



Battleship Game

A turn-based network game built in Python using UDP sockets

Real-time multiplayer
game

Synchronization,
validation, error
handling

Client-server
architecture using UDP

 2 contributors

Project summary

This project implements a networked, turn-based Battleship game in Python 3.13 using UDP sockets.

Two players connect to a central server, place their fleets on a 10×10 grid, and fire at each other's ships. The server mediates all messages between players, handling connection, shot requests, hit/miss verification, turn control, and game termination.

The architecture demonstrates low-latency UDP communication, error handling for invalid messages, and modular code design for client and server components. The system has been designed for simplicity, clarity, and robustness.

Project Objectives & Requirements



Core Objectives

Implement a multiplayer Battleship game using UDP, manage turn-based firing with hit/miss verification, and design modular client-server code for maintainability.



Functional Requirements

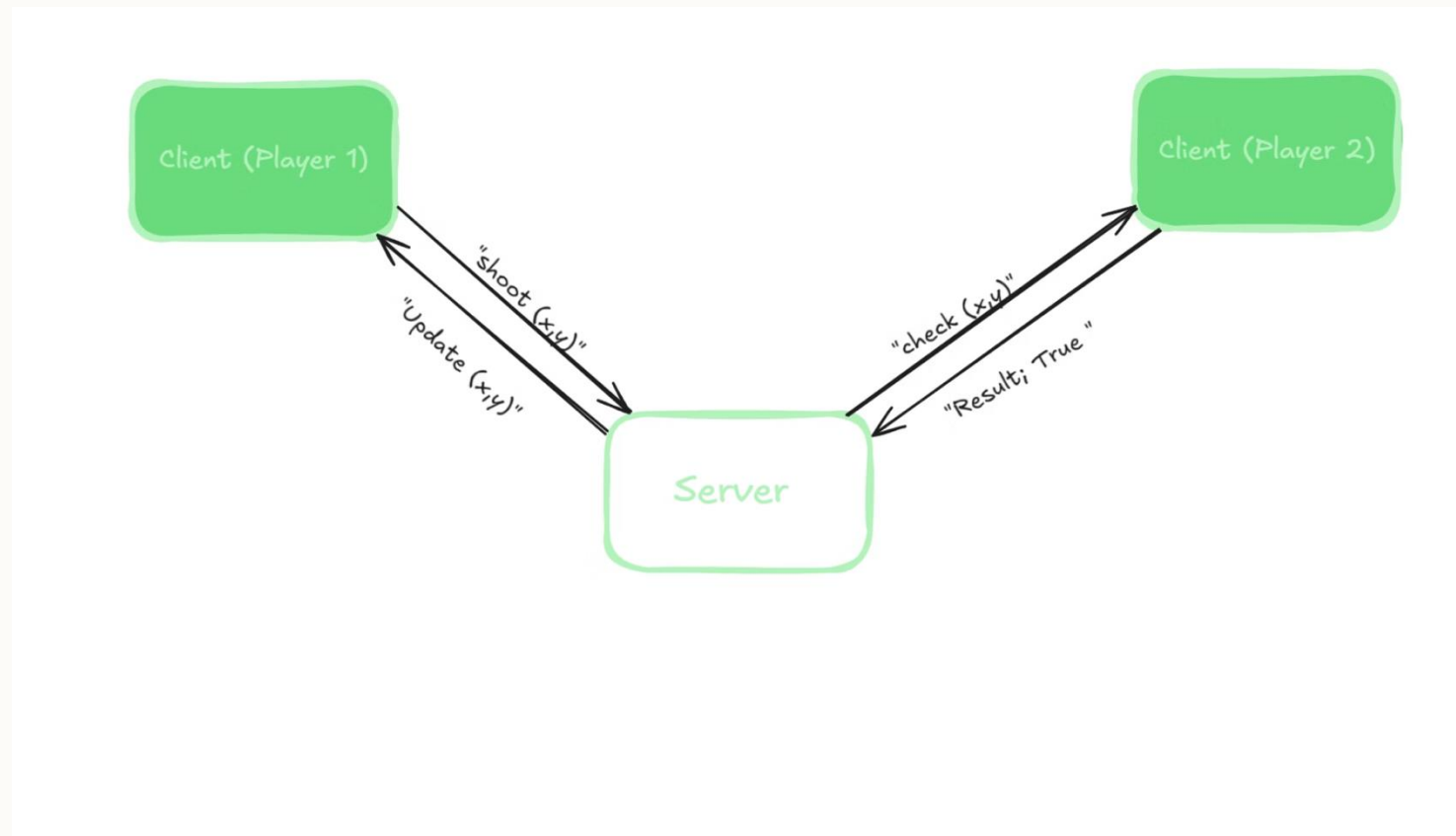
Allow two players to connect via UDP, place ships manually or using predefined layouts, alternate turns automatically, validate hits/misses, and end the game when one player either loses all ships, exceeds the time limit, or surrenders.



Non-Functional Requirements

Ensure low-latency communication, robust error handling, modular code structure, user-friendly console UI, and support for multiple concurrent game rooms.

System Architecture – Client-Server Model with UDP



Follows a client-server model with 2 clients and 1 server acting as the game coordinator.

The server handles:

- Room creation and player assignment
- Turn coordination and message relaying
- Game termination detection

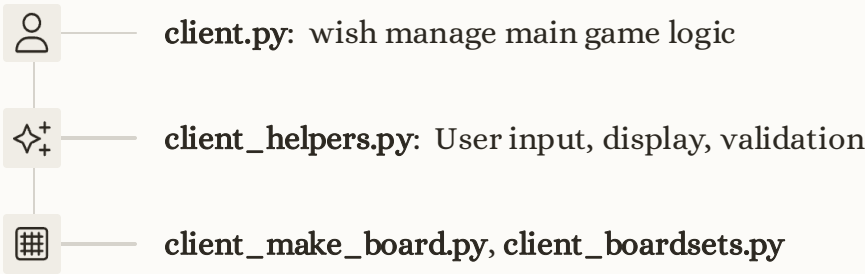
Clients interact only through the server (no peer-to-peer communication).

Communication uses UDP to reduce latency.

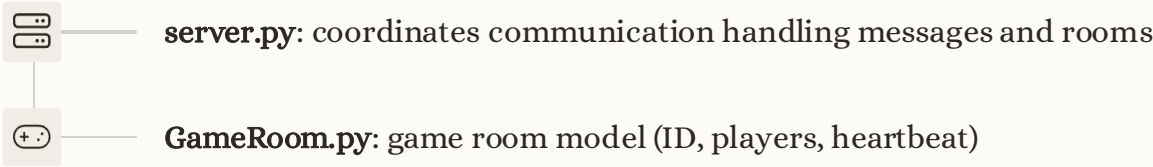
The server maintains game state consistency and validation

