

***Personal Consultation Services***

**Developed by:** Ganesh Pandi E

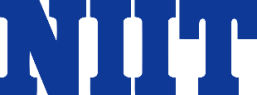
**Reg No:** R200002100293

**Batch Code:** 210167

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

**Date of Submission:** 12/12/2020

***CERTIFICATE***

This is to certify that “Ganesh Pandi E” student of Batch-S210167 has successfully completed their Project on “Professional Consultancy Services” under the guidance of “Miss.Lopamudra Bera”.

Miss.Lopmudra Bera Ganesh Pandi E

***ACKNOWLEDGEMENT***

I would like to express my special thanks of gratitude to my teacher

“Miss.Lopamudra Bera” for their able guidance and support in completing my Project.

Date: Ganesh Pandi E

12/12/2020  R200002100293

***ABSTRACT***

This Project will mainly use for professional networking, allowing registered employees to post jobs and their information is organized and easily accessible to all registered users such as job seekers, employers and HR consultant.

Users will register themselves in the application and fill in their professional details along with their core competencies. And Employers can be able to search relevant profiles for their openings using the keywords that matches with the profiles. Based on the match, relevant profiles should be shown to the employers with their contact details and International Business Unit name.

|  |  |
| --- | --- |
| ***CONFIGURATION***   |  | | --- | | * Hardware   + - Core i3 Processor 2.10 GHz x64 bit     - 8 Gigabytes of RAM * Operating System   + - Windows 10 Home * Software   + Eclipse IDE     - JAVA     - *MYSQL* | |

***OBJECTIVE***

This Project will mainly use for professional networking, allowing registered employees to post jobs and their information is organized and easily accessible to all registered users such as job seekers, employers and HR consultant.

Users will register themselves in the application and fill in their professional details along with their core competencies. And Employers can be able to search relevant profiles for their openings using the keywords that matches with the profiles. Based on the match, relevant profiles should be shown to the employers with their contact details and International Business Unit name.

***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.

***STATEMENT OF REQUIREMENTS***

**Key Features of the application:**

* Skill mappingfeature is the most prominent feature of this software that will reduce time and effort spent on profile screening and mapping the employees’ profiles with the current business requirement.
* Also this software holds the other features such as having a separate window for Employees, Project Managers and HR’s to access their portal.
* It has the registration window to register with the company by themselves.
* Application has the same login window for everyone and it detects by itself which designation you are in and give access accordingly.
* HR can able to activate and deactivate an employee’s profile and recruit an employee to their organization.
* HR can able to view all the employee’s status and can add, delete, activate or deactivate a skill in the portal.
* Project Manager can add a job on the portal and deactivate or activate a job while it is needed.
* PM can also monitor the employees who has applied for the posted job on the portal and also sort out the jobs using particular skill name.
* Employee will have access to view and update their profile anytime and can see what the available jobs on their required verticals are.
* They can apply for the jobs by mentioning the job id that will immediately reflect on the HR’s portal.

**PM**

* Activate or Deactivate Employee
* View All Employee
* View Selective Employee
* Add Skill
* View All Skill
* Activate or Deactivate Skill
* Add and Update Job
* Activate or Deactivate a Job
* View All Job
* View Any Employee Profile
* View Employee Who Apply For Job

HR

EMP

In the above model is that the overall functionality model of the application that we have already discussed in the previous chapter.

***HIGH LEVEL CASE DIAGRAM***

**HR**

**Discussing the needed** skills

**Screening and recruiting**

**JOB POSTING**

**EMP**

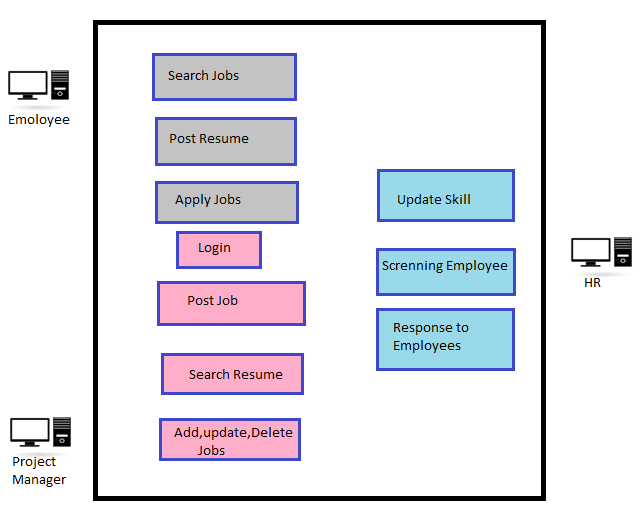
**PME**

**DATABASE**

**MYSQL**

A High-level design document adds the necessary details to the current project description to represent a suitable model for coding.

***LOW LEVEL CASE DIAGRAM***

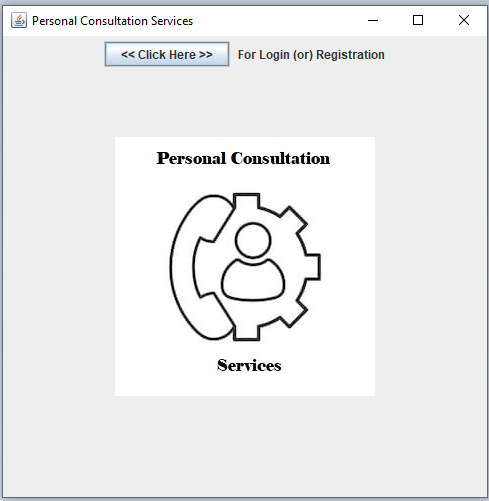


Here is the Low level case diagram which represents everything a user can do. An Employee can search for a job, update their profile and apply for a job.

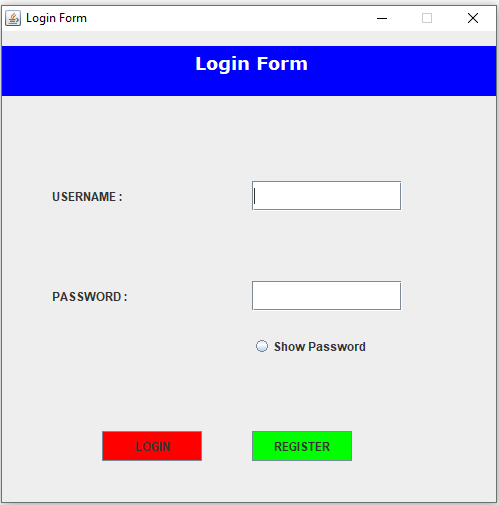
Project Manager can post the job necessary for the organization and monitor the cycle. He can able to activate and deactivate a job according to the situation of the company. Similarly, HR can monitor the profiles who are applied for the particular jobs and screening them by matching the skill sets to pick a suitable employee to fit the project.

