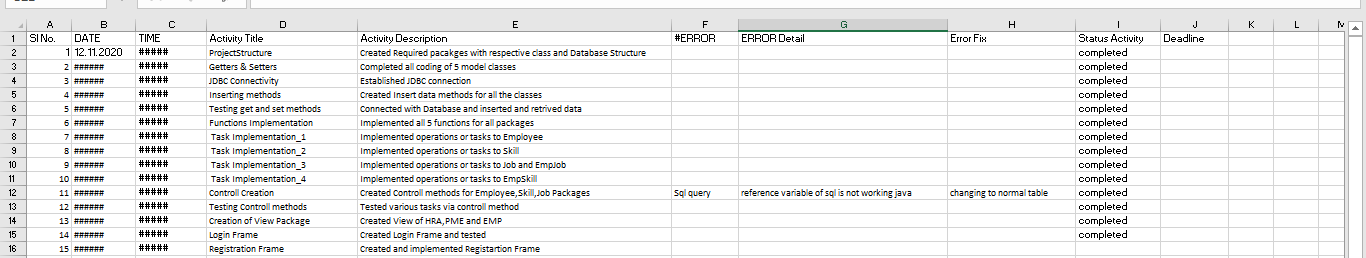
There are over 22,000 new and 50,000 experienced PCS professionals providing their services to 150 clients aligned with the consultancy. The consultancy needs to maintain the information of every PCS employee focusing on their industry vertical. All PCS employees are registered with PCS and are given a unique identification number. Profile validation is done by the HR experts and requirements are full filled by mapping skills and requirements manually.

The Consultancy wants to introduce automation in their Internal Job Posting (IJP) selection and recruitment process so that the potential PCS employees and HR department have an online platform based on a skill map engine to connect directly with each other and aid their job search within the company. HR and PM can also post their jobs and refine their search by using keywords that matches the profiles.

### PROJECT PLAN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Phases | W1 | W2 | W3 | W4 | W5 | W6 | W7 |
| Recruitment Gathering |  |  |  |  |  |  |  |
| High Level Design |  |  |  |  |  |  |  |
| Low Level Design |  |  |  |  |  |  |  |
| UI Design |  |  |  |  |  |  |  |
| Code Development |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |
| Roll Out |  |  |  |  |  |  |  |

### WEEKLY STATUS



### STATEMENT OF REQUIREMENTS

PROJECT OBJECTIVES

|  |  |
| --- | --- |
| Title | PCS Automation |
| Subtitle | Employee Management System |
| Author(s) | Arpan Bhadra |
| Author’s E-mail | arpan.kb@gmail.com |
| Author’s Phone | 9804839407 |
| Description | Automated Consultancy Service |
| Version | 1.00 |

**About Your Company**

Professional Consultancy Services (PCS) is a business consultancy which provides a wide range of business services to clients.

**Need for Process Automation**

* Adds consistency and quality to recruitment
* Improves the productivity of HR team
* Saves time by easing the workload
* Enables organizations to find the right talent

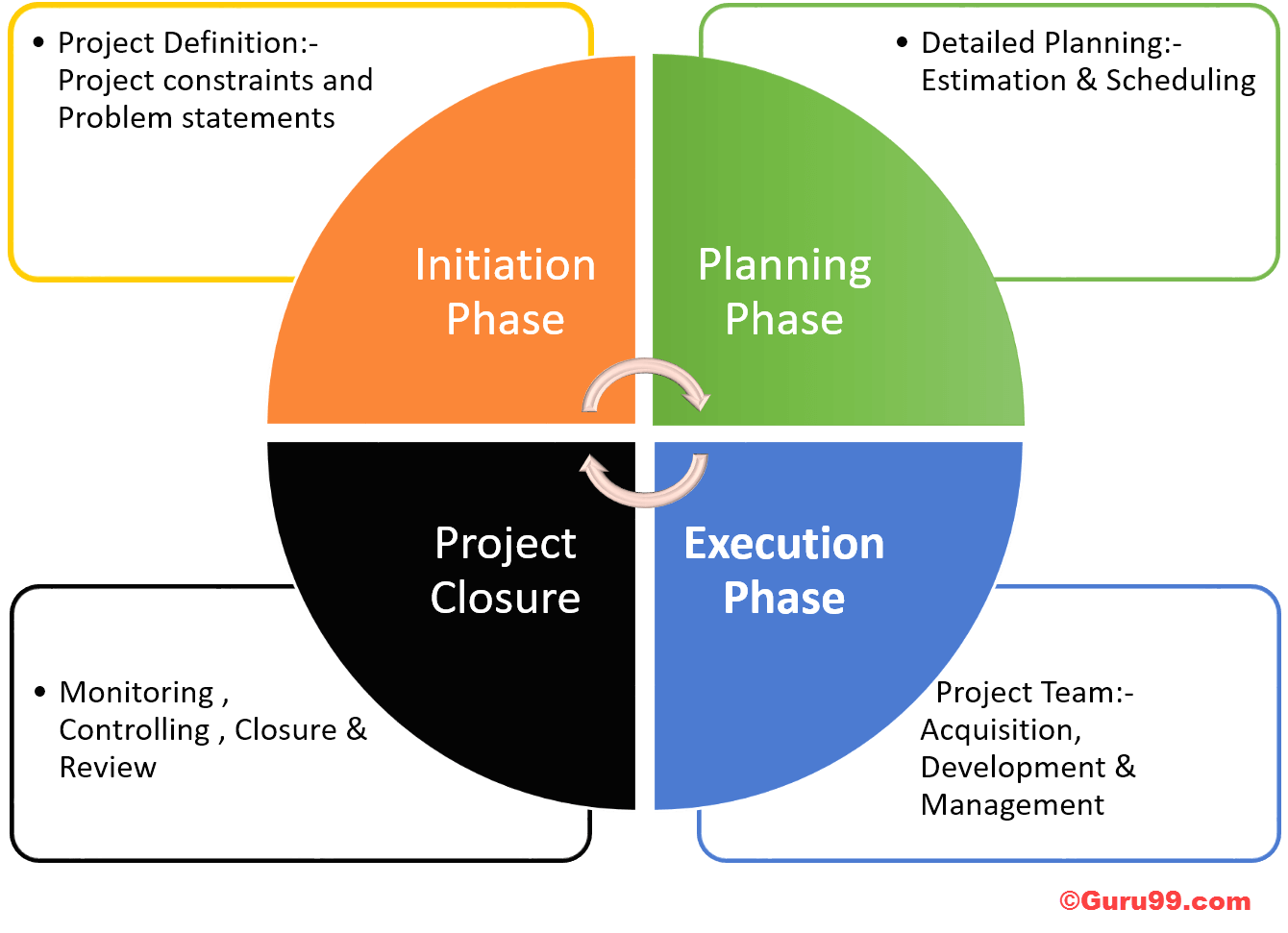
**Software Requirements**

* JRE
* ECLIPSE
* MYSQL

**Benefits**

* Improves the Recruitment Qualities.
* Pre-Screening of Candidates.

### PROJECT LIFE CYCLE MODEL



**The Initiation Phase:** The initiation phase aims to define and authorize the project.

**Vision: Automate the Selection Process**

**The Planning Phase:** The purpose of this phase is to lay down a detailed strategy of how the project has to be performed and how to make it a success.

Strategic Planning- overall approach to the project

Implementation Planning-ways to apply the decisions

**The Execution Phase:** In this phase, the decisions and activities defined during the planning phase are implemented.

**The Termination Phase:** This is the last phase of any project, and it marks the official closure of the project.

### SYSTEM ARCHITECTURE

1. Tier Architecture
2. A **Presentation Layer** that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, etc.
3. An **Application Layer** that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, etc.
4. A **Data Layer** which is a database management system that provides access to application data. This could be MSSQL, MySQL, Oracle, or PostgreSQL, Mongo, etc.

