NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT

Tên đề tài: Xây dựng Hệ thống giao tiếp thời gian thực.

Sinh viên thực hiện: Trần Thông Thành Luân

Số thẻ SV: 102140079 Lớp: 14T2

Người hướng dẫn: ThS. Mai Văn Hà

Hệ thống bao gồm: Website giao tiếp thời gian thực thông qua Video Call và Chat.

Các chức năng chính:

- Gọi video call nhiều người trong một phòng mà không cần đăng nhập.
- Mọi người có thể giao tiếp với nhau thông qua Chat.
- Chia sẻ màn hình Screen của mình cho mọi người.
- Người dùng có thể thiết lập các cài đặt về camera hoặc audio của chính mình.
- Trong khi chia sẻ nếu người dùng không muốn chia sẻ audio/video của mình nữa thì có thể tắt đi.
- Người dùng có thể thiết lập mật khẩu phòng.

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Trần Thông Thành Luân Số thẻ sinh viên: 102140079

Lớp: 14T2 Khoa: Công nghệ thông tin Ngành: Công nghệ thông tin

1. *Tên đề tài đồ án:* Xây dựng hệ thống giao tiếp thời thực

Đề tài thuộc diện: ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

2. *Các số liệu và dữ liệu ban đầu:*

……..……..……..…
……..……
……..……..……

3. *Nội dung các phần thuyết minh và tính toán:*

…...……

…...……

…...……

…...……

…...……

4. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

…...……

…...……

…...……

…...……

5. *Họ tên người hướng dẫn:* ThS. Mai Văn Hà

6. *Ngày giao nhiệm vụ đồ án:* *18/02/2019*

7. *Ngày hoàn thành đồ án:* *21/05/2019*

Đà Nẵng, ngày tháng năm 201

Trưởng Bộ môn …….. **Người hướng dẫn**

LỜI CẢM ƠN

Trên thực tế, không có thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù là trực tiếp hay gián tiếp của người khác. Trong suốt thời gian năm năm qua từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô, gia đình và bạn bè.

Đầu tiên em xin gửi lời cảm ơn chân thành nhất đến tất cả các thầy cô khoa Công nghệ Thông tin – Trường Đại học Bách khoa Đà Nẵng đã tận tình dạy bảo và dìu dắt em trong suốt năm năm học vừa qua tại trường.

Em xin chân thành cảm ơn thầy - ThS. Mai Văn Hà, người đã tận tình hướng dẫn, cũng như đã tạo mọi điều kiện để em có thể hoàn thành đồ án tốt nghiệp này. Nếu như không nhờ những lời hướng dẫn, những đánh giá và chỉ bảo sâu sắc của thầy thì em nghĩ việc hoàn thành đồ án tốt nghiệp của em sẽ khó khăn hơn rất nhiều. Một lần nữa em xin chân thành cảm ơn thầy.

Đồ án được thực hiện trong khoảng thời gian ba tháng. Bước đầu đi vào thực tế, tìm hiểu lĩnh vực ứng dụng web, với kiến thức còn hạn chế và nhiều bỡ ngỡ. Do vậy, không thể tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô và các bạn để kiến thức trong lĩnh vực này của em được hoàn thiện hơn.

Sau cùng, em xin kính chúc quý thầy cô trong Khoa Công nghệ Thông tin và thầy Mai Văn Hà thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Trân trọng!

Đà Nẵng, ngày ...., tháng ...., năm 2019

Trần Thông Thành Luân

LỜI CAM ĐOAN

Tôi xin cam đoan :

1. Nội dung trong đồ án này là do tôi thực hiện dưới sự hướng dẫn trực tiếp của ThS. Mai Văn Hà.
2. Các tham khảo dùng trong luận văn đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.
3. Nếu có những sao chép không hợp lệ, vi phạm, tôi xin chịu hoàn toàn trách nhiệm.

Sinh viên thực hiện

Trần Thông Thành Luân

MỤC LỤC

LỜI CẢM ƠN ..... **I**

LỜI CAM ĐOAN ..... **II**

MỤC LỤC ..... **III**

DANH SÁCH HÌNH ẢNH ..... **VI**

DANH SÁCH CÁC BẢNG ..... **VIII**

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT ..... **IX**

MỞ ĐẦU **1**

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT **3**

1.1. Giới thiệu về công nghệ WebRTC 3

1.1.1. Khái niệm WebRTC 3

1.1.2. Đặc điểm của WebRTC 4

1.1.3. Các API của WebRTC 6

1.1.4. Giao tiếp Peer-to-peer 9

1.2. Giới thiệu về Node.js 15

1.2.1. Định nghĩa Node.js 15

1.2.2. Mô hình hoạt động của Node.js 15

1.2.3. So sánh giữa mô hình của Node.js cùng các mô hình truyền thống khác 17

1.3. Tổng quan Socket.io 20

1.3.1. Cùng nhìn lại về Websocket 20

1.3.2. Giới thiệu về Socket.io.....21

1.4. Giới thiệu về React.js.....22

1.4.1. React.js là gì?.....22

1.4.2. Đặc điểm của React.js.....22

1.5. Kết chương.....26

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....27

2.1. Giới thiệu.....27

2.2. Mục đích và đối tượng hướng đến.....27

2.2.1. Mục đích.....27

2.2.2. Đối tượng.....27

2.3. Mô tả chung.....28

2.3.1. Tổng quan:.....28

2.3.2. Chức năng.....28

2.3.3. Môi trường hoạt động.....28

2.3.4. Yêu cầu giao diện.....29

2.3.5. Yêu cầu phi chức năng.....29

2.4. Thiết kế hệ thống.....29

2.4.1. Các tác nhân trong hệ thống.....29

2.4.2. Các biểu đồ.....29

2.5. Kết chương.....43

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ.....45

3.1. Triển khai hệ thống..... 45
3.1.1. Môi trường triển khai..... 45
3.1.2. Cài đặt môi trường..... 45
3.1.3. Kết quả thực nghiệm..... 45
3.1.4. Nhận xét và đánh giá kết quả..... 49
3.2. Kết chương..... 50
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... 51
TÀI LIỆU THAM KHẢO..... 53

DANH SÁCH HÌNH ẢNH

Hình 1.1 : WebRTC....3
Hình 1.2 : Các API của WebRTC....6
Hình 1.3 : Media Stream....6
Hình 1.4 : Mô tả RTCPeerConection....8
Hình 1.5 : Tường lửa Firewall....10
Hình 1.6 : Thiết bị NAT....11
Hình 1.7 : STUN và TURN server....12
Hình 1.8 : Cơ chế signaling, sessions, protocols....13
Hình 1.9 : Kết nối peer-to-peer....14
Hình 1.10 : Node.js....15
Hình 1.11 : Cách thức hoạt động của Node.js....16
Hình 1.12 : Mô hình Multi-Threaded Request-Response....17
Hình 1.13 : Mô hình Single Threaded with Event loop....19
Hình 1.14 : Giới thiệu socket.io....21
Hình 1.15 : React.js là gì?....22
Hình 1.16 : Virtual DOM trong React.js....23
Hình 1.17 : One-way Binding trong React.js....24
Hình 1.18 : React JSX....25
Hình 1.19 : React components....25
Hình 2.1 : Biểu đồ ca sử dụng chức năng của người dùng....30
Hình 2.2 : Biểu đồ ca sử dụng chức năng gọi video....31
Hình 2.3 : Biểu đồ tuần tự chức năng tham gia phòng....34
Hình 2.4 : Biểu đồ tuần tự chức năng gọi video....35
Hình 2.5 : Biểu đồ tuần tự chức năng gửi tin nhắn chat....36
Hình 2.6 : Biểu đồ tuần tự chức năng thay đổi cài đặt thiết bị....37

Hình 2.7 : Biểu đồ tuần tự chức năng thay đổi mật khẩu phòng.....38
Hình 2.8 : Biểu đồ hoạt động chức năng tham gia phòng.....39
Hình 2.9 : Biểu đồ hoạt động chức năng bật hoặc tắt camera/microphone.....40
Hình 2.10 : Biểu đồ hoạt động chức năng thoát phòng.....41
Hình 2.11 : Biểu đồ hoạt động chức năng copy link.....42
Hình 2.12 : Biểu đồ hoạt động chức năng chat.....43
Hình 3.1 : Giao diện dạng list.....46
Hình 3.2 : Giao diện dạng lưới.....46
Hình 3.3 : Giao diện khi ẩn toolbar.....47
Hình 3.4 : Giao diện khi chia sẻ màn hình.....47
Hình 3.5 : Giao diện chat.....48
Hình 3.6 : Giao diện thông tin phòng.....48
Hình 3.7 : Giao diện thay đổi cài đặt thiết bị.....49

DANH SÁCH CÁC BẢNG

Bảng 2.1 : Chức năng tham gia phòng..... 31

Bảng 2.2 : Chức năng gọi video..... 32

Bảng 2.3 : Chức năng thay đổi cài đặt thiết bị..... 32

Bảng 2.4 : Chức năng gửi tin nhắn chat..... 33

Bảng 3.1 : Đánh giá kết quả 49

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Từ viết tắt	Diễn giải
IP	Internet Protocol
API	Application Programming Interface
URL	Uniform Resource Locator
MVC	Model – View - Controller

MỞ ĐẦU

1. Tổng quan về đề tài

WebRTC là một web API có khả năng hỗ trợ trình duyệt (Browser) giao tiếp với nhau thông qua VideoCall, VoiceCall hay chuyển tiếp data "Peer-to-Peer" (P2P) mà không cần browser phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài.

Nói cách khác, WebRTC là tập hợp các tiêu chuẩn và giao thức cho phép các trình duyệt Web thực hiện trực tiếp các tính năng truyền thông đa phương tiện thời gian thực như Video Call, Chat, truyền dữ liệu bằng các API Javacript.

Hiện nay WebRTC đang được chuẩn hóa ở cấp độ API của W3C và cấp độ giao thức của IETF, những thứ này đang được hỗ trợ bởi các trình duyệt Google Chrome, Mozilla Firefox, Opera trên PC và Android.

2. Mục đích và ý nghĩa của đề tài

2.1. Mục đích

Đồ án tập trung tìm hiểu về công nghệ WebRTC, các APIs của trình duyệt, các giao thức được WebRTC sử dụng để có thể chia sẻ và truyền dữ liệu trực tiếp thời gian thực giữa các trình duyệt trong môi trường mạng. Đồ án cũng phân tích yêu cầu tính chất "thời gian thực" khi truyền dữ liệu media và cách thức WebRTC đang được xây dựng để giải quyết, cũng như cách thức vượt NAT, Firewall để thiết lập kết nối Peer to Peer.

2.2. Ý nghĩa

Hệ thống cho phép người dùng có thể gọi Video call cho nhiều người trong cùng một phòng mà không cần đăng nhập hoặc cài đặt thêm gì cả. Ngoài ra, người dùng còn có thể chia sẻ màn hình của mình cho những người khác. Màn hình được chia sẻ có thể là toàn bộ screen hoặc là một phần của screen tùy theo nhu cầu người sử dụng.

Trong khi meeting, người dùng có thể trao đổi với nhau thông qua cách gửi tin nhắn chat. Hệ thống sẽ lưu lại lịch sử chat để những người mới vào phòng cũng có thể xem được những đoạn hội thoại trước đó.

3. Phương pháp thực hiện

Công nghệ áp dụng: WebRTC, React - Redux, Node.js, socket.io.

Công cụ thực hiện: VSCode.

Ngôn ngữ lập trình: JavaScript, HTML, CSS.

Thư viện bổ trợ: Antd, styled-components.

3. Bố cục của đồ án

Đồ án bao gồm các nội dung sau:

Mở đầu

Chương 1: Cơ sở lý thuyết

Chương 2: Phân tích thiết kế hệ thống

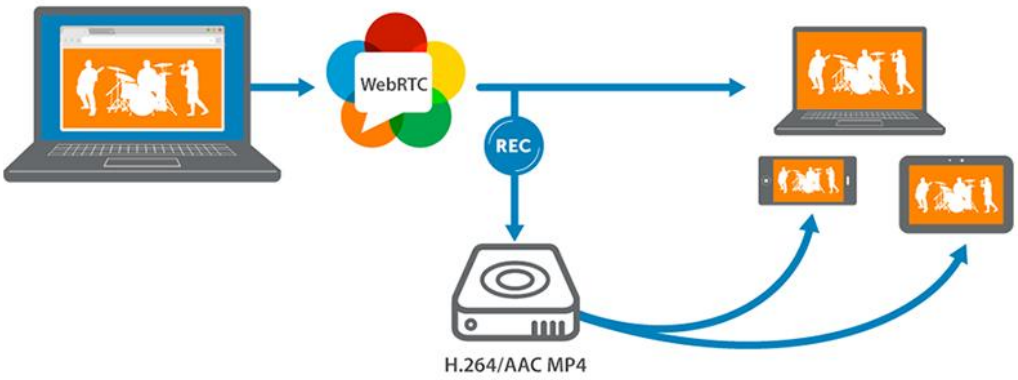
Chương 3: Triển khai và đánh gia kết quả

Kết luận và hướng phát triển.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu về công nghệ WebRTC

1.1.1. Khái niệm WebRTC



Hình 1.1: WebRTC

Theo **Arin Sime** - CEO của công ty AgilityFeat có trụ sở tại Charlottesville, bang Virginia, Hoa Kỳ, người đã đưa ra định nghĩa rất ngắn gọn về WebRTC như sau:

"WebRTC là một tiêu chuẩn HTML5 cho mã hóa peer-to-peer (P2P) trao đổi video, âm thanh và dữ liệu trực tiếp giữa hai máy tính".

Các tính năng nổi bật nhất của WebRTC đó là được tích hợp trực tiếp vào trình duyệt, chúng ta có thể dùng HTML5 và JavaScript để sử dụng nó.

WebRTC không chỉ là một sản phẩm hay một hàm API duy nhất. Nó là một tập hợp rất nhiều các hàm, có thể được lập trình viên sử dụng cho nhiều mục đích khác nhau. Có hàm chỉ để làm những việc đơn giản như đòi quyền truy cập vào camera và microphone của máy tính, có hàm phức tạp hơn thì để thiết lập kết nối giữa hai người dùng với nhau, có hàm còn dùng để chia sẻ màn hình với người dùng khác. Và có hàm để kết nối (Gọi) video, đây là chức năng quan trọng nhất của WebRTC.

