**XYOT EVENTS SUMMARY**

**Note**: Since the server code is in NodeJS, so the data passed back and forth is expected to be JSON/javascript object literal for any communication made to the server.

{

prop1: value1,

prop2: value2,

} .

**Events to emit (public room)**

**1. join\_game**

**Description**: “joins a public room according to the gameType(3, 4, 5)”

*socket.emit(****‘join\_game’****, mainPlayer, gameType)*

**2. play\_turn**

**Description**: Should be emitted when the player makes the move along with the player details and index of the grid on the board.

Socket.emit(‘play\_turn’, mainPlayer, gridIndex)

**Events to listen (public room)**

**1. player\_registered**

**Description**: “fired when server creates and assigns room to a player successfully”

socket.on(‘player\_registered’, player => {

})

**2. player\_joined**

**Description**: Fired when any player joins the room of the main player.

Socket.on(**‘player\_joined’**, (players, playerJoined) => {

})

**3. game\_started**

**Description:** Fired when all the players have joined the player’s room and game has started + timers.

Socket.on(**‘game\_started’**, room => {

})

**4. turn**

**Description:** Broadcasted with the current player’s turn to everyone in the room.

socket.on('turn', playerTurn => {

    // it will receive the player with that given turn...

    if (playerTurn.socketId === mainPlayer.socketId && playerTurn.turn) {

        mainPlayerTurn = true

        statusDisplay.innerHTML = 'Your turn...'

    } else {

        mainPlayerTurn = false

        statusDisplay.innerHTML = `Waiting for ${playerTurn.name}'s turn...`

    }

})

**5. turn\_played**

**Description**: Broadcasted to all players when someone makes a move.

Socket.on(‘turn\_played’, (player, gridIndex) => {

})

**6. game\_won**

**Description**: Server broadcasts this game won event when the game is won by a player.

Socket.on(‘game\_won’, winnerPlayer => {

})

**7. game\_draw**

**Description:** Similar to game won event but broadcasted when the game is a draw.

Socket.on(‘game\_draw’, () => {

})

**8. game\_over**

Listen to the game over event to detect if the game is over.

socket.on('game\_over', (result, player) => {

    statusDisplay.textContent = `Game Over. ${result.message}`

    gameOver = true

})

**9. timeout**

**Description**: Runs every second on every tick of the timer.

Socket.on(‘timeout’, time => {

// time is 0-10 seconds

}

**10. player\_left**

**Description:** Broadcasted when a player leaves the game.

Socket.on(‘player\_left’, playerWhoLeft => {

}

**EVENTS TO EMIT FROM THE CLIENT SIDE (PRIVATE ROOM)**

**1. create\_room**

**Description:** Emit this event from the client when a new room needs to be created.

Socket.emit(‘create\_room’, mainPlayer, gameType)

**2. join\_room**

**Description:** Emit this event from the client when the player wants to join a private room using the room code.

Socket.emit(‘join\_room’, mainPlayer, roomCode)

**Events to listen on the client side (private room)**

**1. room\_created**

**Description:** Once the server creates the room, this event will run. You can listen to it similar to this:

Socket.on(‘room\_created’, (player, roomCode) => {

})

**2. room\_joined**

**Description:** If a player joined the room successfully, then the room joined event is fired.

Socket.on(‘room\_joined, (player, roomId, room) => {

})

**Room API**

**Description**: Get a room by room code from server. Use only for checking if the room exists in joining room. Room code is the encoded version of actual roomId.

GET <http://socketserver.xyot4.com:3000/rooms/:>roomCode

On Success:

Status: 200

It will return a json response:

{

**message**: ‘room found’,

**room**: { … the room object with that roomId }

}

On Failure:

Status: 404/400

{

**message**: ‘room not found’

}