let peer;

let myStream;

let connections = [];

let userName;

/\* ✅ Ajout de la vidéo cadrée avec le nom \*/

function ajoutVideo(stream, name = "Moi") {

const container = document.createElement('div');

container.classList.add('video-container');

const video = document.createElement('video');

video.autoplay = true;

video.srcObject = stream;

video.style.transform = "scaleX(-1)"; // 🔄 Corrige la vidéo inversée

const label = document.createElement('p');

label.textContent = name;

container.appendChild(video);

container.appendChild(label);

document.getElementById('participants').appendChild(container);

// ✅ Ajouter un élément audio séparé pour les flux entrants et baisser le volume

if (name !== "Moi") {

const audio = document.createElement("audio");

audio.srcObject = stream;

audio.autoplay = true;

audio.volume = 0.5; // Réduit le volume pour minimiser l'écho

document.body.appendChild(audio);

}

}

/\* ✅ Inscription PeerJS \*/

function register() {

userName = document.getElementById('name').value.trim();

if (!userName) return alert("Veuillez entrer un nom !");

peer = new Peer(userName, { debug: 2 });

peer.on('open', (id) => console.log("Mon ID PeerJS :", id));

navigator.mediaDevices.getUserMedia({

video: true,

audio: {

echoCancellation: true,

noiseSuppression: true

}

})

.then((stream) => {

myStream = stream;

ajoutVideo(stream, "Moi");

document.getElementById('register').style.display = 'none';

document.getElementById('userControls').style.display = 'block';

peer.on('call', (call) => {

call.answer(myStream);

call.on('stream', (remoteStream) => ajoutVideo(remoteStream, call.peer));

});

peer.on('connection', (conn) => {

connections.push(conn);

conn.on('data', ({ sender, message }) => afficherMessage(sender, message));

});

})

.catch(() => alert("Impossible d'accéder à la caméra/microphone."));

}

/\* ✅ Ajouter un participant \*/

function appelUser() {

const name = document.getElementById('add').value.trim();

if (!name) return alert("Veuillez entrer un nom !");

if (!peer) return alert("Vous devez être connecté d'abord !");

const call = peer.call(name, myStream);

call.on('stream', (remoteStream) => ajoutVideo(remoteStream, name));

const conn = peer.connect(name);

conn.on('open', () => {

connections.push(conn);

// ✅ Maintenant, on reçoit aussi les messages des autres

conn.on('data', (data) => afficherMessage(data.sender, data.message));

});

}

/\* ✅ Partage d'écran \*/

function addScreenShare() {

const name = document.getElementById('share').value.trim();

if (!name) return alert("Veuillez entrer un nom !");

navigator.mediaDevices.getDisplayMedia({ video: true, audio: true })

.then((stream) => {

peer.call(name, stream);

})

.catch(() => alert("Impossible de partager l'écran."));

}

/\* ✅ Envoyer un message \*/

function sendMessage() {

const message = document.getElementById('messageInput').value.trim();

if (!message) return;

afficherMessage("Moi", message);

connections.forEach((conn) => conn.send({ sender: userName, message }));

document.getElementById('messageInput').value = "";

}

/\* ✅ Afficher un message avec nom \*/

function afficherMessage(sender, message) {

const msgContainer = document.getElementById('messages');

const newMsg = document.createElement('p');

newMsg.innerHTML = `<strong>${sender}:</strong> ${message}`;

msgContainer.appendChild(newMsg);

msgContainer.scrollTop = msgContainer.scrollHeight;

}

**Html**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Chat Voice - Hady Kaloga</title>

<link rel="stylesheet" href="style.css">

<script src="https://unpkg.com/peerjs@1.5.2/dist/peerjs.min.js"></script>

</head>

<body>

<!-- ✅ Message d'annonce -->

<div id="announcement">

<p>Bienvenue dans le site chat voice de <strong>Hady Kaloga</strong>. Vos données personnelles sont protégées.</p>

</div>

<!-- ✅ Inscription -->

<div id="register">

<h1>Ton Nom:</h1>

<form onsubmit="register(); return false;">

<input type="text" id="name" required>

<input type="submit" value="Join!">

</form>

</div>

<!-- ✅ Connexion entre utilisateurs -->

<div id="userControls" style="display: none;">

<h1>Se Connecter à un autre utilisateur :</h1>

<form onsubmit="appelUser(); return false;">

<input type="text" id="add" required>

<input type="submit" value="Ajouter">

</form>

<h1>Partager son écran :</h1>

<form onsubmit="addScreenShare(); return false;">

<input type="text" id="share" required>

<input type="submit" value="Partager">

</form>

</div>

<!-- ✅ Zone des vidéos -->

<div id="participants"></div>

<!-- ✅ Chat en temps réel -->

<div id="chat">

<h2>Chat en temps réel</h2>

<div id="messages"></div>

<form onsubmit="sendMessage(); return false;">

<input type="text" id="messageInput" placeholder="Écris un message..." required>

<input type="submit" value="Envoyer">

</form>

</div>

**Css**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap');

body {

font-family: 'Poppins', sans-serif;

background-color: #f4f4f9;

text-align: center;

margin: 0;

padding: 0;

}

h1, h2 {

color: #333;

}

/\* ✅ Annonce toujours visible \*/

#announcement {

background-color: #007bff;

color: white;

padding: 10px;

font-size: 18px;

}

/\* ✅ Conteneur des vidéos \*/

#participants {

display: flex;

flex-wrap: wrap;

justify-content: center;

margin-top: 20px;

gap: 15px;

}

/\* ✅ Style vidéo \*/

.video-container {

display: flex;

flex-direction: column;

align-items: center;

background: white;

border-radius: 10px;

padding: 10px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

}

.video-container video {

width: 200px;

height: 150px;

border-radius: 5px;

background: black;

}

.video-container p {

margin-top: 5px;

font-weight: bold;

}

/\* ✅ Style du chat \*/

#chat {

margin-top: 20px;

padding: 20px;

background: white;

border-radius: 10px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

width: 300px;

display: inline-block;

}

#messages {

height: 200px;

overflow-y: auto;

text-align: left;

padding: 10px;

border: 1px solid #ccc;

background: #f9f9f9;

}

#messages p {

margin: 5px 0;

padding: 5px;

background: #e1f5fe;

border-radius: 5px;

}

#messages p strong {

color: #007bff;

}

form {

margin-top: 10px;

}

input[type="text"] {

padding: 10px;

width: 200px;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 16px;

}

input[type="submit"] {

background-color: #007bff;

color: white;

border: none;

padding: 10px 15px;

cursor: pointer;

border-radius: 5px;

font-size: 16px;

}

input[type="submit"]:hover {

background-color: #0056b3;

}