***USER INTERFACE DESIGN***

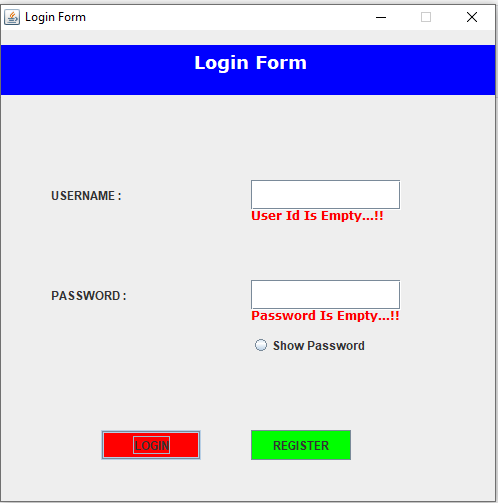
The common user interface for the entire system that includes an Employee, Project Manager and HR.



The user has to be Click “Clcik Here” button before login or registered.



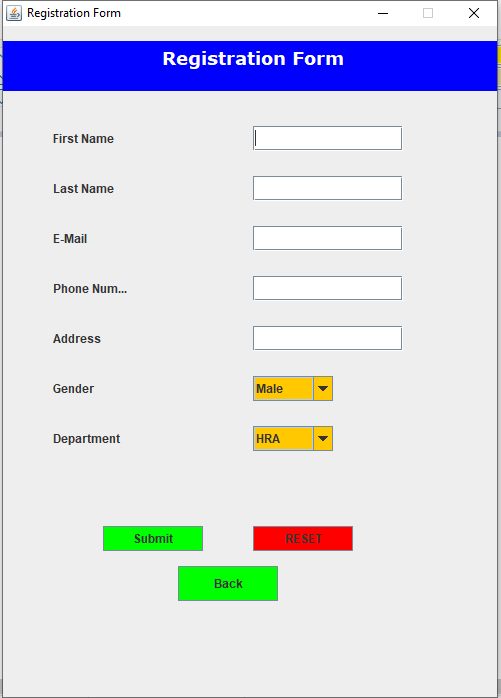
User must fill the Text field before click login (or) it will thorw error message.



The above snap explains must fill the textfield before click login.

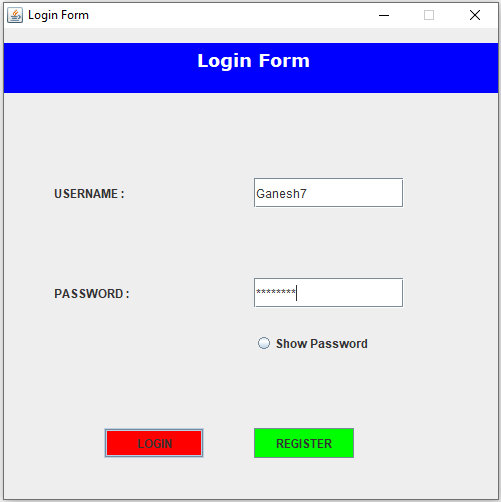
The user has to be registered as Professional Consultancy Services Employee before login. This frame will connect to the database and checks the user’s authorization and then also set which designation they belong to Employee Window, PME Window, HR Window and then will give access to the user . If user doesn’t have authorization for login then user has to click “REGISTER” button for Registration.

This is the Registration form that the user has to fill their details to get registered with the Professional Consultancy Services.



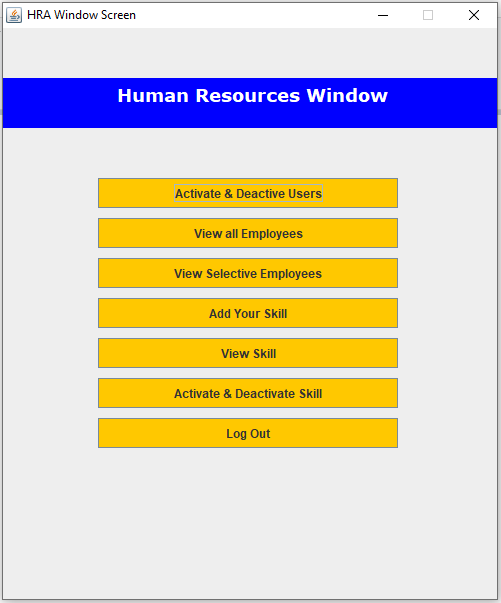
The System must not miss out any kind of information while dealing with the registration form. So, the authentication is very much important for the organization. The registration form would check each and every column for null value and alert the user to fill that missing field and also it makes sure that the entered all textbox without miss any textbox.



In above snap Phone Num textbox is missed to fill and if you clicked Submit button .It will throw error message and fill the required field to complete registration.

This is how a user can login to their portal. This is a HR login credentials. This navigates us to HR portal. While registering with the portal, one need to choose that the user is which an Employee / Project Manager / HR .It will walk us through which portal we are having the access to.

HR Window Screen:-

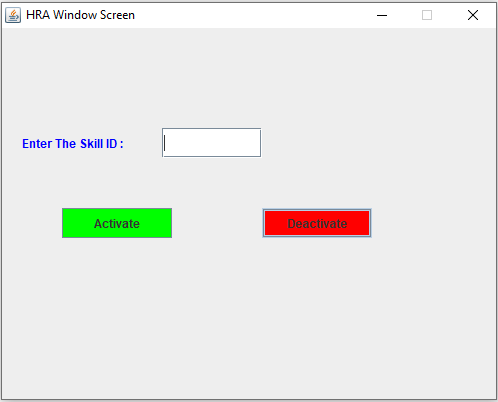


This is how the HR Window looks like. Here, can see that what HR can have the access to do.

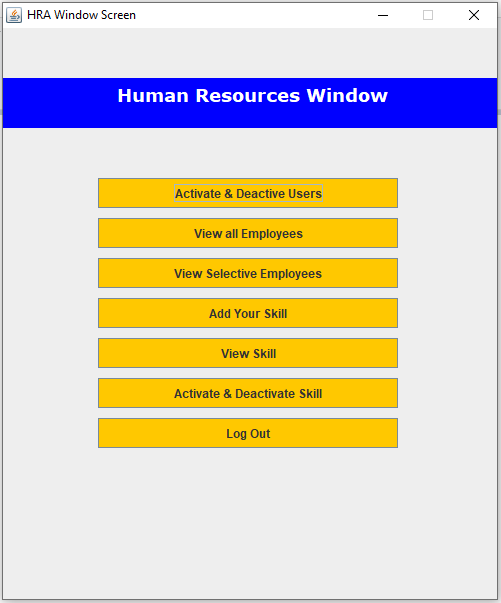
HR can

* Activate or deactivate Users
* View all employees
* View Selective Employee
* Add Your skill
* View all available skills in the database
* Activate, deactivate Skill.

