Startup Gym - Interview Project
Fullstack Engineer

Our goal is to build a simple version of a Tic-Tac-Toe game: https://en.wikipedia.org/wiki/Tic-tac-toe. The project needs to have a separate backend server and frontend client, and we have been provided with a set of product level requirements that we must meet. The requirements are as follows:

1. Need an API to call to start a new game. The response should give me some kind of ID for me to use in other API calls to tell the backend what game I am referring to
2. Need an API to call to play a move in the game. The API should take as inputs the Game ID (from the first API), a player number (either 1 or 2), and the position of the move being played. The response should include a data structure with the representation of the full board so that the UI can update itself with the latest data on the server. The response should also include a flag indicating whether someone has won the game or not and who that winner is if so.
3. The API that handles moves being played should perform some basic error handling to ensure the move is valid, and that it is the right players turn (ie. a player cannot play two moves in a row, or place a piece on top of another player's piece)
4. A UI that shows a start screen that can present two options to the user: either start a new game, or join an existing one. Starting a new game should show the game ID that can be shared to the other player, joining an existing one should be allowed only if in that game there is max 1 move.
5. A UI showing a basic Tic-Tac-Toe board (don't worry about making it pretty) and exercising the APIs to make the moves. The UI should assume the second player is in front of the same PC if a second move is done in a row (switching between two players assumed to be playing together on the same computer locally) or allow the move from a different PC in case a second user joins using the game ID. After the second move the game can only continue either locally or from different clients.
6. A UI for when a game ends showing the winner and allowing to go back to the start screen.

Ideally, you should include test cases for your API using cURL commands or any other API testing tool you prefer. Your test cases should cover expected and unexpected scenarios of calls to your API.

It's recommended to use a Node.js/Express.js stack for the backend, and ReactJs for the client side; you can pick any datastore you would like to store the game state. You can decide to change/extend some of the requirements as long as the final goal is achieved; we will discuss your architecture/security design decisions so please be prepared to defend your choices.

Keep your solution simple, but it should also be readable. The programming of this assignment is not expected to be very complex so focus on a short and simple solution: cutting corners here and there while keeping the general goals of the specs is welcome. You can allocate how much time you think it's appropriate. There is no right and wrong with this task, but it is meant to serve as a foundation for our more in-depth technical discussion

where we will both discuss your current implementation as well as potential changes as a result of new requirements we may get that have not yet been shared in this project brief.

When you are done with your solution, please zip up your code and test cases and email it or share a git repo: giovanni@startupgym.it.

Looking forward to seeing what you come up with!