Client responsibilities

The client is responsible for all user-facing behavior and execution of game logic. All client responsibilities are implemented in the browser using HTML for structure, CSS for presentation, and JavaScript for control and logic.

- Renders the game interface, including both boards, status messages, and controls.
- Captures and processes user input such as firing at grid cells and resetting or clearing the game.
- Enforces turn-taking rules and game mechanics.
- Executes the AI logic for the computer player.
- Persists and restores game state using browser storage.

Server responsibilities

There is no server component in this architecture. This design choice simplifies deployment and is appropriate for a single-player game.

- The application is entirely client-side with no backend services used.
- Game state, logic, and persistence are handled locally within the browser.

Where game state lives

Game state is maintained in JavaScript as in-memory data, which acts as the single source. The state is stored in localStorage to ensure persistence across page reloads. The user interface is reconstructed from this stored state when the page loads.

- Board state for both players, including ship locations, hits, and misses.
- Turn state indicating whose turn it is.
- Game status indicating whether the game is active or complete.
- Shot statistics for both the user and the AI.
- Internal AI memory used to support hunt and target behavior.

How state transitions occur

State transitions are event-driven and governed by explicit game rules:

- User actions (clicking a cell) trigger validation, state updates, and UI updates.
- AI actions are invoked automatically when it becomes the AI's turn and follow deterministic hunt/target logic.
- Turn control is determined by the outcome of a shot: successful hits allow consecutive turns, while misses transfer control.
- Win conditions are checked after each successful hit sequence.
- Each meaningful state change results in the game state being saved to persistent storage.