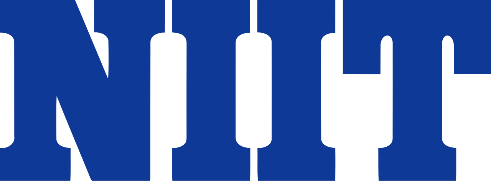


|  |
| --- |
| JOB POSTING- RECRUITMENT PORTAL |
|  |
| DECEMBER 12  Developed by: Kaniyan Pandiarasan K  Reg No: R200002100292 |



Batch Code: 210167

Name of the Cordinator: Ms. Lopamudra Bera

Date of Submission: 12/12/2020

# CERTIFICATE

|  |
| --- |
| *This is to certify that this report, titled Job Posting – Recruitment Portal embodies the original work done by Kaniyan Pandiarasan K, in partial fulfillment of his course requirement at NIIT.* |
| Coordinator: Ms. Lopamudra Bera |

# ACKNOWLEDGEMENT

|  |
| --- |
| *It gives me great pleasure in expressing my gratitude to all those people who have supported us and had their contribution in making this project work possible. First and foremost, I must acknowledge and thank the Almighty for blessing, protecting and guiding me throughout this period.*  *I express my profound sense of reverence to my Coordinator Ms. Lopamudra Bera, for her constant guidance, support, motivation and untiring help during the course of my project work. She has given us enough freedom during our project work, and she has always been nice to us. I am thankful to the Almighty for giving us a teacher like her.* |
| 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* | |

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| |  | | --- | | Chapter 1 – Introduction   * 1. Objective   2. Case Study   Chapter 2 – Project Requirement Specification  2.1. Statement of requirement  Chapter 3 – Project Analysis  3.1. Project Plan  3.2. Business Model  3.3. High Level Use Case Diagram  Chapter 4 – Project Design  4.1. Low Level Use Case Diagram  4.2. User Interface Design  4.3. Data Structure Model  4.4. Data Model  4.5. Project Structure  Chapter 5 – Project Testing  5.1. Test Cases  5.2. Personal Observation | |

# 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. | |

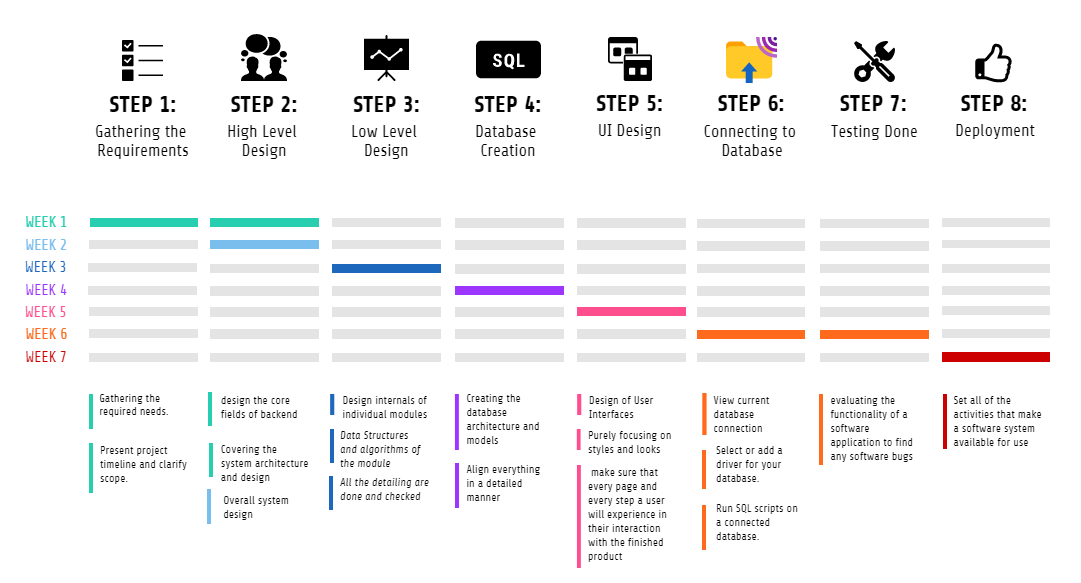
# CASE STUDY

|  |  |
| --- | --- |
| |  | | --- | | 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 mapping feature 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. | |

# PROJECT PLAN



# BUSINESS MODEL

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

# HIGH LEVEL CASE DIAGRAM

MYSQL

DATABASE

Discussing the needed skills

PM

EMP

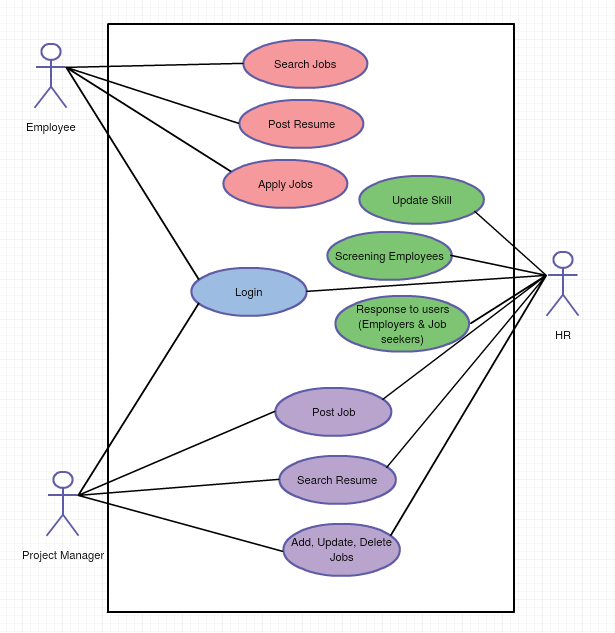
JOB POSTING

Screening and recruiting

HR

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.

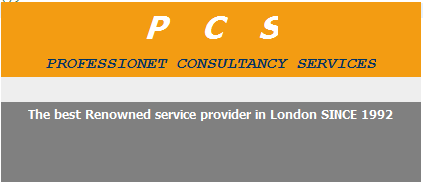
Project Manager can post the job necessary for the organization and monitor the cycle. He can able to activate and deactivate a job according to the situation of the company.

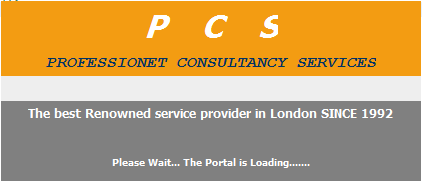
Similarly, HR can monitor the profiles who are applied for the particular jobs and screening them by matching the skill sets to pick a suitable employee to fit the project.

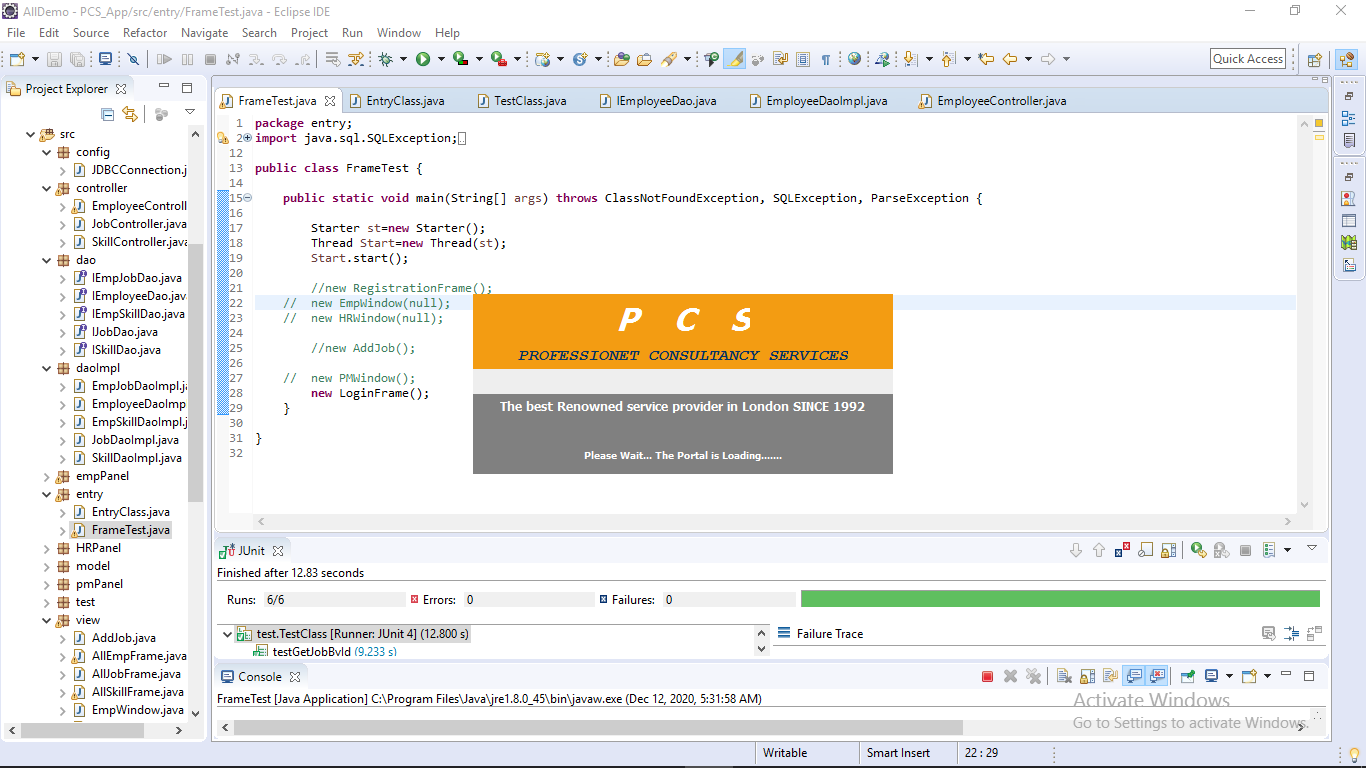
# USER INTERFACE DESIGN

The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website.

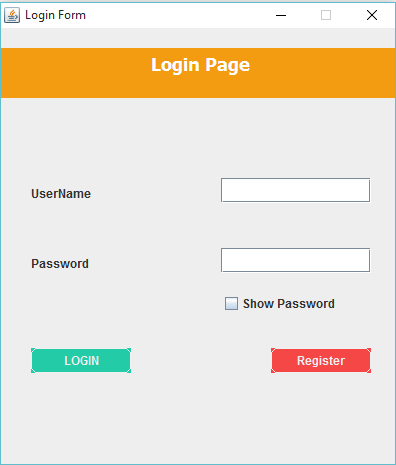
While stating up the software it displays a frameless window until the database is connecting to the software.





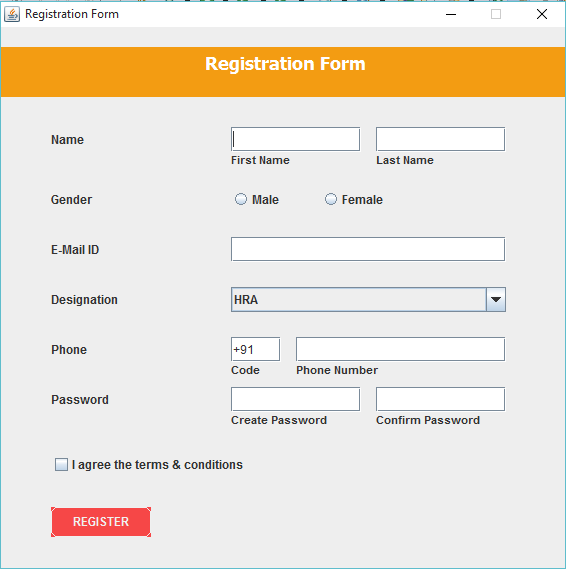


This is how the actual window looks like on the desktop. It will take 6 to 8 seconds to connect to the database. In that mean time, this frameless window will be displayed and once the database loaded, it will automatically close by itself and leads you to the login frame. That will be shown in the next image.

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

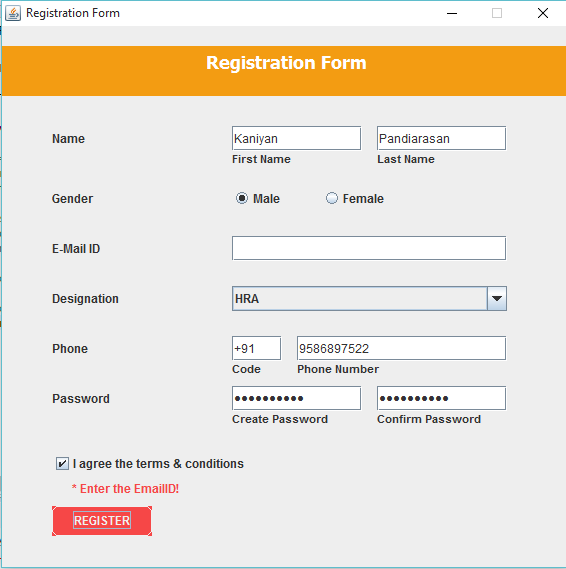
The user has to be registered as PCS Employee before login. This frame will connect to the database and checks the user’s authentication and also decides which designation they belong to (EMP, PM, HR) and will give access to the user accordingly.

This is the Registration form that the user has to fill their details to get registered with the PCS Community.



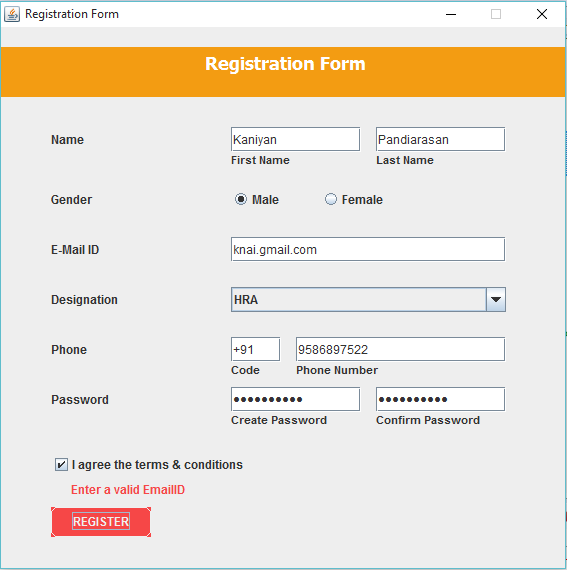
The System must not miss out any kind of information while dealing with the registration form. So, the authentication is very much important for the organization. The registration form would check each and every column for null value and alert the user to fill that missing field and also it makes sure that the entered E-mail id, phone number, password is a valid one and meet the system requirements.

Here are some snaps with explanation.



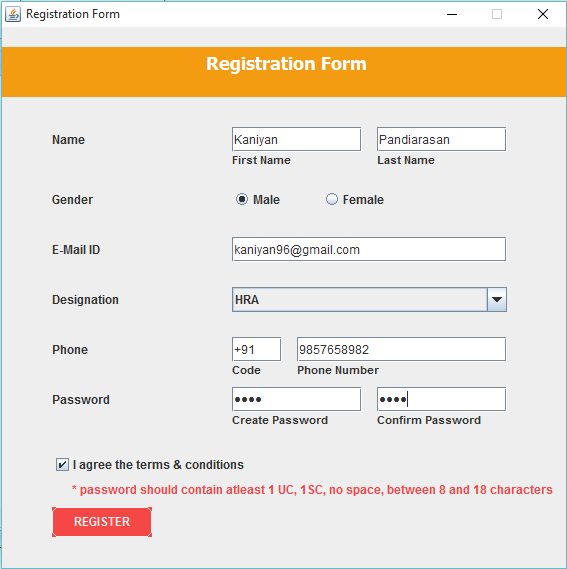
In above form, E-mail id field is missed to fill. And if you click the register button, it will throw an indication stating that to fill the required field to complete registration.

If the user enters the invalid mail address then a message will be displayed stating that to enter the valid email ID. Following is the example:

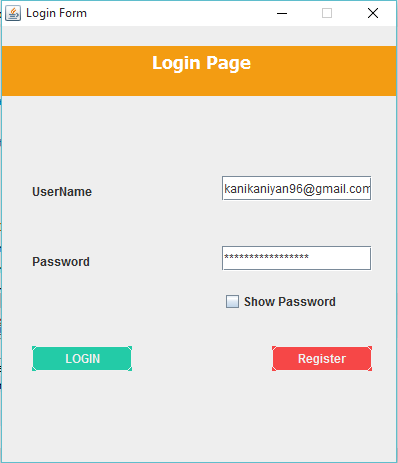


The same will be checked for validating the phone number. The phone number column has to have 10 digits, doesn’t accept the letters and should start with 7, 8, and 9.

Here comes the password validation. The password has to have 8-18 characters with no space and contains at least one uppercase, one lowercase, one special character and one number. If it entered wrong, a message will be displayed on the screen to enter a valid password.

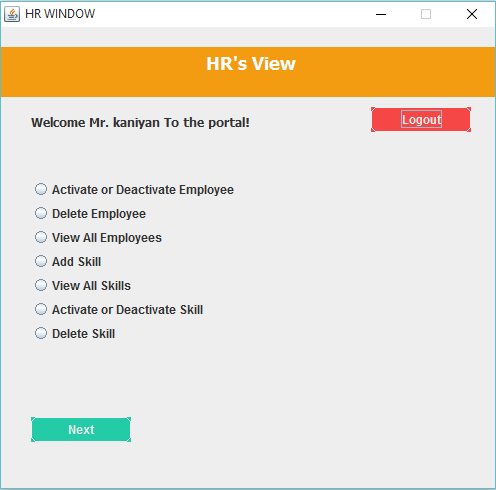


If the password matches with the required criteria, then it will cross check with the confirm password to make sure that the user is entering the valid known password. In case, password doesn’t match with the confirm password, then a message would be displayed as password doesn’t match.



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 needs to choose that the user is an employee / Project Manager / HR. It will walk us through which portal we are having the access to.

HR Window

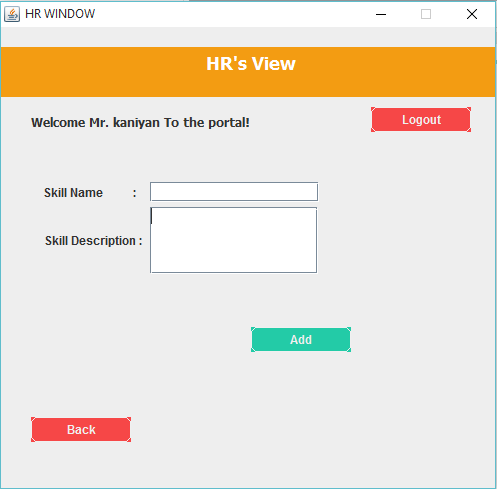


This is how the HR Window looks like. Nicely designed user interface for the HR’s View. Here, can see that what HR can have the access to do.

HR can

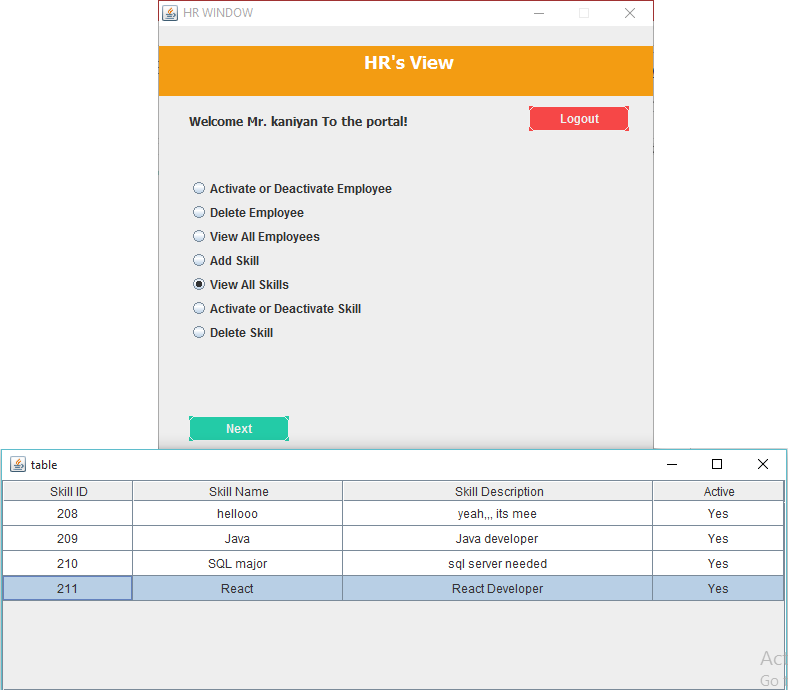
* Activate or deactivate employee
* Delete an employee
* View all employees
* Add a necessary skill
* View all available skills in the database
* Activate, deactivate or delete a skill.

