HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

——————— * ———————

PROJECT REPORT

NETWORK PROGRAMMING:

Ultimate Tic-Tac-Toe

GROUP 15

VU MINH DUNG – 20205179

dung.vm205179@sis.hust.edu.vn

NGUYEN VIET HUNG – 20205157

hung.nv205157@sis.hust.edu.vn

Lecturer: PhD. Tran Nguyen Ngoc

Department: Computer Science

Hà Nội, tháng 1 năm 2024

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our dedicated group members for their unwavering support and collaboration efforts throughout this project. Their commitment and teamwork have been invaluable. We extend our heartfelt appreciation to PhD. Tran Nguyen Ngoc for providing guidance, expertise, and encouragement at every stage of our journey. Their insights have significantly enriched our learning experience and contribute to the success of this endeavour. Thank you to both our group and PhD. Tran Nguyen Ngoc for their outstanding contributions to the realization of this project.

Contents

CHAPTER 1. INTRODUCTION

1.1. Motivation

Tic-tac-toe, a classic and timeless game, is a two-player contest that embodies simplicity and strategy. Played on a 3x3 grid, participants take turns marking X and O to achieve a row, column, or diagonal of their symbol. The game's straightforward rules make it accessible to players of all ages, yet its strategic depth challenges even the most seasoned enthusiasts. With only nine squares to navigate, each move carries strategic significance, requiring players to anticipate their opponent's moves while planning their own. Despite its humble origins, tic-tac-toe remains a beloved pastime, offering a quick and engaging way to test one's tactical acumen in a friendly competition.

To make the game more complicated, we upgraded a tic-tac-toe game to a high level, called Ultimate Tic-tac-toe. Ultimate Tic-Tac-Toe represents a captivating evolution of the classic game, introducing a heightened level of complexity and strategy. Played on a 9x9 grid, it consists of nine individual 3x3 boards. The unique aspect lies in the interconnection between these boards, where the outcome of each small board determines the permissible moves for the opponent in the corresponding larger board. Winning a small board grants control over a specific section of the larger grid. This dynamic structure transforms the game into a multi-layered challenge, demanding not only tactical moves within individual boards but also a broader strategic approach. Ultimate Tic-Tac-Toe provides a refreshing twist on the traditional game, offering players a more intricate and engaging experience that tests their strategic acumen in a novel and exciting way.

1.2. Objective and scope

The system will primarily function as a socket-based application, encompassing all the essential features as specified by PhD. Tran Nguyen Ngoc

The primary objective of this game is to make a competitive environment for players to enhance their thinking. To win this game, players must know how to control specific sub-boards, block the opponent, and deceive the opponent. So that, the player can enhance the ability to build long-term strategies, and execute strategies moves.

1.3. Game rules

1.3.1. Move

The first player can place their designated mark at any cell on board. Whichever square that the first player places in a small tic-tac-toe game, the game will determine which next small tic-tac-toe game the next player gets to place their shape at. They are confined to that small tic-tac-toe game **ONLY.** If the next board is full or having winner, the player can place their shape at any available spot in the larger space.

1.3.2. Winning Conditions

A player can win the whole game if they have three smaller winning game in a row horizontally, vertically, or diagonally. If all board are full, then the winner is the one who have more winning game.

CHAPTER 2. METHODOLOGY AND REQUIREMENTS

2.1. Methodology

2.1.1. Core technology

C++ is an ideal choice for developing socket applications because of its high performance, memory control, multitasking and multithreading, robust support libraries, and cross-platform capabilities.

About the high performance, C++ is a compiled, statically typed language that is close to the machine. This means that C++ code is compiled directly to machine code, allowing applications to run quickly and efficiently in term of resources. And because it is a multi-player online game so that there might have several players accessing to this game at time. So that the supporting in multitasking and multithreading of C++ is so helpful. It enables the handling of multiple connections simultaneously without compromising the application’s performance

Next is about memory control. C++ provides unique memory control capabilities, allowing programmers to manage and release memory accurately. This helps avoid memory leaks and enhances the application's performance.

2.1.2. GUI technology

Qt is a powerful C++ framework widely utilized for developing cross-platform applications, and it offers several advantages when it comes to creating socket-based games like Ultimate Tic Tac Toe. Qt enables developers to write code once and deploy it on multiple platforms seamlessly. The cross-platform capability is crucial for ensuring that the Ultimate Tic Tac Toe game can be enjoyed by users on various operating systems without significant modifications.

2.1.3. Socket communication

The socket.h in Linux is a fundamental component of the Berkeley Sockets API, providing a set of functions and data structures for network programming. It enables the creation, configuration, and management of sockets, which are essential for establishing communication channels between processes over a network. Developers use functions defined in socket.h to create, bind, connect, send, and receive data through sockets. This header file plays a crucial role in implementing various networking protocols, allowing applications to communicate locally or over the internet. The functionality provided by socket.h is integral for building networked applications on Linux, facilitating the development of robust and efficient communication between processes or systems.

2.2. Requirements

2.2.1. Functional Overview

The application will include the following functions:

- Log in: players must log in before joining lobby.
- Log out: players can log out their account.
- Register: new players can sign up a new account to play game.
- Create room: players can create a new room.
- Join room: players can join an existing room.
- View Profile: players can view their information.
- Start game: the host can start the game.
- Surrender: players can surrender while the game is starting
- Online Player List: players can check the list of online players.
- Send challenge: players can send a challenge request to another player.
- Update online list: whenever players login or logout, the player list off another players can update.

2.2.2. Flow charts

Client:

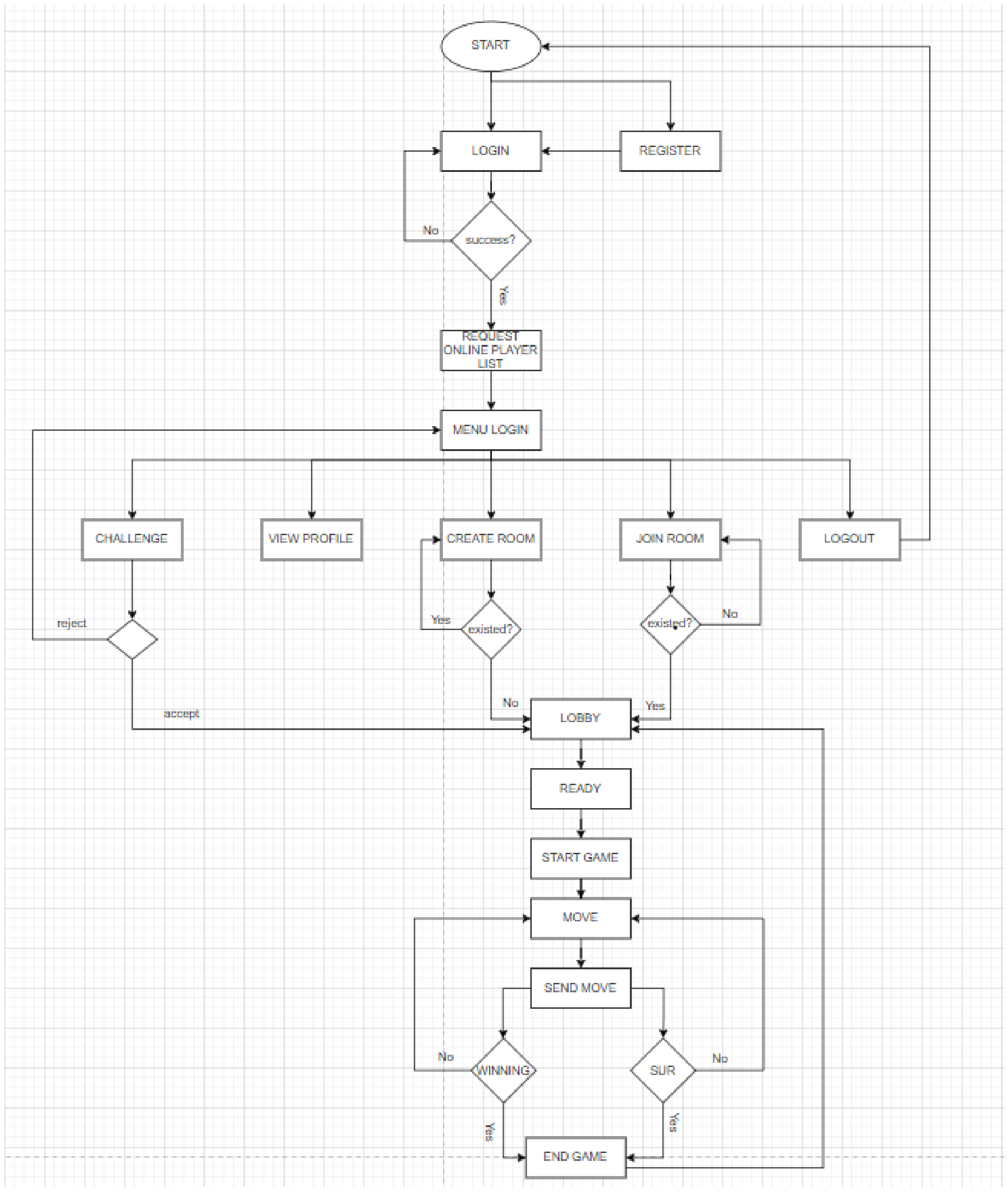


Figure 1: Client Flow Chart

From start, players must login to access menu login. If player doesn't have account, they can register by fill a form. After access menu login, players can send challenge request to another players, view their profile, create room, join room or logout. If players are in a Lobby, they are waiting for host starting the game. In game, players can make a move, client will send move to server to handle some logic game, and if that game is end, two players will go back to the lobby.

Server:

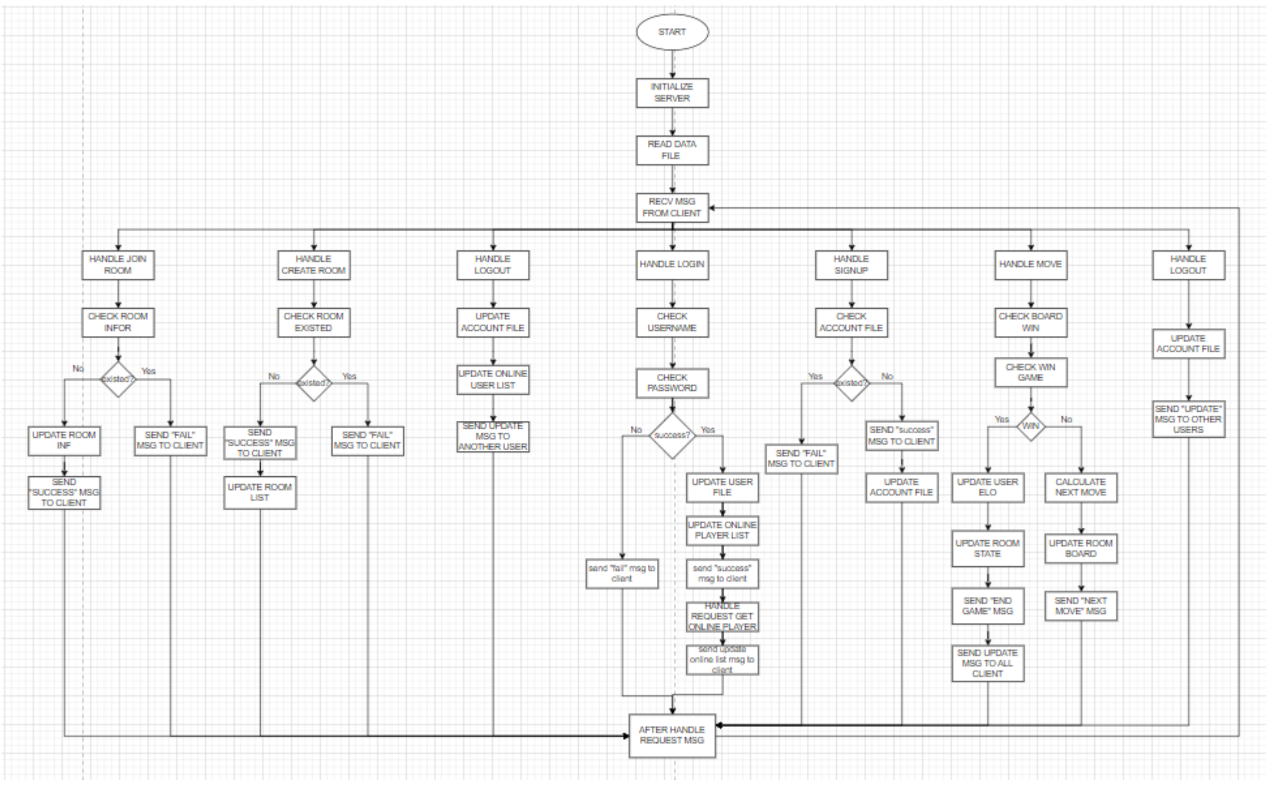


Figure 2: Infinite State of Server

The figure above shows the infinite state of Server such that anytime the server receives request, it will handle this request. After that, the server will back to the waiting state. This process will happen again and again.

