Requirement ID	Description	Story Points	Priority	Sprint No	
1	Setup basic web dotnet project scaffold	2	1	1	Done
2	Build a basic WebSocket endpoint to for clients to connect to	3	2	1	Done
3	Implement RoomHandler using the singleton design pattern to be able to create find and delete rooms	5	3	1	Done
4	Implement frontends connection/state updates to backend (Join, Leave, click, sync, error)	5	3	1	Done
5	Implement GameHandler for the board: Apply move, validate turn/update turn, detect win/draw, Allow for restarting	8	3	2	Partially Done
6	Frontend UI to show board	5	2	1	Done
7	Allow for multiple rooms by way of urls	5	4	1	Done
8	Allow for spectators of a game (So if player A-B is in a room if a third join it doesn't expect 3 players but puts 3rd as spectator)	2	4		
9	In room chat (Can be based on a couple of preset messages. Think clash royale/chess. com where you have a "good game" button or something)	3	6		
10	Setting up for basic win tracking/leaderboard display. Could be done with a full-stack salt + hash account creation or could be a simple "Enter your name" for simplicity	5	5		
11	Unit testing applied to backend GameHandler/RoomHandler	3	6		
12	Make site responsive/work properly on mobile/Different resolutions. (Done with breakpoints on frontend)	5	7		
13	Data validation of types on backend to ensure low crash rate (Making sure "event" that was passed is a string "row" was a int, etc)	3	5		
14	"Quick Play" i.e: button on homepage that allows one to search roomhandler for any 1 player room and join it.	3	8		
15	Locking of room so you can't join through quick play but only link	3	8		
16	Documentation	5	9		