JAVA CODING CHALLENGE-CAREERHUB

STEP 1:Creating SQL Schema For the application

```sql
create database if not exists CareerHub;
use CareerHub;
create table if not exists Companies (
    CompanyID int primary key auto_increment,
    CompanyName varchar(100)not null,
    Location varchar(100) NOT NULL
);

create table if not exists Jobs (
    JobID int primary key auto_increment,
    CompanyID int,
    JobTitle varchar(200) not null,
    JobDescription text,
    JobLocation varchar(250),
    Salary decimal(10,2),
    JobType varchar(50),
    PostedDate datetime,
    foreign key (CompanyID) References Companies(CompanyID) on delete cascade
);

Create table if not exists Applicants (
    ApplicantID int primary key auto_increment,
    FirstName varchar(100) not null,
    LastName varchar(100) not null,
    Email varchar(255) unique not null,
    Phone varchar(20),
    Resume text
);

Create table if not exists Applications (
    ApplicationID int primary key auto_increment,
    JobID int,
    ApplicantID int,
    ApplicationDate datetime,
    CoverLetter text,
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID) ON DELETE CASCADE,
    FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID) ON DELETE CASCADE
);
```

Creating and implementing the mentioned class and the structure in the application.

JobListing Class:

```
package entity;

import java.time.LocalDateTime;
import java.util.List;

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 LocalDateTime postedDate;

    public JobListing() {}

    public JobListing(int jobId, int companyId, String jobTitle, String
jobDescription,
                      String jobLocation, double salary, String jobType,
LocalDateTime 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;
    }
    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) { this.salary = salary; }

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

    public LocalDateTime getPostedDate() { return postedDate; }
```

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

    public void apply(int applicantID, String coverLetter) {
        System.out.println("Applicant " + applicantID + " applied to job " +
jobId + " (Handled in DAO).");
    }

    public List<Applicant> getApplicants() {
        System.out.println("Returning applicants for job (Handled in DAO).");
        return null;
    }
}
```

Company Class:

```
package entity;

import java.util.List;

public class Company {
    private int companyId;
    private String companyName;
    private String location;

    public Company() {}

    public Company(int companyId, String companyName, String location) {
        this.companyId = companyId;
        this.companyName = companyName;
        this.location = location;
    }

    // Getters & Setters
    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; }

    // Method Stubs (to be implemented in DAO layer)
    public void postJob(String jobTitle, String jobDescription, String
jobLocation, double salary, String jobType) {
        System.out.println("Post job called (Handled in DAO Layer).");
    }

    public List<JobListing> getJobs() {
        System.out.println("Retrieve company jobs (Handled in DAO Layer).");
        return null;
    }
}
```

Applicant Class:

```
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;
    }

    // Getters & Setters
    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) { 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; }

    // Method Stubs
    public void createProfile(String email, String firstName, String
lastName, String phone) {
        System.out.println("Create profile called. (Handled in DAO Layer)");
    }

    public void applyForJob(int jobID, String coverLetter) {
        System.out.println("Applied to job " + jobID + " (Handled in DAO
Layer)");
    }
}
```

JobApplication Class:

```
package entity;

import java.time.LocalDateTime;

public class JobApplication {
    private int applicationId;
    private int jobId;
    private int applicantId;
    private LocalDateTime applicationDate;
    private String coverLetter;

    public JobApplication() {}

    public JobApplication(int applicationId, int jobId, int applicantId,
                          LocalDateTime applicationDate, String coverLetter)
{
        this.applicationId = applicationId;
        this.jobId = jobId;
        this.applicantId = applicantId;
        this.applicationDate = applicationDate;
        this.coverLetter = coverLetter;
    }

    // Getters & Setters
    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 LocalDateTime getApplicationDate() { return applicationDate; }
    public void setApplicationDate(LocalDateTime applicationDate) {
this.applicationDate = applicationDate; }

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

2. DatabaseManager Class:

```
package dao;

import entity.*;
import util.DBConnUtil;

import java.sql.*;
```

```
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class DataBaseManager {

    private Connection getConnection() throws SQLException {
        return DBConnUtil.getConnection();
    }

    public void initializeDatabase() {
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement()) {

            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS Companies (" +
                    "CompanyID INT PRIMARY KEY, " +
                    "CompanyName VARCHAR(100), " +
                    "Location VARCHAR(100))");

            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS Applicants (" +
                    "ApplicantID INT PRIMARY KEY, " +
                    "FirstName VARCHAR(50), " +
                    "LastName VARCHAR(50), " +
                    "Email VARCHAR(100), " +
                    "Phone VARCHAR(15), " +
                    "Resume TEXT)");

            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS Jobs (" +
                    "JobID INT PRIMARY KEY, " +
                    "CompanyID INT, " +
                    "JobTitle VARCHAR(100), " +
                    "JobDescription TEXT, " +
                    "JobLocation VARCHAR(100), " +
                    "Salary DECIMAL(10,2), " +
                    "JobType VARCHAR(50), " +
                    "PostedDate DATETIME, " +
                    "FOREIGN KEY (CompanyID) REFERENCES
Companies(CompanyID))");

            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS Applications (" +
                    "ApplicationID INT PRIMARY KEY, " +
                    "JobID INT, " +
                    "ApplicantID INT, " +
                    "ApplicationDate DATETIME, " +
                    "CoverLetter TEXT, " +
                    "FOREIGN KEY (JobID) REFERENCES Jobs(JobID), " +
                    "FOREIGN KEY (ApplicantID) REFERENCES
Applicants(ApplicantID))");

            System.out.println("Database initialized successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void insertCompany(Company company) {
```

```
        String sql = "INSERT INTO Companies VALUES (?, ?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, company.getCompanyId());
            ps.setString(2, company.getCompanyName());
            ps.setString(3, company.getLocation());
            ps.executeUpdate();
            System.out.println("Company inserted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void insertApplicant(Applicant applicant) {
        String sql = "INSERT INTO Applicants VALUES (?, ?, ?, ?, ?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, applicant.getApplicantId());
            ps.setString(2, applicant.getFirstName());
            ps.setString(3, applicant.getLastName());
            ps.setString(4, applicant.getEmail());
            ps.setString(5, applicant.getPhone());
            ps.setString(6, applicant.getResume());
            ps.executeUpdate();
            System.out.println("Applicant inserted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void insertJobListing(JobListing job) {
        String sql = "INSERT INTO Jobs VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, job.getJobId());
            ps.setInt(2, job.getCompanyId());
            ps.setString(3, job.getJobTitle());
            ps.setString(4, job.getJobDescription());
            ps.setString(5, job.getJobLocation());
            ps.setDouble(6, job.getSalary());
            ps.setString(7, job.getJobType());
            ps.setTimestamp(8, Timestamp.valueOf(job.getPostedDate()));
            ps.executeUpdate();
            System.out.println("Job inserted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void insertJobApplication(JobApplication application) {
        String sql = "INSERT INTO Applications VALUES (?, ?, ?, ?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, application.getApplicationId());
            ps.setInt(2, application.getJobId());
            ps.setInt(3, application.getApplicantId());
            ps.setTimestamp(4,
```

```
Timestamp.valueOf(application.getApplicationDate()));
            ps.setString(5, application.getCoverLetter());
            ps.executeUpdate();
            System.out.println("Application inserted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<JobListing> getJobListings() {
        List<JobListing> list = new ArrayList<>();
        String sql = "SELECT * FROM Jobs";
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                JobListing job = new JobListing(
                        rs.getInt("JobID"),
                        rs.getInt("CompanyID"),
                        rs.getString("JobTitle"),
                        rs.getString("JobDescription"),
                        rs.getString("JobLocation"),
                        rs.getDouble("Salary"),
                        rs.getString("JobType"),
                        rs.getTimestamp("PostedDate").toLocalDateTime()
                );
                list.add(job);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    public List<Company> getCompanies() {
        List<Company> list = new ArrayList<>();
        String sql = "SELECT * FROM Companies";
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Company company = new Company(
                        rs.getInt("CompanyID"),
                        rs.getString("CompanyName"),
                        rs.getString("Location")
                );
                list.add(company);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    public List<Applicant> getApplicants() {
        List<Applicant> list = new ArrayList<>();
        String sql = "SELECT * FROM Applicants";
```

```
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Applicant app = new Applicant(
                        rs.getInt("ApplicantID"),
                        rs.getString("FirstName"),
                        rs.getString("LastName"),
                        rs.getString("Email"),
                        rs.getString("Phone"),
                        rs.getString("Resume")
                );
                list.add(app);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    public List<JobApplication> getApplicationsForJob(int jobID) {
        List<JobApplication> list = new ArrayList<>();
        String sql = "SELECT * FROM Applications WHERE JobID = ?";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, jobID);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                JobApplication app = new JobApplication(
                        rs.getInt("ApplicationID"),
                        rs.getInt("JobID"),
                        rs.getInt("ApplicantID"),
                        rs.getTimestamp("ApplicationDate").toLocalDateTime(),
                        rs.getString("CoverLetter")
                );
                list.add(app);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }
}
```

