

Project Report

on

Resource Skill Management System

Developed By:

Name: Anzla Sharma Reg. No.: **R201171400503**

**Batch Code: S210050**

**Start Date: 1/06/2020**

**End Date: 20/07/2020**

**Name of the Coordinator: Ms. Lopamudra Bera**

**Names of the Developer: Anzla Sharma**

**Date of Submission: 22/07/2020**

**CERTIFICATION**

**This is to certify that this report, titled Resource Skill Management System embodies the original work done by Anzla Sharma, in partial fulfillment of his course requirement at NIIT.**

**Coordinator: Ms. Lopamudra Bera**

**ACKNOWLEDGEMENT**

My efforts bore fruit with the completion of this project. However, there are many others who share the reward of this effort simply because it would never have been this good without their help. I acknowledge the cooperation, encouragement and austerity of Ms. Lopamudra Bera whose guidance did half the magic of keeping me thrilled throughout this project.

ABSTRACT

This project manages the entire process of allocating the project to an employee as per his/ her skillset working in the company.

This application provides a single window system to Employee, HR and Project Managers of a company to cater the skill specific requirements emerging in Projects.

CONFIGURATION

Hardware: X64 Based PC

Software:

* Windows 10
* RAM: 128 MB
* Disk space: 124 MB for JRE; 2 MB for Java Update

Contents

|  |  |
| --- | --- |
| **Sr. No.** | **Description** |
| **1.** | Introduction  -Objective |
| **2.** | Software Requirement Specifications |
| **3.** | Project Requirement Analysis  -Project Plan  -System Architecture |
| **4.** | Project Design  -High Level Design  -Low Level Design  -User Interface Design  -System Input and Output Design  -Data Model |
| **5.** | Project Code |
| **6.** | Benefits |

**OBJECTIVE**

This project manages the entire process of allocating the project to an employee as per his/ her skillset working in the company.

This application provides a single window system to Employee, HR and Project Managers of a company to cater the skill specific requirements emerging in Projects.

**System Requirement Specification (SRS)**

|  |  |
| --- | --- |
| **S.No.** | **System Requirement** |
| 1 | Login screen for all roles like HR, PM , Employee |
| 2 | Application should have an option for HR to activate newly registered employee |
| 3 | Application should have an option for HR ta activate skills of Employee |
| 4 | Application should have an option for HR to recruit the employee for new job they are applied to |
| 5 | Application should have an option for HR to deactivate the employee who is no longer working for the organization |
| 6 | Application should have an option for Employee to view his own |
| 7 | There should an provision for employee to raise request to HR for changing the base information |
| 8 | Application should have an option for Employee to apply for a new job |
| 9 | In this application PM should have an option to generate new jobs for the employee |
| 10 | Application should have an option for PM to view the Employee who ap[plied for the job |
| 11 | In this application PM should have an option to deactivate the job created by him |
| 12 | There should be login /logout facility for the HR,PM and Employee |

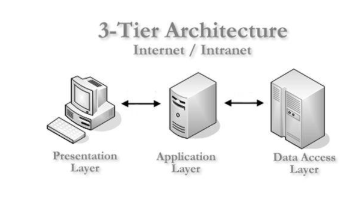
**Project Plan**

Phase wise plan -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Phases | W1 | W2 | W3 | W4 | W5 | W6 | W7 |
| Requirement gathering |  |  |  |  |  |  |  |
| High Level Design |  |  |  |  |  |  |  |
| Low Level Design |  |  |  |  |  |  |  |
| UI design |  |  |  |  |  |  |  |
| Code Development |  |  |  |  |  |  |  |
| testing |  |  |  |  |  |  |  |
| Roll Out |  |  |  |  |  |  |  |

This project was planned to be completed in 7 Weeks’ time.

**System Architecture**



An **N-Tier Application** program is one that is distributed among three or more separate computers in a distributed network.

The most common form of n-tier is the 3-tier Application, and it is classified into three categories.

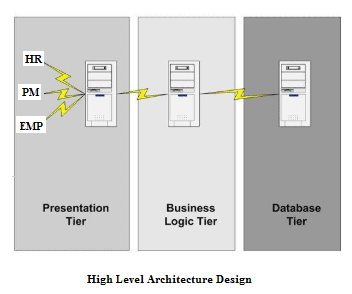
* User interface programming in the user's computer
* Business logic in a more centralized computer, and
* Required data in a computer that manages a database.

This architecture model provides Software Developers to create Reusable application/systems with maximum flexibility.

**3-tier architecture**has three different layers.

* Presentation layer
* Business Logic layer
* Database layer

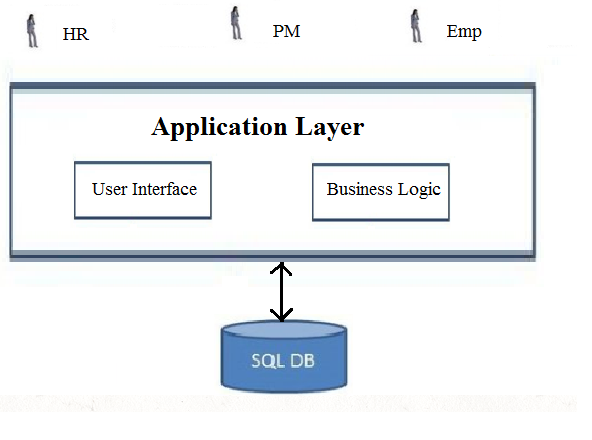
**High Level Architecture Design**

****

In the above figure the users namely EMP, HR , PM interact with the presentation tier of the application. After that the control is passed to business logic tier which interacts with database to fetch the required data. Once the required data is fetched the user can see the result on the presentation tier.

**High Level Design**

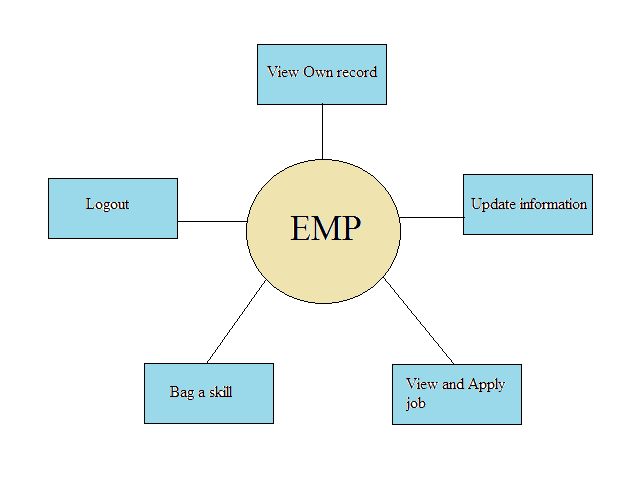
High-level design explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces



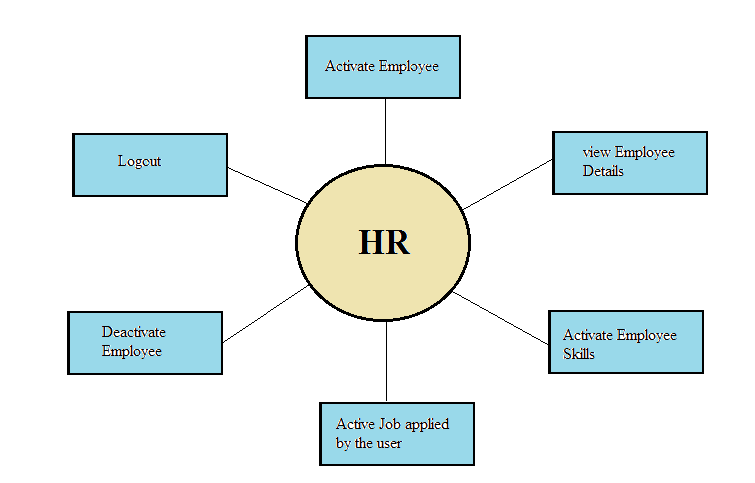
The user/s interact with the application layer via UI, which passes the request to the Business logic layer

The business logic layer interacts with the database tier where all records are stored to be fetched.

. **Low Level Design**

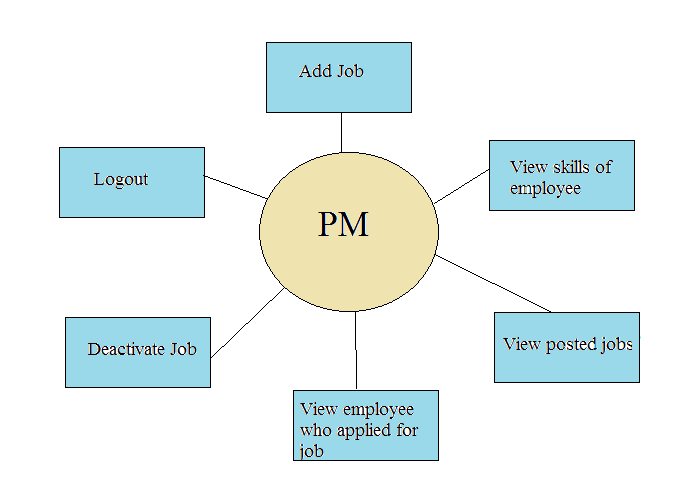
Here is the low level design of the EMP(employee). In this application the EMP can perform following task:

* Employee can view his/her own information.
* Employee can request to HR for updating his/her information.
* Employee can add more skills which he or she knows.
* Employee can apply for job assigned by the PM.
* Employees can logout his/her account for the safety purpose.



Here is the low level design of the HR. In this application the EMP can perform following task:

* HR can view employee information.
* HR can update employee information.
* HR can update requested skills of the employee.
* HR can recruit employee by the job assigned by the PM.
* HR can deactivate the employee who is no longer the part of organization.
* HR can activate new employees.
* HR can logout his/her account for the safety purpose.

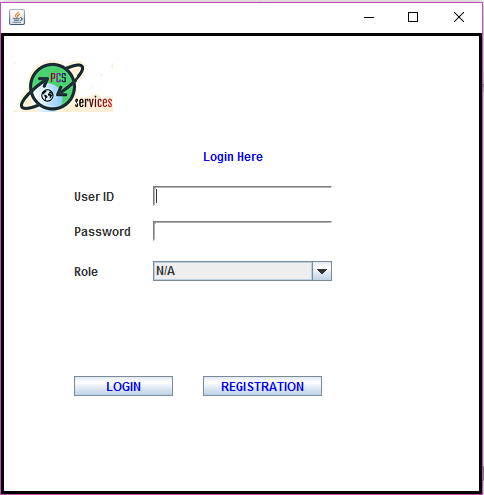


Here is the low level design of the PM. In this application the EMP can perform following task:

* PM can view skills of the employee.
* PM can add a new job for employees.
* PM can view the employees who applied for the job.
* PM can deactivate the job created by him/her.
* PM can view all posted jobs.
* PM can logout his/her account for the safety purpose.

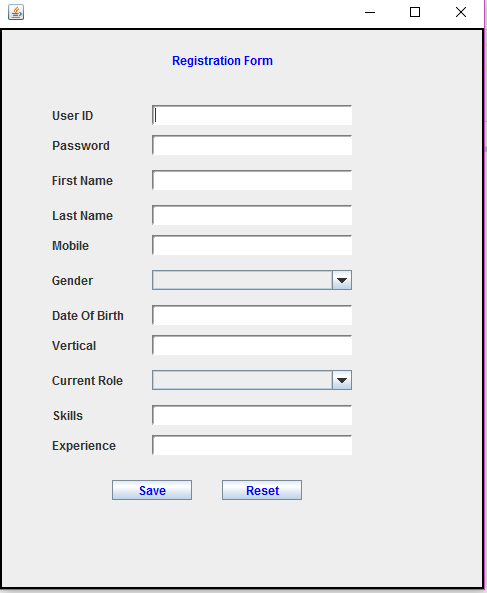
**User Interface Design**

Below is the common login page for HR, PM, EMP. The user ID and password are stored in database while the user registered him/ her at the first time.



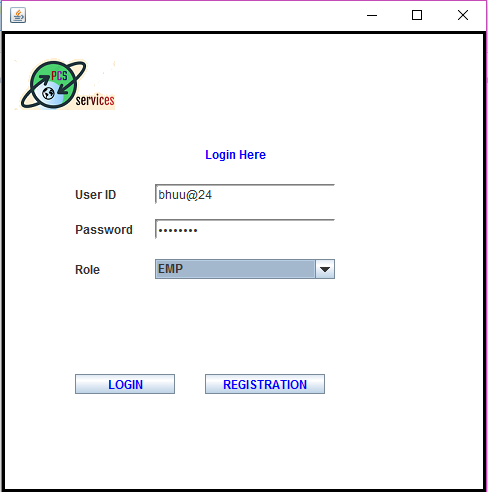
Login Page

Below is the Registration Frame where new employee can register himself/herself.

