High-Level Design - Battleship Game

1. Overview

This is a simple Battleship game where players fire shots on a 10x10 grid to sink enemy ships. The game has 03 main parts:

• React Frontend (User Interface)

• .NET 8 API Backend (Game Logic)

• Console Version (Command-line-based alternative)

2. Technologies Used

• Frontend: React (JavaScript)

• Backend: C# (.NET 8 API)

• Console Game: C# .NET

3. Game Rules

• The player fires shots at a 10x10 grid.

• There is 1 Battleship (size 5) and 2 Destroyers (size 4).

• The Battleship is visible, but Destroyers are hidden.

• The game tracks hits, misses, and sunken ships.

• The player cannot fire at their own Battleship.

• The game ends when both Destroyers are sunk.

4. System Architecture

1. Frontend (React)

• Displays the game grid.

• Sends API requests to fire shots.

• Updates the board based on API responses.

• Prevents shooting at the same position twice.

2. Backend (C# .NET API)

• Places ships randomly on the grid.

• Receives shot positions from the frontend.

• Checks for hits, misses, and sunk ships.

• Sends updated board status to the frontend.

3. Console Version (C#)

• Runs as a command-line game.

• The player enters positions like 'A5'.

• Displays hits, misses, and sunken ships in text format.

5. API Endpoints

Method	**Endpoint**	**Description**
POST	/api/Game/reset	Resets the game.
GET	/api/Game/ships	Fetches ship positions.
POST	/api/Game/fire	Fires a shot at a location.