3. Exceptions handling

Invalid Email Format Handling:

```
package exception;

import java.util.Scanner;
```

```
public class InvalidEmailException extends Exception {
    public InvalidEmailException(String message) {
        super(message);
    }

    public static class EmailValidationProgram {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter email: ");
            String email = scanner.nextLine();

            try {
                if (!email.contains("@") || !email.matches("^[\\w.-]+@[\\w.-
]+\\.\\w+$")) {
                    throw new InvalidEmailException("Invalid email format!");
                }
                System.out.println("Email is valid. Proceeding with
registration...");
            } catch (InvalidEmailException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

Salary Calculation Handling:

```
package exception;

import java.util.Scanner;

public class InvalidSalaryException extends Exception {
    public InvalidSalaryException(String message) {
        super(message);
    }

    public static class SalaryCalculator {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            double total = 0;
            int count = 0;

            System.out.print("Enter number of job listings: ");
            int n = scanner.nextInt();

            for (int i = 1; i <= n; i++) {
                System.out.print("Enter salary for job " + i + ": ");
                double salary = scanner.nextDouble();
                try {
                    if (salary < 0) {
                        throw new InvalidSalaryException("Negative salary
found for job " + i);
                    }
                    total += salary;
                    count++;
```

```
                } catch (InvalidSalaryException e) {
                    System.err.println(e.getMessage());
                }
            }

            if (count > 0) {
                System.out.println("Average Salary: " + (total / count));
            } else {
                System.out.println("No valid salaries to calculate
average.");
            }
        }
    }
}
```

File Upload Exception Handling:

```
package exception;

import java.io.File;

public class FileUploadException extends Exception {
    public FileUploadException(String message) {
        super(message);
    }

    public static class ResumeUploader {
        public static void main(String[] args) {
            String filePath = "resumes/sample.pdf";

            try {
                File file = new File(filePath);
                if (!file.exists()) {
                    throw new FileUploadException("Resume file not found.");
                }

                if (!file.getName().endsWith(".pdf")) {
                    throw new FileUploadException("Unsupported file format.
Only .pdf allowed.");
                }

                if (file.length() > 2 * 1024 * 1024) {
                    throw new FileUploadException("File size exceeded. Max
size: 2MB");
                }

                System.out.println("Resume uploaded successfully.");
            } catch (FileUploadException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

Application Deadline Handling:

```java
package exception;

import java.time.LocalDateTime;

public class ApplicationDeadlineException extends Exception {
    public ApplicationDeadlineException(String message) {
        super(message);
    }

    public static class ApplicationDeadlineChecker {
        public static void main(String[] args) {
            LocalDateTime deadline = LocalDateTime.of(2025, 4, 1, 23, 59);
            LocalDateTime current = LocalDateTime.now();

            try {
                if (current.isAfter(deadline)) {
                    throw new ApplicationDeadlineException("Application
deadline has passed!");
                }
                System.out.println("Application submitted on time.");
            } catch (ApplicationDeadlineException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

Database Connection Handling:

```java
package exception;

public class DatabaseConnectionException extends Exception {
    public DatabaseConnectionException(String message) {
        super(message);
    }
}
```

4. Database Connectivity

DBConnUtil

```
package util;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnUtil {

    private static final String fileName = "db.properties";

    public static Connection getConnection() {
        Connection con = null;
        String connString = null;

        try {
            connString = DBPropertyUtil.getConnectionString(fileName);
        } catch (IOException e) {
            System.out.println("Connection String Creation Failed");
            e.printStackTrace();
        }

        if (connString != null) {
            try {
                con = DriverManager.getConnection(connString);
            } catch (SQLException e) {
                System.out.println("Error While Establishing DB
Connection...");
                e.printStackTrace();
            }
        }

        return con;
    }
}
```

DBPropertyUtil.java

```
package util;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class DBPropertyUtil {

    public static String getConnectionString(String fileName) throws
IOException {
        String connStr = null;
        Properties props = new Properties();
```

```
        FileInputStream fis = new FileInputStream("resources/" + fileName);
        props.load(fis);

        String user = props.getProperty("user");
        String password = props.getProperty("password");
        String protocol = props.getProperty("protocol");
        String system = props.getProperty("system");
        String port = props.getProperty("port");
        String database = props.getProperty("database");

        connStr = protocol + "//" + system + ":" + port + "/" + database +
                "?user=" + user + "&password=" + password;

        return connStr;
    }
}
```

db.properties:

```
protocol=jdbc:mysql:
system=localhost
port=3306
database=CareerHub
user=root
password=Sandhiya_21
```

Main.java

```
package Main;

import dao.DataBaseManager;
import entity.*;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Scanner;

public class MainModule {
    static Scanner sc = new Scanner(System.in);
    static DataBaseManager db = new DataBaseManager();

    public static void main(String[] args) {
        db.initializeDatabase();
        boolean running = true;

        while (running) {
            System.out.println("\n=== CareerHub Job Board ===");
            System.out.println("1. View All Job Listings");
            System.out.println("2. Create Applicant Profile");
            System.out.println("3. Submit Job Application");
            System.out.println("4. Post a Job (Company)");
```

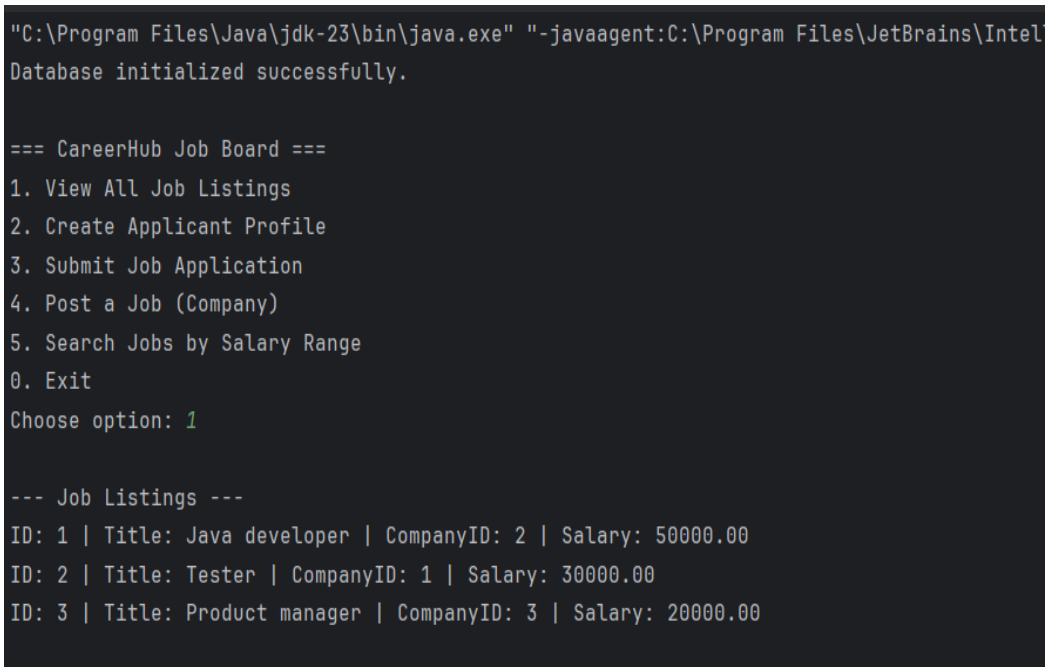
```
            System.out.println("5. Search Jobs by Salary Range");
            System.out.println("0. Exit");
            System.out.print("Choose option: ");