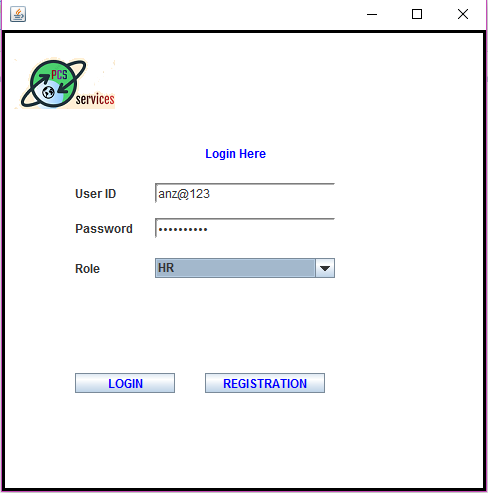


Registration Frame

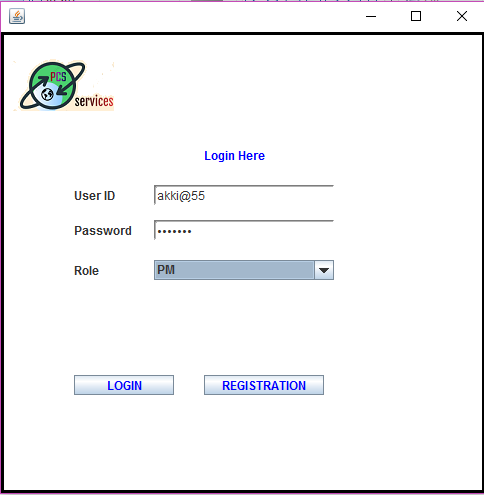
Login frame of employee where employees has to enter his userId and password for entering into the application.



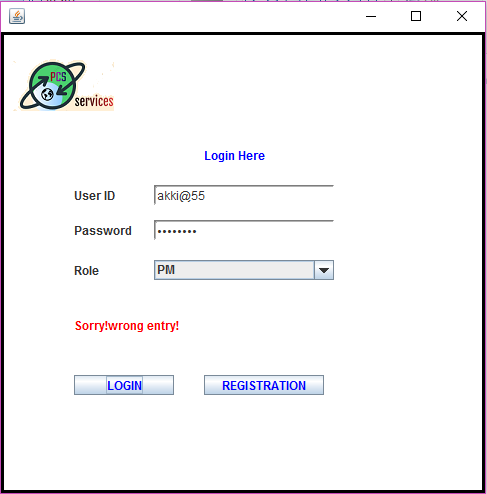
Login frame of HR where employee has to enter his userId and password for entering into the application.



Login frame of PM where employee has to enter his userId and password for entering into the application.



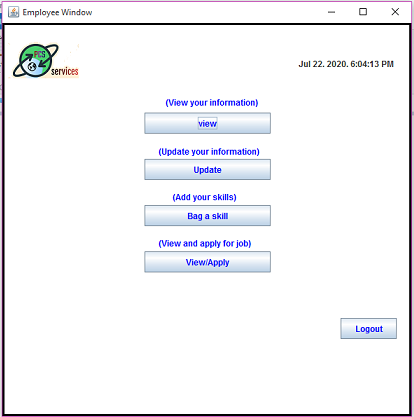
If the entered information does not matches with the stored user information this message will occur and the user will not able to enter the application.



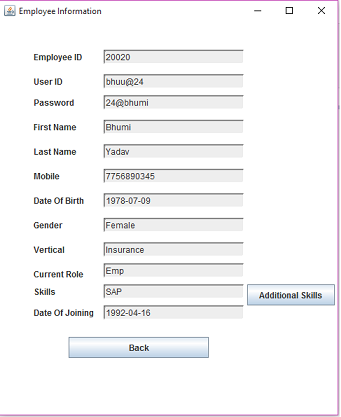
Validation @ login

System Input & Output Design

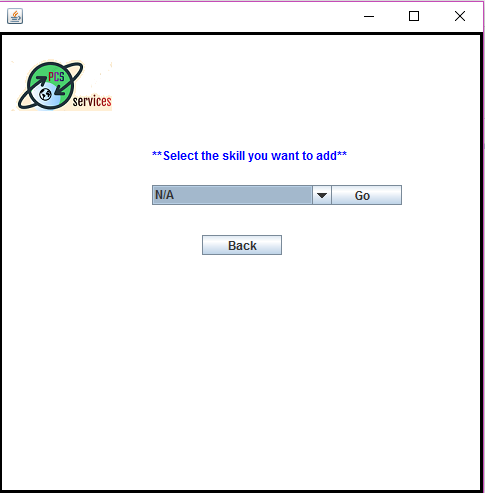
Employee Window has the following options.



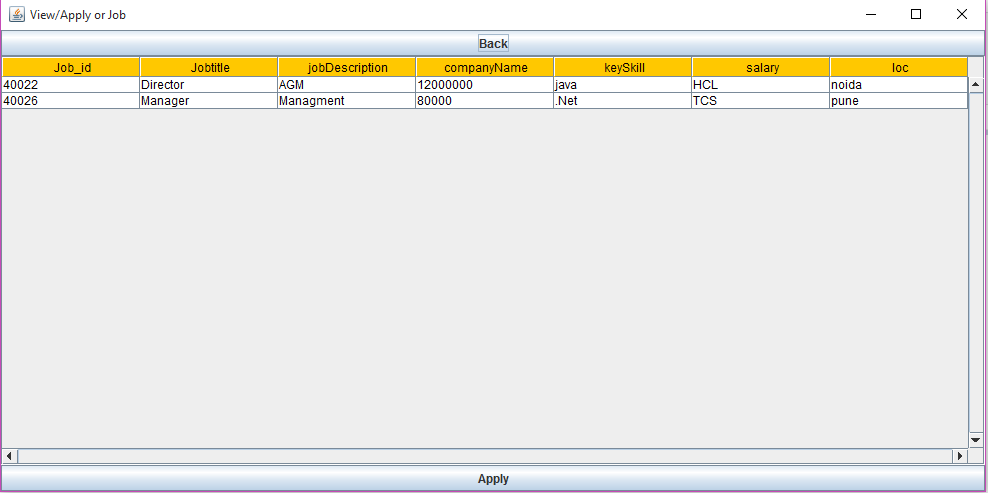
After Clicking on first option the employee can view his /her information.



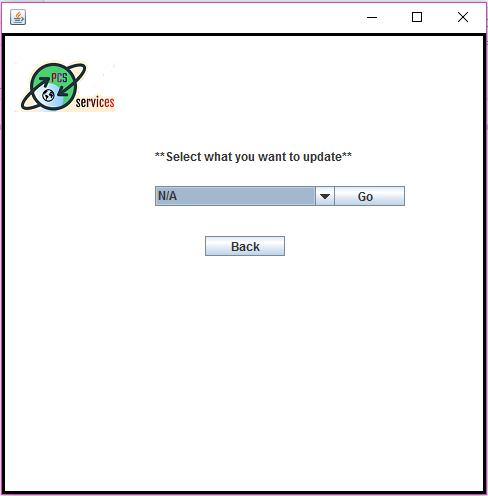
For adding more skills user must select the skill and can request HR for approval.



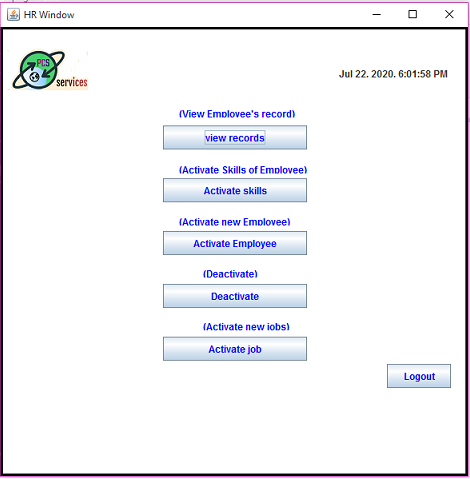
Through this option user can apply for the job created by the PM.



