### **Quá trình thiết lập cuộc gọi WebRTC**

User A vào VideoCall site -> Nhấn Start Call :

* Bắt đầu tạo PeerConnection
* Hiện LocalStream trên web user A
* Tạo Offer gửi đến user B. Offer này là một đối tượng SDP(Session Description Protocol),chứa thông tin về: Các track(audio, video), các codec mà trình duyệt hỗ trợ, các thông số ICE(Interactive Connectivity Establishment) để tìm đường truyền mạng giữa hai thiết bị.
* Chờ User B: Signaling ở trạng thái: have-local-offer

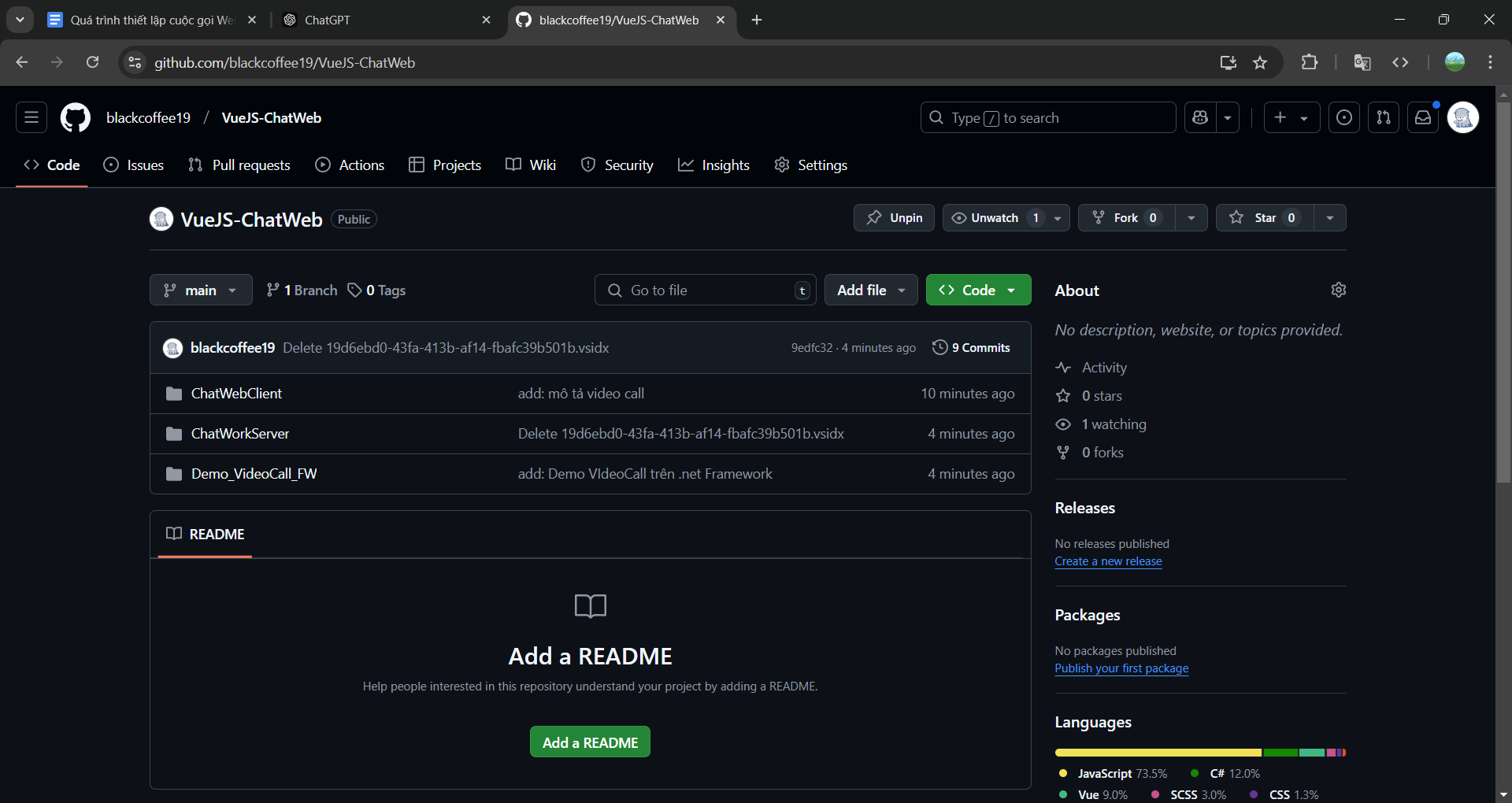
User B nhận được thông báo cuộc gọi, truy cập vào VideoCall site -> Nhấn Accept:

* Xử lý Offer từ A đã nhận được và lưu, tạo Answer và gửi lại cho user A , sau khi nhấn accept bắt đầu truyền Offer và stream cho User A

User A nhận Offer và xử lý thêm Remotestream vào và gửi answer cho user B

Kết nối P2P: Sau khi hoàn tất, hai user có thể giao tiếp trực tiếp (P2P) mà không cần phụ thuộc vào signaling server.

