CSCI3170 (2021-2022 Term 2)
Introduction to Database Systems
Project – Car Renting System

TA In Charge: ZHOU, Qihui (qhzhou@cse.cuhk.edu.hk)

Group Registration Deadline: 23:59 2022/2/6
Phase 1 Deadline: 23:59 2022/2/27
Phase 2 Deadline: 23:59 2022/4/10

1. Introduction

You are required to implement a Car Renting System. The system shall support interactive operations from operators. You are required to use Java JDBC API to access the database. Our tutors will give a tutorial on how to use the JDBC API. You are required to implement a Java command line application to realize all system functions stated in this specification.

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

In Phase 2, you are required to implement the Car Renting 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 three members. ONLY one copy of solution is required for each group. Please fill out the group registration form in 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 Specification

All data files of the system are in Linux text file format (i.e., Newline character is **\n**) encoded in ASCII. Your Java command line application is required to read records stored in the files and inserts them into appropriate tables of the provided MySQL DBMS via JDBC API. There are four five files, a list of user categories, a list of users, a list of car categories, a list of cars and a list of renting records. Each line of each input file is a sequence of attributes delimited by a tab (**\t**) 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. User Categories – user_category.txt

Users are divided into different categories. The loan period and the maximum number of cars that can be borrowed by a user are determined by his or her category. Each user category has a unique user category ID.

Attribute Name	Format	Description
User Category ID (e.g. 0)	Non-empty positive integer with 1 digit	A unique identifier for a user category.
Max cars (e.g. 5)	Non-empty positive integer with 1 digit	The maximum number of cars that can be borrowed by the library user in the corresponding category.
Loan period (e.g. 16)	Non-empty positive integer with at most 2 digits	The maximum number of days for the user in the corresponding category to rent a car.

3.2. Users – user.txt

Each user has a unique user ID and belongs to exactly one category.

Attribute Name	Format	Description
User ID (e.g. s1155152066)	Non-empty string with 12 characters	A unique identifier of the user.
Name (e.g. ZHOU Qihui)	Non-empty string with at most 25 characters	The name of the user.
Age (e.g. 23)	Non-empty positive integer with 2 digits	The age of the user.
Occupation (e.g. Student)	Non-empty string with at most 20 characters	The job of the user.
User Category ID (e.g. 0)	Non-empty positive integer with 1 digit	It indicates to which category the user belongs.

3.3. Car Categories – car_category.txt

Cars are divided into different categories. Each car category has a unique car category ID.

Attribute Name	Format	Description
Car Category ID (e.g. 0)	Non-empty positive integer with 1 digit	A unique identifier for a car category.
Car Category Name (e.g. van)	Non-empty string with at most 20 characters	The name of the car category.

3.4. Cars – car.txt

There are many different types of cars and there are multiple cars of the sample type. Each type of car has a unique call number.

Attribute Name	Format	Description
Call number (e.g. abcdefgh)	Non-empty string with 8 characters	It is used for the users to search for the car.
Number of copies (e.g. 6)	Non-empty positive integer with 1 digit	The number of identical copies of the car.
Car Name (e.g. AE 86)	Non-empty string with at most 10 characters	The name of the car.
Company (e.g. TOYOTA)	Non-empty string with at most 25 characters.	The name of the company that produce the car.
Date of manufacture	Date format see 4.2	The date that the car is produced. (We assume that the same type of cars are produced at the same date)
Number of times rented (e.g. 99)	Non-empty non-negative integer with 2 digits	The number of times the car has been rented.
Car Category ID (e.g. 1)	Non-empty positive integer with 1 digit	A unique identifier for a car category.

3.5. Renting Records - rent.txt

Each renting record shows a renting history of a user.

Attribute Name	Format	Description
Call number(e.g. abcdefgh)	Non-empty 8 characters	The call number of the car copy.
Copy Number (e.g. 1)	Non-empty 1-digit Positive integer	The copy number of the rented car
User ID (e.g. s1155152066)	Non-empty 10 characters	The user ID of the user.
Rent Date	Date format see 4.2	The date that the car are rented.
Return date	Date format see 4.2	The date that the car is returned.