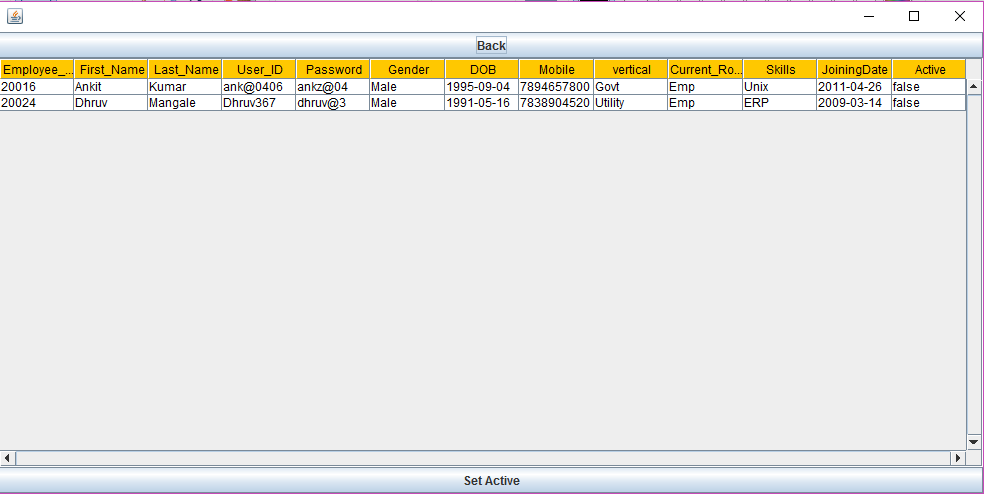
Through this option Employee can update his /her base information.



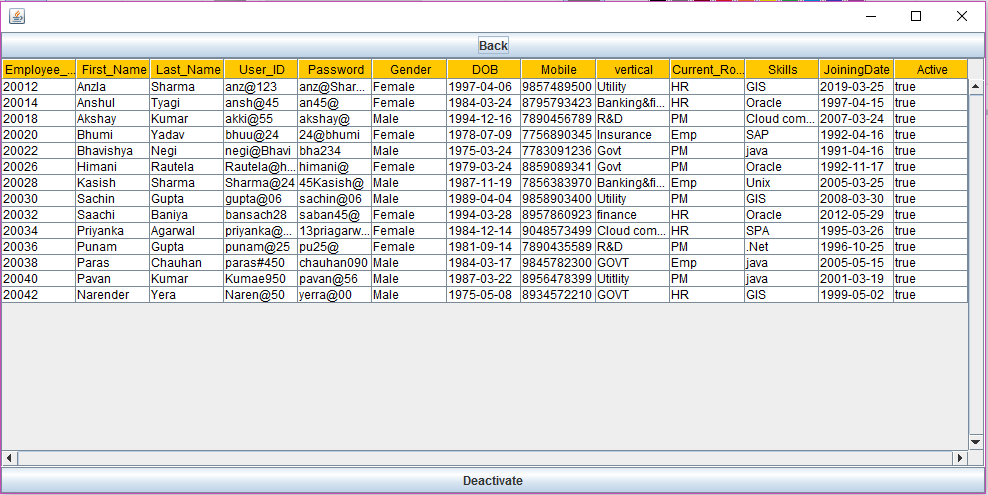
HR Window has the following options.



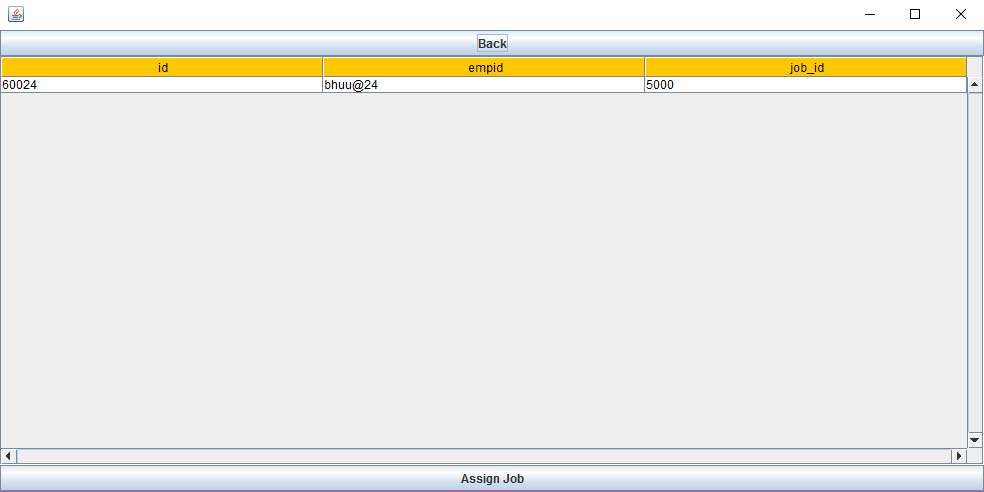
HR can activate the new employees.



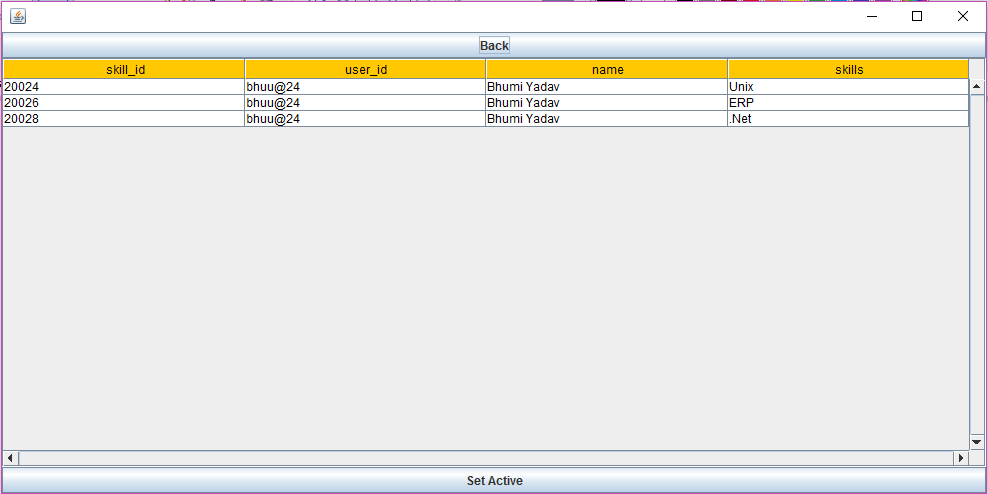
HR can deactivate Employee who is no longer the part of the organization



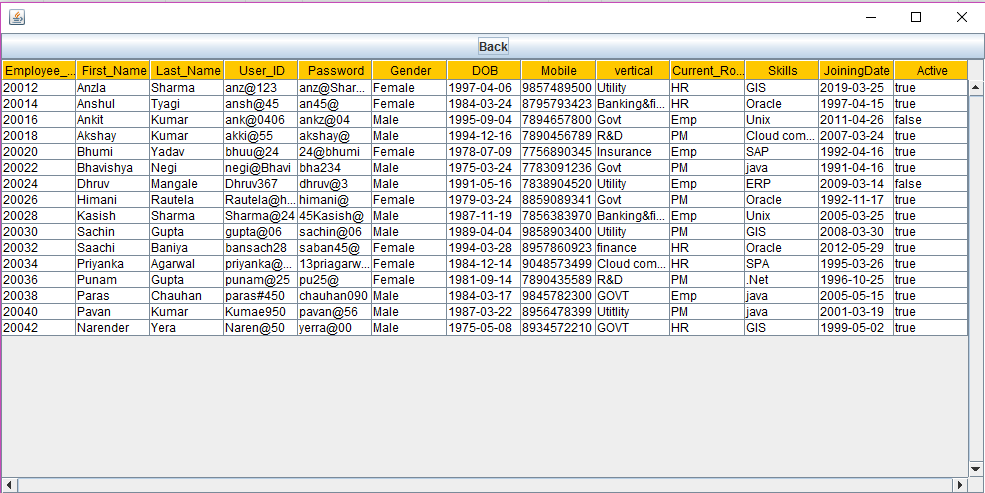
HR can recruit the employee applied for new job.



HR can activate the skills of the employee.



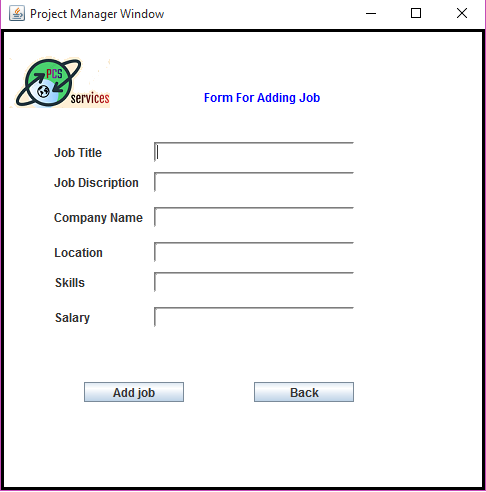
view all information of the employee.

HR can

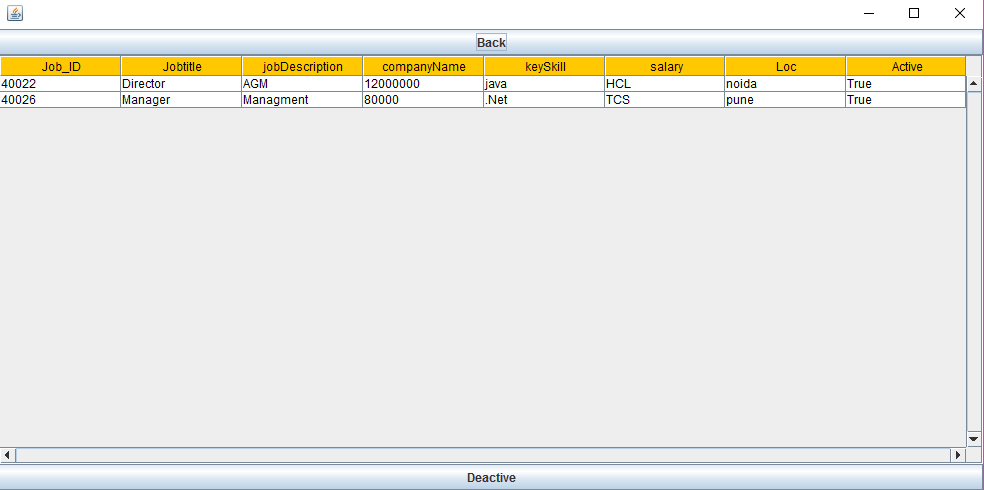
PM Window has the following options



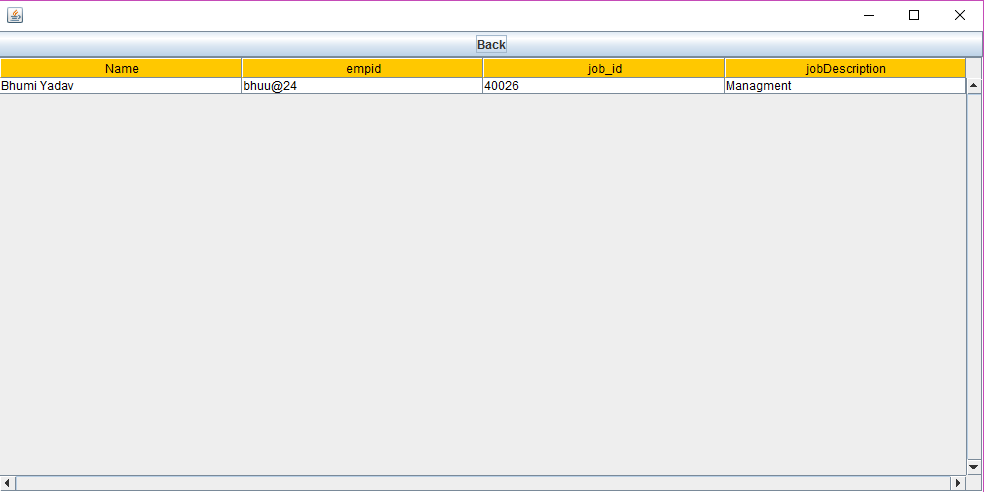
.PM frame to generate the new job.

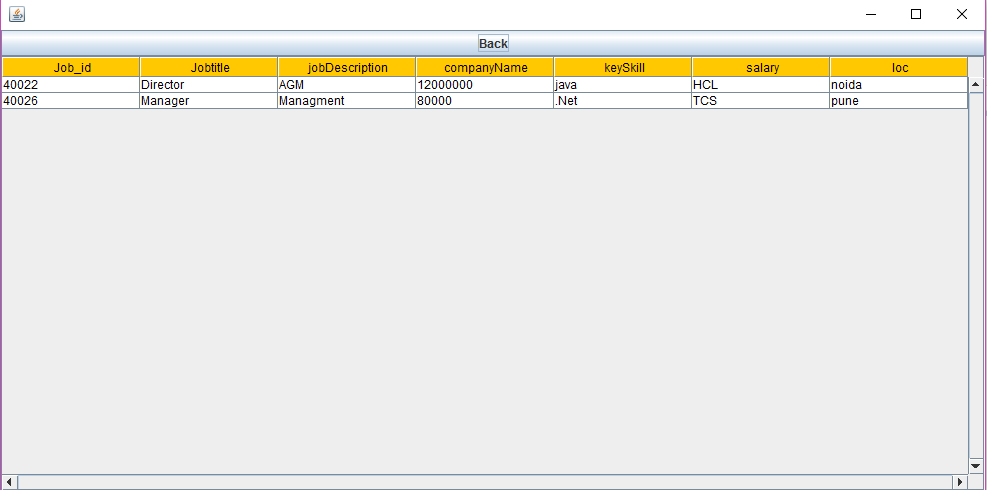


PM has option to deactivate jobs .