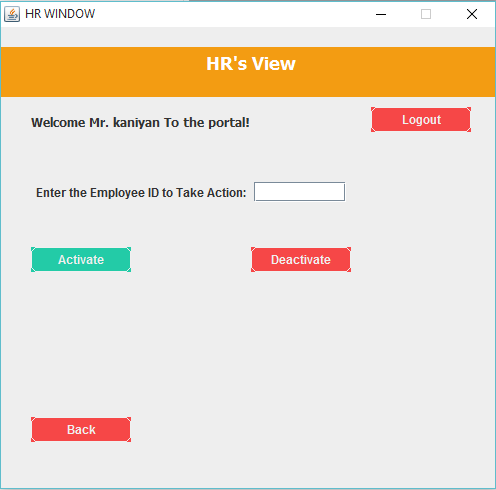
HR can able to add a skill.

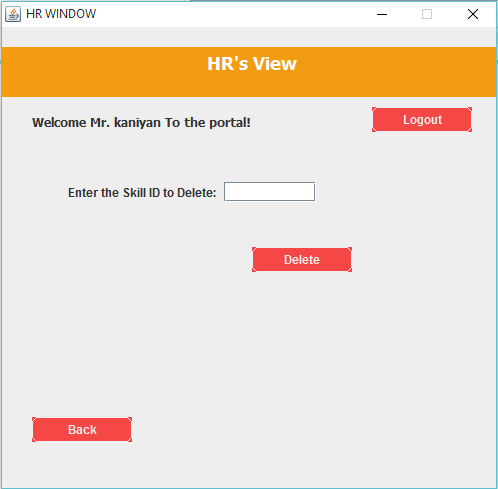


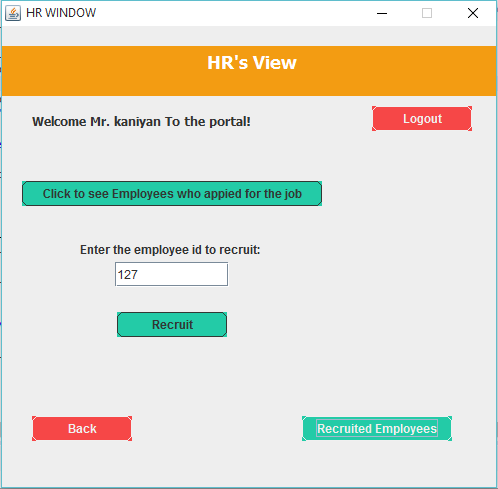
HR can able to add the skill by entering the skill name and the skill description. It will be added to the database and immediately it will reflects on the application.

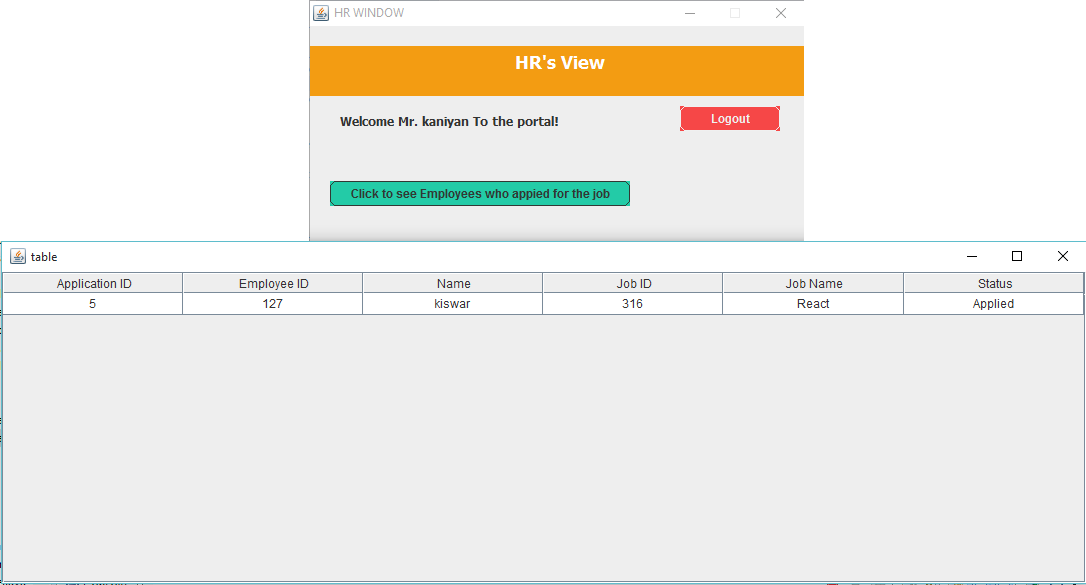
HR can see all the skills that includes deactivated skills to check the data.

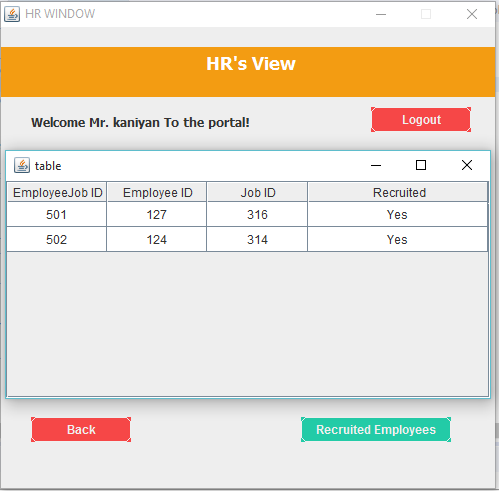


Activate, deactivate and delete Employee/skill with employee ID/skill ID.



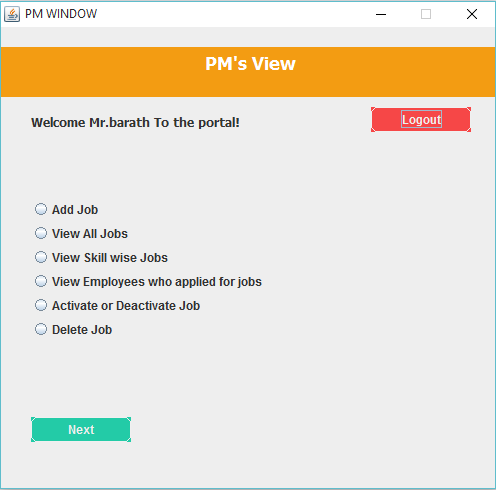
HR’s Recruitment view





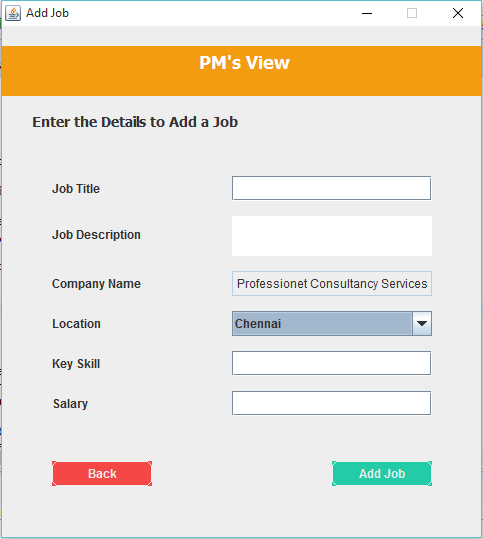
This the recruited employees by the HR by entering the employee ID on the Text Field.

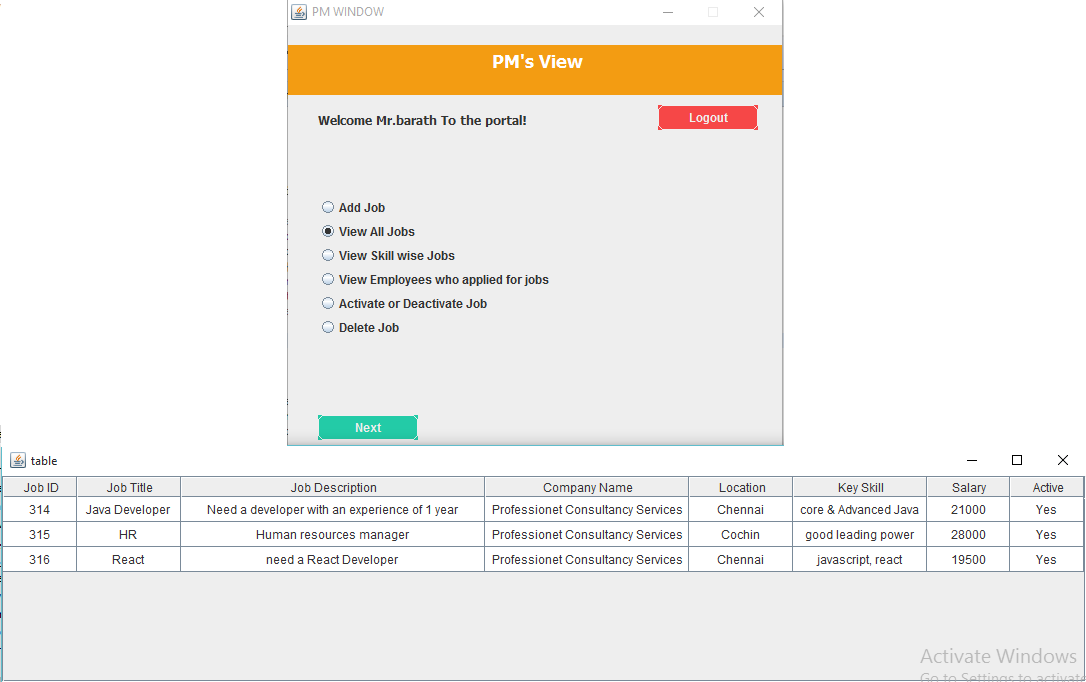
Project Manager’s View

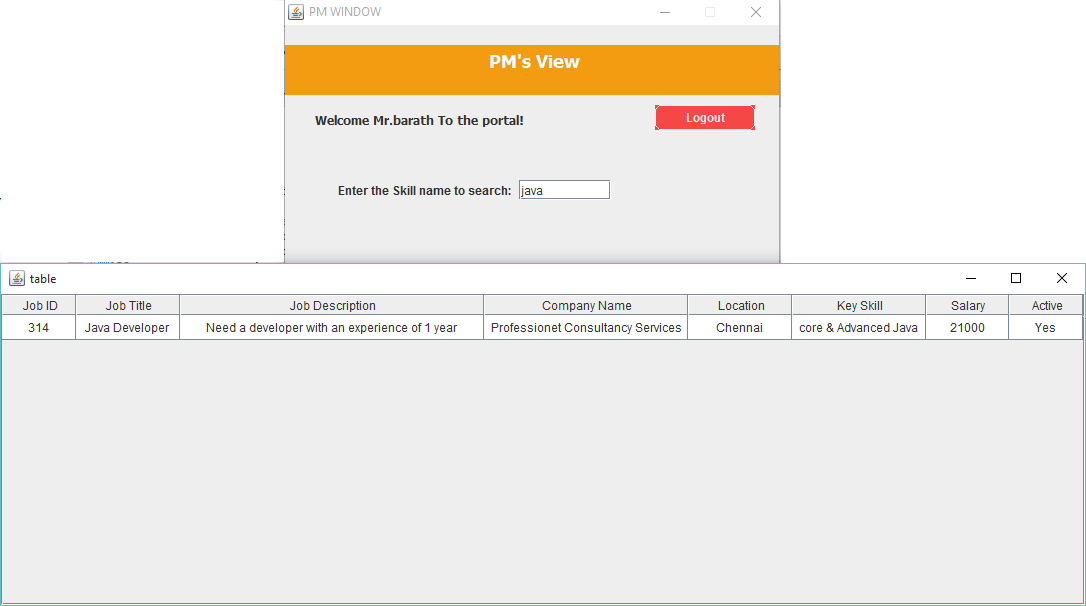


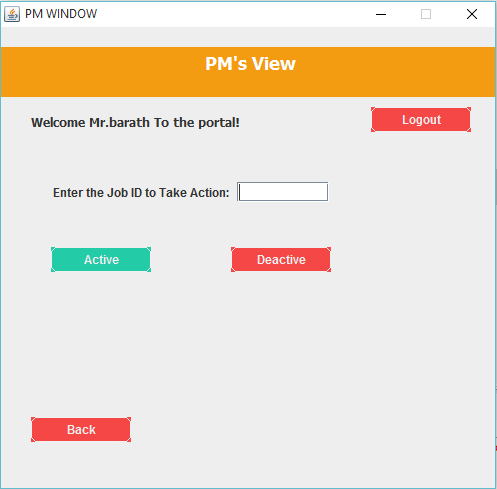
Nice and sleekly designed user interface that allows the managers to take navigate through the process quickly and effectively. The above pictures shows that what a PM capable of doing in that organization. He is responsible for the entire process of monitoring employees who belongs to that firm.

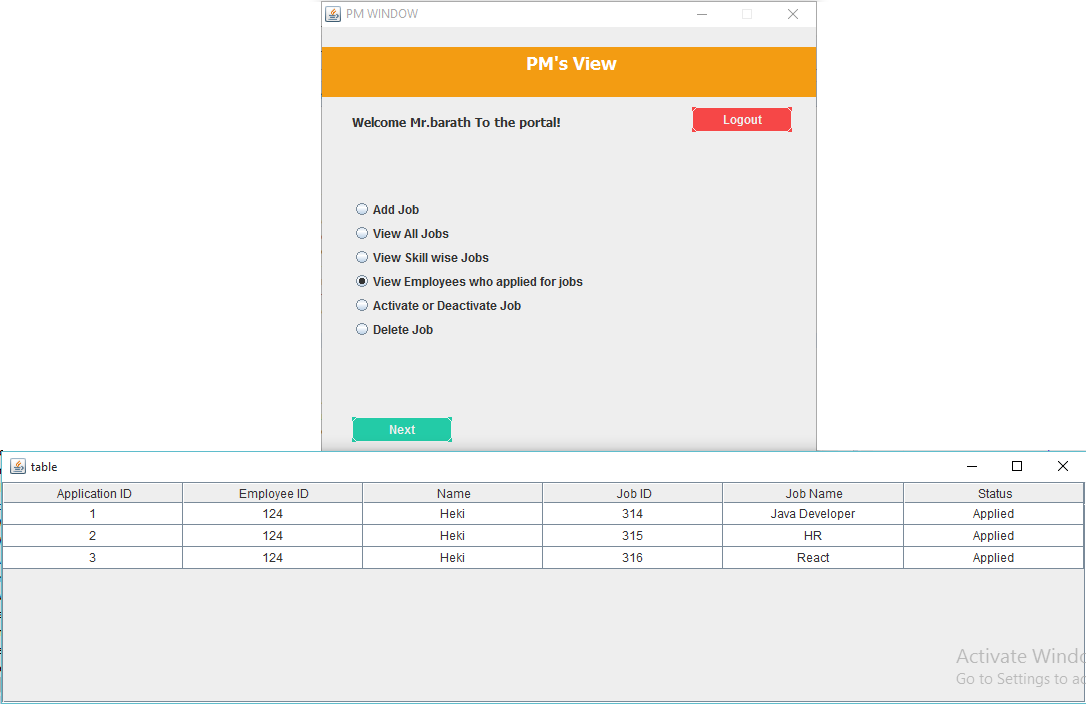
PM can add a Job by entering the necessary details.



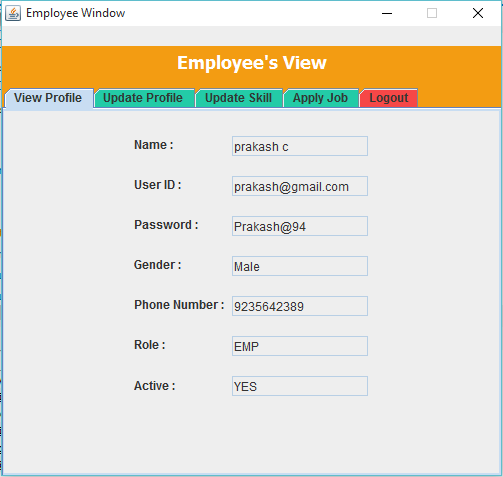
PM can able to view all the jobs that he has added to maintain the circle.

He can filter the jobs using the keywords.

PM can activate and deactivate the posted job in the portal.



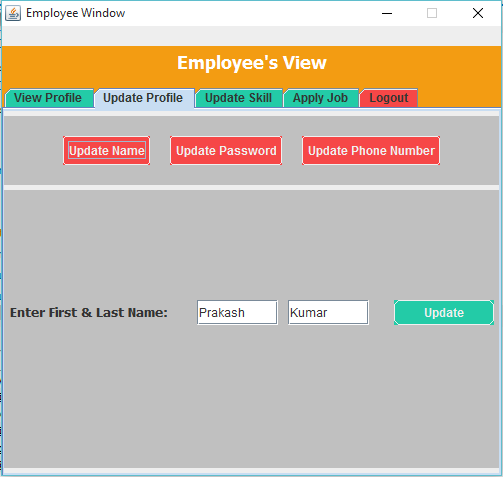
Manager can monitor the employees who have applied for the job and manage the job flow and response to that particular project.

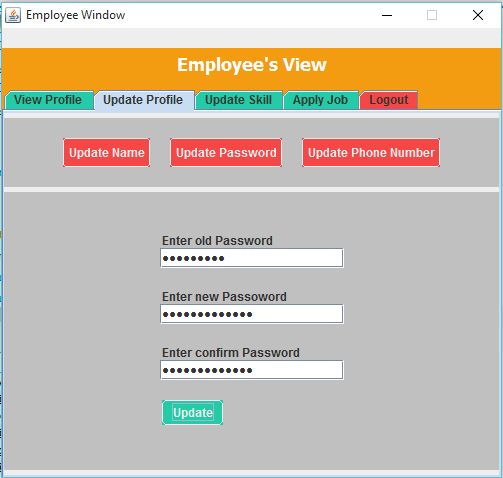
Employee’s View

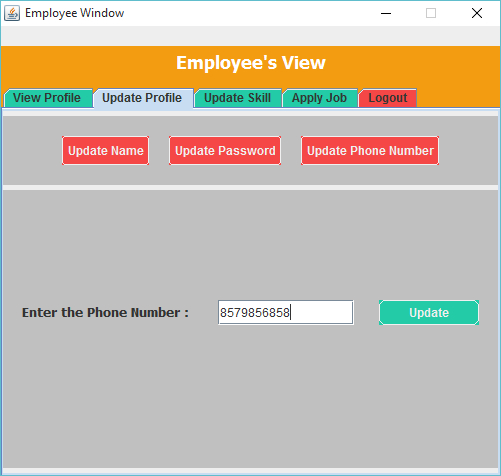
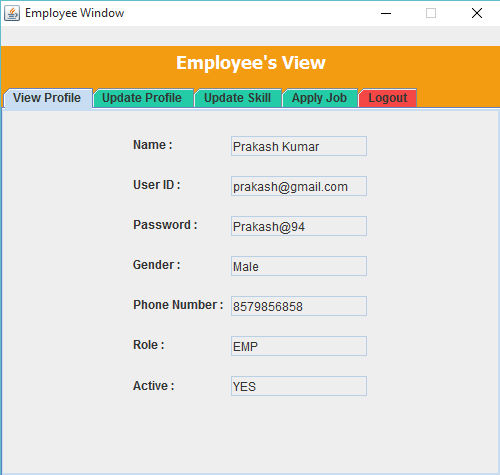
Once you login as an employee, you will be automatically directed to your profile view.

Employee’s view can give access to the employees to view their profile. Update their profile and skill and also apply the available jobs.

Update profile:





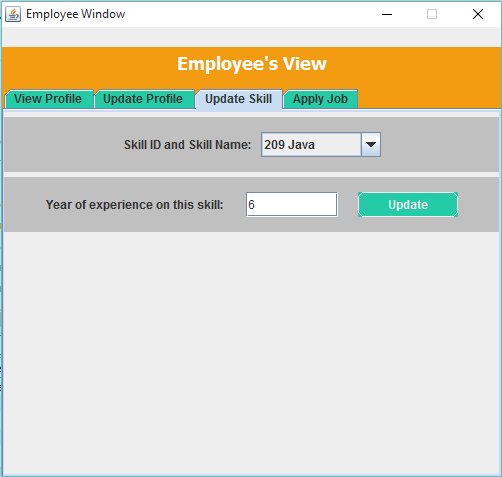


The above four images belongs to update profile in which the user can able to update their name, password and phone number.

When comes to the password, authentication is really important that some else in the room can able to change your password and access your data. So, we makes sure that the user has to enter the old password to change to a new one. The old password entered by the user will be crossed checked with the database. Once the authentication is approved by the system, the user can change their password. And similarly, for the phone number, the user has to enter 10 digit number and that has to starts with 7, 8 and 9 respectively.

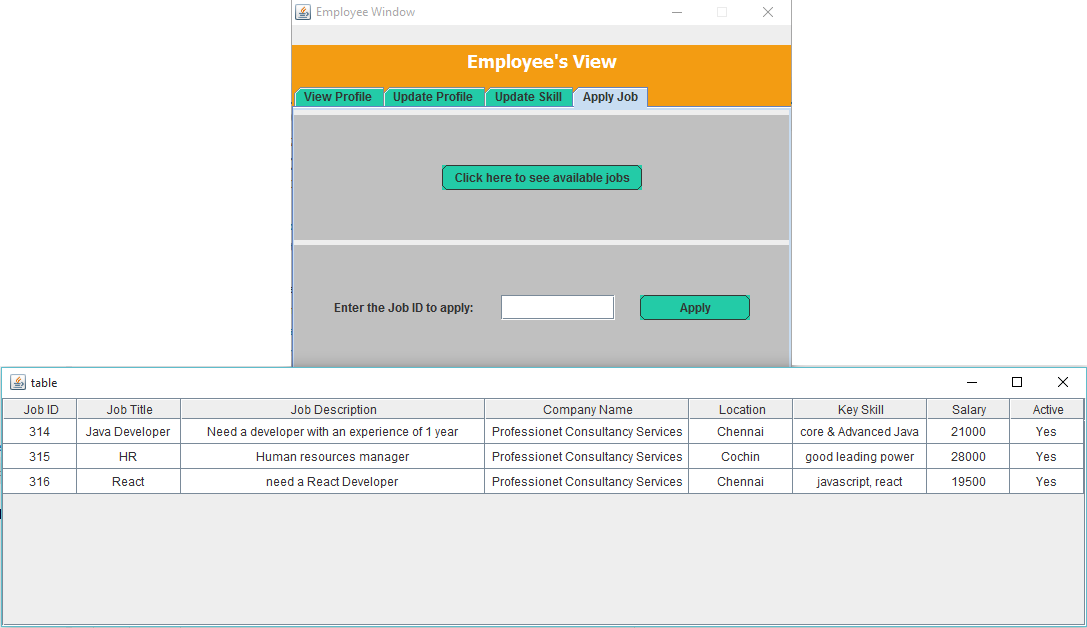
As you see on the updating images, every update will reflect immediately on their portal and they need not logout and login again.

Skill Updating:

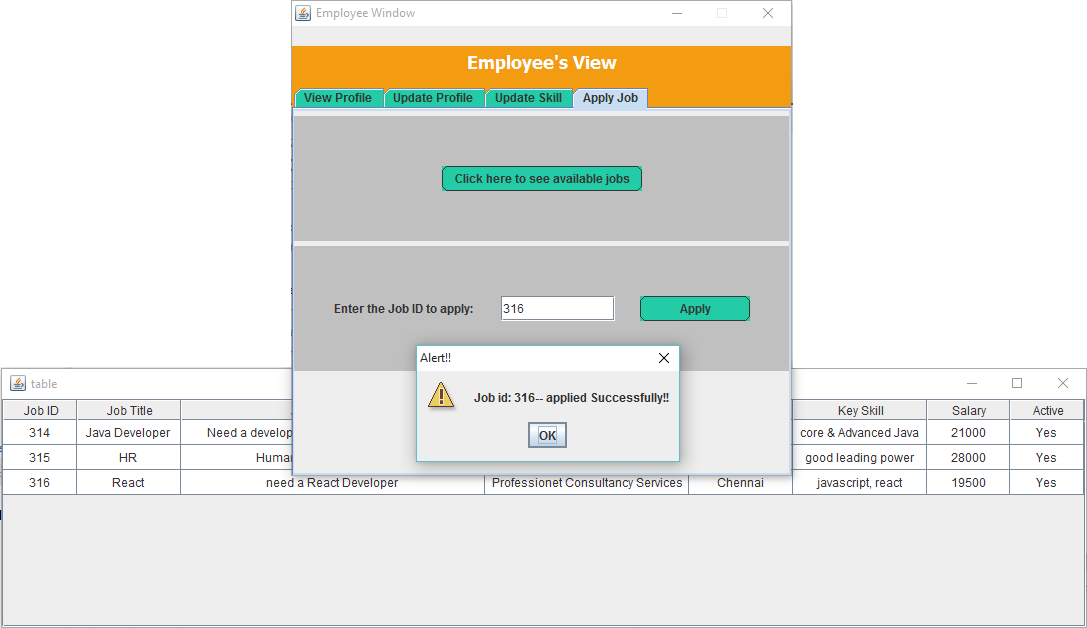


Employee can able to update their skill which are made available and required by the HR and PM. In this panel, the user required to select the skill id and name of the skill along with the number of professional experience the user having. And the user can update the skill.

Apply Job:



In the apply job panel, the user can have the access to view the available jobs on the portal and accordingly the one can apply for the job by entering the job id with ease. Below is the successful application.



# DATA STRUCTURE MODEL

Employee

c

EmployeeID int

FirstName varchar

LastName varchar

UserID varchar

Password varchar

Gender varchar

PhoneNumber Long

Role varchar

Active char

Skill

c

SkillID int

SkillName varchar

SkillDescription varchar

Active char

Job

c

JobID int

JobTitle varchar

JobDescription varchar

CompanyName varchar

Location varchar

KeySkill varchar

Salary numeric

Active varchar

EmpSkill

c

ESID int

EmployeeID int

SkillID int

expYear int

EmpJob

c

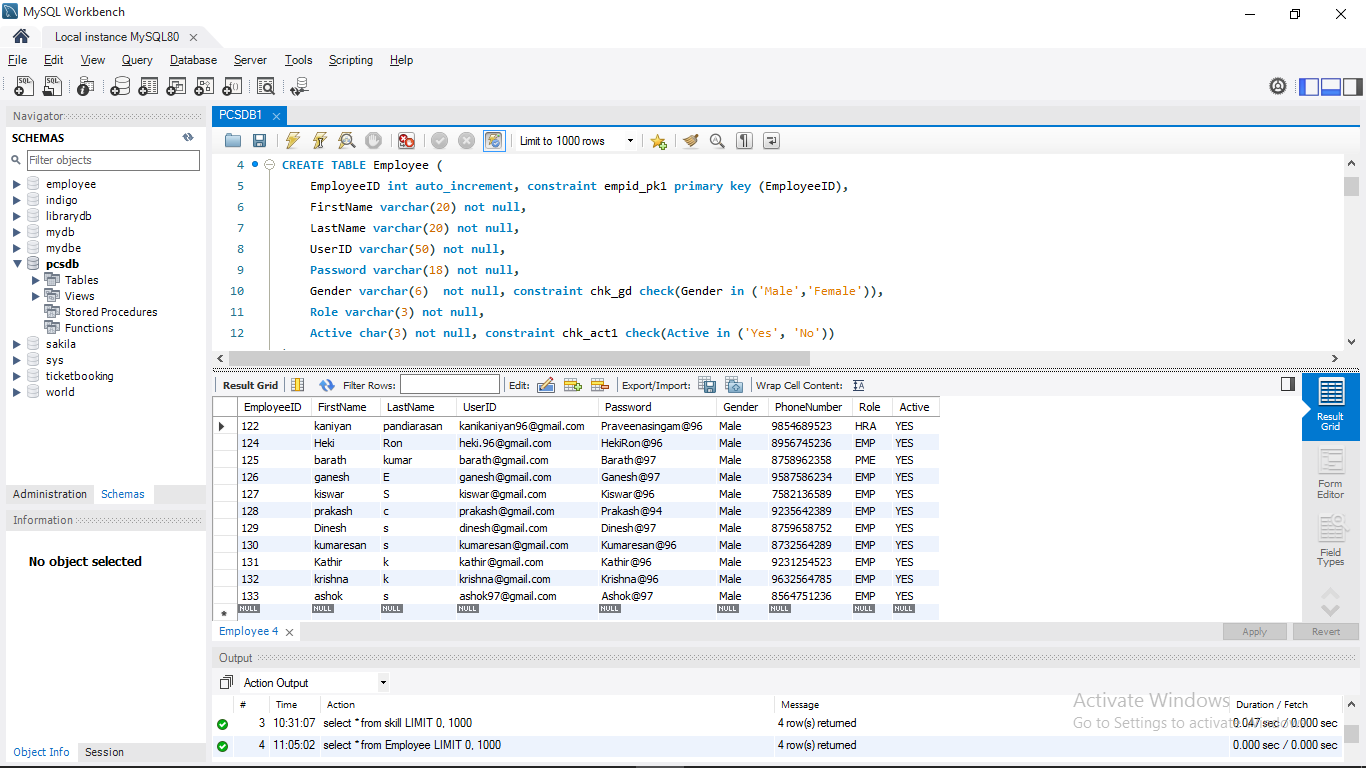
EJID int

EmployeeID int

JobID int

Recruited varchar

# DATA MODEL

EMPLOYEE TABLE:

Create database PCSDB;

Use PCSDB;

CREATE TABLE Employee (

EmployeeID int auto\_increment, constraint empid\_pk1 primary key (EmployeeID),

FirstName varchar (20) not null,

LastName varchar (20) not null,

UserID varchar (50) not null,

Password varchar (18) not null,

Gender varchar (6) not null, constraint chk\_gd check (Gender in ('Male’, ‘Female')),

PhoneNumber long not null,

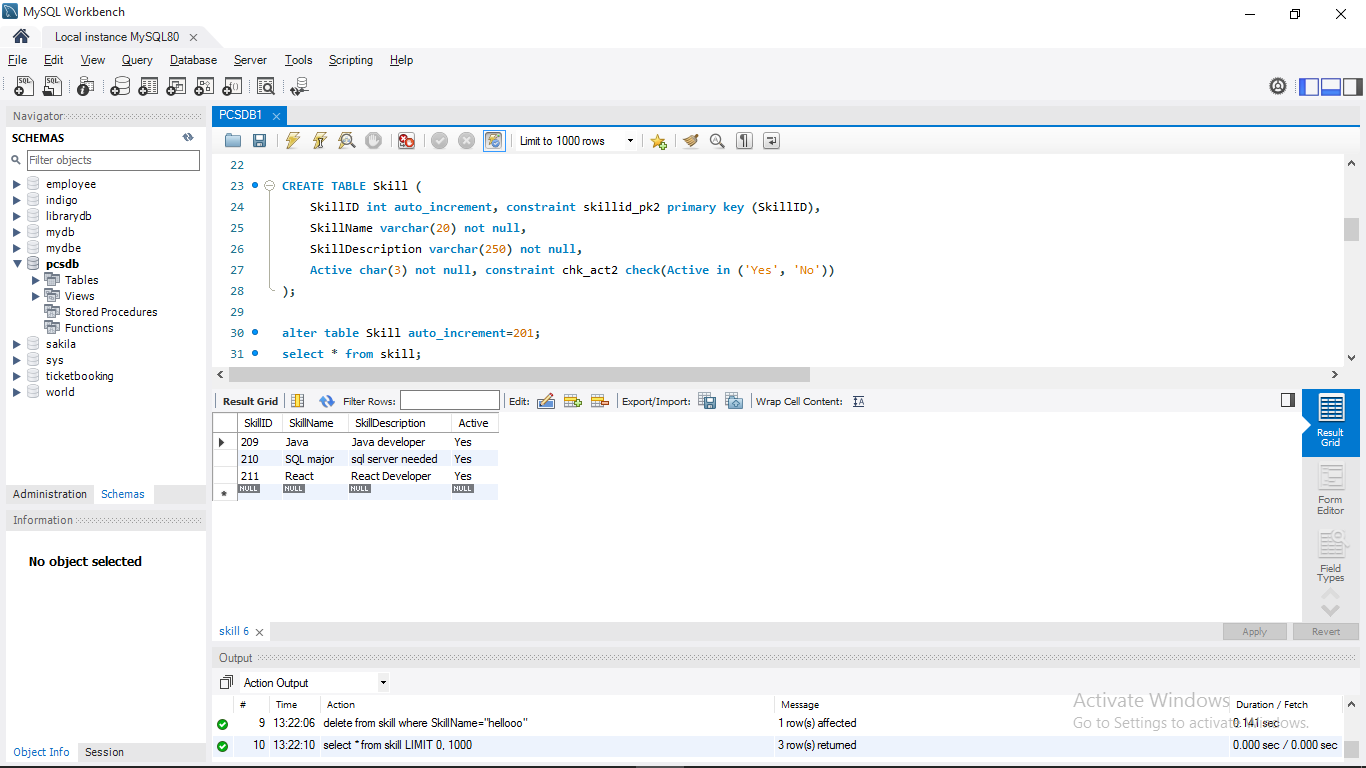
Role varchar (3) not null,

Active char (3) not null, constraint chk\_act1 check (Active in ('Yes', 'No'))

);

Select \* from Employee;

SKILL TABLE:



CREATE TABLE Skill (

SkillID int auto\_increment, constraint skillid\_pk2 primary key (SkillID),

SkillName varchar (20) not null,

SkillDescription varchar (250) not null,

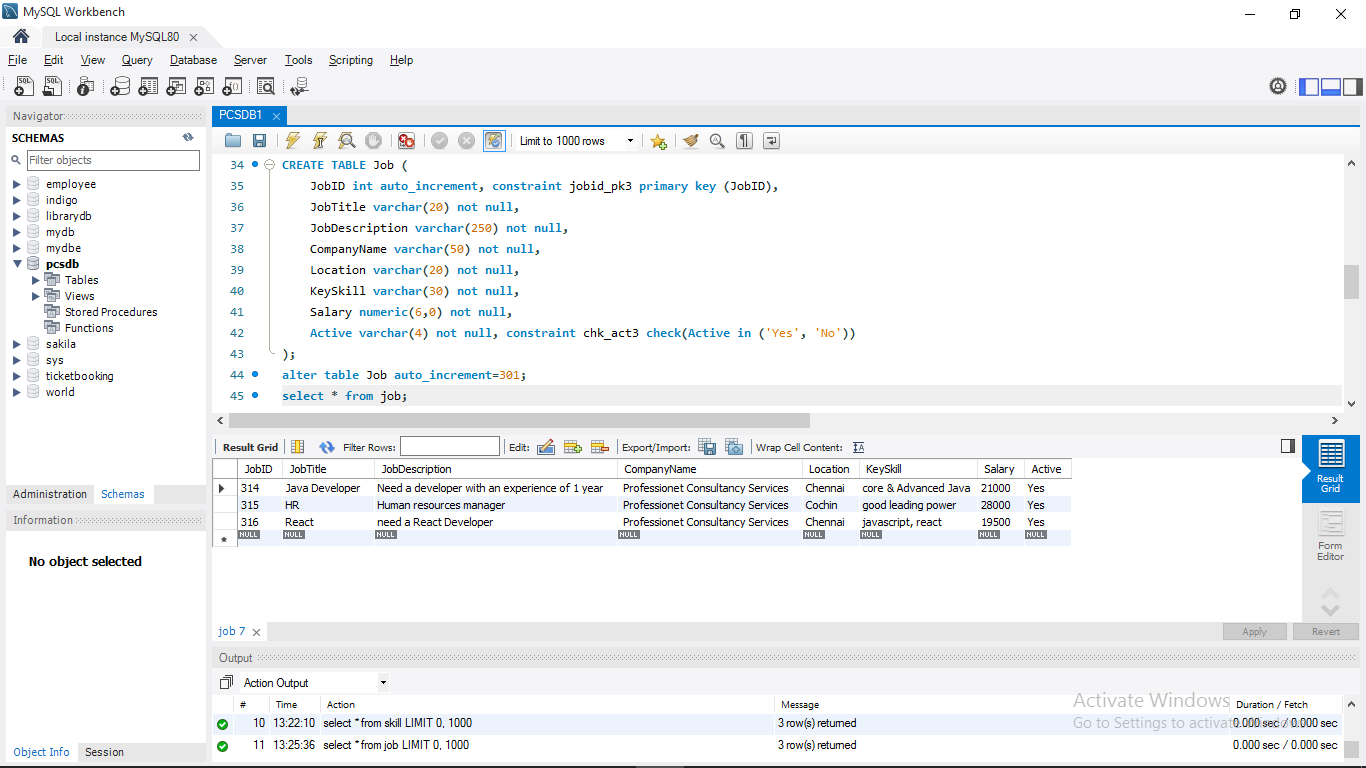
Active char (3) not null, constraint chk\_act2 check (Active in ('Yes', 'No'))

);

Alter table Skill auto\_increment=201;

Select \* from skill;

JOB TABLE:



CREATE TABLE Job (

JobID int auto\_increment, constraint jobid\_pk3 primary key (JobID),

JobTitle varchar (20) not null,

JobDescription varchar (250) not null,

CompanyName varchar (50) not null,

Location varchar (20) not null,

KeySkill varchar (30) not null,

Salary numeric (6,0) not null,

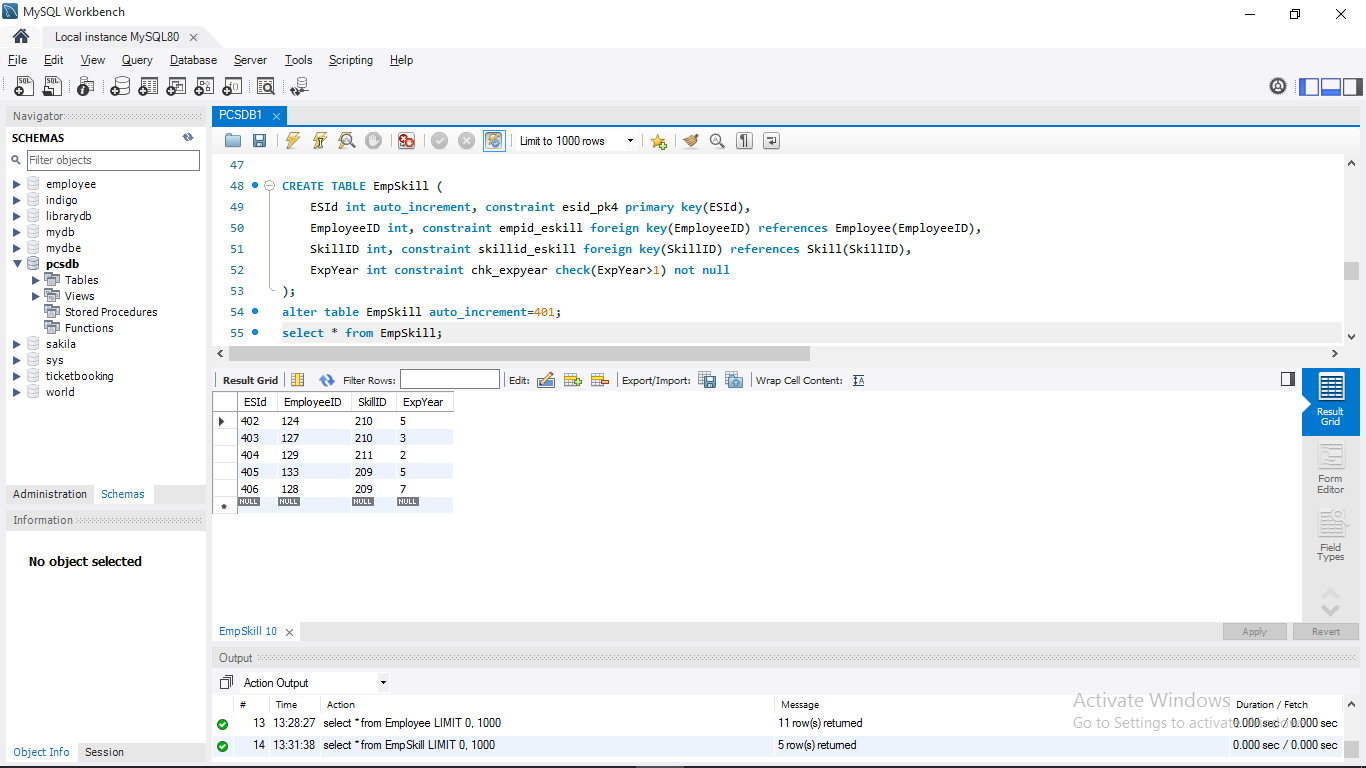
Active varchar (4) not null, constraint chk\_act3 check(Active in ('Yes', 'No'))

);

Alter table Job auto\_increment=301;

Select \* from job;

EMPSKILL TABLE:



CREATE TABLE EmpSkill (

ESId int auto\_increment, constraint esid\_pk4 primary key (ESId),

EmployeeID int, constraint empid\_eskill foreign key (EmployeeID) references Employee (EmployeeID),

SkillID int, constraint skillid\_eskill foreign key (SkillID) references Skill (SkillID),

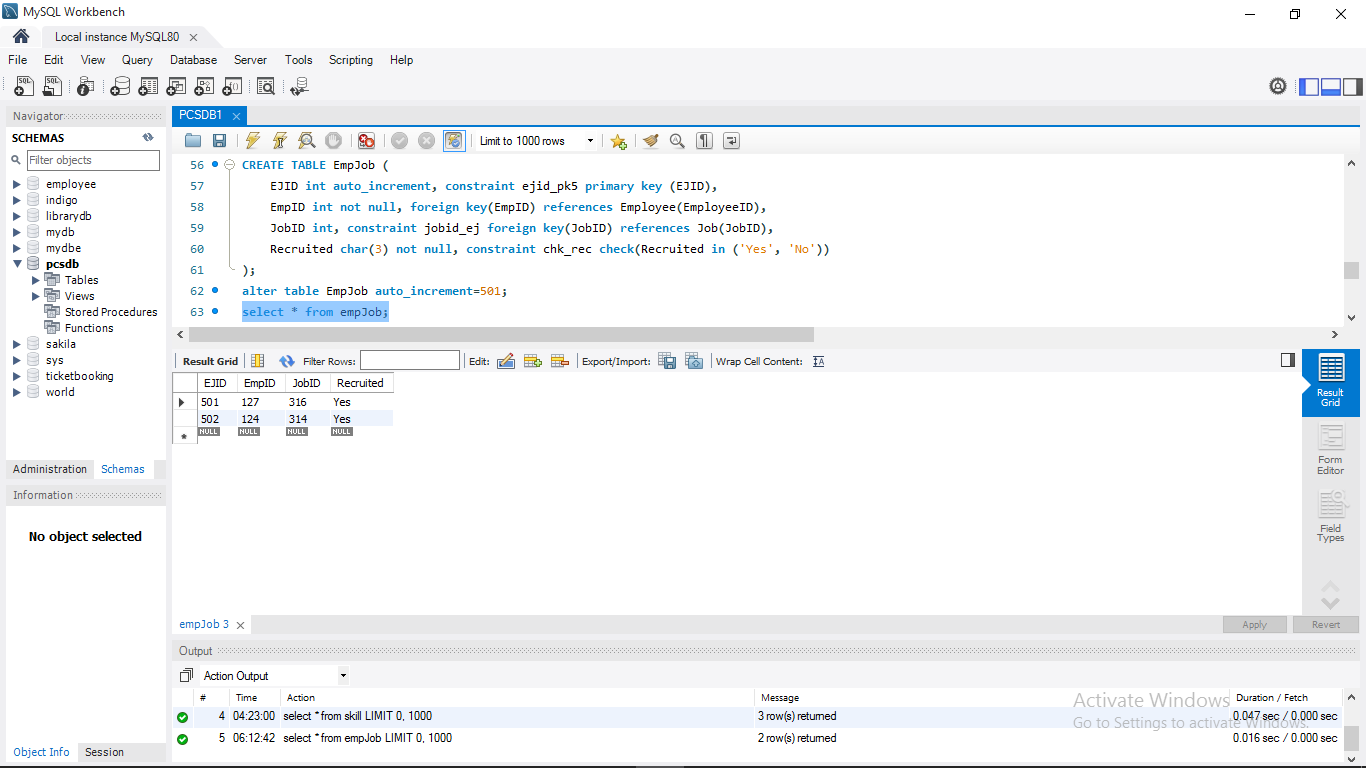
ExpYear int constraint chk\_expyear check (ExpYear>1) not null

);

Alter table EmpSkill auto\_increment=401;

Select \* from EmpSkill;

EMPJOB TABLE:



CREATE TABLE EmpJob (

EJID int auto\_increment, constraint ejid\_pk5 primary key (EJID),

JobID int, constraint jobid\_ej foreign key (JobID) references Job (JobID),

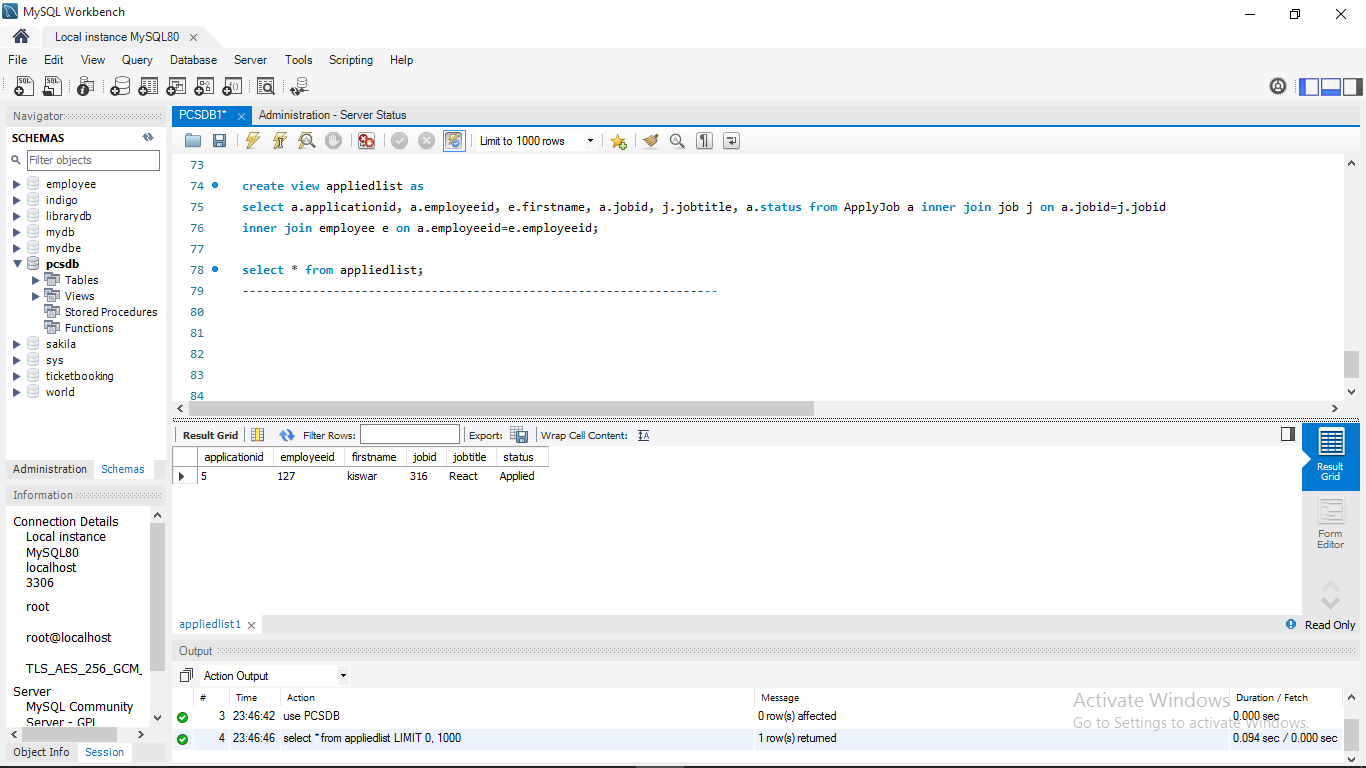
Recruited char (3) not null, constraint chk\_rec check (Recruited in ('Yes', 'No'))