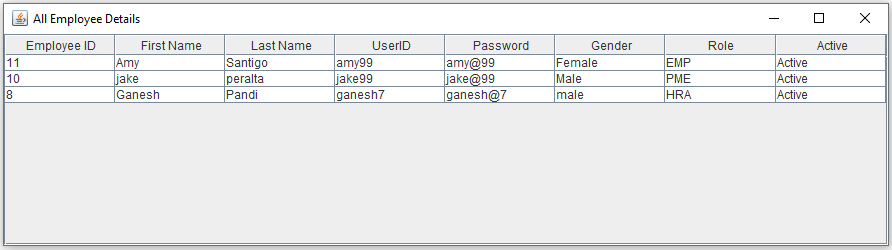
HR Can to Activate or deactivate Employee/Skill with Employee ID/Skill ID

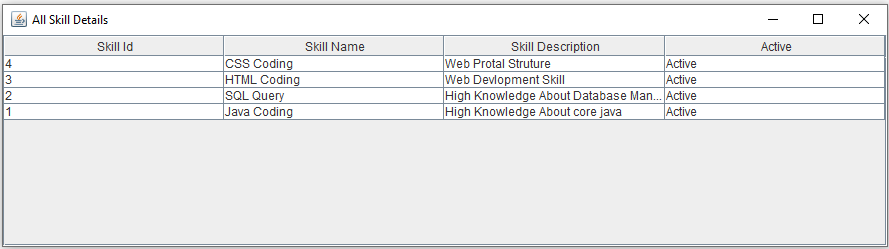
HR can see all the Employee/skills that includes to see deactivated Employee/Skills to check the data



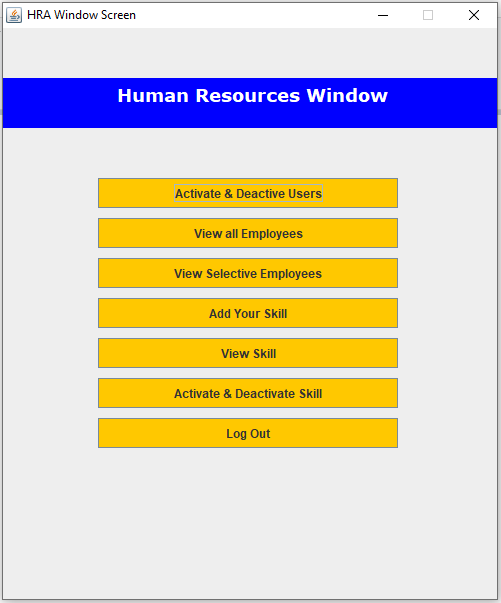
All Employee Details:-

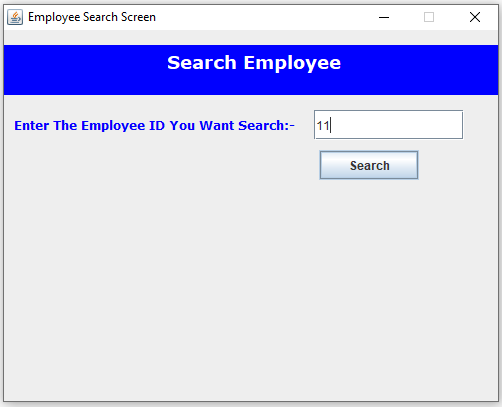


All Skill Details:-

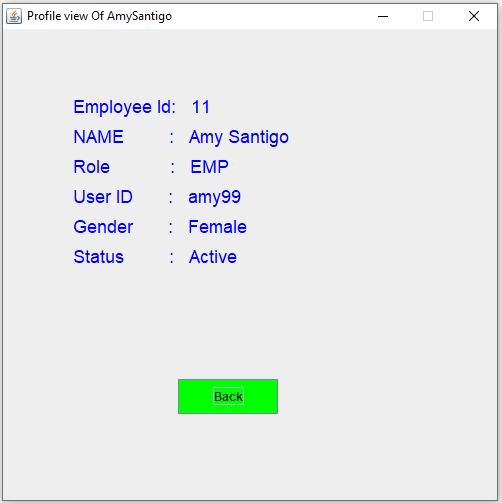


HR Can able to see a particular selective employee’s details with Employee ID



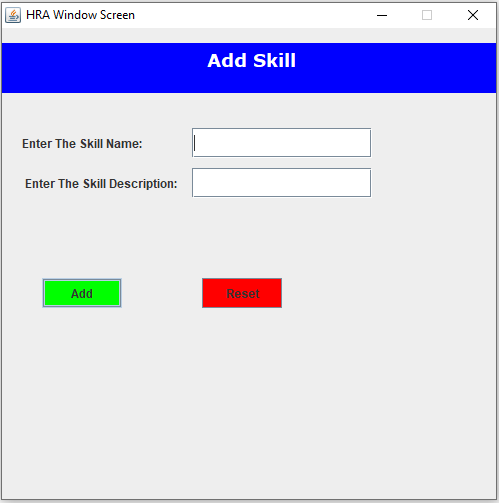


After enter Employee ID click the Search button then HR can see a particular employee details.



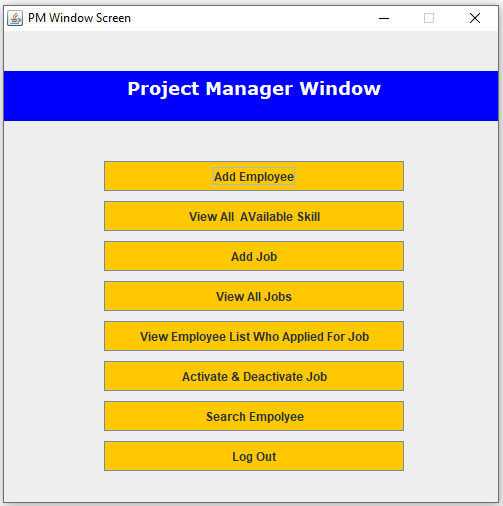
Above snap explain how the view selective employee profile

HR can able to add a skill.



HR can able to add the skill by entering the skill name and the skill description. It will update into database and application.

Project Manager View:-

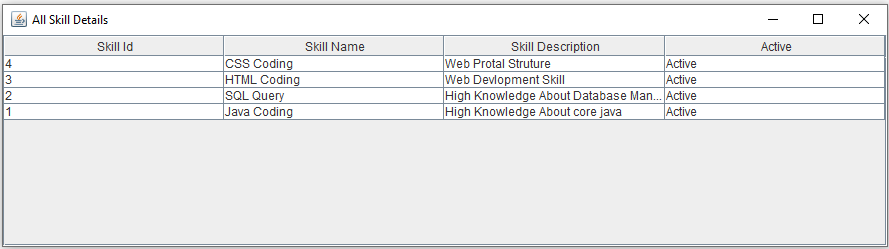


Nice and sleekly designed user interface that allows the managers to take navigate through the process quickly and effectively. The above pictures shows that what a PM capable of doing in that organization. He is responsible for the entire process of monitoring employees who belongs to that firm.

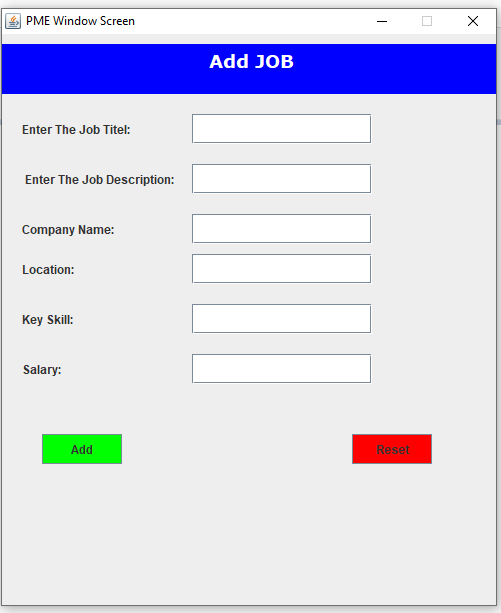
PM can add a Employee details:



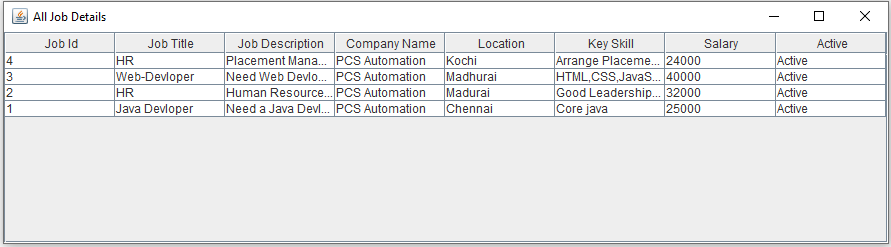
Pm Can view all avaible skill:-



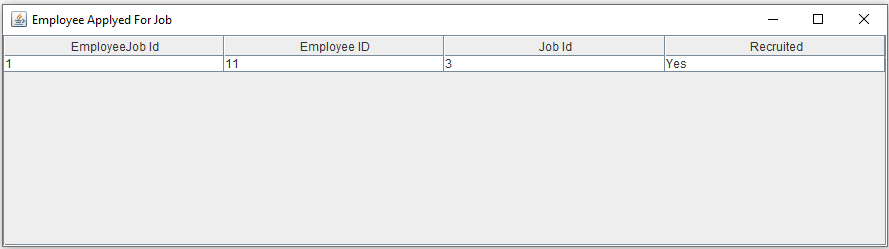
Pm Can be Posted or Insert a Job details:



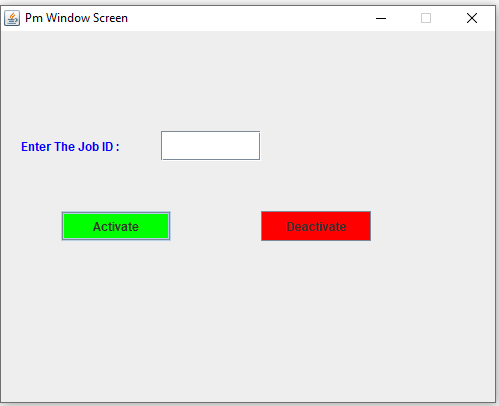
PM can able to view all the jobs that he has added



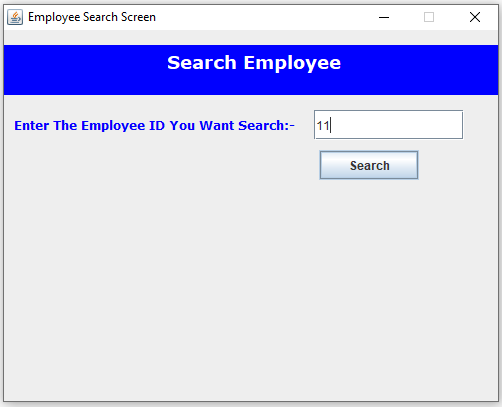
Manager can monitor the employees who have applied for the job and manage the job flow and response to that particular project.

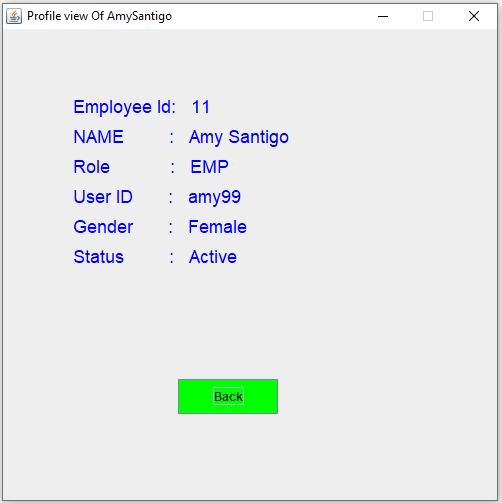


Project Manger can to Activate/Deactivate a particular Job



Pm can a see a selective employee deatlis:

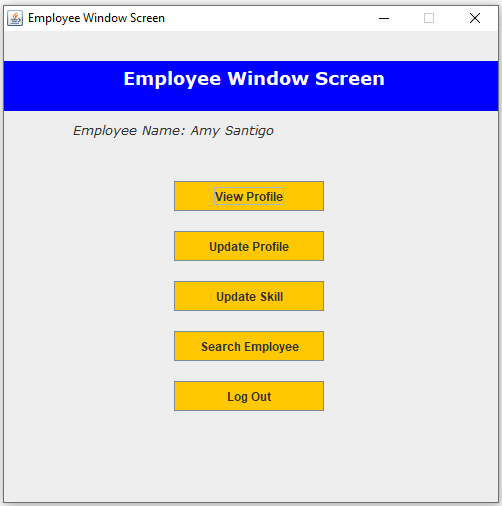




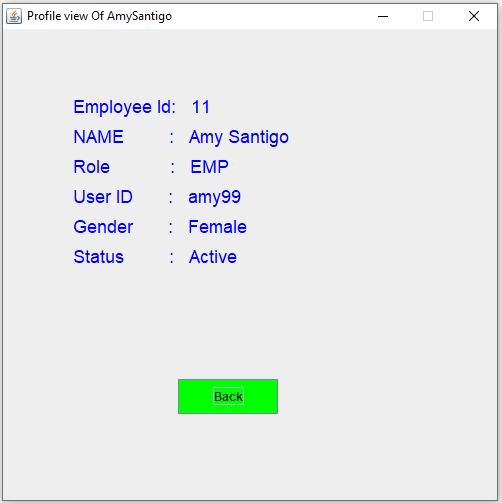
Employee Window Screen:-

Once you login as an employee, you will be automatically directed to your profile view as Employee to “Employee Window Screen”.

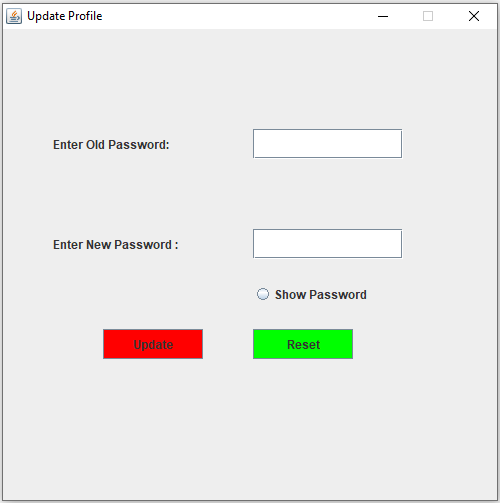
Employee view can give access to the employees to view their profile. Update their profile , Update their skill and Search Employee.



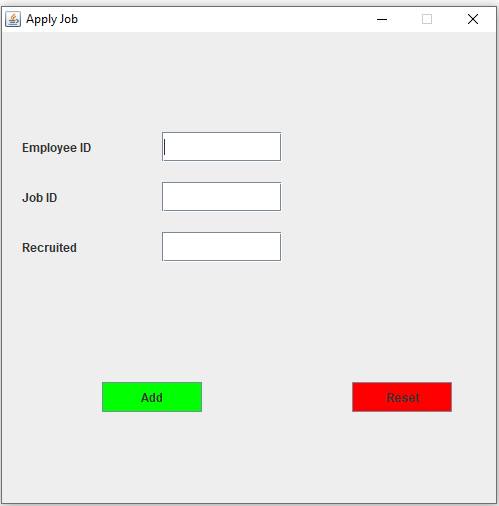
Employee Can see their own Profiles



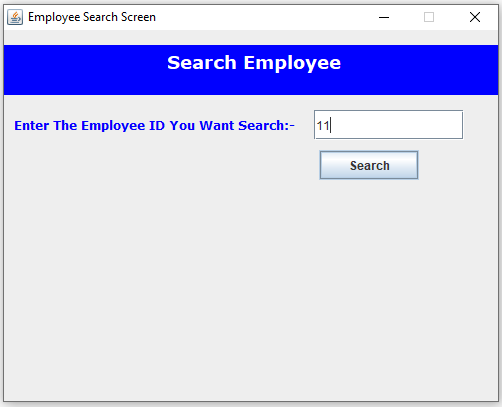
Employee Can Update their Profile like password.



Employee Can Apply For Job which Project they want to be join



Employee Can to search a particular selective employee and view their profile



***DATA STRUCTURE MODEL***

EmployeeId int

FirstNam varchar

LastName varchar

UserId varchar

Password varchar

Gender varchar

Role varchar

Active varchar

****

ESId int

EmployeeId int

SkillId int

ExpYear int

****

****

JobId int

JobTitle varchar

JobDescription varchar

CompanyName varchar

Location varchar

KeySkill varchar

Salary int

Active varchar

SkillId int

SkillName varchar

SkillDescription varchar

Active varchar

****

EJId int

EmployeeId int

JobId int

Recruited varchar

****

First\_Name varchar

Last\_Name varchar

E\_Mail varchar

Phone\_Number long

Address varchar

lGender varchar

lDepartment varchar



Employee Tabel:-

Create Database PCSDB;

use PCSDB;

create table Employee( EmployeeId int auto\_increment primary key not null,

FirstName varchar(30) not null,

LastName varchar(30) not null,

UserId varchar(30) not null,

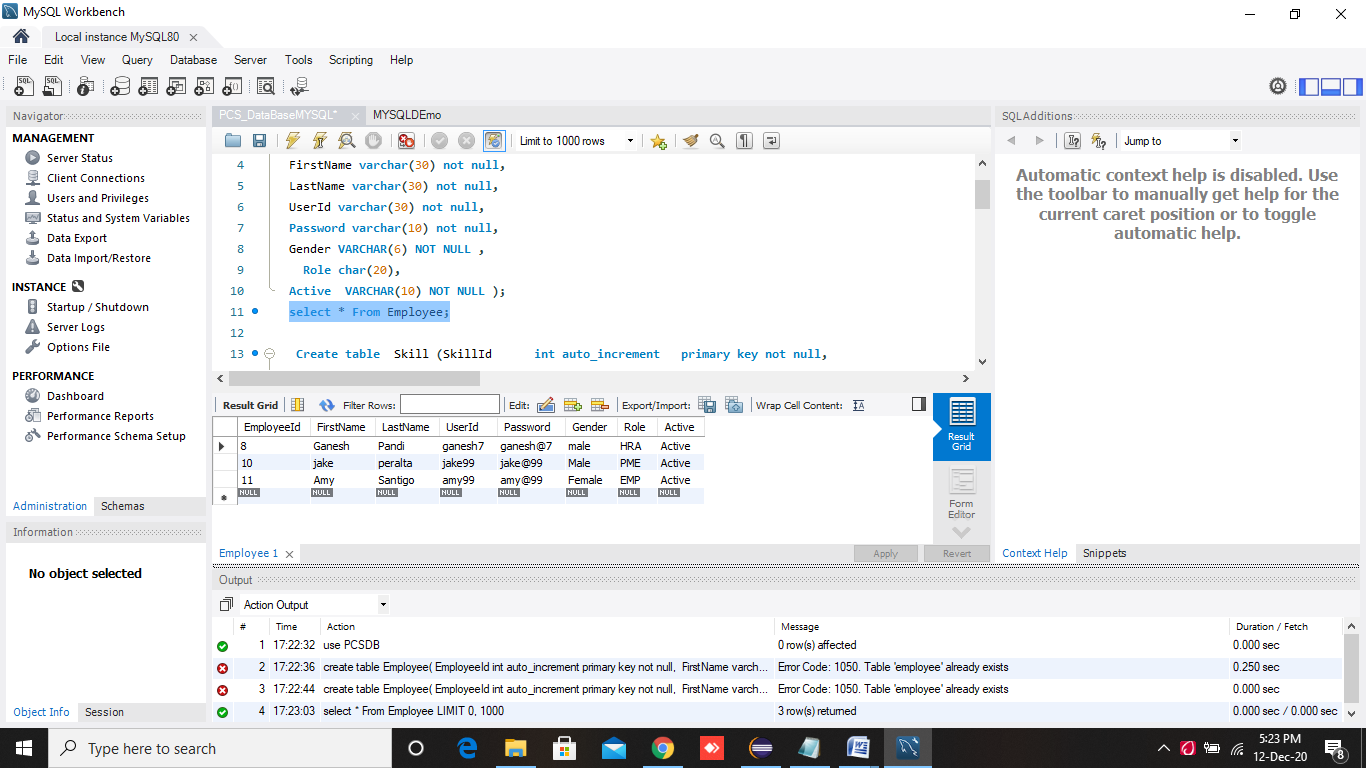
Password varchar(10) not null,

Gender VARCHAR(6) NOT NULL ,

Role char(20),

Active VARCHAR(10) NOT NULL );

select \* From Employee;



Skill Tabel:-

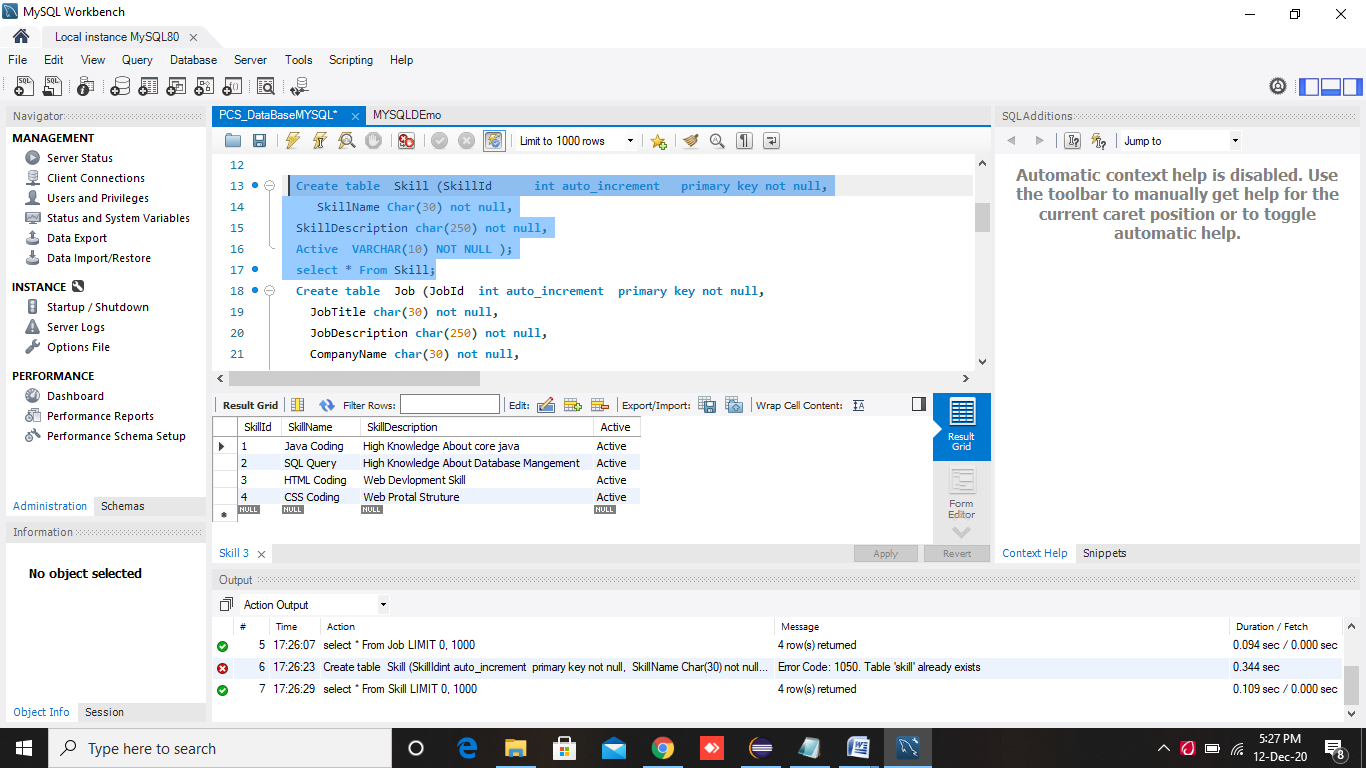
Create table Skill (SkillId int auto\_increment primary key not null,

SkillName Char(30) not null,

SkillDescription char(250) not null,

Active VARCHAR(10) NOT NULL );

select \* From Skill;



Job Tabel:-

Create table Job (JobId int auto\_increment primary key not null,

JobTitle char(30) not null,

JobDescription char(250) not null,

CompanyName char(30) not null,

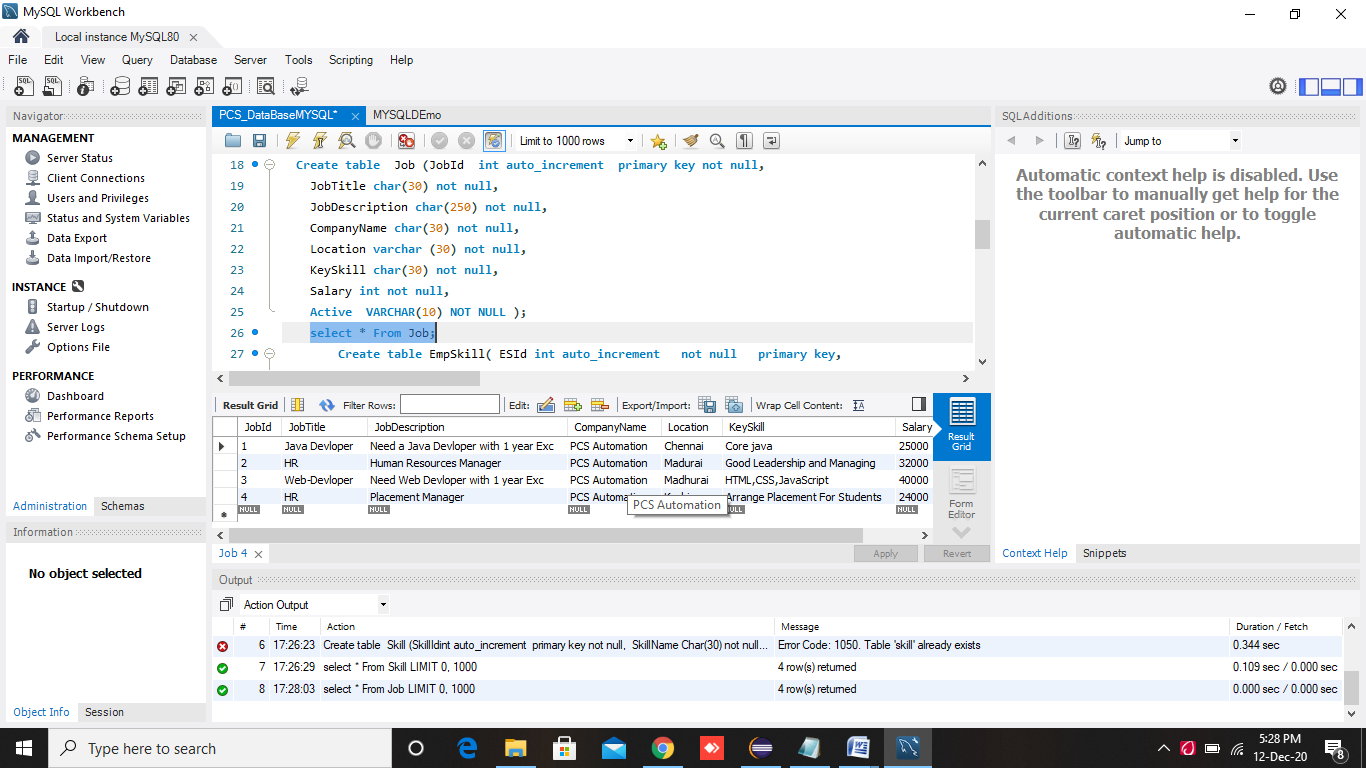
Location varchar (30) not null,

KeySkill char(30) not null,

Salary int not null,

Active VARCHAR(10) NOT NULL );

select \* From Job;