CHAPTER 3. DESIGN AND IMPLEMENTATION

3.1. Architecture design

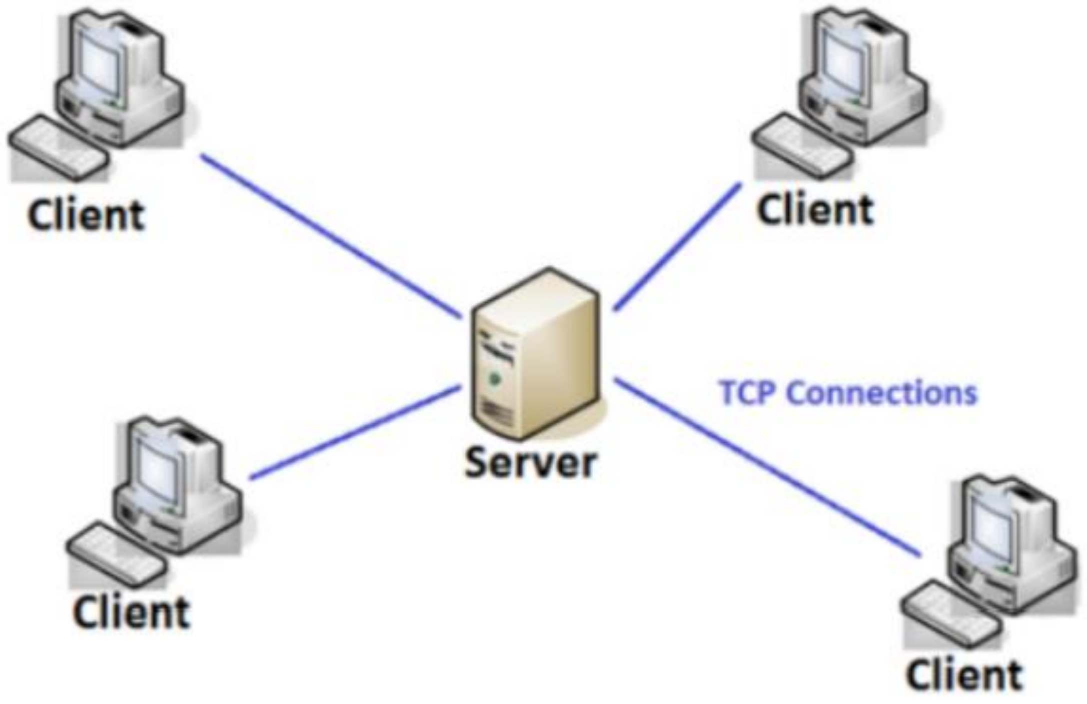


Figure 3: Client-Server

The system architecture is based on a Client-Server model, with communication facilitated exclusively through TCP sockets. The client-server architecture using TCP connections is a fundamental design pattern for building networked applications. In this architecture, the server hosts resources, or services, while clients initiate requests for these services. The communication between them is facilitated through the Transmission Control Protocol (TCP), providing a reliable, connection-oriented, and stream-based communication channel. The server listens for incoming connections on a specific port, and upon receiving a connection request from a client, a dedicated TCP socket is established for communication. The client, in turn, connects to the server's IP address and port. Through this TCP connection, data is exchanged in a structured and ordered manner. The server processes client requests, performs the necessary actions, and sends back the results. This architecture allows for scalable and modular systems, as multiple clients can connect to a single server concurrently. It is widely employed in various applications, such as web servers, databases, and online services, owing to its reliability and versatility in handling data communication over the network.

3.2. Protocol Design

In the client-server architecture utilizing TCP connections, the protocol design is paramount for facilitating smooth and standardized communication between clients and servers. The protocol defines a set of rules governing the format, sequence, and interpretation of messages exchanged over the TCP connection. A typical protocol involves a handshake to establish the connection and follows a request-response model. Each message, whether a client's request or a server's response, adheres to a specified format, often using serialization formats like JSON or XML. Error handling mechanisms are integrated into the protocol, detailing how errors are reported and managed between the client and server. Additionally, considerations for state management, concurrency, security features, and scalability are addressed within the protocol design. A well-structured protocol ensures interoperability, security, and flexibility, allowing for efficient communication and coordination in diverse client-server applications built on TCP connections.

3.2.1. Message format

Each message exchanged between the client and server has following format: the **Header** represents the type of message – considered as Enum type, the **Body** consists of data – considered as JSON type.

No	Request from Client to Server	
	Header (Type)	**Body (Data)**
1	LOGIN	username
		password
2	LOGOUT	username
3	REGISTER	username
		password
		ingame
4	GETONLINEUSER	
5	CREATE ROOM	room_name
6	JOIN ROOM	room_name
7	READY	room_name
		username
8	UNREADY	room_name
		username
9	START	room_name
10	MOVE	room_name
		current_cell
		current_board
		player_username
11	SURRENDER	room_name
		player_surrender_username
		another_player_username

Figure 4: Request from Client to Server Message

No	Response from Server to Client		
	Header (Type)	**Body (Data)**	
1	LOGIN	message	
		user	username
			status
			ingame
			wins
			loses
			elo
			win rate
2	REGISTER	message	
3	UPDATEONLINEUSER	message	
		user	username
			status
			ingame
			loses
			win
			win rate
4	CREATE ROOM	message	
		room_name	
		player_X_username	
5	JOIN ROOM	message	
		room_name	
		player_O_username	
6	READY	room_name	
		ready_player_username	
7	UNREADY	room_name	
		unready_player_username	
8	START	room_name	
9	NEXTTURN	room_name	
		next_board	
		previous_cell	
		previous_board	
10	SURRENDER	room_name	
		player_surrender_username	
		another_player_username	
11	WINBOARD	next_board = -1	
		room_name	
		current_board	
		current_cell	
12	WINGAME	room_name	
		winner	
		loser	

Figure 5: Response from Server to Client

3.2.2. Server implementation

The server is implemented using a multithreaded approach. Upon a new client connection, the server creates a new thread for that client. Each thread processes only one client’s requests, handles game actions, updates user scores, and manages PvP challenges.

3.2.3. Client implementation

The client initiates a connection to the server using TCP/IP. It sends requests to the server based on user interactions such as login, register, create room, join room… The client also receives responses from the server.