4. System Function Requirements

You are required to write a simple command line application in Java. After performing a function, the program should **display the last appeared menu**. The following sections describe the functionalities of the system.

In the system, you're required to ask the user to choose one of the three interfaces – Administrator, User, and Manager.

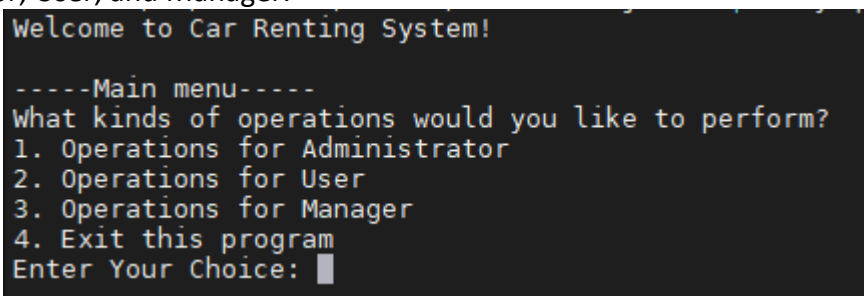

Figure 1: Example main menu.

4.1. Administrator

The system should let administrators to perform the following operations:

- **Create table schemas in the database**: This function creates all the tables for this system based on the relational schema given.

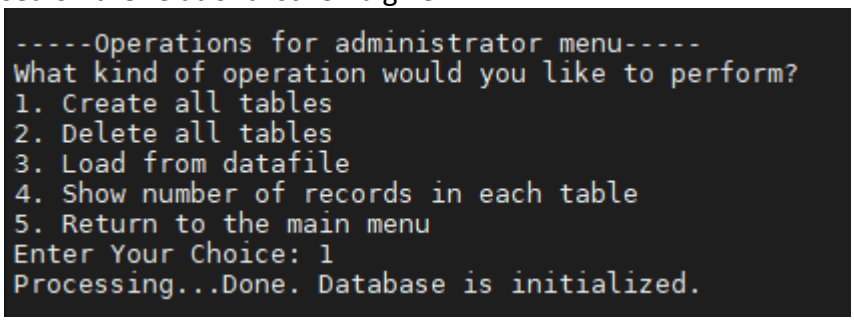

Figure 2: Example interactive input and output while creating table schemas.

- **Delete table schemas in the database**: This function deletes all existing tables in the system.

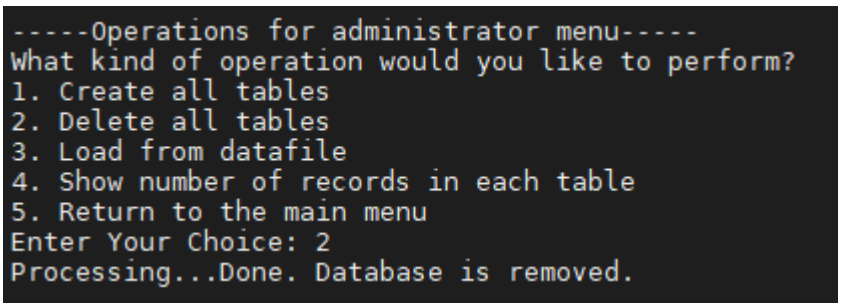

Figure 3: Example interactive input and output while deleting table schemas.

- **Load data from a dataset**: After a user enters the path of the folder that contains the data files, the system reads all data files from the user-specified folder and inserts the records into the appropriate table in the database. (Your program can assume that the user-specified folder must contain all 5 data files. These 5 input files are named user_category.txt, user.txt, car_category.txt, car.txt and rent.txt. Each data file stores the data corresponding to its filename.)

