Snail Race[broken]

Lets create a web page that all users in the same wifi can interact to each other

Creating a web page that allows users on the same Wi-Fi network to interact involves several steps. Here's a basic outline of what you need to do:

1. **Set Up the Server:**
    - Use a technology stack that supports real-time communication, such as Node.js with Socket.IO.
    - Set up a server that can handle multiple clients and manage real-time data transfer between them.
2. **Create the Front-End:**
    - Develop a front-end using HTML, CSS, and JavaScript. This will include the user interface for chat, notifications, or any other interaction you want to support.
3. **Establish Real-Time Communication:**
    - Use WebSockets to enable real-time communication. Socket.IO is a popular library for this purpose.
4. **Deploy the Application:**
    - Host your server on a local machine or a local network server.
    - Ensure all devices on the Wi-Fi network can access the server.

Here is a simple example to get you started:

Server (Node.js with Express and Socket.IO)

First, set up a Node.js project:

```bash
mkdir wifi-chat
cd wifi-chat
npm init -y
npm install express socket.io
```

Create a file named `` `server.js` ``:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

app.use(express.static('public'));

io.on('connection', (socket) => {
    console.log('A user connected');

    socket.on('chat message', (msg) => {
        io.emit('chat message', msg);
    });

    socket.on('disconnect', () => {
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Create a folder named `` `public` `` and inside it, an `` `index.html` `` file:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
    </style>
</head>
<body>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" /><button>Send</button>
    </form>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');

        form.addEventListener('submit', function(e) {
            e.preventDefault();
            if (input.value) {
                socket.emit('chat message', input.value);
                input.value = '';
            }
        });

        socket.on('chat message', function(msg) {
            var item = document.createElement('li');
            item.textContent = msg;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
        });
    </script>
</body>
</html>
```

Running the Server

Run your server using Node.js:

```bash
node server.js
```

Accessing the Web Page

- Ensure that your server machine and client devices are connected to the same Wi-Fi network.
- Access the web page by typing the server's local IP address followed by `` `:3000` `` in a web browser. For example, `` `http://192.168.1.5:3000` ``.

This setup provides a simple chat application that users on the same Wi-Fi network can use to interact in real-time. You can expand upon this by adding more features, improving the UI, and enhancing security as needed.

how i check my server's local IP I'm in a local mac

To check your server's local IP address on a Mac, you can use the following steps:

Using System Preferences

1. **Open System Preferences:** Click on the Apple icon in the top-left corner of your screen and select "System Preferences."
2. **Network:** Click on "Network."
3. **Current Connection:** In the left pane, select your current network connection (Wi-Fi or Ethernet). Your local IP address will be displayed on the right side under the connection status.

Using Terminal

1. **Open Terminal:** You can find Terminal in the Applications folder under Utilities, or by searching for it using Spotlight (Cmd + Space and type "Terminal").

2. **Enter the Command:** Type the following command and press Enter:

   ```bash
   ifconfig | grep "inet "
   ```

   This will show a list of all network interfaces and their IP addresses. Look for the IP address associated with your active network interface. Typically, it will look something like `` `192.168.x.x` ``.

Example

After running the command, you might see an output similar to this:

```
inet 127.0.0.1 netmask 0xff000000
inet 192.168.1.5 netmask 0xffffff00 broadcast 192.168.1.255
```

The IP address `` `192.168.1.5` `` in this example is the local IP address of your server.

Running Your Server

Once you have your local IP address, ensure your server is running by executing:

```bash
node server.js
```

Accessing the Server from Other Devices

On any other device connected to the same Wi-Fi network, open a web browser and type in the local IP address followed by `` `:3000` ``. For example:

```arduino
http://192.168.1.5:3000
```

This will allow other devices on the same network to access the chat application you set up.

now i wnat to see the user that sened the message, also add the user name on top of the chat

To add usernames and display them on top of the chat messages, you'll need to make some changes to both your server and client code. Here's how you can do it:

Server (Node.js with Express and Socket.IO)

Update your `server.js` file to handle usernames:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

app.use(express.static('public'));

io.on('connection', (socket) => {
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        io.emit('chat message', { username: socket.username, message: msg });
    });

    socket.on('disconnect', () => {
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` file to include a way for users to set their username:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
```

```
            #messages { list-style-type: none; padding: 0; }
            #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
            #form { display: flex; }
            #input { flex-grow: 1; padding: 10px; }
            #usernameForm { display: flex; margin-bottom: 10px; }
            #usernameInput { flex-grow: 1; padding: 10px; }
        </style>
    </head>
    <body>
        <form id="usernameForm">
            <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
    <button>Set Username</button>
        </form>
        <ul id="messages"></ul>
        <form id="form" action="">
            <input id="input" autocomplete="off" /><button>Send</button>
        </form>

        <script src="/socket.io/socket.io.js"></script>
        <script>
            var socket = io();

            var usernameForm = document.getElementById('usernameForm');
            var usernameInput = document.getElementById('usernameInput');
            var form = document.getElementById('form');
            var input = document.getElementById('input');
            var messages = document.getElementById('messages');

            usernameForm.addEventListener('submit', function(e) {
                e.preventDefault();
                if (usernameInput.value) {
                    socket.emit('set username', usernameInput.value);
                    usernameForm.style.display = 'none';
                    form.style.display = 'flex';
                }
            });

            form.addEventListener('submit', function(e) {
                e.preventDefault();
                if (input.value) {
                    socket.emit('chat message', input.value);
                    input.value = '';
                }
            });

            socket.on('chat message', function(data) {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
                messages.appendChild(item);
                window.scrollTo(0, document.body.scrollHeight);
            });
        </script>
    </body>
    </html>
```

Explanation

1. **Server (`server.js`):**
    - When a user connects, they can set their username by emitting the `set username` event.
    - When a user sends a message, the `chat message` event now includes the username.
    - The server broadcasts the message along with the username to all connected clients.
2. **Client (`index.html`):**
    - A form for setting the username is added at the top (`usernameForm`).

  - When the username is set, the username form is hidden, and the chat message form is displayed.
  - Each chat message now includes the username before the message text.

With these changes, users can set their usernames, and each message in the chat will display the username of the sender.

now keep track of the number of users

To keep track of the number of users connected to your chat application, you can update the server to maintain a count of connected users and broadcast this information to all clients whenever a user connects or disconnects.

Server (Node.js with Express and Socket.IO)

Update your `server.js` file to keep track of the number of connected users:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    io.emit('user count', userCount);
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        io.emit('chat message', { username: socket.username, message: msg });
    });

    socket.on('disconnect', () => {
        userCount--;
        io.emit('user count', userCount);
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `` `index.html` `` file to display the number of connected users:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /> <button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" /><button>Send</button>
    </form>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'flex';
            }
        });

        form.addEventListener('submit', function(e) {
            e.preventDefault();
            if (input.value) {
                socket.emit('chat message', input.value);
                input.value = '';
            }
        });

        socket.on('user count', function(count) {
            userCountDiv.textContent = 'Users online: ' + count;
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
        });
    </script>
```

```
</body>
</html>
```

Explanation

1. **Server (`server.js`):**
   - A variable `userCount` is used to keep track of the number of connected users.
   - When a user connects, `userCount` is incremented, and the new count is broadcasted to all clients.
   - When a user disconnects, `userCount` is decremented, and the new count is broadcasted to all clients.
2. **Client (`index.html`):**
   - A `div` element with the id `userCount` is added to display the number of connected users.
   - An event listener for the `user count` event updates the text content of the `userCount` div to show the current number of users online.

With these changes, the number of connected users will be tracked and displayed in real-time on the chat application.

at the bottom of the page 5 images named(d4,d6,d8,d10,d12)

To add images at the bottom of the page, you can include HTML image elements within a container, styled to ensure they appear correctly. Here's how you can integrate this into your existing `index.html`:

Client (HTML, CSS, and JavaScript)

Update your `index.html` file to include the images:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; }
    </style>
```

```
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" /><button>Send</button>
    </form>

    <div id="images">
        <img src="d4.png" alt="d4">
        <img src="d6.png" alt="d6">
        <img src="d8.png" alt="d8">
        <img src="d10.png" alt="d10">
        <img src="d12.png" alt="d12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'flex';
            }
        });

        form.addEventListener('submit', function(e) {
            e.preventDefault();
            if (input.value) {
                socket.emit('chat message', input.value);
                input.value = '';
            }
        });

        socket.on('user count', function(count) {
            userCountDiv.textContent = 'Users online: ' + count;
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
        });
    </script>
</body>
</html>
```

Explanation

1. **HTML:**
    - Added a new `div` with the id `images` to contain the images.
    - Inside this `div`, added five `img` elements with `src` attributes pointing to the image files (d4.png, d6.png, d8.png, d10.png, d12.png).

