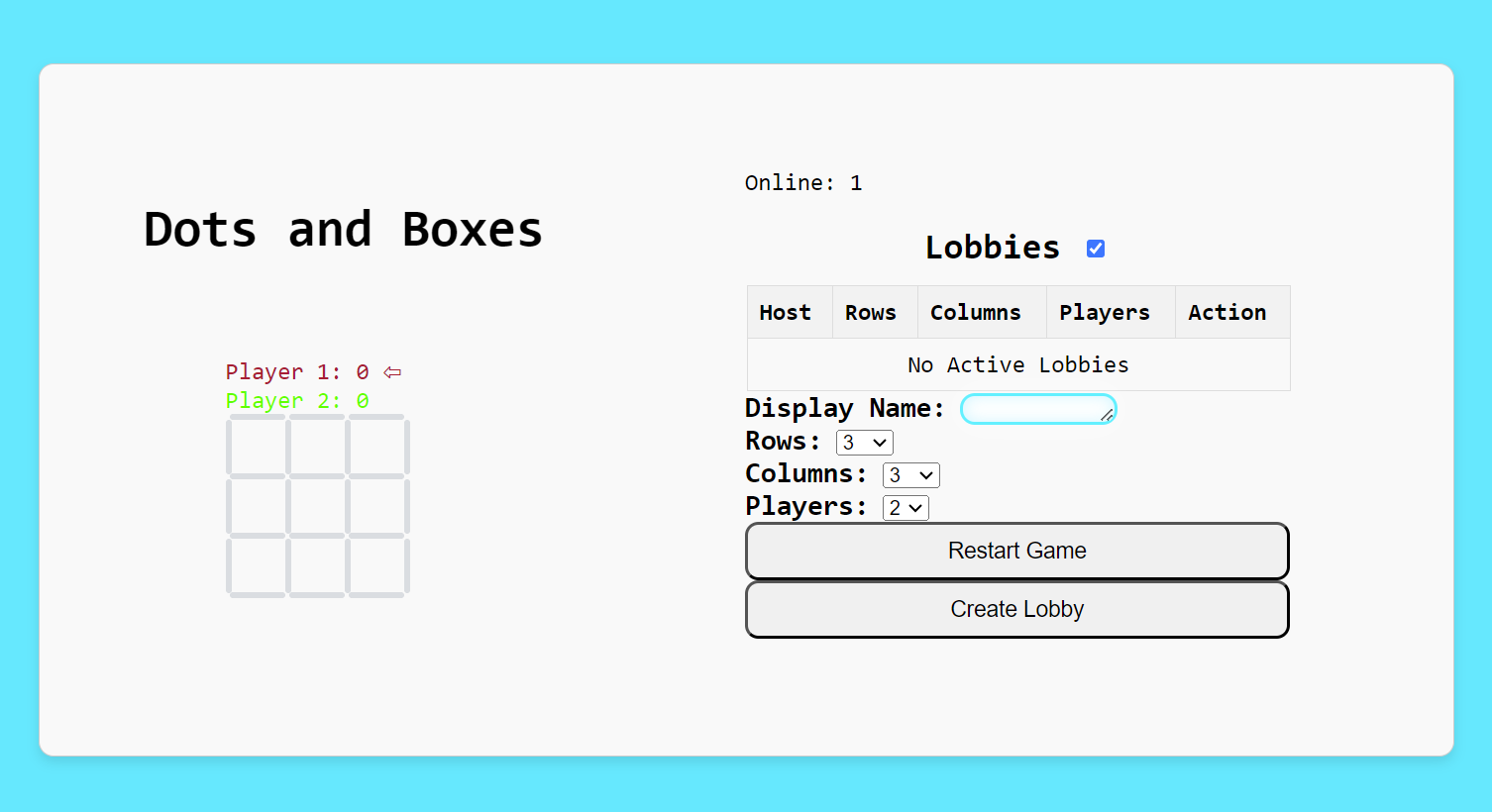
**Dots and Boxes Game**

Screen shot : 

**Installation**

The project is designed with html/css and javascript , as well as node.js and socket.io.   
All on the tool , visual studio code .

You can :

Download **Node.js:** Go to [nodejs.org](https://nodejs.org/) and download the installer for your operating system. Choose the LTS version (Long Term Support) for stability unless you have a specific reason for needing the latest features.

Run **the installer:** Follow the prompts in the installer to complete the installation. This will also install npm (Node Package Manager) automatically.

Verify **installation:** Open a terminal or command prompt and type:

node -v

npm -v

Once you initialize a node js project and you initialise the npm in the directory package JSON you can install socket.io, using the following: npm install socket.io

**Starting the Game**

This code implements a grid-based game known as Dots and boxes where players take turns to draw lines between dots on a grid, aiming to complete squares. The game can be played locally or online with up to 5 players.

You can start it by running ‘node server.js’ on the terminal of Visual Studio code and clicking the link formed off the port created. Alternatively, you can begin on the web by running ‘index.html’.

**Key Functions**

**Server-Side Implementation**

The server-side code is written in Node.js and Socket.IO. It manages game state, player connections, and real-time updates.

**Key Functions and Event Handlers**

* **Connection Handling**: Manages new connections, updates online user count, and broadcasts changes.
* **Lobby Management**: Functions to add, remove, join, and leave lobbies, ensuring synchronization across all clients.
* **Game Logic**: Handles moves, updates the game matrix, checks for completed cells, and manages game state transitions (e.g., turns, points).
* **Real-Time Communication**: Uses Socket.IO to emit events for updating lobbies, starting games, making moves, and handling disconnections.

**Example Functions**

* **deleteEmptyLobbies()**: Removes empty or abandoned lobbies.
* **leaveMatch()**: Handles a player leaving a match, updating the lobby and notifying other players.
* **rematch()**: Initiates a rematch if all players agree.
* **getLiveLobby()**: Retrieves the live lobby a player is part of.
* **updateMatrix()**: Updates the game matrix based on player moves and checks for completed cells.
* **gameOver()**: Determines if the game is over and identifies the winner(s).
* **updateTurn()**: Manages turn transitions.
* **addMove()**: Validates and records player moves.

**Express.js Integration**

1. **Serving Static Files**:

* app.use(express.static(path.resolve("")));
* This middleware serves static files (like index.html, CSS, and JavaScript files) from the root directory of the project.

1. **Routing**

* app.get('/', (req, res) => {

return res.sendFile(path.resolve('index.html'));

});

* This route handler sends the index.html file when a client requests the root URL (/), ensuring that the main HTML file is delivered to the user's browser.

**Client-Side JavaScript**

The client-side JavaScript handles user interactions, game rendering, and communication with the server via Socket.IO.

**Key Features**

* **Online Count Display**: Updates the number of online users.
* **Lobby Creation and Management**: Allows players to create and join lobbies, updating the UI accordingly.
* **Game Updates**: Listens for server events to update the game state, render moves, and display results.
* **Form Interactions**: Manages form inputs for player name, grid size, and player count.

**Conclusion**

The Dots and Boxes project combines a well-structured HTML layout, stylish CSS, and interactive JavaScript to create an online game. The server-side implementation with Node.js and Socket.IO ensures real-time updates and smooth gameplay for the user(s). The project demonstrates a comprehensive approach to web development, integrating frontend and backend technologies to deliver a whole user experience.