FINAL GROUP ASSIGNMENT (35%)

Due: 20 January 2024, 11:59 PM (GMT +7)

OBJECTIVES

This assignment provides a chance for students to practice and apply all concepts learned so far. It is also an opportunity to get familiar with analysis, design and development of a software application with a ***practical project idea*** to get ready for life and work. For simplicity, all interaction with the application will be done via a simple ***text interface (no GUI is required)***. However, it will provide all basic functionalities of a practical application (may be further developed after completion).

DESCRIPTION

Your task for this assignment is to design and implement a program in C++ namely "Time Bank" in Viet Nam.

"TIME BANK" APP

Assume that you are involved in a start-up who came up with an idea to make an app namely "Time Bank" for people to support each other in daily activities, such as plumbing repairs, tutoring, garden care, house cleaning, etc...

There is an initial entry fee of $20 (pay to the system) when **registering** as a member, which earns the new member ***20 credit points***. Initial registrations must capture all the information necessary to be a member, including his/her personal info (*username, full name, phone number, email, home address*), and the info of his/her skills (*English tutoring, children care, etc*..)

The app will allow a registered member to ***earn credit points*** *by helping other members. In contrast, he/she can* ***use the credit points to book other members for help****.* If a member does not have enough credit points, he/she needs to help other members or can **perform a top up** of the credit points using cash ($1 = 1 credit point).

The system will track three separate **score ratings** which can vary from **1** to **5**.

- The **skill-rating score** and **supporter-rating score** is based on the average of ratings of the hosts who asked this member to help in the past (*how well this supporter helped the hosts*). The hosts will rate the supporter for the **skill** performed (*skill-rating*), and his overall performance/kindness (*supporter-rating*).
- The **host-rating score** is derived by averaging the ratings of all the supporters who came to help this member (*how kind this host treated the supporters*).

Any member can **list himself/herself** to be available on particular **periods** (start time – end time for each period), the **skill(s) that can be performed**, **consuming points** (per hour), and with or without specifying ***minimum required* host-rating score** for potential hosts. The *minimum host-rating* can thus be used to prevent the member from being booked by poor rating hosts or those for which no history is available. Of course, the member can **unlist the himself/herself** if needed.

Any member can **view all available supporters** in a **specified time** (start time – end time), and **city in which the supporters are located**. The display list should consist of only those **supporters** for which the member has *adequate credit points and host-rating score.*

The information should also include the *rating scores* of the supporters (skill rating and overall supporter-rating). Members should also be given the option to **view the reviews** (scores and comments) on any of the available supporters.

Any member can **request to book** any supporter for which they are eligible by using their credit points.

Any member can **block other members** who had some conflicts, which will prevent them from view or request for support.

Any member can **view all requests for his/her skilled activity**, including *info and rating of the interested hosts,* and may **choose *a request to reject or accept*** *(automatically reject all other requests with overlapped time)*.

Non-members can **view all supporters' details** (but not their reviews, i.e. score and comments), to encourage non-members to join. Non-members can only request the supporters if they register to become members.

CONSTRAINTS:

For the initial version of the app, some constraints are applied as below:

1. Members are not allowed to cancel if the request is already accepted.
2. The application is only available to users in Ha Noi and Sai Gon.

ASSUMPTIONS:

Transaction that involves the top up of credit points using cash can be authorized with the member's password only.

PROCESS AND GUIDE

A. Class Diagram: Identify classes, attributes, methods and relationships between classes. Sketch a class diagram for the application.

B. *Basic Features:*

Implement and test each feature. The application should have the following basic features:

Note: The program should validate the user's input accordingly. All attributes should be private for data encapsulation.

1. A non-member can register to become a member (information is recorded).
2. A non-member can view all supporters' details (but not their reviews).
3. An admin can login with predefined username and password, and can reset password for any member.
4. A member can login with the registered username and password, and can view all of his/her information.
5. A member can list himself/herself available to be booked (including skills can be performed, consuming points per hour, with/without minimum required host-rating score), and unlist if wanted.
6. A member can search for all available **suitable** supporters for a specified **city** (suitable with his current credit points and rating score).

   Example:

   My credit points is 20, and my host-rating score is 5

There is a supporter with consuming point: 50 credit points/hour and minimum required host-rating score is 4.5.