2. **CSS:**

   - Styled the `images` div to have some margin at the top and use flexbox to space the images evenly.
   - Set a fixed width and height for the images to ensure consistent sizing.

3. **Images:**

   - Ensure you have the image files (`d4.png`, `d6.png`, `d8.png`, `d10.png`, `d12.png`) in the `public` directory so they can be served by the server.

Running the Server

After making these changes, make sure your images are correctly placed in the `public` directory and then run your server again:

```bash
node server.js
```

This setup will display the images at the bottom of the chat application, enhancing the UI as per your requirements.

delete the submit, and when a user pres one of the images send the number coresponding on that image

To remove the submit button and send a message when an image is clicked, you need to modify the client-side code. Specifically, you'll add event listeners to the images and emit a socket event with the corresponding number when an image is clicked.

Here's the updated `index.html` file:

Client (HTML, CSS, and JavaScript)

Update your `index.html` file:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var number = e.target.getAttribute('data-number');
                socket.emit('chat message', number);
            }
        });

        socket.on('user count', function(count) {
            userCountDiv.textContent = 'Users online: ' + count;
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
        });
    </script>
</body>
</html>
```

Explanation

1. **HTML:**
   - Removed the submit button from the chat form.
   - Added a `` `data-number` `` attribute to each image to store the corresponding number.
2. **CSS:**
   - Added a `` `cursor: pointer;` `` style to the images to indicate they are clickable.
3. **JavaScript:**
   - Modified the event listener for `` `usernameForm` `` to hide the form and show the images once the username is set.
   - Added an event listener to the `` `images` `` container that listens for clicks on the images. When an image is clicked, it retrieves the corresponding number from the `` `data-number` `` attribute and emits a `` `chat message` `` event with that number.
   - Updated the `` `chat message` `` event handler to display the username and message in the chat.

Server (Node.js with Express and Socket.IO)

Make sure your `` `server.js` `` handles the `` `chat message` `` event as before:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    io.emit('user count', userCount);
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        io.emit('chat message', { username: socket.username, message: msg });
    });

    socket.on('disconnect', () => {
        userCount--;
        io.emit('user count', userCount);
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

With these changes, when a user clicks on one of the images, the corresponding number is sent as a chat message, and the messages in the chat will display the sender's username and the number they sent.

now, instead of the number send a random number from 1 to the image number

To send a random number between 1 and the image's corresponding number, you can modify the event listener for image clicks to generate a random number within the specified range. Here's how to update your client-side code:

Client (HTML, CSS, and JavaScript)

Update your `index.html` file:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>
    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', randomNumber);
            }
        });

        socket.on('user count', function(count) {
            userCountDiv.textContent = 'Users online: ' + count;
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
        });
    </script>
</body>
</html>
```

Explanation

1. **JavaScript:**
    - The event listener for image clicks now generates a random number between 1 and the value of the `data-number` attribute of the clicked image.
    - This random number is emitted as the message content.
2. **Generating a Random Number:**
    - `Math.floor(Math.random() * maxNumber) + 1` generates a random integer between 1 and `maxNumber`.

Server (Node.js with Express and Socket.IO)

Make sure your `server.js` handles the `chat message` event as before:

javascript

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    io.emit('user count', userCount);
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        io.emit('chat message', { username: socket.username, message: msg });
    });

    socket.on('disconnect', () => {
        userCount--;
        io.emit('user count', userCount);
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

With these changes, clicking an image will send a random number between 1 and the image's corresponding number, and the messages in the chat will display the sender's username and the random number they sent.

now lets create a grid jsut over the images, that has 25 columns and the same numbers of rows as users, each user will get assigned to a row, and an image with different background color will move from the first row to the left based on the number that user get.

To create a grid with 25 columns and the same number of rows as users, and to move an image with a different background color based on the number each user gets, we need to implement a few things:

1. Create a grid with 25 columns and dynamic rows.
2. Assign each user to a row in the grid.
3. Move an image across the grid based on the number a user gets.

Here's how to update your code to achieve this:

Server (Node.js with Express and Socket.IO)

We need to keep track of users to assign them rows in the grid. Update your `server.js` to handle this:

javascript

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    users[socket.id] = userCount;
    io.emit('user count', { count: userCount, users });
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        io.emit('chat message', { username: socket.username, message: msg, row:
users[socket.id] });
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` file to include the grid and handle the movement of images:

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px;
margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid
#ccc; }
        .grid-cell.active { background-color: red; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', randomNumber);
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.count);
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message;
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
            moveImage(data.row, data.message);
        });
        function renderGrid(rows) {
            grid.innerHTML = '';
            for (var i = 0; i < rows; i++) {
                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + i + '-' + j;
                    grid.appendChild(cell);
                }
            }
        }

        function moveImage(row, number) {
            for (var i = 0; i < 25; i++) {
                var cell = document.getElementById('cell-' + (row - 1) + '-' + i);
                if (cell) {
                    cell.classList.remove('active');
                }
            }
            var activeCell = document.getElementById('cell-' + (row - 1) + '-' + (number -
1));
            if (activeCell) {
                activeCell.classList.add('active');
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **HTML:**
    - Added a `div` with the id `grid` to hold the grid cells.
    - Added a `class` and `id` to each cell for identification and styling.
2. **CSS:**
    - Styled the grid using CSS Grid Layout with 25 columns.
    - Added styles for the grid cells, including an `active` class for the cell that the image will move to.
3. **JavaScript:**
    - Modified the `user count` event listener to render the grid based on the number of users.
    - Added a `moveImage` function to move an image to the correct cell based on the number received.
    - Updated the `chat message` event listener to call the `moveImage` function with the user's row and number.

With these changes, each user will be assigned a row in the grid, and when a user clicks an image, an image in the corresponding row will move to the column based on the random number generated.

sum the results for each user

To sum the results for each user, you can maintain a cumulative score for each user on the server side and update it whenever a new number is received. Then, send the updated score back to the client. Here's how you can do it:

Server (Node.js with Express and Socket.IO)

Update your `server.js` to keep track of each user's score:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    users[socket.id] = userCount;
    scores[socket.id] = 0;
    io.emit('user count', { count: userCount, users });
    console.log('A user connected');

    socket.on('set username', (username) => {
        socket.username = username;
    });

    socket.on('chat message', (msg) => {
        scores[socket.id] += msg;
        io.emit('chat message', { username: socket.username, message: msg, row:
users[socket.id], score: scores[socket.id] });
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        delete scores[socket.id];
        io.emit('user count', { count: userCount, users });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` file to display the cumulative score:

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px; margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid #ccc; }
        .grid-cell.active { background-color: red; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /><button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', randomNumber);
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.count);
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message + '
(Total: ' + data.score + ')';
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
            moveImage(data.row, data.message);
        });

        function renderGrid(rows) {
            grid.innerHTML = '';
            for (var i = 0; i < rows; i++) {
                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + i + '-' + j;
                    grid.appendChild(cell);
                }
            }
        }

        function moveImage(row, number) {
            for (var i = 0; i < 25; i++) {
                var cell = document.getElementById('cell-' + (row - 1) + '-' + i);
                if (cell) {
                    cell.classList.remove('active');
                }
            }
            var activeCell = document.getElementById('cell-' + (row - 1) + '-' + (number -
1));
            if (activeCell) {
                activeCell.classList.add('active');
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **Server:**

    - Added an object `scores` to keep track of each user's cumulative score.
    - Updated the `chat message` event to update the user's score and send it along with the message.

2. **Client:**

    - Modified the `chat message` event listener to display the total score along with the message.
    - No change in the logic for moving the image, as it will still move to the appropriate column based on the number sent.

With these changes, the user's score will be summed and displayed along with each message, and the image will move to the appropriate cell in the grid.

on the grid make that the red backround matches the total, also add the user name in the assigned row

To update the grid so that the red background matches the total score and add the username in the assigned row, you need to make a few changes:

1. Modify the server to include the username when sending the user count and update the score.
2. Update the client to render the username in the grid and move the cell based on the total score.

Server (Node.js with Express and Socket.IO)

Update your `server.js` to include the username when sending the user count and chat message:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    users[socket.id] = { id: userCount, username: '', score: 0 };
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected');

    socket.on('set username', (username) => {
        users[socket.id].username = username;
        io.emit('user count', { count: userCount, users: Object.values(users) });
    });

    socket.on('chat message', (msg) => {
        users[socket.id].score += msg;
        io.emit('chat message', { username: users[socket.id].username, message: msg, row:
users[socket.id].id, score: users[socket.id].score });
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` to render the username in the grid and adjust the active cell based on the total score:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px; margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid #ccc; }
        .grid-cell.active { background-color: red; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /> <button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');
        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', randomNumber);
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            var item = document.createElement('li');
            item.innerHTML = '<strong>' + data.username + ':</strong> ' + data.message + ' (Total: ' + data.score + ')';
            messages.appendChild(item);
            window.scrollTo(0, document.body.scrollHeight);
            moveImage(data.row, data.score);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + (user.id - 1) + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function moveImage(row, total) {
            for (var i = 0; i < 25; i++) {
                var cell = document.getElementById('cell-' + (row - 1) + '-' + i);
                if (cell) {
                    cell.classList.remove('active');
                }
            }
            var activeCell = document.getElementById('cell-' + (row - 1) + '-' + (total % 25));
            if (activeCell) {
                activeCell.classList.add('active');
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **Server:**

  - Modified the `users` object to store username and score.
  - Included the updated users object in the `user count` event.
  - Updated the `chat message` event to include the total score.

2. **Client:**

  - Rendered usernames in the grid by adding a `username-cell` before the grid cells for each user.
  - Updated the `moveImage` function to move the active cell based on the total score, using modulo 25 to loop within the grid columns.

With these changes, each user's row will display their username, and the red background cell will match their total score, moving within the grid's columns based on the total.

let's wait for all the users to select one image before adding all the results, remember to reset the selection after all users has made his call, also to avoid confucions, sort the grid by user name, when the names are eqal add a random number at the end, also remmber to update the corresponding grid row for the correspoding user.

To implement the features of waiting for all users to select an image before adding all the results, resetting the selection, sorting the grid by username (with a random number appended if usernames are equal), and updating the corresponding grid row, follow the steps below.

Server (Node.js with Express and Socket.IO)

Update your `server.js` to keep track of the selections made by each user and handle the results after all users have made their selections:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};
let selections = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    const userId = userCount + '-' + Math.floor(Math.random() * 10000); // Ensure unique ID
even if usernames are equal
    users[socket.id] = { id: userId, username: '', score: 0 };
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected');

    socket.on('set username', (username) => {
        users[socket.id].username = username;
        io.emit('user count', { count: userCount, users: Object.values(users).sort((a, b) => a.username.localeCompare(b.username)) });
    });

    socket.on('chat message', (msg) => {
        selections[socket.id] = msg;
        if (Object.keys(selections).length === userCount) {
            Object.keys(selections).forEach(id => {
                users[id].score += selections[id];
            });
            io.emit('chat message', { users: Object.values(users), selections });
            selections = {}; // Reset selections after processing
        }
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        delete scores[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` to handle the updated logic from the server:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px; margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid #ccc; }
        .grid-cell.active { background-color: red; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /><button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4">
        <img src="d6.png" alt="d6" data-number="6">
        <img src="d8.png" alt="d8" data-number="8">
        <img src="d10.png" alt="d10" data-number="10">
        <img src="d12.png" alt="d12" data-number="12">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', randomNumber);
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('active');
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add('active');
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **Server:**
    - Added a `selections` object to keep track of selections made by each user.
    - Once all users have made their selections, the server updates the scores and emits the `chat message` event with the updated user data.
    - Reset the `selections` object after processing.
2. **Client:**
    - Updated the `chat message` event listener to handle the updated data structure and render the grid based on the total scores.
    - Modified the `renderGrid` and `updateGrid` functions to reflect the updated usernames and scores.
    - Ensured the red background cell moves based on the total score.

With these changes, the system will wait for all users to select an image, sum the results, reset the selection, sort the grid by username (with a random number appended if usernames are equal), and update the corresponding grid row for each user.

change the background colour .grid-cell.active depending on the image selected for the user "d4"
use cian
for "d6" Sky Blue
for "d8" Blue
for "d10" Indigo
and for "d12" purple

To implement the color changes for the active grid cell based on the image selected, you need to update both the client and server to pass and handle the selected image information.

Server (Node.js with Express and Socket.IO)

Update the `` `server.js` `` to include the selected image in the `` `chat message` `` event:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};
let selections = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    const userId = userCount + '-' + Math.floor(Math.random() * 10000); // Ensure unique ID
even if usernames are equal
    users[socket.id] = { id: userId, username: '', score: 0, image: '' };
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected');

    socket.on('set username', (username) => {
        users[socket.id].username = username;
        io.emit('user count', { count: userCount, users: Object.values(users).sort((a, b) =>
a.username.localeCompare(b.username)) });
    });

    socket.on('chat message', (msg) => {
        const { randomNumber, image } = msg;
        selections[socket.id] = { randomNumber, image };
        if (Object.keys(selections).length === userCount) {
            Object.keys(selections).forEach(id => {
                users[id].score += selections[id].randomNumber;
                users[id].image = selections[id].image;
            });
            io.emit('chat message', { users: Object.values(users), selections });
            selections = {}; // Reset selections after processing
        }
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `` `index.html` `` to handle the image selection and apply the corresponding background color:

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px; margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid #ccc; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
        .grid-cell.cian { background-color: cyan; }
        .grid-cell.sky-blue { background-color: skyblue; }
        .grid-cell.blue { background-color: blue; }
        .grid-cell.indigo { background-color: indigo; }
        .grid-cell.purple { background-color: purple; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /><button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **Server:**
    - The server now handles an object containing both the `randomNumber` and the `image` color.
    - After all users have made their selections, it updates the users' scores and stores the image color.
    - The server emits the updated user data to the clients.

2. **Client:**
    - The `images` click event now sends both the random number and the image color to the server.
    - The `chat message` event listener on the client updates the grid cells based on the received data, applying the correct color class (`cian`, `sky-blue`, `blue`, `indigo`, `purple`).
    - The `updateGrid` function ensures that the correct color is applied to the active cell based on the user's total score.

These changes ensure that the active grid cell's background color matches the image selected by the user, with colors specific to each type of image.

add an image snail on the active .grid-cell.active

To add an image of a snail to the active grid cell, you need to update the client-side code to include an `<img>` tag within the active grid cell and style it accordingly.

Client (HTML, CSS, and JavaScript)

Update your `index.html` to handle the addition of the snail image:

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px;
margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid
#ccc; position: relative; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-
weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
        .grid-cell.cian { background-color: cyan; }
        .grid-cell.sky-blue { background-color: skyblue; }
        .grid-cell.blue { background-color: blue; }
        .grid-cell.indigo { background-color: indigo; }
        .grid-cell.purple { background-color: purple; }
        .snail { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);
width: 30px; height: 30px; }
    </style>
</head>
```

```
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'snail.png'; // Ensure you have this image in your 'public'
folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **CSS:**
    - Added styles for the `` `.snail` `` class to position the snail image correctly within the grid cell.
2. **HTML:**
    - Ensure you have a `` `snail.png` `` image in your `` `public` `` folder for the snail image to display correctly.
3. **JavaScript:**
    - The `` `updateGrid` `` function now adds the snail image to the active cell and removes it from any previous active cells.
    - An `` `<img>` `` tag is created for the snail image and appended to the active cell.

This will place a snail image inside the active grid cell, positioned at the center, and change the background color based on the image selected by the user.

now, if 2 users pick the same image, his maximum value for that round is 1

To implement the rule that if two users pick the same image, their maximum value for that round is 1, you'll need to adjust the server logic to check for duplicate selections before sending the results to the clients.

Server (Node.js with Express and Socket.IO)

Update the `server.js` to handle the check for duplicate selections:

javascript

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};
let selections = {};

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    const userId = userCount + '-' + Math.floor(Math.random() * 10000); // Ensure unique ID even if usernames are equal
    users[socket.id] = { id: userId, username: '', score: 0, image: '' };
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected');

    socket.on('set username', (username) => {
        users[socket.id].username = username;
        io.emit('user count', { count: userCount, users: Object.values(users).sort((a, b) => a.username.localeCompare(b.username)) });
    });

    socket.on('chat message', (msg) => {
        const { randomNumber, image } = msg;
        selections[socket.id] = { randomNumber, image };

        if (Object.keys(selections).length === userCount) {
            let imageCounts = {};
            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                imageCounts[image] = (imageCounts[image] || 0) + 1;
            });

            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                let randomNumber = selections[id].randomNumber;
                if (imageCounts[image] > 1) {
                    randomNumber = 1; // Set maximum value to 1 if more than one user selected the same image
                }
                users[id].score += randomNumber;
                users[id].image = image;
            });

            io.emit('chat message', { users: Object.values(users), selections });
            selections = {}; // Reset selections after processing
        }
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` to ensure it works with the updated server logic:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px;
margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid
#ccc; position: relative; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-
weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
        .grid-cell.cian { background-color: cyan; }
        .grid-cell.sky-blue { background-color: skyblue; }
        .grid-cell.blue { background-color: blue; }
        .grid-cell.indigo { background-color: indigo; }
        .grid-cell.purple { background-color: purple; }
        .snail { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);
width: 30px; height: 30px; }
    </style>
</head>
<body>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
```

```
          cell.removeChild(snailImg);
        }
      });
      var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
      if (activeCell) {
        activeCell.classList.add(user.image);
        var snailImg = document.createElement('img');
        snailImg.src = 'snail.png'; // Ensure you have this image in your 'public'
folder
        snailImg.className = 'snail';
        activeCell.appendChild(snailImg);
      }
    }
  </script>
</body>
</html>
```

Explanation

1. **Server:**
    - The server now checks for duplicate image selections.
    - If more than one user selects the same image, their `` `randomNumber` `` is set to 1 for that round.
    - The server then updates the scores and emits the updated user data to the clients.
2. **Client:**
    - The client remains mostly the same but ensures that the new logic from the server is correctly handled and displayed.

With these changes, if two users pick the same image, their maximum value for that round is set to 1. The snail image will still appear in the active grid cell, which will change color based on the image selected by the user.

Add the user name at the top of the page

To add the username at the top of the page, you can create a new HTML element to display the username and update it when the user sets their username. Here's how to do it:

Client (HTML, CSS, and JavaScript)

Update your `` `index.html` `` to include a username display area and handle the username display logic:

```html
<!DOCTYPE html>
<html>
<head>
  <title>WiFi Chat</title>
```

```
    <style>
        body { font-family: Arial, sans-serif; }
        #usernameDisplay { font-size: 20px; margin-bottom: 10px; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px; margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid #ccc; position: relative; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
        .grid-cell.cian { background-color: cyan; }
        .grid-cell.sky-blue { background-color: skyblue; }
        .grid-cell.blue { background-color: blue; }
        .grid-cell.indigo { background-color: indigo; }
        .grid-cell.purple { background-color: purple; }
        .snail { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); width: 30px; height: 30px; }
    </style>
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" /> <button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });
        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'snail.png'; // Ensure you have this image in your 'public'
folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **HTML:**

   - Added a `` `div` `` with the id `` `usernameDisplay` `` to show the username at the top of the page.
   - Inside this `` `div` ``, a `` `span` `` with the id `` `username` `` will display the username.

2. **JavaScript:**

   - Updated the event listener for the username form submission to set the content of the `` `usernameDisplay` `` span to the entered username.

These changes will display the username at the top of the page once it has been set.

Apply an invert filter to the snail.png

To apply an invert filter to the `` `snail.png` `` image, you can use CSS. Here's how you can update the `` `index.html` `` to apply the invert filter to the snail image:

Client (HTML, CSS, and JavaScript)

Update your `` `index.html` `` to include the CSS for the invert filter:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <style>
        body { font-family: Arial, sans-serif; }
        #usernameDisplay { font-size: 20px; margin-bottom: 10px; }
        #userCount { margin-bottom: 10px; }
        #messages { list-style-type: none; padding: 0; }
        #messages li { padding: 8px; margin-bottom: 10px; background: #f1f1f1; }
        #form { display: flex; }
        #input { flex-grow: 1; padding: 10px; }
        #usernameForm { display: flex; margin-bottom: 10px; }
        #usernameInput { flex-grow: 1; padding: 10px; }
        #grid { display: grid; grid-template-columns: repeat(25, 40px); grid-gap: 5px;
margin-top: 20px; }
        .grid-row { display: contents; }
        .grid-cell { width: 40px; height: 40px; background-color: #e0e0e0; border: 1px solid
#ccc; position: relative; }
        .username-cell { grid-column: span 25; text-align: left; padding-left: 10px; font-
weight: bold; }
        #images { margin-top: 20px; display: flex; justify-content: space-around; }
        #images img { width: 50px; height: 50px; cursor: pointer; }
        .grid-cell.cian { background-color: cyan; }
        .grid-cell.sky-blue { background-color: skyblue; }
        .grid-cell.blue { background-color: blue; }
        .grid-cell.indigo { background-color: indigo; }
        .grid-cell.purple { background-color: purple; }
        .snail { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);
width: 30px; height: 30px; filter: invert(1); }
    </style>
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'snail.png'; // Ensure you have this image in your 'public'
folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Explanation

1. **CSS:**
    - Added `filter: invert(1);` to the `.snail` class to apply the invert filter to the snail image.

These changes will apply an invert filter to the `snail.png` image, making it appear with inverted colors in the grid.

let's split the code into html and css

To split the code into separate HTML and CSS files, you'll create two files: `index.html` and `styles.css`. Here's how you can do it:

`index.html`

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'snail.png'; // Ensure you have this image in your 'public'
folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

`styles.css`

```css
body {
    font-family: Arial, sans-serif;
}

#usernameDisplay {
    font-size: 20px;
    margin-bottom: 10px;
}

#userCount {
    margin-bottom: 10px;
}

#messages {
    list-style-type: none;
    padding: 0;
}
```

```
#messages li {
    padding: 8px;
    margin-bottom: 10px;
    background: #f1f1f1;
}

#form {
    display: flex;
}

#input {
    flex-grow: 1;
    padding: 10px;
}

#usernameForm {
    display: flex;
    margin-bottom: 10px;
}

#usernameInput {
    flex-grow: 1;
    padding: 10px;
}

#grid {
    display: grid;
    grid-template-columns: repeat(25, 40px);
    grid-gap: 5px;
    margin-top: 20px;
}

.grid-row {
    display: contents;
}

.grid-cell {
    width: 40px;
    height: 40px;
    background-color: #e0e0e0;
    border: 1px solid #ccc;
    position: relative;
}

.username-cell {
    grid-column: span 25;
    text-align: left;
    padding-left: 10px;
    font-weight: bold;
}

#images {
    margin-top: 20px;
    display: flex;
    justify-content: space-around;
}

#images img {
    width: 50px;
    height: 50px;
    cursor: pointer;
}

.grid-cell.cian {
    background-color: cyan;
}

.grid-cell.sky-blue {
    background-color: skyblue;
}

.grid-cell.blue {
    background-color: blue;
}
```

```
.grid-cell.indigo {
    background-color: indigo;
}

.grid-cell.purple {
    background-color: purple;
}

.snail {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 30px;
    height: 30px;
    filter: invert(1);
}
```

Explanation

1. **`index.html`:**
    - This file now contains only the HTML and JavaScript code.
2. **`styles.css`:**
    - This file contains all the CSS styles previously embedded in the HTML.

