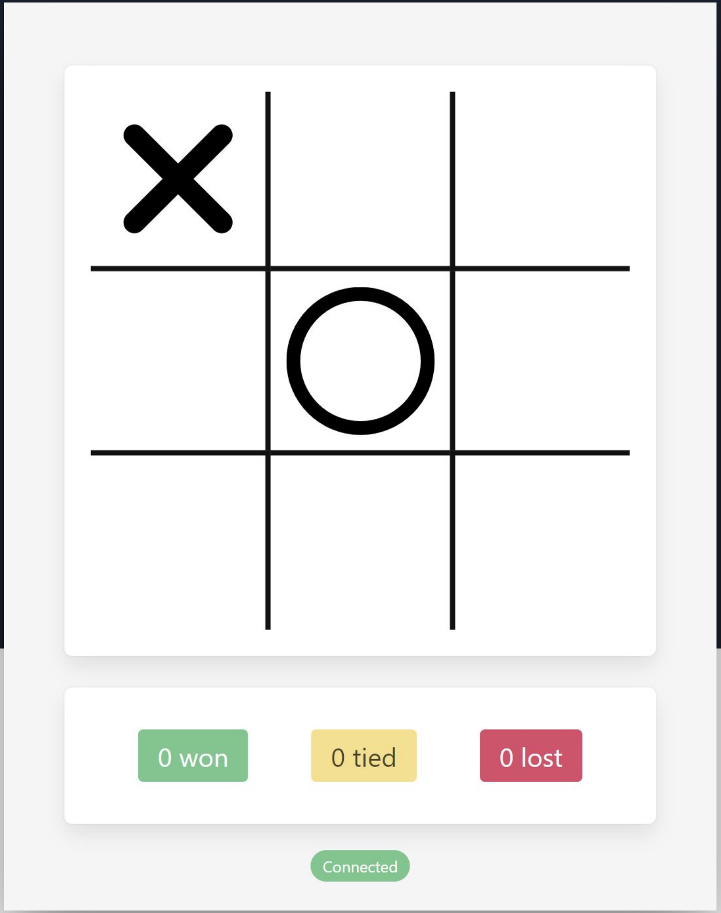
Tic-Tac-Socket

A WebSocket Powered Game

Evan Wieland - Gabe Roy

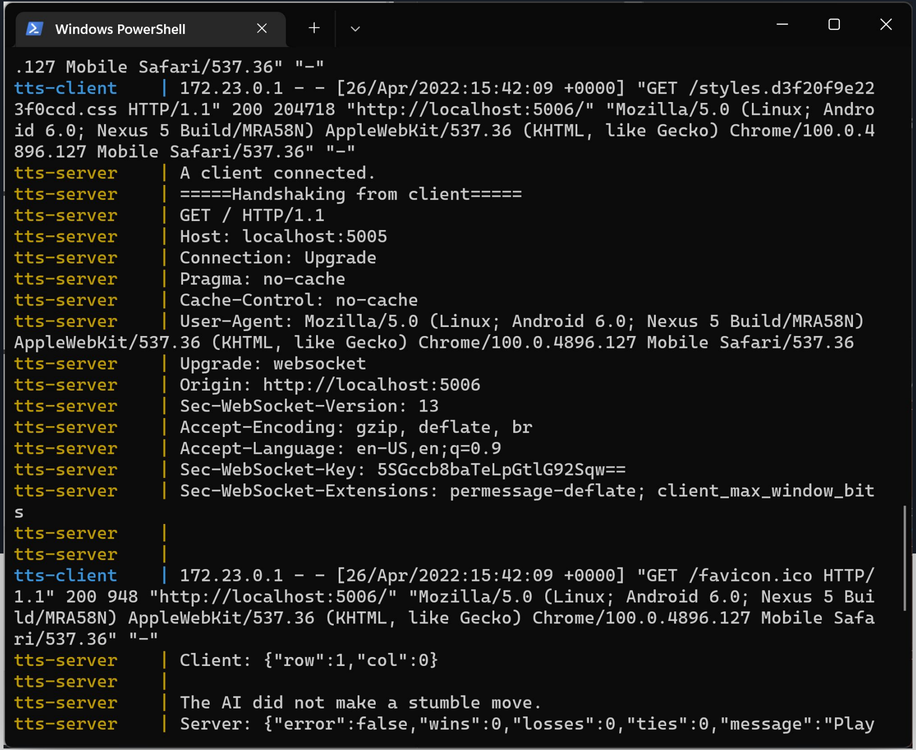
Realtime Tic-Tac-Toe Game

- Human vs. AI
- Session based gameplay
- AI is allowed to stumble to improve gameplay experience



The Back-End

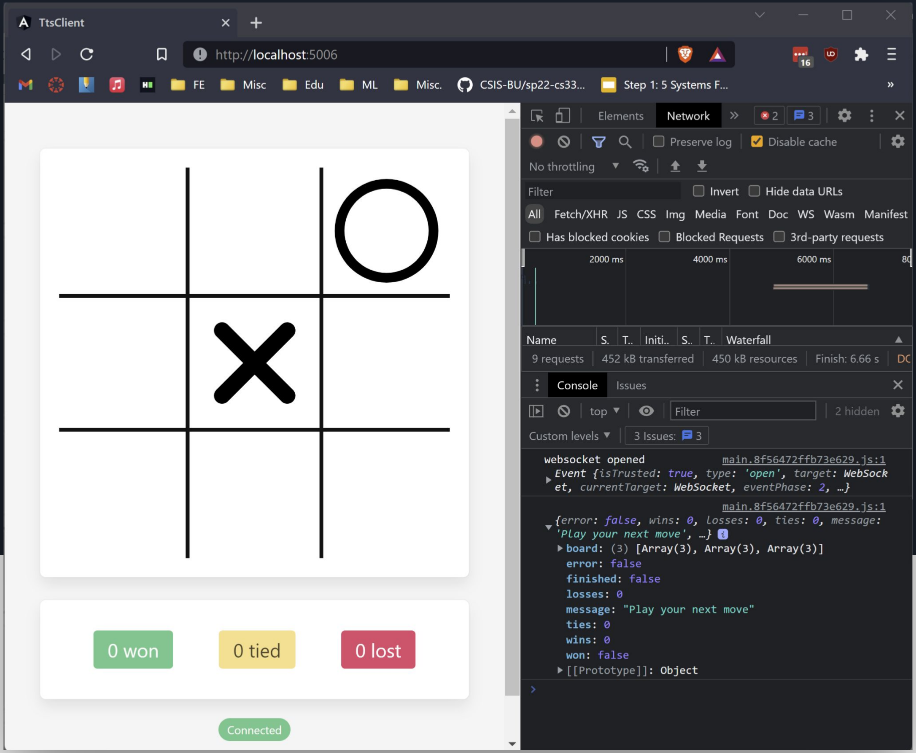
- .NET server
- Allows multiple connections
- Kicks new connections to separate thread
- Uses WebSocket protocol



```
.127 Mobile Safari/537.36" "-"
tts-client    | 172.23.0.1 - - [26/Apr/2022:15:42:09 +0000] "GET /styles.d3f20f9e22
3f0ccd.css HTTP/1.1" 200 204718 "http://localhost:5006/" "Mozilla/5.0 (Linux; Andro
id 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4
896.127 Mobile Safari/537.36" "-"
tts-server    | A client connected.
tts-server    | =====Handshaking from client=====
tts-server    | GET / HTTP/1.1
tts-server    | Host: localhost:5005
tts-server    | Connection: Upgrade
tts-server    | Pragma: no-cache
tts-server    | Cache-Control: no-cache
tts-server    | User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Mobile Safari/537.36
tts-server    | Upgrade: websocket
tts-server    | Origin: http://localhost:5006
tts-server    | Sec-WebSocket-Version: 13
tts-server    | Accept-Encoding: gzip, deflate, br
tts-server    | Accept-Language: en-US,en;q=0.9
tts-server    | Sec-WebSocket-Key: 5SGccb8baTeLpGtlG92Sqw==
tts-server    | Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bit
s
tts-server    |
tts-server    |
tts-client    | 172.23.0.1 - - [26/Apr/2022:15:42:09 +0000] "GET /favicon.ico HTTP/
1.1" 200 948 "http://localhost:5006/" "Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Bui
ld/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Mobile Safa
ri/537.36" "-"
tts-server    | Client: {"row":1,"col":0}
tts-server    |
tts-server    | The AI did not make a stumble move.
tts-server    | Server: {"error":false,"wins":0,"losses":0,"ties":0,"message":"Play
```