Modular Code Structure

Client

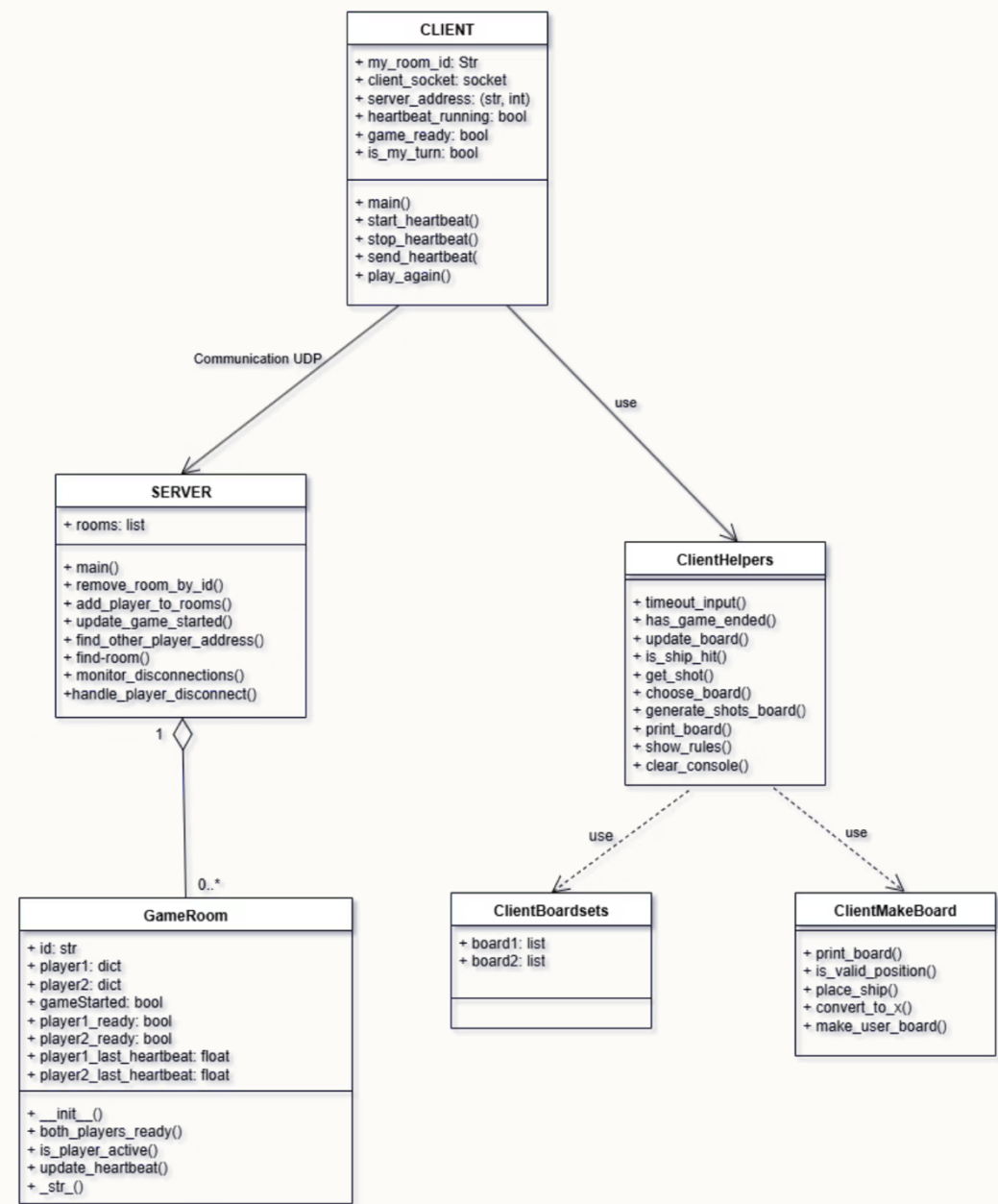


Server



The system follows a modular design where each component has a clearly defined role. On the **client side**, `client.py` manages the main game flow, `client_helpers.py` handles input, display, and validation, while `client_make_board.py` and `client_boardsets.py` are responsible for creating and managing the game board and ship layouts. On the **server side**, `server.py` coordinates communication and room management, and `GameRoom.py` encapsulates game session data, including player tracking and heartbeat monitoring. This modularity improves clarity, debugging, and long-term maintainability.

Class Diagram



Class Diagram

Client

- Responsibilities:

 - Connect to the server using UDP.
 - Manage game flow: setup, turn-taking, game end, replay.
 - Send/receive game data like shots and results.
 - Monitor timeouts and opponent disconnections.
 - Launch heartbeat for disconnection detection.

SERVER

- Responsibilities:

 - Accept new player connections via UDP.
 - Manage matchmaking by creating or filling rooms.
 - Relay messages between players (e.g., moves, results).
 - Monitor player heartbeats for disconnections.
 - Control game flow (start, turns, end).

GameRoom

- Responsibilities:

 - Store and manage state for a 1v1 match.
 - Track readiness and connection status of each player.
 - Identify disconnections based on heartbeat timeout.

ClientHelpers

- Responsibilities:

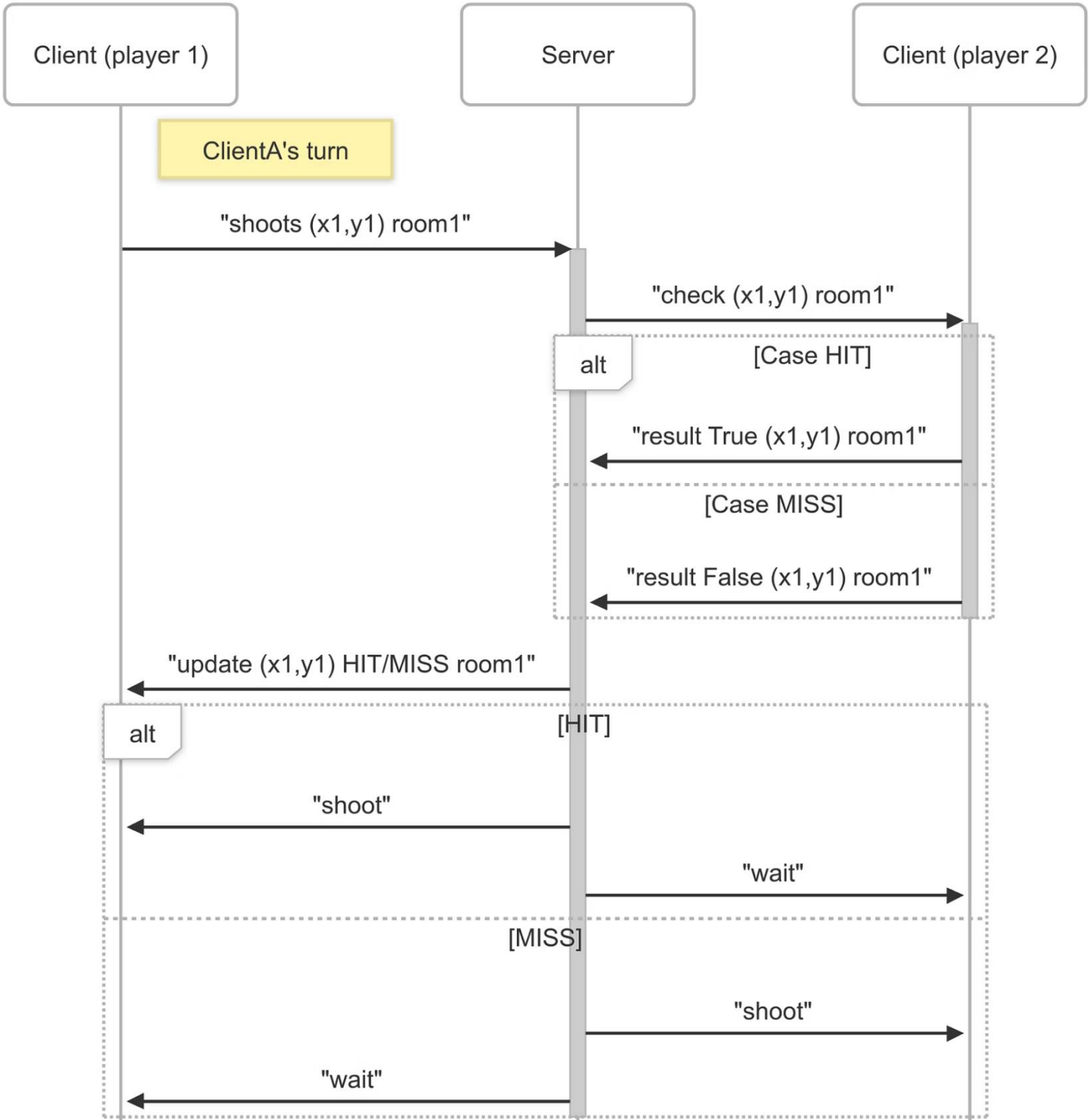
 - Handle timed input from the user.
 - Update and print the game board.
 - Detect if the game is over (all ships hit).
 - Prompt for shooting coordinates and validate input.
 - Present game rules and handle display cleanup.

