TEAM – 7 BATTLESHIP PROJECT

We decided to implement Battleship as a Browser Game.

Technologies Used:
1) HTML
2) CSS
3) JavaScript

Overview of the Interface:
1) The first prompt asks the user to enter the number of ships they want to play with, between 1 and 5. If anything other than numbers between 1 and 5 are entered, it will prompt the user again
2) Firstly, only one board will be seen for the first player to place their ships, after the first player places their ships, the first board disappears, and the second board will be visible.
3) The ship placement can be done either vertically or horizontally, it can be toggled from the radio-buttons in the middle of the screen.
4) After both the users place their ships, A "Start Game" button will pop up which needs to be pressed to start the game.
5) Then the Player 1 can see his board and his ships but cannot see the ships in second player's board. (This is done by hiding the actual board of the Player 2 and rendering a new empty board instead). The same process repeats when teams are switched.
6) Hit is shown by a ✅ mark and miss is shown by a ✖ is also shown in the description box in the middle.
7) Between each turns switch, an overlay is shown which is done to make it a more practical multiplayer game (instead of just showing the opponents board while clicking switch teams.
8) The user can also see which of his ships have been hit, it is shown by a 'HIT' text in the index that was hit by the opponent.
9) After a player hits all their opponents' boards, they immediately win, and the browser will alert which player won. After that pop up is closed, the browser is refreshed – automatically starting a new game.

Details of all the files:
1) Executive.js:
   a. This is the file that starts the game. It first prompts the user about the number of ships, and renderFirstBoard() is called, which renders the first board.
   b. This file also listens to the radio buttons, that the user clicks to decide either horizontally or vertically they want to place their ships. getOrientation().
   c. The updateShipText() updates the description box on the middle of the image as to which ship should now the the player place.

d. RenderFirstorSecondBoard() function decides which board to display, after the first board is done placing ships, it will be called and it hides the first board, render the second board. After both players are done placing ships, it renders to button to "Start Game".

e. StartGame() function listens to the button "Start Game" after both the players are done placing ships. It renders back the first board with ships, and hides the second board with ships but calls the "renderBackofSecond()" functon which will then render an empty board in place of the second board cause it's Player 1's turn to attack first.

2) boardSetup.js:

a. This file contains the functions to render the first and the second game boards. renderFirstBoard() function renders the first board, and renderSecondBoard() function renders the second game board. The board is made by using a double for loop. For each iteration of the for loop, a "div" element in HTML is created and appended to the gameBoardContainer, hence putting it in a double for loop generates a board.

b. Each "div" has an event listener, that when clicked calls the placingShips() function, which places the ship starting at that position, either vertically or horizontally.

c. The first row and the first column of the boards doesn't have any event listeners because they're supposed to generate the headers for the indexes, i.e (A – J) and (1 – 10).

3) placingShips.js:

a. This file contains the placingShips() function which accepts the i and j index that the player has clicked on and the direction that the radio button is currently on.

b. The first if statement checks whether the ship is being place in the first board or the second one to correctly update the 2D arrays associated with them.

c. Then using the "numOfShips" variable which keeps on decreasing after each ship is placed a for loop is run to place the ship starting at that index. If the direction is vertical i++ is run (placing the ship vertically on the board, else j++ is run placing the ship horizontally on the board.

d. After all the ships are placed, numOfShips variable will be 0 which will then update firstBoardDone, and call RenderFirstOrSecondBoard() on Executive.js

e. Likewise, when placingShips() is called for the second board, and all the ships are done placing, renderFirstBoard() is called which then pops up the "Start game" button to start the game.

4) CheckValidIndex.js:

a. All this file does is it check whether the placing of the ships is done in a valid index or not. The function of CheckValidIndex will be called while placing the ships.
b. It will return false and update the description box in the middle both when the placing of the ship is out of bounds from the box or when the ships will overlap with each other.

5) RenderBackofBoards.js:
a. This file comes into play when switching turns. When it is Player 1's turn the renderBackofSecondBoard() will be called which will hide the second player's board with an empty board so that Player 1 cannot see Player 2's ships. Switching turns will do the vice versa with renderBackofFirstBoard().
b. The empty boards are created exactly how the initial boards where created. Instead this time, each of the boxes have event listeners to check whether the clicked box was a "hit" or a "miss" by looking at the visitedArrayForFirst and visitedArrayForSecond respectively.
c. The Hit indexes will then be marked by ✅ and miss indexes will be marked by ❌ and will also update HitArrayForFirst and HitArrayForSecond according to which player's board was hit.

6) SwitchTeam.js:
a. This file has an event listener to the "Switch Team" button after the game is started and players start firing at their opponents.
b. It makes use of the mod operator to know which player's turn it is since anything % 2 is either 0 or 1 and 0 is for the one player's turn, 1 is for the another player's turn.
c. How switching is done is by removing a class adding a class called "hidden" to the opponents board which will hide the opponent's board with ships, then a class of "hidden" from the empty new board to be placed instead of the original board.
d. After turns is switched, the switch team button disappears before the attacker attacks the other person.

7) HitOrMiss.js:
a. CheckIfHitInFirst() and CheckIfHitinSecond() are called after the attacker's attack. Those functions check whether those were hit or misses by looking at the visited arrays.
b. CheckIfWinInFirst() and CheckifWininSecond() functions check whether the game has been won or not by being called after each hit and the functions with the help of a double for loop checks whether we have enough hit counts for a win.
c. The number of hit counts for a win is determined by the recursive helper function findTotalClickCount. Since if there are 5 ships we need 1+2+3+4+5 =

15 hit counts and likewise if there are 4 ships we need 1+2+3+4 = 10 hit counts and so on.

8) toggleOverlay.js:
   a. This functions in this file are responsible for making the game more realistic multiplayer.
   b. After switch team is clicked, an overlay is shown in the frame, and when the person whose turn it is hits the 'g' key only then, the team will be switched.
   c. We decided this is better instead of just switching the boards when clicking switch teams.

9) Arrays.js:
   a. This file contains all our Arrays.
   b. Since the arrays were being called in all our files, we decided to put all our arrays in one file.

10) index.html:
   a. Contains all the HTML tags for this project.

11) Styles.css:
   a. Contains all the styles for this project.