# CSCI3170 (2018-2019 2nd term)
# Introduction to Database Systems
# Project–Recruitment System

**Group Registration Deadline:** 23:59 1st Feb 2019
**Phase 1 Deadline:** 23:59 19th Feb 2019
**Phase 2 Deadline:** 23:59 7th Apr 2019

## 1. Introduction

You are required to implement a Recruitment System for managing data relevant to job recruitment, involving employee, employer, position, and employment history. The system shall provide an interactive interface to system administrator, employee, and employer.

This project is divided into 2 phases:
In phase 1, you are required to design a database for the system (including an ER-diagram and a relational schema). A suggested solution will be provided after phase 1. You are required to use the suggested solution to complete phase 2.

In phase 2, you are required to implement the system as a **Java** command-line program.
Our tutors will give tutorials on how to connect to a **MySQL** database system with JDBC API and deploy your work on the required platform.

This is a group project and each group should have at most 3 members. ONLY one copy of solution is required for each group. Please fill out the group registration form on the blackboard system before the group registration deadline.

## 2. Milestones

Preparation

- Read the document thoroughly and make sure you understand all the assumptions and regulations stated in Section 4.

Phase 1 (20 %)

- According to the data specifications in Section 3, design an ER-diagram and transform it into a relational schema without any redundant fields and tables.

Phase 2 (80 %)

- According to the suggested solution of phase 1, implement a Java application that fulfills all requirements stated in Section 5.
- Debug your system with different datasets and user inputs.
- Write a readme file to describe the compilation and deployment of your system.

# 3. Data Files Specification

All data files have Unix Line Ending (i.e. the newline character is "**\n**") and are encoded in ASCII. The system is required to read records stored in the files and put them into appropriate tables of the provided MySQL DBMS via JDBC API. There are 5 input files in total listed in the subsections. Each line of each input file is a sequence of attributes delimited by a comma (**,**) character. The definition of each attribute in each input file is defined in the corresponding subsections. The order of the attributes within a line of each input file follows that of the attribute in the corresponding subsection. A sample data set will be provided after the deadline of Phase 1.

## 3.1. Employee – employee.csv

| Attribute Name | Format | Description |
|---|---|---|
| **Employee_ID** | A non-empty string with 6 characters | A unique identifier for the employee. |
| **Name** | A non-empty string with at most 30 characters | The name of the employee. |
| **Expected_Salary** | A positive integer | The lower bound of the employee's acceptable salary. |
| **Experience** | A positive integer | It shows how many years the employee had been working. |
| **Skills** | A non-empty string with at most 50 characters | It shows all titles of positions for which the employee is qualified.<br>The format is: $title_1;title_2;\ldots;title_n$ |

## 3.2. Company – company.csv

| Attribute Name | Format | Description |
|---|---|---|
| **Company** | A non-empty string with 30 characters | The name of the company, which uniquely identifies a company. |
| **Size** | A positive integer | It shows how many people working in the company. |
| **Founded** | A positive integer with 4 digits | The full year the company is founded. |

## 3.3. Employer – employer.csv

| Attribute Name | Format | Description |
|---|---|---|
| **Employer_ID** | A non-empty string with 6 characters | A unique identifier for the employer. |
| **Name** | A non-empty string with at most 30 characters | The name of the employer. |
| **Company** | A non-empty string with at most 30 characters | The company that the employer belongs to. |

### 3.4. Position – position.csv

| Attribute Name | Format | Description |
|---|---|---|
| **Position_ID** | A non-empty string with 6 characters | A unique identifier for the position. |
| **Position_Title** | A non-empty string with 30 characters | The title of the position. |
| **Salary** | A positive integer | The upper bound of salary that the company can offer for the position. |
| **Experience** | A non-negative integer | It shows how many years of working experience required for the position. |
| **Employer_ID** | A non-empty string with 4 characters | The ID of the employer who post the position recruitment. |
| **Status** | Bool | True/False means valid/invalid. |

### 3.5. Employment History – history.csv

| Attribute Name | Format | Description |
|---|---|---|
| **Employee_ID** | A non-empty string with 6 characters | The ID of the employee. |
| **Company** | A non-empty string with at most 30 characters | The company that the employee worked. |
| **Position_ID** | A non-empty string with 6 characters | The position that the employee served. |
| **Start** | Date format see 4.1 | The date when the employee started working in the company. |
| **End** | Date format see 4.1&4.6 | The date when the employee stopped working in the company. |

# 4. Assumption and Regulations

## 4.1. System

- All numerical values will not be larger than the maximum integer value that can be handled by Java.
- All dates follow the format "YYYY-MM-DD", e.g. "2018-06-09".
- There is no duplicate row in any input files.
- You may assume that any user inputs to the system are **correct in format only**.
- You may assume that all data files are **correct in format and content**.

## 4.2. Employee

- Each employee is uniquely identified by his/her Employee_ID.
- Each employee cannot work at more than one company at the same time.
- Each employee cannot serve at more than one position at the same time.
- Each employee can mark multiple positions as interested.