This supporter won't be available in my search (since I do not have enough credit points to pay).

7. A member can request to book a supporter.
8. A member can view all requests to his listed skills.
9. A member can reject or accept a request (reject other requests).
10. A member can block another member from viewing his/her information or requesting support.
11. A member as a host can rate his/her supporter (score and comments).
12. A member as a supporter can rate his/her host (score and comments).
13. Data must be saved into data **file(s)** before the program is ended, and loaded into the program when it is started.

C. ***Time Period Feature***: when you already completed the basic features above, improve the application by supporting time period matter for the program as in its description, especially for features 5-9 *(e.g. list himself/herself available for specific time periods; search for a specified time period and city; request the supporter in a period; only automatically reject non-accepted requests with overlapped time, etc.).*

D. Welcome Screen

When completing your application, it should have a welcome screen with example content structure as below. You are free to adjust its format if prefer.

Note: … in the example content means so on and so forth depends on what you want to show.

```
EEET2482/COSC2082 ASSIGNMENT
"TIME BANK" APPLICATION

Instructor: Mr. Tran Duc Linh
Group: Group No.
sXXXXXXX, Student Name
sXXXXXXX, Student Name
sXXXXXXX, Student Name
sXXXXXXX, Student Name

Use the app as 1. Guest   2. Member   3. Admin

Enter your choice: 2
Enter username:
...

This is your menu:
0.  Exit
1.  View Information
2.  ...

Enter your choice: 1
...
```

IMPORTANT INSTRUCTIONS

1. You are required to write your program in C++ programming language only.
2. You will need to demonstrate the OOP skills that you have learned in the course.
3. You must thoroughly document your code using comments (// or /* … */).
4. Your code should be separated logically into multiple source code (.cpp) and header (.h) files that overall have a single purpose, i.e. your code should be structured in a way that each file has a clear scope and goal. Make sure you use the header files and header guard correctly.

ACADEMIC INTEGRITY

This assessment requires that you meet RMIT's expectations for academic integrity. You must not ask for or accept help from anyone else on completing the tasks. You must not show your work to another student/group enrolled in this course who might gain unfair advantage from it. These things are considered plagiarism or collusion. Plagiarism is a form of cheating. It is the presentation of the work, idea or creation of another person as though it is your own. If you use any resources, write an acknowledgment in the file where the resources were used. Any submission found in whole or in part plagiarized will be considered as plagiarism. To be on the safe side, never ever release your code to the public. For example, if you use GitHub to store your code, make sure that you set the repository as private. For more information, visit RMIT's website for Academic Integrity and Consequence of Plagiarism.

The penalty for plagiarism is extremely severe at RMIT. If you are found to plagiarize, you will receive a mark of zero for an assessment or even an entire course. Repeated plagiarism will lead to exclusion from RMIT.

Report and Presentation

Complete your **report** with class diagram and explanation using the provided Report Template.

For demonstration, please record a **video** of maximum 20 mins that present and demonstrate for your work. All members should present in the presentation video.

RUBRIC

Refer to the Rubric published in the Group Assignment page in the course Canvas.

SUBMISSION INSTRUCTIONS

As a group, submit following files separately (*only ONE member needs to do the submission***) to the course Canvas, DON'T zip them together:**

1. An executable file of your program, i.e. ***GroupX_Program.exe*** *(where X denotes your group number).*
2. A zip file of all source code including .h and .cpp files and data file, i.e. ***GroupX_Sources.zip.***
3. A report in PDF format, i.e. ***GroupX_Report.pdf***.
4. A text file with link of your presentation video, ***GroupX_Video.txt***.
5. Record of Individual Contribution in PDF format, i.e. ***GroupX_Individual_Contribution.pdf.*** Refer to the Individual Contribution form available in the Group Assignment page in the course Canvas.

Note: *The files must be submitted separately to TURNITIN, and make sure your executable file can run. If you do not submit it, or it fails to run/crashes, you may lose up to 50% of the grade (although some marks may be given for the effort).*

Late work: 10% penalty per day. Submissions later than 03 days will not be accepted and a mark of zero will be given.

--- END OF ASSIGNMENT ---