);

Alter table EmpJob auto\_increment=501;

Select \* from empJob;

VIEW:



create view appliedlist as

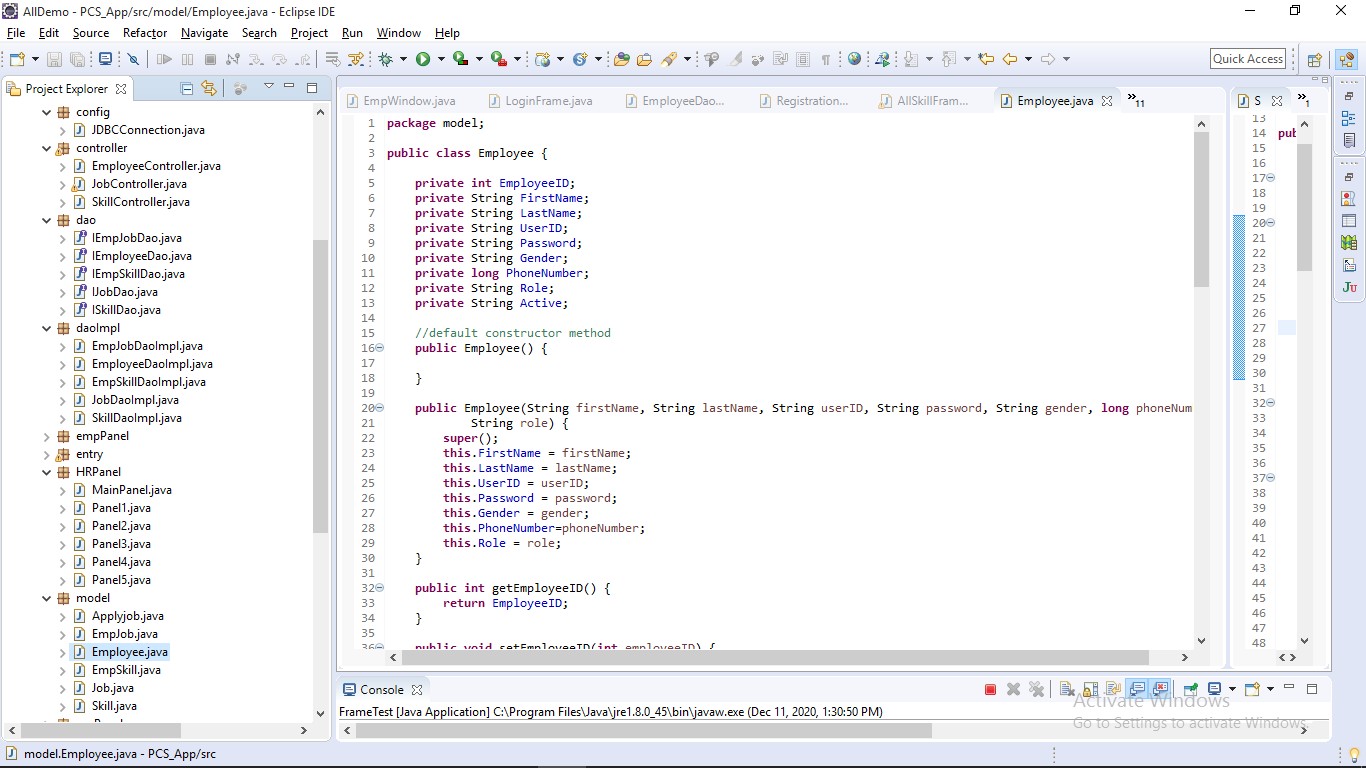
select a.applicationid, a.employeeid, e.firstname, a.jobid, j.jobtitle, a.status from ApplyJob a inner join job j on a.jobid=j.jobid

inner join employee e on a.employeeid=e.employeeid;

select \* from appliedlist;

Here, I have created a view to retrieve the details of all the employees who have applied for the job for the HR’s reference.

# 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 etc., 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.

Config package provides the configuration and it loads the JDBC driver to connect to the database.

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 = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

//LOADING DRIVER

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 EmployeeID;

private String FirstName;

private String LastName;

private String UserID;

private String Password;

private String Gender;

private long PhoneNumber;

private String Role;

private String Active;

//default constructor method

public Employee () {

}

public Employee (String firstName, String lastName, String userID, String password, String gender, long phoneNumber,

String role) {

super ();

this.FirstName = firstName;

this.LastName = lastName;

this.UserID = userID;

this.Password = password;

this.Gender = gender;

this.PhoneNumber=phoneNumber;

this.Role = role;

}

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 getGender() {

return Gender;

}

public void setGender(String gender) {

this.Gender = gender;

}

public long getPhoneNumber() {

return PhoneNumber;

}

public void setPhoneNumber(long phoneNumber) {

this.PhoneNumber = phoneNumber;

}

public String getRole() {

return Role;

}

public void setRole(String role) {

this.Role = role;

}

public String getActive() {

return Active;

}

public void setActive(String active) {

this.Active = active;

}

@Override

public String toString() {

return "Employee [EmployeeID=" + EmployeeID + ", FirstName=" + FirstName + ", LastName=" + LastName

+ ", UserID=" + UserID + ", Password=" + Password + ", Gender=" + Gender + ", PhoneNumber="

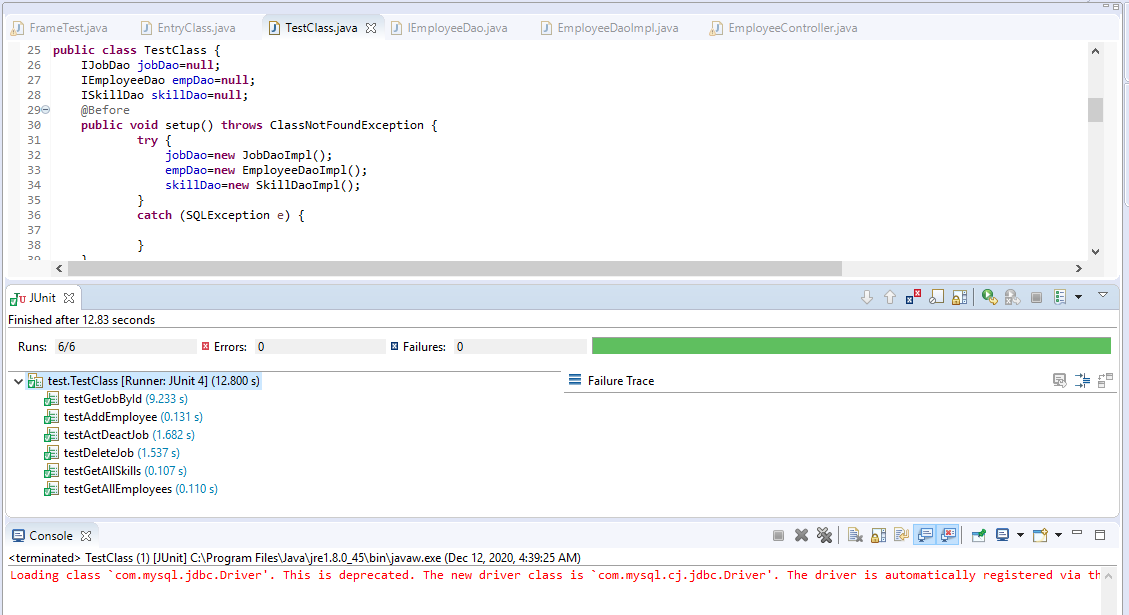
+ PhoneNumber + ", Role=" + Role + ", Active=" + Active + "]";

}

}

# TEST CASES

In Software Engineering, a test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program oath or to verify compliance with a specific requirement.



The above picture shows that the test cases that I have tested with JUNIT and all the test cases has been progressed successfully.

Here are some test cases:

@Test

public void testGetJobById() {

Job jobActual=new Job();

Job jobExpected=new Job(314, "Java Developer", "Need a developer with an experience of 1 year", " Professionet Consultancy Services", "Chennai", "core & Advanced Java", 21000, "Yes");

jobActual=jobDao.getJobById(314);

*assertEquals*(jobExpected.getJobTitle(), jobActual.getJobTitle());

}

The above test case checks whether the expected output matches with the actual output by using assertEquals statement.

@Test

public void testActDeactJob() {

Job jobActual=new Job();

Job JobExpected=new Job(314, "Java Developer", "Need a developer with an experience of 1 year", " Professionet Consultancy Services", "Chennai", "core & Advanced Java", 21000, "NO");

Job job=jobDao.getJobById(314);

job.setActive("NO");

jobActual=jobDao.deactivateJob(job, "Deactivated");

*assertEquals*(JobExpected.getActive(), jobActual.getActive());

}

The above test case monitors whether the job has been deactivated or not. If its deactivated, it will be set to “NO” and that will be compared with the assert equals statement.

@Test

public void testDeleteJob() {

Job jobActual=new Job();

new Job(315, "HR", "Human resources manager", " Professionet Consultancy Services", "Cochin", "good leading power", 28000, "Yes");

jobActual=jobDao.deleteJob(315);

*assertNull*(jobActual);

}

This test case implies that the actual output will be null. Because, here we are deleting the employee. Once it is deleted, it will be no longer available in the database. So, it will be checked using the assertNull statement.

@Test

public void testGetAllEmployees() {

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

*assertNotNull*(allEmpList);

Employee emp =allEmpList.get (3);

*AssertEquals (*126, emp.getEmployeeID ());

Employee emp1 =allEmpList.get (8);

*assertEquals* ("Kathir", emp1.getFirstName());

*assertTrue* ("Is Active ?", emp1.getActive().equals("YES"));

*assertThat* (allEmpList.size(), *is* (11));

}

Here, in the above test case, the details of all the employees are retrieved to the user interface to display to the user. It has to be checked whether it’s collecting all the available data from the database. So, the entire details will be collected in the List and first, we are checking for null values using *assertNotNull*. Then, using get method we are randomly checking the data using the index value. And also here *AssertThat* is used to check size of the array that is equals to the actual size of the List.

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.