            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1 -> viewJobs();
                case 2 -> createApplicant();
                case 3 -> applyJob();
                case 4 -> postJob();
                case 5 -> searchSalaryRange();
                case 0 -> {
                    System.out.println("Exiting...");
                    running = false;
                }
                default -> System.out.println("Invalid choice.");
            }
        }
    }

    // 1. View all job listings
    private static void viewJobs() {
        List<JobListing> jobs = db.getJobListings();
        System.out.println("\n--- Job Listings ---");
        for (JobListing job : jobs) {
            System.out.printf("ID: %d | Title: %s | CompanyID: %d | Salary:
%.2f%n",
                    job.getJobId(), job.getJobTitle(), job.getCompanyId(),
job.getSalary());
        }
    }

    // 2. Create applicant profile
    private static void createApplicant() {
        try {
            System.out.print("Enter Applicant ID: ");
            int id = sc.nextInt(); sc.nextLine();
            System.out.print("First Name: ");
            String fname = sc.nextLine();
            System.out.print("Last Name: ");
            String lname = sc.nextLine();
            System.out.print("Email: ");
            String email = sc.nextLine();
            System.out.print("Phone: ");
            String phone = sc.nextLine();
            System.out.print("Resume (text): ");
            String resume = sc.nextLine();

            Applicant applicant = new Applicant(id, fname, lname, email,
phone, resume);
            db.insertApplicant(applicant);
        } catch (Exception e) {
            System.err.println("Failed to create profile: " +
e.getMessage());
        }
```

```
    }

    // 3. Apply to a job
    private static void applyJob() {
        try {
            System.out.print("Enter Application ID: ");
            int appId = sc.nextInt(); sc.nextLine();
            System.out.print("Applicant ID: ");
            int applicantId = sc.nextInt(); sc.nextLine();
            System.out.print("Job ID: ");
            int jobId = sc.nextInt(); sc.nextLine();
            System.out.print("Cover Letter: ");
            String cover = sc.nextLine();

            JobApplication application = new JobApplication(
                    appId, jobId, applicantId, LocalDateTime.now(), cover);
            db.insertJobApplication(application);
        } catch (Exception e) {
            System.err.println("Failed to apply: " + e.getMessage());
        }
    }

    // 4. Company posts a job
    private static void postJob() {
        try {
            System.out.print("Enter Job ID: ");
            int jobId = sc.nextInt(); sc.nextLine();
            System.out.print("Company ID: ");
            int companyId = sc.nextInt(); sc.nextLine();
            System.out.print("Title: ");
            String title = sc.nextLine();
            System.out.print("Description: ");
            String desc = sc.nextLine();
            System.out.print("Location: ");
            String location = sc.nextLine();
            System.out.print("Salary: ");
            double salary = sc.nextDouble(); sc.nextLine();
            System.out.print("Job Type: ");
            String type = sc.nextLine();

            JobListing job = new JobListing(
                    jobId, companyId, title, desc, location, salary, type,
LocalDateTime.now());
            db.insertJobListing(job);
        } catch (Exception e) {
            System.err.println("Failed to post job: " + e.getMessage());
        }
    }

