Your task is to implement a web-based game board for "connect 4" that allows two players to play against each other from different web browser instances. The application does NOT need to detect when someone has won. The application only needs to support one game at a time. The first two players to join the game will be the players of the game. The game should have a button to reset the board that a player can click at any point in time when they want to reset the game.

Some helpful resources for understanding the game:

- Official Connect 4 game rules: https://www.fgbradleys.com/rules/Connect%20Four.pdf
- Video about how to play connect 4: https://www.youtube.com/watch?v=utXzIFEVPjA
    - Feel free to google and find other resources on this If you don't understand the game

To save you time, we're providing some code. You're not required to use this code but we encourage you to do so to save time. We are providing code in JavaScript and TypeScript, you can use either language, you do not need both. The zip file we are providing contains the following folders:

- **mockup**: a static HTML site with HTML and CSS "mockup" of the application. Consider this the mockup from your teams' designer. It looks the way it's expected to look, but it was created by a designer not a developer.
- **frontend-js**: A JavaScript based react "hello world" application.
- **frontend-ts**: A TypeScript based react "hello world" application.
- **backend-js:** A JavaScript based express "hello world" application.
- **backend-ts:** A TypeScript based express "hello world" application.

Expected time to complete: 2-5 hours.

If you have not finished after 5 hours, please send us the partial implementation you were able to complete in that time. It's OK if you've never used one of these frameworks before, use this opportunity to learn as you work on the problem.

Deliverables:

- Source code for a frontend React based application that we can run locally, either in JavaScript or TypeScript.
- Source code for an Express based backend application that we can run locally, either in JavaScript or TypeScript.
- Instructions for building and running your application locally
- (Optional) Notes on any decisions you made that you feel may need explaining, if you have questions that are not answered in this document, make a reasonable decision considering the evaluation criteria and document the decision.
- Please deliver everything as 1 zip file or as a single publicly accessible GitHub repo
    - If using GitHub, please remove any reference to Bio-Rad from your repo
    - Please exclude your node_modules folder(s)

Requirements:

- Frontend must use React: https://www.npmjs.com/package/react
- Backend must use Express: https://www.npmjs.com/package/express
- Frontend and Backend must use Typescript or JavaScript (your choice)
- Solution must include a mechanism for running the frontend and backend locally
- Communication between backend and frontend must happen over HTTP (HTTP polling is OK)
- Game must support 2 players in two different browser instances playing the game against each other

To save time, skip these areas:

- Detecting when the game is over – the algorithm to know when the game is over is complex, skip this, assume that the two users will know when the game is over on their own and reset the game on their own using the button.
- Single player mode – We are only expecting a 2 player game, we do not expect you to implement the single player vs a Bot scenario.
- Any rules about who goes first - Just assign one of the players to go first randomly, or by who joined first, it's not important.
- Edge cases – don't worry about what happens if 3 players try to join, don't worry about any race conditions, don't worry about malicious users.
- Security – we'll talk more about ideas for securing the application in person
- UI Improvements – we are NOT expecting you to make the UI any better looking than the static HTML page we sent, it is much more important to have a working solution and well-organized readable code.
- Data persistence – if the server goes down, the game in progress will be lost, that's OK, just use in-memory storage to store the game in progress. Similarly reloading the browser mid game can be ignored. We do not want you to have to configure any databases or other data storage systems.
- Scalability – this game only needs to run locally and only needs to support two users and one game at a time
- Infrastructure – the game only needs to run on local development machines, no need to figure out how to deploy this to a cloud server

Evaluation Criteria:

We will be evaluating your solution based on the following criteria:

- Competency at using Typescript or JavaScript
- Competency at using React and Express
- The design of the HTTP interface between your backend and frontend
- Code organization, readability, and maintainability
- Satisfaction of the requirements above