Tuy nhiên, tất cả các hàm lập trình nằm trong bộ API có đặc điểm chung là thực thi hầu hết các tác vụ theo thời gian thực. Và nó không chỉ được dùng cho việc gọi video giữa hai trình duyệt mà còn có thể thực hiện nhiều tác vụ khác, để hai hoặc nhiều người dùng liên lạc với nhau. WebRTC được hỗ trợ chính thức bởi Google, Mozilla, Opera cùng nhiều hãng khác.

1.1.2. Đặc điểm của WebRTC

1.1.2.1. Chuẩn Video

WebRTC chưa có chuẩn video thống nhất. Đặc điểm nổi bật nhất của WebRTC là khả năng kết nối video từ trình duyệt web của người sử dụng. Nhưng hiện nay, các hãng trình duyệt cũng chưa thống nhất với nhau là chuẩn video nào sẽ được dùng cho WebRTC.

Google và Mozilla thì muốn dùng VP8 hoặc VP9, một codec video do chính Google phát triển theo mô hình mã nguồn mở. Thực tế cho thấy, đối với trình duyệt web Chrome 50 trở lên, codec VP9 tốt hơn. Nó hỗ trợ cho chất lượng tốt hơn trong truyền tải phương tiện truyền thông nhờ khả năng dự phòng (RED), cơ chế FEC tương tác với ước lượng băng thông và kiểm soát các thông số (NACK, POI, RTX, remb, ...) trong trình duyệt, khả năng tương tác với các trình duyệt khác tốt hơn VP8. Do đó, VP9 cho độ phân giải cao hơn, giảm thiểu các lỗi (nhờ hỗ trợ cho Error Correction Forward).

Trong khi đó, Microsoft và một số công ty khác thì muốn đề xuất sử dụng H.264 hoặc H.265 cho WebRTC, vốn đang là codec được dùng phổ biến nhất hiện nay trên Internet. Và tất nhiên, H.264 lại thuộc quyền sở hữu của Hiệp hội MPEG LA và phải trả phí bản quyền để sử dụng.

Hiện nay, nhóm làm việc của Google gồm Niklas Blum, Justin Uberti và Per Ahgren đã thực hiện được những cải tiến về hiệu suất, đó là trình duyệt Chrome đã có thể bớt thời gian kết nối trung bình từ 2 giây xuống còn 650ms cho một kết nối 1Mbps. Họ cũng đang làm việc để mang lại VP9 – codec video đến các thiết bị di động, nhờ đó sẽ cho phép đạt chất lượng video cao với yêu cầu băng thông ít hơn.

1.1.2.2. Tính an toàn, bảo mật của công nghệ WebRTC

1.1.2.2.1. Tính an toàn của WebRTC

Mã hóa là bắt buộc đối với tất cả thành phần WebRTC, bao gồm các cơ chế báo hiệu (Signaling)

Truy xuất camera và microphone phải được cấp quyền rõ ràng và khi camera hoặc microphone đang hoạt động thì phải được thể hiện ra ở phía giao diện người dùng.

Thành phần của WebRTC chạy trong sandbox của trình duyệt, các thành phần không yêu cầu cài đặt riêng và được cập nhật bất cứ khi nào trình duyệt được cập nhật. Sandbox là một kỹ thuật qua trọng trong lĩnh vực bảo mật có tác dụng cô lập các ứng dụng, ngăn chặn các phần mềm độc hại để chú không thể làm hỏng hệ thống máy tính, hay cài cắm các mã độc nhằm ăn cắp thông tin cá nhân của người sử dụng.

1.1.2.2.2. Tính bảo mật của WebRTC

Công nghệ WebRTC sử dụng giao thức bảo mật DTLS và SRTP:

- DTLS (Datagram Transport Layer Security) cho phép các ứng dụng được thiết kế để ngăn chặn việc nghe trộm, giả mạo, hoặc tin nhắn giả mạo. Các gói giao thức DTLS đảm bảo các ứng dụng khô bị chậm trễ kết hợp với việc xử lý khi mất gói tin và dữ liệu.

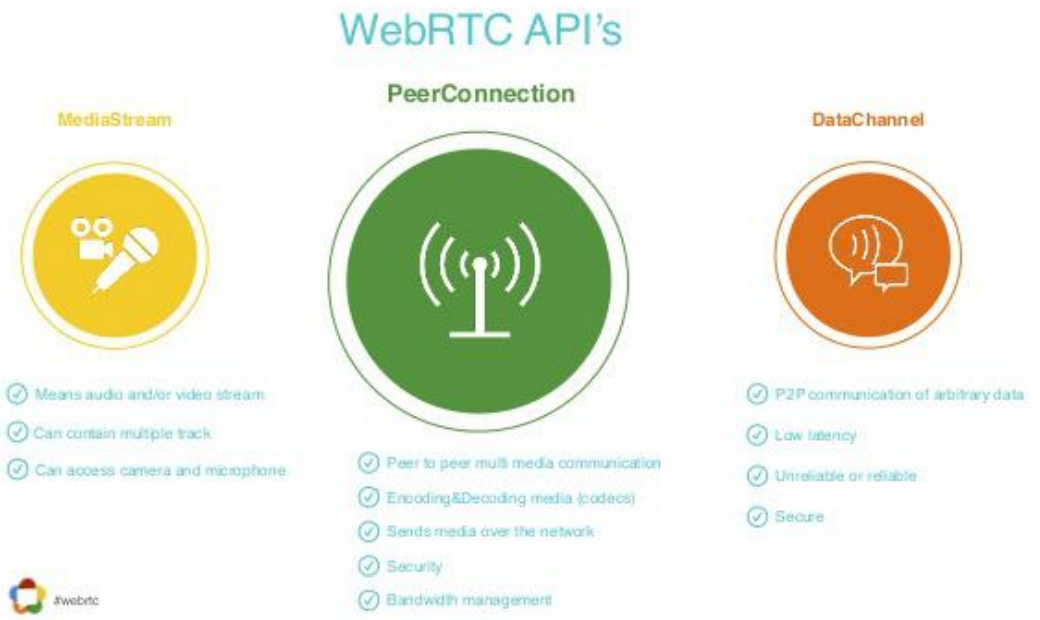
- Giao thức SRTP (Secure Real-time Trn sport Protocol) cung cấp tính bảo mật, tính toàn vẹn và xác thực thông báo trong môi trường truyền thông (tiếng nói và hình ảnh). Bằng việc sử dụng thuật toán dẫn xuất khoá nguyên thuỷ, SRTP có khả năng tối thiểu hoá tính toán và sử dụng tài nguyên để sinh các khoá mật mã qua cơ chế quả lý khoá ngoài.

1.1.2.3. WebRTC không phải là một plugin

Thành phần của WebRTC chạy trong sandbox của trình duyệt, các thành phần không yêu cầu cài đặt riêng và được cập nhật bất cứ khi nào trình duyệt được cập nhật.

Sandbox là một kỹ thuật qua trọng trong lĩnh vực bảo mật có tác dụng cô lập các ứng dụng, ngăn chặn các phần mềm độc hại để chú không thể làm hỏng hệ thống máy tính, hay cài cắm các mã độc nhằm ăn cắp thông tin cá nhân của người sử dụng.

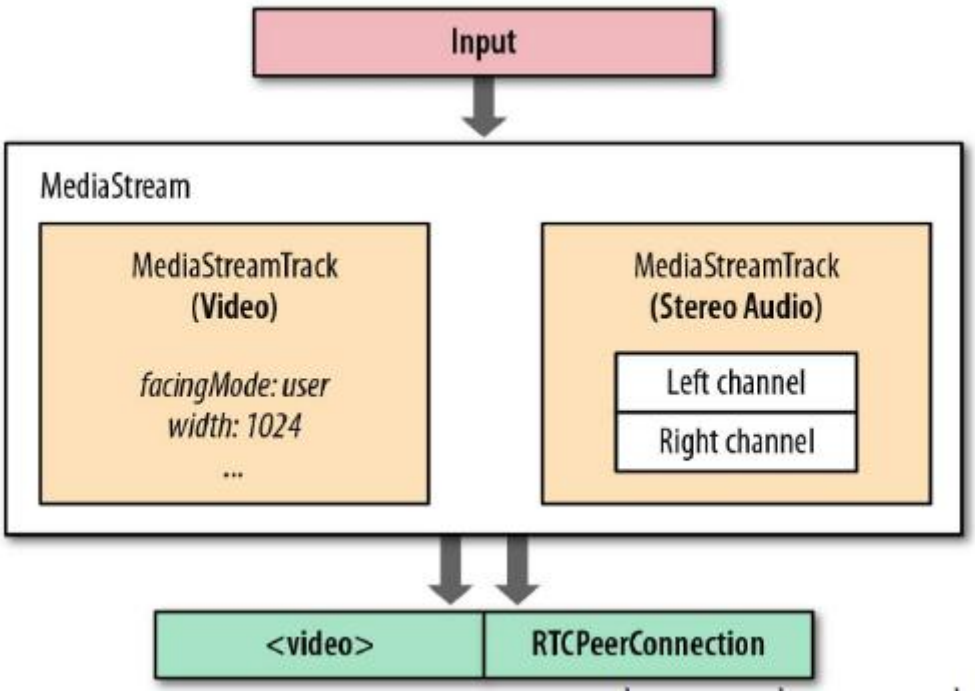
1.1.3. Các API của WebRTC



Hình 1.2: Các API của WebRTC

WebRTC có ba loại hàm API thường được dùng.

1.1.3.1. Media Capture and Stream



Hình 1.3: Media Stream

API **Media Capture & Streams**, thường được gọi là Media Stream API hoặc Stream API, là 1 API hỗ trợ các phương thức để làm việc với những luồng (Stream) dữ liệu âm thanh hay hình ảnh.

Các API này cho phép bạn truy xuất vào các thiết bị đầu vào, chẳng hạn như microphone hay web camera và lấy ra luồng media từ các thiết bị đó.

Phương thức **getUserMedia**() của **MediaDevices** nhắc nhở người dùng cấp quyền để sử dụng đầu vào media, nó tạo ra 1 MediaStream kèm với các track chứa kiểu request của media. Luồng đó có thể chứa 1 track video (được tạo ra bởi phần cứng hay nguồn video ảo chẳng hạn như camera, thiết bị ghi video, dịch vụ chia sẻ màn hình,...), 1 track audio (được tạo ra bởi nguồn ghi vật lý hoặc nguồn ghi ảo như microphone, bộ chuyển A/D...) và có thể có cả những loại track khác.

getUserMedia() sẽ trả về 1 Promise với resolve là object MediaStream. Nếu người dùng từ chối cấp quyền hoặc là media phù hợp không tồn tại thì promise đó sẽ bị reject với lỗi là PermissionDeniedError hoặc NotFoundError.

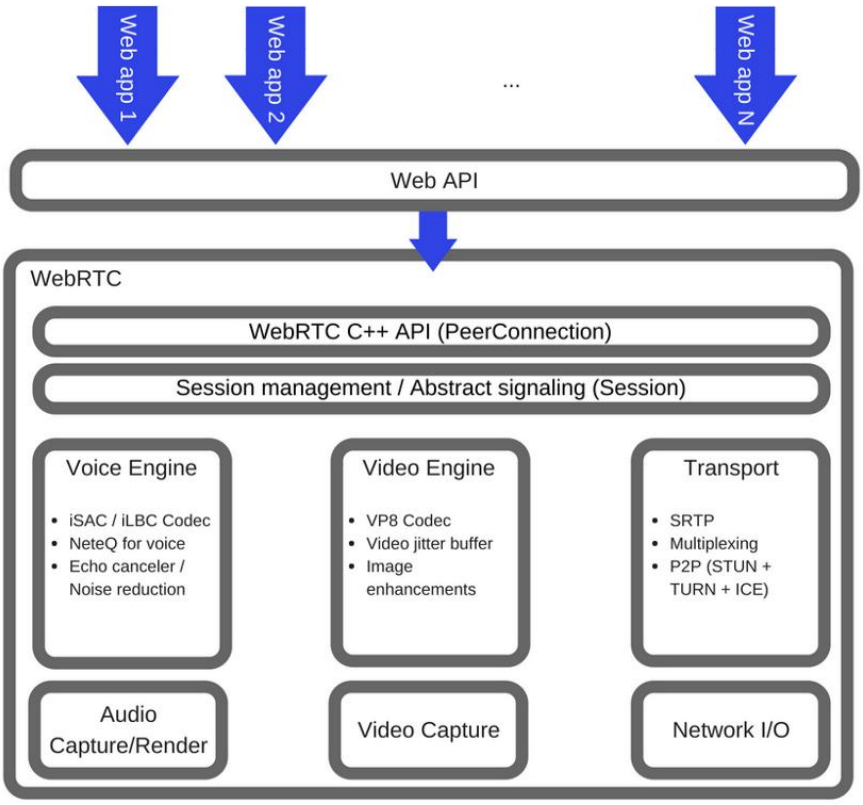
MediaDevice có thể được truy xuất thông qua object navigator như sau:

```
navigator.mediaDevices.getUserMedia(constraints)
.then(function(stream) {
/* use the stream */
})
```

Ta cần truyền object constraints để nói cho API loại stream nào để trả về. Ta có thể cấu hình tất cả những thứ liên quan đến camera và microphone như: Camera trước / sau, tần suất khung hình, độ phân giải,...

1.1.3.2. RTCPeerConection

RTCPeerConnection thể hiện 1 kết nối WebRTC giữa máy tính local và remote. Nó cung cấp các phương thức để kết nối đến peer từ xa, duy trì, kiểm soát kết nối và ngắt kết nối một khi nó không cần dùng tới nữa.



Hình 1.4: Mô tả RTCPeerConection

Các mã codec và giao thức được dùng bởi WebRTC thực hiện 1 lượng lớn công việc để làm cho giao tiếp real-time hoạt động được, kể cả với các mạng không đáng tin cậy:

- ➢ Che dấu mất gói tin (Package loss concealment - PLC, là 1 kỹ thuật dùng để che đậy ảnh hưởng của tình trạng mất gói tin trong giao tiếp VoIP)
- ➢ Hủy bỏ phản hồi (Echo cancellation - trong mạng máy tính thì "echo" được hiểu là quá trình gửi trả về gói tin đã gửi đi)
- ➢ Khả năng thích ứng băng thông
- ➢ Bộ đệm jitter động (jitter buffer - Trong kỹ thuật VoIP thì jitter buffer là vùng dữ liệu chia sẻ nơi mà các gói tin âm thanh có thể được thu thập, lưu trữ và gửi đi đến bộ xử lý âm thanh)
- ➢ Tự động chiếm quyền kiểm soát
- ➢ Giảm nhiễu và xóa nhiễu
- ➢ "Dọn dẹp" hình ảnh

1.1.3.3. RTCDataChannel

Cũng như hình ảnh và âm thanh, WebRTC cũng hỗ trợ giao tiếp real-time cho các loại dữ liệu khác. API RTCDataChannel cho phép peer-to-peer trao đổi các dữ liệu tùy ý.