Registration Tabel:-

create table registration(First\_Name varchar(30) not null,

Last\_Name varchar(30) not null,

E\_Mail varchar(30) not null,

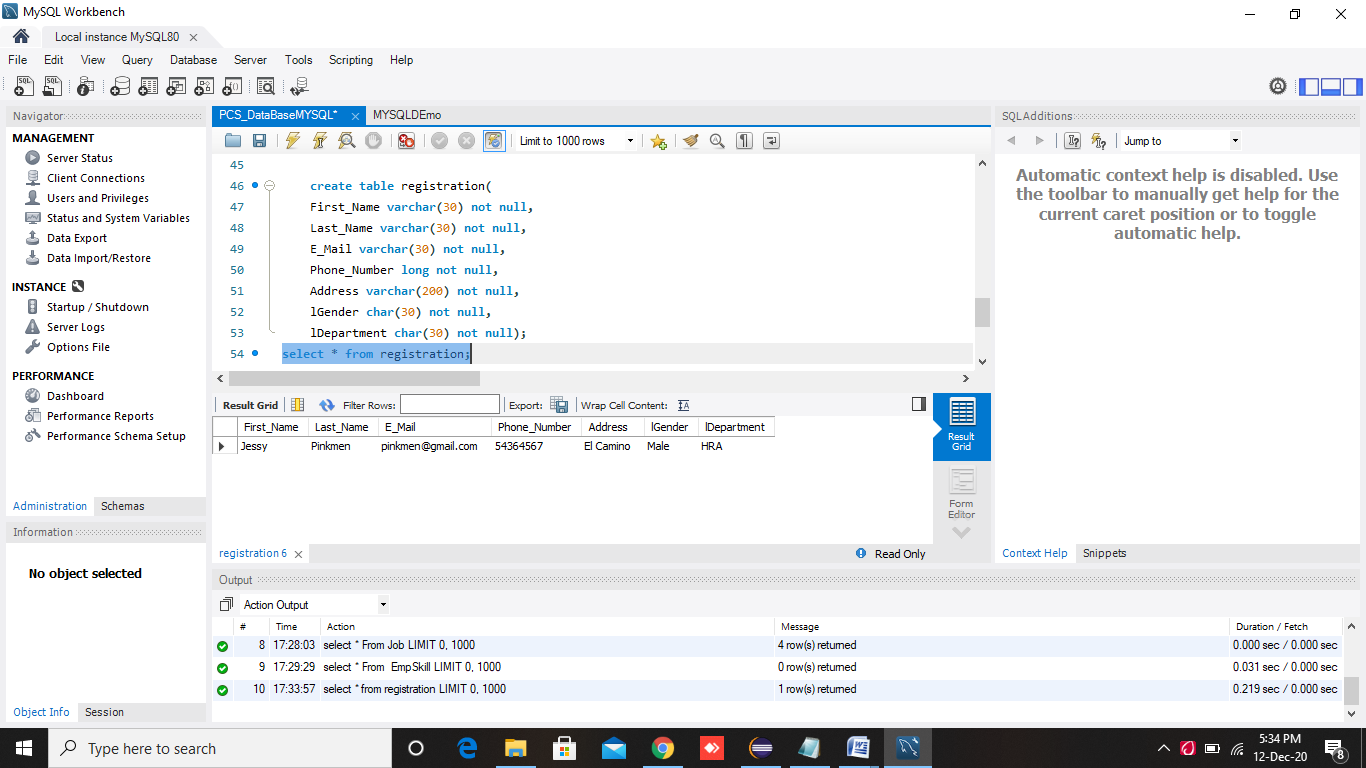
Phone\_Number long not null,

Address varchar(200) not null,

lGender char(30) not null,

lDepartment char(30) not null);

select \* from registration;



Employee Skill Tabel:-

Create table EmpSkill( ESId int auto\_increment not null primary key,

EmployeeId int not null,

SkillId int not null,

ExpYear int not null,

CONSTRAINT FK\_EmployeeID FOREIGN KEY (EmployeeId)

REFERENCES Employee(EmployeeId),

CONSTRAINT FK\_SkillID FOREIGN KEY (SkillId)

REFERENCES Skill(SkillId));

select \* From EmpSkill;

Employee Job Tabel:

Create table EmpJob (EJId int auto\_increment not null primary key,

EmployeeId int not null,

JobId int not null,

Recruited char(30) not null,

CONSTRAINT job\_EmployeeID FOREIGN KEY (EmployeeId)

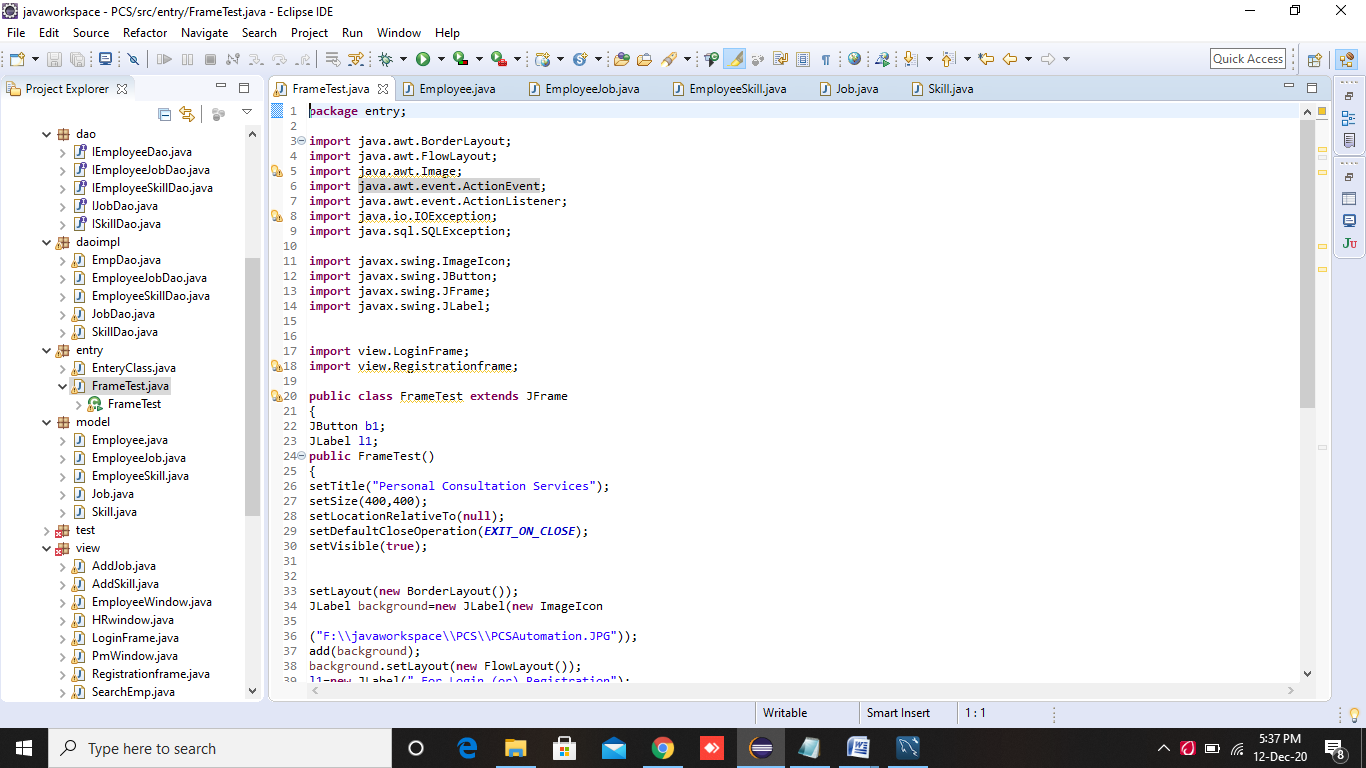
REFERENCES Employee(EmployeeId),

CONSTRAINT FK\_Jobid FOREIGN KEY (JobID)

REFERENCES job(JobId));

select \* From Empjob;

***PROJECT MODEL***

******

This is the project model that I have created on Eclipse IDE. The structure of the project is present on the left most side of the window under Project Explorer. There you will get to see different kind of packages like config , controller, dao, daoimpl, entry, model, test, view .which holds different classes that are categorized in terms of different packages. These are the classes which can handle the entire backend process of the project structure.