**Hướng dẫn chạy Project:**

****

**File quan trọng thực hiện chức năng Video Call là :**

1. ChatWebClient/src/stores/videocall.js
2. ChatWebClient/src/services/SignalingService.js
3. ChatWebClient/src/views/VideoCallView.vue
4. ChatWorkServer/Services/VideoCallHub.cs
5. ChatWorkServer/Program.cs (Cấu hình SignalR)
6. Demo\_VideoCall\_FW/Models/VideoCallHub.cs
7. Demo\_VideoCall\_FW/Views/Home/index.cshtml
8. Demo\_VideoCall\_FW/Startup.cs (Cấu hình SignalR)

* **ChatWebClient:** Cài đặt:

1. npm install
2. npm run dev

* **ChatWorkServer:** Project ASP.NET Web API đã cài đặt package: Microsoft.AspNet.SignalR và cấu hình SignalR

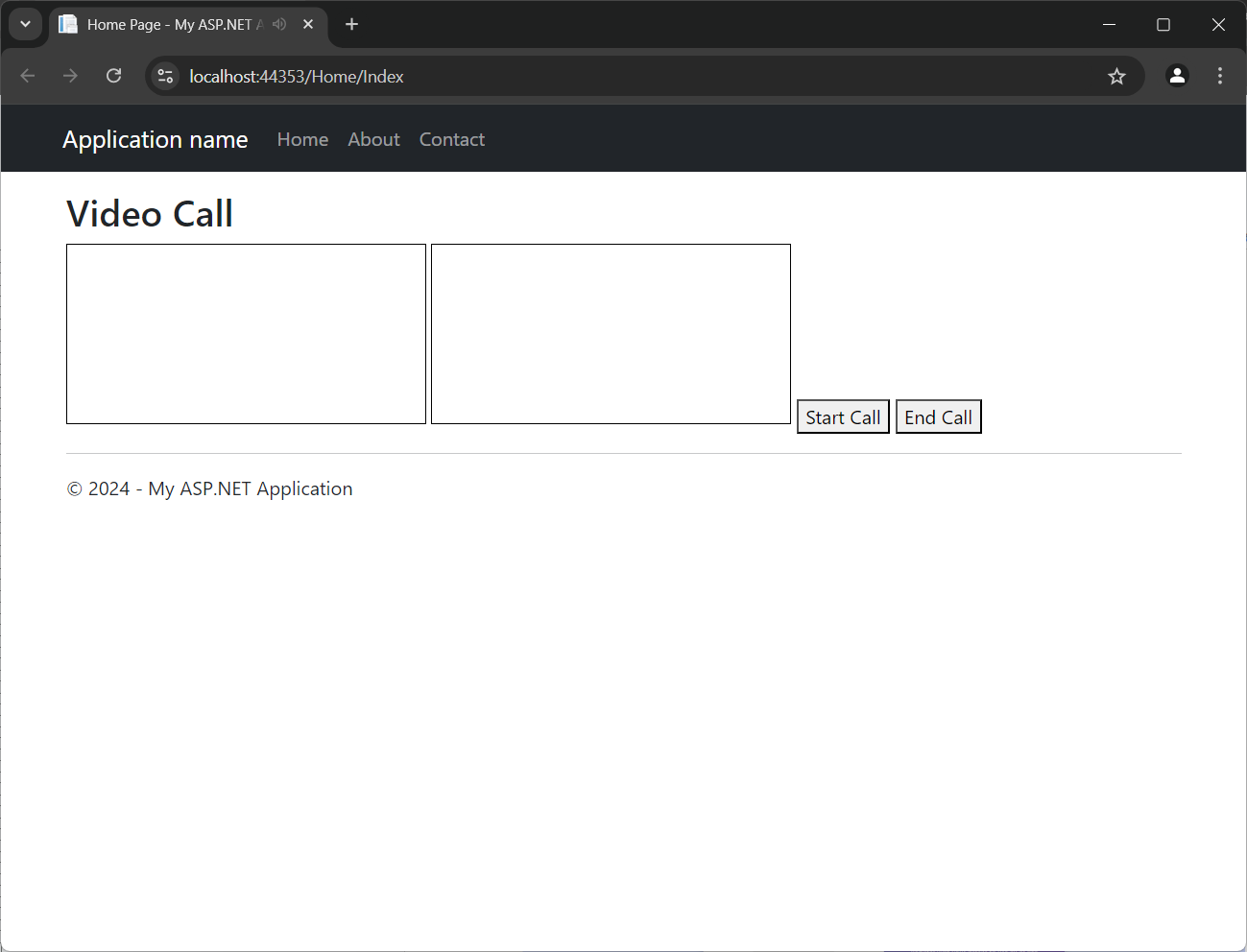
**Cần tạo db ChatDb trước khi chạy ChatWorkServer**

trong *ChatWorkServer/appsettting.json* có Connection String, thay đổi nếu cần để phù hợp với SQL Server

"Data Source=.;Database=ChatDB;User Id=sa;Password=sa@123;TrustServerCertificate=True;"

* **Demo\_VideoCall\_FW:** Project .NET Framework riêng cấu hình **SignalR** đơn giản, không giao diện, khi nhất start bắt đầu cuộc gọi sẽ gửi Offer cho tất cả client

Giao diện của **Demo\_VideoCall\_FW:** đã cấu hình cho **SignalR,** có thể gửi Offer qua websocket sau khi nhất Start Call, tuy nhiên chưa sẵn sàng test

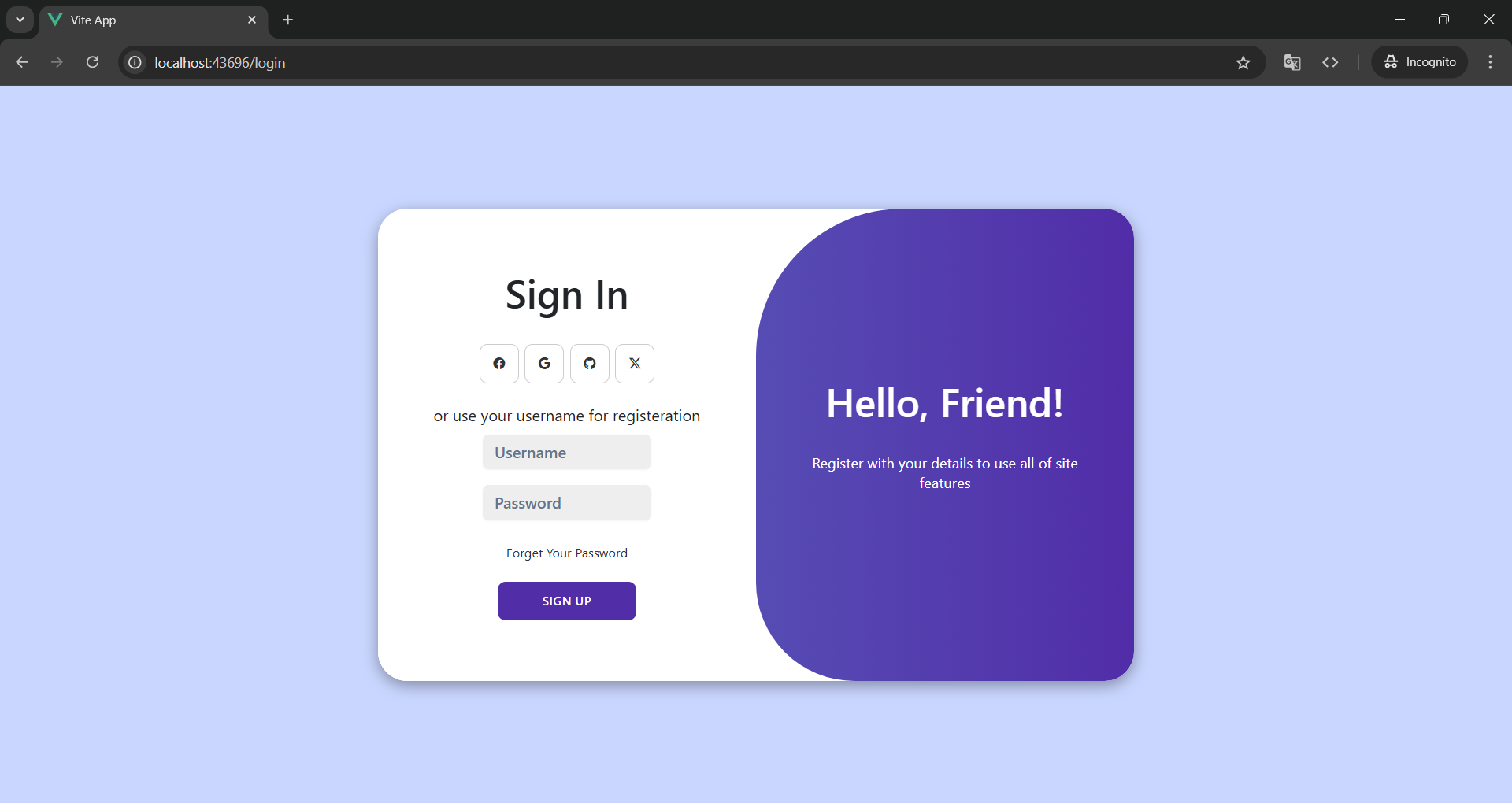
****

**Giao diện khi chạy app ChatWebClient:**

Để test video call chạy song 2 trình duyệt

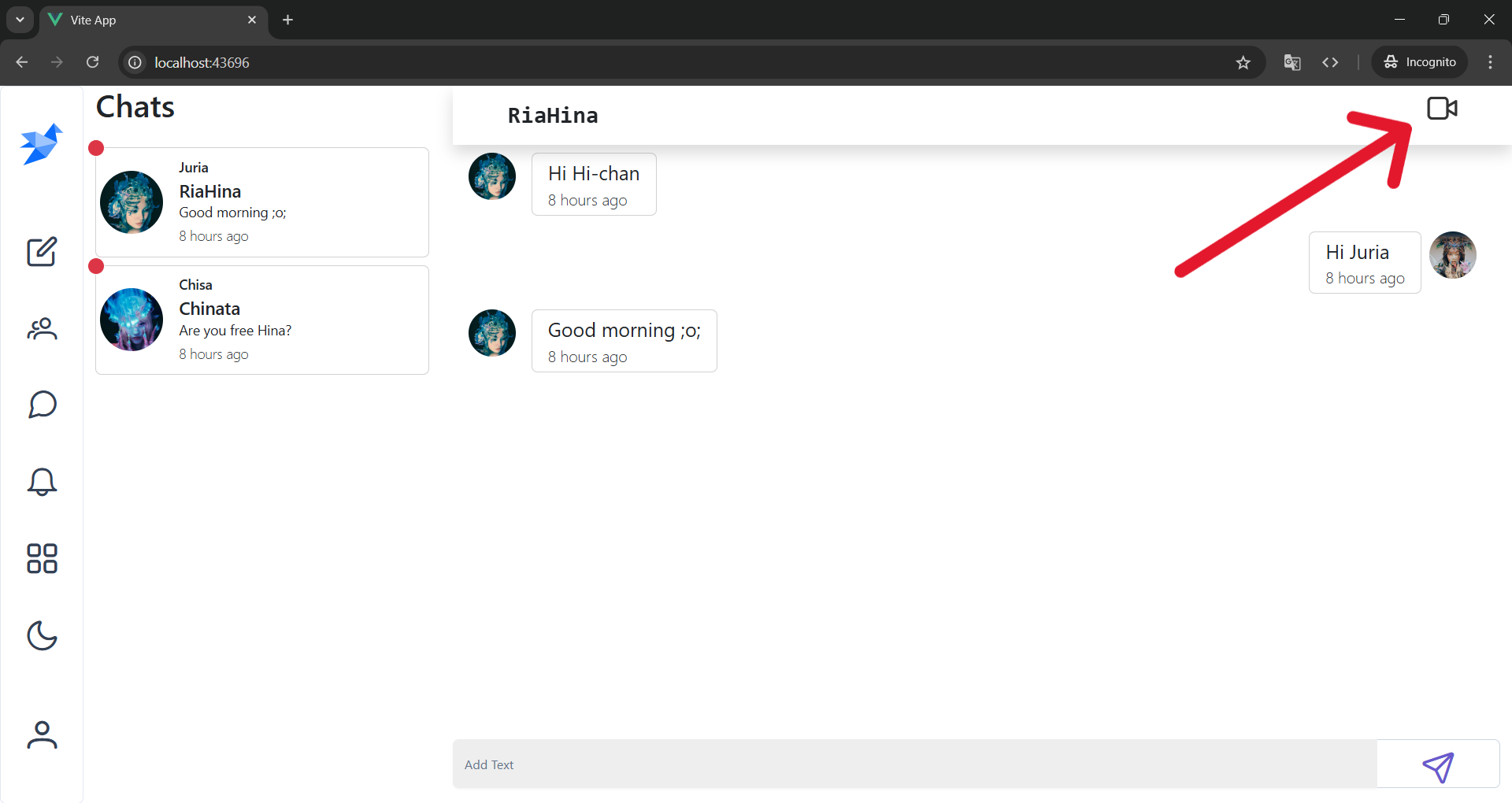
**đăng nhập các user đã seeding sẵn:**

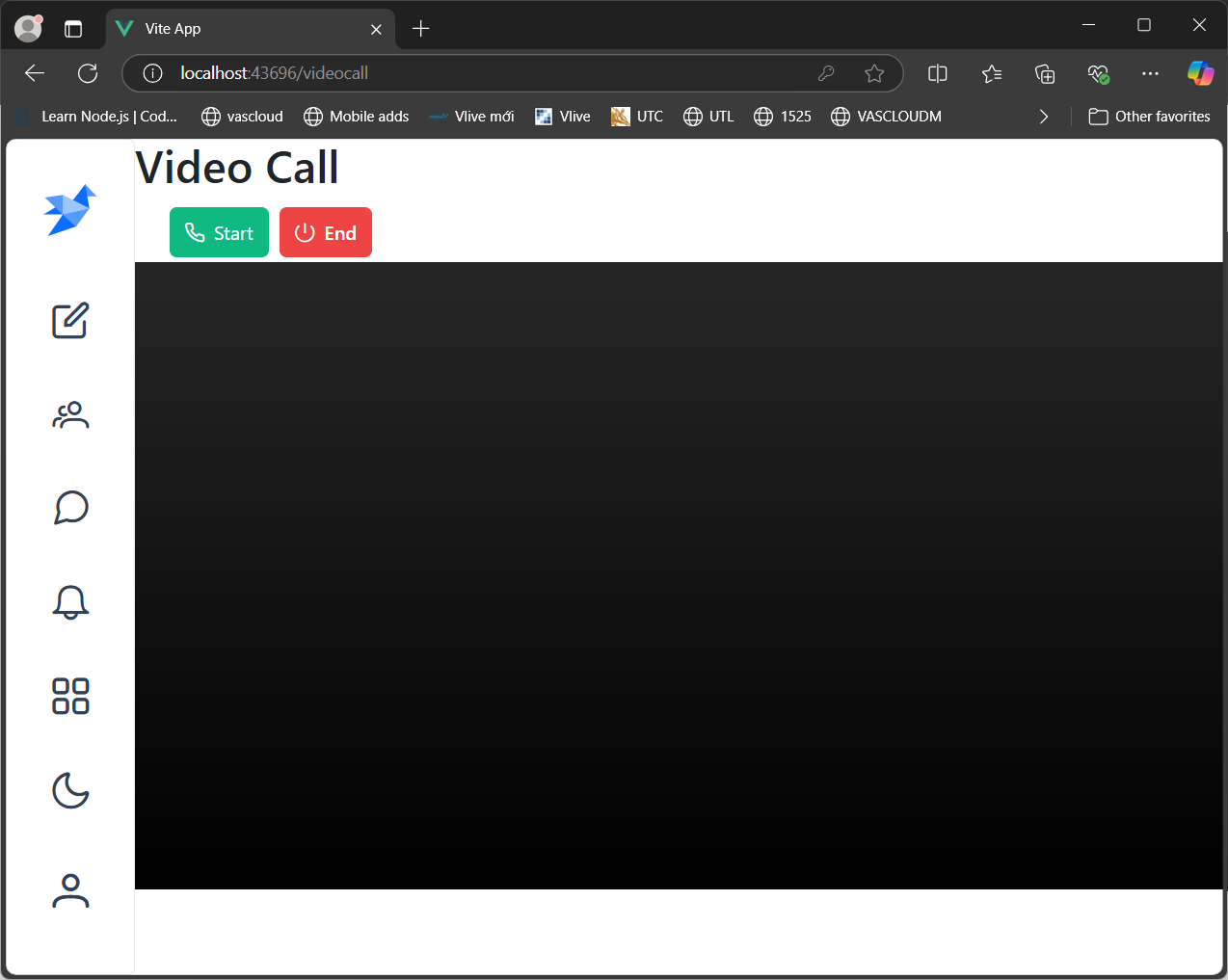
* **username :** hinata **password**: hinata
* **username:** juria **password:** juria
* **username:** chisa **password:** chisa

****

Nhấn chọn 1 user trong sidebar Chats

Trên ChatHeader có nút icon Camera

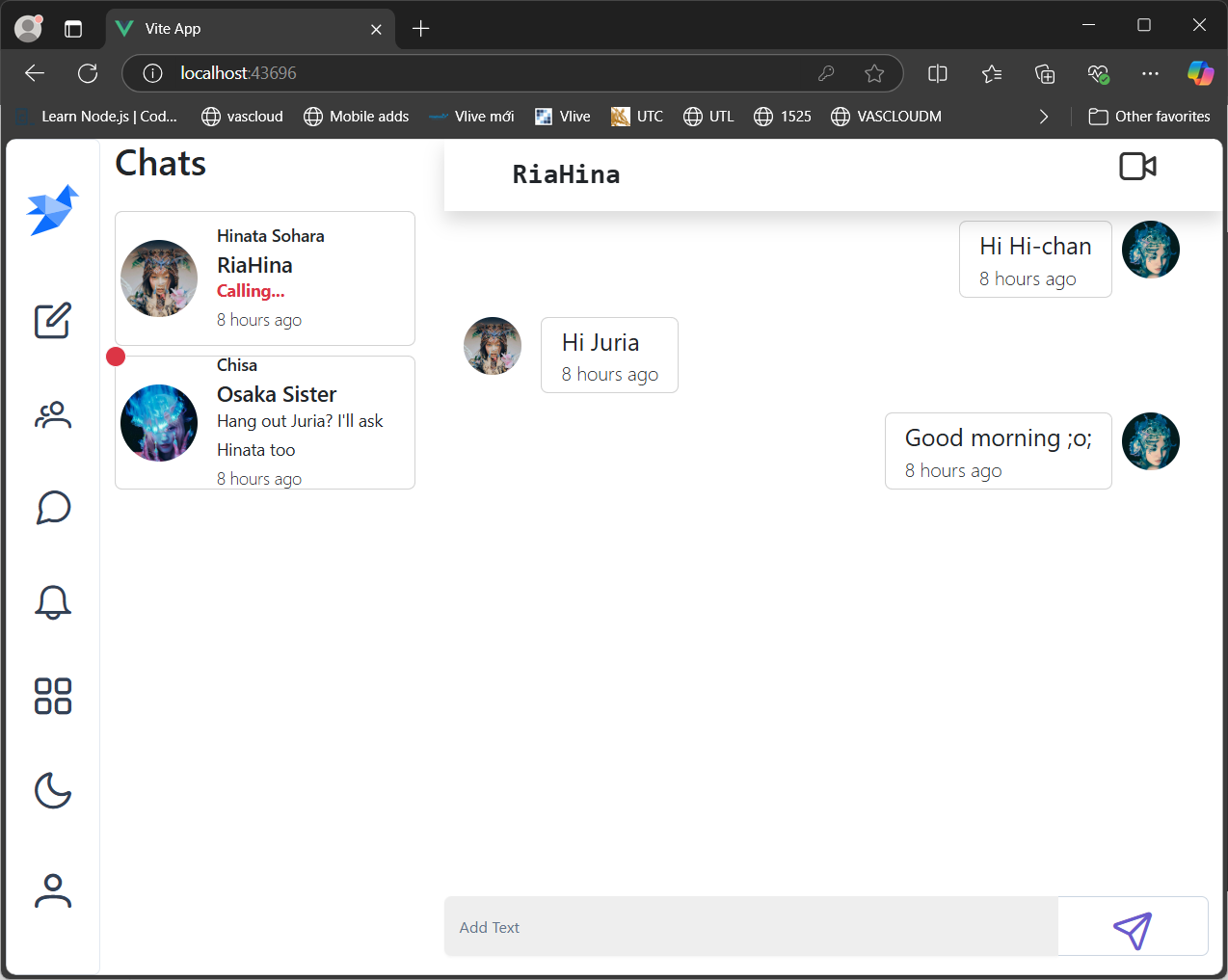




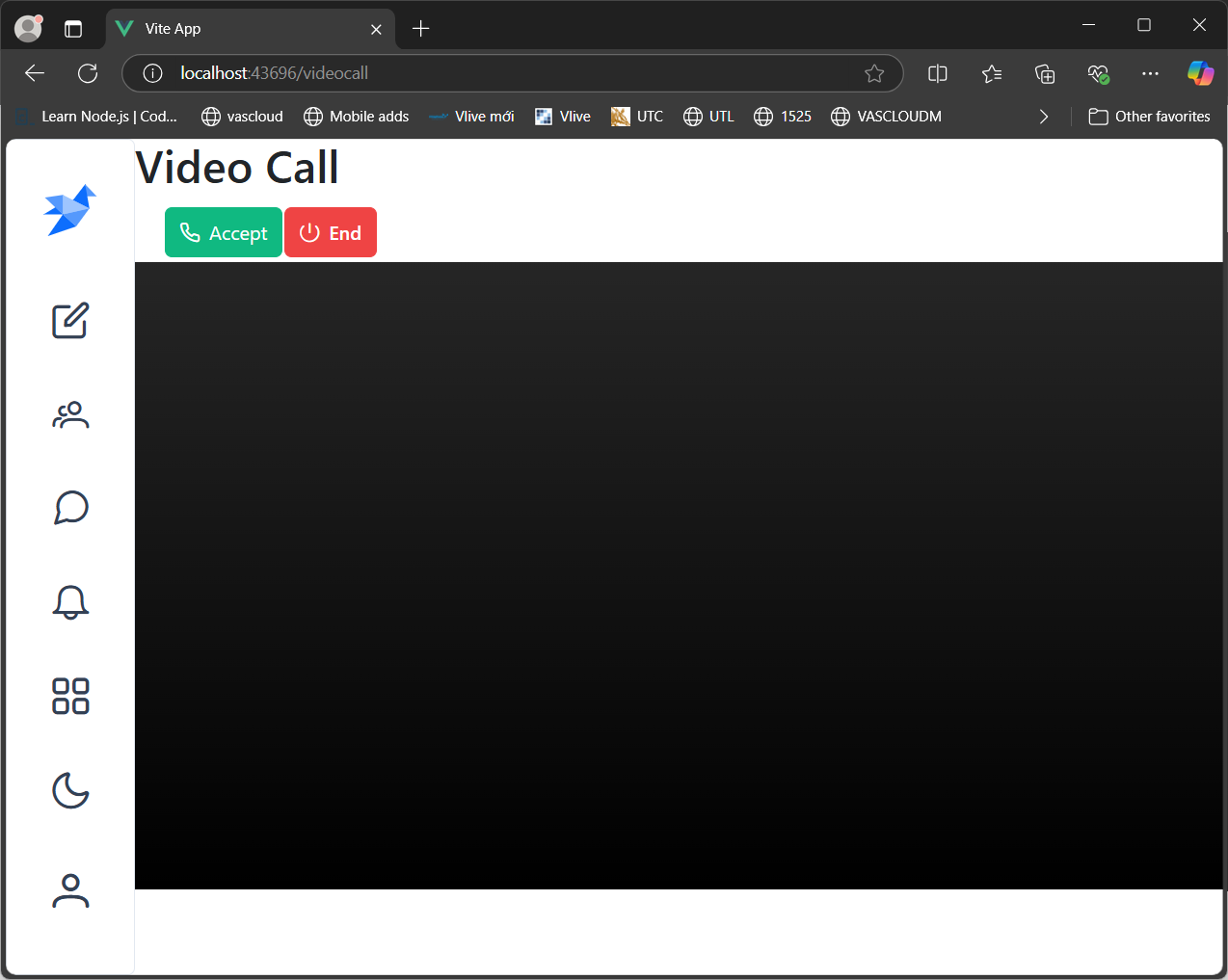
Nhấn Start để bắt đầu gọi video

Ảnh dưới: trong 1 trình duyệt khác: đã đăng nhập 1 tài khoản user khác, khi thấy Calling. nhất vào bật Chat