Ensure that both `index.html` and `styles.css` are in the same directory, and that `styles.css` is correctly linked in the `<head>` of `index.html` with the `<link>` tag. The changes ensure a clean separation of HTML and CSS, improving maintainability and readability.

move the images into a folder

To move the images into a folder, let's assume you have a folder named `images` in your project's directory. You'll update the HTML to reflect the new paths for the images.

Project Directory Structure

```bash
/your-project-directory
    /images
        d4.png
        d6.png
        d8.png
        d10.png
        d12.png
        snail.png
    index.html
    styles.css
```

`index.html`

Update the image paths to point to the new `images` folder:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="cian">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="sky-blue">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="indigo">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="purple">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('cian', 'sky-blue', 'blue', 'indigo', 'purple');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

`styles.css`

css

```css
body {
    font-family: Arial, sans-serif;
}

#usernameDisplay {
    font-size: 20px;
```

```
        margin-bottom: 10px;
    }

    #userCount {
        margin-bottom: 10px;
    }

    #messages {
        list-style-type: none;
        padding: 0;
    }

    #messages li {
        padding: 8px;
        margin-bottom: 10px;
        background: #f1f1f1;
    }

    #form {
        display: flex;
    }

    #input {
        flex-grow: 1;
        padding: 10px;
    }

    #usernameForm {
        display: flex;
        margin-bottom: 10px;
    }

    #usernameInput {
        flex-grow: 1;
        padding: 10px;
    }

    #grid {
        display: grid;
        grid-template-columns: repeat(25, 40px);
        grid-gap: 5px;
        margin-top: 20px;
    }

    .grid-row {
        display: contents;
    }

    .grid-cell {
        width: 40px;
        height: 40px;
        background-color: #e0e0e0;
        border: 1px solid #ccc;
        position: relative;
    }

    .username-cell {
        grid-column: span 25;
        text-align: left;
        padding-left: 10px;
        font-weight: bold;
    }

    #images {
        margin-top: 20px;
        display: flex;
        justify-content: space-around;
    }

    #images img {
        width: 50px;
        height: 50px;
        cursor: pointer;
    }
```

```
.grid-cell.cian {
    background-color: cyan;
}

.grid-cell.sky-blue {
    background-color: skyblue;
}

.grid-cell.blue {
    background-color: blue;
}

.grid-cell.indigo {
    background-color: indigo;
}

.grid-cell.purple {
    background-color: purple;
}

.snail {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 30px;
    height: 30px;
    filter: invert(1);
}
```

Explanation

1. **`index.html`:**
    - Updated the image paths to point to the new `images` folder.
    - Ensure you have moved the images (`d4.png`, `d6.png`, `d8.png`, `d10.png`, `d12.png`, and `snail.png`) into the `images` folder.

2. **`styles.css`:**
    - No changes needed here, just make sure this file is linked properly in `index.html`.

These changes ensure that the images are correctly loaded from the `images` folder, and the project remains organized.

lets change some colours d4 to orange, d6 to purple, d10 to green,d12 to yellow

To update the colors for the respective images, you'll need to modify both the HTML (for the data attributes) and the CSS (for the class styles).

`index.html`

Update the data attributes for the images:

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="orange">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="purple">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="green">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="yellow">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < 25; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('orange', 'purple', 'blue', 'green', 'yellow');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
25));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

`styles.css`

Update the CSS classes for the new colors:

```css
body {
    font-family: Arial, sans-serif;
}

#usernameDisplay {
    font-size: 20px;
    margin-bottom: 10px;
}

#userCount {
    margin-bottom: 10px;
}

#messages {
    list-style-type: none;
    padding: 0;
}

#messages li {
    padding: 8px;
    margin-bottom: 10px;
    background: #f1f1f1;
}

#form {
    display: flex;
}

#input {
    flex-grow: 1;
    padding: 10px;
}

#usernameForm {
    display: flex;
    margin-bottom: 10px;
}

#usernameInput {
    flex-grow: 1;
    padding: 10px;
}

#grid {
    display: grid;
    grid-template-columns: repeat(25, 40px);
    grid-gap: 5px;
    margin-top: 20px;
}

.grid-row {
    display: contents;
}

.grid-cell {
    width: 40px;
    height: 40px;
    background-color: #e0e0e0;
    border: 1px solid #ccc;
    position: relative;
}

.username-cell {
    grid-column: span 25;
    text-align: left;
    padding-left: 10px;
    font-weight: bold;
}

#images {
    margin-top: 20px;
    display: flex;
    justify-content: space-around;
}

#images img {
    width: 50px;
    height: 50px;
    cursor: pointer;
}

.grid-cell.orange {
    background-color: orange;
}

.grid-cell.purple {
    background-color: purple;
}
.grid-cell.blue {
    background-color: blue;
}

.grid-cell.green {
    background-color: green;
}

.grid-cell.yellow {
    background-color: yellow;
}

.snail {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 30px;
    height: 30px;
    filter: invert(1);
}
```

Explanation

1. **`index.html`:**
    - Updated the **`data-color`** attributes for the images to the new colors.
2. **`styles.css`:**
    - Added new CSS classes for the colors: **`orange`**, **`purple`**, **`green`**, and **`yellow`**.

These changes ensure the colors correspond to the respective dice images and are properly reflected in the grid cells.

Make that the number of columns and all releted stuff depends on a single variable value

To make the number of columns and all related elements depend on a single variable value, you can define a variable for the number of columns in your JavaScript code and use it to dynamically generate the grid and update styles.

Here's how you can update your code to achieve this:

`index.html`

Update the HTML to remove the fixed grid-template-columns and handle the grid generation in JavaScript:

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div id="usernameDisplay">Welcome, <span id="username"></span></div>
    <div id="userCount">Users online: 0</div>
    <form id="usernameForm">
        <input id="usernameInput" placeholder="Enter your username" autocomplete="off" />
<button>Set Username</button>
    </form>
    <ul id="messages"></ul>
    <form id="form" action="">
        <input id="input" autocomplete="off" style="display: none;" />
    </form>

    <div id="grid"></div>

    <div id="images">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="orange">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="purple">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="green">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="yellow">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script>
        var socket = io();
        var numColumns = 25; // Variable for number of columns

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.style.gridTemplateColumns = `repeat(${numColumns}, 40px)`;
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < numColumns; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('orange', 'purple', 'blue', 'green', 'yellow');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
numColumns));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