**GUI Component**

(Login,Registration,HRA,EMP,PME)

**Presentation Layer**

**DATABASE**

(Tables-Emp,Job,

Skill,EmpSkill

EmpJob)

**Code**

Controller

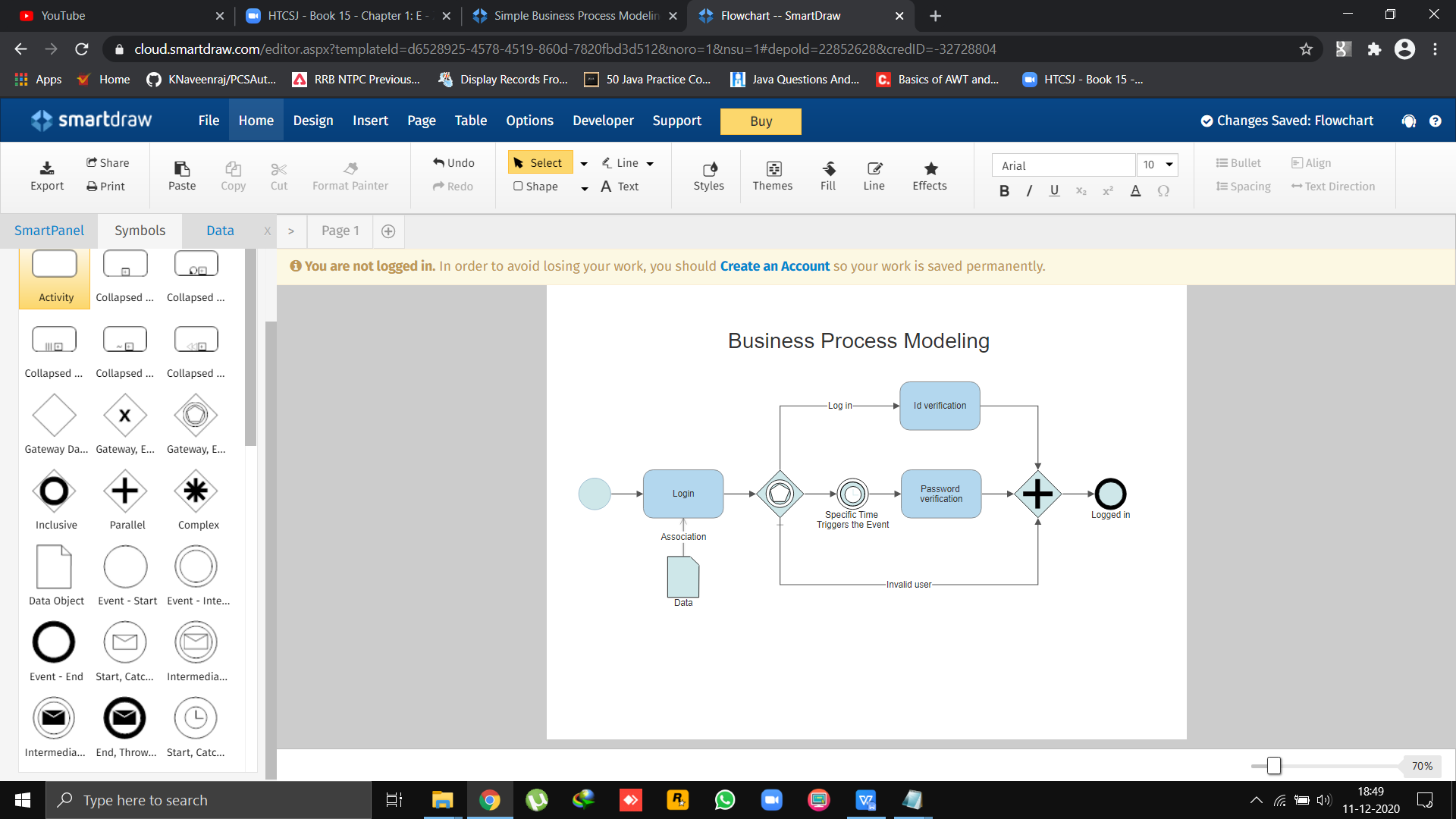
Gui Code

**JDBC Connectivity**

**JRE**

**Application Layer**  **Data Access Layer**

### BUSINESS PROCESS MODEL



Business process modelling (or) process modelling, is the analytical representation or put simply an illustration of an organization’s business processes. Modelling processes is a critical component for effective [business process management](https://kissflow.com/bpm/business-process-management-overview/).

Bbenefits of business process modeling:

* Gives everyone a clear understanding of how the process works
* Provides consistency and controls the process
* Identifies and eliminates redundancies and inefficiencies
* Sets a clear starting and ending to the process

##### SOFTWARE REQUIREMNETS SPECIFICATION

Table of Contents

1. Introduction
   1. Purpose
   2. Scope
   3. Definitions, acronyms and abbreviations
   4. References
   5. Overview
2. General Description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumption and dependencies

1. Specific Requirements

3.1 Functional requirements

3.2 Non-functional requirements

3.3 External interface requirements

3.4 Performance requirements

3.5 Design constraints

3.6 Attributes

3.7 Other requirements

1. Appendices
2. Index

**HIGH LEVEL USER CASE DIAGRAMS**

Login

Registration

**Role Check**

HRA

EMP

PME

PME

EMP

HRA

New user

Registration

View skill-wise Employee list

View All available skills

Add Job

View all jobs

View skill wise jobs

View Employee list who applied for job

Deactivate Job

Logout

View profile

Update Profile

Update Skill

Apply Job

Logout

Activate employee

Deactivate employee

View all employees

View Selective employees

Add Skill

View all skills

Deactivate skill

Logout

GUI CODE

DATABASE

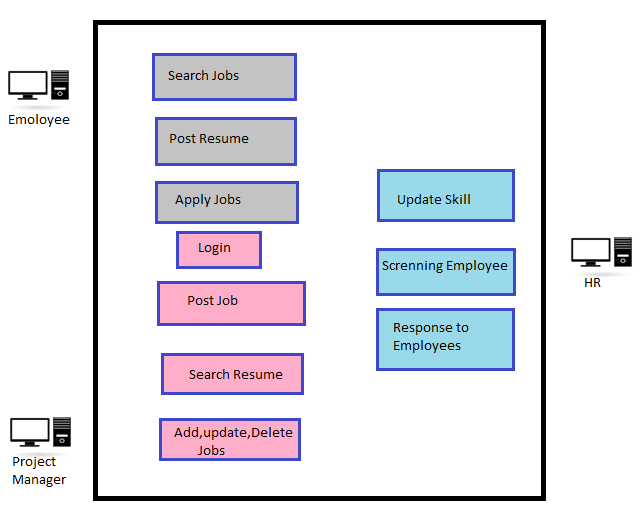
JDBC

Operation

CODE

CODE

#### **LOW LEVEL USER CASE DIAGRAMS**



**EMPHOME**

* Employee can view his/her own information
* Employee can request to HR for Updating his/her information
* Employee apply for job assigned by the PME

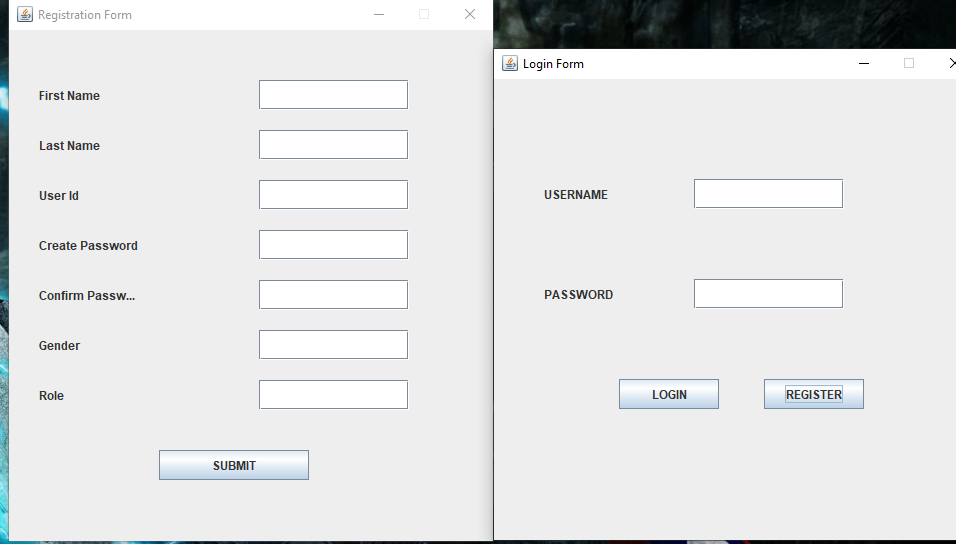
**PMEHOME**

* PME can view Skill of the employee
* PME can add a new job for employee
* PME can view all project jobs

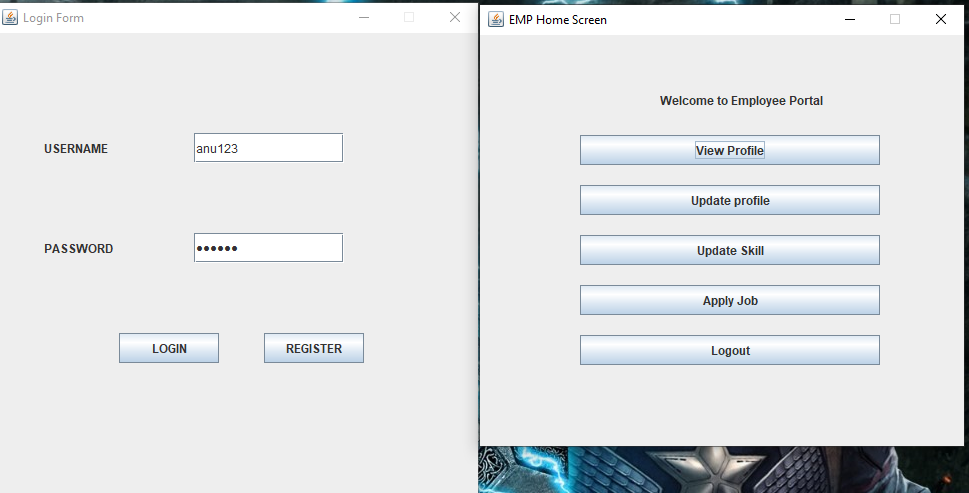
**HREHOME**

* HR can view employee information
* HR can update employee information
* HR can update requested skills of the employee

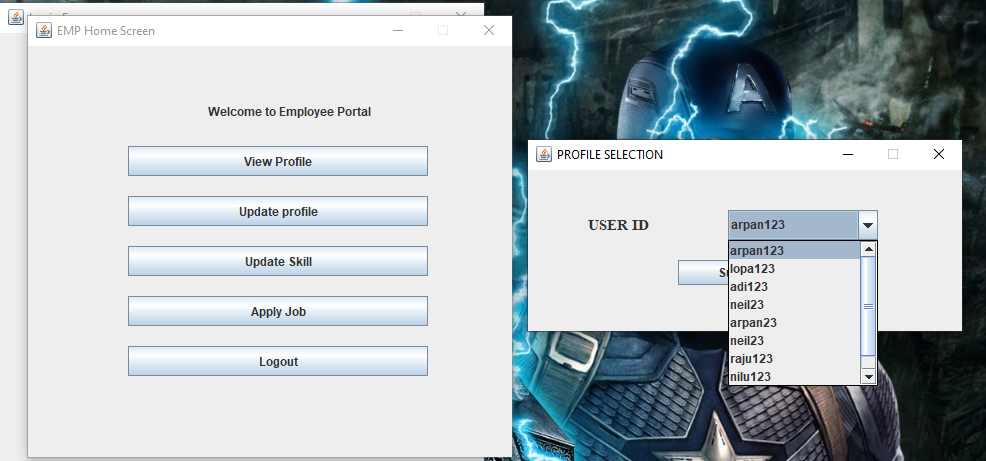
USER INTERFACE DESIGN



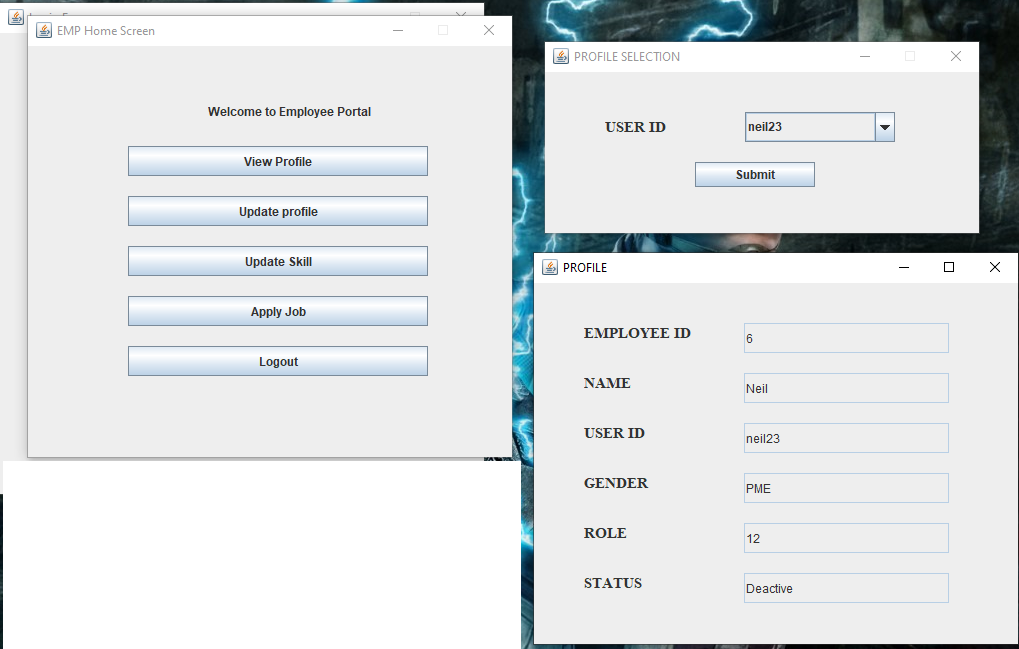
**Login and Registration Frame**

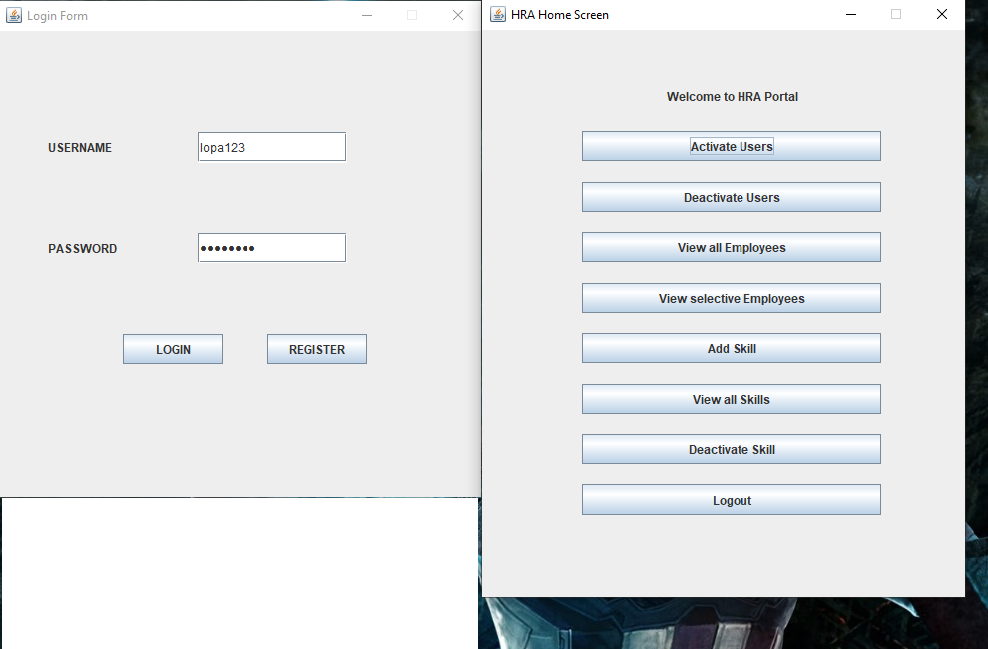


**EMPHome**

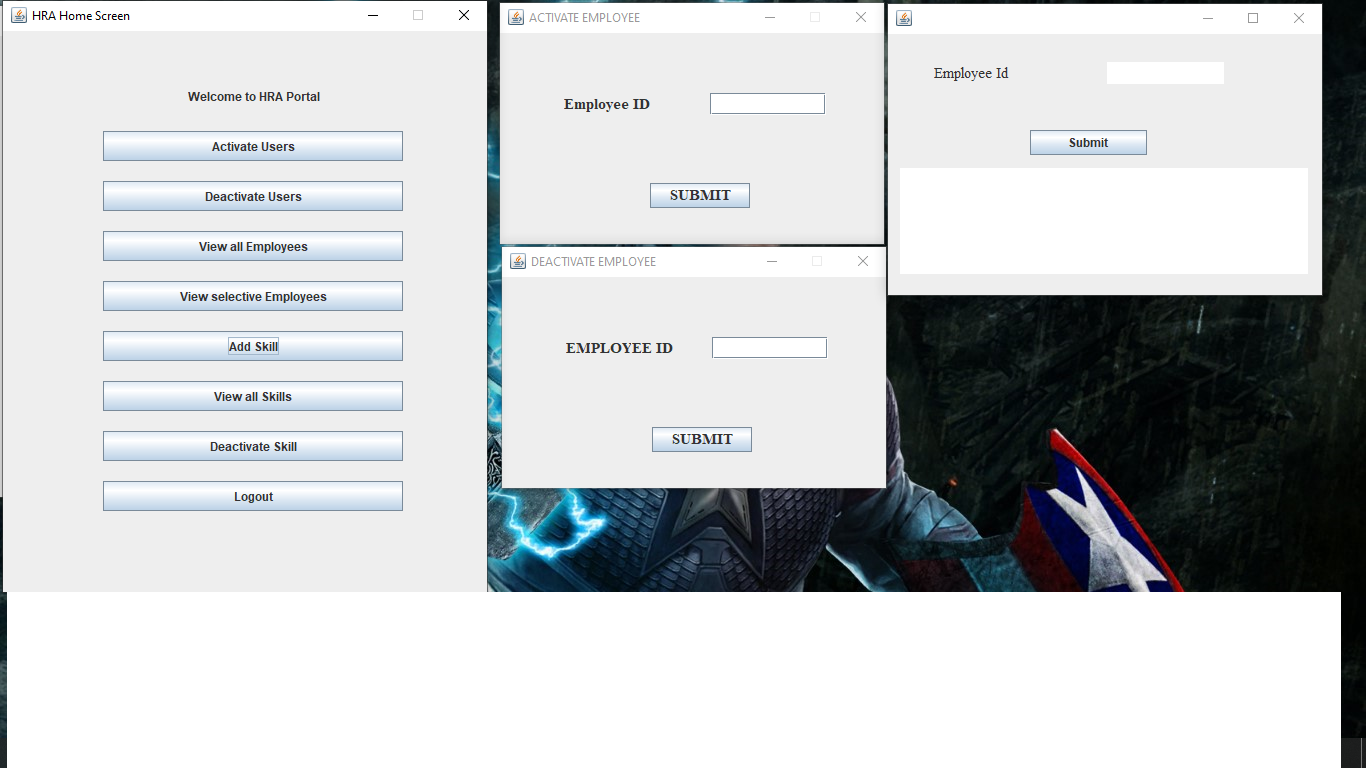


**View Profile**

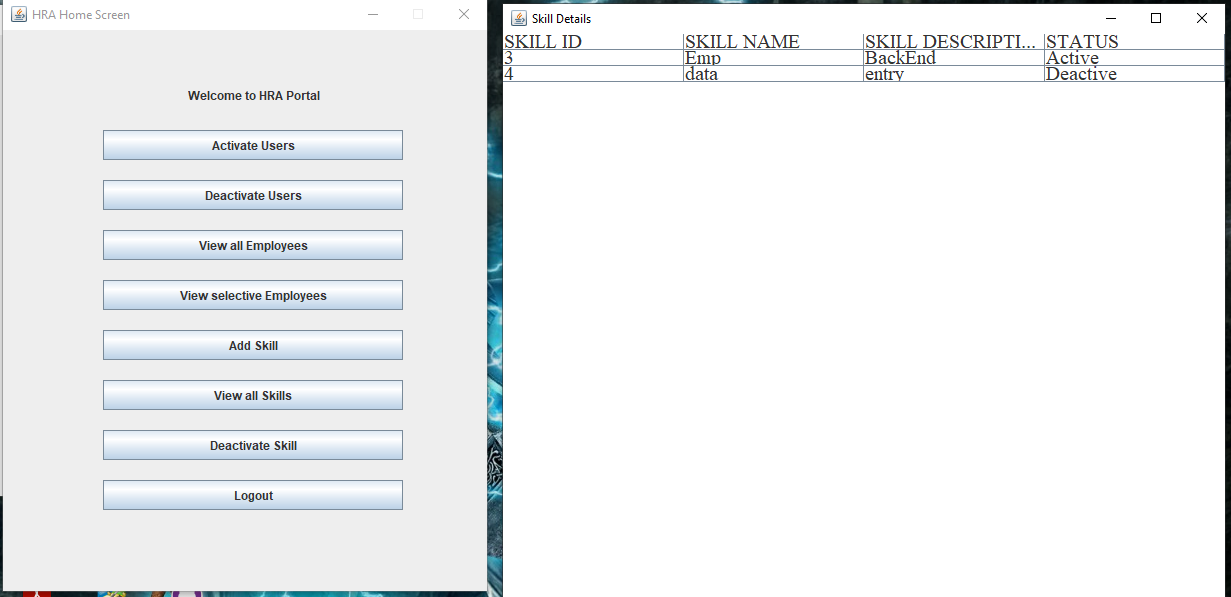


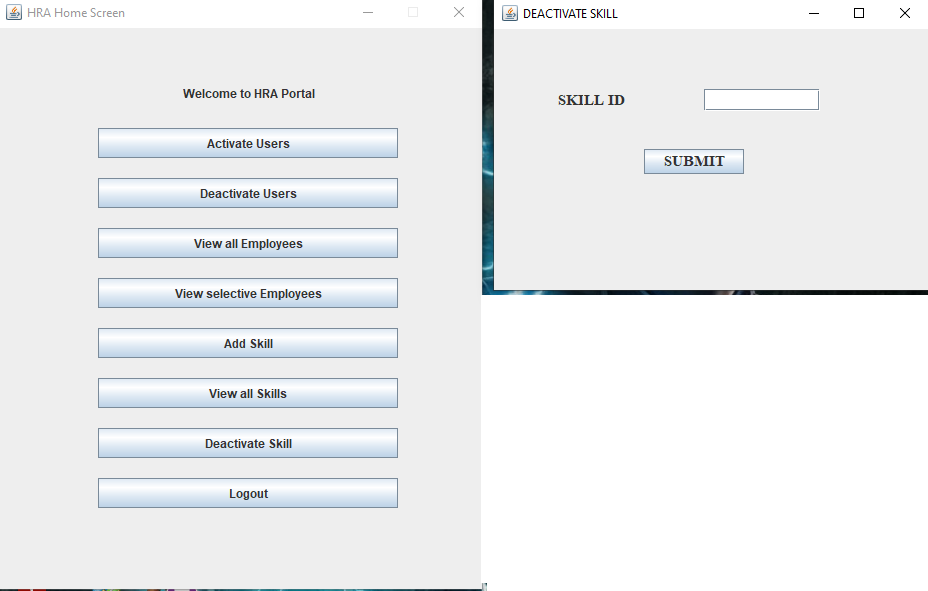


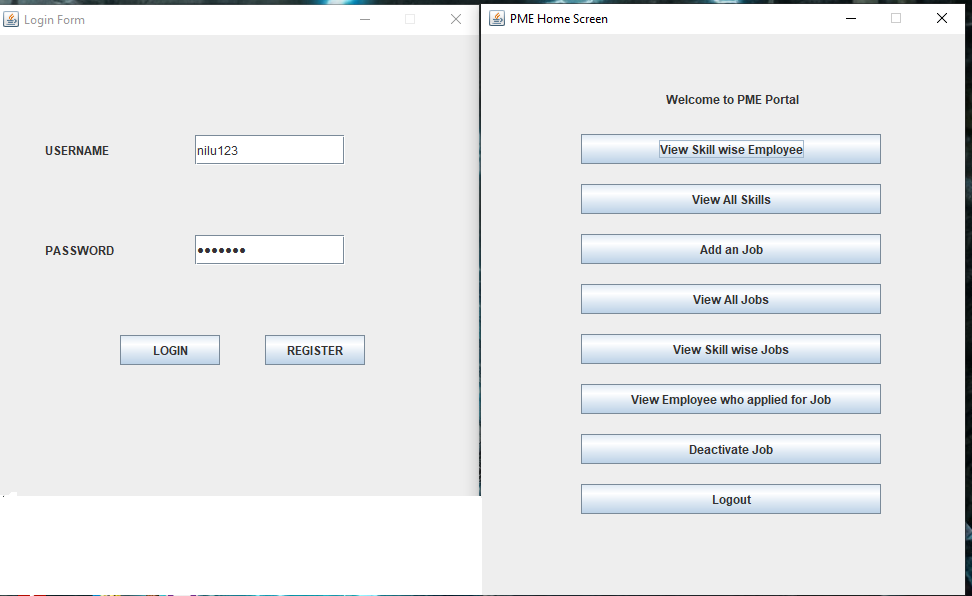
**HRAHome**



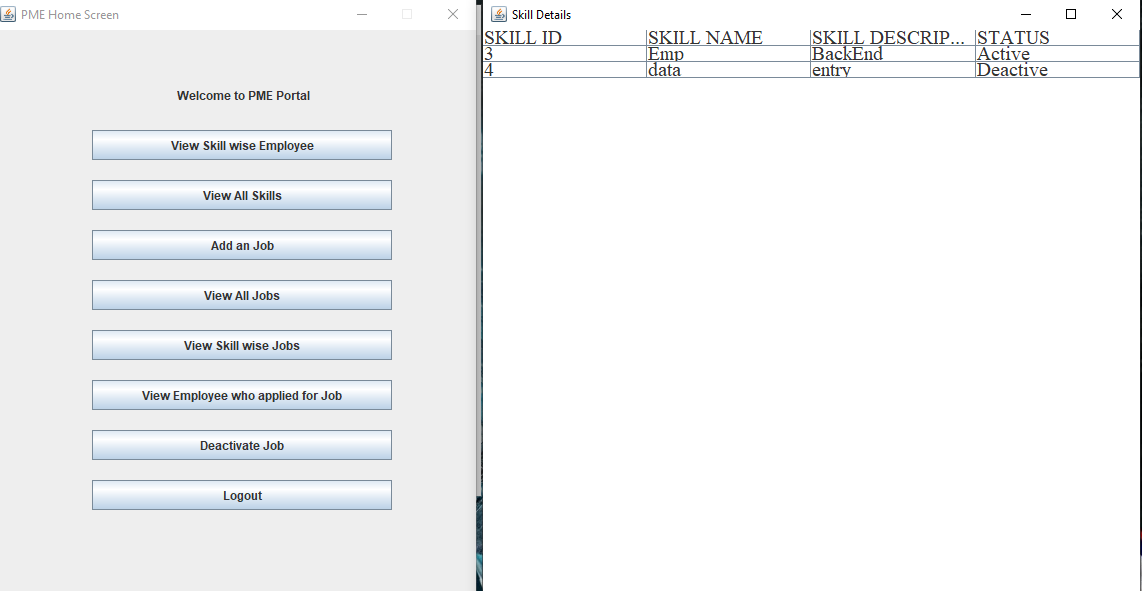
### 

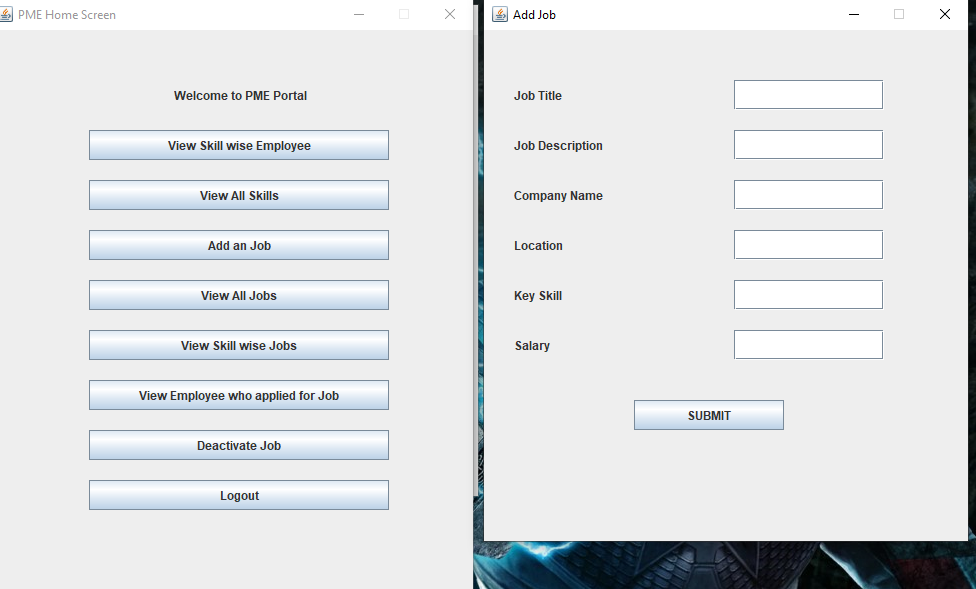


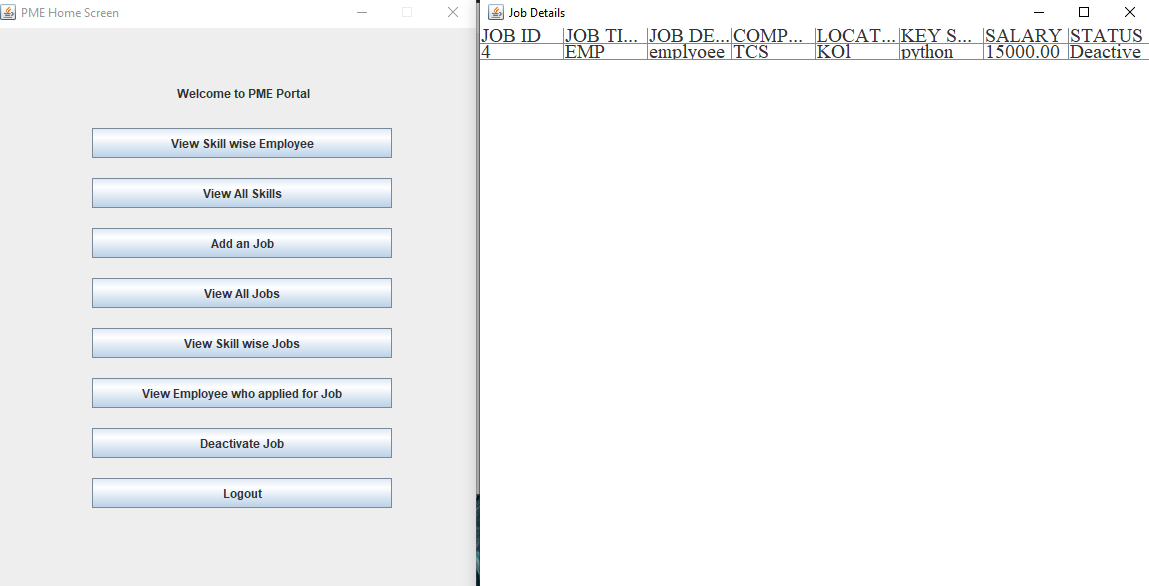


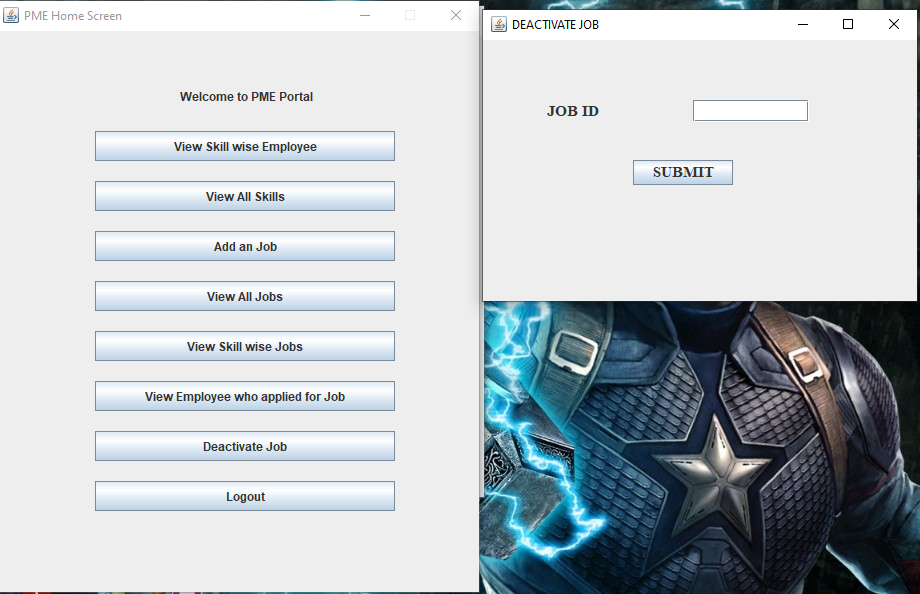


**PMEHome**

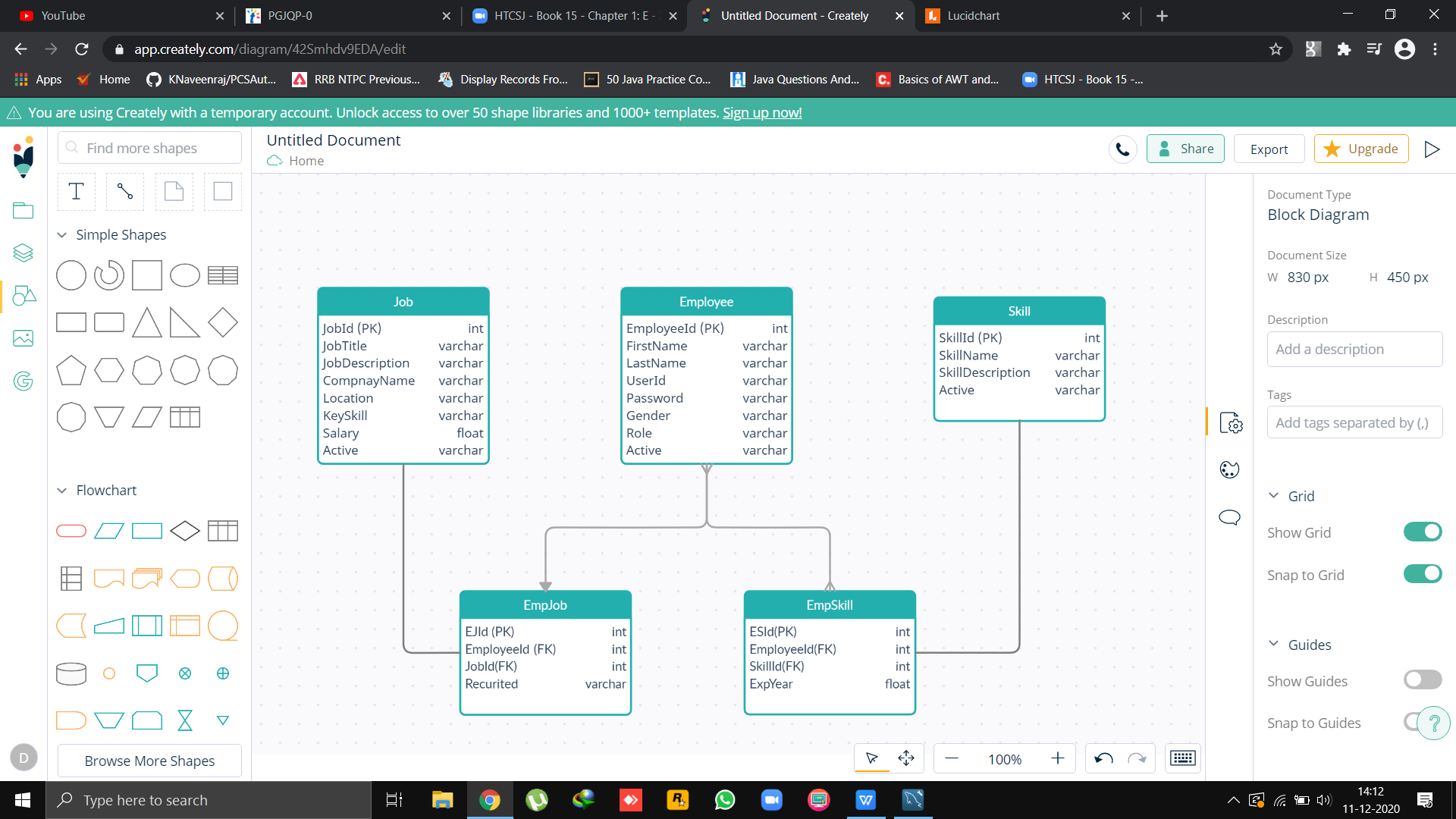








### DATABASE STRUCTURE



//SQL Script

Create pcsdb Data Base in mysql

use pcsdb;

-- Creating Table:: Employee

create table Employee(

empId int auto\_increment,

FirstName varchar(30) not null,

LastName 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(empId)

);

-- Creating Table:: empjob

CREATE TABLE EmpJob (

create table empJob(

EJId int auto\_increment ,

EmployeeId int not null ,

JobId int not null,

Recruited varchar(10)not null,

primary key(EJId)

);

);

-- Creating Table:: empskill

create table empSkill(

ESId int auto\_increment ,

EmployeeId int not null ,

SkillId int not null,

ExpYear int not null,

primary key(ESID)

);

-- Creating Table:: job

create table job(

jobId int auto\_increment,

jobTitle varchar(10) not null,

jobDescription varchar(30) not null,

companyName varchar(30) not null,

location varchar(20) not null,

keySkill varchar(10) not null,

salary decimal(8,2) not null,

Active varchar(10) not null,

primary key(jobId)

);

-- Creating Table:: skill

create table skill(

skillId int auto\_increment,

skillName varchar(20) not null,

skillDescription varchar(30) not null,

Active varchar(10) not null,

primary key(skillId)

);

Select \* from employee;

Select \* from skill;

Select \* from job;

Select \* from empSkill;

Select \* from empJob;

### SAMPLE CODE

**Model package:**

***//Employee.java:***

**package** model;

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

**private** **int** employeeID;

**private** String firstName;

**private** String lastName;

**private** String userID;

**private** String password;

**private** String role;

**private** String gender;

**private** String active;

**public** Employee(**int** employeeID,String firstName, String lastName, String userID, String password, String role, String gender,String active) {

**super**();

**this**.employeeID=employeeID;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.userID = userID;

**this**.password = password;

**this**.role = role;

**this**.gender = gender;

**this**.active = active;

}

**public** Employee() {

// **TODO** Auto-generated constructor stub

}

**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 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 getRole() {

**return** role;

}

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

**this**.role = role;

}

**public** String getGender() {

**return** gender;

}

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

**this**.gender = gender;

}

**public** String getActive() {

**return** active;

}

