Individual work: Battleship in REACT.JS

The goal of this individual work is to program a functioning prototype of the Battleship game using the REACT.JS technology. The Battleship game is a game in which two players set up a board with ships of different sizes, and then try to guess the position of the opponent's ships. In this individual work, you have to build a prototype that allows a player to play against a board set by the computer.

At first, you have to build a page that allows the player to define the size of the board (the board is typically square, but it is okay to make it rectangular), with a button to start the game. You can use a text field to introduce the desired dimension (or the number of rows and columns if you make the board rectangular). The minimum dimension will be 5x5 squares.

Upon clicking the start button, the computer will generate a board with the desired dimensions (i.e., a matrix with all blank cells) and will set four ships: two 3x1 ship, a 4x1 ship, and a 5x1 ship. Ships should be placed randomly, following the traditional rules of the game: they must occupy consecutive cells in the board and should not touch each other. Cells will remain blank.

Then, the game starts, and the player can try to guess where the ships are located by clicking on the cells of the matrix. If a cell contains a portion of a ship, then it will be colored in green. If it doesn't, it will be colored in blue. When a ship is fully uncovered, all of its cells will turn red.

If the player manages to discover all of the ships, then the game ends in victory. However, the player has a maximum amount possible attempts at failure, equal to half the size of the board. In other words, when a player misses for (boardSize/2) times, the game ends in defeat. As an example, if the board has 25 cells, when 13 blue cells become uncovered, the player is defeated. In both victory and defeat, the remaining uncovered cells become inactive, and a text line is shown below the board.

An example of the board's behavior can be found at the following image: