CS408 – Computer Networks – Spring 2023

Term Project : Networked Multiplayer Tic-Tac-Toe Game

This project is made of two steps; each has different deadlines as specified below.

Project Step 1 Deadline: May 5th, 2023, 23:59

Project Step 1 Demo: to be announced

Project Step 2 Deadline: May 22, 2023, 23:59

Project Step 2 Demo: to be announced

Introduction

In this assignment, you will work in a group to design and develop a networked multiplayer Tic-Tac-Toe game. This assignment is divided into two stages: a first version and a final version. In the first version, you will focus on implementing the basic gameplay mechanics and networking code. In the final version, you will add additional features such as adding more users, error handling and game statistics.

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 Tic-Tac-Toe 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 the game and play the game properly. Also, the server should keep the scores during the game. Moreover, it broadcasts the overall score table as well.

The server listens on a predefined port and accepts incoming client connections. Max 4 clients connect to the server at the same time. Each client knows the IP address and the listening port of the server (to be entered through an interface). 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: May 5th, 2023, 23:59):

In this step, you are going to develop a server module that can accept 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. The game is to be played as explained below.

Game Rules

- The game is played on a 3x3 grid.
- Each cells in the grid is numbered (see the example below)
- Players take turns placing their symbol (X or O) on an empty cell on the grid.
- The server should pick who will start first by stating e.g. X's turn
- Player should enter the grid cell number to put his/her move
- Player cannot pick an already taken cell number. If he/she tries to enter that number, the system will warn with a message.
- The player who successfully places three of their symbols in a row (horizontally, vertically, or diagonally) wins the game.
- If all cells on the grid are filled and no player has won, the game ends in a draw.
- Two players will play the game.
- Each client must have a unique name. This name must be entered using the Client UI. This name is also sent to the server. The server identifies the clients using their names.

Sample Game board scenes :

Initially:	After the first move	After the second move	After the third move	After the third move	Finally
1 | 2 | 3 4 | 5 | 6 7 | 8 | 9 X's turn X: 5	1 | 2 | 3 4 | **X** | 6 7 | 8 | 9 O's turn O: 1	**O** | 2 | 3 4 | **X** | 6 7 | 8 | 9 X's turn X: 4	**O** | 2 | 3 **X** | **X** | 6 7 | 8 | 9 O's turn O: 2	**O** | **O** | 3 **X** | **X** | 6 7 | 8 | 9 X's turn X: 6	**O** | **O** | 3 **X** | **X** | **X** 7 | 8 | 9 Game over. X wins!

Networking

- The game will be palyed with network communication.
- The server will be responsible for creating game room and handling incoming player requests.
- For step-1 game room can have two players.
- Players will send messages to the server to request to join the game room, make a move, or leave the game.
- The server will broadcast the current state of the game board to all players in the game room after each player's turn.
- There is only one server running on the system.
- For the client, the server IP address and the port number must not be hardcoded and must be entered via the Client UI
- The server will start listening on the specified port. It must handle multiple clients simultaneously

- All activities of the server should be reported using a rich text box on the Server UI including names of connected clients as well as all the 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.

Project Step 2 (Deadline: May 22, 2023, 23:59):

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. In the final version of the game, you will build upon the first version by adding the following features:

- The game can accommodate up to four players at a time. If a fifth player tries to join, they will be refused.
- The first arrived two players will play the game while the other players will wait (they watch the game). If a player leaves the game, the next waiting player will take that position and continue the game.
- The server will start a new game after the previous game is over.
- The players in the game can disconnect at their will and this should not cause your program to crash. If the player reconnects, it cannot join the ongoing game.
- Error handling: The game should handle errors gracefully and provide informative error messages to the players.
- Game statistics: The game should keep track of the number of games played, number of wins/losses/draws for each player, and display these statistics to the players.
- As in the step-1, All activities of the server should be reported using a text box on the Server UI including names of connected clients as well as all the 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.

Group Work

- You can work in groups of four - 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 Müge Kuşkon (mugekuskon@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 TAs or myself are responsible for dispute resolution among the group members.

Programming Rules

- Preferred languages are C#, Python and Java, but C# is recommended.
- 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 mail 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).

For personal questions and support, you can contact to course TAs.

Good luck!

Müge Kuşkon, mugekuskon@sabanciuniv.edu

Begüm Arslanhan, arslanhanbegum@sabanciuniv.edu

Emre Ekmekçioğlu, eekmekcioglu@sabanciuniv.edu

Kürşat Çağıltay, kursat.cagiltay@sabanciuniv.edu