**================= Employee Hiring System =================**

The project allows employees to register their details and companies to post job openings while connecting both based on skills and experience.

It consists of User and Company classes to model employee and company information, respectively, and uses the EmployeeHiring class to manage database operations, caching, and user interaction.

The program connects to a MySQL database and employs two types of caches: a Level 1 (l1Cache) cache using ConcurrentHashMap to store employee data for quick access, and a Level 2 (l2Cache) cache using Google’s Guava library to store companies based on required skills to optimise retrieval.

Employees can register their details or view a list of eligible companies, while companies can add job openings or view potential candidates.

**GITHUB LINK:**

[LalithaRavuri/EmployeeHiring (github.com)](https://github.com/LalithaRavuri/EmployeeHiring)

**Code:**

package com.mycompany.app.HiringSystemProject;

import java.sql.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.ConcurrentHashMap;

import java.util.function.Supplier;

import com.google.common.cache.Cache;

import com.google.common.cache.CacheBuilder;

class User1 {

String name;

String skills;

int workExperience;

User1(String name, String skills, int workExperience) {

this.name = name;

this.skills = skills;

this.workExperience = workExperience;

}

}

class Company1 {

String name;

String requiredSkill;

int minExperience;

int openPositions;

Company1(String name, String requiredSkill, int minExperience, int openPositions) {

this.name = name;

this.requiredSkill = requiredSkill;

this.minExperience = minExperience;

this.openPositions = openPositions;

}

}

public class Projecty {

static ArrayList<User1> users = new ArrayList<>();

static ArrayList<Company1> companies = new ArrayList<>();

static Scanner sc = new Scanner(System.in);

// Caches

static Map<String, User1> l1Cache = new ConcurrentHashMap<>();

static Cache<String, ArrayList<Company1>> l2Cache = CacheBuilder.newBuilder()

.maximumSize(10000)

.expireAfterAccess(10, TimeUnit.MINUTES)

.build();

private static final String URL = "jdbc:mysql://localhost:3306/employeehiring";

private static final String USERNAME = "root";

private static final String PASSWORD = "root";

private static Connection connection;

private static PreparedStatement preparedStatement;

private static ResultSet resultSet;

public static void main(String[] args) {

try {

connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

preloadData();

boolean exit = false;

while (!exit) {

System.out.println("\n\*\* Job Portal \*\*");

System.out.println("1. Employee");

System.out.println("2. Company");

System.out.println("3. View Cache");

System.out.println("4. Exit");

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

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

handleEmployee();

break;

case 2:

handleCompany();

break;

case 3:

viewCache();

break;

case 4:

exit = true;

System.out.println("Exiting the portal.");

break;

default:

System.out.println("Invalid choice. Try again.");

}

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

closeResources();

}

}

public static void preloadData() {

System.out.println("Pre-loaded data initialized successfully!");

}

public static void handleEmployee() {

System.out.println("\n-- Employee Menu --");

System.out.println("1. Enter New Details");

System.out.println("2. Show Eligible Companies");

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

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

addEmployee();

break;

case 2:

showEligibleCompanies();

break;

default:

System.out.println("Invalid choice.");

}

}

public static void handleCompany() {

System.out.println("\n-- Company Menu --");

System.out.println("1. Add More Details");

System.out.println("2. Select an Employee");

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

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

addCompany();

break;

case 2:

showPreferredEmployees();

break;

default:

System.out.println("Invalid choice.");

}

}

public static void addEmployee() {

System.out.print("Enter your name: ");

String name = sc.nextLine();

System.out.print("Enter your skills (comma separated, e.g., Java, Python): ");

String skills = sc.nextLine();

System.out.print("Enter your work experience (in years): ");

int workExperience = sc.nextInt();

String query = "INSERT INTO employees (first\_name, skills, work\_experience) VALUES (?, ?, ?)";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, name);

preparedStatement.setString(2, skills);

preparedStatement.setInt(3, workExperience);

preparedStatement.executeUpdate();

System.out.println("Employee details added successfully!");

l1Cache.put(name, new User1(name, skills, workExperience)); // L1 Cache

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void addCompany() {

System.out.print("Enter company name: ");

String name = sc.nextLine();

System.out.print("Enter required skill (e.g., Java, Python): ");

