|  |  |
| --- | --- |
|  | INFO-6120  Project – Part 2  Due Date: August 6, 2023, 11:59pm |

**Goal/Objective**

Now that we have designed a database for Yoober, we need an easy way to view and manipulate the data in the database. To do this, you are being asked to help develop a console application to communicate with your database.

The requirements for the console application are as follows:

* When starting the application, a welcome message is shown
* Immediately following the welcome message, a menu of options supported by the application should be shown. The supported options should be:
  1. View all account details
  2. Calculate the average rating for a specific driver
  3. Create a new account
  4. Submit a ride request
  5. Complete a ride
* Then, the application should accept user input to select a menu option. The user will enter the number of the menu option they wish to perform. For example, if they want to view all account details, they would enter the number **1**.
* Based on the option entered by the user, the application should handle the request appropriately. The required behaviour for each option is as follows:
  1. The following data for each account should be displayed to the user:
     + First and last name
     + Full address (i.e. street, city, province, postal code)
     + Phone number
     + Email address
     + Whether account is used by a passenger, driver, or both
  2. The user should be instructed to enter the email address of the driver in which they are interested. Then, the average of all available ratings given by passengers for all trips the specified driver has provided should be calculated and displayed to the user.
  3. With the data provided by the user specified below, insert new records into the database as appropriate. The user should be prompted to enter values for:
     + - First and last name
       - Birthdate
       - Full address (i.e. street, city, province, postal code)
       - Phone number
       - Email address
     + Then, the user should be asked if the new account will be used by a passenger, driver, or both.
       - If passenger, prompt user to enter:
         * Credit card number
       - If driver, prompt user to enter
         * Driver’s license number
         * Driver’s license expiry date
       - If both, prompt the user to enter the information listed above for passengers and drivers
  4. With the data provided by the user specified below, insert new records as appropriate into the database. The user should be instructed to enter:
     + - The email address of the passenger making the request
     + Then, ask the user whether they want to choose their destination from the specified passenger’s list of favourite destinations
       - If yes:
         * Display a list containing the name and full address details of all the passenger’s favourite destinations
         * Prompt the user to enter the name corresponding to their choice of destination from their list of favourites
       - If no:
         * Prompt the user to enter the full address details of their destination. Save this address in the database if necessary.
         * Ask the user if they want to make this destination a new favourite

If they do, prompt the user to provide a name for the location, and add the destination as a new favourite location for the passenger

* + - Then, prompt the user to enter:
      * Desired pick-up date and time
      * Total number of riders
    - For simplicity, we will assume the pick-up location is always the address tied to the passenger’s account
  1. Display a list containing the following details of all **uncompleted** rides requests:
     + - ID
       - First and last name of passenger who requested the trip
       - Street and city of both the pick-up and destination addresses
       - Desired pick-up date and time
     + Then, prompt the user to enter the ID corresponding to the ride they want to complete (assume input will always be a number)
     + Complete the ride in the database with the following information provided by the user:
       - Driver’s email address
         * Use this to help specify the driver for the trip
       - Actual start/end date/time
       - Distance travelled
       - Cost
       - Driver’s rating
       - Passenger’s rating
* Every time the user selects a menu option and finishes the option’s workflow, they should be prompted again to select a new option. They may exit the application by entering the word “exit” instead of a menu option.
* Remember, all addresses should only be stored in the database once. So, whenever address details are provided by the user, a new address should only be created in the database if it doesn’t already exist.
* For options 2, 4, and 5, if no passenger/driver can be found with the email address entered by the user, indicate this to the user and ask for a new email address until a valid one is provided
* When the user is expected to enter numeric information (e.g. distance travelled for trip), assume they input valid data. There is no requirement to validate the input is numeric.

**Instructions**

You will be provided with a program that contains code for the overall workflow of the project. This code will already handle the menu option functionality, reading user input, and basic validation. Your job for the project is to implement all the methods in **src > database > DatabaseMethods.java**. The methods will perform all necessary database interactions needed to support our application.

DO NOT modify any code in the program EXCEPT for the method bodies in the **DatabaseMethods** class, as well as your group members’ names in the header comment.

**Technical Requirements**

You must use the provided starter program, and only modify the method bodies in the **DatabaseMethods** class. Do not modify the method signatures in **DatabaseMethods,** and do not modify any other files. Test your work by running the program and confirming the data returned, or data inserted into the database, matches what you expect, and that no errors are thrown.

**Submission Details**

You will be working in groups of two to complete the project. You will submit ONLY the completed **DatabaseMethods.java** file, not the entire Java project.

The project is due on August 6, 2023, by 11:59pm.

**Grading**

Each method implementation in **DatabaseMethods.java** will be graded according to the rubric below. Any method that throws an exception while testing will automatically receive a mark of 0.

|  |  |
| --- | --- |
| Method | Available Mark |
| getAllAccounts | 2 |
| getAverageRatingForDriver | 2 |
| createAccount | 4 |
| insertAccount | 2 |
| insertPassenger | 1 |
| insertDriver | 2 |
| insertLicense | 1 |
| insertAddressIfExists | 2 |
| insertFavouriteDestination | 2 |
| checkDriverExists | 2 |
| checkPassengerExists | 2 |
| getPassengerIdFromEmail | 1 |
| getDriverIdFromEmail | 1 |
| getAccountAddressIdFromEmail | 1 |
| getFavouriteDestinationsForPassenger | 2 |
| getUncompletedRideRequests | 2 |
| insertRideRequest | 2 |
| insertRide | 2 |
| TOTAL | **33** |

If you have any questions related to the project, please don’t hesitate to contact your professor as soon as possible.