**CareerHub, The Job Board**

1. **Entity/Model Classes**

**JobListing.java**

package entity;

import java.time.LocalDate;

public class JobListing {

private int jobID;

private int companyID;

private String jobTitle;

private String jobDescription;

private String jobLocation;

private double salary;

private String jobType;

private LocalDate postedDate;

public JobListing() {}

public JobListing(int jobID, int companyID, String jobTitle, String jobDescription,

String jobLocation, double salary, String jobType, LocalDate postedDate) {

this.jobID = jobID;

this.companyID = companyID;

this.jobTitle = jobTitle;

this.jobDescription = jobDescription;

this.jobLocation = jobLocation;

this.salary = salary;

this.jobType = jobType;

this.postedDate = postedDate;

}

// Getters and Setters

public int getJobID() { return jobID; }

public void setJobID(int jobID) { this.jobID = jobID; }

public int getCompanyID() { return companyID; }

public void setCompanyID(int companyID) { this.companyID = companyID; }

public String getJobTitle() { return jobTitle; }

public void setJobTitle(String jobTitle) { this.jobTitle = jobTitle; }

public String getJobDescription() { return jobDescription; }

public void setJobDescription(String jobDescription) { this.jobDescription = jobDescription; }

public String getJobLocation() { return jobLocation; }

public void setJobLocation(String jobLocation) { this.jobLocation = jobLocation; }

public double getSalary() { return salary; }

public void setSalary(double salary) {

if (salary < 0) throw new IllegalArgumentException("Salary cannot be negative.");

this.salary = salary;

}

public String getJobType() { return jobType; }

public void setJobType(String jobType) { this.jobType = jobType; }

public LocalDate getPostedDate() { return postedDate; }

public void setPostedDate(LocalDate postedDate) { this.postedDate = postedDate; }

}

**Company.java**

package entity;

import java.util.ArrayList;

import java.util.List;

public class Company {

private int companyID;

private String companyName;

private String location;

private List<JobListing> jobListings;

public Company() {

jobListings = new ArrayList<>();

}

public Company(int companyID, String companyName, String location) {

this.companyID = companyID;

this.companyName = companyName;

this.location = location;

this.jobListings = new ArrayList<>();

}

public int getCompanyID() { return companyID; }

public void setCompanyID(int companyID) { this.companyID = companyID; }

public String getCompanyName() { return companyName; }

public void setCompanyName(String companyName) { this.companyName = companyName; }

public String getLocation() { return location; }

public void setLocation(String location) { this.location = location; }

public void postJob(JobListing job) {

jobListings.add(job);

}

public List<JobListing> getJobs() {

return jobListings;

}

}

**Applicant.java**

package entity;

public class Applicant {

private int applicantID;

private String firstName;

private String lastName;

private String email;

private String phone;

private String resume;

public Applicant() {}

public Applicant(int applicantID, String firstName, String lastName, String email, String phone, String resume) {

this.applicantID = applicantID;

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.phone = phone;

this.resume = resume;

}

public int getApplicantID() { return applicantID; }

public void setApplicantID(int applicantID) { this.applicantID = applicantID; }

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 getEmail() { return email; }

public void setEmail(String email) {

if (!email.contains("@")) throw new IllegalArgumentException("Invalid email format.");

this.email = email;

}

public String getPhone() { return phone; }

public void setPhone(String phone) { this.phone = phone; }

public String getResume() { return resume; }

public void setResume(String resume) { this.resume = resume; }

}

**JobApplication.java**

package entity;

import java.time.LocalDate;

public class JobApplication {

private int applicationID;

private int jobID;

private int applicantID;

private LocalDate applicationDate;

private String coverLetter;

public JobApplication() {}

public JobApplication(int applicationID, int jobID, int applicantID, LocalDate applicationDate, String coverLetter) {

this.applicationID = applicationID;

this.jobID = jobID;

this.applicantID = applicantID;

this.applicationDate = applicationDate;

this.coverLetter = coverLetter;

}

public int getApplicationID() { return applicationID; }

public void setApplicationID(int applicationID) { this.applicationID = applicationID; }

public int getJobID() { return jobID; }

public void setJobID(int jobID) { this.jobID = jobID; }

public int getApplicantID() { return applicantID; }

public void setApplicantID(int applicantID) { this.applicantID = applicantID; }

public LocalDate getApplicationDate() { return applicationDate; }

public void setApplicationDate(LocalDate applicationDate) { this.applicationDate = applicationDate; }

public String getCoverLetter() { return coverLetter; }

public void setCoverLetter(String coverLetter) { this.coverLetter = coverLetter; }

}

1. **DAO Classes**

**JobListingDAO.java (Interface)**

package dao;

import entity.JobListing;

import java.util.List;

public interface JobListingDAO {

void insertJobListing(JobListing job) throws Exception;

List<JobListing> getAllJobListings() throws Exception;

List<JobListing> getJobsBySalaryRange(double minSalary, double maxSalary) throws Exception;

}

**JobListingDAOImpl.java (Implementation)**

package dao;

import entity.JobListing;

import util.DBConnUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class JobListingDAOImpl implements JobListingDAO {