Có rất nhiều trường hợp sử dụng API này, ví dụ: Dùng trong các game, Ứng dụng chat real-time, Truyền file, Các mạng phi tập trung,...

API cũng có nhiều tính năng để tận dụng tối đa RTCPeerConnection và kích hoạt sức mạnh cũng như sự linh động của giao tiếp peer-to-peer:

- ➢ Tận dụng cài đặt phiên của RTCPeerConnection
- ➢ Đa kênh đồng thời với khả năng phân chia mức độ ưu tiên
- ➢ Ngữ cảnh vận chuyển đáng tin cậy và không đáng tin cậy.
- ➢ Tích hợp sẵn bảo mật (DTLS) và kiểm soát tắc nghẽn.

Sự giao tiếp diễn ra trực tiếp giữa các trình duyệt, vì thế RTCDataChannel có thể nhanh hơn nhiều so với WebSocket kể cả khi cần đến 1 server chuyển tiếp (TURN).

1.1.4. Giao tiếp Peer-to-peer

Để có thể giao tiếp lẫn nhau thông qua trình duyệt web, mỗi trình duyệt của user phải thực hiện những bước sau đây:

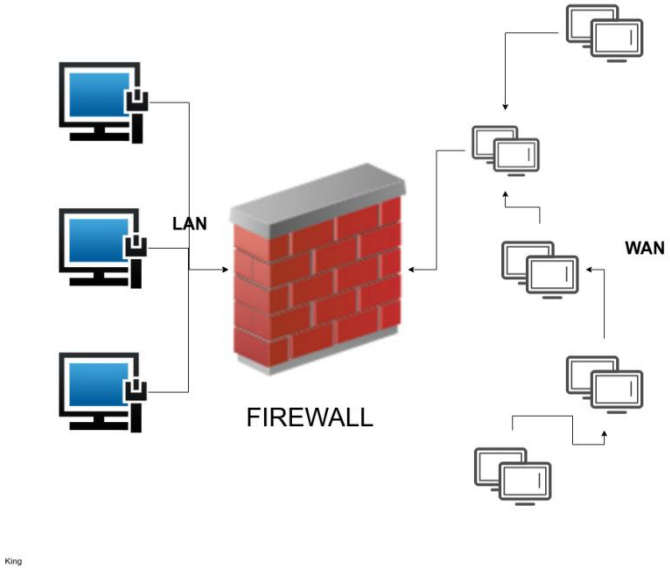
- ➢ Đồng ý để bắt đầu giao tiếp
- ➢ Biết cách xác định vị trí của đối tượng
- ➢ Vượt qua an ninh và tưởng lửa bảo vệ
- ➢ Chuyển giao tất cả các giao tiếp đa phương tiện theo real-time

Một trong số những thách thức lớn nhất liên quan đến các giao tiếp P2P dựa trên trình duyệt là làm sao để biết vị trí & thiết lập một kết nối socket mạng (Network socket connection) với một trình duyệt khác để vận chuyển dữ liệu hai chiều. Ta sẽ xem xét những khó khăn liên quan đến việc thiết lập kết nối này.

Khi WebApp cần dữ liệu hoặc tài nguyên, nó sẽ lấy về từ server thông qua các request. Tuy nhiên, nếu muốn tạo ra một ứng dụng video chat bằng cách kết nối trực

tiếp đến trình duyệt của người khác thì đây lại là một vấn đề. Ta không biết địa chỉ cụ thể của họ bởi vì trình duyệt của người kia không phải là một web server.

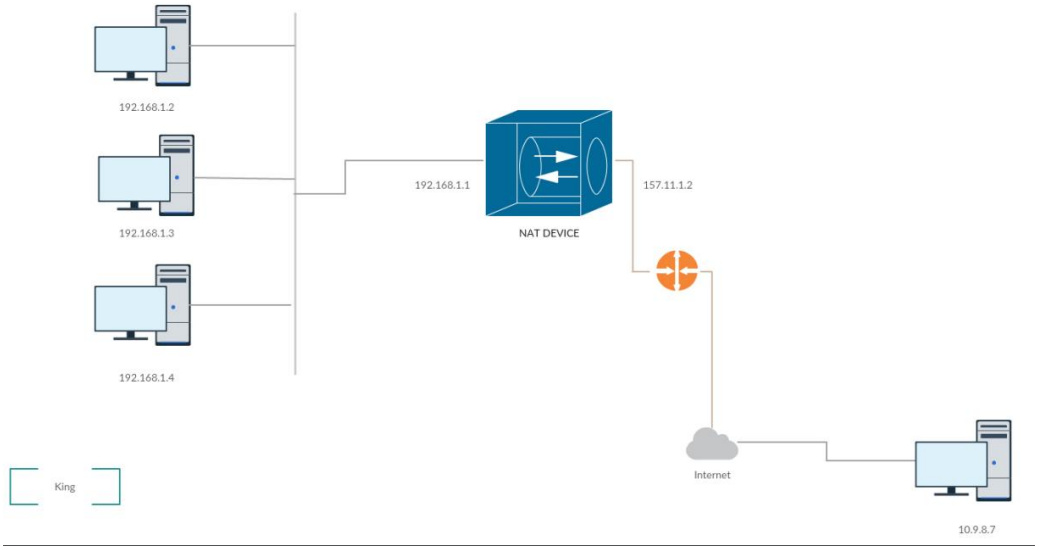
1.1.4.1. Tường lửa và NAT Traversal



Hình 1.5: Tường lửa Firewall

Thường thì máy tính của người dùng khôngcó địa chỉ IP public tĩnh. Lý do là máy tính của người dùng phải ở đằng sau tường lửa và thiết bị NAT (Network access translation - Bộ phiên dịch truy cập mạng).

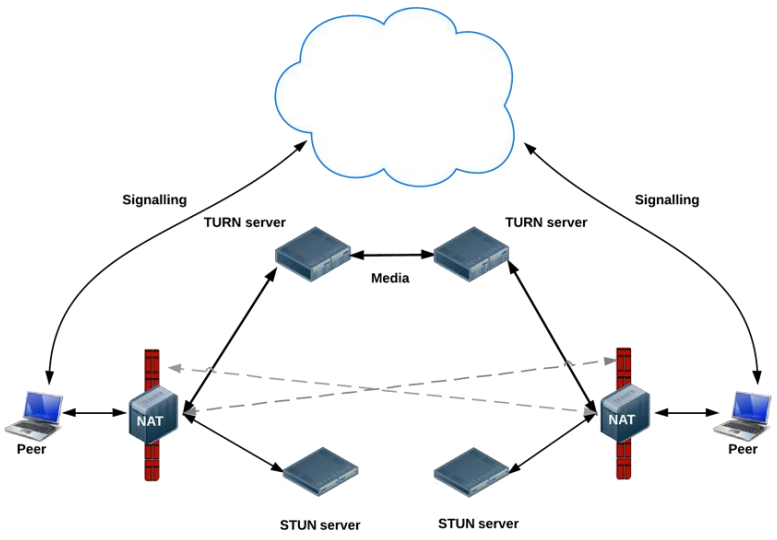
Thiết bị NAT sẽ dịch địa chỉ IP cá nhân từ bên trong tường lửa thành địa chỉ IP công khai (Public-facing IP). Ta cần các thiết bị NAT để bảo mật và giải quyết sự giới hạn của IPv4 đối với những địa chỉ IP công khai sẵn có. Đó là lý do tại sao WebApp không nên giả định rằng thiết bị hiện tại có 1 địa chỉ IP public tĩnh.



Hình 1.6: Thiết bị NAT

Nếu như bạn đang ở trong môi trường công cộng và kết nối vào mạng WiFi, máy tính của bạn sẽ được gán 1 địa chỉ IP mà nó chỉ tồn tại đằng sau NAT. Giả sử IP là **172.0.23.4**, tuy nhiên, với thế giới bên ngoài, địa chỉ IP của bạn có thể mang giá trị khác, ví dụ **164.53.27.98**. Vì vậy, thế giới bên ngoài sẽ thấy các request của bạn đến từ địa chỉ **164.53.27.98** nhưng thiết bị NAT sẽ đảm bảo các response cho những request (Được gửi từ máy của bạn) sẽ trả về đúng chỗ là **172.0.23.4**. Nhờ vào các bảng ánh xạ (Mapping table). Ngoài địa chỉ IP thì cổng (Port) cũng là điều kiện cần thiết cho các giao tiếp mạng.

Do có sự tham gia của các thiết bị NAT, trình duyệt của người dùng cần tìm được địa chỉ IP của máy tính có trình duyệt mà họ muốn giao tiếp.



Hình 1.7: STUN và TURN server

Đến đây thì ta lại phải nhờ đến các server **STUN** (**S**ession **T**raversal **U**tilities for **N**AT - Tiện ích truyền tải theo phiên cho NAT) và **TURN** (**T**raversal **U**sing **R**elays around **N**AT - Truyền tải sử dụng điểm chuyển tiếp vòng quanh NAT). Để các công nghệ WebRTC hoạt động được, đầu tiên một request hỏi địa chỉ IP public của bạn sẽ được gửi đến server STUN. Ta cứ nghĩ theo hướng kiểu như máy tính đang tạo truy vấn đến 1 server từ xa để hỏi về địa chỉ IP mà server đó nhận câu truy vấn là bao nhiêu. Server từ xa sẽ trả về địa chỉ IP Public mà nó thấy.

Cũng giống như STUN tuy nhiên TURN hỗ trợ cả giao thức TCP làm giao thức truyền tải. TURN bổ xung cho hạn chế của STUN là hỗ trợ Symmetric NAT. Dữ liệu thay vì được gửi trực tiếp tới các peer thì các peer sẽ gửi dữ liệu tới các TURN server và TURN server sẽ đóng vai trò trung gian vận chuyển gói tin. Điều này nâng cao giúp chất lượng dịch vụ của ứng dụng mà còn đảm bảo an toàn thông tin khi truyền dẫn. Bất lợi của TURN là chi phí sử dụng lớn, vì sẽ có một lưu lượng băng thông lớn được sử dụng.

Giả sử tiến trình này hoạt động bình thường và ta nhận được địa chỉ IP public của mình cũng như số port. Sau đó ta có thể nói với những peer ngang hàng khác làm thế nào để kết nối trực tiếp đến mình. Những peer này cũng có thể làm cùng một việc là sử dụng STUN & server TURN và nói cho ta biết nên liên lạc đến địa chỉ nào.

1.1.4.2. ICE (Interactive Communication Establishment)

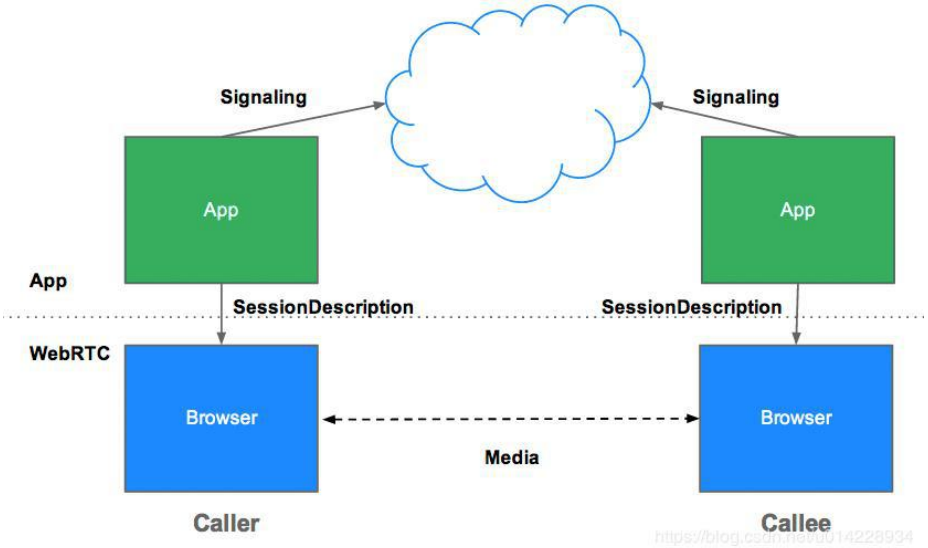
ICE là một giao thức được dùng để thiết lập phiên media dựa trên UDP đi qua NAT một cách nhanh nhất. ICE sẽ tìm đường tốt nhất để kết nối giữa các peer, nó thử tất cả khả năng có thể kết nối một cách song song và lựa chọn con đường hiệu quả nhất.

ICE bao gồm: Những metadata cần thiết, các định tuyến mạng (Địa chỉ IP và port) và các thông tin media thường dùng để giao tiếp với mỗi peer.

Đầu tiên ICE sẽ cố gắng tạo ra một kết nối bằng cách sử dụng địa chỉ thu được từ hệ điều hành và card mạng của thiết bị, nếu không thành công (Có thể do thiết bị ở đằng sau NAT) thì ICE sẽ lấy Public IP của thiết bị bằng cách sử dụng máy chủ STUN, nếu không thành công nữa thì nó sẽ chuyển lưu lượng mạng qua một máy chủ chuyển tiếp là TURN.

Việc sử dụng server TURN về cơ bản là dùng một server hoạt động như người trung gian và nó chuyển tiếp bất kỳ dữ liệu nào truyền qua lại giữa các peer. Lưu ý rằng đây không phải là giao tiếp peer-to-peer thực thụ trong đó các peer truyền dữ liệu hai chiều trực tiếp đến với nhau. Mỗi peer sẽ không cần phải biết làm thế nào để liên lạc và truyền dữ liệu đến bên kia. Thay vào đó, nó cần biết server TURN public nào để gửi và nhận dữ liệu đa phương tiện theo thời gian thực xuyên suốt phiên giao tiếp.

1.1.4.3. Signaling, Sessions, Protocols



Hình 1.8: Cơ chế signaling, sessions, protocols

Signaling dựa trên 1 chuẩn JSEP (**J**avascript **S**ession **E**stablishment **P**rotocol - Giao thức thiết lập phiên của Javascript). Signaling bao gồm cả khám phá mạng (Network discovery), NAT Traversal, tạo và quản lý phiên, bảo mật giao tiếp, siêu dữ liệu (Metadata) và phối hợp khả năng của media, xử lý lỗi.

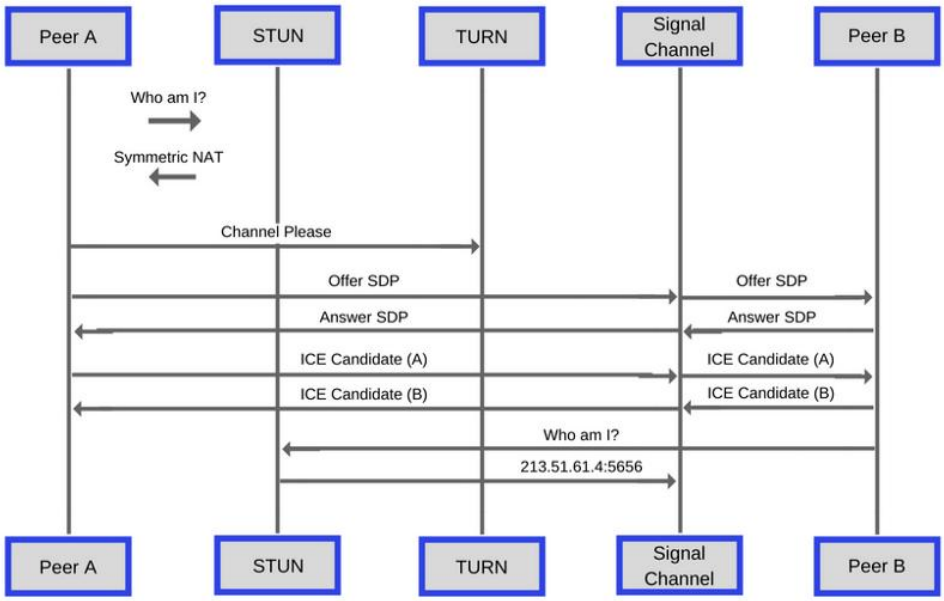
Để kết nối có thể hoạt động, peer thu nhập những điều kiện về local media cho metadata (Ví dụ: Chuẩn màn hình, kiểu codec,...) và gom góp các địa chỉ mạng có thể có cho host của ứng dụng. Cơ chế signaling dùng để truyền tới / lui những thông tin quan trọng này không được định nghĩa trong API của WebRTC.

Signaling không được quy định bởi chuẩn WebRTC và nó không được triển khai bằng WebRTC API để cho phép sử dụng một cách linh động các công nghệ và giao thức cần thiết (XMPP, SIP, Socket,...)

Giao thức signaling được chọn phải hoạt động với 1 giao thức ở tầng ứng dụng gọi là **S**ession **D**escription **P**rotocol (SDP - Giao thức mô tả phiên), giao thức này được

sử dụng trong trường hợp của WebRTC. Tất cả các metadata đa phương tiện cụ thể được truyền đi bằng giao thức SDP này.

Bất kỳ peer nào khi giao tiếp với một peer khác đều sinh ra một tập các ứng viên giao thức Interactive Connectivity Establishment (ICE Candicate). Những ứng viên này biểu diễn 1 bộ kết hợp của địa chỉ IP, port, giao thức giao vận được dùng. Lưu ý rằng một máy tính có thể có nhiều giao diện mạng (Không dây, có dây,...), vì thế có thể được gán nhiều địa chỉ IP cho mỗi giao diện.

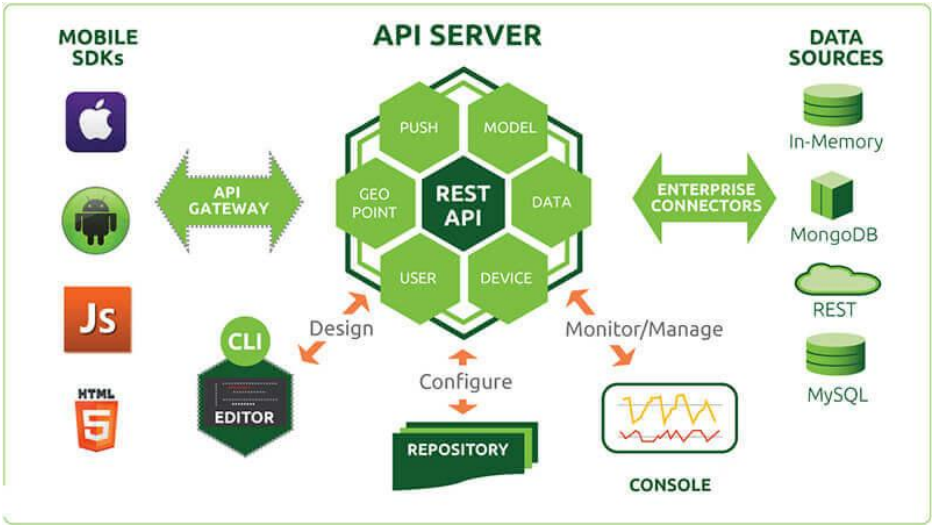


Hình 1.9: Kết nối peer-to-peer

Một khi 2 hoặc nhiều peer đã cùng kết nối vào 1 "kênh" thì những peer đã có thể giao tiếp và trao đổi thông tin phiên. Về cơ bản, peer khởi tạo ban đầu sẽ gửi 1 "lời đề nghị" (CreateOffer) sử dụng một giao thức signaling chẳng hạn như Session Initiation Protocol (SIP - Giao thức khởi tạo phiên) và SDP. Người khởi tạo sẽ chờ để nhận được "câu trả lời" (CreateAnswer) từ bất kỳ người nhận nào đã kết nối với "kênh". Khi đã nhận được câu trả lời, 1 tiến trình diễn ra để xác định và trao đổi giao thức ứng viên ICE tốt nhất mà mỗi peer thu thập được.

1.2. Giới thiệu về Node.js

1.2.1. Định nghĩa Node.js



Hình 1.10: Node.js

Node.js là một phần mềm mã nguồn mở được viết dựa trên ngôn ngữ JavaScript cho phép lập trình viên có thể xây dựng các ứng dụng chạy trên máy chủ. Ban đầu, Node.js được phát triển bởi Ryan Dahl. Phiên bản đầu tiên của Node.js được cho ra mắt vào năm 2009.

Node.js có thể chạy được trên nhiều nền tảng khác nhau như Windows, Linux hay Mac OS. Node.js được phát triển sử dụng V8 Engine là bộ thư viện JavaScript được Google phát triển để viết trình duyệt web Chrome.

Bản thân Node.js không phải là một ngôn ngữ lập trình mới, thay vào đó Node.js là một nền tảng mã nguồn mở (hay phần mềm mã nguồn mở) được viết dựa trên ngôn ngữ JavaScript.

Sự ra đời của Node mang đến một luồng gió mới trong giới lập trình. Nếu trước đây ai cũng chỉ nghĩ Javascript chỉ dùng cho việc phát triển các ứng dụng web phía người dùng cuối, thì giờ đây cùng với Node, JavaScript đã có mặt tại phía Server.

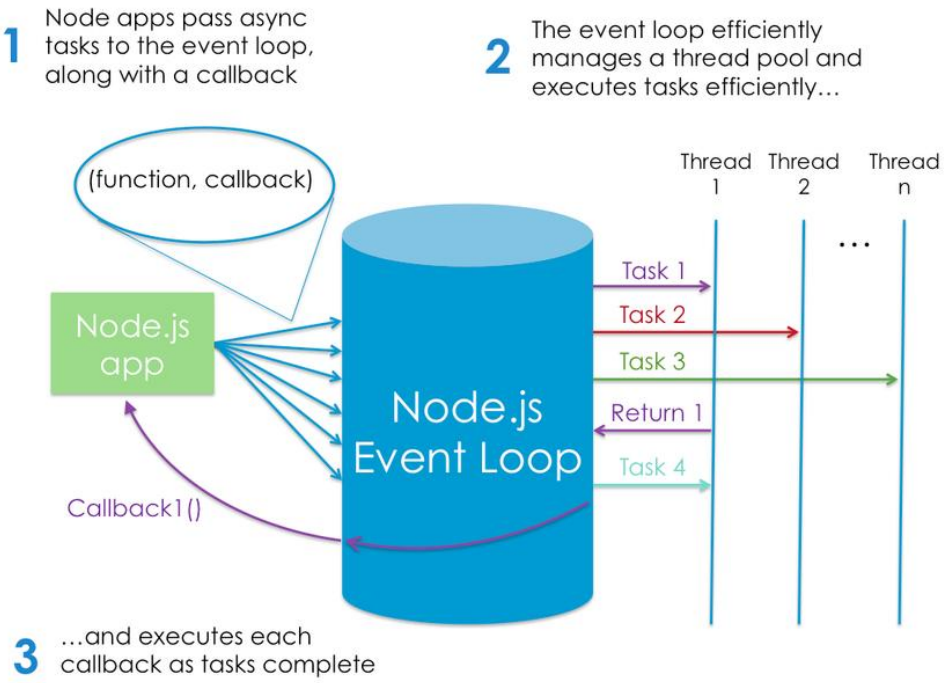
1.2.2. Mô hình hoạt động của Node.js

Khi ta connect đến một server truyền thống, chẳng hạn Apache, nó sẽ sinh ra một thread mới để xử lý request. Ở các ngôn ngữ như PHP hay Ruby, mỗi một phép toán I/O (ví dụ truy cập database) sẽ chặn execution trên code của bạn cho đến khi phép toán đó hoàn thành. Nói cách khác, server sẽ đợi cho đến khi database được

duyệt xong mới xử lý kết quả. Nếu có những request mới, server lại tiếp tục sinh những thread mới để xử lý chúng. Điều này dẫn đến nguy cơ kém hiệu quả, khi một lượng lớn thread được tạo ra sẽ khiến cho hệ thống trở nên chậm chạp, tệ hơn nữa có thể khiến site bị sập. Cách thông thường để giải quyết tình trạng này là bổ sung thêm server.

Node.js, mặt khác là single-threaded. Nó cũng thuộc dạng event-driven hay nói cách khác tất cả những gì xảy ra trong Node là để phản hồi lại với một sự kiện. Ví dụ, khi một request được gửi đến, server bắt đầu xử lý nó. Nếu nó gặp phải phép toán I/O, thay vì đợi cho phép toán này kết thúc, nó sẽ đăng ký một callback trước khi tiếp tục xử lý event tiếp theo. Khi phép toán I/O kết thúc, server sẽ chạy callback và tiếp tục làm việc trên request ban đầu.

Ở tầng bên dưới, Node sử dụng thư viện **libuv** để thực hiện hoạt động asynchronous (non-blocking) này. Mô hình hoạt động này của Node giúp server có thể xử lý một lượng lớn kết nối đến đồng thời. Quan điểm truyền thống để scale một Node app là clone nó và để instance được clone chia sẻ công việc.



Hình 1.11: Cách thức hoạt động của Node.js

Việc Node chạy trên một thread duy nhất dẫn đến một số hạn chế. Ví dụ như nên tránh các phép toán blocking I/O, và errors luôn cần được handle một cách đúng đắn nếu không có thể dẫn đến toàn bộ process bị crash.

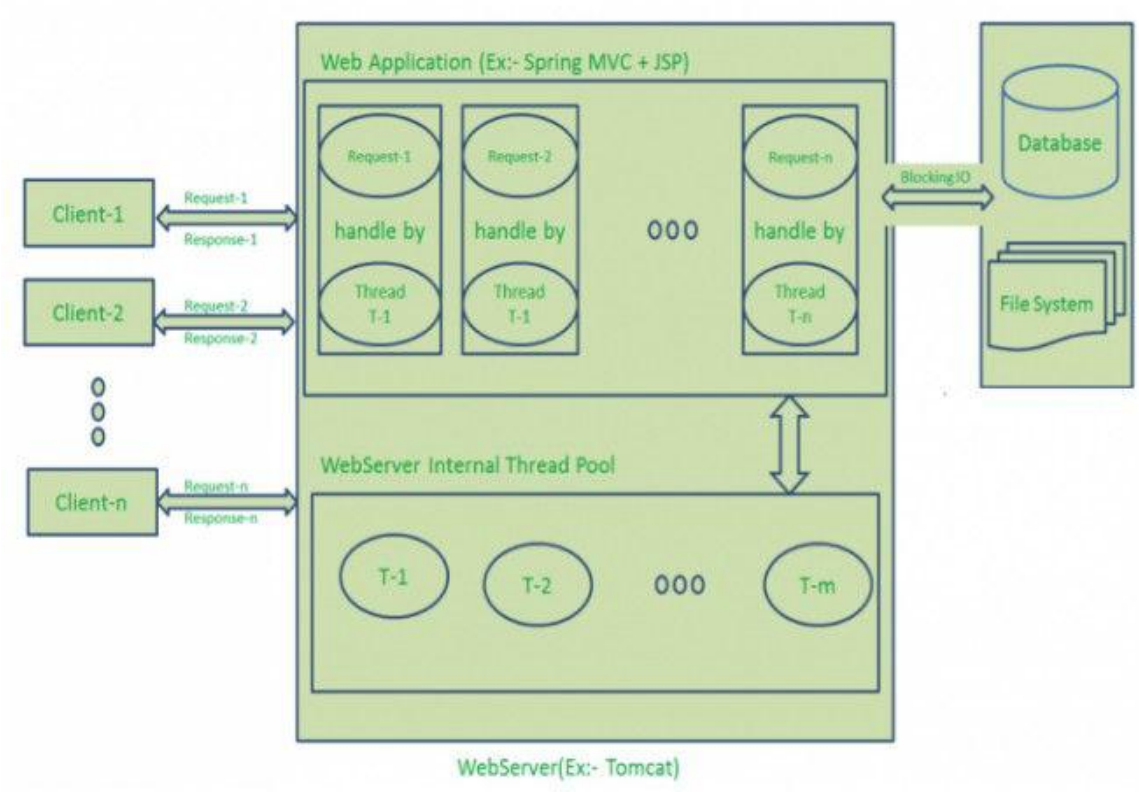
1.2.3. So sánh giữa mô hình của Node.js cùng các mô hình truyền thống khác

1.2.3.1. Mô hình truyền thống

Bất kỳ ứng dụng Web nào được phát triển ngoài Node JS, thường theo mô hình "Multi-Threaded Request-Response". Đơn giản chúng ta có thể gọi nó là mô hình Request/Response.

Clients gửi yêu cầu đến server, sau đó server xử lý một vài tiến trình dựa trên request từ phía clients, chuẩn bị response và gửi lại cho clients.

Mô hình này sử dụng giao thức HTTP. Vì HTTP là một giao thức Stateless, mô hình Request/Response này cũng là mô hình Stateless. Vì vậy chúng ta có thể gọi nó là Request / Response Stateless Model.



Hình 1.12: Mô hình Multi-Threaded Request-Response

Các bước xử lý của mô hình Request/Response:

➢ Clients gửi yêu cầu tới Web Server.

- Web Server duy trì một Limited Thread pool để cung cấp dịch vụ cho các request của clients.
- Web Server đang ở trong Loop vô hạn và chờ đợi request từ phía clients.
- Web Server nhận được những request đó.
    - Web Server chọn 1 yêu cầu từ phía clients
    - Chọn một Thread từ Thread pool
    - Gán Thread này cho request của clients
    - Thread này sẽ đọc yêu cầu của clients, xử lý yêu cầu của client, thực hiện bất kỳ Blocking IO nào (nếu cần) và chuẩn bị phản hồi
    - Thread này gửi phản hồi trở lại Web Server
    - Web Server lần lượt gửi phản hồi này đến Client tương ứng.

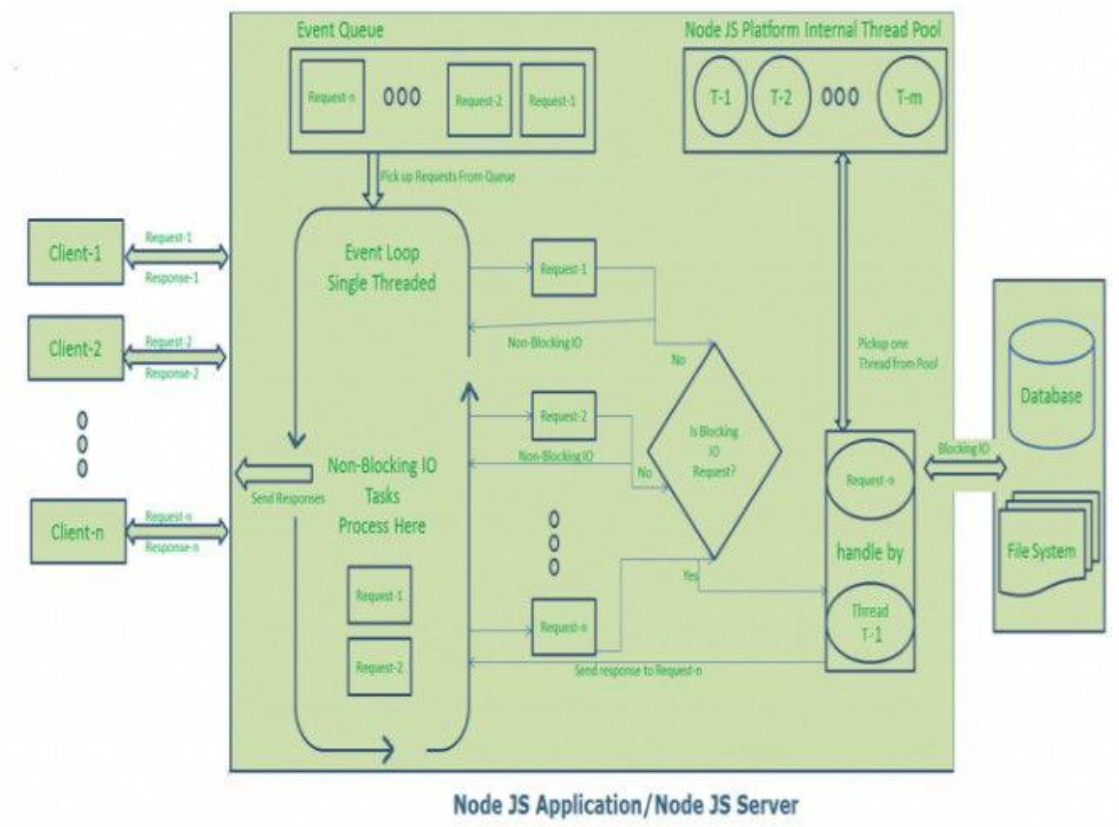
Web Server đợi trong vòng lặp vô hạn và thực hiện tất cả các bước phụ như đã đề cập ở trên cho tất cả n Clients. Điều đó có nghĩa là mô hình này tạo ra một Thread per Client.

Nếu nhiều request của client yêu cầu Blocking IO Operations, thì hầu như tất cả các Thread đều bận rộn trong việc chuẩn bị phản hồi. Các request clients còn lại sẽ phải chờ trong một khoảng thời gian.

1.2.3.2. Mô hình của Node.js

Node JS Platform không theo mô hình Request/Response Multi-Threaded Stateless. Nó tuân theo mô hình Single Threaded with Event Loop. Mô hình chính Node JS Processing dựa trên mô hình cơ bản Javascript Event với cơ chế Javascript callback.

Trái tim của mô hình Node JS Processing là "Event Loops". Nếu chúng ta hiểu điều này, thì rất dễ hiểu các Node JS Internals.



Hình 1.13: Mô hình Single Threaded with Event loop

Các bước xử lý mô hình Single Threaded with Event Loop:

- Clients gửi yêu cầu tới Web Server.
- Node JS Web Server duy trì một Limited Thread Pool để cung cấp dịch vụ cho các yêu cầu của clients.
- Node JS Web Server nhận được các yêu cầu đó và đưa chúng vào hàng đợi. Nó được hiểu như là là "Event Queue".
- Node JS Web Server nội bộ có một Components, gọi là "Event Loop". Tại sao nó có tên này là vì nó sử dụng vòng lặp vô hạn để nhận yêu cầu và xử lý chúng.
- Event Loop sử dụng một Single Thread duy nhất. Đó là trọng tâm của mô hình Node JS Platform Processing Model.
- Event Loop kiểm tra bất kỳ client request nào được đặt trong Event Queue. Nếu không, nó sẽ chờ các request đến.
- Nếu có, nó chọn một yêu cầu client từ Event Queue

- Bắt đầu quá trình yêu cầu của client.
- Nếu client request không yêu cầu bất kỳ Blocking IO Operations, thì xử lý mọi thứ và chuẩn bị phản hồi và gửi lại cho client.
- Nếu client request yêu cầu một số Blocking IO Operations như tương tác với cơ sở dữ liệu, file systems, extenal services thì nó sẽ theo cách tiếp cận khác.
    - Kiểm tra các Thread có sẵn từ Internal Thread Pool
    - Chọn một Thread và chỉ định request client này cho Thread đó.
    - Thread này chịu trách nhiệm lấy request đó, xử lý nó, thực hiện các Blocking IO Operations, chuẩn bị phản hồi và gửi nó trở lại Event Loop
    - Event Loop lần lượt gửi phản hồi cho client tương ứng.

1.3. Tổng quan Socket.io

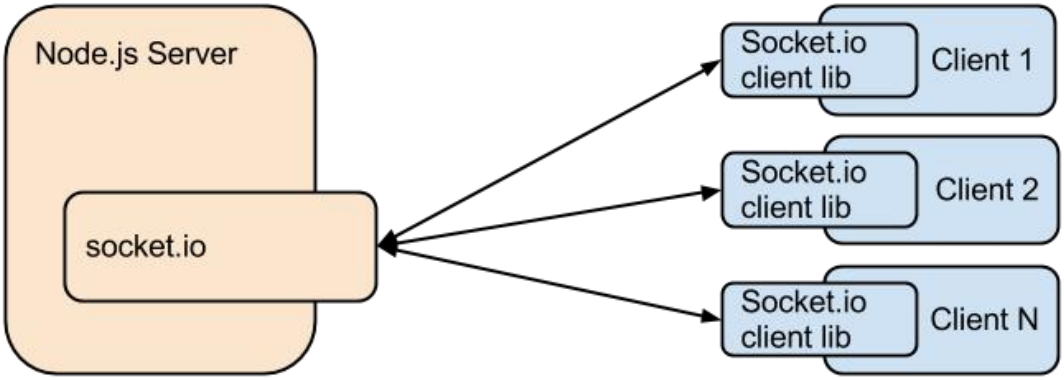
1.3.1. Cùng nhìn lại về Websocket

Trong trình duyệt mà được hỗ trợ WebSockets protocol, một kết nối giữa server và browser được tạo ra quanh HTTP và được gọi là một "HTTP hanshake" (cái bắt tay). Một kết nối được tạo ra trình duyệt và server mở ra một cổng giao tiếp liên tục thông qua một TCP socket. Nó sẽ hỗ trợ cho cả việc gửi và truy vấn message trên một cổng kết nối. Điều này giúp server load ít hơn, giảm số message bị trễ.

Tuy nhiên WebSocket mắc phải vấn đề đó là HTTP proxies, firewall và hosting provider. Khi Websocket sử dụng một phương thức giao tiếp ngoài HTTP, một phần nhiều trong số đó chưa được hỗ trợ và block bất cứ kết nối socket nào. Vấn đề này chỉ được giải quyết khi sử dụng thư viện trừu tượng mà có thể dễ dàng thay đổi giữa các giao thức dựa trên tài nguyên có sẵn.

Socket.io được xây dựng để giải quyết vấn đề này và nó luôn sẵn sàng được sử dụng cho NodeJS developer.

1.3.2. Giới thiệu về Socket.io



Hình 1.14: Giới thiệu socket.io

Socket.io được xây dựng nhằm mục đích tạo ra real-time NodeJS application. Socket.io cung cấp cho lập trình viên các đặc trưng như event, room và tự động phục hồi lại kết nối.

Khi chúng ta include Socket.io module vào trong ứng dụng của mình nó sẽ cung cấp cho chúng ta hai object đó là: socket server quản lý functionality phía server và socket client điều khiển funtionality phía client.

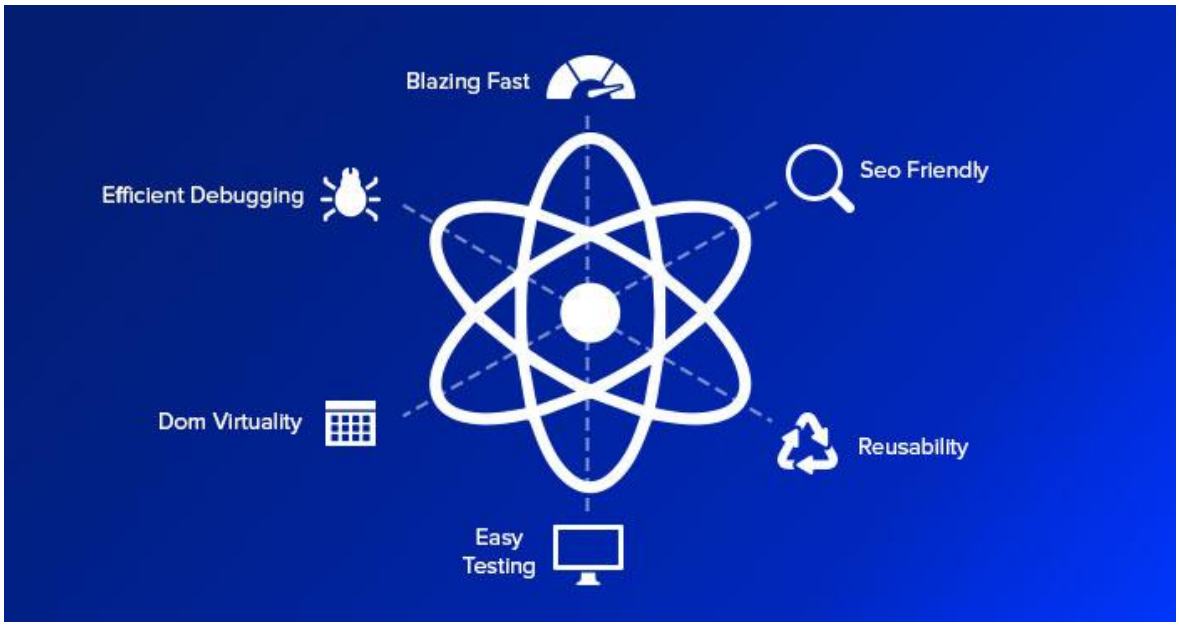
Khi client muốn kết nối tới Socket.io server, nó sẽ gửi cho server một "handshake HTTP request". Server sẽ phân tích request đó với những thông tin cần thiết trong suốt quá trình kết nối. Nó sẽ tìm cấu hình của middleware mà đã được đăng ký với server và thực thi chúng trước khi đưa ra sự kiện kết nối. Khi kết nối thành công thì connection event listener được thực thi, tạo ra một instance mới của socket có thể coi như định danh của client mà mỗi một client kết nối tới sẽ có 1 định danh.

Sự khác biệt rõ ràng nhất giữa Socket.IO và WebSockets:

- Socket.io cho phép send/emit messages bằng cách xác định bằng một tên event.
- Trong trường hợp của Socket.io: một message từ Server sẽ đến tất cả các Client đang kết nối. Với WebSockets: Nếu muốn điều tương tự như Socket.IO tôi cần phải tạo một mảng của tất cả các kết nối và lặp qua nó để gửi message đến tất cả các client.

1.4. Giới thiệu về React.js

1.4.1. React.js là gì?



Hình 1.15: React.js là gì?

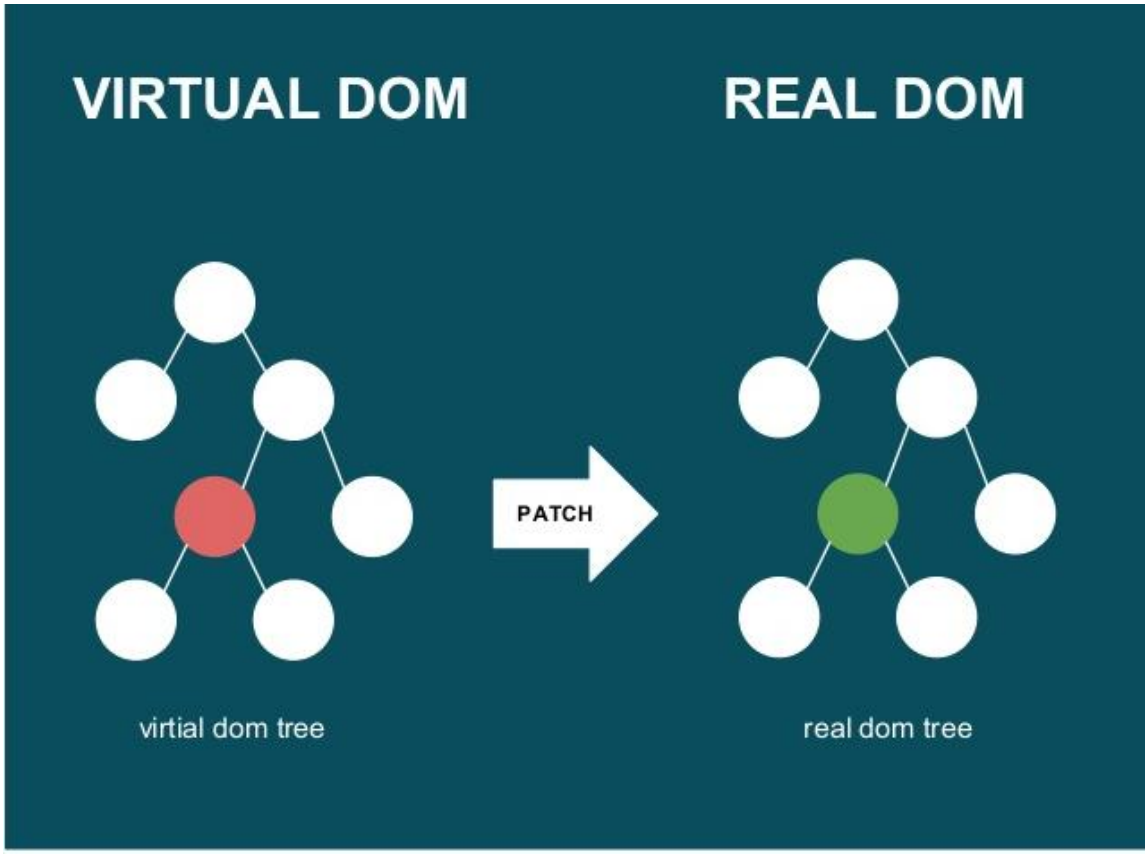
React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javasscript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX.Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM. Trước khi đến cài đặt và cấu hình, chúng ta sẽ đi đến một số khái niệm cơ bản:

1.4.2. Đặc điểm của React.js

1.4.2.1. Virtual DOM

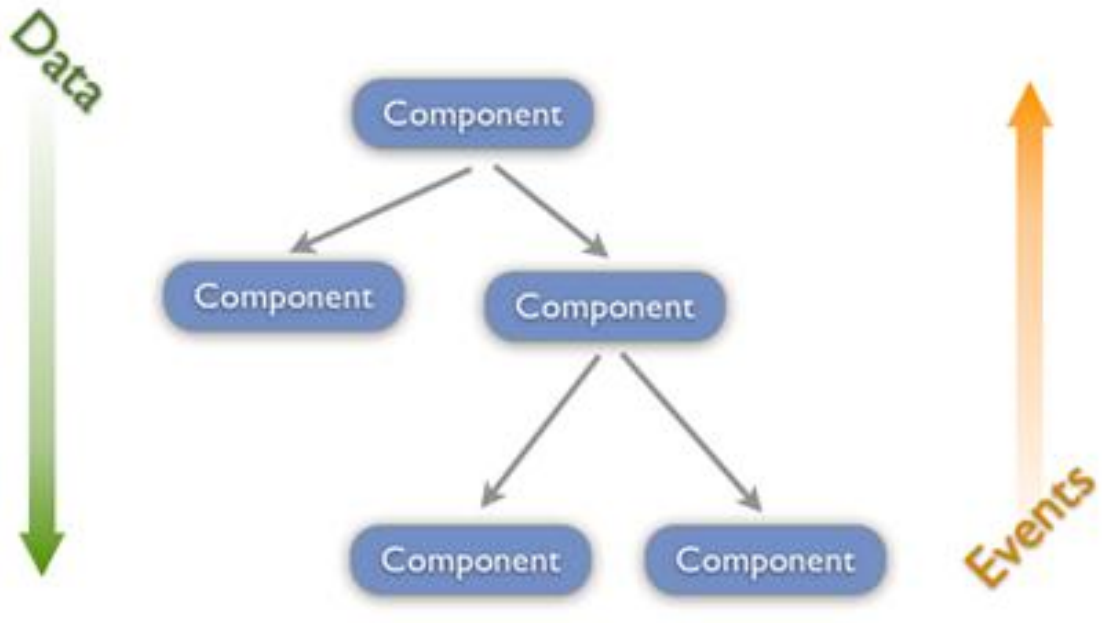
Virtual DOM là một object Javascript, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi nó sẽ tính toán sự thay đổi giữa object và tree thật, điều này sẽ giúp tối ưu hoá việc re-render DOM tree thật.



Hình 1.16: Virtual DOM trong React.js

Công nghệ DOM ảo giúp tăng hiệu năng cho ứng dụng. Việc chỉ node gốc mới có trạng thái và khi nó thay đổi sẽ tái cấu trúc lại toàn bộ, đồng nghĩa với việc DOM tree cũng sẽ phải thay đổi một phần, điều này sẽ ảnh hưởng đến tốc độ xử lý. React.js sử dụng Virtual DOM (DOM ảo) để cải thiện vấn đề này.

1.4.2.2. One-way Binding



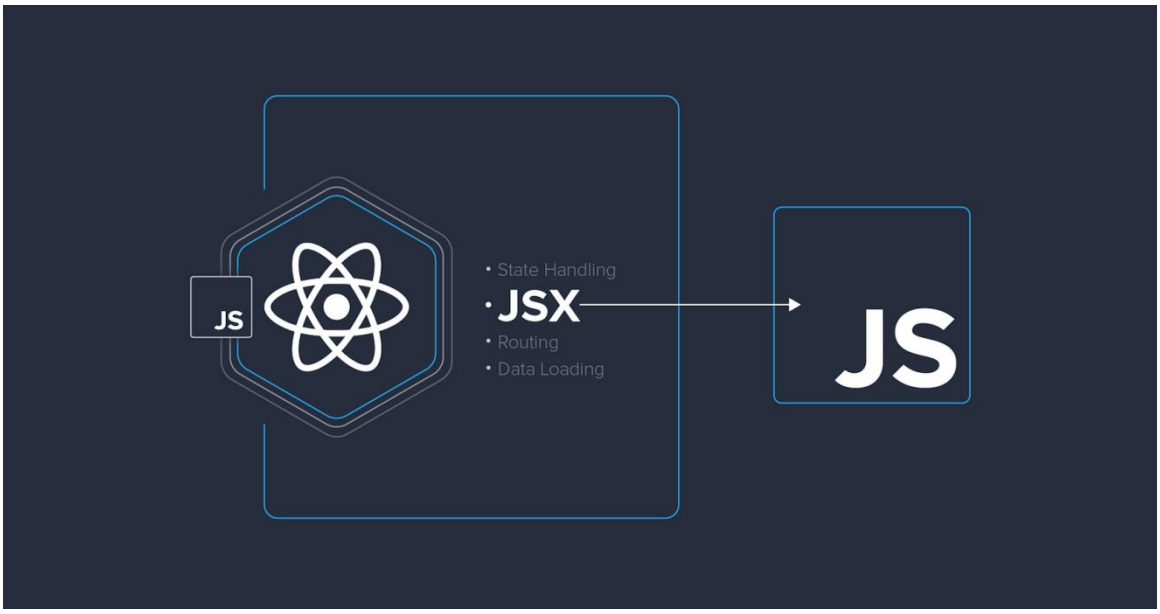
Hình 1.17: One-way Binding trong React.js

React sử dụng cơ chế one-way data binding – luồng dữ liệu 1 chiều. Dữ liệu được truyền từ parent đến child thông qua props. Luồng dữ liệu đơn giản giúp chúng ta dễ dàng kiểm soát cũng như sửa lỗi.

Với các đặc điểm ở trên, React dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian. Dữ liệu thay đổi thì hầu hết kèm theo sự thay đổi về giao diện. Ví dụ như Facebook: Trên Newsfeed cùng lúc sẽ có các status khác nhau và mỗi status lại có số like, share, comment liên tục thay đổi. Khi đó React sẽ rất hữu ích để sử dụng.

1.4.2.3. JSX

JSX hay có thể hiểu là Javascript Extension là dạng cú pháp cú pháp mở rộng cho javascript để hỗ trợ việc định nghĩa cấu trúc giao diện. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javacsript. Các mã này cho thời gian thực hiện nhanh hơn nhiều so với một mã tương đương viết trực tiếp bằng Javascript.

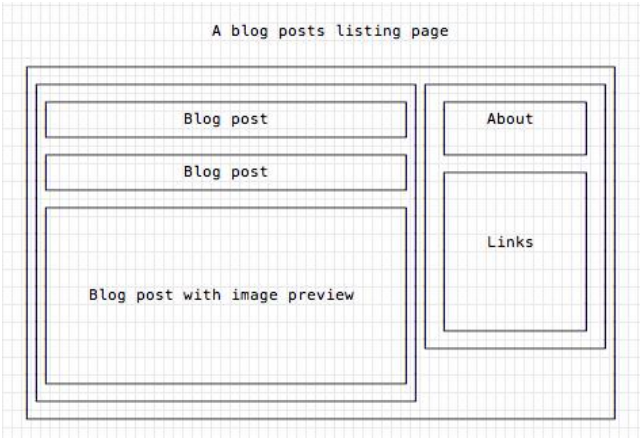


Hình 1.18: React JSX

Ngược với Javascript, JSX là kiểu statically-typed, nghĩa là nó được biên dịch trước khi chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt

Việc sử dụng JSX cho phép ta không những định nghĩa các thẻ giống như HTML mà còn cho phép ta truyền vào đó những đoạn code logic cần thiết để xử lý các sự kiện trên component của mình. Đồng thời cho phép ta gán nó vào một biến sau đó kết hợp với code javascript thông thường để thực hiện việc hiển thị giao diện

1.4.2.4. Components



Hình 1.19: React components

React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần (component) nhỏ.

Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái.

Mọi thứ React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một method render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.

Trong component có 2 dạng data cơ bản là:

- State: State chính là dữ liệu chỉ có thể thay đổi trong Component mà ta đã khai báo. Các Component khác không thể dử dụng cũng như thay đổi được (Nó cũng giống như phương thức private mà ta biết đối với Class).Muốn thay đổi state thì ta gọi hàm this.setState().

- Props: Giúp các component tương tác với nhau, component nhận input gọi là props, và trả thuộc tính mô tả những gì component con sẽ render. Bạn không thể thay đổi trực tiếp props trong component đã nhận nó vì giá trị props là bất biến.

1.5. Kết chương

Qua những nội dung đã nêu ở chương 1, chúng ta đã có một cái nhìn tổng quát về mặt cơ sở lý thuyết cũng như những công nghệ được áp dụng trong đồ án.

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1. Giới thiệu

Cùng với sự bùng nổ công nghệ, người dùng Internet, nhu cầu giao tiếp, chia sẻ thông tin, trao đổi dữ liệu ngày càng lớn. Về chia sẻ thông tin và dữ liệu, trên thế giới đã có rất nhiều hình thức với các công nghệ, giao thức, ứng dụng khác nhau, từ FTP, Email đến các hình thức chia sẻ P2P (Peer-to-Peer) như Bitorrent, hoặc ứng dụng dịch vụ cloud như Dropbox, OneDrive, Google Drive...

Về giao tiếp thời gian thực thì đã có những ứng dụng messenger rất thành công và được người dùng chào đón như Skype, Viber, Whatsapp, Line, Hangouts…Tuy nhiên, vì nhiều lý do từ tốc độ, bảo mật an toàn thông tin và đặc biệt là sự tiện dụng, vẫn tiếp tục có các nghiên cứu để đơn giản hóa việc giao tiếp, chia sẻ dữ liệu, hỗ trợ người dùng một cách nhanh nhất mà không đòi hỏi phải thao tác nhiều hay cài đặt thêm các plugin hoặc ứng dụng trên máy.

Cụ thể hơn, mong muốn sử dụng trình duyệt không chỉ để lướt web, check mail mà như là một công cụ hỗ trợ tất cả nhu cầu từ chia sẻ file đến giao tiếp thời gian thực từ lâu đã được nhen nhóm và thực sự phát triển mạnh từ năm 2009.

2.2. Mục đích và đối tượng hướng đến

2.2.1. Mục đích

Dựa trên kết quả nghiên cứu về lý thuyết cho WebRTC và hiện trạng của hệ thống hỗ trợ khách hàng trực tuyến của các website thương mại điện tử ở Việt Nam phải sử dụng các ứng dụng chat bên ngoài như Yahoo messenger và Skype, tôi đã tiến hành xây dựng một ứng dụng web video chat thời gian thực sử dụng WebRTC để cải tiến hệ thống hỗ trợ người dùng này.

2.2.2. Đối tượng

Hệ thống phục vụ người dùng có nhu cầu trao đổi thông tin trực quan với nhau một cách nhanh chóng chỉ bằng cách dùng Chrome hoặc Firefox gốc, không cần phải cài thêm bất kì một plugin nào.

2.3. Mô tả chung

2.3.1. Tổng quan:

Trước đây khi chưa có công nghệ WebRTC, chúng ta vẫn có thể thực hiện các cuộc gọi video, audio và chat trên trình duyệt, tuy nhiên nó đòi hỏi phải cài đặt thêm các plugin cho trình duyệt, và thậm chí cả hai người thực hiện cuộc gọi cùng phải cài đặt một loại plugin. Và nếu người sử dụng chuyển sang một máy tính khác hoặc sử dụng một trình duyệt web khác, thì lại phải cài đặt lại plugin để có thể thực hiện cuộc gọi được. Việc sử dụng plugin thường hay gặp phải các vấn đề về bảo mật và gây khó khăn cho người sử dụng. Hệ thống được xây dựng nhằm thực hiện một ứng dụng web có các tính năng tương tự như các tính năng Skype™ cung cấp mà không cần phải cài đặt thêm bất kỳ phần mềm hay plug-in nào của bên thứ ba nào.

2.3.2. Chức năng

Các chức năng chính của hệ thống được mô tả như sau:

- Tham gia vào room meeting không cần mật khẩu mà chỉ cần biết tên phòng hoặc có URL của room.
- Chức năng chia sẻ video từ camera của thiết bị sử dụng.
- Người dùng có thể tùy ý lựa chọn các cài đặt về camera cũng như audio trong khi chia sẻ như: Đổi camera, đổi audio, kiểm tra audio output,...
- Nếu không muốn chia sẻ camera hoặc audio của mình nữa người dùng có thể mute trạng thái camera, audio tùy ý.
- Người dùng cũng có thể chia sẻ toàn màn hình hoặc một phần màn hình của mình cho những người trong room.
- Hệ thống cho phép những người dùng trong cùng một room có thể gửi tin nhắn cho nhau.

2.3.3. Môi trường hoạt động

Hệ thống hoạt động trên môi trường web, có thể chạy trên các trình duyệt web có kết nối internet.

2.3.4. Yêu cầu giao diện

Giao diện dễ nhìn, trực quan và hiển thị được đầy đủ thông tin nhất. Giao diện chia thành nhiều khu vực, nhưng phải đảm bảo sự thống nhất và không gây rối mắt khi thao tác.

2.3.5. Yêu cầu phi chức năng

2.3.5.1. Hiệu suất

Hệ thống phải đạt độ ổn định tương đối, đáp ứng tốc độ tải trang nhất định với lượng truy cập không lớn. Hệ thống đảm bảo được sự tối ưu trong việc tải các thành phần của trang, không gây lãng phí tài nguyên.

2.3.5.2. Yêu cầu an toàn

Hệ thống hoạt động độc lập, không phụ thuộc vào bất kỳ phần mềm nào, không ảnh hưởng đến các phần mềm khác.

2.3.5.3. Yêu cầu bảo mật

Hệ thống đảm bảo các thông tin người dùng phải được bảo mật nhất, không sử dụng các thư viện JavaScript không rõ nguồn gốc.

2.3.5.4. Yêu cầu chất lượng

Hệ thống phải đảm bảo chất lượng tốt, hoạt động tốt, ít các lỗi cơ bản. Tiết kiệm được dung lượng web, giảm băng thông. Chỉ sử dụng các thư viện cần thiết và sử dụng đúng chỗ.

2.4. Thiết kế hệ thống

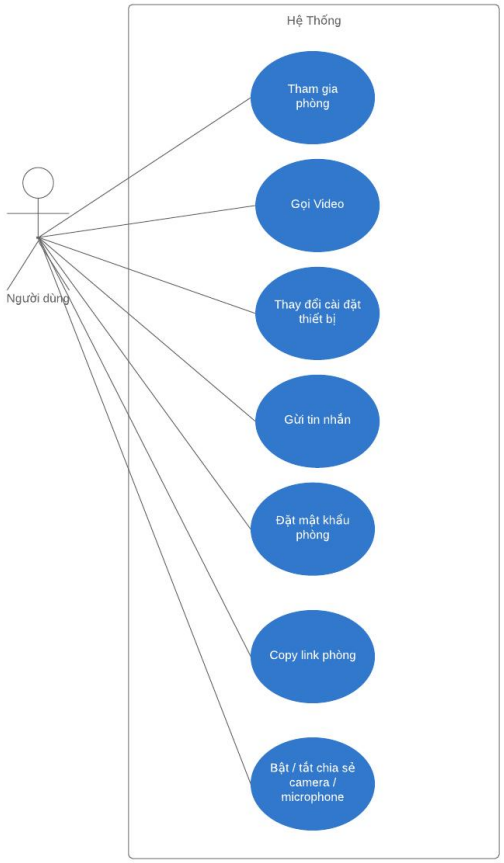
2.4.1. Các tác nhân trong hệ thống

Hệ thống có tác nhân sau: Người dùng sử dụng các browser hiện đại hỗ trợ WebRTC như Chrome, Firefox,...

2.4.2. Các biểu đồ

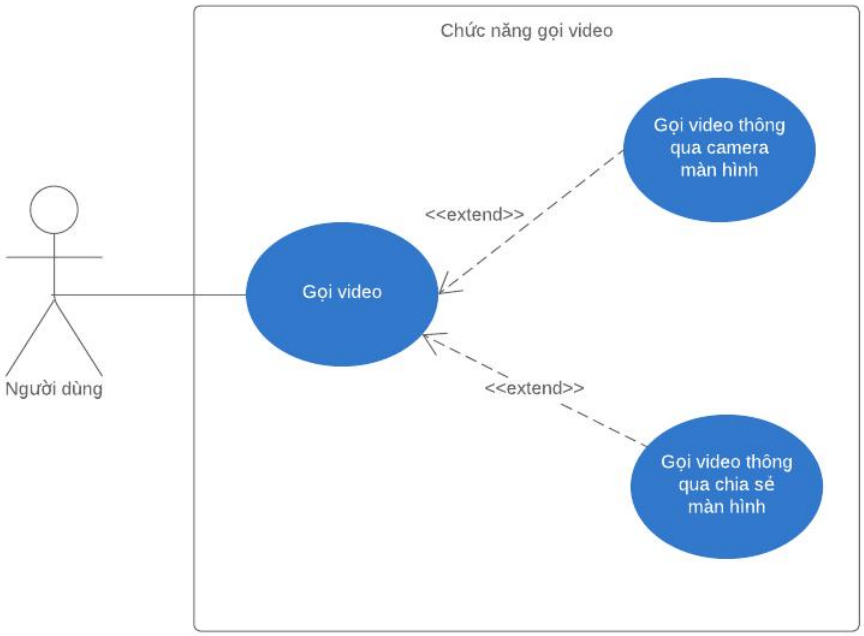
2.4.2.1. Biểu đồ ca sử dụng

2.4.2.1.1. Biểu đồ usecase người dùng



Hình 2.1: Biểu đồ ca sử dụng chức năng của người dùng

Giải thích: Đây là biểu đồ ca sử dụng tổng quát các chức năng mà người dùng có thể sử dụng trong hệ thống. Những chức năng đó bao gồm: Tham gia phòng, gọi video, thay đổi cài đặt các thiết bị, gửi tin nhắn, đổi mật khẩu phòng, copy link phòng, bật/tắt chia sẻ camera/microphone.



Hình 2.2: Biểu đồ ca sử dụng chức năng gọi video

Giải thích: Hình 2.2 mô tả chức năng gọi video mà người dùng có thể sử dụng trong hệ thống. Đối với gọi video, người dùng có thể chia sẻ video của mình thông qua camera hoặc trạng thái màn hình hiện tại.

2.4.2.1.2. Đặc tả một số ca sử dụng

Bảng 2.1: Chức năng tham gia phòng

Tên chức năng	Tham gia phòng
Độ ưu tiên	Cao
Kích hoạt	Truy cập trang chủ
Điều kiện trước	Truy cập trang web
Mô tả xử lý	Hệ thống sẽ tạo ra một form input để nhập tên phòng sẽ tham gia meeting.
Điều kiện sau	Những thay đổi sẽ được thực hiện.
Xử lý ngoại lệ	Nếu người dùng không yêu cầu gì thêm có thể thoát khỏi trang web.

Kịch bản	**Tác nhân**	**Hệ thống**
	Nhập tên phòng	Hiển thị màn hình nhập tên phòng .
	Nhấn nút OK.	Chuyển hướng đến trang gọi video chat.

Bảng 2.2: Chức năng gọi video

Tên chức năng	Gọi Video	
Độ ưu tiên	Cao	
Kích hoạt	Truy cập đường dẫn /meeting/:roomName	
Điều kiện trước	Truy cập trang web	
Mô tả xử lý	Hệ thống sẽ chia sẻ video của người dùng từ camera hoặc từ màn hình đến những người khác	
Điều kiện sau	Những thay đổi sẽ được thực hiện.	
Xử lý ngoại lệ	Nếu người dùng không yêu cầu gì thêm có thể thoát khỏi trang web.	
Kịch bản	**Tác nhân**	**Hệ thống**
	Cho phép sử dụng camera / audio	Hiển thị popup cho phép sử dụng camera / audio
	Click nút cho phép	Chia sẻ hình ảnh, âm thanh của người dùng đến người khác
	Click nút Share Screen	Chia sẻ màn hình của người dùng

Bảng 2.3: Chức năng thay đổi cài đặt thiết bị

Tên chức năng	Thay đổi cài đặt thiết bị	
Độ ưu tiên	Trung bình	
Kích hoạt	Truy cập đường dẫn /meeting/:roomName	
Điều kiện trước	Truy cập trang web	
Mô tả xử lý	Hệ thống sẽ cho phép người dùng thay đổi những cài đặt về camera hoặc microphone	
Điều kiện sau	Những thay đổi sẽ được thực hiện.	
Xử lý ngoại lệ	Nếu người dùng không yêu cầu gì thêm có thể thoát khỏi trang web.	
Kịch bản	**Tác nhân**	**Hệ thống**
	Click nút Settings	Hiển thị modal thay đổi cài đặt
	Click nút thay đổi.	Thay đổi camera / microphone dựa theo cài đặt của người dùng.

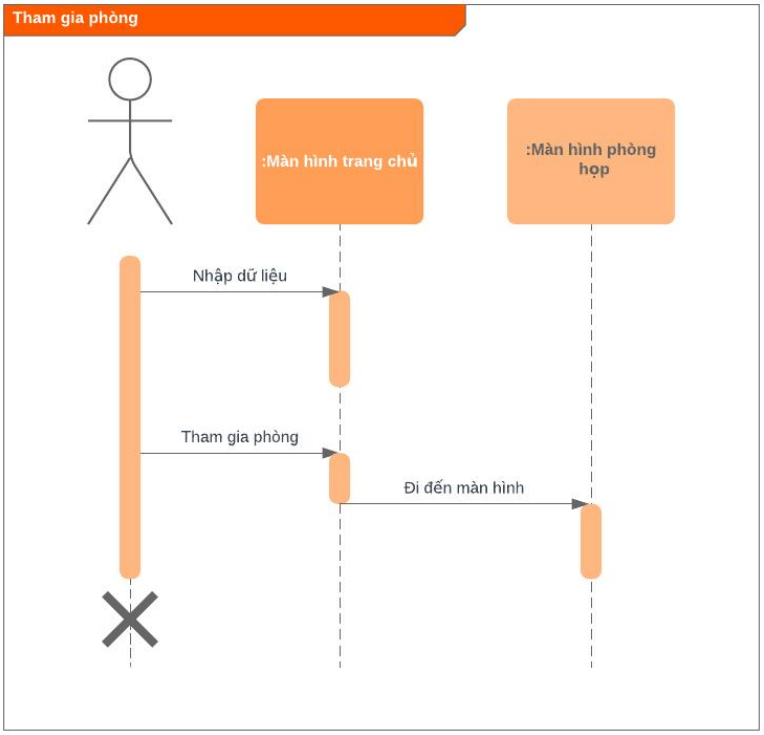
Bảng 2.4: Chức năng gửi tin nhắn chat

Tên chức năng	Gửi tin nhắn chat
Độ ưu tiên	Cao
Kích hoạt	Truy cập đường dẫn /meeting/:roomName
Điều kiện trước	Truy cập trang web
Mô tả xử lý	Hệ thống sẽ gửi tin nhắn của người dùng đến mọi người trong room
Điều kiện sau	Những thay đổi sẽ được thực hiện.

Xử lý ngoại lệ	Nếu người dùng không yêu cầu gì thêm có thể thoát khỏi trang web.	
Kịch bản	**Tác nhân**	**Hệ thống**
	Click vào Chat	Hiển thị drawer chat ở bên trái màn hình
	Gửi tin nhắn Chat	Gửi tin nhắn của người dùng gửi lên cho mọi người trong cùng room đó

