MEHKMM Inc.

The Battleship Assistant

Who are we?

- Ethan Hastings
  - Front end UI Lead
- Matthew Mangan
  - Back End Game Function Developer
- Marcus King
  - Back End Calculation Algorithm Programmer
- Isaak NIjakoski
  - Organization, Testing, and Documentation Lead

What was the Problem?

Our problem we were trying to solve was short and sweet. We were tired of losing. We wanted an advantage over our friends whenever we decided to play them in battleship.

What is our solution?

To solve this problem we wanted to create a assistant that would aid us throughout the game. This assistant would take in our hit and miss inputs and calculate the chance of landing a hit for each unguessed grid space on the opponent's side of the board.

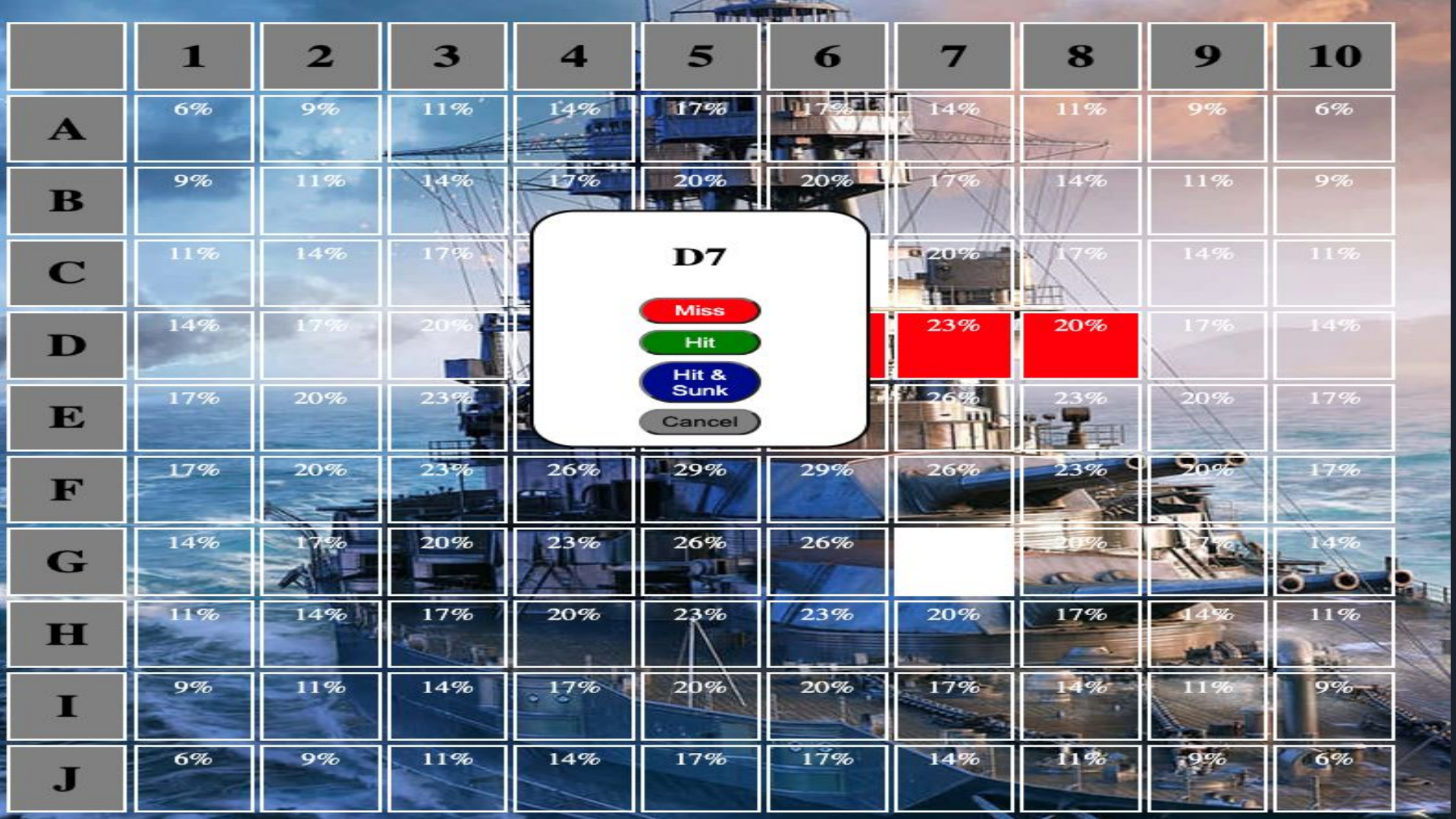
User Stories

As a user I want to see a grid representing my opponents board

	1	2	3	4	5	6	7	8	9	10
A	6%	9%	11%	14%	17%	17%	14%	11%	9%	6%
B	9%	11%	14%	17%	20%	20%	17%	14%	11%	9%
C	11%	14%	17%	20%	23%	23%	20%	17%	14%	11%
D	14%	17%	20%	23%	26%	26%	23%	20%	17%	14%
E	17%	20%	23%	26%	28%	28%	26%	23%	20%	17%
F	17%	20%	23%	26%	28%	28%	26%	23%	20%	17%
G	14%	17%	20%	23%	26%	26%	23%	20%	17%	14%
H	11%	14%	17%	20%	23%	23%	20%	17%	14%	11%
I	9%	11%	14%	17%	20%	20%	17%	14%	11%	9%
J	6%	9%	11%	14%	17%	17%	14%	11%	9%	6%

As a user I want to be able to add my hits and misses to the grid

As a user I want to see changes to the grid percentages after I make my moves


1 2 3 4 5 6 7 8 9 10
A 6% 9% 11% 14% 17% 17% 14% 11% 9% 6%
B 9% 11% 14% 17% 20% 20% 17% 14% 11% 9%
C 11% 14% 17% 20% 17% 14% 11%
D 14% 17% 20% 23% 20% 17% 14%
E 17% 20% 23% 26% 23% 20% 17%
F 17% 20% 23% 26% 29% 29% 26% 23% 20% 17%
G 14% 17% 20% 23% 26% 26% 14%
H 11% 14% 17% 20% 23% 23% 20% 17% 14% 11%
I 9% 11% 14% 17% 20% 20% 17% 14% 11% 9%
J 6% 9% 11% 14% 17% 17% 14% 11% 9% 6%
D7
Miss
Hit
Hit & Sunk
Cancel

	1	2	3	4	5	6	7	8	9	10
A	6%	9%	12%	14%	17%	17%	14%	12%	9%	6%
B	9%	12%	14%	17%	20%	20%	17%	14%	12%	9%
C	12%	14%	17%	20%	23%		20%	17%	14%	12%
D	14%	17%	20%	23%		26%	0%	20%	17%	14%
E	17%	20%	23%	26%	29%	29%	23%	23%	20%	17%
F	17%	20%	23%	26%	29%	29%	23%	23%	20%	17%
G	14%	17%	20%		26%	26%		20%	17%	14%
H	12%	14%	17%	20%	23%	23%	17%	17%	14%	12%
I	9%	12%	14%	17%	20%	20%	17%	14%	12%	9%
J	6%	9%	12%	14%	17%	17%	14%	12%	9%	6%

Architecture

Server Communication

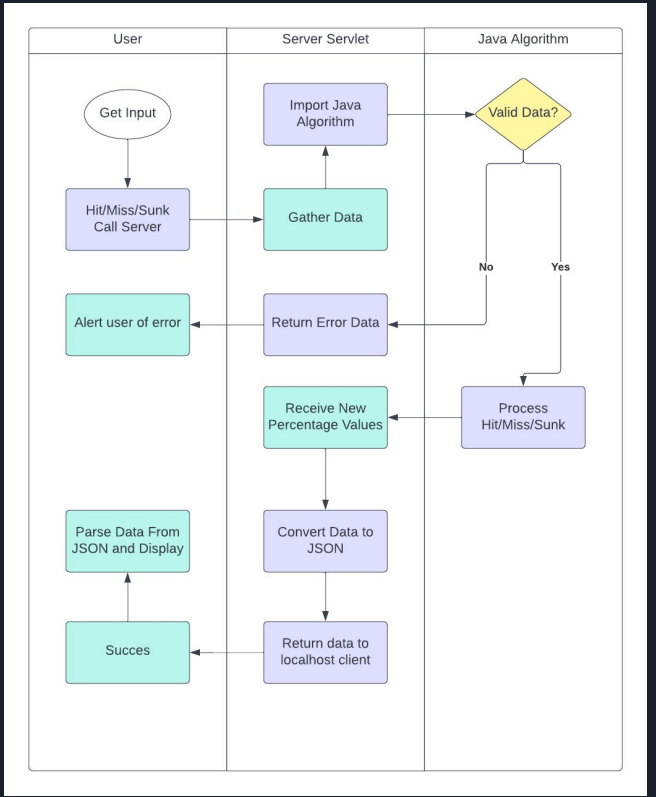
Server is built from .war file deployed on tomcat

Tomcat local server hosted by the machine

Server processes requests through a Servlet and returns data processed by the Java Algorithm

Server Communication (Cont.)

- User input is gathered by the client
- Server receives data and begins validating
- If data cannot be processed, it returns an error message
- Valid data is run through the Java algorithm
- The returned information is converted to JSON and returned to the client
- The client parses the data from JSON format and displays it to the user



Calculation Algorithm

Implemented in Java

Takes the user’s most recent guess

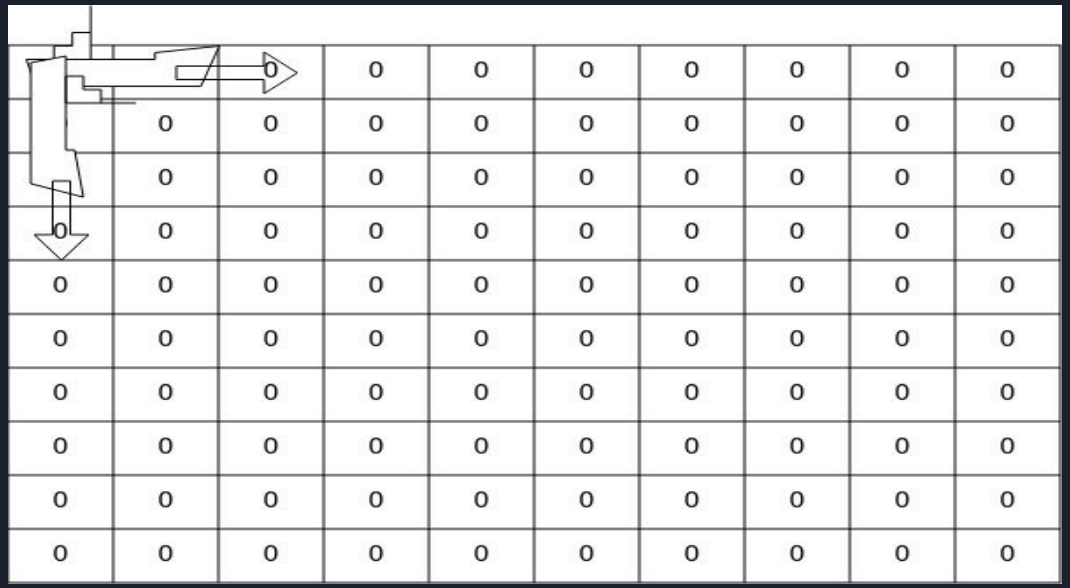
Outputs the probabilities for each cell in the grid.

Calculation Algorithm (Cont.)

Keep an integer “score” for each cell in the grid

Increment score for each possible intersection with an unsunk ship

Convert to percentages



Calculation Algorithm (Cont.)

Initial scores:

10	15	19	21	22	22	21	19	15	10
15	20	24	26	27	27	26	24	20	15
19	24	28	30	31	31	30	28	24	19
21	26	30	32	33	33	32	30	26	21
22	27	31	33	34	34	33	31	27	22
22	27	31	33	34	34	33	31	27	22
21	26	30	32	33	33	32	30	26	21
19	24	28	30	31	31	30	28	24	19
15	20	24	26	27	27	26	24	20	15
10	15	19	21	22	22	21	19	15	10

Calculation Algorithm (Cont.)

Scores after a hit and a miss:

10	15	23	21	22	21	21	19	15	10
15	20	32	26	27	25	26	24	20	15
23	32	HIT	35	35	29	31	28	24	19
21	26	35	32	33	28	32	30	26	21
22	26	33	29	29	MISS	28	27	25	21
22	27	33	33	34	29	33	31	27	22
21	26	31	32	33	29	32	30	26	21
19	24	28	30	31	29	30	28	24	19
15	20	24	26	27	26	26	24	20	15
10	15	19	21	22	22	21	19	15	10

Design Specification Implementation

The User Shall Be Able to View a Battleship Grid

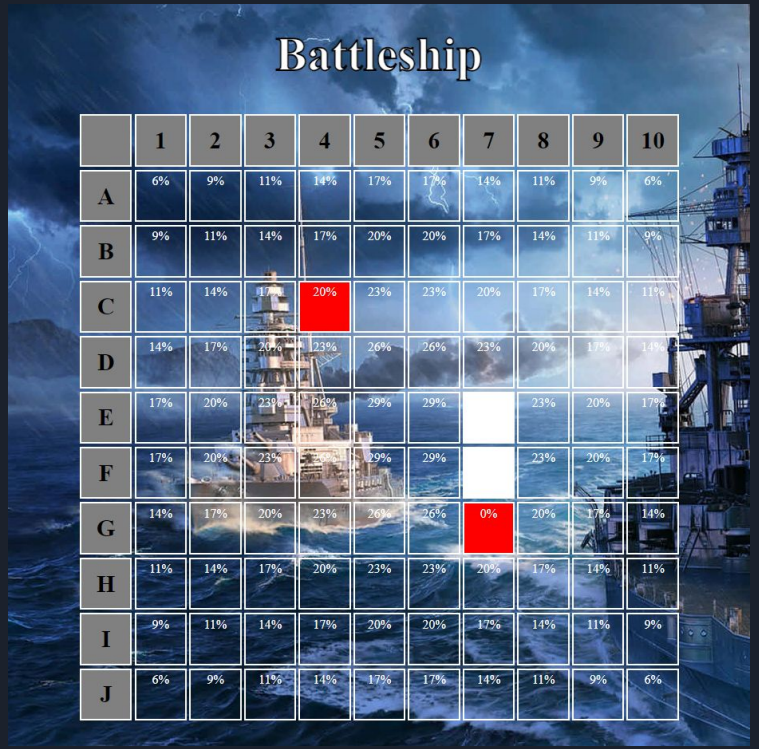
- A 10x10 Battleship grid must be viewable and interactable to the user in which the user can click on each part of the grid separately

- A background image representing battleship will exist

The User Shall Be Able to Update the Game Board

- The user must be able to modify the state of the game board through clicking the individual grid squares, and the board should update accordingly

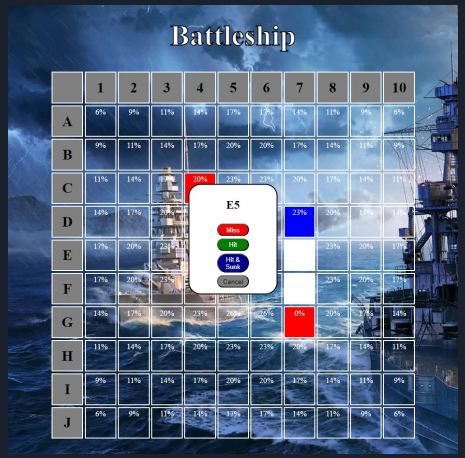
- If the user input is valid, the frontend board display will update automatically without any additional input from the user

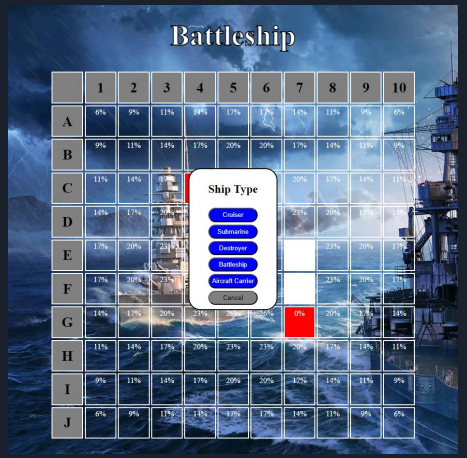


The User Shall Be Able to Update the Game Board (Cont.)

- The user will be prompted to choose whether the guess was a hit, miss, or sunk

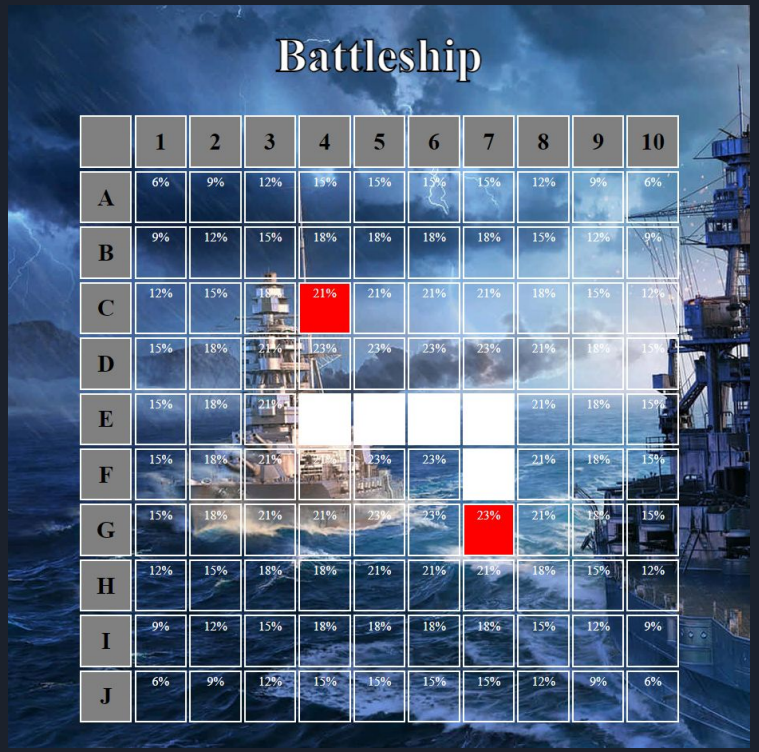
- If the user selects sunk, they will be prompted to select which ship was sunk





The User Shall Be Guided Where to Play Next

- A text value will be present in each unguessed grid cell informing the user of the chance of landing a hit on that cell
- After each attack, the square will become white or red to represent a hit or miss respectively
- The algorithm will update the status of the game state to the best of its ability given the info from previous guesses

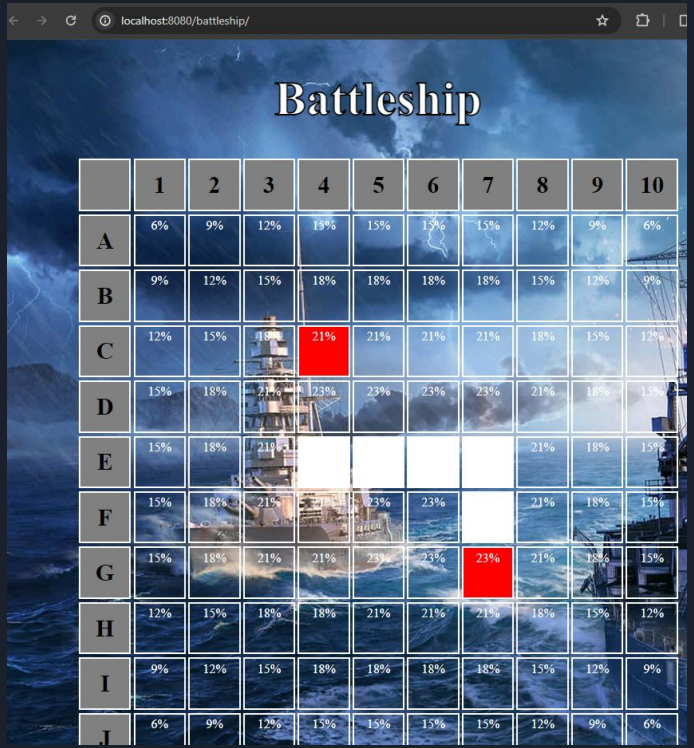


The Board Shall Restart After a Page Reset

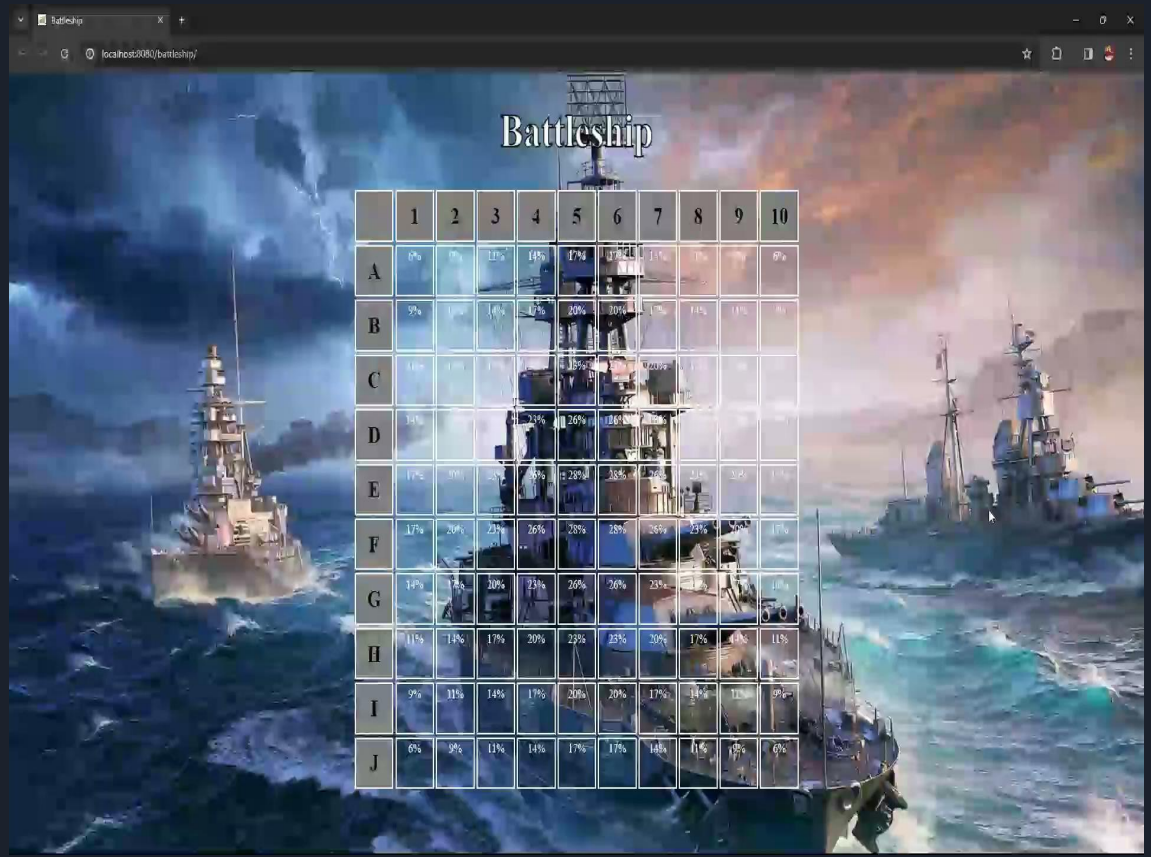
- Board is cleared of all user input when the refresh browser button is pressed

The System Shall Use a Server to Connect the Client to the Java Algorithm

- User is connected to the local tomcat server hosted by the machine visible in the URL bar



Project Demonstration



If we had more time and resources

Data Tracking

- If we had unlimited time with this product we would have implemented data tracking help produce more accurate percentages in the grid
- For example after 1,000 games played if A5 was a hit 800 times that would up the percentage of possibility on our grid.
- That would however require a lot of work and a soft release of our product to start gathering data