`styles.css`

Remove the fixed `grid-template-columns` from the CSS:

```css
body {
    font-family: Arial, sans-serif;
}

#usernameDisplay {
    font-size: 20px;
    margin-bottom: 10px;
}

#userCount {
    margin-bottom: 10px;
}
```

```
#messages {
    list-style-type: none;
    padding: 0;
}

#messages li {
    padding: 8px;
    margin-bottom: 10px;
    background: #f1f1f1;
}

#form {
    display: flex;
}

#input {
    flex-grow: 1;
    padding: 10px;
}

#usernameForm {
    display: flex;
    margin-bottom: 10px;
}

#usernameInput {
    flex-grow: 1;
    padding: 10px;
}

#grid {
    display: grid;
    grid-gap: 5px;
    margin-top: 20px;
}

.grid-row {
    display: contents;
}

.grid-cell {
    width: 40px;
    height: 40px;
    background-color: #e0e0e0;
    border: 1px solid #ccc;
    position: relative;
}

.username-cell {
    grid-column: span 25;
    text-align: left;
    padding-left: 10px;
    font-weight: bold;
}

#images {
    margin-top: 20px;
    display: flex;
    justify-content: space-around;
}

#images img {
    width: 50px;
    height: 50px;
    cursor: pointer;
}

.grid-cell.orange {
    background-color: orange;
}

.grid-cell.purple {
    background-color: purple;
}
```

```
.grid-cell.blue {
    background-color: blue;
}

.grid-cell.green {
    background-color: green;
}

.grid-cell.yellow {
    background-color: yellow;
}

.snail {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 30px;
    height: 30px;
    filter: invert(1);
}
```

Explanation

1. **JavaScript (`index.html`):**
    - Added a variable `numColumns` to control the number of columns.
    - Updated the `renderGrid` function to use `numColumns` for setting the `grid-template-columns` and for generating grid cells.
    - Updated the `updateGrid` function to use `numColumns` for calculating the active cell.
2. **CSS (`styles.css`):**
    - Removed the fixed `grid-template-columns` property. This will now be set dynamically in the JavaScript.

By using a single variable `numColumns`, you can easily control the number of columns in your grid and ensure all related functionality adapts accordingly.

can we add a popular theme to make it more beautifull?

Certainly! We can use a popular CSS framework like Bootstrap to make the page more visually appealing. Below, I'll integrate Bootstrap into the HTML and CSS and adjust the layout to take advantage of Bootstrap's components and styles.

Updated `index.html` with Bootstrap

```html
<!DOCTYPE html>
<html>
<head>
    <title>WiFi Chat</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body class="container mt-4">
    <div class="text-center mb-4">
        <h1>WiFi Chat</h1>
        <p id="usernameDisplay" class="lead">Welcome, <span id="username"></span></p>
        <div id="userCount" class="badge badge-info">Users online: 0</div>
    </div>
    <form id="usernameForm" class="form-inline justify-content-center mb-4">
        <input id="usernameInput" class="form-control mr-2" placeholder="Enter your
username" autocomplete="off" />
        <button class="btn btn-primary">Set Username</button>
    </form>
    <ul id="messages" class="list-group mb-4"></ul>
    <form id="form" action="" style="display: none;">
        <input id="input" autocomplete="off" class="form-control" />
    </form>

    <div id="grid" class="mb-4"></div>

    <div id="images" class="d-flex justify-content-around">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="orange" class="dice-
img">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="purple" class="dice-
img">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue" class="dice-
img">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="green" class="dice-
img">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="yellow"
class="dice-img">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js">
</script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
</script>
    <script>
        var socket = io();
        var numColumns = 25; // Variable for number of columns

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.className = 'list-group-item';
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.style.gridTemplateColumns = `repeat(${numColumns}, 40px)`;
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < numColumns; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('orange', 'purple', 'blue', 'green', 'yellow');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
numColumns));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Updated `styles.css`

Use Bootstrap classes and custom styles:

css

```css
body {
    font-family: Arial, sans-serif;
}

#usernameDisplay {
    font-size: 20px;
    margin-bottom: 10px;
}

#userCount {
    margin-bottom: 10px;
}

#messages {
    list-style-type: none;
    padding: 0;
}

#messages li {
    padding: 8px;
    margin-bottom: 10px;
    background: #f1f1f1;
}

#form {
    display: flex;
}

#input {
    flex-grow: 1;
    padding: 10px;
}

#usernameForm {
    display: flex;
    margin-bottom: 10px;
}

#usernameInput {
    flex-grow: 1;
    padding: 10px;
}

#grid {
    display: grid;
    grid-gap: 5px;
    margin-top: 20px;
}

.grid-row {
    display: contents;
}

.grid-cell {
    width: 40px;
    height: 40px;
    background-color: #e0e0e0;
    border: 1px solid #ccc;
    position: relative;
}

.username-cell {
    grid-column: span 25;
    text-align: left;
    padding-left: 10px;
    font-weight: bold;
}

#images {
    margin-top: 20px;
    display: flex;
```

```
        justify-content: space-around;
    }

    #images img {
        width: 50px;
        height: 50px;
        cursor: pointer;
    }

    .grid-cell.orange {
        background-color: orange;
    }

    .grid-cell.purple {
        background-color: purple;
    }

    .grid-cell.blue {
        background-color: blue;
    }

    .grid-cell.green {
        background-color: green;
    }

    .grid-cell.yellow {
        background-color: yellow;
    }

    .snail {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 30px;
        height: 30px;
        filter: invert(1);
    }

    .dice-img {
        transition: transform 0.2s;
    }

    .dice-img:hover {
        transform: scale(1.2);
    }
```

Explanation

1. **HTML (`` `index.html` ``):**
    - Integrated Bootstrap CSS and JS.
    - Used Bootstrap classes for layout and styling (`` `container` ``, `` `text-center` ``, `` `mb-4` ``, `` `lead` ``, `` `badge` ``, `` `form-inline` ``, `` `form-control` ``, `` `btn` ``, `` `d-flex` ``, `` `justify-content-around` ``).
    - Added jQuery and Popper.js scripts required for Bootstrap.

2. **CSS (`` `styles.css` ``):**
    - Removed unnecessary styles since Bootstrap handles the styling.
    - Added custom styles for grid cells and dice images.
    - Added a hover effect on dice images.

This should enhance the visual appeal of your page using Bootstrap, while still maintaining the custom functionality and styles you need.

change the app name to SnailRace, and add a .ico image

To change the app name to "SnailRace" and add a `` `.ico` `` image, you need to update the HTML `` `<title>` `` and include a link to the favicon (`` `.ico` `` image). Here's how you can do it:

1. **Update the app name in the HTML `` `<title>` `` tag.**
2. **Add a link to the `` `.ico` `` image in the HTML `` `<head>` ``.**
3. **Ensure the `` `.ico` `` image is placed in the correct location in your project directory.**

Updated `` `index.html` ``

```html
<!DOCTYPE html>
<html>
<head>
    <title>SnailRace</title>
    <link rel="icon" href="images/favicon.ico" type="image/x-icon">
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body class="container mt-4">
    <div class="text-center mb-4">
        <h1>SnailRace</h1>
        <p id="usernameDisplay" class="lead">Welcome, <span id="username"></span></p>
        <div id="userCount" class="badge badge-info">Users online: 0</div>
    </div>
    <form id="usernameForm" class="form-inline justify-content-center mb-4">
        <input id="usernameInput" class="form-control mr-2" placeholder="Enter your username" autocomplete="off" />
        <button class="btn btn-primary">Set Username</button>
    </form>
    <ul id="messages" class="list-group mb-4"></ul>
    <form id="form" action="" style="display: none;">
        <input id="input" autocomplete="off" class="form-control" />
    </form>

    <div id="grid" class="mb-4"></div>

    <div id="images" class="d-flex justify-content-around">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="orange" class="dice-img">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="purple" class="dice-img">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue" class="dice-img">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="green" class="dice-img">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="yellow" class="dice-img">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
```

```
</script>
    <script>
        var socket = io();
        var numColumns = 25; // Variable for number of columns

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var usernameForm = document.getElementById('usernameForm');
        var usernameInput = document.getElementById('usernameInput');
        var form = document.getElementById('form');
        var input = document.getElementById('input');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        usernameForm.addEventListener('submit', function(e) {
            e.preventDefault();
            if (usernameInput.value) {
                socket.emit('set username', usernameInput.value);
                usernameDisplay.textContent = usernameInput.value;
                usernameForm.style.display = 'none';
                form.style.display = 'none';
                images.style.display = 'flex';
            }
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        socket.on('user count', function(data) {
            userCountDiv.textContent = 'Users online: ' + data.count;
            renderGrid(data.users);
        });

        socket.on('chat message', function(data) {
            messages.innerHTML = ''; // Clear previous messages
            data.users.forEach(user => {
                var item = document.createElement('li');
                item.className = 'list-group-item';
                item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                messages.appendChild(item);
                updateGrid(user);
            });
            window.scrollTo(0, document.body.scrollHeight);
        });

        function renderGrid(users) {
            grid.style.gridTemplateColumns = `repeat(${numColumns}, 40px)`;
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < numColumns; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('orange', 'purple', 'blue', 'green', 'yellow');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
numColumns));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Updated `styles.css`

No changes needed here except ensuring the dice images are visually appealing and adjusting any additional styles if required.