## 4.3. Employer

- Each employer is uniquely identified by his/her Employer_ID.
- Each employer should belong to only one company.
- Each employer can post multiple positions recruitments.

## 4.4. Company

- Each company is uniquely identified by its name, i.e. the Company attribute.

## 4.5. Position Recruitment

- Each position is uniquely identified by its Position_ID.
- Each position is posted by exactly one employer.

## 4.6. Employment History

- The End attribute will be NULL if the employee is still working in the company.

# 5. System Function Requirements

You are required to write a simple command line application in Java. After performing an operation, the program should display the last appeared menu. The Java program should provide the following functions:

## 5.1. Administrator

The system should let administrators to perform the following operations:

- **Create tables in the database**

  This function creates all the tables for this system based on the relational schema given.

```
Welcome! Who are you?
1. An administrator
2. An employee
3. An employer
4. Exit
Please enter [1-4].
1
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
1
Processing...Done! Tables are created!
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
```

Figure 1: Example output of creating tables.

- **Delete tables in the database**

  This function deletes all existing tables in the system.

```
Processing...Done! Tables are created!
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
2
Processing...Done! Tables are deleted!
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
```

Figure 2: Example output of deleting tables.

- **Load data**

  Prompt for user input, the system should then read all data files from the input folder path and load them into the appropriate tables in the database. Your program can assume that the user input folder

contains all 5 data files, namely employee.csv, company.csv, employer.csv, position.csv, and history.csv. See section 3 for data files specifications.

```
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
3
Please enter the folder path.
test_data
Processing...Data is loaded!
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
```

Figure 3: Example input and output of loading data.

- **Check tables in the database**

  For each table in the database, display the number of records in it.

```
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
4
Number of records in each table:
Table1: xxx
Table2: xxx
Table3: xxx
Table4: xxx
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
```

Figure 4: Example output of showing number of records in each table.

Note: Please replace words in Italic (i.e. Table1: XXX, Table2: XXX, Table3: XXX, Table4: XXX) in figure 4 with the tables in relational schema given in the suggested solution of phase 1. The number of tables may not be the same as shown in Figure 4.

## 5.2. Employee

- **Show Available Positions**

  A position is available to an employee if the following criteria hold:

  1. the Status of the position is True(valid);
  2. the employee is qualified for the position(i.e. the title of the position is included in the skills of the employee);
  3. the Salary of the position is not less than the Expected_Salary of the employee;
  4. the Experience of the employee is not less than the required Experience of the position.

After the employee enters his/her Employee_ID, the system shall return all available records.
Each record should contain the detail of the position including Position_ID, Position_Title, Salary, and the detail of the company including Company, Size, Founded.

```
Welcome! Who are you?
1. An administrator
2. An employee
3. An employer
4. Exit
Please enter [1-4].
2
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
1
Please enter your ID.
002301
Your available poistions are:
Position_ID, Position_Title, Salary,Company, Size, Founded
pid001, hokage, 90000, village leaves, 10000, 1999
pid009, writer, 80000, world, 60000, 1900
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
```

Figure 5: Example input and output of Showing Available Positions.

- **Mark Interested Position**

An employee may be interested in a position if:

1. the position is available to him/her;
2. the position is not from any company he/she worked in before(you need to check the Employment History);
3. the position has not been marked as interested by the employee before.

After the employee enters his/her Employee_ID, the system shall show all positions that the employee may be interested, then prompt the employee to mark one position as interested, and save this information.

```
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
2
Please enter your ID.
002301
Your interested poistions are:
Position_ID, Position_Title, Salary,Company, Size, Founded
pid009, writer, 80000, world, 60000, 1900
Please enter one interested Position_ID.
pid009
Done.
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
```

Figure 6: Example input and output of Marking Interested Position.

- **Check Average Working Time**
  An employee can check his/her average working time. The average working time is defined by the average of the working **days** of the last 3 records (excluding the current position he/she servers). The system should display the average working time if enough records are found, otherwise, display an error message.

```
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
3
Please enter your ID.
002300
Your average working time is: 3650 days.
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
3
Please enter your ID.
002301
Less than 3 records.
Employee, what would you like to do?
1. Show Available Positions
2. Mark Interested Position
3. Check Average Working Time
4. Go back
Please enter [1-4].
```

Figure 7: Example input and output of Checking Average Working Time.

