# Final project

You will build a simplified version of the Classic Battleship game with the following ship configuration:

- 1 ship of size **4**
- 2 ships of size **3**
- 3 ships of size **2**
- 4 ships of size **1**

**Important rule:** ships must **not touch each other**, even diagonally.

Your repository root should look roughly like this:

```
├─ main.py
├─ data/
│  ├─ player_ships.csv
│  ├─ bot_ships.csv
│  ├─ game_state.csv
├─ src/
│  ├─ ship_input.py
│  ├─ bot_generation.py
│  ├─ gameplay.py
│  └─ utils.py
├─ outputs/
│  └─ (any logs or additional result files)
├─ requirements.txt
└─ README.md
```

**Requirements:**

`main.py` is the **entry point**. After installing the requirements I should be able to run `python main.py` and everything happens automatically. Inside `main.py`, you must:

## Part 1 — Player Ship Input (`src/ship_input.py`)

Write a script that:

1. Reads ship positions from user input (in any format you choose—free-form, coordinates list, grid, etc.).
2. Validates the input:
    - Ships must have correct sizes (4, 3, 3, 2, 2, 2, 1, 1, 1, 1)
    - Ships must be inside the 10×10 board.
    - Ships cannot touch each other (even diagonally).
    - Saves the valid ship placement into `data/player_ships.csv`
3. The CSV format is free-form, but must clearly represent ship coordinates.

## Part 2 — Bot Ship Generation (src/bot_generation.py)

Write a script that:

1. Automatically generates a **valid random ship layout** for the bot.
2. It must follow all rules:
    - No adjacency
    - No going outside the board
    - Ship counts and sizes must be correct
    - Saves the bot layout to: `data/bot_ships.csv`
3. Should be the same format as player_ships.csv.

This ensures the bot and the player use **compatible** formats.

## Part 3 — Game State Tracking (src/gameplay.py)

You must create a continuously updated CSV file tracking all moves: `data/game_state.csv`

This file must contain (at minimum):

- turn number
- player move (coordinate + hit/miss)
- bot move (coordinate + hit/miss)
- state of all cells after the move

A few **important rules**:

### 3.1 Display Game State After Each Move

Every time a move is made:

1. Update the CSV.
2. Print a **visual 10×10 board** to the terminal for both players:
    - Hits
    - Misses
    - Unknown cells

### 3.2 Marking Terrain After a Ship Is Destroyed

If a ship is fully destroyed:

- **All surrounding cells** (8 directions) must automatically be marked as **miss** in the game state.
- These marks must also be reflected in the CSV file.

## Part 4 — Gameplay Loop

Your game must be **playable**:

- Read user input (any format you define).
- Apply rules.

- Update the state.
- Display the updated board.
- Stop when one side has no ships remaining.

The bottom line: **The player must be able to play Battleship through your terminal.**

# Part 5 — Bot Move Logic

You must implement a simple but intelligent bot with the following behavior:

## 5.1 Random Shooting

The bot initially picks random untested cells.

## 5.2 Smart Follow-up After First Hit

If the bot scores a hit and the ship is **not size 1**, then:

- It chooses one of the 4 adjacent cells (up/down/left/right) that is still available.

## 5.3 Axis Locking After a Second Hit

If the bot gets a **second consecutive hit on the same ship** and the ship is not size 2:

- Determine the orientation (horizontal or vertical).
- Continue searching **only along that axis**.
- It must check both directions, but **cannot continue past misses** or outside the board.

When the ship is destroyed:

- Bot returns to **random mode**.

# README.md Requirements

Your report must include:

- How your input format works
- How you validate ship placements
- How you update and display the game state
- Any design decisions or trade-offs

## Git Requirements

You **must**:

- Initialize a Git repository in the project root.
- Make at least one feature branch (e.g., `feature/bot_ai`).
- Do work there and merge it into main.
- Push the repository to GitHub.
- Submit **only the GitHub URL**.

# THE TASK IS DUE 19.12.2025 10:45, THE GITHUB REPO URL SHOULD BE SENT TO YOUR TEACHING ASSISTANT.