ClientBoardsets

- Responsibilities:

 - Offer ready-made board configurations to reduce setup time.
 - Support player choice during game preparation.

Sequence Diagram



- **Client (Player 1):** Initiates the game by taking a turn and shooting at coordinates (x1, y1) in room1.
- **Server:** Receives the shot, checks if it's a **HIT** or **MISS**, and sends the result (**True** for hit, **False** for miss) back to Player 1.
- **Also server:** Updates the board with the result (HIT/MISS) and either continues shooting (if it's their turn) or waits for Player 2's turn.
- **Alternate Outcomes:**
 - **Player Surrenders:** Triggers an "end" message, declaring the winner and terminating the game.
 - **All Ships Sunk:** Results in an "end_you_won" message for the winning player, followed by a "winner" announcement and game termination.
- **Client (Player 2):** Acts as the opponent, waiting for their turn or responding to game-ending conditions.

Implementation Details

- **Network Layer (UDP Protocol)**

- UDP is used for low-latency, connectionless communication.
- Messages are formatted as simple strings: `shoots`, `check`, `result`, etc

- **Turn-Based Control (Server-side Enforcement)**

- The server strictly controls player turns by sending `shoot` and `wait`.
- Only one player is allowed to act at any given time, avoiding race conditions.

- **Game State Management**

- The server manages game rooms, player readiness, and game progression.
- Clients maintain their own boards and update them only upon receiving server messages (e.g., `update`).
- Victory is detected locally when all enemy ships are hit.

- **Timeout & Fault Tolerance**

- Players have 30 seconds to make a move; otherwise, they lose automatically.
- Clients send regular heartbeat signals every 10 seconds.
- The server monitors these signals and removes inactive players after 30 seconds of silence.

- **Error & Disconnection Handling**

- Invalid messages are ignored to prevent crashes.
- If a player quits or disconnects, the opponent is notified and the game ends gracefully.
- Game sessions are cleaned up automatically to free resources.

Validation & Testing



Manual Gameplay

Complete games played from board creation to win detection



Edge Case Handling

Handled invalid inputs, out-of-bounds shots, repeated coordinates, surrender



Interface Testing

Checked board rendering, error messages, and turn-based prompts



Communication

Verified UDP message exchange and server logging under real conditions

Validation & Testing

```
C:\Windows\System32\cmd.exe - py client.py
=====
Shoot on ['5', '1'] is a HIT!
=====

Your board          Your shots
 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10
1 X - X - - - - - 1 - - - H H - - - -
2 X - X - - X X - - 2 - - - - - - - -
3 X - X - - - - - 3 - - - - - - - -
4 X - - - X - - - - 4 - - - - - - - -
5 - - X - - - X X - - 5 - - - - - - - -
6 - - X - - - - - 6 - - - - - - - -
7 - - X - X - - - - 7 - - - - - - - -
8 - - - - - - - - 8 - - - - - - - -
9 - X - - - - X X - - 9 - - - - - - - -
10 - - - X - - - - - 10 - - - - - - - -
=====

Your turn
Enter your shot
Enter x coordinate: 5
Enter y coordinate: 1
This place has already been shot! Enter new data.
Enter your shot
Enter x coordinate: _
```

When a player attempts to shoot a location that has already been targeted, the system displays an appropriate message.

Validation & Testing

```
C:\Windows\System32\cmd.exe - py client.py
Write 3 - if you want to choose predefined set number 2
Your choice: 1
Your board
  0 1 2 3 4 5 6 7 8 9
0 - - - - -
1 - - - - -
2 - - - - -
3 - - - - -
4 - - - - -
5 - - - - -
6 - - - - -
7 - - - - -
8 - - - - -
9 - - - - -
Enter coordinates for a ship of size 4 (x,y): az
Invalid data. Try again.
Enter coordinates for a ship of size 4 (x,y): 1223
Invalid data. Try again.
Enter coordinates for a ship of size 4 (x,y): a,a
Invalid data. Try again.
Enter coordinates for a ship of size 4 (x,y): 2,3
Enter the ship orientation (h - horizontal, v - vertical):
Invalid orientation! Please enter 'h' for horizontal or 'v' for vertical.
Enter the ship orientation (h - horizontal, v - vertical): l
Invalid orientation! Please enter 'h' for horizontal or 'v' for vertical.
```

When invalid ship placement input is provided during custom board setup, the system displays appropriate error messages to guide the user:

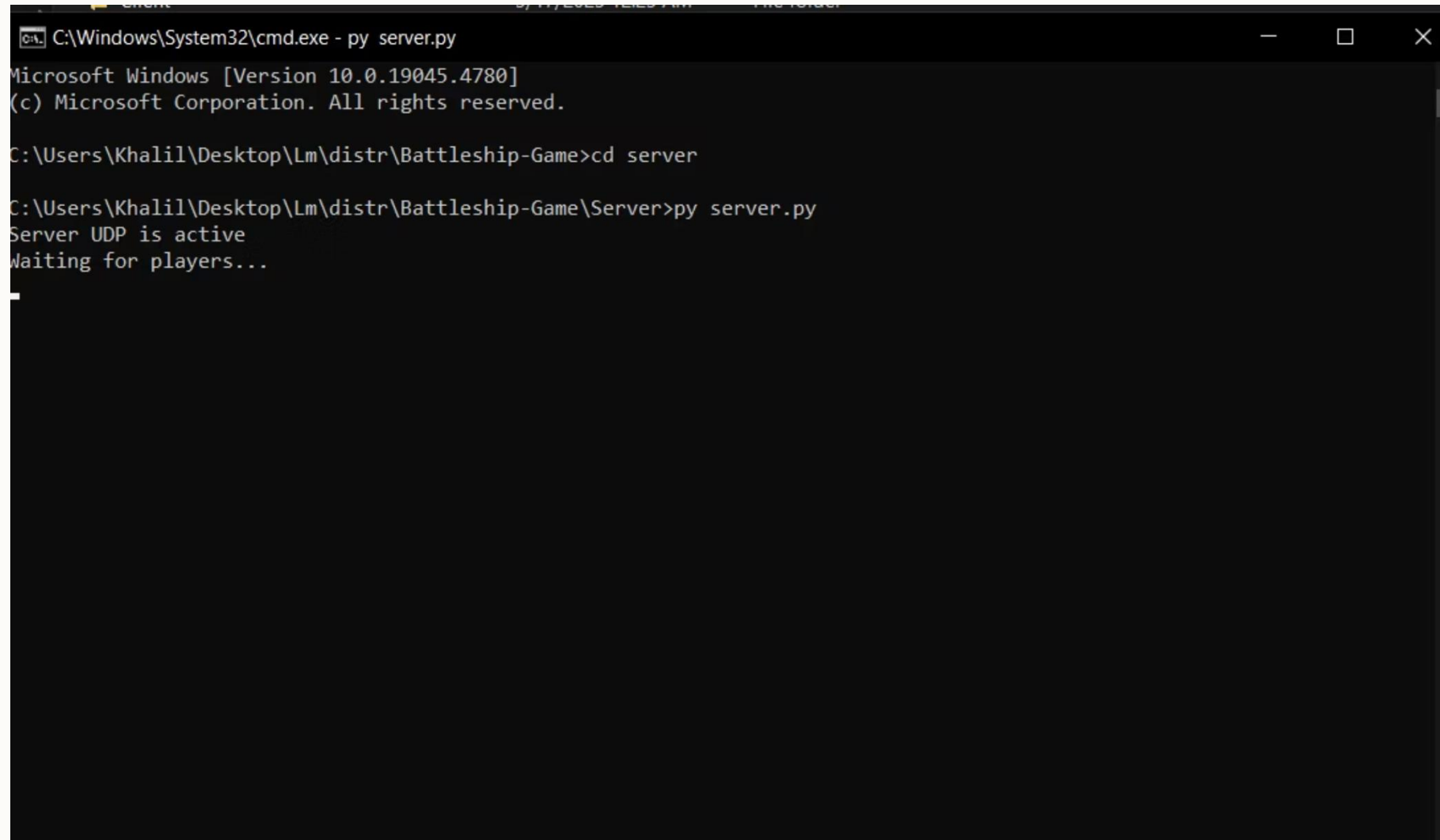
- Rejects non-numeric coordinates (e.g., "az", "a,a").
- Blocks coordinates outside the valid range (must be **1-10**).
- Ensures ship orientation is strictly "h" (horizontal) or "v" (vertical).

Usage Example

The Battleship game implementation showcases both client and server functionality in action. These screenshots demonstrate key interactions during actual gameplay.

The terminal-based interface provides a clean, responsive experience while maintaining the UDP communication architecture described earlier.

Server Initialization



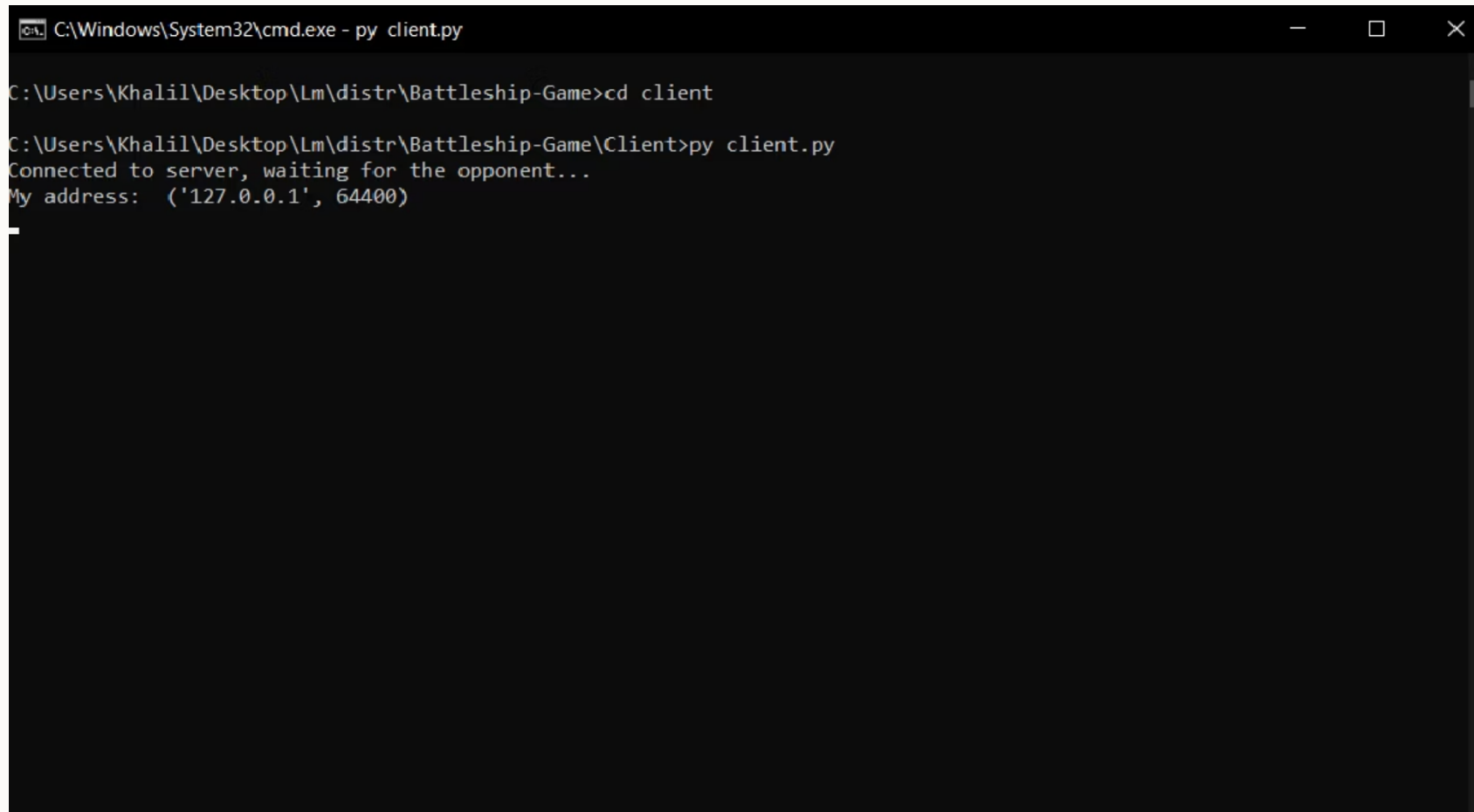
```
C:\Windows\System32\cmd.exe - py server.py
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game>cd server

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game\Server>py server.py
Server UDP is active
Waiting for players...
```

- Launching the game server using UDP protocol
- Server displays active status message
- Ready to accept player connections

Player Joining the Game



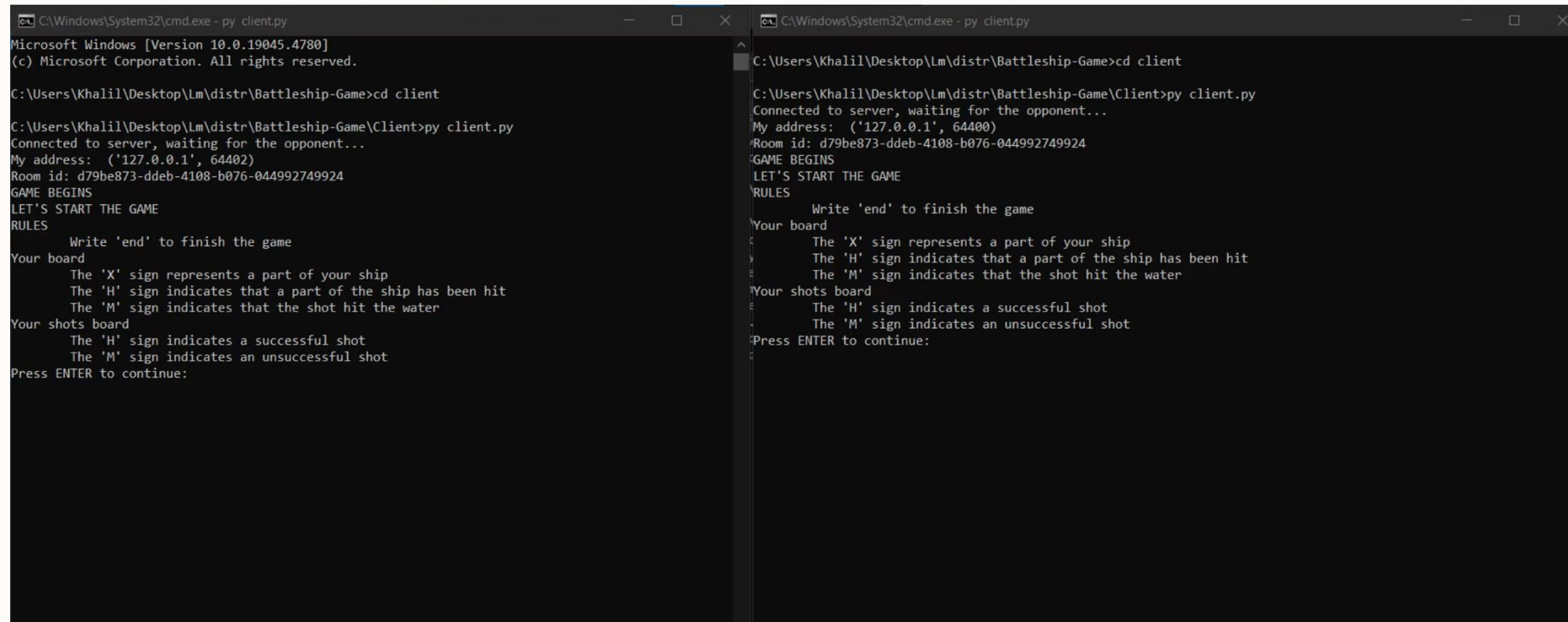
```
C:\Windows\System32\cmd.exe - py client.py

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game>cd client

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game\Client>py client.py
Connected to server, waiting for the opponent...
My address: ('127.0.0.1', 64400)
```

- Client successfully connects to server
- Displays local IP address and port
- Waiting state for opponent connection

Battle Instructions



```
C:\Windows\System32\cmd.exe - py client.py
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Khalil\Desktop\Lm\distr\Battleship-Game>cd client

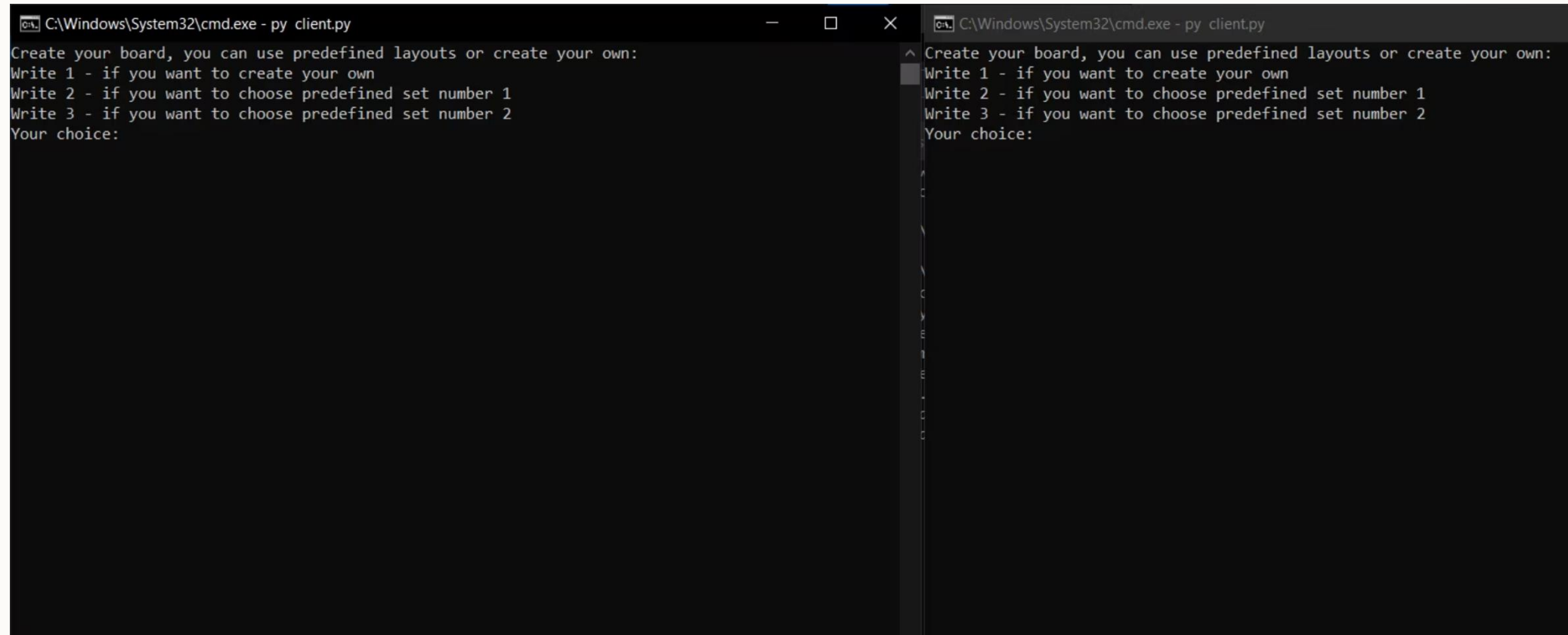
C:\Users\Khalil\Desktop\Lm\distr\Battleship-Game\Client>py client.py
Connected to server, waiting for the opponent...
My address: ('127.0.0.1', 64402)
Room id: d79be873-ddeb-4108-b076-044992749924
GAME BEGINS
LET'S START THE GAME
RULES
    Write 'end' to finish the game
Your board
    The 'X' sign represents a part of your ship
    The 'H' sign indicates that a part of the ship has been hit
    The 'M' sign indicates that the shot hit the water
Your shots board
    The 'H' sign indicates a successful shot
    The 'M' sign indicates an unsuccessful shot
Press ENTER to continue:
```

```
C:\Windows\System32\cmd.exe - py client.py
C:\Users\Khalil\Desktop\Lm\distr\Battleship-Game>cd client

C:\Users\Khalil\Desktop\Lm\distr\Battleship-Game\Client>py client.py
Connected to server, waiting for the opponent...
My address: ('127.0.0.1', 64402)
Room id: d79be873-ddeb-4108-b076-044992749924
GAME BEGINS
LET'S START THE GAME
RULES
    Write 'end' to finish the game
Your board
    The 'X' sign represents a part of your ship
    The 'H' sign indicates that a part of the ship has been hit
    The 'M' sign indicates that the shot hit the water
Your shots board
    The 'H' sign indicates a successful shot
    The 'M' sign indicates an unsuccessful shot
Press ENTER to continue:
```

- Key game symbols explained:
 - X = Ship
 - H = Successful hit
 - M = Missed shot
- End game command shown ('end')
- Interactive prompt to begin gameplay

Board Setup Options



The image shows two side-by-side terminal windows. Both windows have a title bar that reads 'C:\Windows\System32\cmd.exe - py client.py'. The left window displays the following text: 'Create your board, you can use predefined layouts or create your own:', 'Write 1 - if you want to create your own', 'Write 2 - if you want to choose predefined set number 1', 'Write 3 - if you want to choose predefined set number 2', and 'Your choice:'. The right window displays the same text, but with a cursor positioned at the end of the 'Your choice:' line.

- Players choose their battle configuration:
 - 1 - Create custom board (manual ship placement)
 - 2 & 3 - Use predefined board layouts

Multiplayer Session Initialization

```
C:\Windows\System32\cmd.exe - py server.py
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game>cd server

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game\Server>py server.py
Server UDP is active
Waiting for players...
New player joined: ('127.0.0.1', 64400)
Created new room(d79be873-ddeb-4108-b076-044992749924) and added ('127.0.0.1', 64400) as player 1
New player joined: ('127.0.0.1', 64402)
Added ('127.0.0.1', 64402) as player 2 to room d79be873-ddeb-4108-b076-044992749924
Room d79be873-ddeb-4108-b076-044992749924 | GAME STARTS
Room d79be873-ddeb-4108-b076-044992749924 | Player that begins: ('127.0.0.1', 64402)
```

- Server automatically creates game room when 2 players connect
- Unique room ID generated for each session
- Assigns player numbers (1 & 2)
- Randomly selects starting player

Custom Board Creation

```
C:\Windows\System32\cmd.exe - py client.py
Create your board, you can use predefined layouts or create your own:
Write 1 - if you want to create your own
Write 2 - if you want to choose predefined set number 1
Write 3 - if you want to choose predefined set number 2
Your choice: 1
Your board
 0 1 2 3 4 5 6 7 8 9
0 - - - - -
1 - - - - -
2 - - - - -
3 - - - - -
4 - - - - -
5 - - - - -
6 - - - - -
7 - - - - -
8 - - - - -
9 - - - - -
Enter coordinates for a ship of size 4 (x,y): 3,2
Enter the ship orientation (h - horizontal, v - vertical): h
Your board
 0 1 2 3 4 5 6 7 8 9
0 - - - - -
1 - - - - -
2 - - 4 4 4 4 - -
3 - - - - -
4 - - - - -
5 - - - - -
6 - - - - -
7 - - - - -
8 - - - - -
9 - - - - -
Enter coordinates for a ship of size 3 (x,y): 1,1
Enter the ship orientation (h - horizontal, v - vertical): v
Your board
 0 1 2 3 4 5 6 7 8 9
0 - - - - -
1 - 3 - - - - -
2 - 3 - 4 4 4 4 - -
3 - 3 - - - - -
4 - - - - -
5 - - - - -
6 - - - - -
7 - - - - -
8 - - - - -
9 - - - - -
Enter coordinates for a ship of size 3 (x,y):
```

- Step-by-step ship placement:
 1. Enter coordinates (x,y)
 2. Choose orientation (H/V)
 3. Progress through ship sizes (4 → 3 → etc.)
- Visual feedback after each placement

Instant Battle Setup

```
C:\Windows\System32\cmd.exe - py client.py

=====

Your board          Your shots
 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10
1 X - X - - - - - - 1 - - - - - - - - -
2 X - X - - X X - - - 2 - - - - - - - - -
3 X - X - - - - - - - 3 - - - - - - - - -
4 X - - - X - - - - - 4 - - - - - - - - -
5 - - X - - - X X - - 5 - - - - - - - - -
6 - - X - - - - - - - 6 - - - - - - - - -
7 - - X - X - - - - - 7 - - - - - - - - -
8 - - - - - - - - - - 8 - - - - - - - - -
9 - X - - - - X X - - 9 - - - - - - - - -
10 - - - X - - - - - - 10 - - - - - - - - -

=====

Your turn
Enter your shot
Enter x coordinate:
```

- Players selecting options 2 or 3 get:
 - Ready-to-play naval formations
 - Automatically generated ship positions
 - Immediate transition to gameplay

Player Boards & Turn-Based Gameplay

```
C:\Windows\System32\cmd.exe - py client.py
=====
Your board      Your shots
 1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
1 - - - X X X - - - - 1 - - - - - - - - -
2 - - - - - - - X - 2 - - - - - - - - -
3 - - - - - - - X - 3 - - - - - - - - -
4 - - X X - - - X - 4 - - - - - - - - -
5 - - - - X X - - - 5 - - - - - - - - -
6 X - - - - - - - - 6 - - - - - - - - -
7 X - X - X - - X - 7 - - - - - - - - -
8 X - - - - - X - - 8 - - - - - - - - -
9 X - - - - - - - - 9 - - - - - - - - -
10 - - X - - - X - - 10 - - - - - - - - -
=====

Enemy's turn...
-

C:\Windows\System32\cmd.exe - py client.py
=====
Your board      Your shots
 1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
1 X - X - - - - - - 1 - - - - - - - - -
2 X - X - - X X - - - 2 - - - - - - - - -
3 X - X - - - - - - - 3 - - - - - - - - -
4 X - - - X - - - - - 4 - - - - - - - - -
5 - - X - - - X X - - 5 - - - - - - - - -
6 - - X - - - - - - - 6 - - - - - - - - -
7 - - X - X - - - - - 7 - - - - - - - - -
8 - - - - - - - - - 8 - - - - - - - - -
9 - X - - - - X X - - 9 - - - - - - - - -
10 - - X - - - - - - 10 - - - - - - - - -
=====

Your turn
Enter your shot
Enter x coordinate:
```

- Each player has their own board to place ships (X).
- Players take turns attacking coordinates.
- Hits are marked as H, misses as M.
- Turn continues until a miss occurs.
- First to sink all enemy ships wins

Turn-Based Combat & Hit Feedback

```
C:\Windows\System32\cmd.exe - py server.py
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Khalil\Desktop\Lm\distr\ Battleship-Game\Server>py server.py
Server UDP is active
Waiting for players...
New player joined: ('127.0.0.1', 60517)
Created new room(62db7175-bd9c-4c62-bc6f-deda6861962d) and added ('127.0.0.1', 60517) as player 1
New player joined: ('127.0.0.1', 60518)
Added ('127.0.0.1', 60518) as player 2 to room 62db7175-bd9c-4c62-bc6f-deda6861962d
Room 62db7175-bd9c-4c62-bc6f-deda6861962d | GAME STARTS
Room 62db7175-bd9c-4c62-bc6f-deda6861962d | Player that begins: ('127.0.0.1', 60518)
Room 62db7175-bd9c-4c62-bc6f-deda6861962d | Player ('127.0.0.1', 60518) shoots on coordinates: ['4', '1']

Room 62db7175-bd9c-4c62-bc6f-deda6861962d | Sending shoot verification to: ('127.0.0.1', 60517)
Room 62db7175-bd9c-4c62-bc6f-deda6861962d | Player ('127.0.0.1', 60517) verifies shoot as: True
Room 62db7175-bd9c-4c62-bc6f-deda6861962d | Updating boards ('127.0.0.1', 60518)

C:\Windows\System32\cmd.exe - py client.py
=====
Shoot on ['4', '1'] is a HIT!
=====

Your board
 1 2 3 4 5 6 7 8 9 10
1 X - X - - - - -
2 X - X - - X X - -
3 X - X - - - - -
4 X - - - X - - - -
5 - - X - - - X X -
6 - - X - - - - -
7 - - X - X - - - -
8 - - - - - - - -
9 - X - - - - X X -
10 - - - X - - - -

Your shots
 1 2 3 4 5 6 7 8 9 10
1 - - - H - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - - - - - -
5 - - - - - - - -
6 - - - - - - - -
7 - - - - - - - -
8 - - - - - - - -
9 - - - - - - - -
10 - - - - - - -

=====

Your turn
Enter your shot
Enter x coordinate:
```

- Player hits opponent at ['4', '1'] (marked as H)
- Turn continues until a miss occurs.
- Server updates both players' boards in real time.
- Left grid: Player's ships (X = intact, H = hit).
- Right grid: Attack history (H = hit).
- Goal: Sink all enemy ships first to win.

Timeout Penalty - Automatic Loss

```
C:\Windows\System32\cmd.exe - python client.py

=====

Your board      Your shots
  1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
  1 - - - X X X - - - -  1 - - - - - - - - -
  2 - - - - - - - - X -  2 - - - - - - - - -
  3 - - - - - - - - X -  3 - - - - - - - - -
  4 - - X X - - - - X -  4 - - - - - - - - -
  5 - - - - - X X - - -  5 - - - - - - - - -
  6 X - - - - - - - - -  6 - - - - - - - - -
  7 X - X - X - - - X - -  7 - - - - - - - - -
  8 X - - - - - - X - -  8 - - - - - - - - -
  9 X - - - - - - - - -  9 - - - - - - - - -
 10 - - X - - - X - - -  10 - - - - - - - - -

=====

Board setup complete! Waiting for opponent to be ready...
Both players ready! GAME BEGINS
Your turn
Enter your shot (you have 30 seconds)
Enter x coordinate: Time's up! You took too long to enter coordinates.
You took too long to make your move. You lose by timeout!
Do you want to play again?
_
```

- Player failed to input move within 30 seconds
- System message: "You took too long! You lose by timeout"
- Game offers replay option (Y/N)

```
C:\Windows\System32\cmd.exe - python client.py

=====

Your board      Your shots
  1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
  1 X - X - - - - -  1 - - - - - - - - -
  2 X - X - - X X - - -  2 - - - - - - - - -
  3 X - X - - - - - -  3 - - - - - - - - -
  4 X - - - X - - - - -  4 - - - - - - - - -
  5 - - X - - X X - - -  5 - - - - - - - - -
  6 - - X - - - - - -  6 - - - - - - - - -
  7 - - X - X - - - - -  7 - - - - - - - - -
  8 - - - - - - - - -  8 - - - - - - - - -
  9 - X - - - X X - - -  9 - - - - - - - - -
 10 - - - X - - - - -  10 - - - - - - - - -

=====

Board setup complete! Waiting for opponent to be ready...
Both players ready! GAME BEGINS
Enemy's turn...
Your opponent has lost by time. You win!
```

- Opponent failed to respond in time
- Victory message: "Opponent lost by time. You win!"
- Shows player's intact board and unused shots grid
- Prompts for rematch (Y/N)

Mid-Game Disconnection - Automatic Win

```
C:\Windows\System32\cmd.exe - python client.py

=====
Your board          Your shots
  1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
  1 - - - X X X - - - -  1 - - - - - - - - - -
  2 - - - - - - - - X -  2 - - - - - - - - - -
  3 - - - - - - - - X -  3 - - - - - - - - - -
  4 - - X X - - - - X -  4 - - - - - - - - - -
  5 - - - - - X X - - -  5 - - - - - - - - - -
  6 X - - - - - - - - -  6 - - - - - - - - - -
  7 X - X - X - - X - -  7 - - - - - - - - - -
  8 X - - - - - - X - -  8 - - - - - - - - - -
  9 X - - - - - - - - -  9 - - - - - - - - - -
 10 - - X - - - X - - - 10 - - - - - - - - - -
=====

Board setup complete! Waiting for opponent to be ready...
Both players ready! GAME BEGINS
Enemy's turn...
Your opponent has disconnected during the game! You win by default.
Do you want to play again?
Enter Y or N: _
```

- Opponent disconnects during active gameplay
- Clear victory message: "Your opponent has disconnected! You win by default."
- Shows player's current board state and shot history
- Prompts for rematch (Y/N)
- Server detects disconnect within 5-15 seconds via heartbeat

Pre-Game Disconnection - Waiting Reset

```
C:\Windows\System32\cmd.exe - python client.py

=====
Your board      Your shots
 1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
1 X - X - - - - - - 1 - - - - - - - - -
2 X - X - - X X - - - 2 - - - - - - - -
3 X - X - - - - - - 3 - - - - - - - -
4 X - - - X - - - - - 4 - - - - - - - -
5 - - X - - - X X - - 5 - - - - - - - -
6 - - X - - - - - - 6 - - - - - - - -
7 - - X - X - - - - - 7 - - - - - - - -
8 - - - - - - - - - 8 - - - - - - - -
9 - X - - - - X X - - 9 - - - - - - - -
10 - - - X - - - - - 10 - - - - - - - -
=====

Board setup complete! Waiting for opponent to be ready...
Your opponent has disconnected! You will be returned to the waiting room...
Connected to server, waiting for the opponent...
My address: ('127.0.0.1', 55567)
```

- Opponent disconnects during board setup phase
- Non-punitive message: "Opponent disconnected! Returning to waiting room"
- Preserves player's configured board
- Maintains connection to server for new opponent
- Shows seamless reconnection process

Server-Side Disconnection Handling

```
Created new room(b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17) and added ('127.0.0.1', 63352) as player 1
New player joined: ('127.0.0.1', 55449)
Added ('127.0.0.1', 55449) as player 2 to room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | Both players connected, preparing game
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | Player 2 is ready
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | Player 1 is ready
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | GAME STARTS
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | Player that begins: ('127.0.0.1', 55449)
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 | Player 2 disconnected
Game in room $b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 ended
Room b9d74fc9-f75f-4d5a-bbe3-fd50687a2d17 removed due to player ('127.0.0.1', 55449) disconnection
-
```

- Detects and processes disconnection in real-time
- Automatically cleans up game room

Server Shutdown - Connection Lost

```
C:\Windows\System32\cmd.exe

=====

Your board                Your shots
 1 2 3 4 5 6 7 8 9 10    1 2 3 4 5 6 7 8 9 10
1 - - - X X X - - - - 1 - - - - - - - - -
2 - - - - - - - - X - 2 - - - - - - - - -
3 - - - - - - - - X - 3 - - - - - - - - -
4 - - X X - - - - X - 4 - - - - - - - - -
5 - - - - - X X - - - 5 - - - - - - - - -
6 X - - - - - - - - - 6 - - - - - - - - -
7 X - X - X - - X - - 7 - - - - - - - - -
8 X - - - - - - X - - 8 - - - - - - - - -
9 X - - - - - - - - - 9 - - - - - - - - -
10 - - X - - - X - - - 10 - - - - - - - - -

=====

Board setup complete! Waiting for opponent to be ready...
Both players ready! GAME BEGINS
Enemy's turn...
Connection lost during gameplay. Server might be unavailable. Exiting...

C:\Users\Khalil\Desktop\Lm\ Battleship-Game\ Client>
```

- **Server Crash/Shutdown:** Shows player's perspective when server becomes unavailable mid-game.
- **Clean Exit:** Displays warning: "Connection lost. Server might be unavailable. Exiting..."
- **Automatic Closure:** Client safely terminates after detecting server disconnect.

Victory Screen & Game Conclusion

```
C:\Windows\System32\cmd.exe - py client.py
=====
Shoot on ['4', '10'] is a HIT!
=====

Your board
 1 2 3 4 5 6 7 8 9 10
1 M - - X X X - - -
2 - - - - - X -
2 - - - - - X -
4 - - X X - - - X -
5 - - - - X X - -
6 X - - - - - -
7 X - X - X - - X -
8 X - - - - - X -
6 X - - - - - -
10 - - X - - X - -

Your shots
 1 2 3 4 5 6 7 8 9 10
1 H - H - - - - -
2 H M H - - H H - -
2 H - H - - - - -
4 H - - - H - - -
5 - - H - - H H - -
6 - - H - - - - -
7 - - H - H - - -
8 - - - - - - -
6 - H - - - H H - -
10 - - - H - - - -

=====
Your turn
YOU WON!!!
Do you want to play again?
Enter Y or N:

C:\Windows\System32\cmd.exe - py client.py
^ ENEMY HIT YOUR SHIP
=====

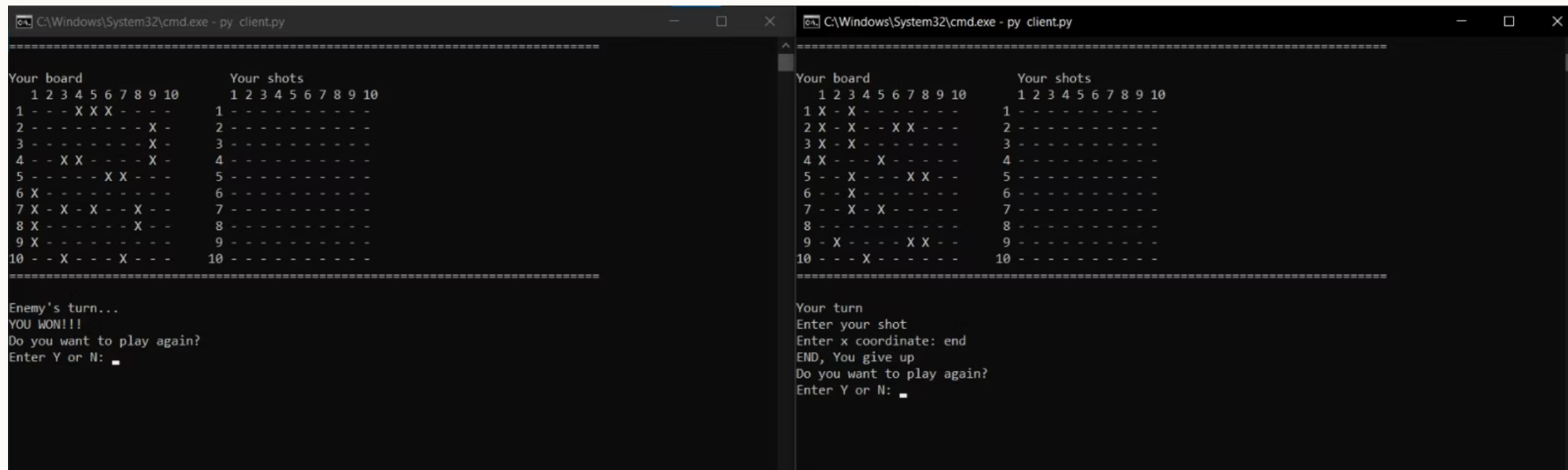
Your board
 1 2 3 4 5 6 7 8 9 10
1 H - H - - - - -
2 H M H - - H H - -
1 H - H - - - - -
4 H - - - H - - -
5 - - H - - H H - -
6 - - H - - - - -
7 - - H - H - - -
8 - H - - - H H - -
9 - H - - - - -
10 - - - H - - - -

Your shots
 1 2 3 4 5 6 7 8 9 10
1 M - - - - - -
2 - - - - - -
1 - - - - - -
4 - - - - - -
5 - - - - - -
6 - - - - - -
7 - - - - - -
8 - - - - - -
9 - - - - - -
10 - - - - - -

=====
Enemy shoots on: 4 , 10
GAME ENDED, DEFEAT
Do you want to play again?
Enter Y or N: 
```

- Winning player sees: "YOU WON!!!"
- Losing player sees: "GAME ENDED DEFEAT"
- Prompts players to replay: "Enter Y/N to play again"
- Cleanly ends or restarts the game based on choice

Surrender & Game End



The image shows two side-by-side terminal windows running a Python script. The left window displays the game board and shots after the player's turn. The right window shows the game board and shots after the enemy's turn, where the player has entered 'end' to surrender.

```
=====
Your board      Your shots
 1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
1 - - - X X X - - - 1 - - - - - - - - -
2 - - - - - - - X - 2 - - - - - - - - -
3 - - - - - - - X - 3 - - - - - - - - -
4 - - X X - - - - X - 4 - - - - - - - - -
5 - - - - - X X - - 5 - - - - - - - - -
6 X - - - - - - - - 6 - - - - - - - - -
7 X - X - X - - X - - 7 - - - - - - - - -
8 X - - - - - X - - 8 - - - - - - - - -
9 X - - - - - - - - 9 - - - - - - - - -
10 - - X - - - X - - 10 - - - - - - - - -
=====

Enemy's turn...
YOU WON!!!
Do you want to play again?
Enter Y or N: _

=====
Your board      Your shots
 1 2 3 4 5 6 7 8 9 10  1 2 3 4 5 6 7 8 9 10
1 X - X - - - - - - 1 - - - - - - - - -
2 X - X - - X X - - 2 - - - - - - - - -
3 X - X - - - - - - 3 - - - - - - - - -
4 X - - - X - - - - 4 - - - - - - - - -
5 - - X - - - X X - - 5 - - - - - - - - -
6 - - X - - - - - - 6 - - - - - - - - -
7 - - X - X - - - - 7 - - - - - - - - -
8 - - - - - - - - - 8 - - - - - - - - -
9 - X - - - X X - - 9 - - - - - - - - -
10 - - X - - - - - - 10 - - - - - - - - -
=====

Your turn
Enter your shot
Enter x coordinate: end
END, You give up
Do you want to play again?
Enter Y or N: _
```

- Player can surrender by typing **"END"**.
- Message displayed: **"You give up!"** (for quitter).
- Opponent sees: **"You win!"** (automatic victory).
- Prompts both players: **"Play again? (Y/N)"**

Conclusion & Future Work

Achievements

Successfully built a stable, responsive multiplayer game using UDP

Educational Value

Practical application of networking concepts and distributed programming



Lessons Learned

Gained experience with distributed systems, real-time communication, and state management

Future Improvements

Graphical interface, persistent storage, multi-room support