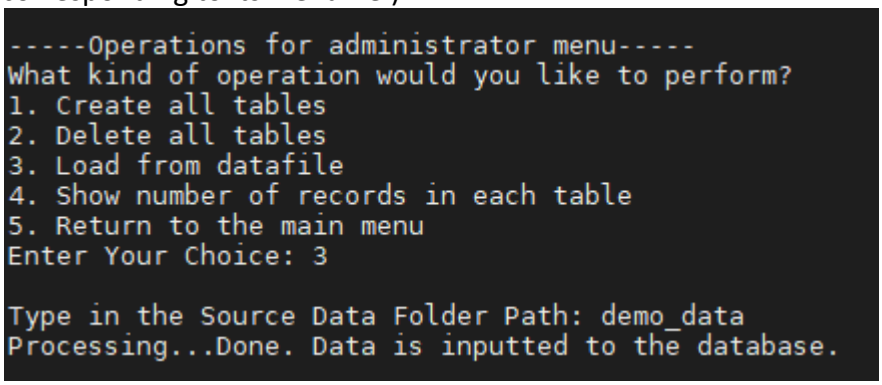

Figure 4: Example interactive input and output while loading table schemas from the database

- **Show the number of records in each table:** For each table in the database, display the number of records in it.

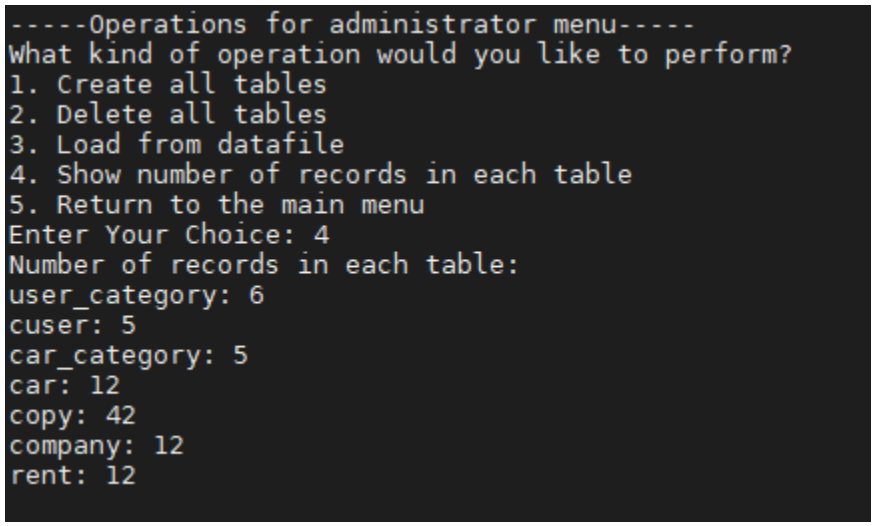

Figure 5: Example interactive input and output while showing number of records in each table.

4.2. User

- **Search for cars**: The system is required to provide an interface to allow a user to search for the cars in three different ways:
  - By call number (exact matching)

- By car name (partial matching)
- By company (partial matching)

You can assume that only one searching method can be selected by the user for each query and the whole string entered by the user is considered as one search word (e.g., When a user entered “AE 86”, The system will consider “AE 86” as one and only one search keyword instead of two search keywords “AE” and “86”). After the user entered the search keyword, the program should perform the query and return all matching cars. The results of the query should be sorted in ascending order of *call number* and outputted as a table as follows:

```
Welcome to Car Renting System!

-----Main menu-----
What kinds of operations would you like to perform?
1. Operations for Administrator
2. Operations for User
3. Operations for Manager
4. Exit this program
Enter Your Choice: 2

-----Operations for user menu-----
What kind of operation would you like to perform?
1. Search for Cars
2. Show loan record of a user
3. Return to the main menu
Enter Your Choice: 1
Choose the Search criterion:
1. call number
2. name
3. company
Choose the search criterion: 2
Type in the Search Keyword:Car1
|Call Num|Name|Car Category|Company|Available No. of Copy|
|97801326|Car12|Van|Company12|6|
|GA769D3K|Car11|Van|Company11|6|
|QA762011|Car1|Van|Company1|0|
|S5583200|Car10|Race|Company10|4|
End of Query

-----Operations for user menu-----
What kind of operation would you like to perform?
1. Search for Cars
2. Show loan record of a user
3. Return to the main menu
```