The Front-End

- Angular Application
- Establishes connection to server
- Communicates via WebSocket

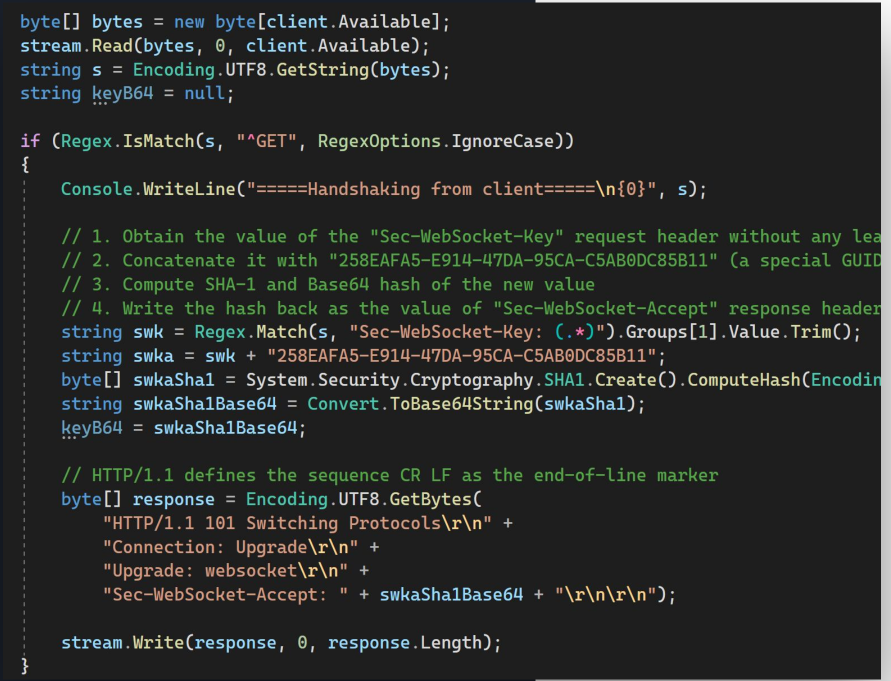


```
9 requests | 452 kB transferred | 450 kB resources | Finish: 6.66 s
```

```
websocket opened                                   main.8f56472ffb73e629.js:1
▸ Event {isTrusted: true, type: 'open', target: WebSock
  et, currentTarget: WebSocket, eventPhase: 2, …}
                                                   main.8f56472ffb73e629.js:1
▾ {error: false, wins: 0, losses: 0, ties: 0, message:
  'Play your next move', …}
   ▸ board: (3) [Array(3), Array(3), Array(3)]
     error: false
     finished: false
     losses: 0
     message: "Play your next move"
     ties: 0
     wins: 0
     won: false
   ▸ [[Prototype]]: Object
```