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

**this**.active = active;

}

@Override

**public** String toString() {

**return** "Employee [employeeID=" + employeeID + ", firstName=" + firstName + ", lastName=" + lastName

+ ", userID=" + userID + ", password=" + password + ", role=" + role + ", gender=" + gender

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

}

}

**Dao package:**

***//IEmployeeDao.java:***

***package dao;***

***import model.Employee;***

***import java.util.\*;***

***public interface IEmployeeDao {***

***ArrayList<Employee> getAllEmployees();***

***void addEmployee(Employee emp);***

***Employee getEmployeeById(int id);***

***void updateEmployee(Employee emp);***

***void deleteEmployee(int id);***

***void deactivateEmployee(int id);***

***//void deactivateEmployee(Employee emp);***

***Employee checkLogin(String userId,String password);***

***void activateEmployee(int id);***

***Employee checkUser(String userId);***

***}***

**DaoImpl package:**

***//EmployeeDaoImpl.java:***

package daoImpl;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import config.JDBCConnection;

import dao.IEmployeeDao;

import model.Employee;

import view.RegistrationFrame;

public class EmployeeDaoImpl implements IEmployeeDao {

Connection conn=null;

public EmployeeDaoImpl() throws ClassNotFoundException, SQLException{

//Opened connection

conn=JDBCConnection.getDBConnection();

}

public Employee checkLogin(String userId, String password ){

Employee emp=new Employee();

try{

PreparedStatement pst=conn.prepareStatement("select \* from Employee where userId=? and password=?");

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.setUserID(rst.getString(4));

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

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

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

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

}

}

}

catch(SQLException ex) {

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

}

if(userId==emp.getUserID()&&password==emp.getPassword())

{

return emp;

}

return emp;

}

public Employee checkUser(String userId){

Employee emp=new Employee();

try{

PreparedStatement pst=conn.prepareStatement("select \* from Employee where userId=?");

pst.setString(1, userId);

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.setUserID(rst.getString(4));

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

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

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

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

}

}

}

catch(SQLException ex) {

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

}

if(userId==emp.getUserID())

{

return emp;

}

return emp;

}

@Override

public ArrayList<Employee> getAllEmployees() {

ArrayList<Employee> allEmpList=new ArrayList<Employee>(); //1

try{

Statement stmt=conn.createStatement();

ResultSet rst=stmt.executeQuery("select \* from Employee");

if(rst!=null) {

Employee emp=null;

while(rst.next()) {

emp=new Employee();

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

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

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

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

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

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

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

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

allEmpList.add(emp); //2

}

}

}

catch(SQLException ex) {

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

}

return allEmpList;

}

@Override

public void addEmployee(Employee emp){

try {

//creating PreparedStatement object by passing query string

PreparedStatement pst=conn.prepareStatement("insert into Employee(FirstName, LastName,UserId,Password,Gender,Role,Active) values(?,?,?,?,?,?,?)");

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

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

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

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

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

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

pst.setString(7, 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.getMessage());

}

}

@Override

public Employee getEmployeeById(int id) {

Employee emp=null;

try{

PreparedStatement pst=conn.prepareStatement("select \* from Employee where empId=?");

pst.setInt(1, id);

ResultSet rst=pst.executeQuery();

if(rst!=null) {

if(rst.next()) {

emp=new Employee();

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

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

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

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

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

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

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

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

}

}

}

catch(SQLException ex) {

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

}

return emp;

}

@Override

public void updateEmployee(Employee emp) {

try {

//creating PreparedStatement object by passing query string

PreparedStatement pst=conn.prepareStatement("update Employee set Role=? Password=? where empId=?");

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

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

int i=pst.executeUpdate();

}

catch(SQLException ex) {

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

}

}

public void activateEmployee(int id) {

try {

//creating PreparedStatement object by passing query string

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

pst.setString(1, "Active");

pst.setInt(2, id);

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 deactivateEmployee(int id) {

try {

//creating PreparedStatement object by passing query string

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

pst.setString(1, "Deactive");

pst.setInt(2, id);

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 deleteEmployee(int id) {

try {

//creating PreparedStatement object by passing query string

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

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());

}

}

}

**Controller package:**

***//EmployeeController.java:***

package controller;

import java.io.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JFrame;

import dao.IEmployeeDao;

import daoImpl.EmployeeDaoImpl;

import dao.IEmpJobDao;

import dao.IEmpSkillDao;

import daoImpl.EmpJobDaoImpl;

import daoImpl.EmpSkillDaoImpl;

import model.EmpJob;

import model.EmpSkill;

import model.Employee;

public class EmployeeController extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

static IEmployeeDao empDao=null;

public EmployeeController() throws ClassNotFoundException, SQLException{

empDao=new EmployeeDaoImpl();

}

public Employee checkLogin(String userId,String password) {

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

return emp;

}

public Employee checkUser(String userId) {

Employee emp=empDao.checkUser(userId);

return emp;

}

public void addEmployee(String s1, String s2, String s3, String s4,String s5, String s6) {

Employee emp=new Employee();

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

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

emp.setFirstName(s1);

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

emp.setLastName(s2);

//System.out.println("Enter UserId:");

emp.setUserID(s3);

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

emp.setPassword(s4);

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

emp.setGender(s5);

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

String role=s6;

emp.setRole(s6);

if (role.equals("HRA") || role.equals("EMP") || role.equals("PME")) {

emp.setActive("Active");

}

else {

emp.setActive("Deactive");

}

//Calling dao method for insert record

empDao.addEmployee(emp);

}

public ArrayList<Employee> getAllEmployees() {

ArrayList<Employee> allEmpList=empDao.getAllEmployees();

for(Employee emp:allEmpList) {

System.out.println(emp);

}

return allEmpList;

}

public Employee getEmployeeById(String EmployeeId) {

int id;

id=Integer.parseInt(EmployeeId);

Employee emp=empDao.getEmployeeById(id);

System.out.println(emp);

return emp;

}

public void updateEmployee() {

try {

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

int id;

String password, confirmpassword;

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

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

Employee emp=empDao.getEmployeeById(id);

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);

empDao.updateEmployee(emp);

}

else {

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

}

}

catch(IOException ex) {

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

}

}

public void deactiveEmployee(int id) {

/\*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());\*/

empDao.deactivateEmployee(id);

}

/\*

public void deactivateEmployee() {

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=empDao.getEmployeeById(id);

empDao.deactivateEmployee(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());

empDao.deleteEmployee(id);

}

catch(IOException ex) {

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

}

}

public void activateEmployee(int id) {

empDao.activateEmployee(id);

}

IEmpJobDao ejb=new EmpJobDaoImpl();

public List<EmpJob> getAllEmpJob(){

List<EmpJob> allEmpJoblist=ejb.getAllEmpJob();

for(EmpJob ejb:allEmpJoblist) {

System.out.println(ejb);

}

return allEmpJoblist;

}

public void addEmpJob(int s1, int s2, String s3) {

EmpJob emp=new EmpJob();

//emp.setEjID(s1);

emp.setEmployeeId(s1);

emp.setRecruited(s3);

emp.setJobId(s2);

ejb.addEmpJob(emp);

}

public EmpJob getEmpJobById(String ejId) {

int id;

id=Integer.parseInt(ejId);

EmpJob emp=ejb.getEmpJobById(id);

System.out.println(emp);

return emp;

}

public void updateEmpJob() {

try {

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

int id;

String recruited, confirm;

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

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

EmpJob emp=ejb.getEmpJobById(id);

System.out.println("Enter yes if recruited:");

recruited=reader.readLine();

System.out.println("Re-enter yes if recruited:");

confirm=reader.readLine();

if(recruited.equals(confirm)) {

emp.setRecruited(recruited);

ejb.updateEmpJob(emp);

}

else {

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

}

}

catch(IOException ex) {

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

}

}

/\*

public void deactiveEmpJob(int id) {

/\*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());

ejb.deactivateEmpJob(id);

}\*/

public void deleteEmpJob() {

try {

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

int id;

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

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

ejb.deleteEmpJob(id);

}

catch(IOException ex) {

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

}

}

IEmpSkillDao ek=new EmpSkillDaoImpl();

public List<EmpSkill> getAllSkills(){

List<EmpSkill> allEmpSkilllist=ek.getAllSkills();

for(EmpSkill ek:allEmpSkilllist) {

System.out.println(ek);

}

return allEmpSkilllist;

}

public void addEmpSkill(int s1, int s2, int s3) {

EmpSkill emp=new EmpSkill();

emp.setSkillId(s1);

emp.setEmployeeId(s2);

emp.setExpYear(s3);

ek.addSkills(emp);

}

public EmpSkill getSkillById(String ejId) {

int id;

id=Integer.parseInt(ejId);

EmpSkill emp=ek.getSkillById(id);

System.out.println(emp);

return emp;

}

public void updateSkill() {

try {

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

int id;

int expdate, confirmexpdate;

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

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

EmpSkill emp=ek.getSkillById(id);

System.out.println("Enter Exp Year:");

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

System.out.println("Re-enter Exp Year:");

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

if(expdate==confirmexpdate) {

emp.setExpYear(expdate);

ek.updateSkill(emp);

}

else {

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

}

}

catch(IOException ex) {

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

}

}

public void deleteSkill() throws IOException {

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

int id;

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

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

ek.deleteSkill(id);

}

}

**config package:**

***//JDBCConnection.java:***

**package** config;

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

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

**public** **static** Connection getDBConnection() **throws** ClassNotFoundException,SQLException{

//DataSourse information

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

String username="root";

String password="niit@123";

//Loading Driver

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

//Creating connection object

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

**return** conn;

}

}

**Entry package:**

***//EntryClass.java:***

package entry;

import java.util.Scanner;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.sql.SQLException;

import config.JDBCConnection;

import controller.EmployeeController;

import controller.SkillController;

import controller.JobController;

import java.lang.String;

public class EntryClass {

public void testConnection() throws ClassNotFoundException, SQLException{

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

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

}

else {

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

}

}

public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {

//EntryClass enter=new EntryClass();

//enter.testConnection();

String TableName;

Scanner s=new Scanner(System.in);

int choice = 0;

char c='y';

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

while(c=='y' || c=='Y') {

System.out.println("Enter the table name for performing CRUD operation(Employee/Skill/Job/EmpSkill/EmpJob): ");

TableName=reader.readLine();

if(TableName.equals("Employee")) {

EmployeeController empController=new EmployeeController();

Scanner sc=new Scanner(System.in);

int option;

char ch='y';

while(ch=='y' || ch=='Y') {

System.out.println("CRUD Operation Menu:");

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

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

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

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

System.out.println("5. Deactive an Employee record");

System.out.println("6. Active an Employee record");

System.out.println("7. Delete an Employees record");

System.out.println("8. View all Employee Skill records");

System.out.println("9. View single Employee Skill record");

System.out.println("10. Add an Employee Skill");

System.out.println("11. Update an Employee Skill record");

System.out.println("12. Delete an Employee Skill record");

System.out.println("13. View all Employee Job records");

System.out.println("14. View single Employee Job record");

System.out.println("15. Add an Employee Job");

System.out.println("16. Update an Employee Job record");

System.out.println("17. Delete an Employee Job record");

System.out.println("18. Exit");

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

option=sc.nextInt();

switch(option) {

case 1:

System.out.println("Following are all Employees Details:");

empController.getAllEmployees();

break;

case 2:

String id;

System.out.println("Enter the Id:");

id=s.next();

empController.getEmployeeById(id);

break;

case 3:

String s1,s2,s3,s4,s5,s6;

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

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

s1=s.next();

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

s2=s.next();

System.out.println("Enter UserId:");

s3=s.next();

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

s4=s.next();

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

s5=s.next();

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

s6=s.next();

empController.addEmployee(s1, s2, s3, s4, s5, s6);

break;

case 4:

empController.updateEmployee();

break;

case 5:

int ID;

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

ID=s.nextInt();

empController.deactiveEmployee (ID);

break;

case 6:

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

ID=s.nextInt();

empController.activateEmployee(ID);

break;

case 7:

empController.deleteEmployee();

break;

case 8:

System.out.println("Following are all Employee Skill Details:");

empController.getAllSkills();

break;

case 9:

System.out.println("Enter the Id:");

id=s.next();

empController.getSkillById(id);

break;

case 10:

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

int i1,i2,i3;

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

i1=s.nextInt();

System.out.println("Enter Skill ID:");

i2=s.nextInt();

System.out.println("Enter ExpYear:");

i3=s.nextInt();

empController.addEmpSkill(i1, i2, i3);;

break;

case 11:

empController.updateSkill();

break;

case 12:

empController.deleteSkill();

break;

case 13:

System.out.println("Following are all EmpJob Details:");

empController.getAllEmpJob();

break;

case 14:

System.out.println("Enter the Id:");

id=s.next();

empController.getEmpJobById(id);

break;

case 15:

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

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

i1=s.nextInt();

System.out.println("Enter Job ID:");

i2=s.nextInt();

System.out.println("Enter Recruted:");

s1=s.next();

empController.addEmpJob(i1,i2,s1);

break;

case 16:

empController.updateEmpJob();

break;

case 17:

empController.deleteEmpJob();

break;

case 18:

System.exit(0);

break;

default:

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

}

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

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

}

}

else if(TableName.equals("Skill")) {

SkillController skillController=new SkillController();

Scanner sc=new Scanner(System.in);

int option;

char ch='y';

while(ch=='y' || ch=='Y') {

System.out.println("CRUD Operation Menu:");

System.out.println("1. View all Skill records");

System.out.println("2. View single Skill record");

System.out.println("3. Add a Skill");

System.out.println("4. Update a Skill record");

System.out.println("5. Deactive a Skill record");

System.out.println("6. Delete a Skill record");

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

option=sc.nextInt();

switch(option) {

case 1:

System.out.println("Following are all Skill Details:");

skillController.getAllSkills();

break;

case 2:

System.out.println("Enter the Id:");

String id=s.next();

skillController.getSkillById(id);

break;

case 3:

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

String s1,s2;

System.out.println("Enter Skill name:");

s1=s.next();

System.out.println("Enter Skill Description:");

s2=s.next();

skillController.addSkill(s1, s2);

break;

case 4:

skillController.updateSkill();

break;

case 5:

int ID;

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

ID=s.nextInt();

skillController.deactivateSkill(ID);

break;

case 6:

skillController.deleteSkill();

break;

case 7:

System.exit(0);

break;

default:

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

}

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

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

}

}

else if(TableName.equals("Job")) {

JobController jobController=new JobController();

int option;

Scanner sc=new Scanner(System.in);

char ch='y';

while(ch=='y' || ch=='Y') {

System.out.println("CRUD Operation Menu:");

System.out.println("1. View all Job records");

System.out.println("2. View single Job record");

System.out.println("3. Add a Job");

System.out.println("4. Update a Job record");

System.out.println("5. Deactive a Job record");

System.out.println("6. Delete a Job record");

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

option=sc.nextInt();

switch(option) {

case 1:

System.out.println("Following are all Job Details:");

jobController.getAllJob();

break;

case 2:

jobController.getJobById();

break;

case 3:

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

String s1,s2,s3,s4,s5,s6;

System.out.println("Enter Job Title :");

s1=s.next();

System.out.println("Enter Job Description:");

s2=s.next();

System.out.println("Enter Company Name :");

s3=s.next();

System.out.println("Enter Location:");

s4=s.next();

System.out.println("Enter KeySkill:");

s5=s.next();

System.out.println("Enter Salary:");

s6=s.next();

jobController.addJob(s1,s2,s3,s4,s5,s6);

break;

case 4:

jobController.updateJob();

break;

case 5:

jobController.deactivateJob();

break;

case 6:

jobController.deleteJob();

break;

case 7:

System.exit(0);

break;

default:

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

}

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

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

}

}

System.out.println("Do you want perfom CRUD operation on other tables?(y/n)");

c=s.next().charAt(0);

}

}

}

**Test package:**

***//TestClass.java:***

package test;

import static org.junit.Assert.assertEquals;

import java.sql.SQLException;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

import dao.IEmployeeDao;

import daoImpl.EmployeeDaoImpl;

import model.Employee;

public class TestClass {

IEmployeeDao empDao=null;

@Before

public void setup() {

try {

empDao=new EmployeeDaoImpl();

}

catch(SQLException ex) {

}

catch(ClassNotFoundException ex) {

}

}

@Test

public void testGetEmployeeById() {

Employee empActual=new Employee();

Employee empExpected=new Employee(1,"Lopamudra","Bera","lopa123","pass@123","HRA","Female","Active");

empActual=(Employee)empDao.getEmployeeById(2);

//System.out.println(empActual);

assertEquals(empExpected.getFirstName(),empActual.getFirstName());

}

}

**View package:**

***//EMPHome.java:***

package view;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import controller.EmployeeController;

import model.Employee;

import java.sql.\*;

public class EMPHome extends JFrame {

Container container=null;

JLabel lTitle;

JButton bViewprofile,bUpdateProfile,bUpdateSkill,bApplyJob,bLogout;

private EmployeeController empController;

public EMPHome(Employee emp) {

container=getContentPane();

try {

empController = new EmployeeController();

} catch (ClassNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

lTitle=new JLabel("Welcome to Employee Portal");

bViewprofile=new JButton("View Profile");

bViewprofile.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

try {

new ViewProfile();

} catch (ClassNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

bUpdateProfile=new JButton("Update profile");

bUpdateProfile.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

new UpdateProfile();

}

});

bUpdateSkill=new JButton("Update Skill");

bUpdateSkill.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

new UpdateProfile();

}

});

bApplyJob=new JButton("Apply Job");

bApplyJob.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

}

});

bLogout=new JButton("Logout");

bLogout.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

bLogoutActionPerformed(evt);

}

});

setLayoutManager();

setLocationAndSize();

addComponentsToContainer();

this.setTitle("EMP Home Screen");

this.setVisible(true);

this.setBounds(10,10,500,450);

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

this.setResizable(false);

this.setLocationRelativeTo(null);

}

public void setLayoutManager() {

container.setLayout(null);

}

public void setLocationAndSize() {

lTitle.setBounds(180,50,300,30);

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

bUpdateProfile.setBounds(100, 150, 300, 30);

bUpdateSkill.setBounds(100, 200, 300, 30);

bApplyJob.setBounds(100, 250, 300, 30);

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

}

public void addComponentsToContainer() {

container.add(lTitle);

container.add(bViewprofile);

container.add(bUpdateProfile);

container.add(bUpdateSkill);

container.add(bApplyJob);

container.add(bLogout);

}

private void bLogoutActionPerformed(ActionEvent evt) {

System.exit(0);

}

}

***//LoginFrame.java:***

package view;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

import controller.EmployeeController;

import model.Employee;

public class LoginFrame extends JFrame{

Container container;

JLabel lUserId, lPassword, lMessage;

JTextField tUserId;

JPasswordField tPassword;

JButton bLogin, bRegister;

//JCheckBox cShowPassword;

EmployeeController empController=null;

public LoginFrame() throws ClassNotFoundException, SQLException {

container=getContentPane();

lUserId=new JLabel("USERNAME");

lPassword=new JLabel("PASSWORD");

lMessage=new JLabel();

tUserId=new JTextField();

tPassword=new JPasswordField();

bLogin=new JButton("LOGIN");

empController=new EmployeeController();

//Event handling for Login button

bLogin.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

String userId,password;

Employee emp=new Employee();

userId=tUserId.getText();

password=new String(tPassword.getPassword());

emp=empController.checkLogin(userId, password);

if(emp==null) {

lMessage.setText("You are not authorized user! Retry or Register!");

}

else {

if(emp.getGender().equals("HRA"))

{

try {

new HRAHome();

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

else if(emp.getGender().equals("EMP"))

{

new EMPHome(emp);

}

else

{

new PMEHome();

}

}

}

});

bRegister=new JButton("REGISTER");

//Event handling for Register button

bRegister.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

try {

new RegistrationFrame();

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

});

//cShowPassword=new JCheckBox("Show Password");

setLayoutManager();

setLocationAndSize();

addComponentsToContainer();

this.setTitle("Login Form");

this.setVisible(true);

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

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

this.setResizable(false);

this.setLocationRelativeTo(null);

}

public void setLayoutManager() {

container.setLayout(null);

}

public void setLocationAndSize() {

lMessage.setBounds(50, 80, 300, 30);

lUserId.setBounds(50, 100, 100, 30);

lPassword.setBounds(50, 200, 100, 30);

tUserId.setBounds(200, 100, 150, 30);

tPassword.setBounds(200, 200, 150, 30);

bLogin.setBounds(125, 300, 100, 30);

bRegister.setBounds(270, 300, 100, 30);

}

public void addComponentsToContainer() {

container.add(lUserId);

container.add(tUserId);

container.add(lPassword);

container.add(tPassword);

container.add(bLogin,BorderLayout.CENTER);

container.add(bRegister,BorderLayout.CENTER);

container.add(lMessage);

}

}

***Benefits of Implementing Automated Consultancy Service***

* Integration of legacy data
* Automated Process to smoothen the work flow
* Easily Scalable to match the fast-paced HR operation
* User Management
* Permission and Access controls
* HR Recruitment and Activity Management
* Real time Employee Skill Data