Figure 6: Example input and output while searching for Cars

- **Show all renting records of a user:** The system is required to provide an interface to allow a user to show all his/her renting records of with a given *user ID*. After the user enters his/her *user ID*, the program will perform the query and return all the matching records. The records should be sorted in descending order of *renting date* and outputted as a table as follows:

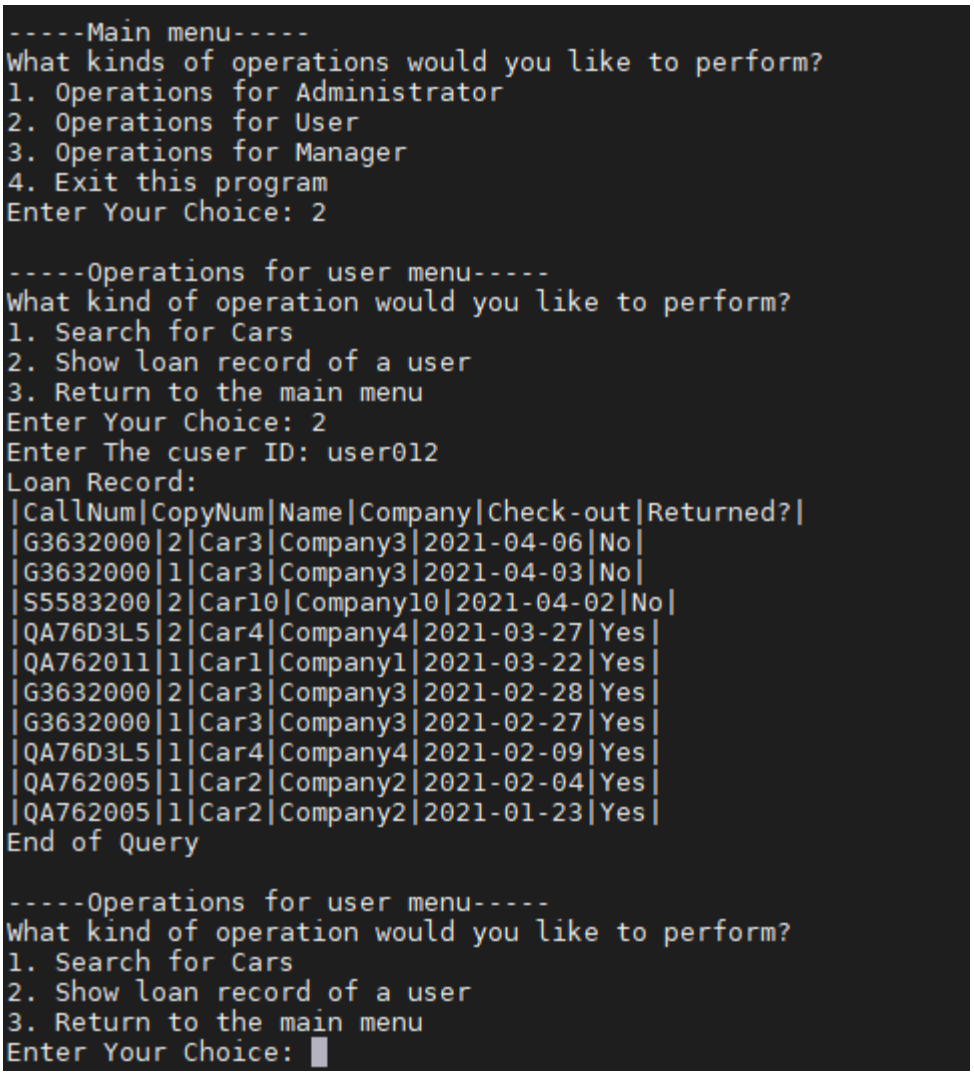

Figure 7: Example input and output while showing all check-out records of a user

4.3. Manager

- **Rent a car copy:** A manager can perform the car renting procedure through the car renting System. First, he/she needs to input *call number* and *copy number* of the car copy being borrowed and the *user ID* of the user. Then the system should check whether that car is available to be rented (i.e., There is no rent record of the specified car copy with NULL *return date*). If the car copy is available, it is then borrowed and a new check-out record of the specified car copy and user with NULL *return date* should be added to the database accordingly. Finally, there should be an informative message whether the car copy can be lent successfully in layman terms.

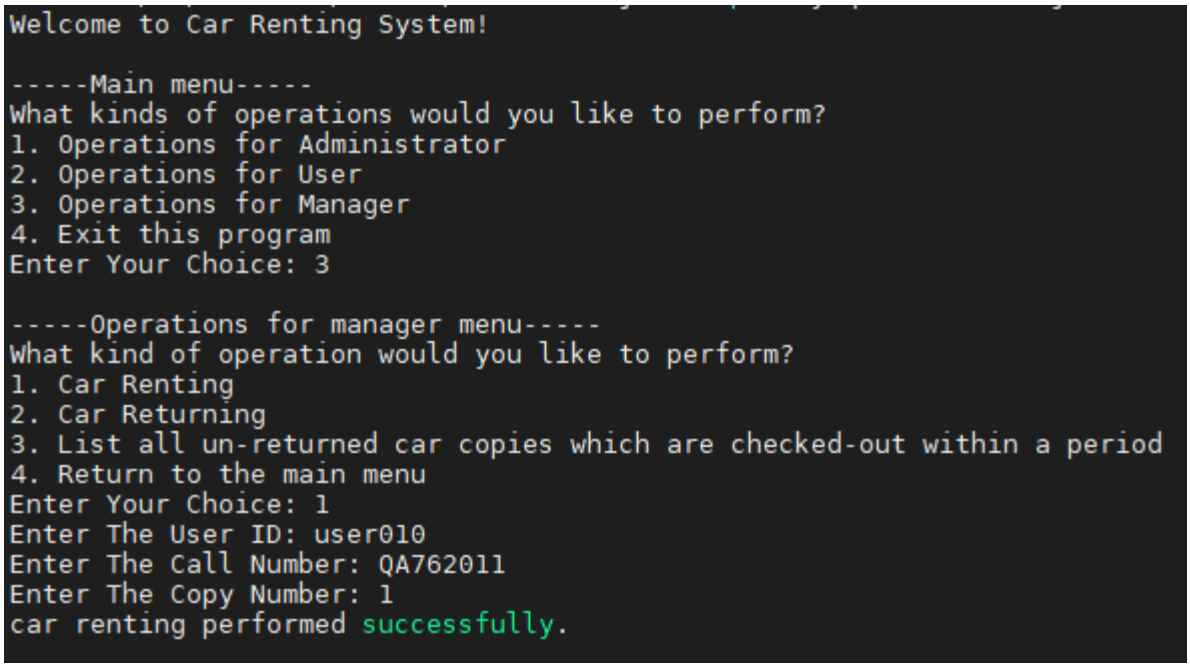

Figure 8: Example input and output while a manager processes a car renting request

- **Return a car:** A manager can perform the car returning procedure through the car renting System. First, he/she needs to input *call number* and *copy number* of the car copy being borrowed and the *user ID* of the user. Then the system should check if a renting record corresponding to the specified *user ID*, *call number* and *copy number* exists and the return date is null. If such record is found, the car copy can be returned, and the *return date* of the renting record found is updated to be the current date of the database server.
- Finally, there should be an informative message whether the car copy can be returned successfully in layman terms. (For the sake of simplicity, you are not required to check whether the car is overdue or not)

