ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo project 3

NGHIÊN CỨU KỸ THUẬT VÀ XÂY DỰNG WEBRTC VIDEOCALL

Ngành Công nghệ thông tin và truyền thông

Giảng viên hướng dẫn: TS. Trần Quang Đức _____

Chữ kí GVHD

Sinh viên thực hiện: Nguyễn Ngọc Ánh 20205228

Khoa: Công nghệ thông tin Việt-Pháp

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 12/2023

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. GIỚI THIỆU

1.1 Bối cảnh thực hiện đề tài

Trong thời đại công nghệ số hiện nay, việc giao tiếp và chia sẻ thông tin trở nên dễ dàng hơn bao giờ hết. Tuy nhiên, với sự phát triển không ngừng của công nghệ, nhu cầu của người dùng cũng ngày càng tăng cao. Người dùng không chỉ muốn giao tiếp và chia sẻ thông tin một cách nhanh chóng và tiện lợi, mà còn muốn đảm bảo an toàn và bảo mật cho thông tin của họ.

Trong bối cảnh này, việc tạo ra một ứng dụng chat video trực tuyến đáp ứng được những yêu cầu trên trở thành một vấn đề cần giải quyết. Đặc biệt, với sự phổ biến của các thiết bị di động và sự tiện lợi của việc sử dụng trình duyệt web, việc tạo ra một ứng dụng chat video trực tuyến hoạt động trên trình duyệt web trở thành một xu hướng không thể tránh khỏi.

1.2 Vấn đề cần giải quyết

Vấn đề cần giải quyết trong đề tài này là tạo ra một ứng dụng chat video trực tuyến hoạt động trên trình duyệt web, cho phép người dùng tạo ra hoặc tham gia vào các phòng chat video, gửi tin nhắn văn bản và chia sẻ màn hình của họ trong quá trình chat video. Ứng dụng cần đảm bảo an toàn và bảo mật cho thông tin của người dùng, và cung cấp một giao diện người dùng thân thiện và dễ sử dụng.

1.3 Đề xuất nội dung thực hiệni

Để giải quyết vấn đề trên, chúng tôi đề xuất nội dung thực hiện như sau:

1. Phân tích yêu cầu: Xác định rõ ràng các yêu cầu của người dùng và các tính năng cần có của ứng dụng.

2. Thiết kế giao diện người dùng: Thiết kế một giao diện người dùng thân thiện và dễ sử dụng, với các thành phần giao diện như khung chat, danh sách thành viên, và video.

3. Xây dựng ứng dụng: Sử dụng công nghệ WebRTC để xây dựng ứng dụng chat video trực tuyến hoạt động trên trình duyệt web. Các tính năng cần xây dựng bao gồm tạo phòng chat, tham gia phòng chat, gửi tin nhắn, chat video, và chia sẻ màn hình.

4. Kiểm thử và cải tiến: Kiểm thử ứng dụng để đảm bảo rằng nó hoạt động đúng như mong đợi, và cải tiến ứng dụng dựa trên phản hồi của người dùng.

1.4 References

NONE

CHƯƠNG 2. Nghiên cứu tổng quan

2.1 Các công nghệ sử dụng chính

2.1.1 Mạng ngang hàng (P2P) và các ứng dụng

a, Giới thiệu

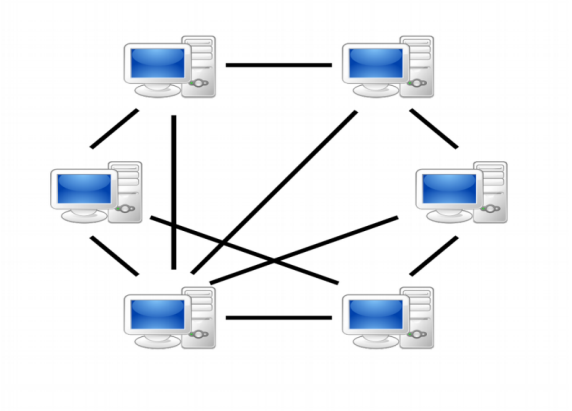
- Mạng ngang hàng (P2P) hay còn gọi là Peer-To-Peer bắt đầu xuất hiện từ năm 1999 và đã thu hút sự quan tâm của giới công nghệ thông tin trong những năm gần đây. Đặc biệt, việc áp dụng các mô hình P2P trong việc xây dựng những ứng dụng chia sẻ file (file sharing), video call, điện thoại trên nền tảng Internet (Internet-based telephony) dã đạt được nhiều thành công.

- Hiện nay, các ứng dụng P2P chiếm khoảng 50% (thậm chí 75%) băng thông trên Internet.

- Các ứng dụng của kiểu mạng này như là: Napster,Skype, BitTorrent, v.v..

b, Định nghĩa

- Mạng ngang hàng là một kiểu mạng được thiết kế cho các thiết bị trong đó có chức năng và khả năng của các thiết bị là như nhau. - Mạng P2P không có khái niệm trạm (client) hay máy chủ (server) mà chỉ có khái niệm các nốt (peers) đóng vai trò như cả client và server.



Hình 2.1: Mô hình mạng ngang hàng (P2P)

- Mạng ngang hàng là một hệ thống phân tán đặc biệt trong tầng ứng dụng, ở đó mỗi cặp điểm nút có thể giao tiếp với nhau thông qua giao thức định tuyến trong các tầng mạng ngang hàng. Mỗi điểm nút giữ một đối tượng dữ liệu nào đó có thể là nhạc, ảnh tài liệu, v.v… Mỗi điểm nút có thể truy vấn tới đối tượng nó cần từ các điểm nút khác thông qua kết nối logic trong tầng mạng ngang hàng

c, So sánh mô hình Client-Server và mô hình Peer-To-Peer:

- Ưu điểm

Client-Server	**P2P**
- Tốc độ truy cập nhanh. - Khả năng mở rộng cao. - Hoạt động với bất kỳ loại ứng dụng nào. - Sử dụng được với các ứng dụng chia sẻ CSDL. - Đáng tin cậy (có server riêng). - Mức độ an toàn cao nhất.	- Không cần Server riêng, các nốt chia sẻ tài nguyên. - Khi mạng càng mở rộng, khả năng hoạt động càng tốt. - Chi phí thấp. - Dễ cài đặt và bảo trì. - Thuận lợi cho việc chia sẻ file, máy in, CD-ROM, v.v...

d, Mục đích và ứng dụng của mạng P2P

- Mục đích: Mạng ngang hàng hoạt động chủ yếu dựa vào khả năng tính tính toán và băng thông của các máy tham gia chứ không tập trung vào một số nhỏ các server trung tâm như các mạng thông thường. Tất cả các máy trong mạng đều tham gia đóng góp tài nguyên, bao gồm băng thông, lưu trữ và khả năng tính toán nên càng nhiều máy tham gia thì khả năng của mạng càng mạnh.

