Battleship Game Project Write-up

Challenges Faced During Development

One of the biggest challenges in building this game was managing the different game states—especially syncing hits and misses across both the player and computer boards. I used the Context API to handle state globally, which took some careful planning. Creating the computer AI was also tricky: I wrote a targeting system where the AI starts with random shots but switches to nearby tiles after a hit. It took a few rounds of tweaking to make it feel smart but fair. For the game board, I used a 1D array to simulate a 2D grid—it helped with performance but made the logic for ship placement and coordinate conversion more complex. The drag-and-drop ship placement was another tough part, since I had to handle orientation, overlaps, and boundaries, all while giving the player real-time visual feedback. Lastly, I spent time making sure the game works well on both desktop and mobile, including switching layouts for smaller screens.

Potential Future Improvements

I think adding a multiplayer mode with websockets would take the game to the next level playing against real people is always more fun than just battling a bot. On the visual side, adding sound effects and animations for hits, misses, and sunk ships would definitely make the game more immersive. I'd also like to add different difficulty levels for the AI, so both beginners and more experienced players can enjoy it. Right now the leaderboard is static, so hooking it up to a real database to track scores would add a nice competitive touch. And finally, letting users customize the game like choosing board size, number of ships, or enabling special power-ups would really boost replay value.

Assumptions Made

While working on the game, I made a few assumptions to help guide the design. First, I assumed users were already familiar with how Battleship works, so I didn't include too many instructions—I expected most people to figure things out just by using the interface. I also built and tested the game mainly on modern browsers, assuming people wouldn't be using outdated ones. For screen size, I made the layout responsive, but I did assume most users would be on devices big enough to show the whole board comfortably. In terms of difficulty, I tried to make the AI challenging but not too frustrating—basically aiming for something fun for the average player. And finally, I used localStorage to save game state, assuming users would have it enabled in their browser.

Time to Complete

This project took approximately 42 hours to complete, spread across the following areas:

- Initial planning and design: 4 hours
- Core game logic implementation: 10 hours
- UI component development: 8 hours
- State management with Context API: 5 hours
- Computer AI implementation: 6 hours
- Styling and responsive design: 3 hours
- Testing and debugging: 6 hours