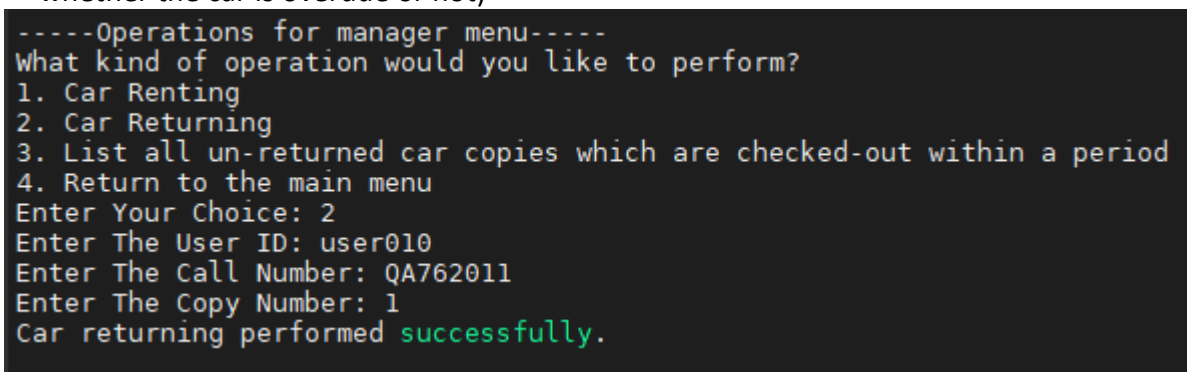

Figure 9: Example input and output while a manager processes a car returning request

- **List all un-returned car copies which are checked-out within a period:** The system is required to provide an interface to allow a manager to list all un-returned car copies which are checked-out within a given period (e.g., from 20/03/2021 to 20/04/2021). After the manager enters the period, the program will perform the query and return a

list of all un-returned car copies in terms of *user ID, call number*, *copy number and check-out date* in descending order of *check-out date* within the inputted period *inclusively*.

```
-----Operations for manager menu-----
What kind of operation would you like to perform?
1. Car Renting
2. Car Returning
3. List all un-returned car copies which are checked-out within a period
4. Return to the main menu
Enter Your Choice: 3
Type in the starting date [dd/mm/yyyy]: 01/01/2021
Type in the ending date [dd/mm/yyyy]: 12/12/2021
List of UnReturned Cars:
|UID|CallNum|CopyNum|Checkout|
|user011|QA762011|1|2021-04-10|
|user012|G3632000|2|2021-04-06|
|user012|G3632000|1|2021-04-03|
|user012|S5583200|2|2021-04-02|
End of Query
```

Figure 10: Example input and output while listing out all unreturned cars

4.4. Error Handling

If a run-time error occurs, the Car renting 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.

```
-----Operations for manager menu-----
What kind of operation would you like to perform?
1. Car Renting
2. Car Returning
3. List all un-returned car copies which are checked-out within a period
4. Return to the main menu
Enter Your Choice: 2
Enter The User ID: user010
Enter The Call Number: QA762011
Enter The Copy Number: 4
[Error]: No Matching car copy found.
```

Figure 11: Example input and output when an error occurs.

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

5. 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.

6. Demonstration

- Depending on pandemic situation, we will hold a face-to-face demonstration or online demonstration.
- All group members should attend the demonstration.
- The duration for the demonstration for each group is about 15 minutes.
- The Java application will be **complied** and **tested** in a Linux 64-bit machine of the CSE department.
- The dataset used in the demonstration may be different from the dataset provided for testing.

7. Submission Methods

7.1. Phase 1

- Submit a PDF file (one copy for each group) to the Blackboard system.
- The PDF file should include an ER diagram, a relational schema, the group number, the names, and the student IDs of all group members of your group.

7.2. Phase 2

- Submit a ZIP file (one copy for each group) to the Blackboard system. The ZIP file should include all your source codes and a README file (README.txt), which contains:
  - Your group number
  - The name and the student ID of each group member
  - Instructions on how to compile and run your system