Battleship

JavaScript, React and State Management

Overview

For this project, you will create a single player-version of the popular board game Battleship. A (very impressive) online implementation can be found here or here and I'll describe more fully below. (Please keep in mind that my expectations for your project will be much simpler that what is linked there.) When users enter your app, they should see a welcome page that contains nothing but the title of the game as well as ways to see the "rules and play the game page. The goal of Battleship is to find and destroy your opponent's fleet before they can destroy yours!

For this assignment, you may work with one additional student. Feel free to post on the discussion board if you are looking for a partner. You are of course welcome to tackle this work by yourself.

Please note: you are welcome to come up with your own app idea, but you must contact the teaching staff at least a week beforehand to ensure that the code is sufficiently worthwhile and can reasonably meet the expectations of the rubric below.

Rubric

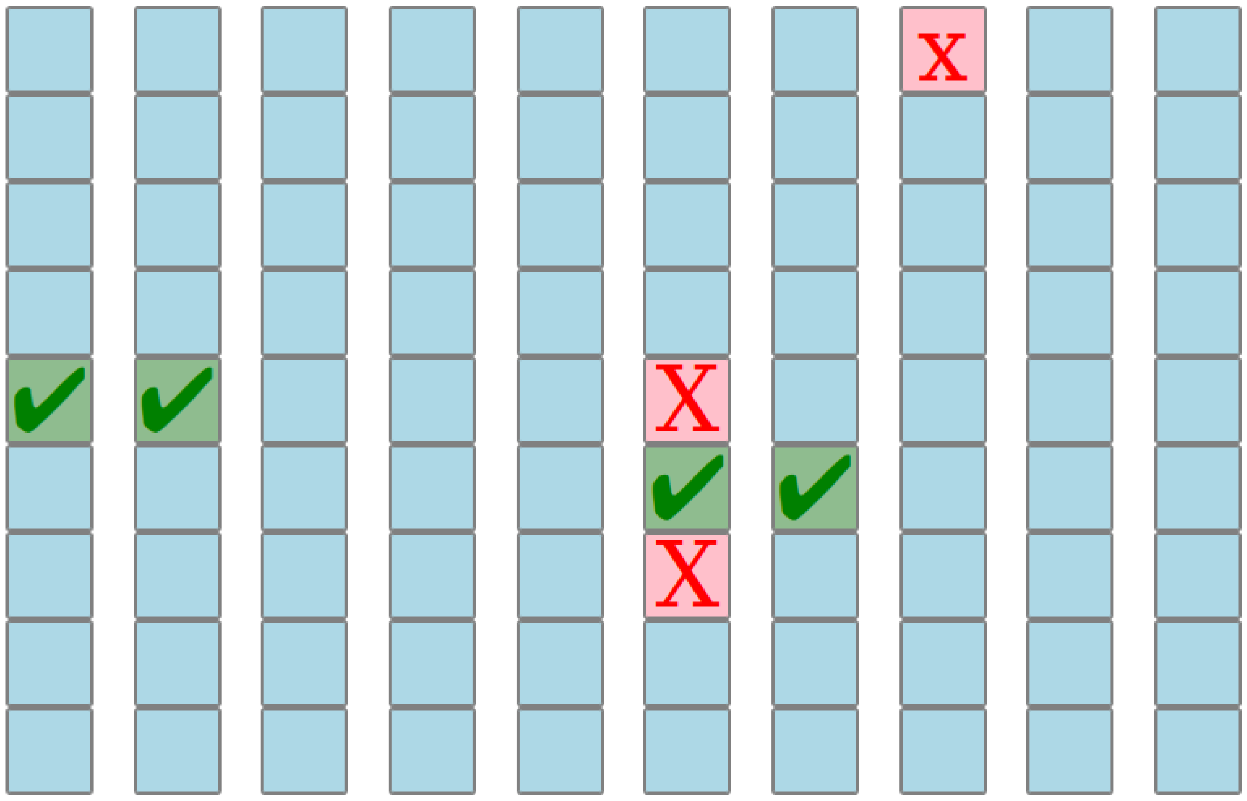
Core Functionality - 25pts
Working Github and Heroku Link - 5pts
Correct Views and Good Styling - 15pts
Well Written Code - 15pts
Stores, Actions and Reducers - 15pts
Demonstrating Proper React Principles - 15pts
Writeup - 10pts
Bonus Points - 10pts

Core Functionality

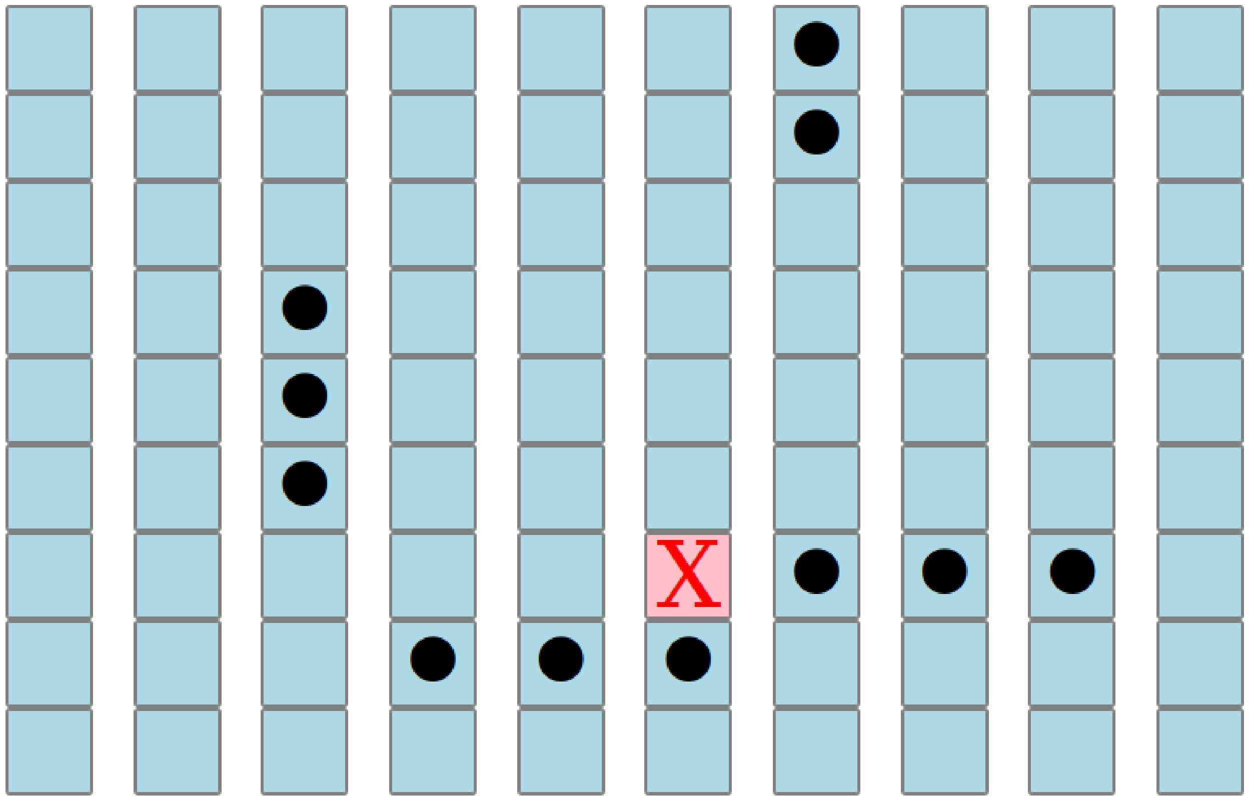
Battleship is a board game that pits 2 players against each other. For this project, you will create a very simple AI that prepares the play area and plays against you.

A game of Battleship is made up of two 10X10 boards, one that represents the player's board and one that represents the opponents. At the start of the game, 5 ships are randomly placed on each board (one 5X1 ship, one 4X1 ship, two 3X1 ships, and one 2X1 ship). Each ship should fit entirely on the board and they should not overlap any other ship on the board.

During the game, you and an AI will take turns (the player always goes first). On your turn, you will select a square on your opponent's board. On your opponent's turn, the AI will randomly select a square on your grid. If you or your opponent hit a ship, then mark that board with a color and symbol. If you or your opponent miss, then mark a spot on the board to remind the players where on the board they have attempted. The AI will not try to hit the same place more than once and the user should not be able to select the same spot more than once. Here is an example of what the screen will look like part way through a game:



An Example of the Enemy Board, the green checks represent misses and the red X's represent hits (you are unable to see the enemy ship from this view)



An Example of The Player's Board. The black dots represents ships, and the red X represents a ship that has been hit.

When one of the players has defeated the other (i.e., has eliminated all of their enemy ships), list at the top of the screen "Game over! {Player or AI} Won!".

At the top of the screen, there should be the following button that perform the following logic when selected:

- Reset: start a new game of Battleship.

Finally, when the user comes to your landing page, they should be able to select one of 2 options for this game. The difficulties are as follows:

- A normal game
- A free play game - only show the opponent board (and remember to hide the enemy ships.) The enemy AI does not take any turns here and their turn will be skipped.

