CN MINI PROJECT - TIC TAC TOE

NAME: N V MANYA NAME: NAVYASHREE J

SRN: PES1UG23AM176 SRN: PER1UG23AM180

SERVER CODE:

```
import socket, threading, ssl
import traceback


class GameRoom:
    def _init_(self, room_id):
        self.room_id = room_id
        self.board = [" "] * 9
        self.players = {}
        self.player_names = {}
        self.scores = {"X": 0, "O": 0}
        self.current_player = "X"
        self.game_active = False
        self.lock = threading.Lock()
        self.play_again_responses = {}


    def add_player(self, player_socket):
        with self.lock:
            symbol = "X" if "X" not in self.players else "O"
            self.players[symbol] = player_socket


            player_socket.send("Enter your name: ".encode())
            name = player_socket.recv(1024).decode().strip()
            self.player_names[symbol] = name
            self.scores[symbol] = 0
```

```
        player_socket.send(f"You are Player {symbol} ({name}) in Room {self.room_id}\n".encode())


        if len(self.players) == 2:
            self.reset_board()
            self.game_active = True
            self.broadcast("\n🎮 Both players connected! Game is starting...\n")
            self.send_leaderboard()
            self.broadcast(self.print_board())
            self.players[self.current_player].send("Your turn! Enter position (0-8): ".encode())
        return symbol


def broadcast(self, message, exclude=None):
    for s, sock in self.players.items():
        if sock != exclude:
            try:
                sock.send(message.encode())
            except:
                pass


def print_board(self):
    b = self.board
    return (
        f"\n {b[0]} | {b[1]} | {b[2]} \n"
        "---+---+---\n"
        f" {b[3]} | {b[4]} | {b[5]} \n"
        "---+---+---\n"
        f" {b[6]} | {b[7]} | {b[8]} \n"
    )


def check_winner(self):
    wins = [(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]
```

```
        for a, b, c in wins:
            if self.board[a] == self.board[b] == self.board[c] != " ":
                return self.board[a]
        return None

    def reset_board(self):
        self.board = [" "] * 9
        self.current_player = "X"

    def reset_game(self):
        self.reset_board()
        self.game_active = True
        self.play_again_responses.clear()
        self.broadcast("\n🔄 New round starting...\n")
        self.send_leaderboard()
        self.broadcast(self.print_board())
        self.players[self.current_player].send("Your turn! Enter position (0-8): ".encode())

    def send_leaderboard(self):
        leaderboard = "\n🏆 Leaderboard:\n"
        for symbol in ["X", "O"]:
            name = self.player_names.get(symbol, "Unknown")
            score = self.scores.get(symbol, 0)
            leaderboard += f"Player {symbol} ({name}): {score} points\n"
        self.broadcast(leaderboard)

class GameServer:
    def _init_(self):
        self.rooms = {}
        self.room_counter = 1
        self.lock = threading.Lock()
```

```
def start(self, host='192.168.134.188', port=5555):
    context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
    context.check_hostname = False
    context.verify_mode = ssl.CERT_NONE
    context.load_cert_chain('cert.pem', 'key.pem')

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"[] Server started at {host}:{port} with SSL")

    while True:
        client, addr = server_socket.accept()
        secure_socket = context.wrap_socket(client, server_side=True)
        print(f"[+] Connection from {addr}")
        threading.Thread(target=self.handle_client, args=(secure_socket,)).start()

def handle_client(self, client_socket):
    room = None
    symbol = None

    try:
        with self.lock:
            for r in self.rooms.values():
                if len(r.players) == 1 and not r.game_active:
                    room = r
                    break
            if not room:
                room = GameRoom(self.room_counter)
```

```
        self.rooms[self.room_counter] = room
        self.room_counter += 1
symbol = room.add_player(client_socket)
while True:
    data = client_socket.recv(1024).decode().strip()
    print(f"[{symbol}] Received input: {data}")  # Debug log

    if not data:
        break

    with room.lock:
        if not room.game_active:
            if data.lower() in ["yes", "no"]:
                room.play_again_responses[symbol] = data.lower()

                if len(room.play_again_responses) == 2:
                    if all(v == "yes" for v in room.play_again_responses.values()):
                        room.reset_game()
                    else:
                        for s, response in room.play_again_responses.items():
                            if response == "no":
                                try:
                                    room.players[s].send("👋 You chose to exit.\n".encode())
                                    room.players[s].close()
                                    del room.players[s]
                                except:
                                    pass
                        room.play_again_responses.clear()
                        room.game_active = False
                        room.broadcast("🕓 Waiting for a new player to join...\n")
            else:
```

```
                client_socket.send("Please respond with 'yes' or 'no': ".encode())
            continue

        # ✅ VALIDATION: Ensure move is a number between 0 and 8
        try:
            move = int(data)
            if move < 0 or move > 8:
                client_socket.send("❌ Invalid input. Please enter a number between 0 and 8.\nYour turn! Enter position (0-8): ".encode())
                continue
        except ValueError:
            client_socket.send("❌ Invalid input. Please enter a number between 0 and 8.\nYour turn! Enter position (0-8): ".encode())
            continue

        if room.board[move] != " ":
            client_socket.send("❌ That cell is already taken. Try another.\nYour turn! Enter position (0-8): ".encode())
        elif symbol != room.current_player:
            client_socket.send("❌ Not your turn! Wait for the other player.\n".encode())
        else:
            room.board[move] = symbol
            winner = room.check_winner()
            if winner:
                room.broadcast(room.print_board())
                room.broadcast(f"\n🏆 Player {winner} wins!\n")
                room.scores[winner] += 1
                room.game_active = False
                room.send_leaderboard()
                room.broadcast("🔁 Play again? (yes/no): ")
            elif " " not in room.board:
                room.broadcast(room.print_board())
                room.broadcast("\n🤝 It's a draw!\n")
```

```
                    room.game_active = False
                    room.broadcast("🔁 Play again? (yes/no): ")
                else:
                    room.current_player = "O" if room.current_player == "X" else "X"
                    room.broadcast(room.print_board())
                    room.players[room.current_player].send("Your turn! Enter position (0-8): ".encode())

        except Exception as e:
            print(f"[⚠] Error: {e}")
            traceback.print_exc()
        finally:
            with self.lock:
                if room and symbol in room.players:
                    del room.players[symbol]
                if room and not room.players:
                    del self.rooms[room.room_id]
            client_socket.close()
if _name_ == "_main_":
    GameServer().start()
```

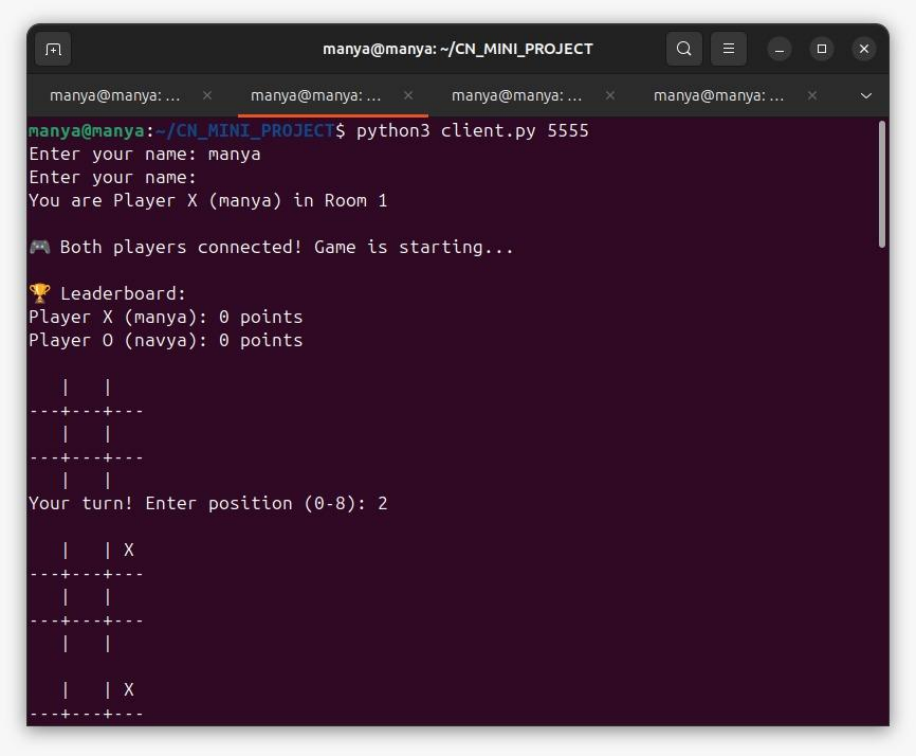
CLIENT CODE:

```
import socket, ssl

host = '192.168.134.188'  # Change if server IP is different
port = 5555
context = ssl.create_default_context()
context.check_hostname = False
context.verify_mode = ssl.CERT_NONE

with socket.create_connection((host, port)) as sock:
    with context.wrap_socket(sock, server_hostname=host) as ssock:
```

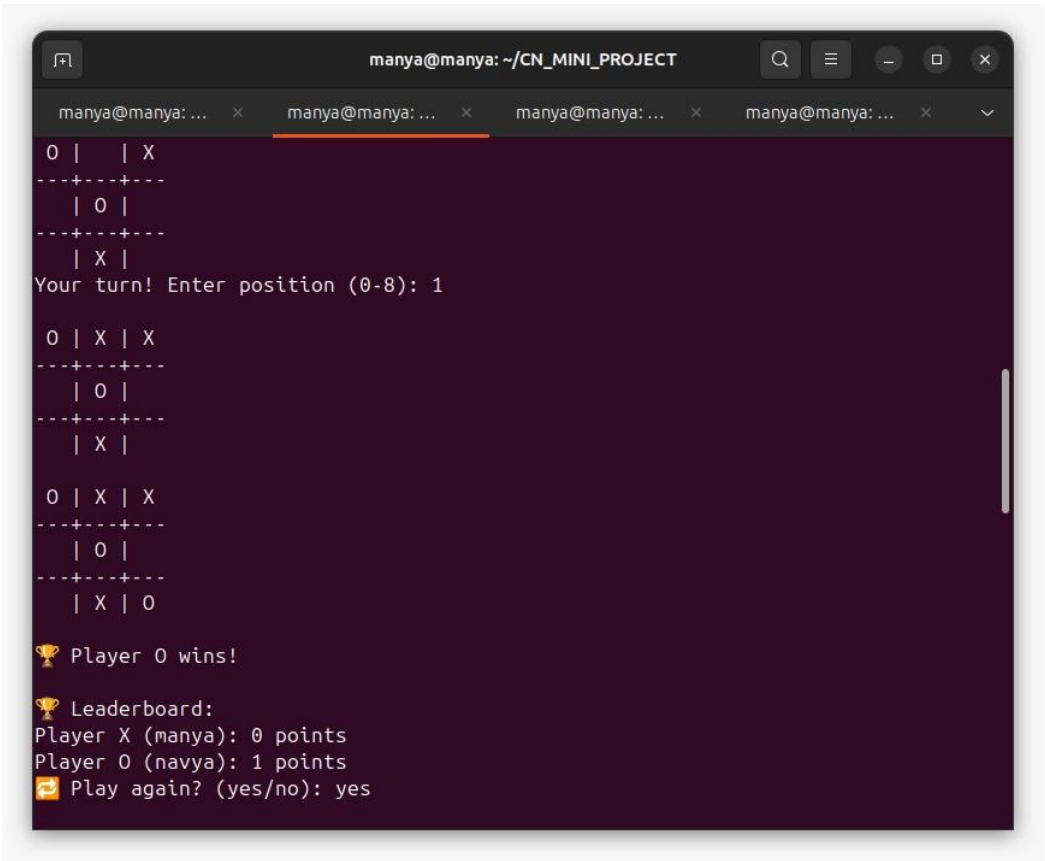
```
name = input("Enter your name: ").strip()
ssock.send(name.encode())
while True:
    data = ssock.recv(4096).decode()
    if not data:
        break
    print(data, end="")
    if any(prompt in data.lower() for prompt in ["your turn", "yes/no", "enter your name"]):
        msg = input().strip()
        ssock.send(msg.encode())
```

OUTPUT:

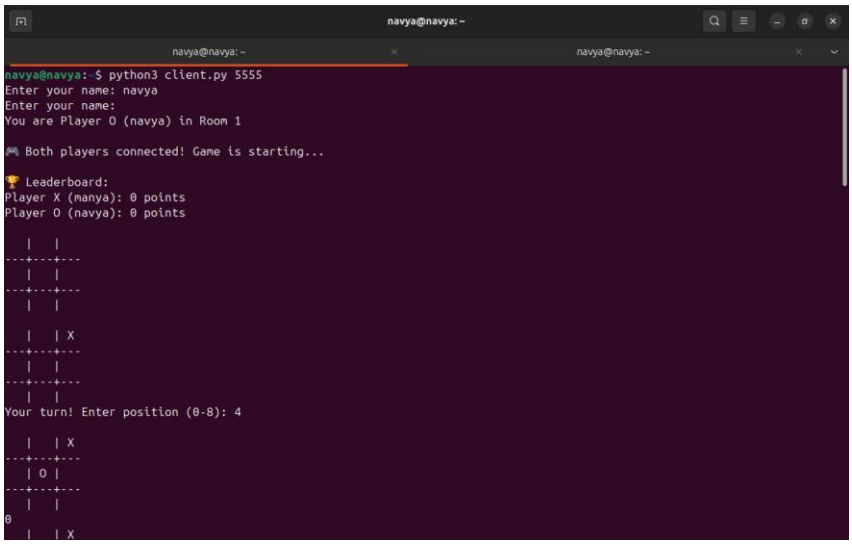
CLIENT1:

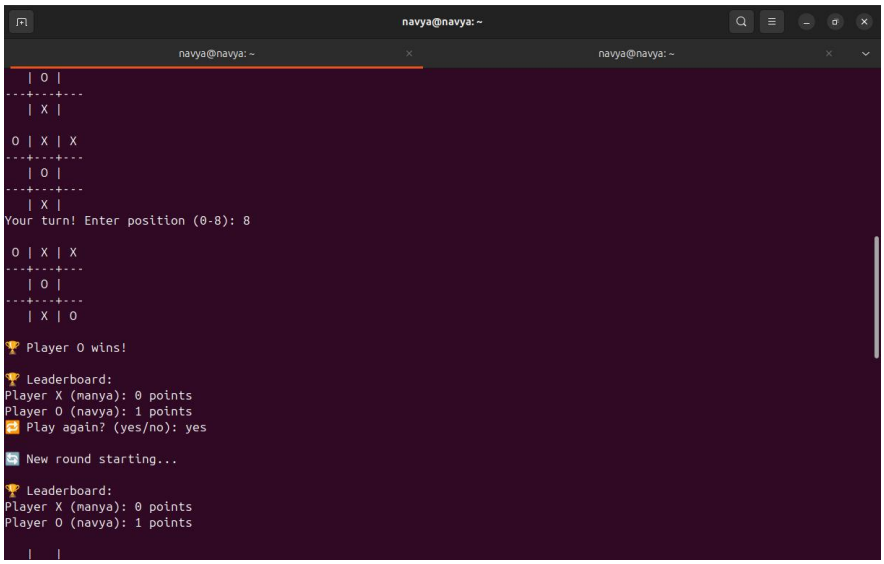


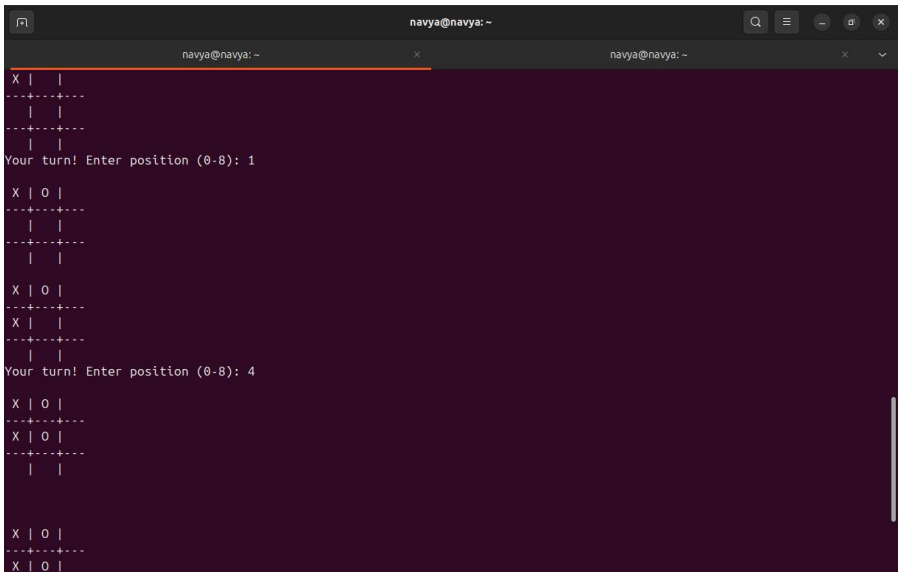

manya@manya: ~/CN_MINI_PROJECT
manya@manya: ...
manya@manya: ...
manya@manya: ...
manya@manya: ...
---+---+---
| |
| | X
---+---+---
| O |
---+---+---
| |
Your turn! Enter position (0-8): 2
❌ That cell is already taken. Try another.
Your turn! Enter position (0-8): 7
| | X
---+---+---
| O |
---+---+---
| X |
O | | X
---+---+---
| O |
---+---+---
| X |
Your turn! Enter position (0-8): 1
O | X | X

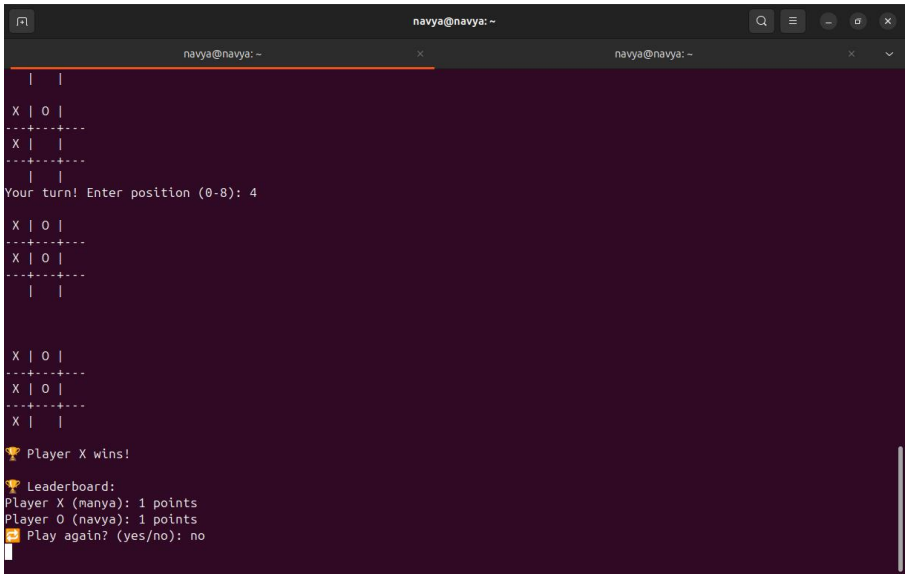

manya@manya: ~/CN_MINI_PROJECT
manya@manya: ...
manya@manya: ...
manya@manya: ...
manya@manya: ...
O | | X
---+---+---
| O |
---+---+---
| X |
Your turn! Enter position (0-8): 1
O | X | X
---+---+---
| O |
---+---+---
| X |
O | X | X
---+---+---
| O |
---+---+---
| X | O
🏆 Player O wins!
🏆 Leaderboard:
Player X (manya): 0 points
Player O (navya): 1 points
🔁 Play again? (yes/no): yes

CLIENT2;

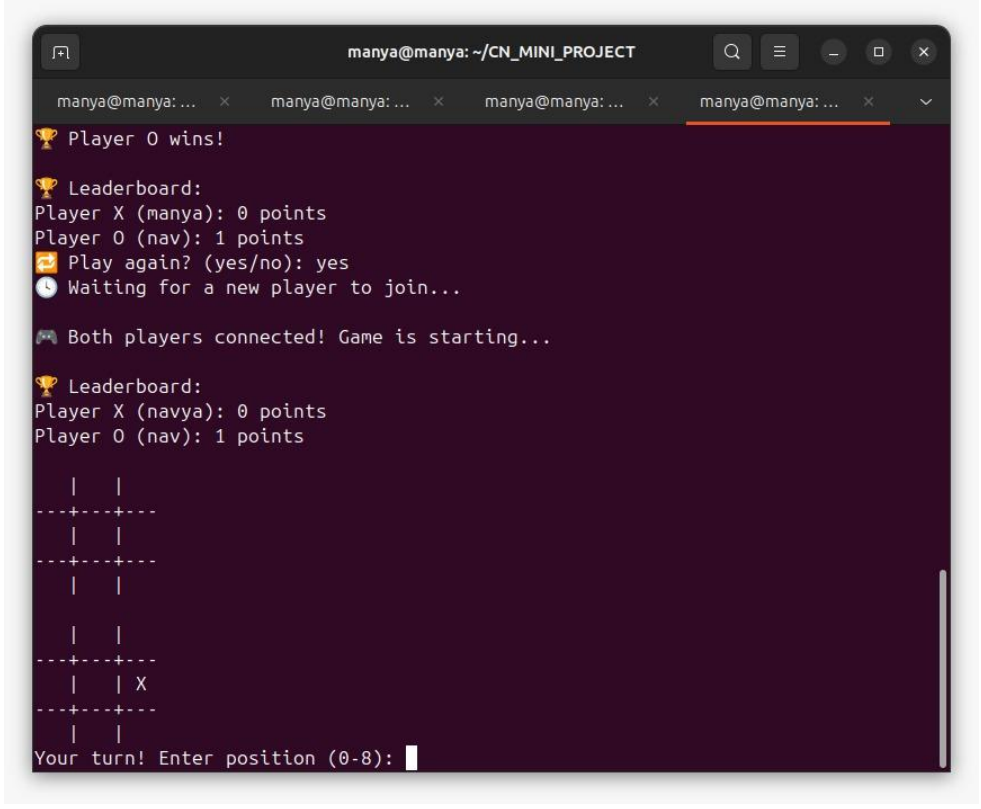

navya@navya: ~
navya@navya: ~
navya@navya: ~
navya@navya: $ python3 client.py 5555
Enter your name: navya
Enter your name:
You are Player O (navya) in Room 1
🎮 Both players connected! Game is starting...
🏆 Leaderboard:
Player X (manya): 0 points
Player O (navya): 0 points
| |
---+---+---
| |
---+---+---
| |
| | X
---+---+---
| |
---+---+---
| |
Your turn! Enter position (0-8): 4
| | X
---+---+---
| O |
---+---+---
| |
0
| | X


navya@navya: ~
navya@navya: ~
navya@navya: ~
| O |
---+---+---
| X |
O | X | X
---+---+---
| O |
---+---+---
| X |
Your turn! Enter position (0-8): 8
O | X | X
---+---+---
| O |
---+---+---
| X | O
🏆 Player O wins!
🏆 Leaderboard:
Player X (manya): 0 points
Player O (navya): 1 points
🔁 Play again? (yes/no): yes
🔄 New round starting...
🏆 Leaderboard:
Player X (manya): 0 points
Player O (navya): 1 points
| |


navya@navya: ~
navya@navya: ~
navya@navya: ~
X | |
---+---+---
| |
---+---+---
| |
Your turn! Enter position (0-8): 1
X | O |
---+---+---
| |
---+---+---
| |
X | O |
---+---+---
X | |
---+---+---
| |
Your turn! Enter position (0-8): 4
X | O |
---+---+---
X | O |
---+---+---
| |
X | O |
---+---+---
X | O |



CLIENT3:



HOW IT WORKS:

1. Player Connection & Room Management

When a client connects, the server checks if there’s a waiting room.

If not, a brand-new game room is created just for them.

Each room supports exactly two players: Player X and Player O.

2. Name Input & Welcome

Each player is asked to enter their name.

They are then told which player they are (X or O) and which room they're in.

3. Game Start

Once two players are connected:

The game board is reset.

A message is broadcast to both players that the game is starting.

The board is displayed, and Player X goes first.

4. Game Turns & Logic

Players take turns choosing a cell (0–8) to place their symbol.

The server checks:

If it's a valid move.

If that spot is already taken.

If the move results in a win or draw.

5. Win or Draw Detection

If a player wins:

The board is shown.

A cute message declares the winner.

Scores are updated.

A leaderboard is shown.

Players are asked if they want to play again.

If it's a draw:

A draw message is shown.

Players are still asked about playing again.

6. Play Again Logic

If both players say "yes", the board resets and a new round begins.

If any player says "no", they are disconnected, and the room waits for a new player.

7. Clean Exit

If a player disconnects or leaves:

They are removed from the room.

If both players leave, the room is deleted.

FEATURES:

🔄 Real-Time Multiplayer	2 players compete interactively
🛡 SSL Encryption	Secure connection with certificates
🎭 Player Identity	Name + Symbol (X or O)
🎮 Turn-Based Gameplay	Prompts and checks for valid moves
🏁 Win/Draw Detection	Classic Tic-Tac-Toe logic
📈 Scoreboard + Leaderboard	Tracks wins for each player
🔁 Replay Option	Ask if players want another round
🧹 Room Cleanup	Frees resources when players leave