WebSocket Protocol

RFC 6455



```
byte[] bytes = new byte[client.Available];
stream.Read(bytes, 0, client.Available);
string s = Encoding.UTF8.GetString(bytes);
string keyB64 = null;

if (Regex.IsMatch(s, "^GET", RegexOptions.IgnoreCase))
{
    Console.WriteLine("=====Handshaking from client=====\n{0}", s);

    // 1. Obtain the value of the "Sec-WebSocket-Key" request header without any lea
    // 2. Concatenate it with "258EAFA5-E914-47DA-95CA-C5AB0DC85B11" (a special GUID
    // 3. Compute SHA-1 and Base64 hash of the new value
    // 4. Write the hash back as the value of "Sec-WebSocket-Accept" response header
    string swk = Regex.Match(s, "Sec-WebSocket-Key: (.*)").Groups[1].Value.Trim();
    string swka = swk + "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";
    byte[] swkaSha1 = System.Security.Cryptography.SHA1.Create().ComputeHash(Encodin
    string swkaSha1Base64 = Convert.ToBase64String(swkaSha1);
    keyB64 = swkaSha1Base64;

    // HTTP/1.1 defines the sequence CR LF as the end-of-line marker
    byte[] response = Encoding.UTF8.GetBytes(
        "HTTP/1.1 101 Switching Protocols\r\n" +
        "Connection: Upgrade\r\n" +
        "Upgrade: websocket\r\n" +
        "Sec-WebSocket-Accept: " + swkaSha1Base64 + "\r\n\r\n");

    stream.Write(response, 0, response.Length);
}
```

- Handshake initiated by client as a standard HTTP request
- Handshake requests an upgrade to WebSocket
- Websocket allows for bi-directional communication
- Eliminates the need for Long Polling

Client - Server Interaction

01 Client sends user gameplay input to server.

02 Server responds with AI’s next move and current game state.

03 Client reflects game state returned from the server.

Example Client Request

```
{
    "row": 2,
    "col": 1
}
```

Example Server Response

```
{
    "error": false,
    "wins": 0,
    "losses": 0,
    "ties": 0,
    "message": "Play your next move",
    "won": false,
    "finished": false,
    "board": [[-1,-1,-1],
               [-1,0,-1],
               [-1,1,-1]]
}
```

Building & Executing

```
$ docker-compose build

$ docker-compose up
```

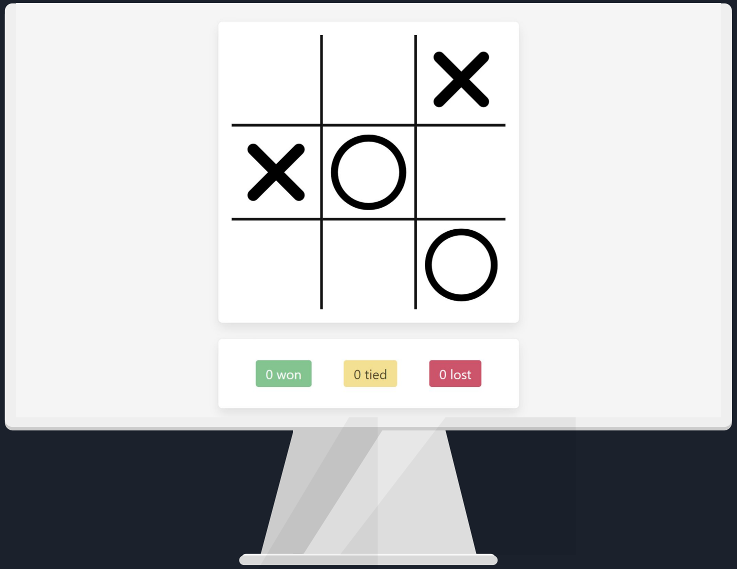
- Application is Dockerized
- Docker Compose builds the client image and server image in unison
- Images have ports exposed

Running the Client

Web application using WebSocket

Once the Docker images are running, connect to:

http://localhost:5006



Thank you!

GitHub Repo