CHAPTER 4. RESULT AND CONCLUSION

4.1. Result

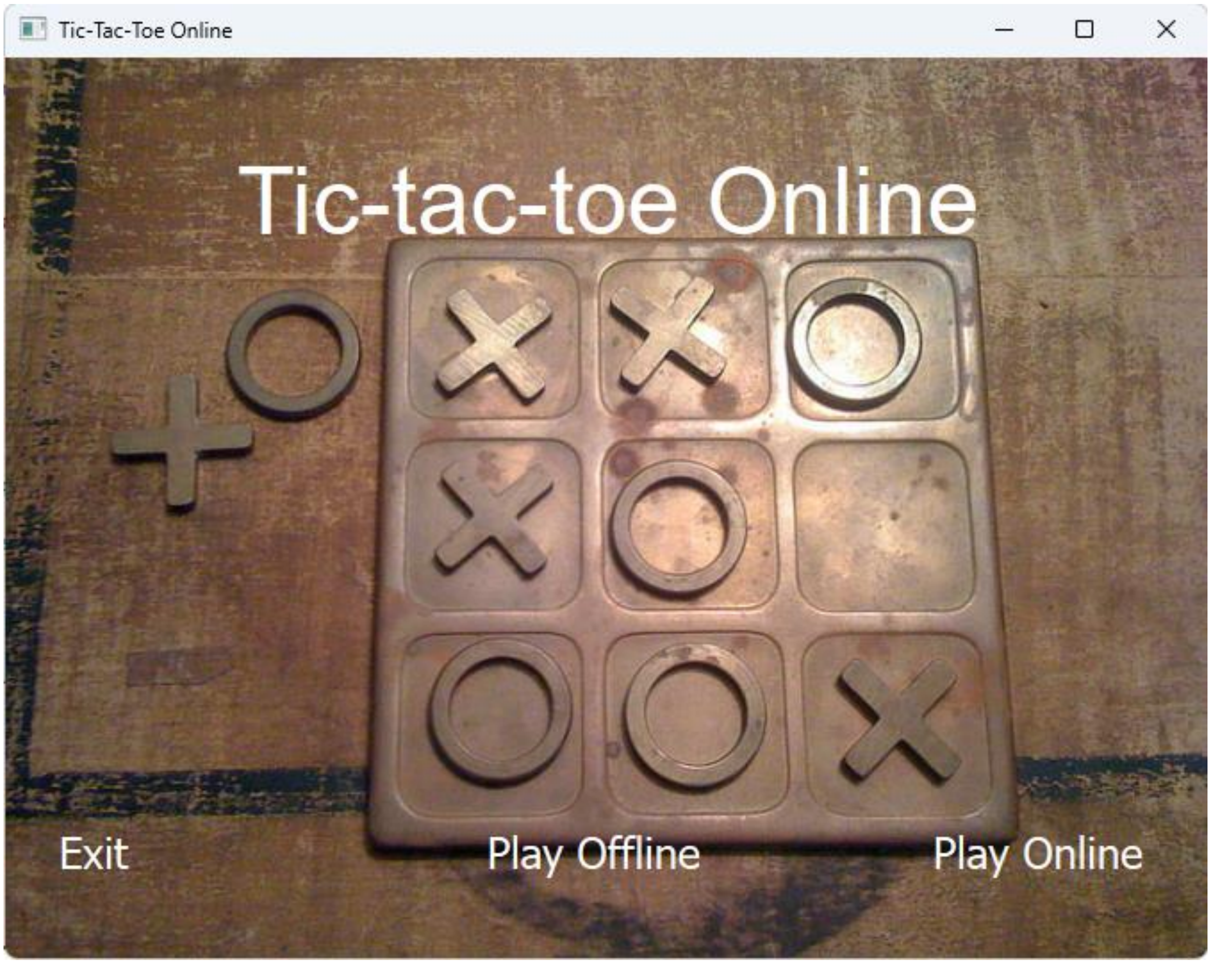


Figure 6: Stating Screen of the game.

In starting screen, the player clicks to button “play online”. After that, the client will be connecting to the server. And the player must fill in the login form to go to the main menu. If the new users do not have an account, they can click to “sign up” create a new account.

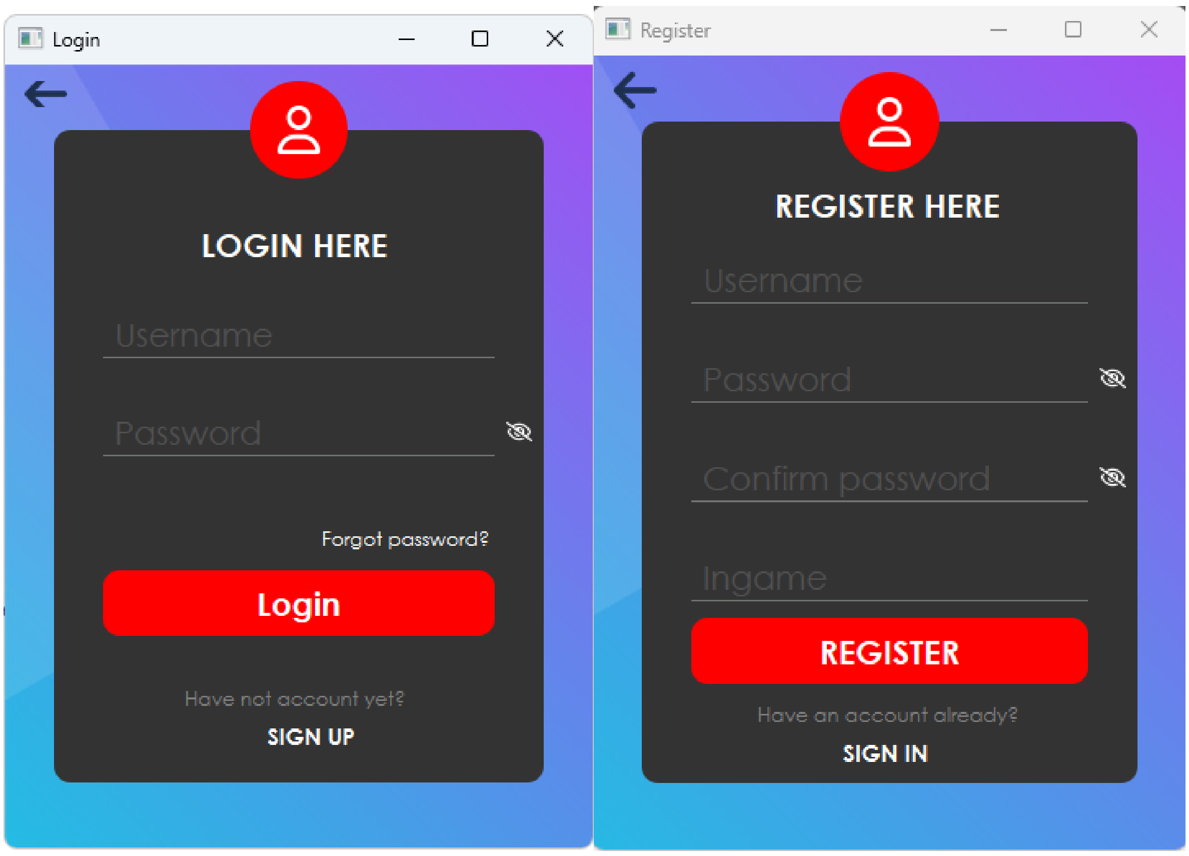


Figure 7: Login and Register form

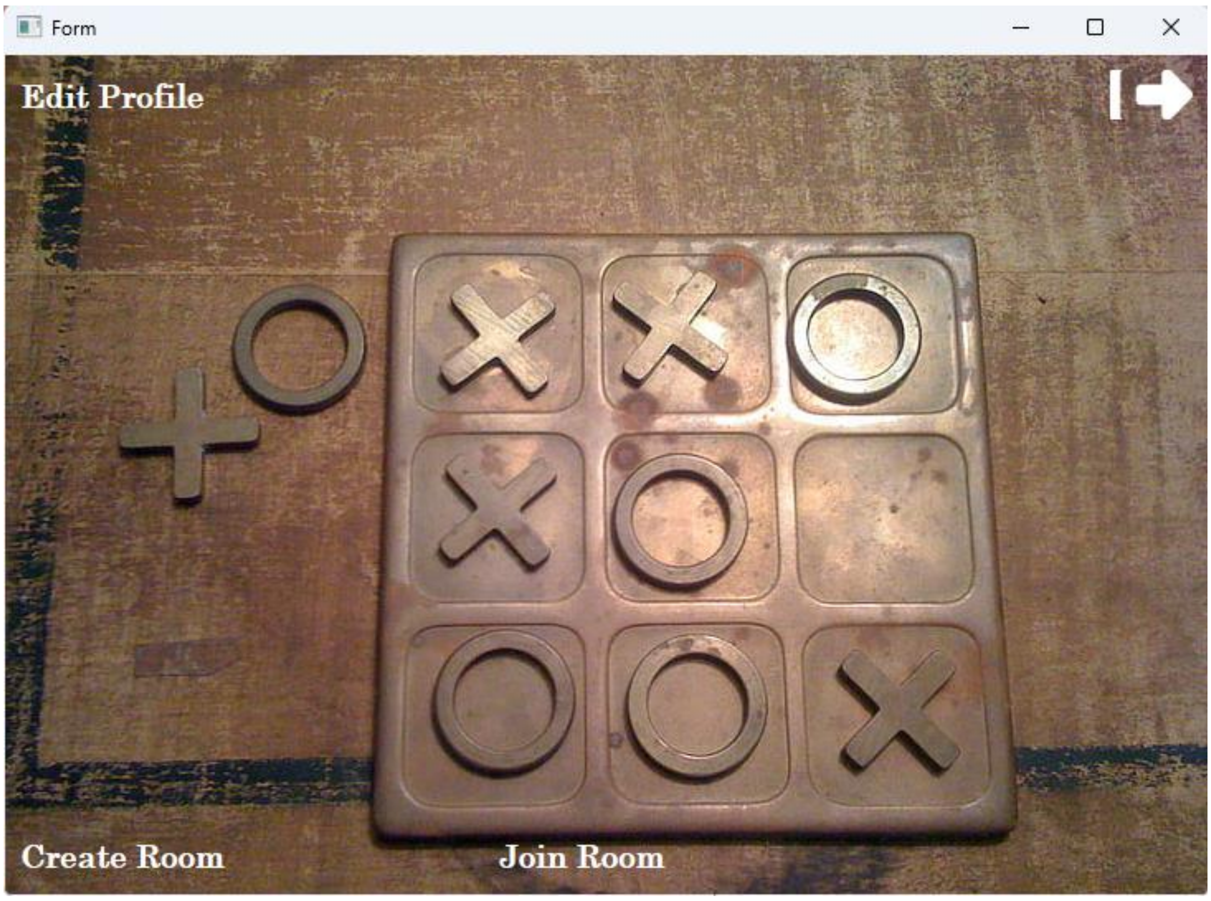


Figure 8: Main menu after login successfully

When the player wants to create a new room, click to "create room" button and insert room name. After that, the client will send this room name to the server. If this room name is existed, then the screen will display error and the player must change the room name