-Ứng dụng: Sự ra đời của mạng ngang hàng đã tạo ra cách thức quản lí mới cho hàng loạt các lĩnh vực ứng dụng như: giao tiếp (communication), chia sẻ file (flie sharing), băng thông (bandwidth), vấn đề lưu trữ (storage), các chu trình xử lí (processor cycles)

2.1.2 WEBRTC

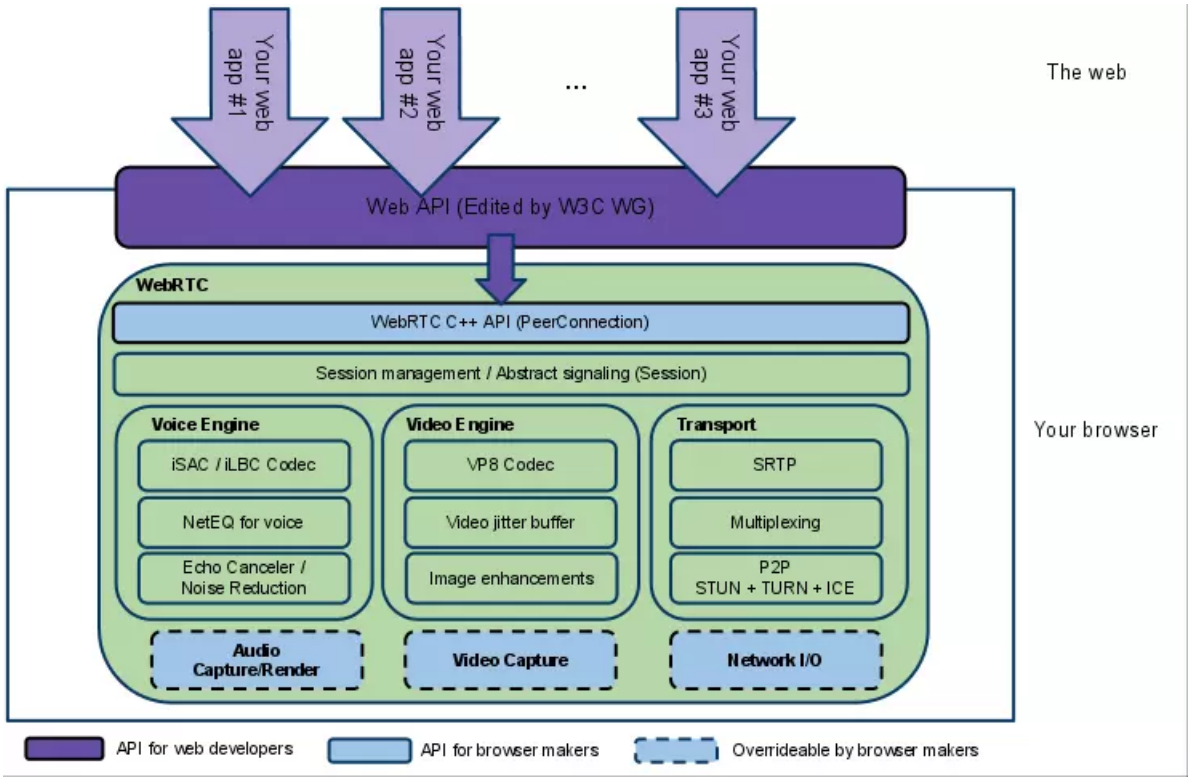
a, Tổng quan

WebRTC là một nỗ lực để xây dựng một framework mở có khả năng giao tiếp audio và video thời gian thực, nó có thể biến các trình duyệt web thành một nền tảng cho giao tiếp giữa người với người. Giao tiếp thời gian thực trong trình duyệt web đã có trước đây tuy nhiên chúng ta phải cài đặt phần mềm của bên thứ ba lên trình duyệt web. WebRTC mang lại hỗ trợ giao tiếp thời gian thực từ ngay bên trong các trình duyệt web và các nhà phát triển web có thể sử dụng một cách tự do thông qua các JavaScript API tiêu chuẩn. Điều này mang lại giao tiếp thời gian thực như là một tính năng cho web, có thể thúc đẩy sự đổi mới hơn nữa. WebRTC (Web Real-Time Communications) là một tập hợp các hàm lập trình dùng cho việc liên

lạc thời gian thực bằng video, âm thanh cũng như các loại dữ liệu khác. WebRTC có thể giúp chúng ta gọi điện video ngay trong trình duyệt mà không cần đăng kí tài khoản, cũng không cần cài thêm plugin gì phức tạp, ngoài ra chúng còn được xài để phát triển game chơi trực tiếp trong trình duyệt và rất nhiều loại ứng dụng khác. WebRTC là gì? , người ta đang dùng nó ra sao và những trở ngại nào đang hiện hữu với chuẩn này.

b, Kiến trúc webrtc

Kiến trúc tổng quan của webRTC như sau:



Hình 2.2: Kiến trúc của WebRTC

Có 2 lớp riêng biệt (distinct layers):

- Browser deveopers sẽ quan tâm đến WebRTC C++ API và các thành phần core sâu hơn của nó như Voice Engine, Video Engine, Transform. Hay dễ hiểu hơn đó là âm thanh, video và kết nối mạng.

- Web App developers sẽ quan tâm tới Web API.

- **Your Web App**

Một ứng dụng phát triển bởi các developer bên thứ 3 với video và audio chat, xây dựng dựa trên Web API để kết nối thời gian thực.

- **Web API**

Một API được sử dụng bởi các developer bên thứ 3, để phát triển web video.

- **WebRTC Native C++ API**

Một tầng API cho phép trình duyệt dễ dàng thực thi Web API.

- **Transport / Session**

Các session component được xây dựng bởi việc sử dụng lại các component từ libjingle, không yêu cầu hoặc sử dụng giao thức xmpp/jingle.

- **RTP Stack**

Một network stack cho RTP (Real Time Protocol).

- **STUN/ICE**

Một component cho phép các cuộc gọi sử dụng STUN và ICE để thiết lập kết nối thông qua các loại networks khác nhau.

- **Session Management**

Một lớp session trừu tượng (abstracted session layer) cho phép thiết lập cuộc gọi và lớp quản lý.

- **VoiceEngine**

VoiceEngine là một framework cho audio media chain, từ card âm thanh tới mạng.

- **iSAC / iLBC / Opus**

Đây là những audio codec, đơn giản hiểu là bộ mã hóa và giải mã tín hiệu âm thanh.

- **Acoustic Echo Canceler (AEC)**

The Acoustic Echo Canceler là một phần mềm dựa trên các thành phần xử lý tín hiệu đã được xóa. Trong real time, acoustic cho kết quả từ voice được chạy tới microphone đang hoạt động.

- **Noise Reduction (NR)**

Noise Reduction component là một phần mềm dựa trên các thành phần xử lý tín hiệu, nhằm loại bỏ các loại tiếng ồn kết hợp với VoIP. (Hiss, fan noise, etc. . .)

- **VideoEngine**

VideoEngine là một framework video media chain cho video, từ camera tới mạng, và từ mạng tới màn hình.

- **VP8**

Video codec từ dự án WebM. Nó phù hợp với RTC như một thiết kế cho độ trễ thấp (low latency).

- **Video Jitter Buffer**

Jitter Buffer động cho video. Giúp che giấu ảnh hưởng của jitter và packet bị mất trong toàn bộ chất lượng video.

- **Image enhancements**

Ví dụ như xóa tiếng ồn video từ ảnh quay bởi webcam.

2.2 Các công nghệ khác

1. HTML/CSS/JavaScript: Đây là ngôn ngữ cơ bản để xây dựng giao diện và chức năng của ứng dụng.

2. Socket.IO: Đây là thư viện JavaScript cho phép realtime, bi-directional communication giữa web clients và servers. Nó được sử dụng trong project này để xử lý các sự kiện realtime như khi có người dùng mới tham gia phòng, hoặc khi có tin nhắn mới.

3. jQuery: Đây là thư viện JavaScript giúp đơn giản hóa việc viết JavaScript, đặc biệt là các tác vụ liên quan đến DOM manipulation và AJAX requests

CHƯƠNG 3. PHÂN TÍCH YÊU CẦU

3.1 Phân tích thiết kế hệ thống

3.1.1 Phân tích Yêu cầu

a, Yêu cầu chức năng

- Tạo phòng chat: Người dùng có thể tạo một phòng chat mới với một ID duy nhất.

- Tham gia phòng chat: Người dùng có thể tham gia vào một phòng chat bằng cách nhập ID phòng.

- Truyền và nhận video/audio: Người dùng có thể truyền và nhận dữ liệu video và audio trong thời gian thực.

- Chat văn bản: Người dùng có thể gửi và nhận tin nhắn văn bản.

- Chia sẻ màn hình: Người dùng có thể chia sẻ màn hình của họ với các thành viên khác trong phòng.

b, Thiết kế Hệ thống

- Front-end: Sử dụng HTML/CSS/JavaScript để xây dựng giao diện người dùng. Sử dụng thư viện React hoặc Angular để quản lý state và cập nhật UI một cách hiệu quả.

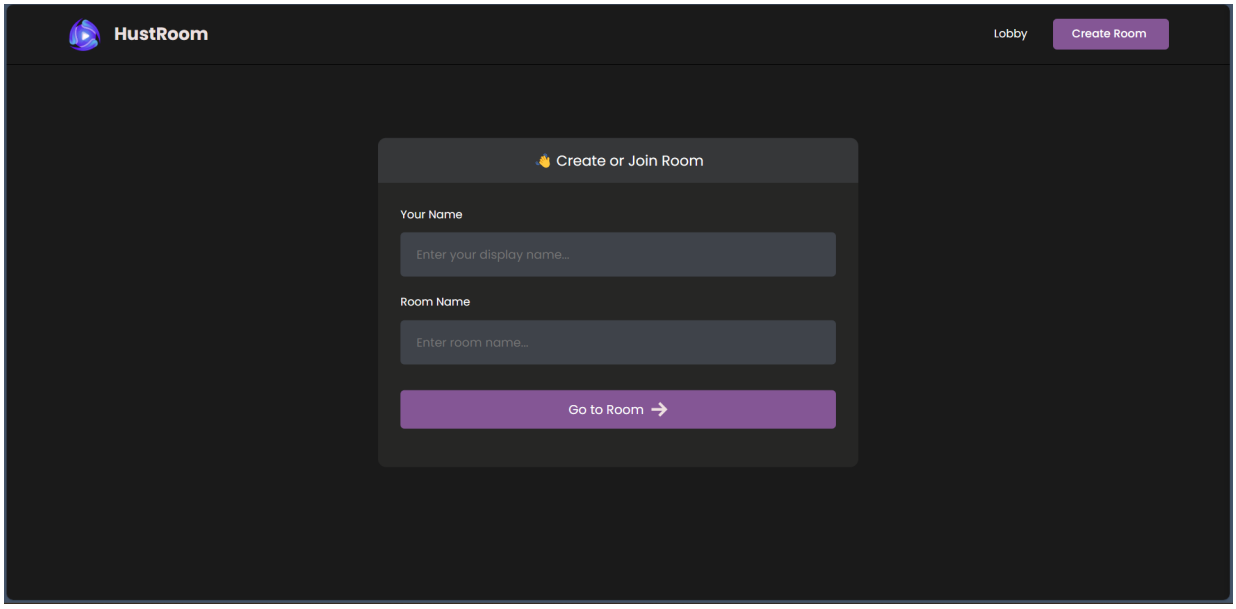
- Back-end: Node.js sẽ được sử dụng để xây dựng server. Express.js có thể được sử dụng để xử lý các yêu cầu HTTP và WebSocket.

- WebRTC: Để xử lý truyền tải video và audio.

- Socket.IO: Để xử lý real-time messaging và signaling.

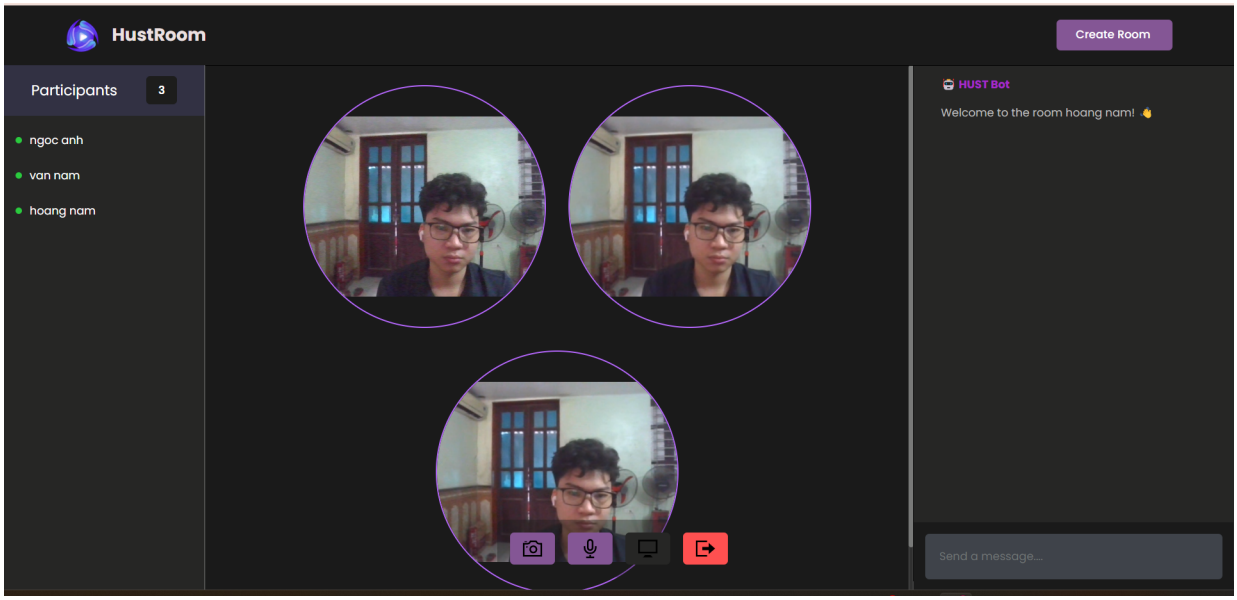
c, Thiết kế Giao diện người dùng (UI)

- Trang chủ : Nơi mà người dùng sẽ nhập tên và tên phòng , nếu phòng chưa đc tạo sẽ tự động tạo và nếu phòng đã taoh rồi sẽ join phòng



Hình 3.1: Giao diện trang chủ

- Phòng chat: Giao diện chat video, bao gồm video của người dùng, danh sách người dùng trong phòng, và khung chat văn bản.



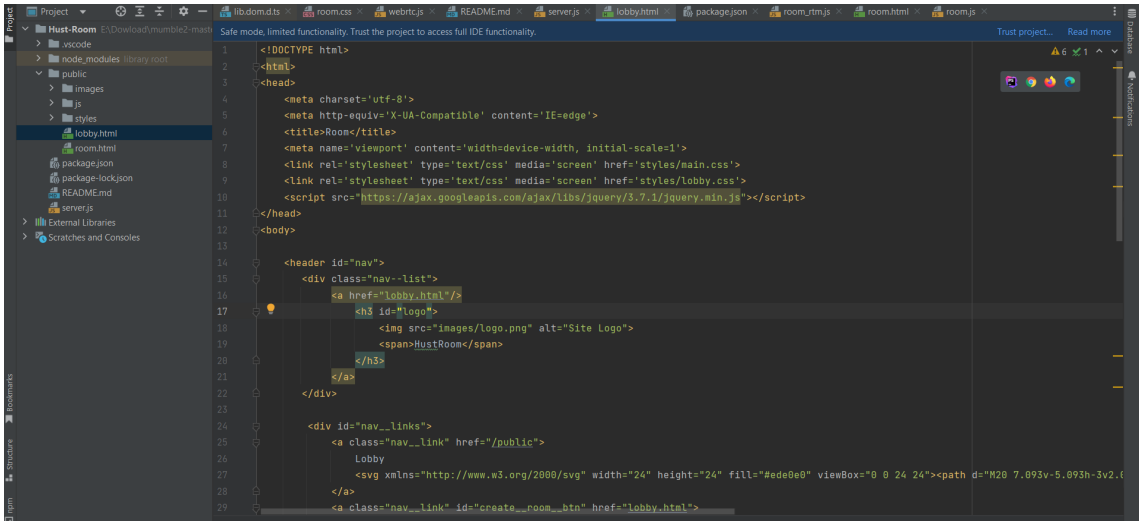
Hình 3.2: Giao diện phòng chat

3.2 Phân tích cấu trúc dự án

3.2.1 Thành phần giao diện người dùng

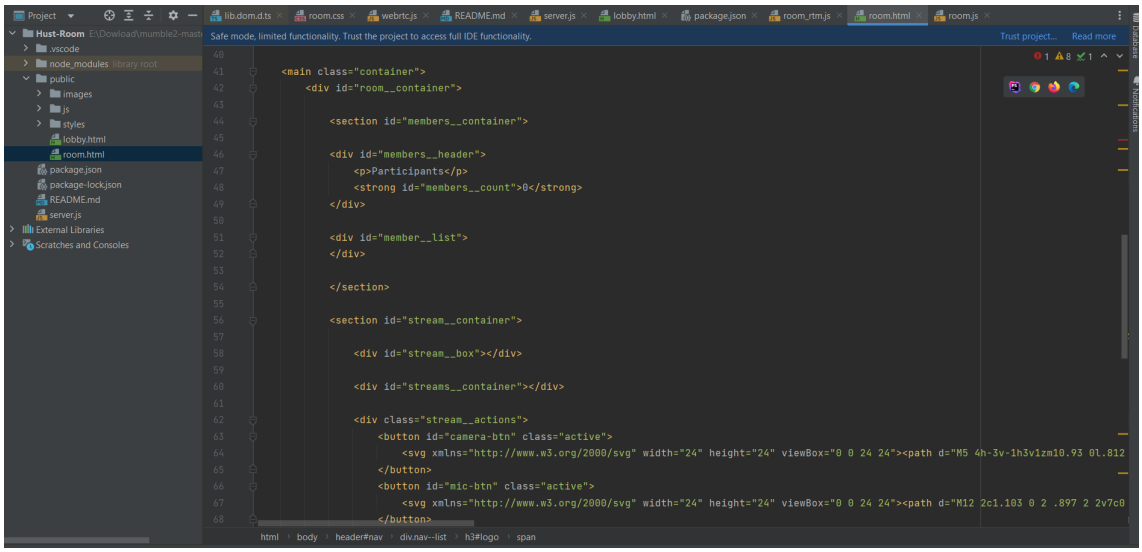
a, HTML Files:

- Lobby.html: Nơi để người dùng tạo hoặc tham gia phòng.



Hình 3.3: Lobby.html

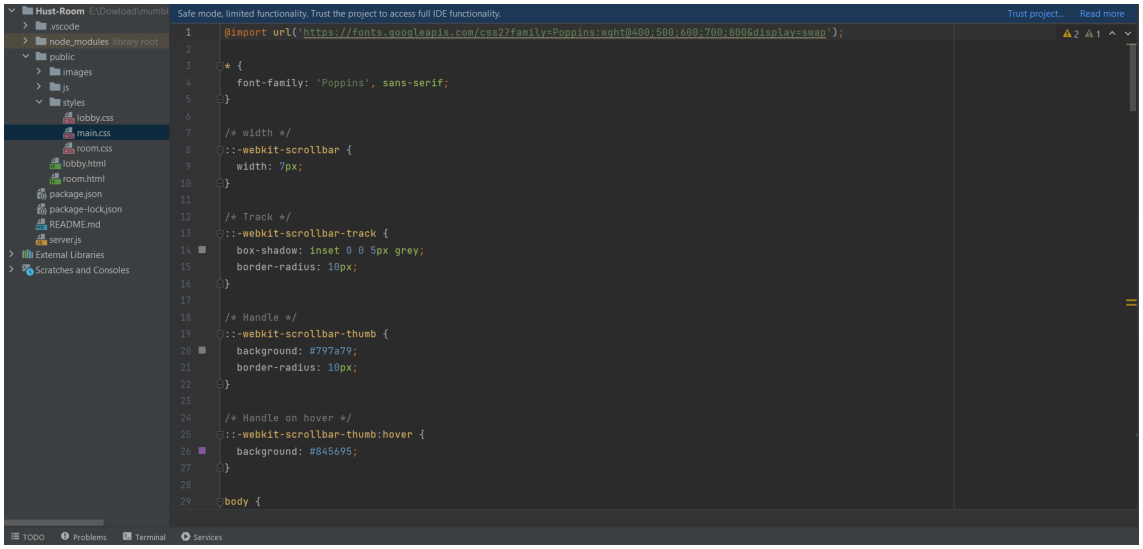
- Room.html: Giao diện cho phòng trò chuyện video nơi diễn ra giao tiếp thực tế.



Hình 3.4: Room.html

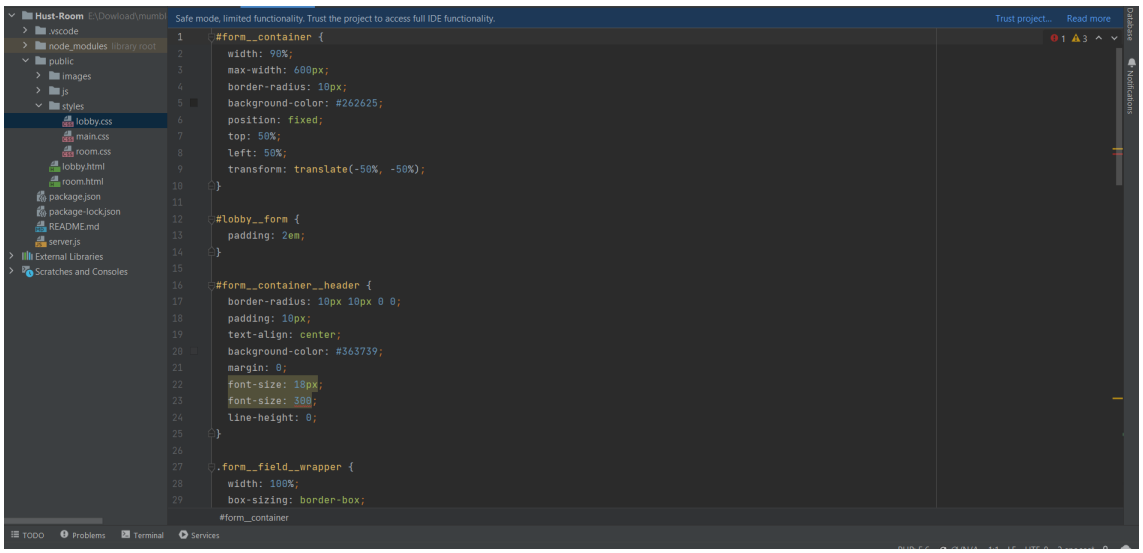
b, CSS Files:

- main.css: Chứa kiểu dáng chính cho ứng dụng.

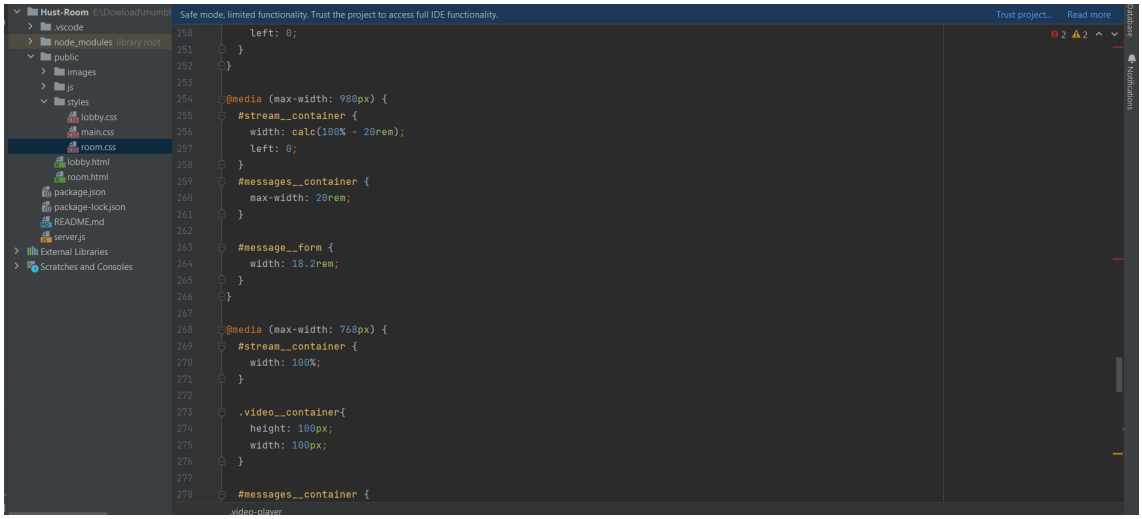


Hình 3.5: main.css

- Lobby.css và room.css: Cung cấp các kiểu cụ thể cho chế độ xem sảnh và phòng.



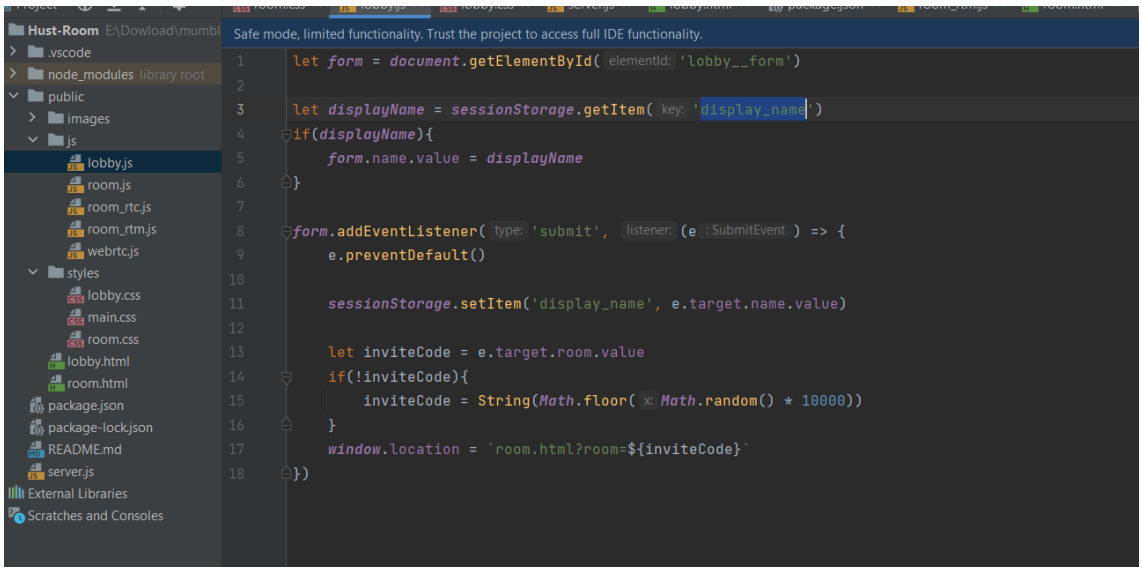
Hình 3.6: Lobby.css



Hình 3.7: room.css

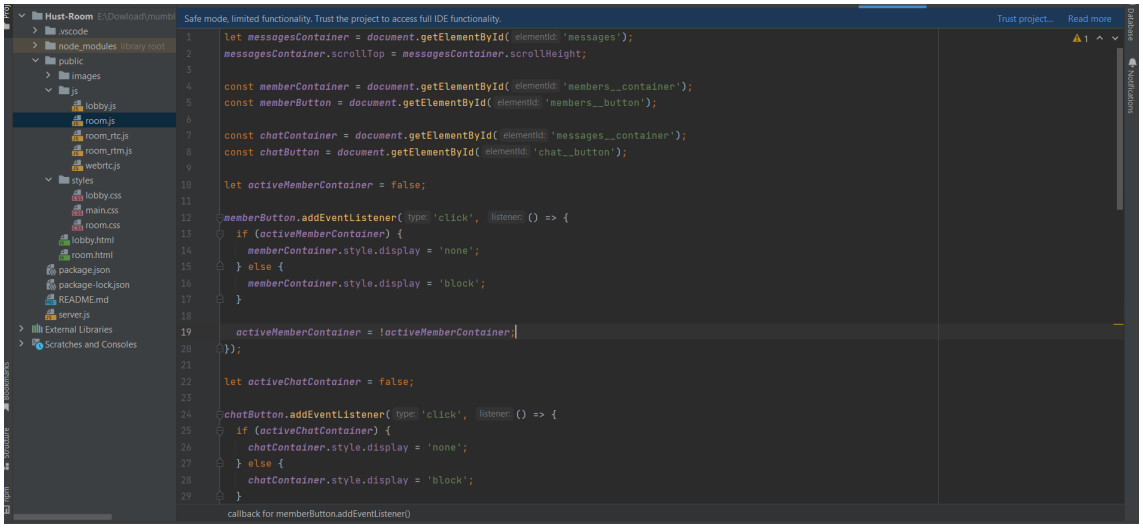
c, JavaScript Files:

- Lobby.js: Xử lý các tương tác ở sảnh, chẳng hạn như tạo hoặc tham gia một phòng.



Hình 3.8: Lobby.js

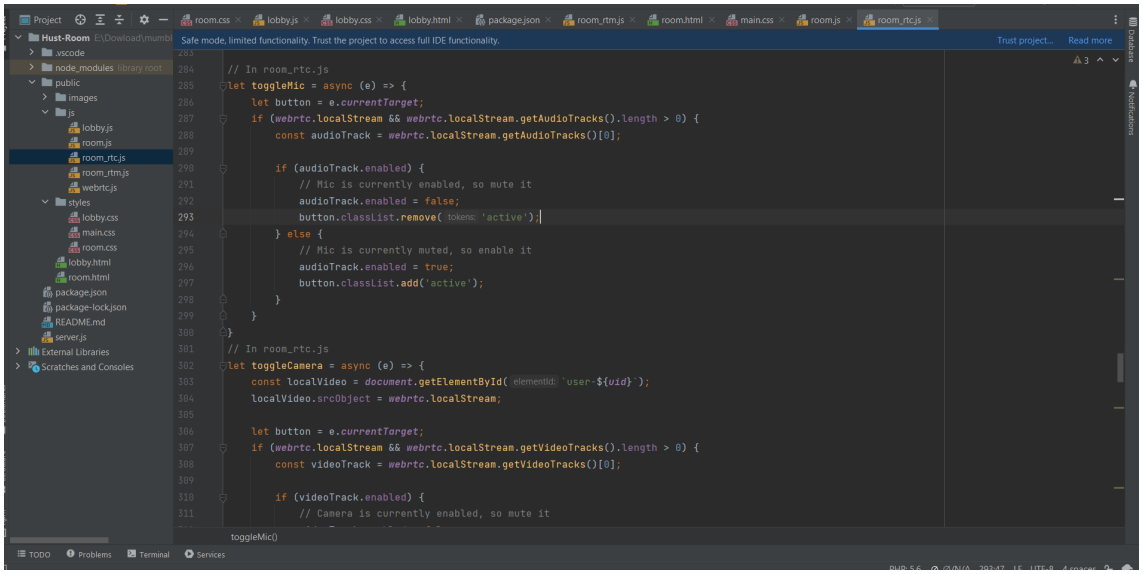
- room.js: Quản lý các tương tác ở cấp phòng và cập nhật giao diện người dùng.



```
let messagesContainer = document.getElementById( elementId: 'messages');
messagesContainer.scrollTop = messagesContainer.scrollHeight;

const memberContainer = document.getElementById( elementId: 'members__container');
const memberButton = document.getElementById( elementId: 'members__button');

const chatContainer = document.getElementById( elementId: 'messages__container');
const chatButton = document.getElementById( elementId: 'chat__button');

let activeMemberContainer = false;

memberButton.addEventListener( type: 'click', listener: () => {
  if (activeMemberContainer) {
    memberContainer.style.display = 'none';
  } else {
    memberContainer.style.display = 'block';
  }

  activeMemberContainer = !activeMemberContainer;
});

let activeChatContainer = false;

chatButton.addEventListener( type: 'click', listener: () => {
  if (activeChatContainer) {
    chatContainer.style.display = 'none';
  } else {
    chatContainer.style.display = 'block';
  }
```

Hình 3.9: room.js

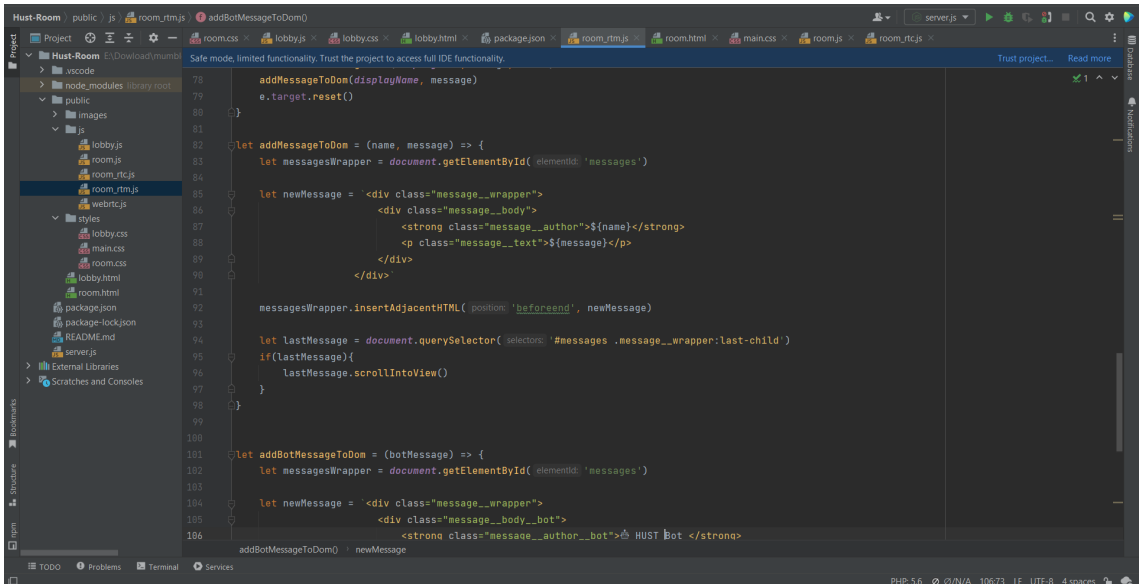
- room_rtc.js: Quản lý các kết nối ngang hàng WebRTC, đường truyền phương tiện và tín hiệu cho giao tiếp video/âm thanh.



```
// In room_rtc.js
let toggleMic = async (e) => {
    let button = e.currentTarget;
    if (webrtc.localStream && webrtc.localStream.getAudioTracks().length > 0) {
        const audioTrack = webrtc.localStream.getAudioTracks()[0];

        if (audioTrack.enabled) {
            // Mic is currently enabled, so mute it
            audioTrack.enabled = false;
            button.classList.remove( tokens: 'active');
        } else {
            // Mic is currently muted, so enable it
            audioTrack.enabled = true;
            button.classList.add('active');
        }
    }
}
// In room_rtc.js
let toggleCamera = async (e) => {
    const localVideo = document.getElementById( elementId: `user-${uid}`);
    localVideo.srcObject = webrtc.localStream;

    let button = e.currentTarget;
    if (webrtc.localStream && webrtc.localStream.getVideoTracks().length > 0) {
        const videoTrack = webrtc.localStream.getVideoTracks()[0];

        if (videoTrack.enabled) {
            // Camera is currently enabled, so mute it
```

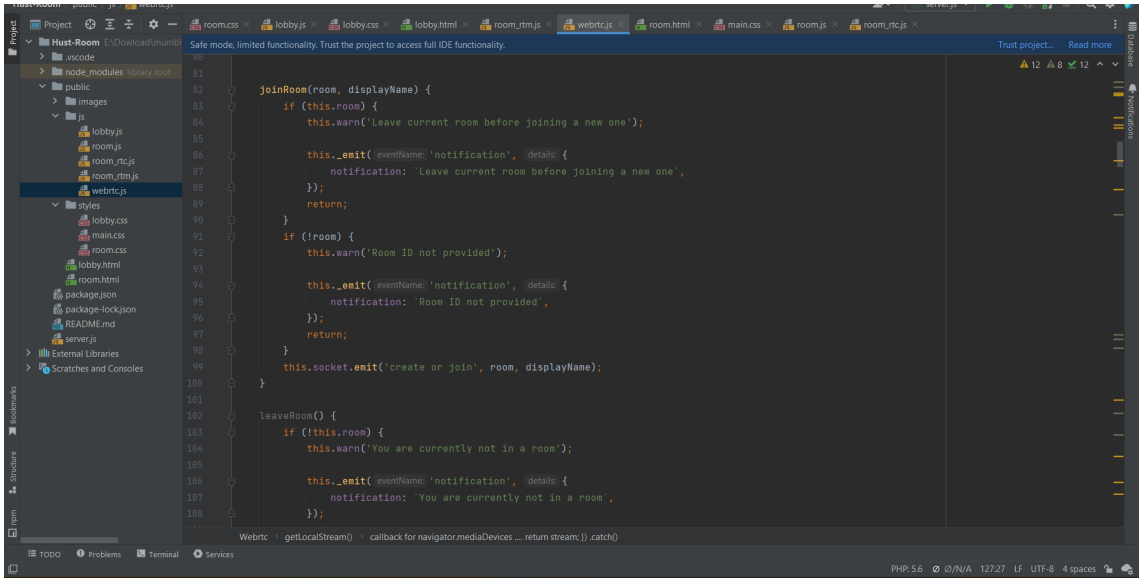
Hình 3.10: room_rtc.js

- room_rtm.js: Xử lý tin nhắn thời gian thực trong phòng bằng Socket.IO.



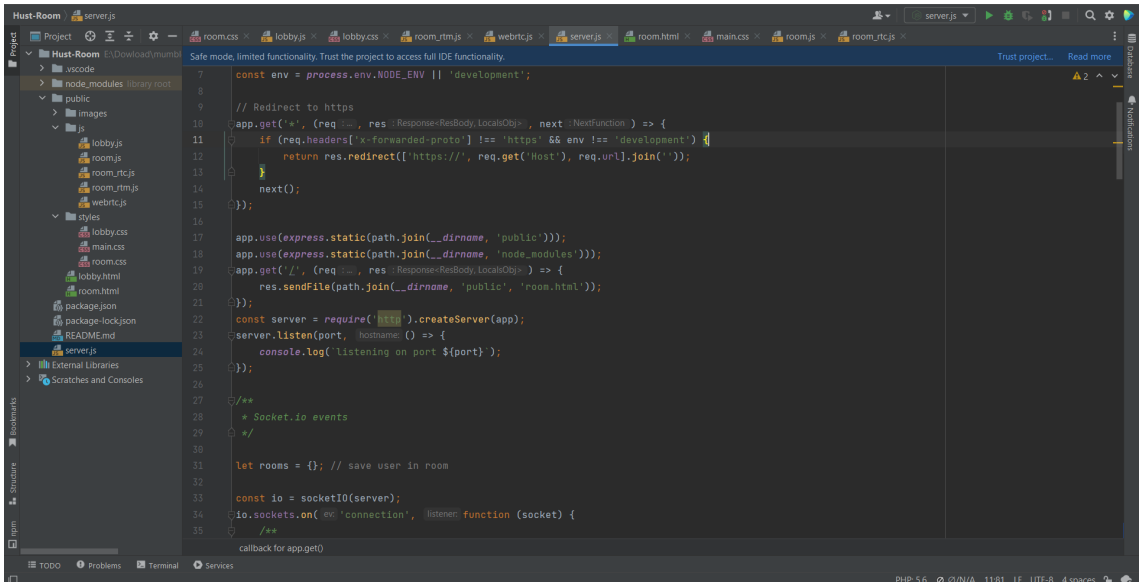
Hình 3.11: room_rtm.js

- webrtc.js: Một thư viện hoặc tệp tiện ích gói gọn chức năng WebRTC. Thành phần phụ trợ



Hình 3.12: webrtc.js

- Node.js Server (server.js): Tệp máy chủ chính khởi tạo máy chủ Socket.IO và thiết lập trình xử lý sự kiện cho các sự kiện socket như kết nối, tin nhắn và quản lý phòng.



Hình 3.13: server.js

CHƯƠNG 4. CÁCH THỨC HOẠT ĐỘNG TỔNG QUAN

4.1 Mô tả cách thức hoạt động

Client-Side (webrtc.js và room_rtc.js)

webrtc.js: Tệp này định nghĩa một lớp Webrtc quản lý các hoạt động WebRTC. Nó bao gồm các phương thức để tạo đề xuất, xử lý câu trả lời, và trao đổi các ứng viên ICE. Lớp này phát sự kiện được lắng nghe bởi các phần khác của ứng dụng để cập nhật giao diện người dùng tương ứng.

Creating Offers: Phương thức `_createAndSendOffer` trong `webrtc.js` chịu trách nhiệm tạo đề xuất SDP. Nó sử dụng API `RTCPeerConnection` để tạo đề xuất và sau đó đặt nó làm mô tả cục bộ. Sau đó, nó gửi đề xuất đến máy chủ bằng cách sử dụng phương thức `_sendMessage`.

Handling Answers: Khi một câu trả lời được nhận từ máy chủ ký hiệu, lớp `Webrtc` xử lý nó bằng cách đặt SDP nhận được làm mô tả từ xa của kết nối đồng đẳng.

ICE Candidates: Lớp `Webrtc` lắng nghe sự kiện `icecandidate` từ `RTCPeerConnection` và gửi bất kỳ ứng viên nào đến máy chủ ký hiệu. Nó cũng xử lý ứng viên ICE đến từ các đồng đồng bằng cách thêm chúng vào kết nối đồng đẳng.

room_rtc.js: Tệp này có thể khởi tạo chức năng WebRTC và liên kết sự kiện giao diện người dùng để kích hoạt các hoạt động WebRTC, chẳng hạn như tham gia một luồng, điều này có thể liên quan đến việc tạo đề xuất.

Server-Side(server.js)

Socket.IO: Tệp `server.js` thiết lập một máy chủ Socket.IO lắng nghe cho các sự kiện liên quan đến quá trình ký hiệu.

Message Handling: Máy chủ lắng nghe các sự kiện tin nhắn, bao gồm các tin nhắn SDP (đề xuất/câu trả lời) và ứng viên ICE. Sau đó, nó chuyển tải những tin nhắn này đến người nhận phù hợp dựa trên phòng mà họ đang ở.

Room Management: Máy chủ quản lý các phòng và đảm bảo rằng các tin nhắn chỉ được gửi đến khách hàng trong cùng một phòng, điều này quan trọng cho quá trình ký hiệu trong môi trường nhiều người dùng.

Signaling Flow

1. **Initialization:** Người dùng trong `room_rtc.js` kích hoạt một sự kiện `join`, kích hoạt quá trình WebRTC.

2. **Offer Creation:** `webrtc.js` tạo một đề xuất SDP và gửi nó đến máy chủ thông qua một sự kiện Socket.IO.
3. **Server Relay:** `server.js` nhận đề xuất và chuyển tải nó đến người nhận dự định.
4. **Answer:** Người nhận tạo một câu trả lời SDP và gửi nó trở lại máy chủ.
5. **. ICE Candidate Exchange:** Cả hai đồng đồng gửi ứng viên ICE đến máy chủ, máy chủ sau đó chuyển tải chúng đến đồng đồng kia.
6. **Connection Establishment:** Khi cả hai đồng đồng đã trao đổi đề xuất, câu trả lời và ứng viên ICE, kết nối trực tiếp từ đồng đồng đến đồng đồng được thiết lập và phương tiện có thể chảy giữa chúng.

Quá trình này được lặp lại cho mỗi người dùng tham gia vào phòng, với máy chủ quản lý các tin nhắn ký hiệu để đảm bảo mỗi khách hàng thiết lập kết nối với mọi khách hàng khác trong phòng.