Explanation

1. **HTML (`index.html`):**
    - Changed the `<title>` tag to "SnailRace".
    - Added a link to the favicon (`<link rel="icon" href="images/favicon.ico" type="image/x-icon">`).
2. **Adding Favicon:**
    - Make sure you have your `favicon.ico` file in the `images` folder within your project directory.

With these updates, your web application is now called "SnailRace," and it includes a favicon. This should make the application look more polished and professional.

lets create a background that fits the theme!!!!

To create a background that fits the "SnailRace" theme, we can use a grassy or garden-themed background image that gives a natural and playful feel. You can either use an existing image or create a custom one. I'll show you how to integrate a background image into your existing setup.

Step 1: Choose or Create a Background Image

First, choose or create a background image that fits the "SnailRace" theme. You can find a suitable image from free stock photo websites like Unsplash or Pexels. Save this image in your `images` folder with a name like `background.jpg`.

Step 2: Update `styles.css` to Include the Background

Add CSS rules to set the background image and make it cover the entire page.

Updated `styles.css`

css

```css
body {
    font-family: Arial, sans-serif;
    background: url('images/background.jpg') no-repeat center center fixed;
    background-size: cover;
    color: #fff;
}

#usernameDisplay {
    font-size: 20px;
    margin-bottom: 10px;
}

#userCount {
    margin-bottom: 10px;
}

#messages {
    list-style-type: none;
    padding: 0;
}

#messages li {
    padding: 8px;
    margin-bottom: 10px;
    background: rgba(255, 255, 255, 0.8);
    color: #000;
}

#form {
    display: flex;
}

#input {
    flex-grow: 1;
    padding: 10px;
}

#usernameForm {
    display: flex;
    margin-bottom: 10px;
}

#usernameInput {
    flex-grow: 1;
    padding: 10px;
}

#grid {
    display: grid;
    grid-gap: 5px;
    margin-top: 20px;
}

.grid-row {
    display: contents;
}

.grid-cell {
    width: 40px;
    height: 40px;
    background-color: rgba(224, 224, 224, 0.8);
    border: 1px solid #ccc;
```

```
        position: relative;
    }

    .username-cell {
        grid-column: span 25;
        text-align: left;
        padding-left: 10px;
        font-weight: bold;
        background-color: rgba(0, 0, 0, 0.5);
        color: #fff;
    }

    #images {
        margin-top: 20px;
        display: flex;
        justify-content: space-around;
    }

    #images img {
        width: 50px;
        height: 50px;
        cursor: pointer;
    }

    .grid-cell.orange {
        background-color: orange;
    }

    .grid-cell.purple {
        background-color: purple;
    }

    .grid-cell.blue {
        background-color: blue;
    }

    .grid-cell.green {
        background-color: green;
    }

    .grid-cell.yellow {
        background-color: yellow;
    }

    .snail {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 30px;
        height: 30px;
        filter: invert(1);
    }

    .dice-img {
        transition: transform 0.2s;
    }

    .dice-img:hover {
        transform: scale(1.2);
    }
```

Updated `index.html`

No changes needed for the background in `index.html`, as the background is set entirely through CSS.

Explanation

1. **CSS (`styles.css`):**

  - Set the background image for the `body` element with `background: url('images/background.jpg') no-repeat center center fixed;`.
  - Use `background-size: cover;` to ensure the image covers the entire background without repeating.
  - Added a semi-transparent background for the messages and username cells to ensure text readability against the background.

By following these steps and updating the CSS, your "SnailRace" theme will have a fitting background that enhances the visual appeal of the web application. Make sure your `background.jpg` image is appropriately placed in the `images` folder and adjust the image path if necessary.

Create a list of appealing visual adjectives, speed adjectives and a snail related name at least 10 of each,
When a user connects pick an appealing visual adjective, an speed adjective and a snail related name and assign that name to that new user
remember that you need to wait until all users made his selection before submitting results

Sure, here's how you can create the lists of adjectives and snail-related names, then assign a random name to each new user upon connection. Additionally, I'll ensure the system waits until all users have made their selections before submitting the results.

Lists of Adjectives and Snail-Related Names

Visual Adjectives:

1. Shiny
2. Glittering
3. Sparkling
4. Radiant
5. Vibrant
6. Luminous
7. Colorful
8. Bright
9. Glossy
10. Majestic

Speed Adjectives:

1. Swift
2. Rapid
3. Quick
4. Speedy
5. Agile
6. Brisk
7. Zippy

8. Nimble
9. Fleet
10. Hasty

Snail-Related Names:

1. Slider
2. Shellie
3. Slippy
4. Slinky
5. Turbo
6. Zoomer
7. Crawlie
8. Whirlie
9. Dash
10. Breeze

Server (Node.js with Express and Socket.IO)

Update your `server.js` to assign a random name to each new user:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};
let selections = {};

const visualAdjectives = ['Shiny', 'Glittering', 'Sparkling', 'Radiant', 'Vibrant',
'Luminous', 'Colorful', 'Bright', 'Glossy', 'Majestic'];
const speedAdjectives = ['Swift', 'Rapid', 'Quick', 'Speedy', 'Agile', 'Brisk', 'Zippy',
'Nimble', 'Fleet', 'Hasty'];
const snailNames = ['Slider', 'Shellie', 'Slippy', 'Slinky', 'Turbo', 'Zoomer', 'Crawlie',
'Whirlie', 'Dash', 'Breeze'];

function getRandomElement(arr) {
    return arr[Math.floor(Math.random() * arr.length)];
}

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    const userId = userCount + '-' + Math.floor(Math.random() * 10000); // Ensure unique ID
even if usernames are equal
    const userName = `${getRandomElement(visualAdjectives)}
${getRandomElement(speedAdjectives)} ${getRandomElement(snailNames)}`;
    users[socket.id] = { id: userId, username: userName, score: 0, image: '' };
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected:', userName);

    socket.on('set username', (username) => {
        users[socket.id].username = username;
        io.emit('user count', { count: userCount, users: Object.values(users).sort((a, b) =>
a.username.localeCompare(b.username)) });
```

```
    });

    socket.on('chat message', (msg) => {
        const { randomNumber, image } = msg;
        selections[socket.id] = { randomNumber, image };

        if (Object.keys(selections).length === userCount) {
            let imageCounts = {};
            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                imageCounts[image] = (imageCounts[image] || 0) + 1;
            });

            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                let randomNumber = selections[id].randomNumber;
                if (imageCounts[image] > 1) {
                    randomNumber = 1; // Set maximum value to 1 if more than one user selected the same image
                }
                users[id].score += randomNumber;
                users[id].image = image;
            });

            io.emit('chat message', { users: Object.values(users), selections });
            selections = {}; // Reset selections after processing
        }
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Client (HTML, CSS, and JavaScript)

Update your `index.html` and `styles.css` if necessary to handle the updated functionality. The changes are mainly on the server side to manage user naming and results submission.

Updated `index.html`

No changes needed here for the naming and results submission. The existing structure handles the display of usernames and scores dynamically based on server data.