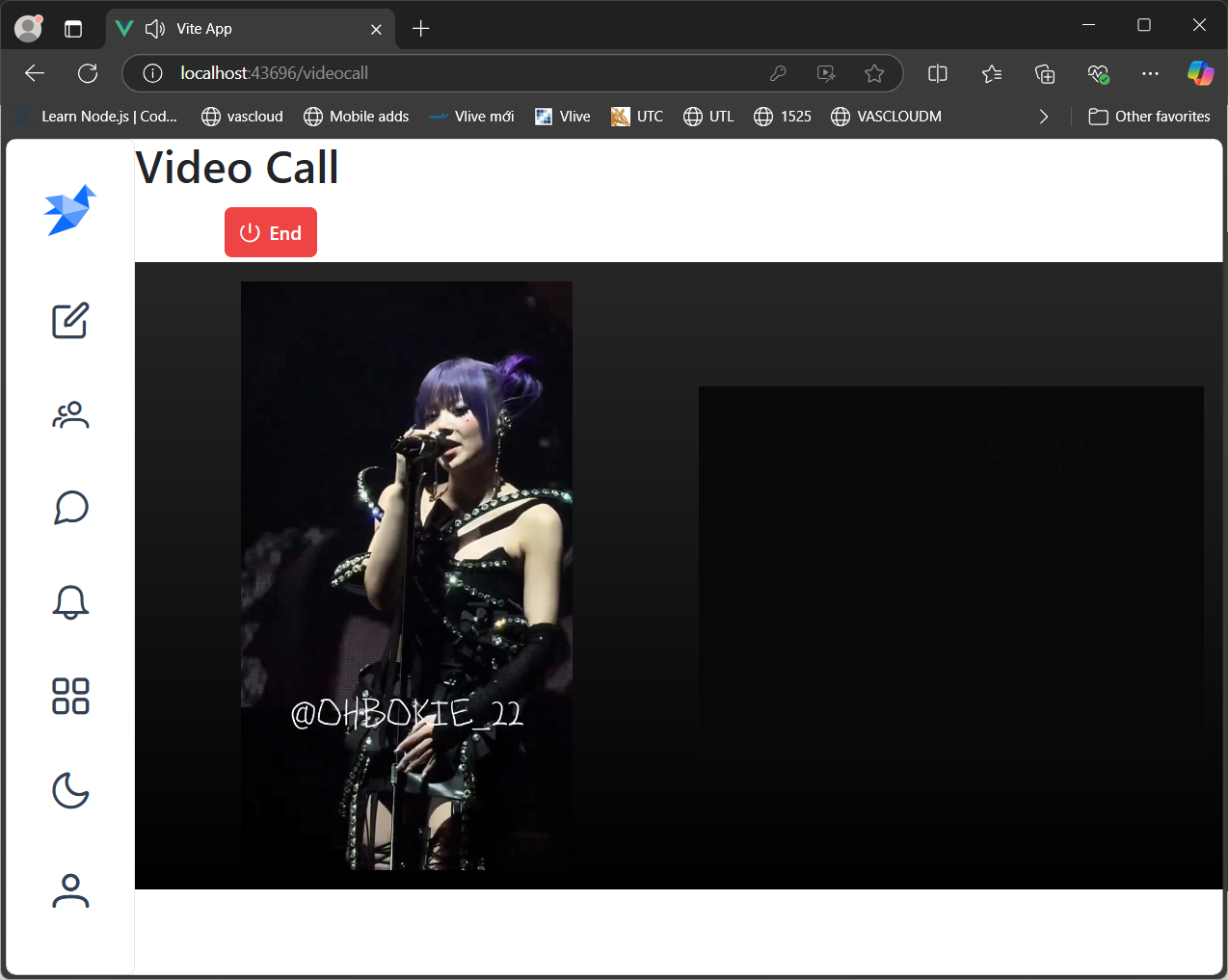
Sau đó nhấn vào Icon Camera



User 2 nhấn Accept để gửi offer và set video stream của User 1

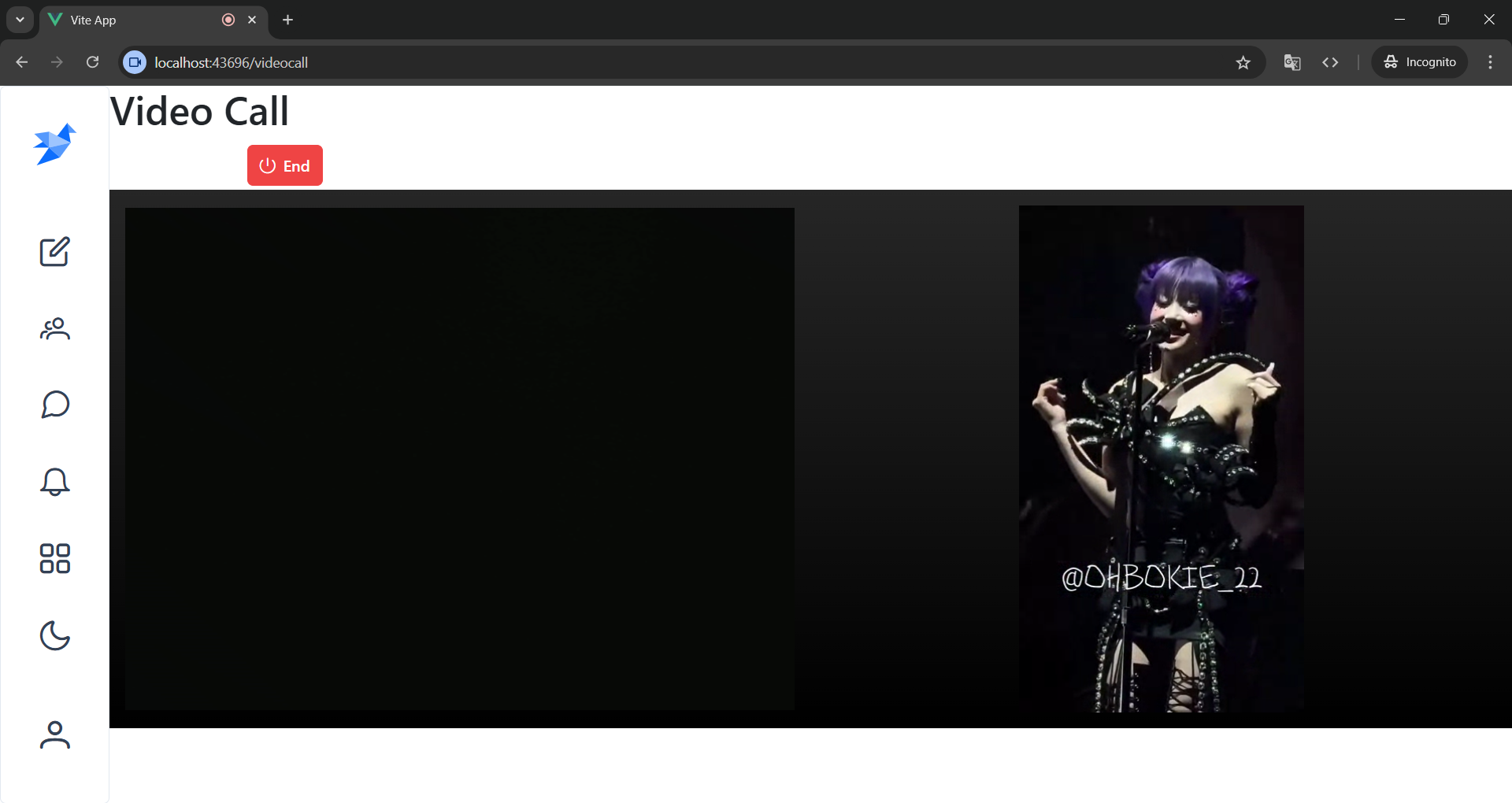


Hiện video mẫu lạ ? do là máy cục bộ và đã mở 1 video call trên trình duyệt User 1 nên dẫn đến lỗi Thiết bị đã hoạt động (camera và mic đã bật ) khi gặp lỗi này sẽ thay stream của máy đó bằng video có sẵn thay vì bật camera và mic



Sau khi 2 trình duyệt (2 user đã kết nối) cả 2 bên đều nhận được và thấy stream của nhau

Trong screenshot, máy User 1 màn hình đen do camera thiết bị bị che đi



**KHÁI NIỆM**

Trong WebRTC, các khái niệm **offer**, **answer**, và **ICE candidate** là các thành phần cơ bản để thiết lập và duy trì kết nối truyền thông thời gian thực (video call, voice call, hoặc dữ liệu). Dưới đây là giải thích chi tiết:

### **1. Offer**

* **Là gì?**
  + Offer là một thông điệp được tạo bởi một trong hai peer (thường là phía bắt đầu cuộc gọi) để khởi tạo một kết nối WebRTC.
  + Nó chứa các thông tin về khả năng phương tiện truyền thông (media capabilities) của người gửi, như:
    - Codec được hỗ trợ (âm thanh, video).
    - Định dạng và cấu hình truyền thông (SDP - Session Description Protocol).
    - Địa chỉ và cổng mạng mà peer có thể nhận dữ liệu.
* **Cách sử dụng:**
  + Peer A tạo offer và gửi nó đến Peer B thông qua **signaling server**.
  + Peer B sẽ dựa vào offer này để hiểu được yêu cầu kết nối từ Peer A.

### **2. Answer**

* **Là gì?**
  + Answer là phản hồi của một peer (thường là Peer B) đối với offer mà họ nhận được.
  + Nó chứa thông tin tương tự như offer nhưng mô tả khả năng truyền thông mà Peer B có thể hỗ trợ.
  + Dựa trên thông tin này, cả hai peer đồng thuận về cách thức kết nối (codec, định dạng, cổng...).
* **Cách sử dụng:**
  + Sau khi nhận được offer từ Peer A, Peer B phân tích và tạo một answer.
  + Answer được gửi lại Peer A thông qua signaling server.
  + Khi cả hai peer có đủ thông tin (offer và answer), kết nối WebRTC sẽ được thiết lập.

### **3. ICE Candidate**

* **Là gì?**
  + ICE (Interactive Connectivity Establishment) candidate chứa thông tin về các địa chỉ và cổng mạng mà một peer có thể sử dụng để kết nối với peer khác.
  + ICE candidate thường bao gồm:
    - Địa chỉ IP và cổng của mạng nội bộ hoặc mạng công khai.
    - Loại candidate (host, relay, hoặc server-reflexive).
    - Giao thức truyền thông (UDP, TCP).
* **Cách sử dụng:**
  + ICE candidate được trao đổi giữa hai peer thông qua signaling server.
  + Khi một peer nhận được ICE candidate, nó sẽ thêm vào kết nối WebRTC và cố gắng thiết lập kết nối dựa trên thông tin đó.
  + ICE giúp các peer vượt qua NAT/firewall và tìm đường truyền tốt nhất.