CS 408 - Computer Networks - Fall 2022

Course Project: Simple Quiz Game

This project is made of two steps; each has different deadlines as specified below.
All group members should submit.

Project Step 1 <u>Deadline</u>: November 30, 2022, 22:00
Project Step 1 <u>Demo</u>: to be announced

Project Step 2 <u>Deadline</u>: December 27, 2022, 22:00
Project Step 2 <u>Demo</u>: to be announced

Introduction

You are going to implement a client-server application, which is a simple quiz game to be played among some users. The application consists of two separate modules. To conduct the game, you need to build a **server** module. Furthermore, for your players to connect and play, you also implement a **client** module. In this document, the client module will also be referred simply as **player,** either term may be used to express the same concept.

The players will be able to enroll in an open quiz game by connecting to the server and will be able to play with other players. On the other hand, the server will act as a facilitator/presenter so that it handles incoming connections from players so that they can participate in a quiz game and play the game properly. The server asks the questions to the players. Also, the server should keep the scores during the game. It broadcasts the question based results after the end of taking the answers for each question. Moreover, it broadcasts the overall score table as well.

The server listens on a predefined port and accepts incoming client connections. There may be one or more clients connected to the server at the same time. Each client knows the **IP address** and the listening **port** of the server (to be entered through the Graphical User Interface (GUI)). Clients connect to the server on the corresponding port and identify themselves with their names. Server needs to keep the names of currently connected clients in order to avoid the same name being connected more than once at a given time to the server.

The abovementioned introduction is for the entire project, which is to be completed in two steps. Second step is built on the first step and each has specific deadlines and demos.

The details of each step and some other general information are given below. Please read them carefully.

Project Step 1 (Deadline: November 30, 2022, 22:00):

In this step, you are going to develop a server module that can accept *two* client connections. The connected clients should have unique names. Thus, if a client wants to connect with a name that is currently connected to the server, then the server should reject that name.

Number of questions to be asked by the server during the game should be entered through Server GUI. When there are precisely two successfully connected players and the number of the questions in the game is entered, the game should be started by the server automatically. After the game starts, the server does not accept any new client connections. The game is to be played as explained below.

1. The questions and their answers are provided to you in a txt file together with the project pack. The server should ask a question to the players by picking the next question from the file. Each question in this file has also its correct answer that will be needed by the server for scoring. Also assume that all answers are numeric.
2. The players receive this question and send their answers to the server.
3. When the server receives both of the answers, the server makes comparison with the correct answer and the player whose answer is the closest to the correct answer gets 1 point (exact correct answer is not a must). In case of a tie, both players get 0.5 point each. After grading, the server should inform the players about the winner/loser/tie of the current question by including the correct answer and both players' given answers.
4. Server should keep a score table in which the players total scores are listed in descending order. This score table must be sent to both players after each question.
5. The steps 1-4 will be repeated for each question in the game (number of questions times, which was entered through Server GUI). If the number of questions entered in the GUI is more than the number of questions in the file, then reuse the same questions from the beginning.
6. When the game finishes, the server should inform the clients about the end of the game. At this point, the sockets will be closed.

If one of the client disconnects during the game, then the game finishes abruptly. If disconnection occurs after asking a question, this question will be cancelled (i.e. not graded). In such a case, total points of the disconnected client become zero and the other user is declared as winner. Server should send reasonable message to the remaining user about this case. After that, the socket will be closed.

If the server disconnects, the game finishes abruptly and forgotten. The clients should understand this without a crash or getting frozen, and should go to initial state that they are disconnected. If the server comes back later, the clients can connect anew to start a brand new game from scratch.

It is possible to start a brand new game when the previous one ends. For this, new connections are needed, but you do not need to restart the server and client programs; i.e. already launched client and server programs should work.

For programming rules and submission specifications, please read the corresponding sections at the end of this document.

Server Specifications:

- There is only one server running on this system.
- The port number on which the server listens is not to be hardcoded; it should be taken from the Server GUI.
- The name of the question file can be taken from the server GUI or it can be written hardcoded.
- The server will start listening on the specified port. It must handle multiple clients simultaneously (in the first step two clients only, but in the second step more than two).
- All activities of the server should be reported using a rich text box on the Server GUI including names of connected clients as well as all the questions, answers, scoring, etc. Be as verbose as possible. We cannot grade your homework if we cannot follow what is going on; so, the details contained in this text box is very important.
- When the server application is closed (even abruptly), nothing should crash or freeze!

Client specifications:

- The server IP address and the port number must not be hardcoded and must be entered via the *Client GUI*.
- In the first step of the project, there will be two clients in the system. However, in the second phase, there could be any number of clients.
- If the server or the client application closes, the other party should understand disconnection and act accordingly as described above. Your program must not crash or freeze!
- All activities of the client should be reported using a rich text box on the *Client GUI* including connection information, question/answers as well as status of each turn, etc. Be as verbose as possible. We cannot grade your homework if we cannot follow what is going on, so the details contained in this text box is very important.
- Each client must have a name. This name must be entered using the *Client GUI*. This name is also sent to the server. The server identifies the clients using their names. Each client must have a unique name.
- Each client can disconnect from the system at any time. Disconnection can be done by pressing a button on Client GUI or just closing the client window.
- Both connection and message transfer operations will be performed using TCP sockets.

----------------- End of Step 1 ---------------------

Project Step 2 (Deadline: December 27, 2022, 22:00):

Second step of the project is built on the first step. In this step, you will modify previous client and server modules to add more functionalities and modify existing ones.

The aim of this step is to make the game a multiplayer one. Thus, now multiple clients should be able to connect to the server. Moreover, the flow and the rules of the game also changes in this step in order to accommodate multiple players, as detailed below.

1) The server accepts several client connections before the game starts. The rule for user name uniqueness continues in this step as well.
2) Server GUI still has the *number of questions* text box. Moreover, Server GUI should have one additional field in this step; a *start game* button. In this step, game does not start automatically, but starts when the *start game* button is clicked. However, there must be at least two clients connected in order to start the game.
3) The players of the game are the ones who are successfully connected when *start game* button is clicked. After that point, new players can connect to the server but will not be able to join the ongoing game. They will wait until the game ends and join the next game.
4) The rules of the game (asking questions, getting answers, scoring, etc.) are the same as first step, but this time we have multiple players and you have to adapt the game to multiple player case properly.
5) When a game finishes, as opposed to previous step, the sockets will not be closed.
6) The server can start a new game by pressing the start game button. This new game will be played among all players currently connected to the server.

You have to handle disconnections as discussed below.

1) The players in the game can disconnect at their will and this should not cause your program to crash. Each disconnected player is eliminated from the ongoing game by making its total point zero. Even if it reconnects, it cannot join the ongoing game (but can join the next one).
2) If the player disconnection occurs in the middle of a question, as opposed to first step, that question is not cancelled and the game does not end. The question is graded among the remaining players and the game continues with them.
3) At any time during the game, if there is only one connected player remaining, the server should announce that player as winner and finish the game even if not all the questions have been asked.
4) If the server disconnects, the game finishes abruptly and forgotten. The clients should understand this without a crash or getting frozen, and should go to initial state that they are disconnected. If the server comes back later, the clients can connect anew for the next game.

As in the step 1, all operations must be clearly shown on the client and server GUIs. For all operations, successful and unsuccessful connection messages, asked questions and answers, scoring details each sent and received message during the game should be shown in detail at the GUIs.

----------------- End of Step 2 ---------------------

Group Work

- You can work in groups of minimum **four** and maximum **five** people for both steps (not less than four except really exceptional cases). No group changes are allowed after the first step. Any group changes must be performed before the submission of the first step. However, we do not recommend to change groups once you start the project since moving codes might lead to plagiarism.
- Equal distribution of the work among the group members is essential. All members of the group should submit all the codes for both client and server. All members should be present during the demos. In case of any dispute within the group, please do not allow the problematic group members to submit the code, so that he/she will not be graded. However, if a group member submits the same code, then the other members automatically accepts his/her contribution. In such a case, the same group continues with the second step. In other words, submitting step 1 together and separating for the second step is not possible.
- If a particular student does not submit the first step of the project (due to being groupless or being excluded from a group due to low contribution/effort), he/she can join another group for the second step.
- Similarly, if a particular group member does not work enough in the second step, please do not let him/her submit.
- Your TA Simge Demir (simgedemir@sabanciuniv.edu) is responsible for keeping track of the groupings. She will send an announcement to the class about how the group information will be collected, how underpopulated groups will be merged and how people without groups will be grouped.
- You have a chance to form your own groups. However, if you cannot find enough people to form a group, then you have to accept to work with people that we assign. If you do not like to work people that you do not know, then form your own groups!
- Please notice that neither Simge and other TAs nor myself are responsible for dispute resolution among the group members.

Programming Rules

- Preferred languages are C#, Python and Java, but C# is recommended.
- Your application should have a graphical user interface (GUI). **It is not a console application!**
- In your application when a window is closed, **all threads related to this window should be terminated**.
- Client and server programs should be working when they are run on at least two separate computers. So please test your programs accordingly.
- Your code should be clearly commented. This affects up to 5% of your grade.
- Your program should be portable. It should not require any dependencies specific to your computer. We will download, compile and run it. If it does not run, it means that your program is not running. So, do test your program before submission.

Submission

- Submit your work to SUCourse+. Each step will be submitted and graded separately.
- All group members must submit the same code! Not submitting means no contribution, so non-submitters will not be graded.
- Delete the content of debug folders in your project directory before submission.
- Create a folder named **Server** and put your server related codes here.

- Create a folder named **Client** and put your client related codes here.
- Create a folder named **XXXX_Lastname_OtherNames_StepY**, where XXXX is your SUNet ID and Y is the project step (1 or 2) (e.g. arslanhanbegum_Arslanhan_Begum_Step1). Put your Server and Client folders into this folder.
    - Compress your XXXX_Lastname_OtherNames_StepY folder **using ZIP or RAR**.
- You will be invited for a demonstration of your work. Date and other details about the demo will be announced later.
- 24 hours late submission is possible with 10 points penalty (out of 100).

We encourage the use of our WhatsApp group for the general questions related to the project. For personal questions and support, you can send email to course TAs.

Good luck!

Simge Demir, simgedemir@sabanciuniv.edu
Begüm Arslanhan, arslanhanbegum@sabanciuniv.edu
Emre Ekmekçioğlu, eekmekcioglu@sabanciuniv.edu
Albert Levi