Working Github and Server Link

For this assignment, I recommend you use Heroku to host your code, but you are welcome to use any web hosting service you are comfortable with. Please follow the instructions from the lectures or contact any of the teaching staff to get this set up if you need help. Please be sure to add the TA's as collaborators.

Correct Views and Good Styling

There are at least 3 views to implement for this assignment:

- A home page that allows users to select if they want a normal or free play game (see above)
- A page that allows the user to play the game
- A page with the rules for the game

You should decide what these views look like and how they behave (are they entirely different pages? Will you swap out components?). Feel free to intermingle these, but justify your decision in your writeup.

The tiles on your board should have the following states:

- Unselected - show a color for a state when the space has not ever been selected
- Selected, hit a ship - show a color and icon to show that a boat was hit here
- Selected, hit nothing - show a color and icon to show that someone selected this spot but nothing was hit
- Hover - show a variation of the above states of the user is hovering over a tile

Additionally, you should have a unique and consistent style across the different pages. While the style is relatively simple here, consider adding style to buttons, on hover styling over cells so that the cursor turns to a pointer, fonts, background images, etc. The screen should also present well on both mobile and desktop browsers: for instance, on mobile you should show the opponent and player board below each other; on desktop, show them next to each other. You should have some sort of navbar or navigation aid that allows users to go between these different views.
You are welcome to use any 3rd party styling libraries, such as Tailwind, React Bootstrap, Material UI, etc.

Finally, if these views are on different pages, consider sensical and good URL design.

Well Written Code

Now that we're writing logic, you must start considering the quality of the code you're writing. Functions should be simple, easy to read and avoid repetition. React components should be small, reusable and have a single purpose. We are not expecting you to use any more advanced JavaScript functionality, but you should be writing code that you would be happy to show to a potential employer.

State Management

For this assignment, consider how and where you are storing information. In web development, there are local and global states:

- Local state is the data stored in a single component (this might be what kind of card to display, etc.)
- Global state is all the data related to the entirety of the app (what cards are available, what cards are selected, etc.)

For global state, you are expected to choose one state management approach (useContext, Redux, or other if you speak with the teaching staff first). Make sure that there are at least 2 state update actions with your state management tool.

Demonstrating Proper React Principles

With the introduction of state, we learned about the unidirectional flow of information and how to separate the views from the data. In this assignment you must:

- Have at least 4 different React components
- One of those components must receive props from a parent component
- You must have nested components
- Some data must be passed from the child to the parent, but you should NOT do this with a function. Rather, use your state management tool

Bonus Points

Bonus points for this assignment are based on concepts that we don't (or very briefly) cover in class but will provide additional learning opportunities.

Bonus Points 1: Local Storage

You will notice that if you reset the browser mid-game, the entire game will reset. The goal here is to use local storage (window.localStorage) to store the state of the game after every action. You will still need to use React Context or Redux, so keep in mind the following concerns:

- You should check that localStorage has data when the app first opens
- You should update the localStorage data after each action
- localStorage should ONLY be accessed through your Redux or React Context code
- You should clear localStorage after the game is over (either through reset or a winner is decided)

Bonus Points 2: Click and Drag Setup

You will add one additional step during the start of the game. The AI will still randomly add their battleships to the board, but the user will be prompted to add their ships manually to the board. Show all the ships to the side and the user will be able to click and drag the ships onto the board to decide their placement. Once all the ships have been added, the game can start. For this, you do not have to have both horizontal and vertical ships.

Writeup

With your submission, you must include a writeup that touches on the following points. You may discuss any other ideas that you deem salient to this work:

- What were some challenges you faced while making this app?
- Given more time, what additional features, functional or design changes would you make
- What assumptions did you make while working on this assignment?
- How long did this assignment take to complete?

Deliverables

Include all the following in your submission on Canvas:

1. A link to your Github repo. If you are working with a partner, you may submit a link to the same repo (for grading purposes, the TA's will likely only look at a single repo so make sure they are identical.) Please note that your Github repo should be named: {firstname}-{lastname}-project2, and if you're working with a second person both names should appear on the repo.
2. A link to your Heroku app. As above, if you are working with a collaborator, please submit the same link.
3. Your writeup. If you are working with a partner, you may each write this together or individually, but please indicate this with your submission.
4. The name of your collaborator, if any
5. Indicate if you attempted either or both of the bonus points

Academic Integrity

As always, all assignments are expected to uphold and follow NEU policies regarding academic integrity. Any students that violate these policies will be reported to OSCCR immediately.