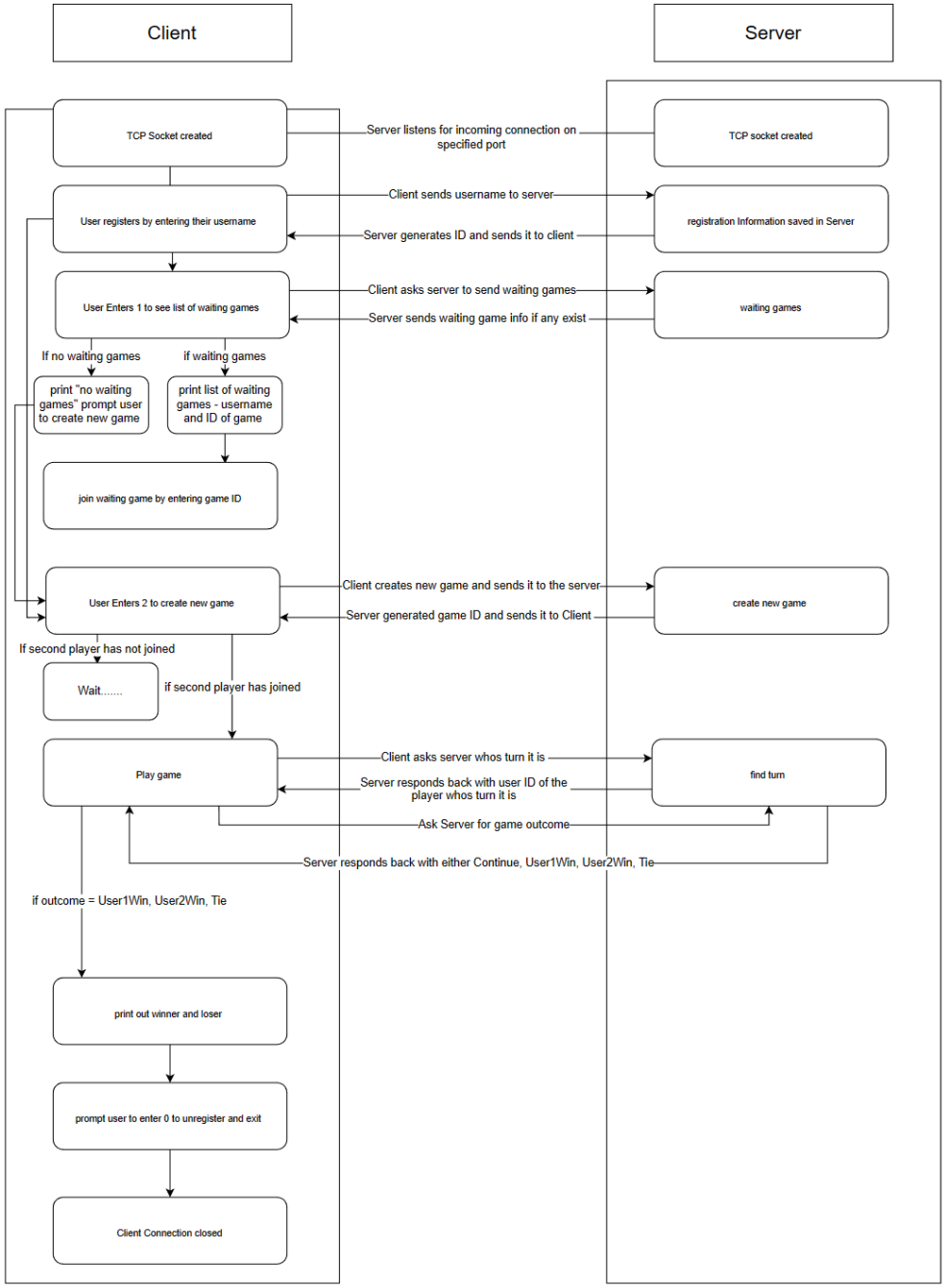
Team Sarah
Sarah Asad
Tic-Tac-Toe Final
Project Documentation

High Level Architecture of Tic-Tac-Toe



Main.cpp

This is where the Client and Server connect() functions are called. The server is powered on and waits for a connection and the Client is able to connect. There is also a diaplayHelp() method added In to help a user understand how to start the client and server if they are having difficulties starting the program. If the command line receives 3 arguments then it knows that it is the server, if the command line receives 4 arguments then it knows that it is the Client.

Server.h

- Contains all the method definitions for Server.cpp
- **Struct User** stores a user's username which is given upon registration as well as a user ID which is assigned to each user upon registration
- **int nextUserID** which is initially assigned to the value 100 and is incremented by 1 for each new user
- **int nextGameID** which is assigned to the initial value of 5000 and is incremented by 1 for each new game created.
- **Map Users** which stores key/value pair of {username, userID)
- **Map activeGames** which stores key/value pair of {gameID, gameBoard}
- **Map waitingGames** which stores key/value pair of {gameID, gameBoard}
- **Struct ClientContext** sores the server information, client socket, userID, and pthread

Server.cpp

The server is build using TCP socket and multithreading allowing multiple games to be played at once. The server socket connection is set up where it them listens for incoming connections. The server also stores all of the player information (player ID, username), game information (game ID, game board), which games only have one player and are waiting for a second player, which games are active and are currently being played, which players turn it is, and the outcome of the game.

Client.cpp

The client is built using TCP socket and connects to the server on the specified poet. The Client them promps the user to register by entering a username. The user then has the option to either get a list of waiting games by entering 1 and from there they can join a waiting game. Or the user can start a new game by entering 2. If a user joing a waiting game then the game instalty starts. If a user has created a new game then they need to wait until a second user joins for the game to start. From there, each player takes turns marking their corresponding symbol in their desired box. After each turn, the Client asks the server for the game outcome to see if it is either a player has won, if there is a tie, or if the game should continue. Each user continues to take turns until there is a win or a tie. After the game is over, the outcome of the game is printed out and the user if prompted to enter 0 to unregister and exit the game.

Socket.cpp

There is a lot of communication back and forth between the client and the server. Socket is meant to make this communication easier with predefined send/receives that can be easily called to send data like integers, strings, and the games between the two

CommandTypes.h

Contains an enum to see what user would like to do and includes:

- RegisterUser
- GetWaitingGames
- JoinExistingGame
- StartNewGame
- IsGameStarted
- IsMyTurn
- PlayTurn

SpcketException.h

There are a lot of places where error checking needs to be done when creating the socket, connecting to a host, binding, listening for incoming connections, and accepting connections. SocketException contains one method that takes a string message and int error makes error checking just a single line of code in the server and client

ByteBuffer.h

Contains a byte Buffer of size uint8_t that is utilized in Socket.cpp and allows the socket to easily read bytes of data sent back and forth.

Game.h

Contains a struct Game which includes:

- Int ID – the ID of the game
- Int user1 – the ID of user1
- String userName1 – the username of player 1
- Int user2 – the ID of user2
- String userName2 – the username of player 2
- Int turnofUser – specifying if it Is player1 or player2's turn
- Int board[3][3] – a two dimensional array used to print the 3x3 board
- Int winnerID – the ID of the winning player

Also contains a constructor to zero out the board and set the user names and ID of each player. winnerID is initially set to -1.

GameOutcomes.h

Contains an enum for all the possible game outcomes

- invalidMove
- Continue
- User1Win

- User2Win
- Tie
- YouWin
- youLose

How to Play Tic-tac-Toe

Go to this directory:

TicTacToe/bin/x64/Debug

Start the Server:

./TicTacToe.out Server <port number>

Ex: Server running on csslab22.uwb.edu

```
[sasad23@csslab22 Debug]$ ./TicTacToe.out Server 5100
Server ready and accepting connections
```

Start the Client:

./TicTacToe.out Server <host name server is running on> <port>

Ex: client running on csslac10.uwb.edu

```
[sasad23@csslab10 Debug]$ ./TicTacToe.out Client csslab22.uwb.edu 5100
Connected to server
Welcome to Tic-Tac-Toe
to begin, enter a username
Username:
```

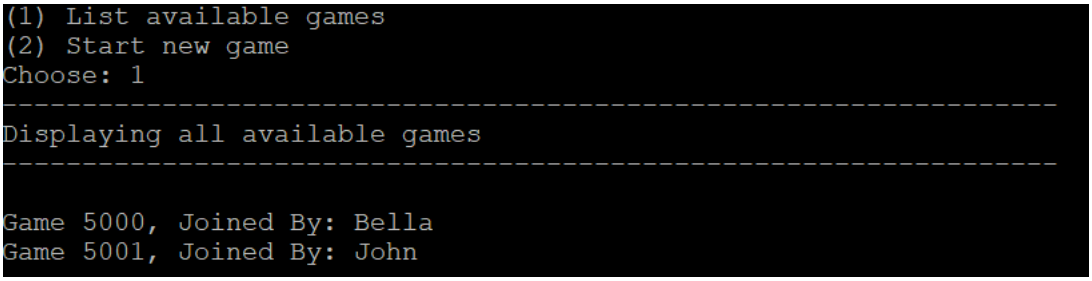
After the Server and Client(s) has been started, the Client prompts the user to register by entering a username. Upon registration, each user is assigned a unique userID to differentiate them from other users.

```
[sasad23@csslab10 Debug]$ ./TicTacToe.out Client csslab22.uwb.edu 5100
Connected to server
Welcome to Tic-Tac-Toe
to begin, enter a username
Username: Sarah
Your User Id is 100
```

The user is them prompted to pick from two options:

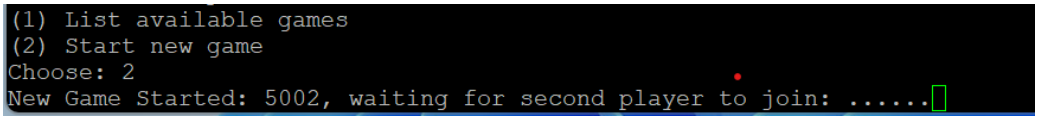
```
(1) List available games
(2) Start new game
```

If a user enters 1, then they are shown a list of all waiting games (games with only one player)



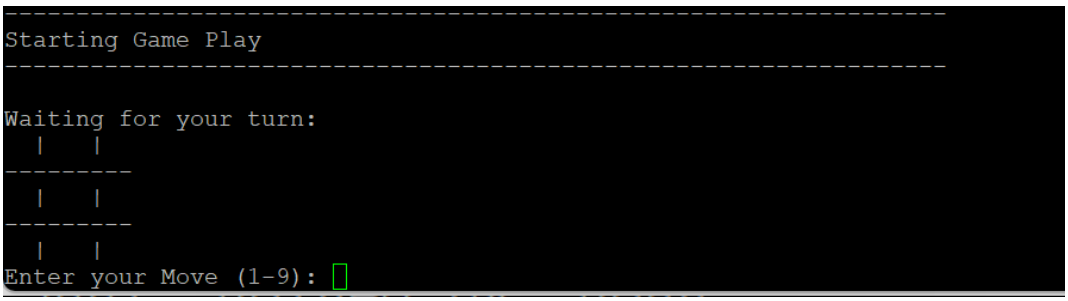
The user can then either choose to enter a game ID to join the waiting games. The game ID's in this case would be 5000 and 5001.

The user can also decide to Enter 2 and create their own game. The User is them prompted to wait for someone else to join the game



Once a second user joins, the game is started. The first person to join the game takes their turn first and is assigned the symbol X. the second player waits for the first player to take their turn and is assigned the symbol O.

Player1:



Player2:

The two users then go back and forth taking their turns. A player wins by getting their symbol in either a vertical, horizontal, or diagonal line.

Player 1 Won (slight bug – does not print out the last move if a user has won. This player has won by getting their symbol in the box 1, 4, and 7) and player 2 Lost

```
Enter your Move (1-9): 4
Waiting for your turn: .....
O |   | X
---------
O | X |
---------
  |   |
Enter your Move (1-9): 7
You Win!!
```

```
Waiting for your turn: ...................
X |   | O
---------
X |   |
---------
  |   |
Enter your Move (1-9): 5
Waiting for your turn: ....You Lose!!
```

User is then asked to Exit and unregister by entering 0

```
Waiting for your turn: ......You Lose!!
Enter 0 to exit and unregister:
0
[sasad23@csslab10 Debug]$
```