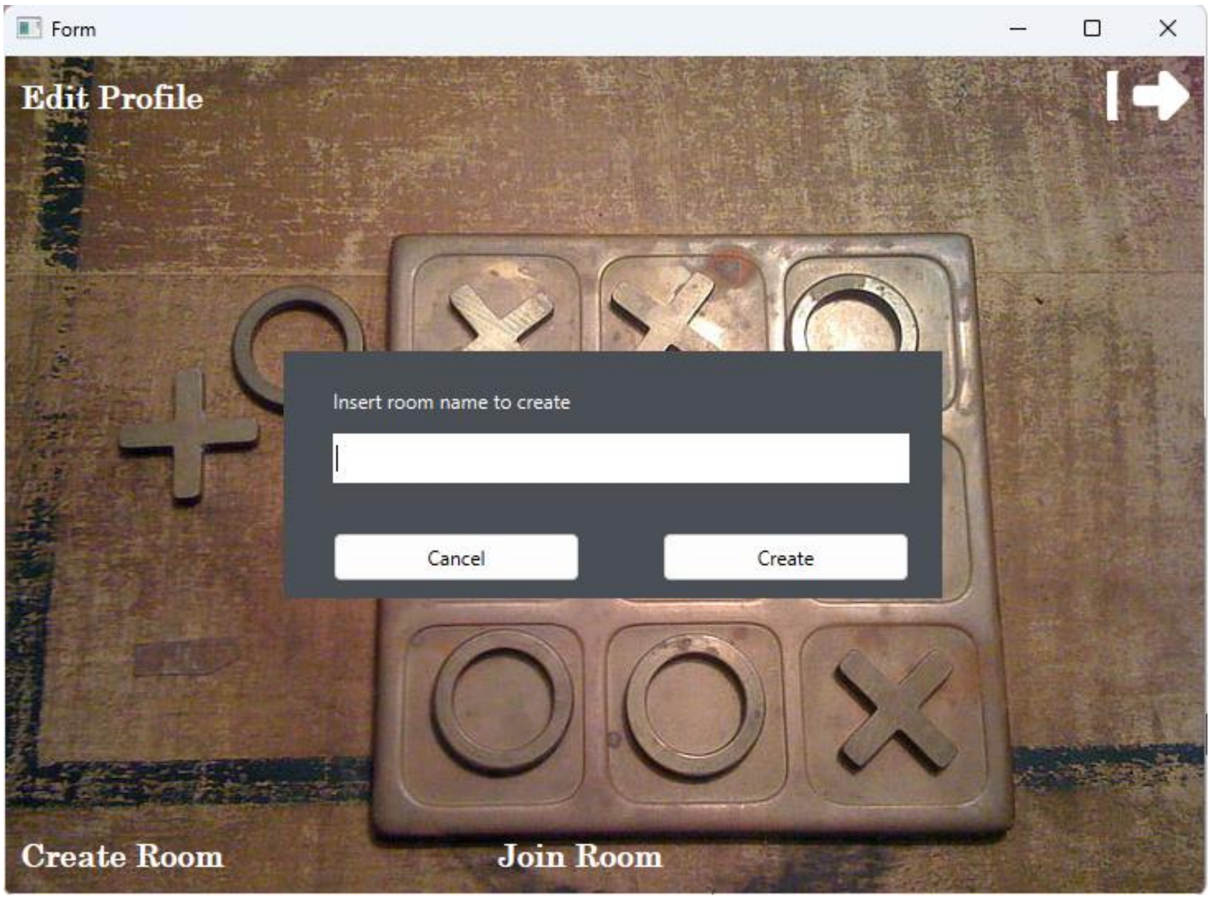


Figure 9: Create room

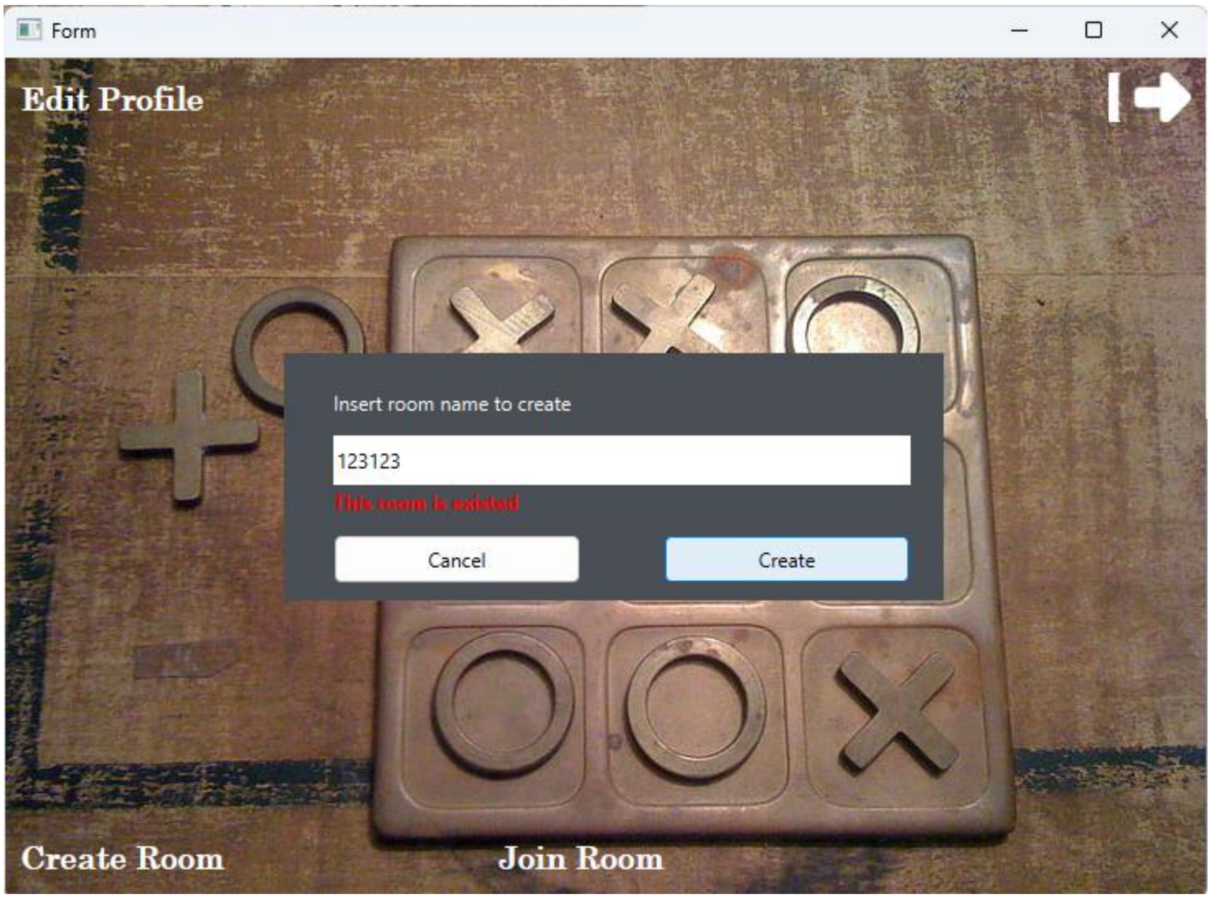


Figure 10: room existed

If the play create room successfully, then the lobby screen will display

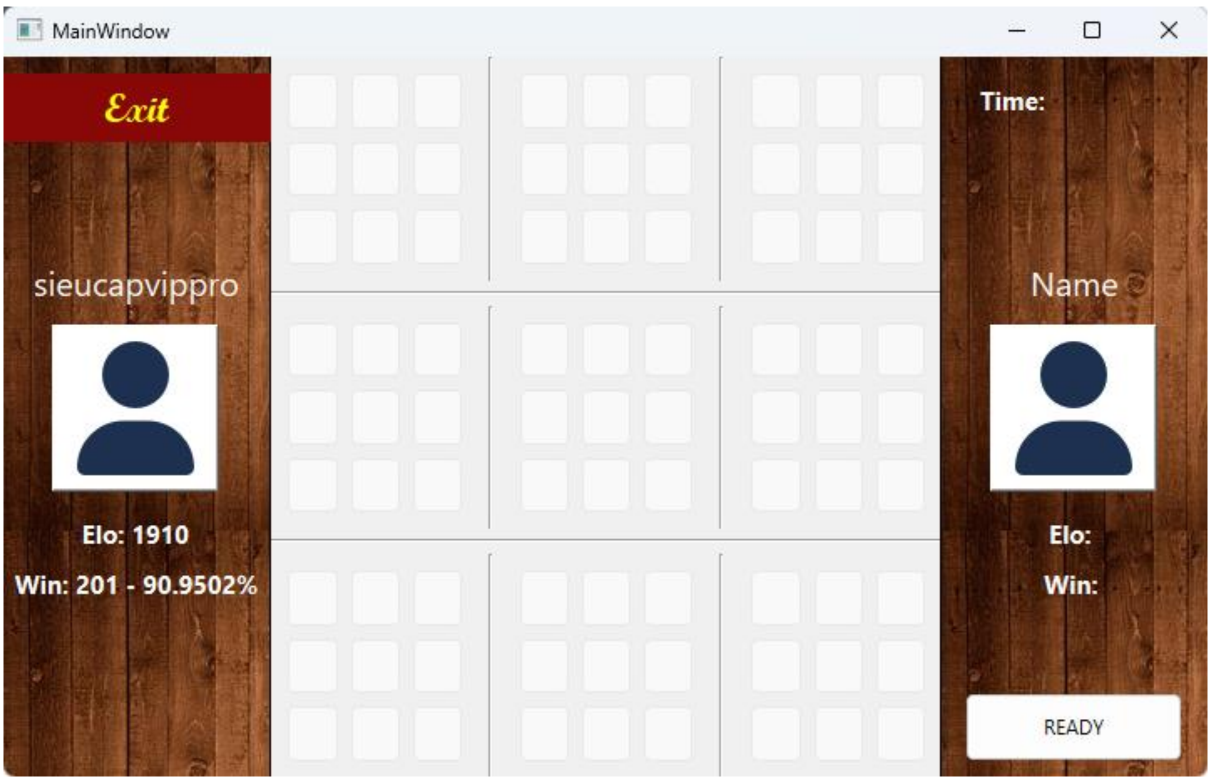


Figure 11: Lobby Screen

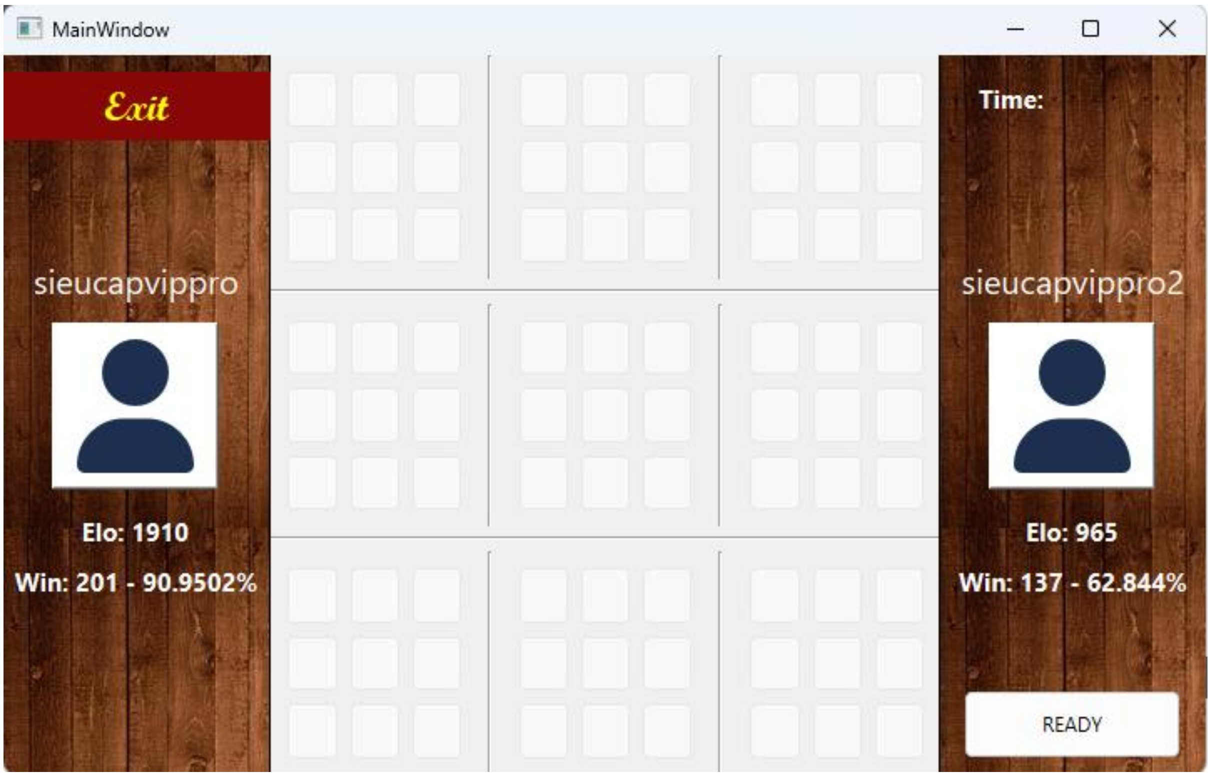


Figure 12: Update UI when player 2 joins

The host only can start the game when the other player is ready. Considering the next move sent from server, the button of invalid move will be disabled by the UI so that the player can not make an invalid move.

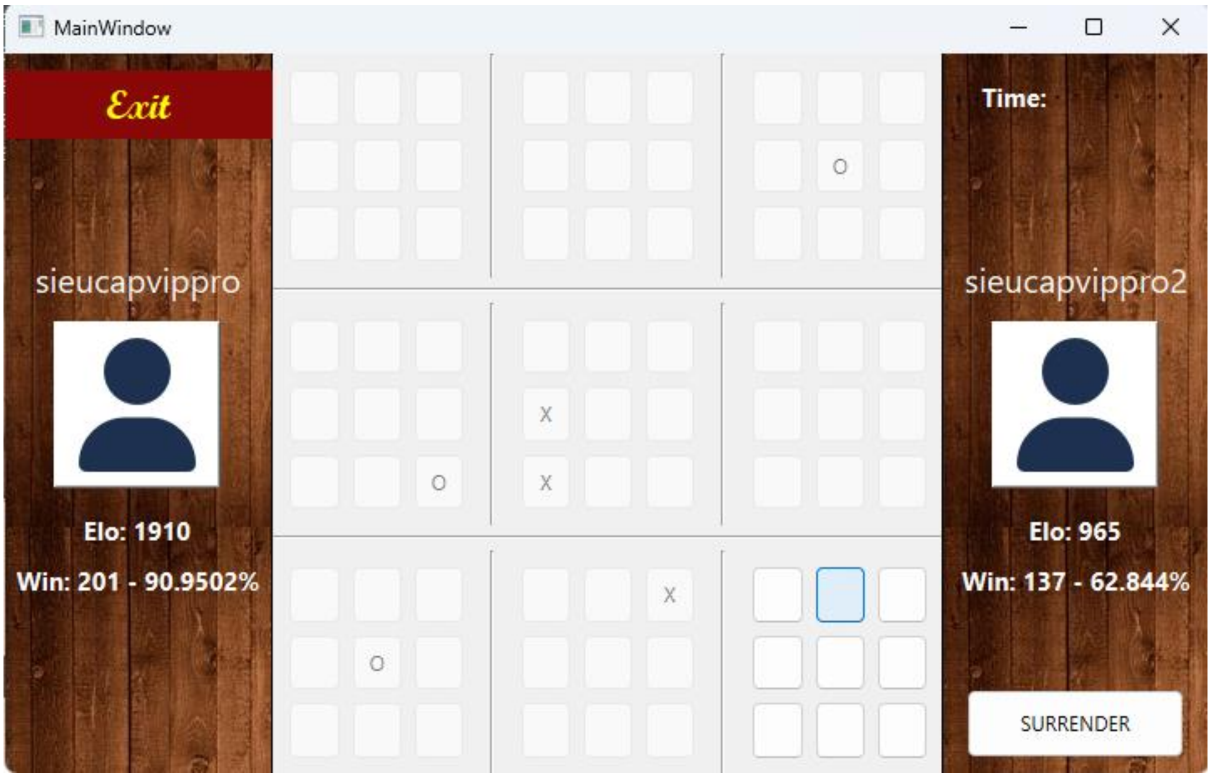


Figure 13: While the game is starting

When the previous player win a board, the next player can place their mark at any available board.