    // 5. Salary range query
    private static void searchSalaryRange() {
        try {
            System.out.print("Min Salary: ");
            double min = sc.nextDouble();
            System.out.print("Max Salary: ");
            double max = sc.nextDouble();
```

```
            List<JobListing> jobs = db.getJobListings();
            System.out.println("\n--- Jobs in Salary Range ---");
            for (JobListing job : jobs) {
                if (job.getSalary() >= min && job.getSalary() <= max) {
                    System.out.printf("ID: %d | %s | Salary: %.2f%n",
                            job.getJobId(), job.getJobTitle(),
job.getSalary());
                }
            }
        } catch (Exception e) {
            System.err.println("Error during search: " + e.getMessage());
        }
    }
}
```

OUTPUT:

1.View job listings



2.Create applicant profile

```
=== CareerHub Job Board ===
1. View All Job Listings
2. Create Applicant Profile
3. Submit Job Application
4. Post a Job (Company)
5. Search Jobs by Salary Range
0. Exit
Choose option: 2
Enter Applicant ID: 4
First Name: nandhini
Last Name: lal
Email: nandy@gmail.com
Phone: 7896543201
Resume (text): Skilled in python development
Applicant inserted.
```

3.Submit job application

```
=== CareerHub Job Board ===
1. View All Job Listings
2. Create Applicant Profile
3. Submit Job Application
4. Post a Job (Company)
5. Search Jobs by Salary Range
0. Exit
Choose option: 3
Enter Application ID: 2
Applicant ID: 1
Job ID: 1
Cover Letter: hope i'll get this job
Application inserted.
```

4.Post a job

```
=== CareerHub Job Board ===
1. View All Job Listings
2. Create Applicant Profile
3. Submit Job Application
4. Post a Job (Company)
5. Search Jobs by Salary Range
0. Exit
Choose option: 4
Enter Job ID: 4
Company ID: 1
Title: account manager
Description: manage accounts
Location: chennai
Salary: 20000
Job Type: full-time
Job inserted.
```

5.Search jobs by salary range

```
=== CareerHub Job Board ===
1. View All Job Listings
2. Create Applicant Profile
3. Submit Job Application
4. Post a Job (Company)
5. Search Jobs by Salary Range
0. Exit
Choose option: 5
Min Salary: 30000
Max Salary: 40000

--- Jobs in Salary Range ---
ID: 2 | Tester | Salary: 30000.00
```

SQL OUTPUT AFTER INSERTING SOME VALUES THROUGH THE CARRERHUB APPLICATION:

APPLICANTS TABLE:

```
65 •  select * from applicants;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ApplicantID	FirstName	LastName	Email	Phone	Resume
1	abi	peter	abi@gmail.com	987654341	I'm interested in this job
2	sandhiya	paraman	sandhiya@gmail.com	9876543209	Skills in java development,oops,database
3	swathi	raj	swathi@gmail.com	8765438901	Skilled in app development
4	nandhini	lal	nandy@gmail.com	7896543201	Skilled in python development
NULL	NULL	NULL	NULL	NULL	NULL

APPLICATIONS TABLE:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ApplicationID	JobID	ApplicantID	ApplicationDate	CoverLetter
1	1	1	2025-04-09 12:04:42	kindly provide me with this job
2	1	1	2025-04-09 13:06:30	hope i'll get this job
NULL	NULL	NULL	NULL	NULL

JOBS TABLE:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

JobID	CompanyID	JobTitle	JobDescription	JobLocation	Salary	JobType	PostedDate
1	2	Java developer	build enterprise applications	bangalore	50000.00	full-time	2025-04-09 12:03:35
2	1	Tester	Test applications	Mumbai	30000.00	Full-time	2025-04-09 12:07:27
3	3	Product manager	Product management	Chennai	20000.00	Full-time	2025-04-09 12:08:19
4	1	account manager	manage accounts	chennai	20000.00	full-time	2025-04-09 13:08:25
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL