Battleship Game System Requirement Specification

1. Introduction

This document outlines the design of a Battleship game implemented as state tracking API. The purpose is to provide a detailed description of the game's architecture, components, and functionality.

2. Game Overview

The Battleship game is a single-player strategy game where each player has a fleet of ships on a 10x10 grid-based board. Players take turns guessing the coordinates to attack each other's ships. The goal is to sink all of the opponent's ships before they sink yours.

3. Game Components

3.1 Game Board

A grid-based board where ships are placed and attacks are made.

Each cell on the board is identified by x-y coordinates (e.g., 0,1).

3.2 Ships

Randomly generated ships with random sized upto 4 ships are place on the board vertically or horizontally and occupy consecutive cells.

3.3 Players

Single players participate in the game, attack on a ship.

Player has opponents board which has randomly generated ships. Player hit API end-points with the co-ordinates to attacks ships

4. Game Logic

4.1 Setup Phase

computer place ships randomly on the board.

Ship placement follows game rules (no overlapping or out-of-bounds placements).

4.2 Gameplay

Players take turns to make attacks by specifying coordinates on the opponent's board.

Attacks result in hits (if a ship occupies the targeted cell) or misses (if the cell is empty).

Sinking a ship requires hitting all of its cells.

4.3 Victory Condition

The game continues until player sinks all of the opponent's ships.

The player who sinks all enemy ships wins the game.

5. API Endpoints

5.1 Game Setup

- https://jayantkbattleshipcodingtest.azurewebsites.net/api/CreateBoard:
  Creates a new board.
- https://jayantkbattleshipcodingtest.azurewebsites.net /api/AddShip:
  Places a ship on the player's board during setup phase.

  Need to Pass TotalShips from Json body

  Ex. {"TotalShips":1}
- https://jayantkbattleshipcodingtest.azurewebsites.net /api/ShowBoard:
  Show board as API response, ships are hidden on the board.
- https://jayantkbattleshipcodingtest.azurewebsites.net /api/RestartGame:
  starts the new game.

5.2 Gameplay

- https://jayantkbattleshipcodingtest.azurewebsites.net /api/FireShip:
  Allows a player to make an attack on the board. API respond with the hit, miss and game win message.

  Need to Pass board coordinates from Json body

  Ex.

  {

  "Xcordinate":4,

  "Ycordinate":7

  }

6. Data Structures

6.1 Board

Represents a player's game board with cells and ships.

Contains methods for placing ships, making attacks.

6.2 Ship

Defines a ship object with attributes such as size, position.

7. Error Handling

Define standard error codes and messages for API responses.

Handle validation errors for invalid requests (e.g. If added 0 ship).

Implement global exception handling to ensure graceful error responses.

8. Testing and Quality Assurance

Unit Testing: Test individual components such as Board and Ship classes.

API Testing: Use tools like Postman for manual API testing.

Quality Assurance: Conduct functional, performance testing to ensure a robust game experience.

9. References

.NET Core documentation and best practices.

Battleship game rules and strategies.

xUnit test frameworks.