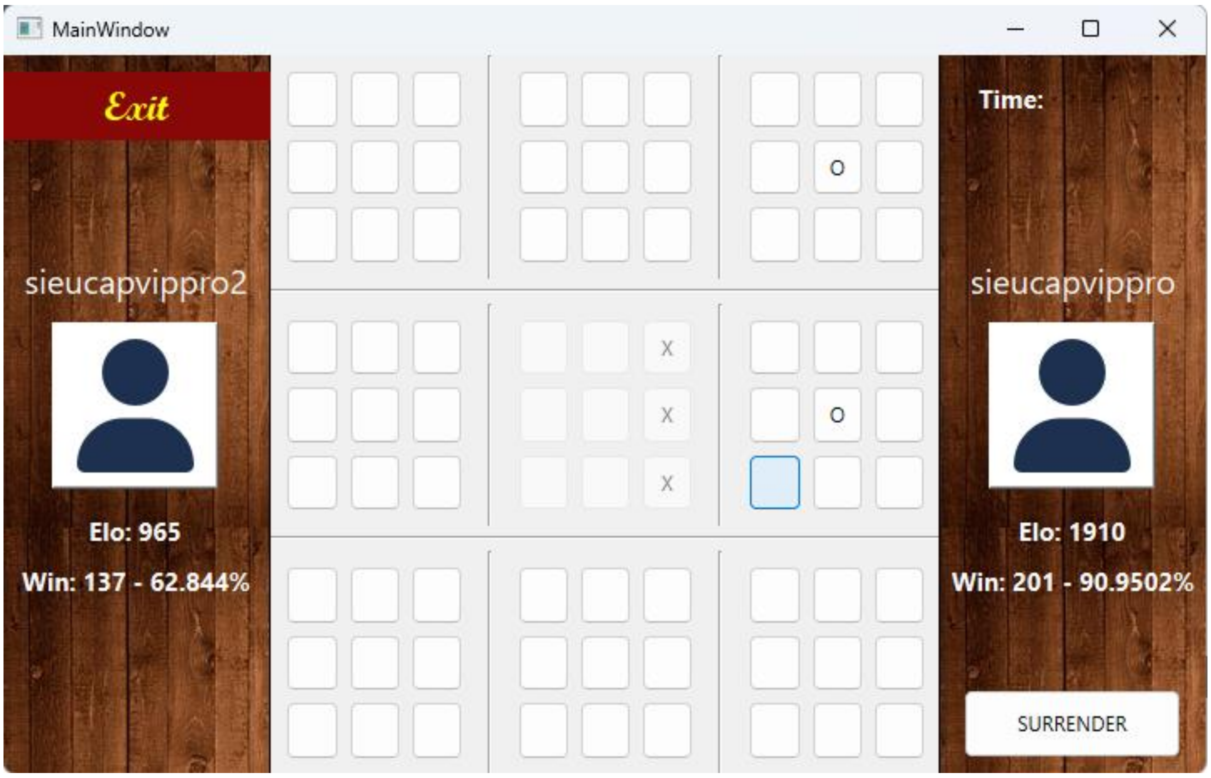


Figure 14: When previou player wins a board

If there is a winner, the screen will display a message

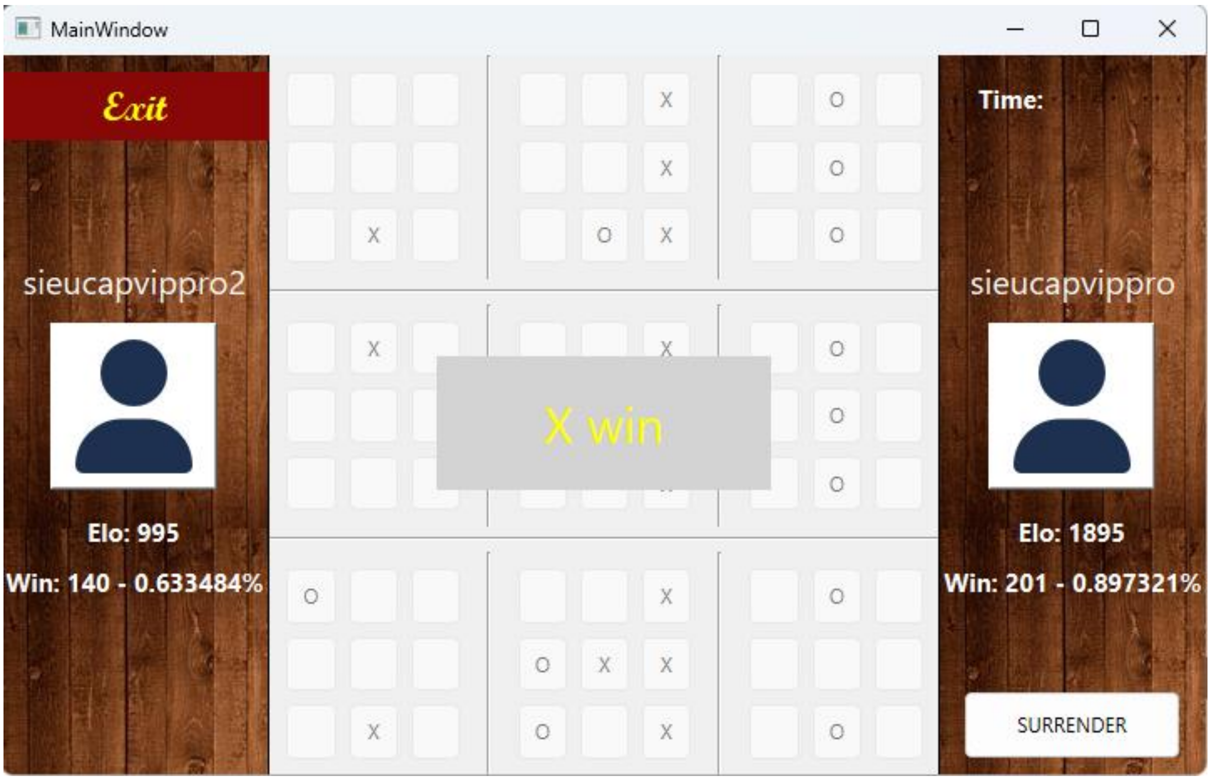
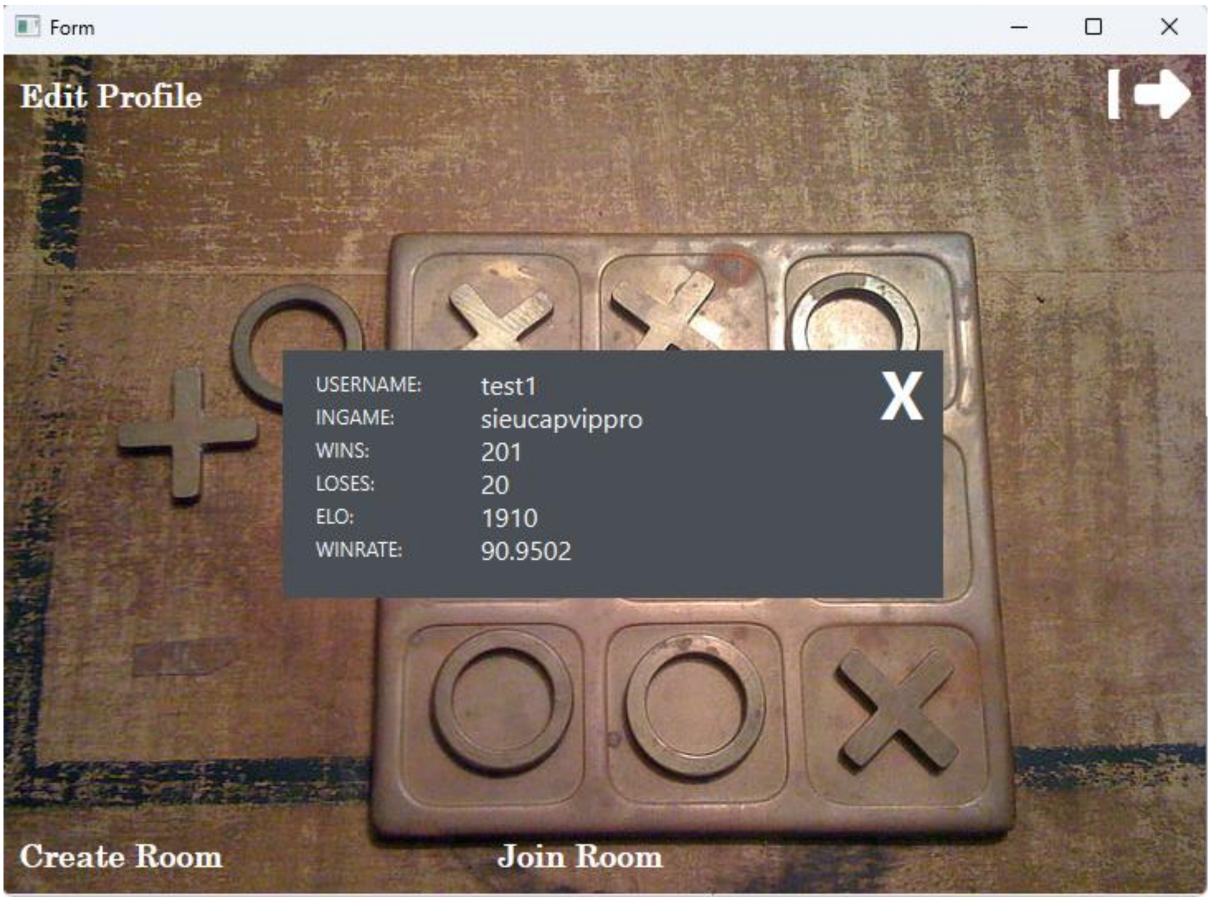


Figure 15: winner notification

The players also can view their profile:



USERNAME:	test1
INGAME:	sieucapvippro
WINS:	201
LOSES:	20
ELO:	1910
WINRATE:	90.9502

Figure 16: view profile

4.2. Conclusion

In conclusion, the development of the “Ultimate Tic-Tac-Toe” socket-based app during course Network Programming has been a rewarding and insightful experience. Throughout the project, we successfully implemented core technologies, such as C++ for socket communication and Qt for interface design, which allowed us to create a smooth and user-friendly application.

In the future, we want to improve this app such as: update UI of online player list, send challenge to other players, change information such as ingame, password, etc...

We are confident that this app can serve as an entertaining and educational platform for users, and we look forward to further improvements and future enhancements to make it an even more enjoyable experience.