String skill = sc.nextLine();

System.out.print("Enter minimum work experience required (in years): ");

int minExperience = sc.nextInt();

System.out.print("Enter number of open positions: ");

int openPositions = sc.nextInt();

String query = "INSERT INTO companies (name, required\_skill, min\_experience, open\_positions) VALUES (?, ?, ?, ?)";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, name);

preparedStatement.setString(2, skill);

preparedStatement.setInt(3, minExperience);

preparedStatement.setInt(4, openPositions);

preparedStatement.executeUpdate();

System.out.println("Company details added successfully!");

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void showEligibleCompanies() {

System.out.print("Enter your name to find eligible companies: ");

String name = sc.nextLine();

User1 user = l1Cache.get(name); // Check L1 Cache

if (user == null) {

String query = "SELECT skills, work\_experience FROM employees WHERE first\_name = ?";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, name);

resultSet = preparedStatement.executeQuery();

if (!resultSet.next()) {

System.out.println("User not found. Please enter your details first.");

return;

}

String skills = resultSet.getString("skills");

int workExperience = resultSet.getInt("work\_experience");

user = new User1(name, skills, workExperience);

l1Cache.put(name, user); // Update L1 Cache

} catch (SQLException e) {

e.printStackTrace();

return;

}

}

String[] userSkills = user.skills.split(",");

System.out.println("\nEligible Companies for " + name + ":");

for (String skill : userSkills) {

ArrayList<Company1> cachedCompanies = l2Cache.getIfPresent(skill.trim()); // Check L2 Cache

if (cachedCompanies == null) {

String companyQuery = "SELECT name, required\_skill, min\_experience, open\_positions FROM companies WHERE required\_skill = ? AND min\_experience <= ?";

try {

preparedStatement = connection.prepareStatement(companyQuery);

preparedStatement.setString(1, skill.trim());

preparedStatement.setInt(2, user.workExperience);

resultSet = preparedStatement.executeQuery();

cachedCompanies = new ArrayList<>();

while (resultSet.next()) {

String companyName = resultSet.getString("name");

int availablePositions = resultSet.getInt("open\_positions");

System.out.println("Company: " + companyName + " | Available Positions: " + availablePositions);

cachedCompanies.add(new Company1(companyName, skill.trim(), user.workExperience, availablePositions));

}

l2Cache.put(skill.trim(), cachedCompanies); // Update L2 Cache

} catch (SQLException e) {

e.printStackTrace();

}

} else {

for (Company1 company : cachedCompanies) {

System.out.println("Company: " + company.name + " | Available Positions: " + company.openPositions);

}

}

}

}

public static void showPreferredEmployees() {

System.out.print("Enter company name to find preferred employees: ");

String companyName = sc.nextLine();

String query = "SELECT required\_skill, min\_experience FROM companies WHERE name = ?";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, companyName);

resultSet = preparedStatement.executeQuery();

if (!resultSet.next()) {

System.out.println("Company not found.");

return;

}

String requiredSkill = resultSet.getString("required\_skill");

int minExperience = resultSet.getInt("min\_experience");

String employeeQuery = "SELECT first\_name, skills, work\_experience FROM employees WHERE skills LIKE ? AND work\_experience >= ?";

preparedStatement = connection.prepareStatement(employeeQuery);

preparedStatement.setString(1, "%" + requiredSkill + "%");

preparedStatement.setInt(2, minExperience);

resultSet = preparedStatement.executeQuery();

System.out.println("\nPreferred Employees for " + companyName + ":");

while (resultSet.next()) {

String employeeName = resultSet.getString("first\_name");

System.out.println("Employee: " + employeeName);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void viewCache() {

System.out.println("L1 Cache (User Data):");

for (String key : l1Cache.keySet()) {

System.out.println("Name: " + key + ", User: " + l1Cache.get(key));

}

System.out.println("L2 Cache (Company Data):");

for (Map.Entry<String, ArrayList<Company1>> entry : l2Cache.asMap().entrySet()) {

System.out.println("Skill: " + entry.getKey() + ", Companies: " + entry.getValue());

}

}

public static void closeResources() {

try {

if (resultSet != null) resultSet.close();

if (preparedStatement != null) preparedStatement.close();

if (connection != null) connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}