## 5.3. Employer

The system should let employers to perform the following operations:

- **Post Position Recruitment**
  Employers can post a position with:
  1. A position title
  2. A upper bound of salary
  3. Required experience (optional)

  After the employer enters his/her Employer_ID, the system should prompt for each criterion. The employer should input the first two criteria, and can press enter without any input to skip the third criterion. The system then count for all potential employees. If there is at least 1 potential employee who meets all the criteria, then the system should post the position requirement, and display the number of potential employees to the employer. Otherwise return an error message for the employer.
  A potential employee is an employee that:
  1. is not working in any company currently;
  2. meet all the criteria: for criterion 1, the employee's Skills should contain the position title; for criterion 2, the employee's Expected_Salary should not larger than the upper bound of salary; for criterion 3, the employee's Experience should not less than the input experience (no input means experience=0)

```
Welcome! Who are you?
1. An administrator
2. An employee
3. An employer
4. Exit
Please enter [1-4].
3
Employer, what would you like to do?
1. Post Position Recruitment
2. Check employees and arrange an interview
3. Accept an employee
4. Go back
Please enter [1-4].
1
Please enter your ID.
002302
Please enter the position title.
doctor
Please an upper bound of salary.
30000
Please enter the required experience(press enter to skip).
2
10 potential employees are found. The position recruitment is posted.
Employer, what would you like to do?
1. Post Position Recruitment
2. Check employees and arrange an interview
3. Accept an employee
4. Go back
Please enter [1-4].
```

Figure 8: Example input and output of Posting Position Recruitment.

- **Check employees and arrange an interview**

  Employers can check which employees are interested in a specific position recruitment, and then he/she can arrange an interview to one of them.

  After the employer enters his/her Employer_ID, the system should show the Position_ID of all the position recruitments posted by the employer, and ask the employer to pick one of the position recruitments. After the employer specifies the position recruitment, the system should display all the details of employees who mark interested in the position recruitment, and then ask the employer to pick one Employee_ID of the employees to arrange an **IMMEDIATE** interview. We assume the interview will be done immediately.

```
Employer, what would you like to do?
1. Post Position Recruitment
2. Check employees and arrange an interview
3. Accept an employee
4. Go back
Please enter [1-4].
2
Please enter your ID.
002302
The id of position recruitments posted by you are:
pid008
pid009
Please pick one position id.
pid009
The employees who mark interested in this position recruitment are:
Employee_ID,Name,Expected_Salary,Experience,Skills
002301, Jiraya, 10000, 30, writer;teacher
Please pick one employee by Employee_ID.
002301
An IMMEDIATE interview has done.
```

Figure 9: Example input and output of checking employees and arranging an interview.

- **Accept an employee**

  An Employer can hire a suitable employee into the company.

  An employee is regarded as suitable iff the employer has arranged an interview to him/her.

  After the employer enters his/her Employer_ID and an Employee_ID, the system should check whether the employee is suitable or not. If suitable, the system should create an Employment History record, then show this record to the employer, and mark the status of the corresponding position recruitment to be invalid. If not suitable, display an error message.

```
Employer, what would you like to do?
1. Post Position Recruitment
2. Check employees and arrange an interview
3. Accept an employee
4. Go back
Please enter [1-4].
3
Please enter your ID.
002302
Please enter the Employee_ID you want to hire.
002301
An Employment History record is created, details are:
Employee_ID, Company, Position_ID, Start, End
002301, world, pid009, 2019-01-01, NULL
```

Figure 10: Example input and output of accepting an employee.

## 5.4. Error Handling

If there is a runtime error, including data not found, incorrect input, and etc. the system should output an information message in layman terms and in a new line as shown below. You are not required to handle all possible errors, just try some.

```
Employer, what would you like to do?
1. Post Position Recruitment
2. Check employees and arrange an interview
3. Accept an employee
4. Go back
Please enter [1-4].
5
[ERROR] Invalid input.
Please enter [1-4].
3
Please enter your ID.
002302
Please enter the Employee_ID you want to hire.
0
The employee is not suitable. Please enter another Employee_ID you want to hire.
```

Figure 11: Example output when an error occurs.

Please note that the outputs of examples in all figures are not model answers of the testing data.

# 6. Grading Policy

The marks are distributed as follows:

| Phase | Content | Mark Distribution |
|---|---|---|
| 1 | ER-diagram | 10% |
| | Relational schema (based on your ER-diagram) | 10% |
| 2 | Java application | 80% |

- There will be a mark deduction if your application is terminated unexpectedly during the demonstration.
- You are not allowed to modify any source code during the demonstration.
- All members in the same group will receive the same marks for the project. In order to encourage every student to participate in the project, a question about this project may be asked in the final examination.

# 7. Demonstration

- All groups need to sign up for a demonstration on their works for phase 2, the registration page would be posted on the course website later.
- All group members should attend the demonstration.
- The duration for the demonstration for each group is about 30 minutes.
- The Java application will be **complied** and **tested** in a Linux 64-bit machine in the CSE department.
- The dataset used in the demonstration may be different from the dataset provided for testing.

# 8. Submission Methods

## 8.1. Phase 1

- Submit a PDF file (one copy for each group) to the collection box at Blackboard platform.
- The PDF file should consist of your groups ER diagram, relational schema, the group number, the names and the student IDs of all group members of your group.

## 8.2. Phase 2

- Submit a ZIP file (one copy for each group) to the collection box at Blackboard platform. The ZIP file should consist of all your source codes and a README file (README.txt), which contains:
  - The group number of your group
  - The name and the student ID of each group members of your group
  - Methods of compilation and execution