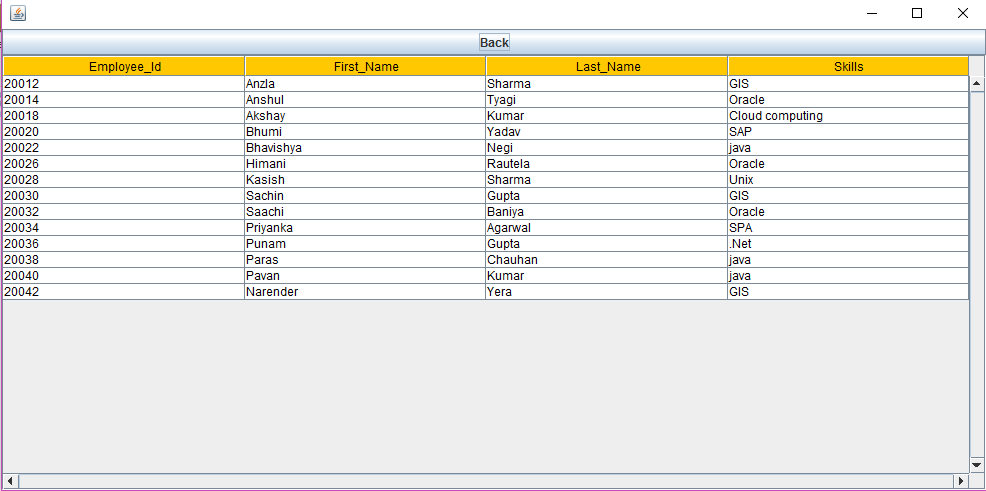
PM has option to view employee who applied for the job.



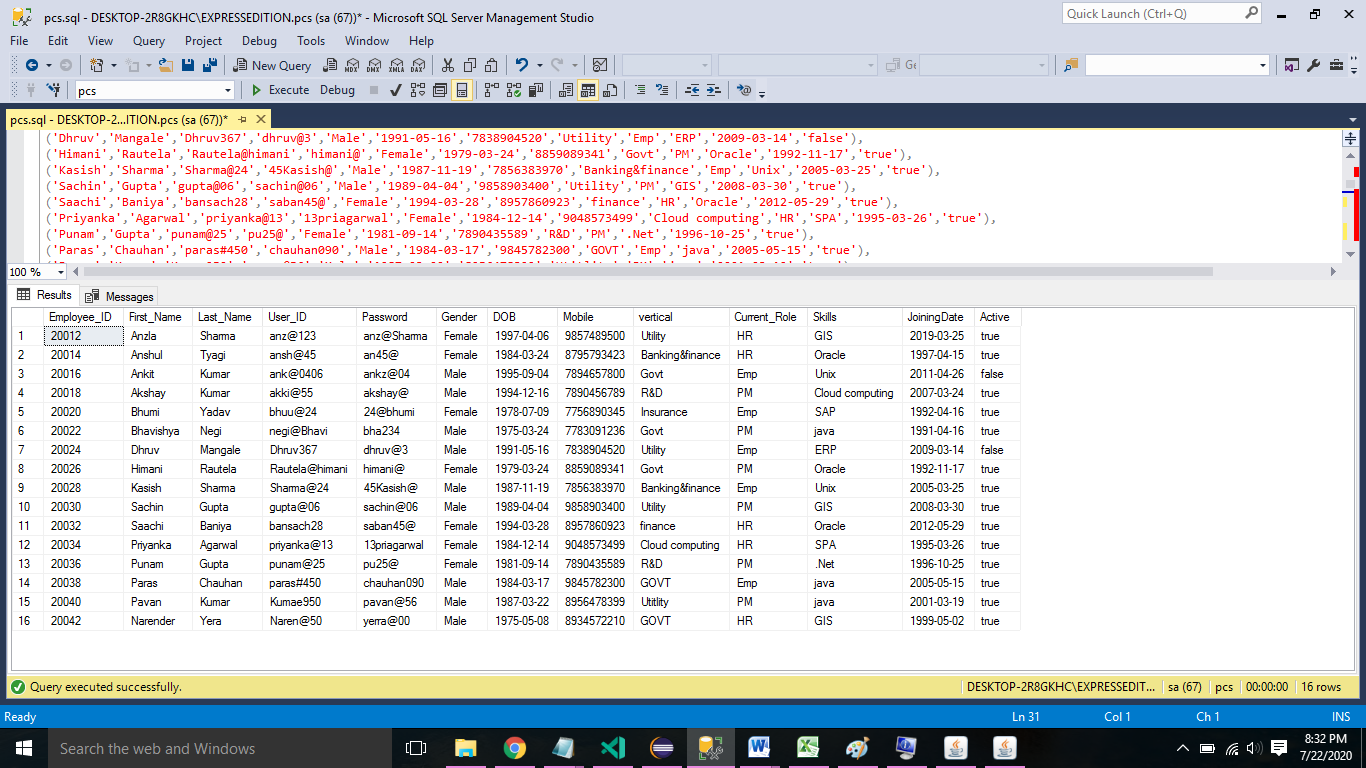
PM can view all posted jobs 

.

PM can view skills of the employee.



**Data Model**



**Create database:**

create database pcs

use pcs

go

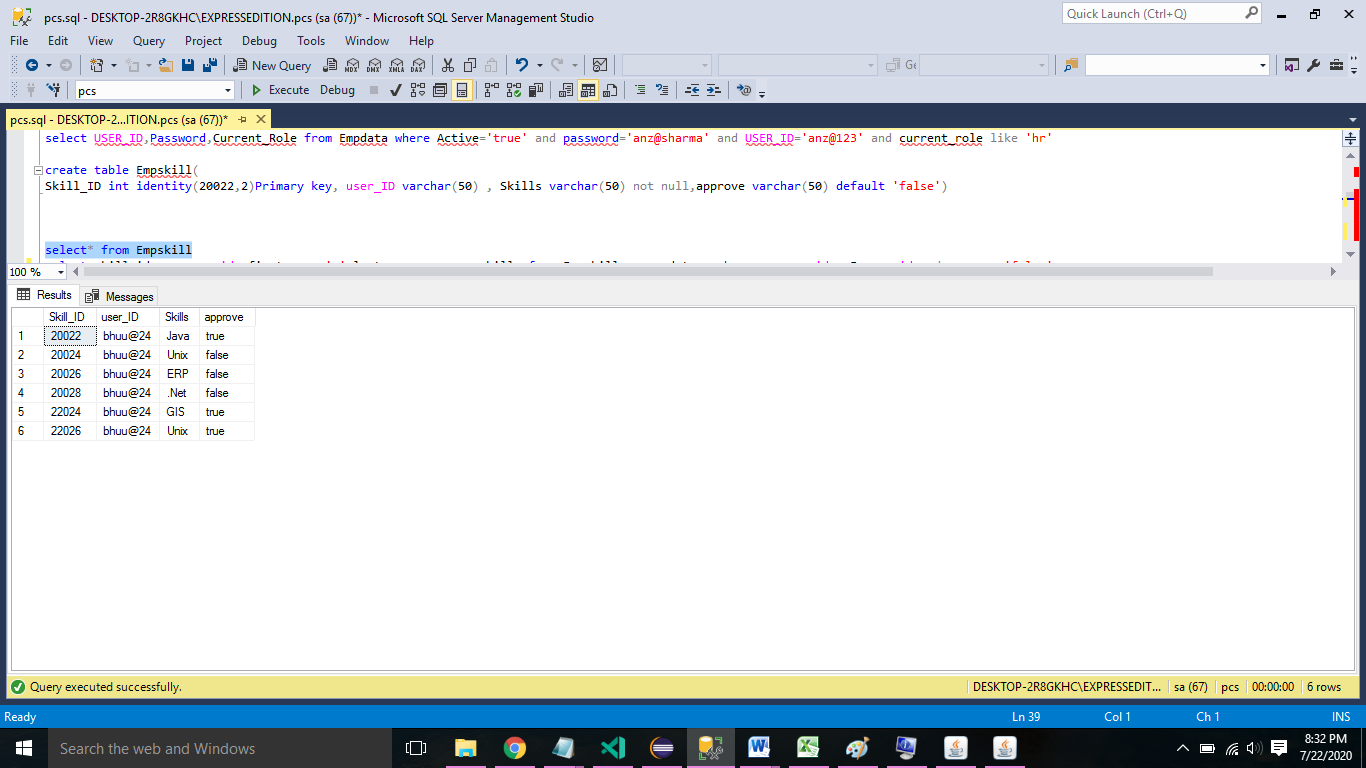
Create table

create table Empdata(

Employee\_ID int identity(20012,2)Primary key,First\_Name varchar(10)not null,Last\_Name varchar(10) not null,User\_ID varchar(20) unique not null,Password varchar(20) not null,

Gender char(6)not null,DOB varchar(10) not null,Mobile varchar(50) not null,vertical varchar(50) not null,Current\_Role varchar(10) not null,

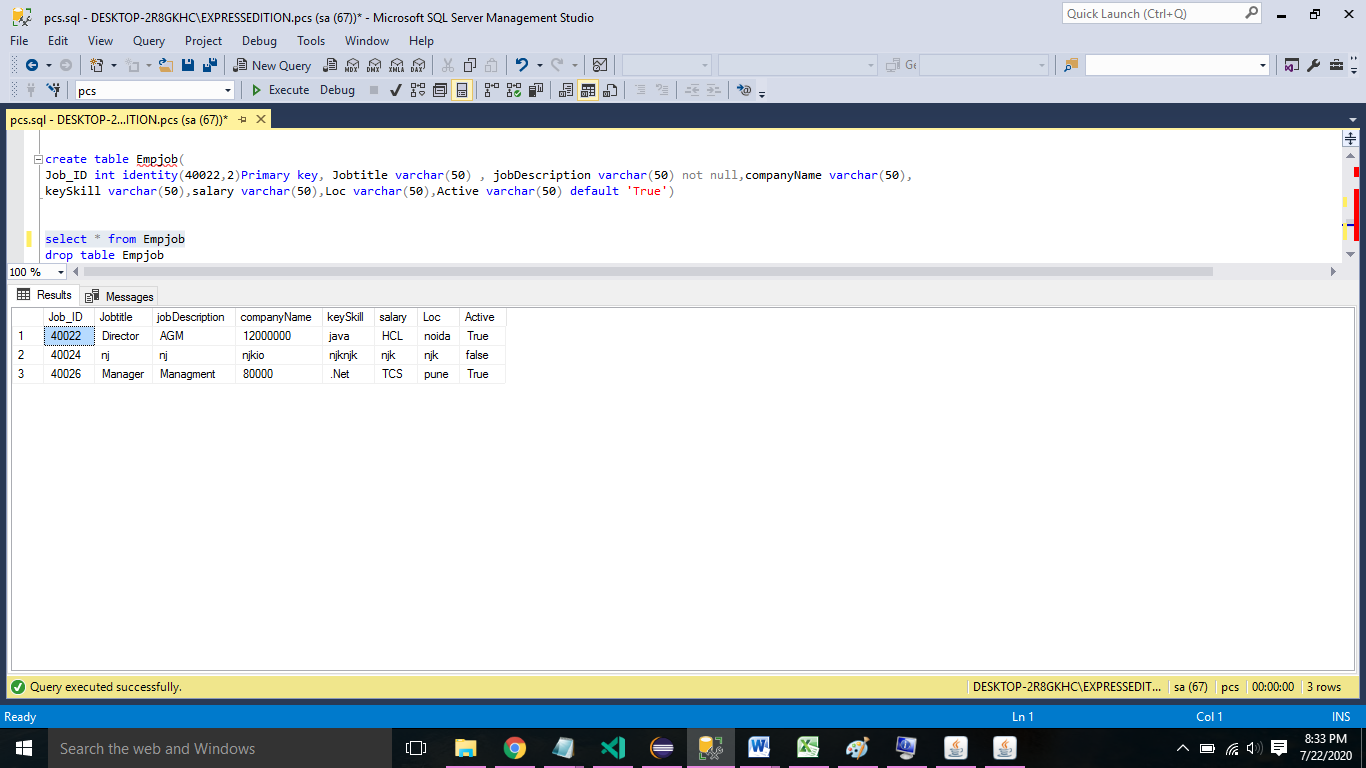
Skills varchar(50) not null,JoiningDate varchar(20)not null,Active varchar(10) )



Create table:

create table Empskill(

Skill\_ID int identity(20022,2)Primary key, user\_ID varchar(50) , Skills varchar(50) not null,approve varchar(50) default 'false')

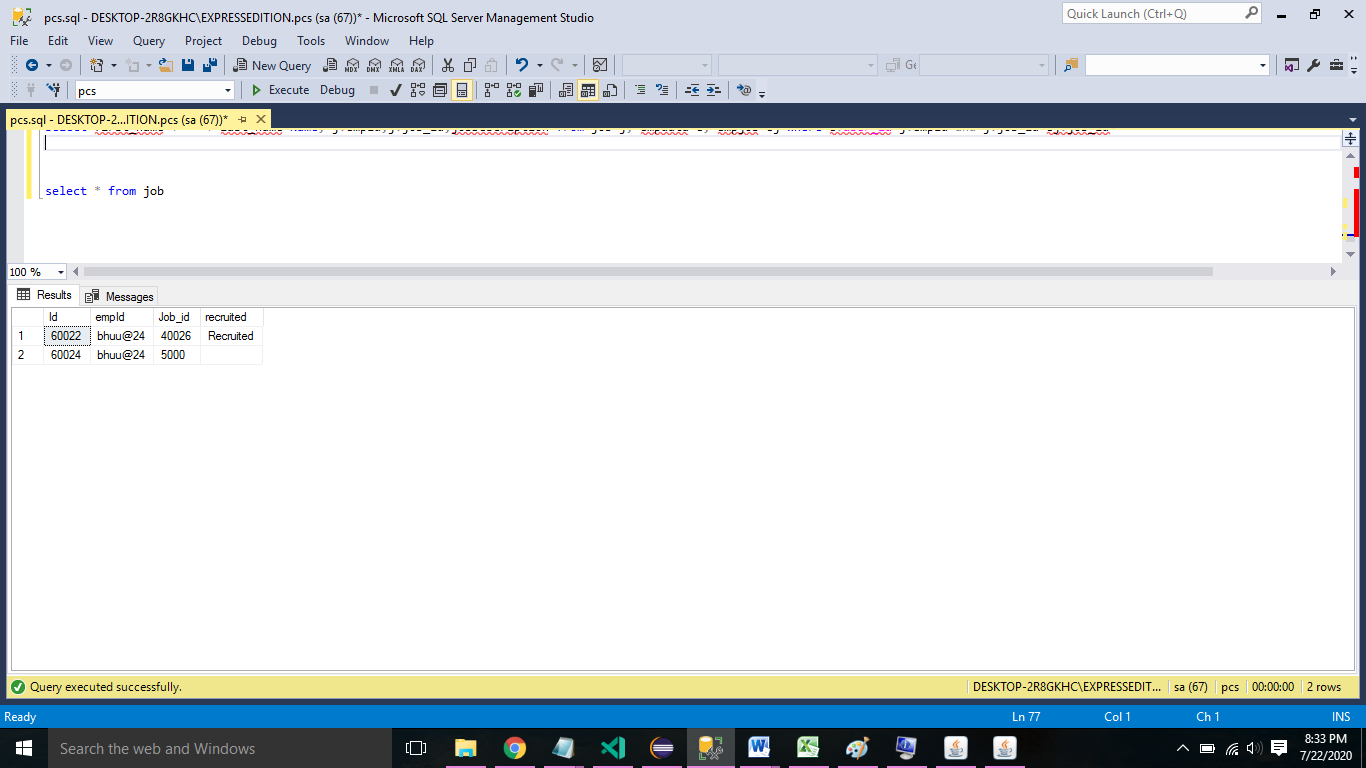


Create table:

create table Empjob(

Job\_ID int identity(40022,2)Primary key, Jobtitle varchar(50) , jobDescription varchar(50) not null,companyName varchar(50),

keySkill varchar(50),salary varchar(50),Loc varchar(50),Active varchar(50) default 'True')



Create table:

create table Job(

Id int identity(60022,2)Primary key,empId varchar(20) ,Job\_id varchar(20),recruited varchar(20) default '')

Sample Project Code –

**Configuration code**

**package** config;

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

**public** **class** connection {

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

Connection con=**null**;

//String url="jdbc:sqlserver://STUDIO03;database=pcs";com.microsoft

String url="jdbc:sqlserver://localhost:1433;instanceName=EXPRESSEDITION;database=pcs";

String username="sa";

String password="anzla@123";

Class.*forName*("com.microsoft.sqlserver.jdbc.SQLServerDriver");

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

**return** con;

}

**public** **static** Statement createStatement() {

// **TODO** Auto-generated method stub

**return** **null**;

}

}

**Interface code**

package dao;

import java.util.List;

import model.Employee;

public interface IEmployeeDao {

void addEmployee(Employee emp);

List<Employee> getAllEmployees();

Employee getEmployeeById(int id);

Employee getEmployeeByUserId(String userId);

void updateEmployee(Employee emp);

void deactivateEmployee(int id);

}

**Employee Class Code**

**package** model;

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

**private** **int** empId;

**public** String firstName;

**private** String lastName;

**private** String userId;

**private** String password;

**private** String currentRole;

**private** String Mobile;

**private** String vertical;

**private** String DOB;

**private** String skills;

**private** String gender;

**private** String DOJ;

**private** String active\_role;

**public** String getActive() {

System.***out***.println("in getactive" + active\_role);

**return** active\_role;

}

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

System.***out***.println("in setactive" + active\_role);

**this**.active\_role = active;

}

**public** Employee() {

}

**public** Employee(String firstName, String lastName, String userId, String password, String currentRole,

String mobile, String vertical, String dOB, String skills, String gender, String doj) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.userId = userId;

**this**.password = password;

**this**.currentRole = currentRole;

**this**.Mobile = mobile;

**this**.vertical = vertical;

**this**.DOB = dOB;

**this**.skills = skills;

**this**.gender = gender;

**this**.DOJ = doj;

}

**public** **int** getEmpId() {

**return** empId;

}

**public** **void** setEmpId(**int** empId) {

**this**.empId = empId;

}

**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 getCurrentRole() {

**return** currentRole;

}

**public** **void** setCurrentRole(String currentRole) {

**this**.currentRole = currentRole;

}

**public** String getMobile() {

**return** Mobile;

}

**public** **void** setMobile(String mobile) {

Mobile = mobile;

}

**public** String getVertical() {

**return** vertical;

}

**public** **void** setVertical(String vertical) {

**this**.vertical = vertical;

}

**public** String getDOB() {

**return** DOB;

}

**public** **void** setDOB(String dOJ) {

DOJ = dOJ;

}

**public** String getDOJ() {

**return** DOJ;

}

**public** **void** setDOJ(String dOJ) {

DOJ = dOJ;

}

**public** String getSkills() {

**return** skills;

}

**public** **void** setSkills(String skills) {

**this**.skills = skills;

}

**public** String getGender() {

**return** gender;

}

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

**this**.gender = gender;

}

@Override

**public** String toString() {

**return** "Employee [empId=" + empId + ", firstName=" + firstName + ", lastName=" + lastName + ", userId=" + userId

+ ", password=" + password + ", currentRole=" + currentRole + ", Mobile=" + Mobile + ", vertical="

+ vertical + ", DOB=" + DOB + ", skills=" + skills + ", gender=" + gender + ", active\_role=" + active\_role + "]";

}

}

Benefits of Implementing Resource Skill Management System (RSMS)

* Integration of legacy data
* Automated processes to smoothen the work-flow
* Easily Scalable to match the fast-paced HR operations
* User Management
* Permission and Access Controls
* HR Recruitment and Activity Management
* Real-Time Employee Skill Data

---------------- End Of Document ------------------