Updated `styles.css`

No changes needed here for the naming and results submission. The existing styles handle the display of usernames and scores dynamically based on server data.

Explanation

1. **Server:**
    - Added lists of visual adjectives, speed adjectives, and snail-related names.
    - Created a function to get a random element from an array.
    - Assigned a random name to each new user upon connection using these lists.

  - The server ensures it waits for all users to make their selections before processing and broadcasting the results.

This setup ensures each user gets a unique and fun snail-related name upon connection and maintains the core functionality of the game.

delete the text boxt to summit a new user name, and show snail-related name as the new user name

To remove the text box for submitting a new username and show the snail-related name assigned to each user automatically, you need to update both the client and server code.

Updated `index.html`

Remove the username input form and update the structure to display the automatically assigned snail-related name:

```html
<!DOCTYPE html>
<html>
<head>
    <title>SnailRace</title>
    <link rel="icon" href="images/favicon.ico" type="image/x-icon">
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body class="container mt-4">
    <div class="text-center mb-4">
        <h1>SnailRace</h1>
        <p id="usernameDisplay" class="lead">Welcome, <span id="username"></span></p>
        <div id="userCount" class="badge badge-info">Users online: 0</div>
    </div>
    <ul id="messages" class="list-group mb-4"></ul>
    <div id="grid" class="mb-4"></div>
    <div id="images" class="d-flex justify-content-around">
        <img src="images/d4.png" alt="d4" data-number="4" data-color="orange" class="dice-
img">
        <img src="images/d6.png" alt="d6" data-number="6" data-color="purple" class="dice-
img">
        <img src="images/d8.png" alt="d8" data-number="8" data-color="blue" class="dice-
img">
        <img src="images/d10.png" alt="d10" data-number="10" data-color="green" class="dice-
img">
        <img src="images/d12.png" alt="d12" data-number="12" data-color="yellow"
class="dice-img">
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js">
</script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
</script>
    <script>
```

```
        var socket = io();
        var numColumns = 25; // Variable for number of columns

        var usernameDisplay = document.getElementById('username');
        var userCountDiv = document.getElementById('userCount');
        var messages = document.getElementById('messages');
        var images = document.getElementById('images');
        var grid = document.getElementById('grid');

        socket.on('connect', function() {
            socket.on('user count', function(data) {
                userCountDiv.textContent = 'Users online: ' + data.count;
                renderGrid(data.users);
            });

            socket.on('chat message', function(data) {
                messages.innerHTML = ''; // Clear previous messages
                data.users.forEach(user => {
                    var item = document.createElement('li');
                    item.className = 'list-group-item';
                    item.innerHTML = '<strong>' + user.username + ':</strong> (Total: ' +
user.score + ')';
                    messages.appendChild(item);
                    updateGrid(user);
                });
                window.scrollTo(0, document.body.scrollHeight);
            });

            socket.on('assign username', function(username) {
                usernameDisplay.textContent = username;
            });
        });

        images.addEventListener('click', function(e) {
            if (e.target.tagName === 'IMG') {
                var maxNumber = parseInt(e.target.getAttribute('data-number'), 10);
                var color = e.target.getAttribute('data-color');
                var randomNumber = Math.floor(Math.random() * maxNumber) + 1;
                socket.emit('chat message', { randomNumber, image: color });
            }
        });

        function renderGrid(users) {
            grid.style.gridTemplateColumns = `repeat(${numColumns}, 40px)`;
            grid.innerHTML = '';
            users.forEach(user => {
                var usernameCell = document.createElement('div');
                usernameCell.className = 'username-cell';
                usernameCell.textContent = user.username;
                grid.appendChild(usernameCell);

                for (var j = 0; j < numColumns; j++) {
                    var cell = document.createElement('div');
                    cell.className = 'grid-cell';
                    cell.id = 'cell-' + user.id + '-' + j;
                    grid.appendChild(cell);
                }
            });
        }

        function updateGrid(user) {
            var userRowCells = document.querySelectorAll('[id^="cell-' + user.id + '-"]');
            userRowCells.forEach(cell => {
                cell.classList.remove('orange', 'purple', 'blue', 'green', 'yellow');
                var snailImg = cell.querySelector('.snail');
                if (snailImg) {
                    cell.removeChild(snailImg);
                }
            });
            var activeCell = document.getElementById('cell-' + user.id + '-' + (user.score %
numColumns));
            if (activeCell) {
                activeCell.classList.add(user.image);
                var snailImg = document.createElement('img');
                snailImg.src = 'images/snail.png'; // Ensure you have this image in your
'public' folder
                snailImg.className = 'snail';
                activeCell.appendChild(snailImg);
            }
        }
    </script>
</body>
</html>
```

Updated `server.js`

Update the server code to automatically assign and emit a snail-related name to each new user:

```javascript
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let userCount = 0;
let users = {};
let scores = {};
let selections = {};

const visualAdjectives = ['Shiny', 'Glittering', 'Sparkling', 'Radiant', 'Vibrant',
'Luminous', 'Colorful', 'Bright', 'Glossy', 'Majestic'];
const speedAdjectives = ['Swift', 'Rapid', 'Quick', 'Speedy', 'Agile', 'Brisk', 'Zippy',
'Nimble', 'Fleet', 'Hasty'];
const snailNames = ['Slider', 'Shellie', 'Slippy', 'Slinky', 'Turbo', 'Zoomer', 'Crawlie',
'Whirlie', 'Dash', 'Breeze'];

function getRandomElement(arr) {
    return arr[Math.floor(Math.random() * arr.length)];
}

app.use(express.static('public'));

io.on('connection', (socket) => {
    userCount++;
    const userId = userCount + '-' + Math.floor(Math.random() * 10000); // Ensure unique ID
even if usernames are equal
    const userName = `${getRandomElement(visualAdjectives)}
${getRandomElement(speedAdjectives)} ${getRandomElement(snailNames)}`;
    users[socket.id] = { id: userId, username: userName, score: 0, image: '' };
    socket.emit('assign username', userName); // Send assigned username to the client
    io.emit('user count', { count: userCount, users: Object.values(users) });
    console.log('A user connected:', userName);

    socket.on('chat message', (msg) => {
        const { randomNumber, image } = msg;
        selections[socket.id] = { randomNumber, image };

        if (Object.keys(selections).length === userCount) {
            let imageCounts = {};
            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                imageCounts[image] = (imageCounts[image] || 0) + 1;
            });

            Object.keys(selections).forEach(id => {
                const image = selections[id].image;
                let randomNumber = selections[id].randomNumber;
                if (imageCounts[image] > 1) {
                    randomNumber = 1; // Set maximum value to 1 if more than one user
selected the same image
```

```
                }
                users[id].score += randomNumber;
                users[id].image = image;
            });

            io.emit('chat message', { users: Object.values(users), selections });
            selections = {}; // Reset selections after processing
        }
    });

    socket.on('disconnect', () => {
        userCount--;
        delete users[socket.id];
        io.emit('user count', { count: userCount, users: Object.values(users) });
        console.log('A user disconnected');
    });
});

server.listen(3000, () => {
    console.log('Listening on *:3000');
});
```

Explanation

1. **HTML (`index.html`):**

    - Removed the username input form.
    - Displayed the automatically assigned username.
    - Ensured the page layout remains consistent with the new design.

2. **JavaScript (Client-side in `index.html`):**

    - Added an event listener to receive the assigned username from the server and display it.
    - Removed references to the username form and submission.

3. **Node.js Server (`server.js`):**

    - Automatically assigned a snail-related name to each new user upon connection.
    - Sent the assigned username to the client using `socket.emit('assign username', userName)`.
    - Ensured the game waits for all users to make their selections before processing results.

With these changes, users will automatically receive a unique snail-related name upon connecting, and the game will proceed smoothly with the updated functionalities.

ChatGPT puede cometer errores. Considera verificar la información importante.