Some code snippets are given for your view.

Config package used for the configuration and it loads the JDBC driver to connect to the database server.

**package** config;

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

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

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

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

String username="root";

String password="ganesh@7";

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

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

**return** conn;

}

}

Model package provides the base for the structure.

**package** model;

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

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

**private** String firstName;

**private** String lastName;

**private** String userId;

**private** String password;

**private** String role;

**private** String gender;

**private** String active;

//default constructor methods

**public** Employee() {

}

//parameterized constructor method

**public** Employee(String firstName, String lastName, String userId, String password, String role, String gender) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.userId = userId;

**this**.password = password;

**this**.role = role;

**this**.gender = gender;

}

//All setter and getter methods

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

+ ", password=" + password + ", role=" + role + ", gender=" + gender + ", active=" + active + "]";

}

}

**Empolyee Controller:-**

**package controller;**

**import java.io.\*;**

**import model.Employee;**

**import model.EmployeeJob;**

**import dao.IEmployeeDao;**

**import dao.IEmployeeJobDao;**

**import daoimpl.EmpDao;**

**import daoimpl.EmployeeJobDao;**

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

**import java.util.List;**

**public class EmployeeController {**

**IEmployeeJobDao empjobdao= null;**

**IEmployeeDao empDao=null;**

**public EmployeeController() throws ClassNotFoundException, SQLException{**

**empDao=new EmpDao();**

**empjobdao=new EmployeeJobDao();**

**}**

**public Employee checkLogin(String userId, String password) {**

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

**return emp;**

**}**

**public void addEmployee() {**

**Employee emp=new Employee();**

**try {**

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

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

**emp.setFirstName(reader.readLine());**

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

**emp.setLastName(reader.readLine());**

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

**emp.setUserId(reader.readLine());**

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

**emp.setPassword(reader.readLine());**

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

**emp.setGender(reader.readLine());**

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

**String role=reader.readLine();**

**emp.setRole(role);**

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

**emp.setActive("Active");**

**}**

**else {**

**emp.setActive("Deactive");**

**}**

**empDao.addEmployee(emp);**

**}**

**catch(IOException ex) {**

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

**}**

**}**

**public List<Employee> getAllEmployee() {**

**List<Employee> allEmpList=empDao.getAllEmployee();**

**return allEmpList;**

**}**

**public Employee getEmployeeById(String EmpId) {**

**int id;**

**id=Integer.parseInt(EmpId);**

**Employee emp=empDao.getEmployeeById(id);**

**return emp;**

**}**

**public Employee updateEmployee(String newpassword , int empid) {**

**Employee emp=new Employee();**

**emp.setPassword(newpassword);**

**emp.setEmpId(empid);**

**empDao.updateEmployee(emp);**

**return emp;**

**}**

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

**int id;**

**id=empId;**

**Employee emp=empDao.getEmployeeById(id);**

**empDao.deactivateEmployee(emp);**

**}**

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

**int id;**

**id=empId;**

**Employee emp=empDao.getEmployeeById(id);**

**empDao.activateEmployee(emp);**

**}**

**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 addEmployeeJob(int empId, int jobId, String recruited) {**

**EmployeeJob Empjob=new EmployeeJob();**

**Empjob.setEmployeeId(empId);**

**Empjob.setJobId(jobId);**

**Empjob.setRecruited(recruited);**

**empjobdao.addEmployeeJob(Empjob);**

**}**

**public List<EmployeeJob> getAllEmployeeJob() {**

**List<EmployeeJob> allEmpJobList=empjobdao.getAllEmployeeJob();**

**return allEmpJobList;**

**}**

**}**

**Employee Interface “IEmployee”:**

**package dao;**

**import model.Employee;**

**import java.util.List;**

**public interface IEmployeeDao {**

**List<Employee> getAllEmployee();**

**void addEmployee(Employee emp);**

**Employee getEmployeeById(int id);**

**void updateEmployee(Employee emp);**

**void deactivateEmployee(Employee Emp);**

**void deleteEmployee(int id);**

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

**void activateEmployee(Employee emp);**

**}**

**Employee Data access object Implements:**

**package daoimpl;**

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

**import java.util.ArrayList;**

**import java.util.List;**

**import config.JDBCConnection;**

**import dao.IEmployeeDao;**

**import model.Employee;**

**public class EmpDao implements IEmployeeDao{**

**Connection conn=null;**

**public EmpDao() 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.setEmpId(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 List<Employee> getAllEmployee() {**

**List<Employee> allEmpList=new ArrayList<Employee>();**

**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.setEmpId(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);**

**}**

**}**

**}**

**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=new Employee();**

**try{**

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

**pst.setInt(1, id);**

**ResultSet rst=pst.executeQuery();**

**if(rst!=null) {**

**if(rst.next()) {**

**emp.setEmpId(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**

**{**

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

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

**pst.setInt(2, emp.getEmpId());**

**int i=pst.executeUpdate();**

**if(i==1) {**

**System.out.println("1 Record Updated....");**

**}**

**else {**

**System.out.println("Updation failed...");**

**}**

**}**

**catch(SQLException ex) {**

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

**}**

**}**

**@Override**

**public void deactivateEmployee(Employee emp) {**

**try {**

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

**pst.setString(1, "Deactive");**

**pst.setInt(2, emp.getEmpId());**

**int i=pst.executeUpdate();**

**}**

**catch(SQLException ex) {**

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

**}**

**}**

**@Override**

**public void deleteEmployee(int id) {**

**try{**

**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());**

**}**

**}**

**@Override**

**public void activateEmployee(Employee emp) {**

**try {**

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

**pst.setString(1, "Active");**

**pst.setInt(2, emp.getEmpId());**

**int i=pst.executeUpdate();**

**}**

**catch(SQLException ex) {**

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

**}**

**}**

**}**

**PERSONAL OBSERVATION**

* Spending enough time interacting to get the needed data as much as possible.
* Reducing the problem of reactivity. Gaining intuitive understanding of the meaning of the structure.
* Addressing problems that are received while compiling the code snippets and sort out the possible solutions.
* Moreover, I personally enjoyed while working in this project and gained more knowledge, boosts my confidence and brings me to successfully submit the project on time.

**Software project developers are always open to something new, but they are not oracles and cannot foresee all the issues that may arise on the way. Moreover, many problems are not an obstruction, but a chance to see the entire project from a different point view. Apply all our responsibility, attention, skills, and mental flexibility, embrace problem-solving and we will always come up with the best possible solution for any problem.**