@Override

public void insertJobListing(JobListing job) throws Exception {

try (Connection conn = DBConnUtil.getConnection()) {

String sql = "INSERT INTO Jobs (JobID, CompanyID, JobTitle, JobDescription, JobLocation, Salary, JobType, PostedDate) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, job.getJobID());

stmt.setInt(2, job.getCompanyID());

stmt.setString(3, job.getJobTitle());

stmt.setString(4, job.getJobDescription());

stmt.setString(5, job.getJobLocation());

stmt.setDouble(6, job.getSalary());

stmt.setString(7, job.getJobType());

stmt.setDate(8, Date.valueOf(job.getPostedDate()));

stmt.executeUpdate();

} catch (SQLException e) {

throw new Exception("Error inserting job listing: " + e.getMessage());

}

}

@Override

public List<JobListing> getAllJobListings() throws Exception {

List<JobListing> jobs = new ArrayList<>();

try (Connection conn = DBConnUtil.getConnection()) {

String sql = "SELECT \* FROM Jobs";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

jobs.add(new JobListing(

rs.getInt("JobID"),

rs.getInt("CompanyID"),

rs.getString("JobTitle"),

rs.getString("JobDescription"),

rs.getString("JobLocation"),

rs.getDouble("Salary"),

rs.getString("JobType"),

rs.getDate("PostedDate").toLocalDate()

));

}

} catch (SQLException e) {

throw new Exception("Error retrieving job listings: " + e.getMessage());

}

return jobs;

}

@Override

public List<JobListing> getJobsBySalaryRange(double minSalary, double maxSalary) throws Exception {

List<JobListing> jobs = new ArrayList<>();

try (Connection conn = DBConnUtil.getConnection()) {

String sql = "SELECT \* FROM Jobs WHERE Salary BETWEEN ? AND ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setDouble(1, minSalary);

stmt.setDouble(2, maxSalary);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

jobs.add(new JobListing(

rs.getInt("JobID"),

rs.getInt("CompanyID"),

rs.getString("JobTitle"),

rs.getString("JobDescription"),

rs.getString("JobLocation"),

rs.getDouble("Salary"),

rs.getString("JobType"),

rs.getDate("PostedDate").toLocalDate()

));

}

} catch (SQLException e) {

throw new Exception("Error retrieving job listings by salary range: " + e.getMessage());

}

return jobs;

}

}

1. **Utility Classes**

**DBConnUtil.java**

package util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnUtil {

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

private static final String USERNAME = "root";

private static final String PASSWORD = "password";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USERNAME, PASSWORD);

}

}

1. **MainModule**

**MainModule.java**

package main;

import dao.JobListingDAO;

import dao.JobListingDAOImpl;

import entity.JobListing;

import java.time.LocalDate;

import java.util.List;

import java.util.Scanner;

public class MainModule {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

JobListingDAO jobListingDAO = new JobListingDAOImpl();

while (true) {

System.out.println("\nJob Board Menu:");

System.out.println("1. Post a Job");

System.out.println("2. View All Jobs");

System.out.println("3. Search Jobs by Salary Range");

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

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

int choice = scanner.nextInt();

switch (choice) {

case 1:

try {

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

int jobID = scanner.nextInt();

System.out.print("Enter Company ID: ");

int companyID = scanner.nextInt();

scanner.nextLine(); // Consume newline

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

String jobTitle = scanner.nextLine();

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

String jobDescription = scanner.nextLine();

System.out.print("Enter Job Location: ");

String jobLocation = scanner.nextLine();

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

double salary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

System.out.print("Enter Job Type: ");

String jobType = scanner.nextLine();

LocalDate postedDate = LocalDate.now();

JobListing job = new JobListing(jobID, companyID, jobTitle, jobDescription, jobLocation, salary, jobType, postedDate);

jobListingDAO.insertJobListing(job);

System.out.println("Job posted successfully!");

} catch (Exception e) {

System.err.println("Error: " + e.getMessage());

}

break;

case 2:

try {

List<JobListing> jobs = jobListingDAO.getAllJobListings();

System.out.println("\nAll Job Listings:");

for (JobListing job : jobs) {

System.out.println("Job ID: " + job.getJobID() + ", Title: " + job.getJobTitle() + ", Salary: " + job.getSalary());

}

} catch (Exception e) {

System.err.println("Error: " + e.getMessage());

}

break;

case 3:

try {

System.out.print("Enter Minimum Salary: ");

double minSalary = scanner.nextDouble();

System.out.print("Enter Maximum Salary: ");

double maxSalary = scanner.nextDouble();

List<JobListing> jobs = jobListingDAO.getJobsBySalaryRange(minSalary, maxSalary);

System.out.println("\nJobs within Salary Range:");

for (JobListing job : jobs) {

System.out.println("Job ID: " + job.getJobID() + ", Title: " + job.getJobTitle() + ", Salary: " + job.getSalary());

}

} catch (Exception e) {

System.err.println("Error: " + e.getMessage());

}

break;

case 4:

System.out.println("Exiting application...");

scanner.close();

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

}