Make sure you read the whole document carefully and follow the guidelines in it.

**Preface:**

You will write a simple server that allows playing Tic-Tac-Toe game between two command-line clients.

**Functionality:**

The server will be started like: node server.js 5050
The server will listen on the specified port (5050) for clients to connect.

The client will be started like: node client.js 127.0.0.1 5050
The client will connect to the server at the specified IP (127.0.0.1) and port (5050). Upon connection the client will display a message and prompt, like: connected to 127.0.0.1 5050

When two clients have connected to the server, the game will begin. The server will send each client the message: Game started. You are the [first | second] player.

The Tic-Tac-Toe board is numbered like this:
1
2
3
4
5
6
7
8
9

The first player can then send a move like:
> 5
This move would place an 'X' at square number 5.

When the move is accepted by the server, it sends the current board position to both clients, like:
...
.x.
...

Let's say the second player make the move:
> 9

Both clients would then receive the new board position of:
...
.x.
..o

Either player can resign the game at any time, by sending 'r'. Even when waiting for the opponent to move. The player who resigned loses the game. When the game is over, the server sends both players a result message, like: Game won by [first | second] player. Or Game is tied.

The clients close the connection to the server after the game is over and exit. The clients can be restarted to connect to the server again to play another game. The server should continue running and handle the next game.

You should use bi-directional sockets for the connection between the client and server. Use the npm libraries socket.io, socket.io-client and read command. You can use additional libraries if you need.

**Requirements:**

- Write clear documentation on how it's designed and how to run the code.
- Write good in-code comments.
- Write good commit messages.
- An online demo is always welcome.
- Provide proper readme which includes steps to setup the code in any system, API documentation (Postman documentation link is preferred).
- Candidate needs to provide the github link and the candidate has to make his repository private.

**Tech stack:**

- Use Node.js and any framework.
- Use any DB. NoSQL DB is preferred.

**Bonus Points:**

- If you are familiar with ES6 standards then it will be a bonus point for you.
- If you follow the good practices of the Node js for coding standard/folder structure then it would be considered as a bonus point.

**What We Care About**

Feel free to use any open-source library as you see fit, but remember that we are evaluating your coding skills and problem solving skills.

Here's what you should aim for:

- Good use of current Node.js & API design best practices.
- Good testing approach.
- Extensible code.

**NOTE: Candidate should not be able for further rounds if we found plagiarism.**