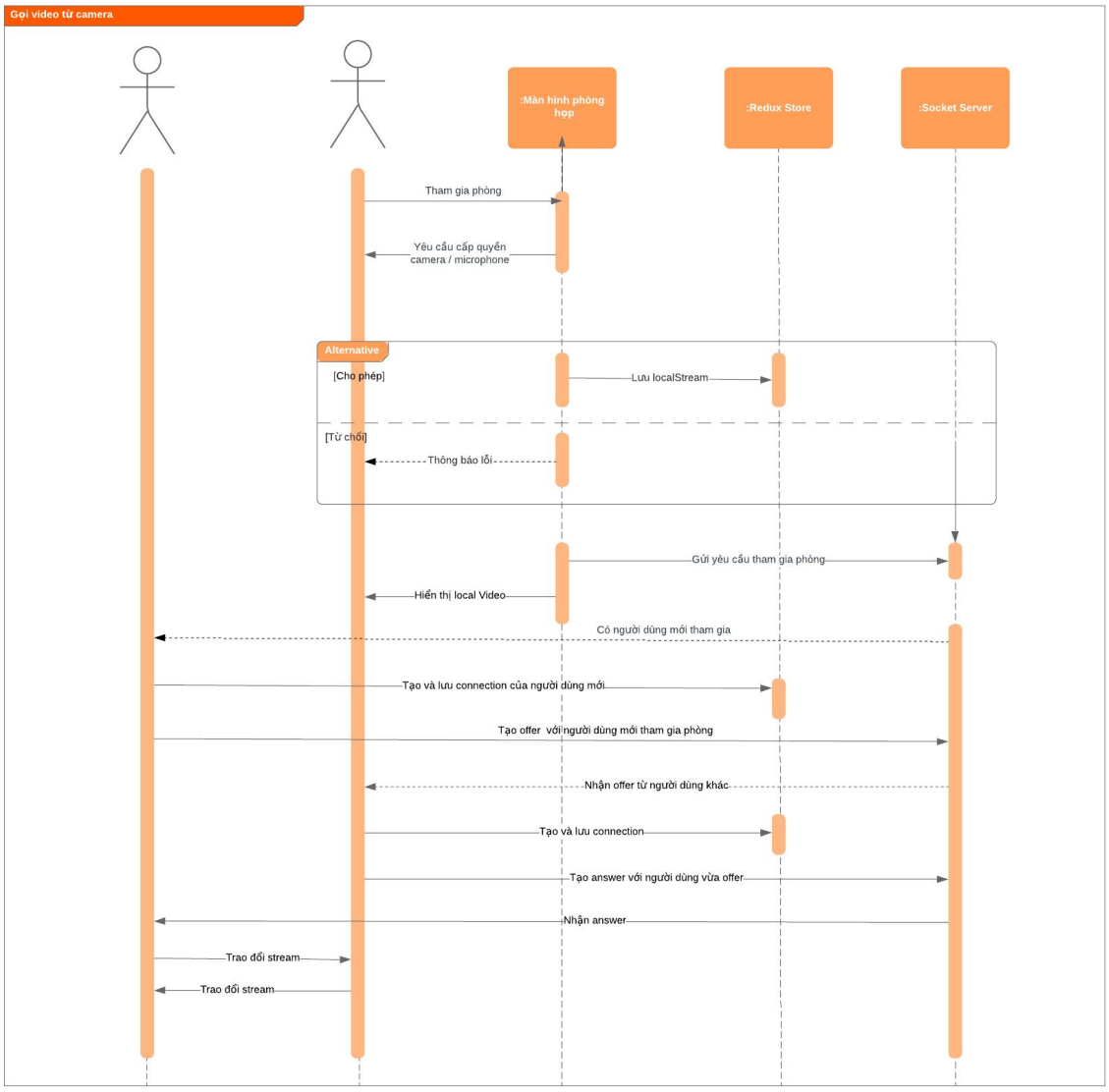
2.4.2.2. Biểu đồ tuần tự

2.4.2.2.1. Chức năng tham gia phòng



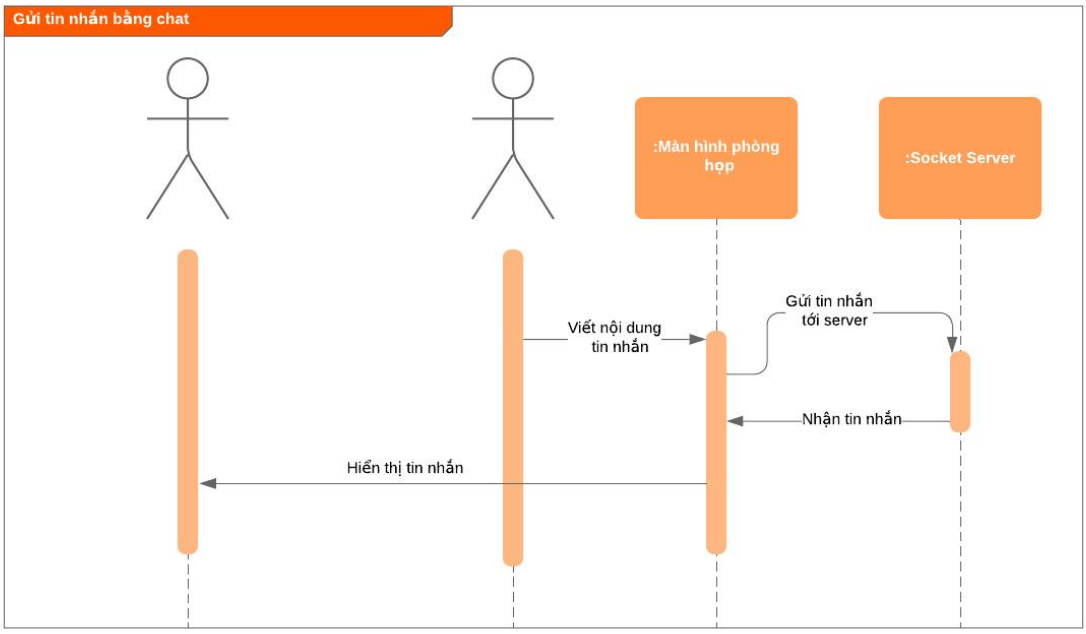
Hình 2.3: Biểu đồ tuần tự chức năng tham gia phòng

2.4.2.2.2. Chức năng gọi video



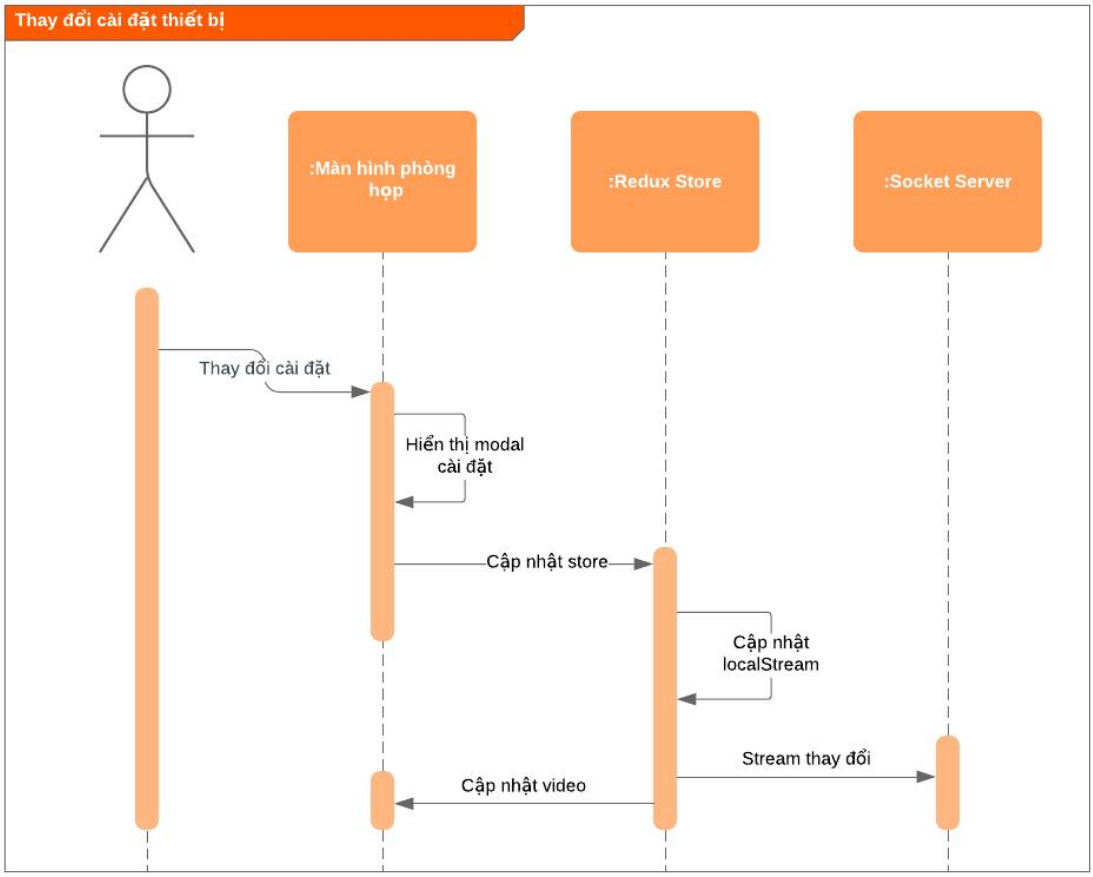
Hình 2.4: Biểu đồ tuần tự chức năng gọi video

2.4.2.2.3. Chức năng gửi tin nhắn chat



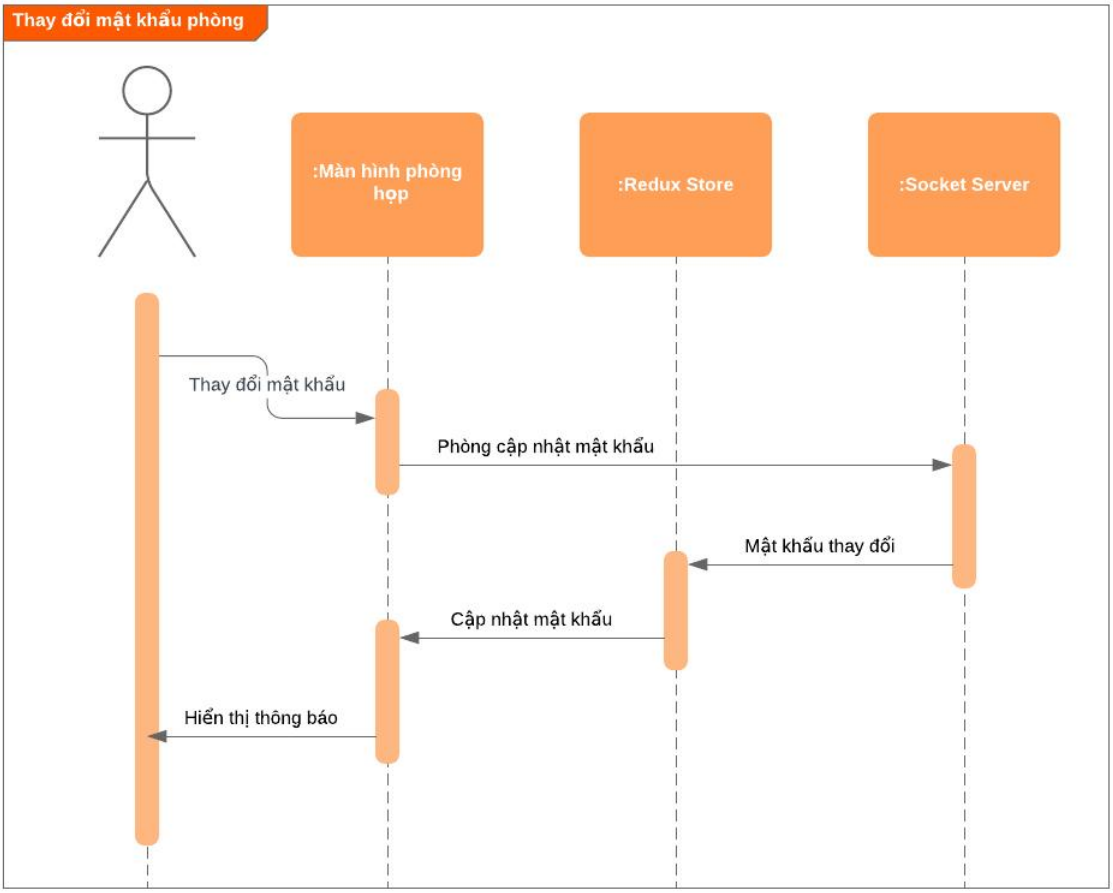
Hình 2.5: Biểu đồ tuần tự chức năng gửi tin nhắn chat

2.4.2.2.4. Chức năng thay đổi cài đặt thiết bị



Hình 2.6: Biểu đồ tuần tự chức năng thay đổi cài đặt thiết bị

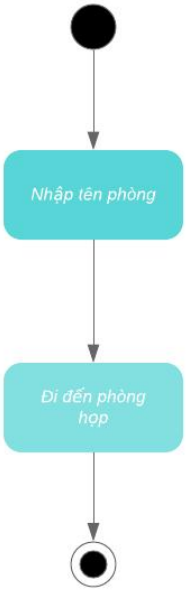
2.4.2.2.5. Chức năng thay đổi mật khẩu phòng



Hình 2.7: Biểu đồ tuần tự chức năng thay đổi mật khẩu phòng

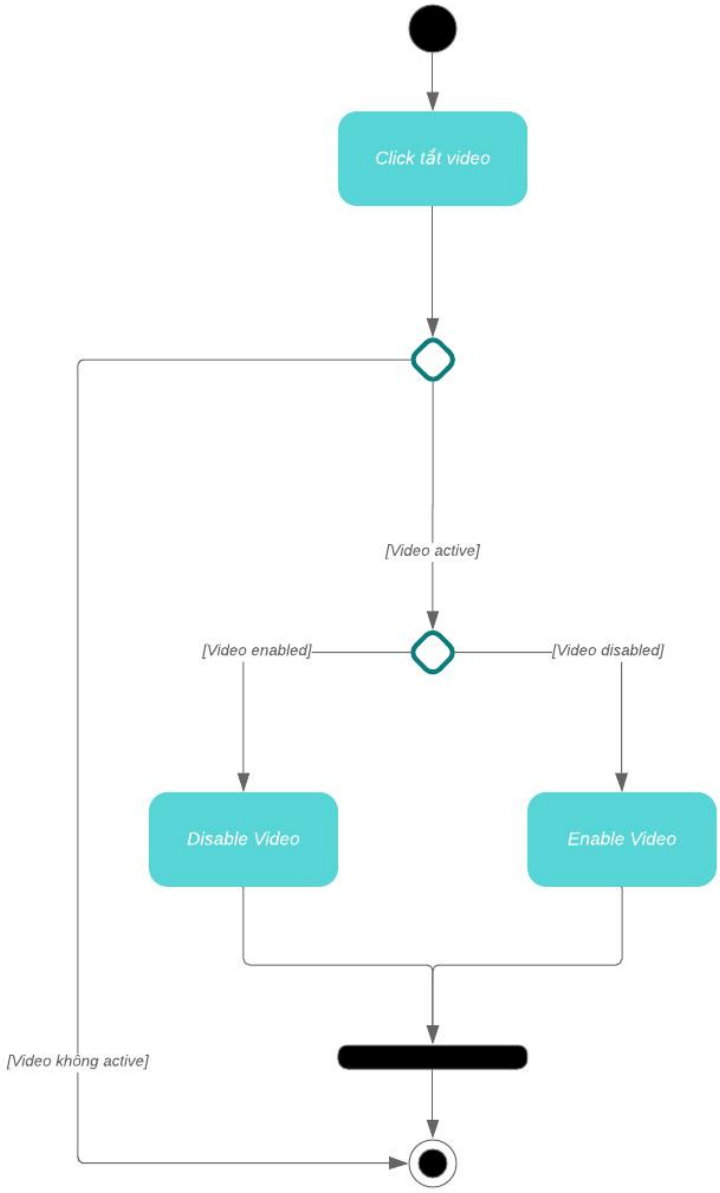
2.4.2.3. Biểu đồ hoạt động

2.4.2.3.1. Chức năng tham gia phòng



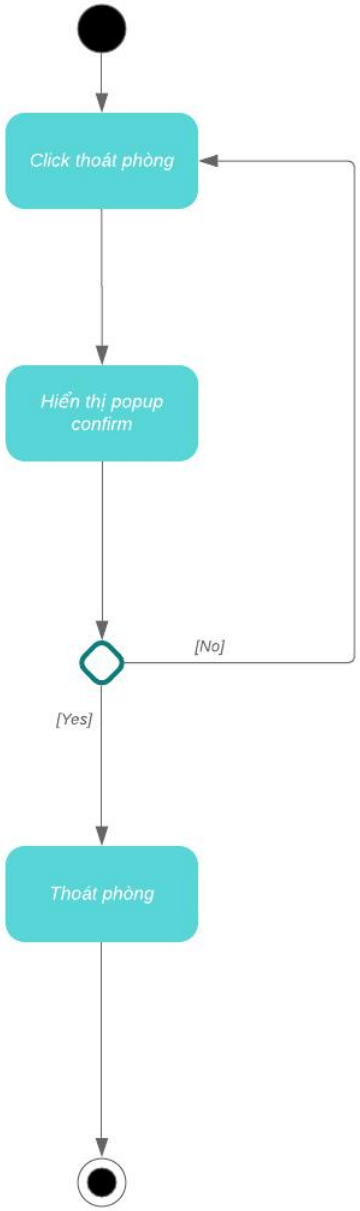
Hình 2.8: Biểu đồ hoạt động chức năng tham gia phòng

2.4.2.3.2. Chức năng bật tắt camera/microphone



Hình 2.9: Biểu đồ hoạt động chức năng bật hoặc tắt camera/microphone

2.4.2.3.3. Chức năng thoát phòng

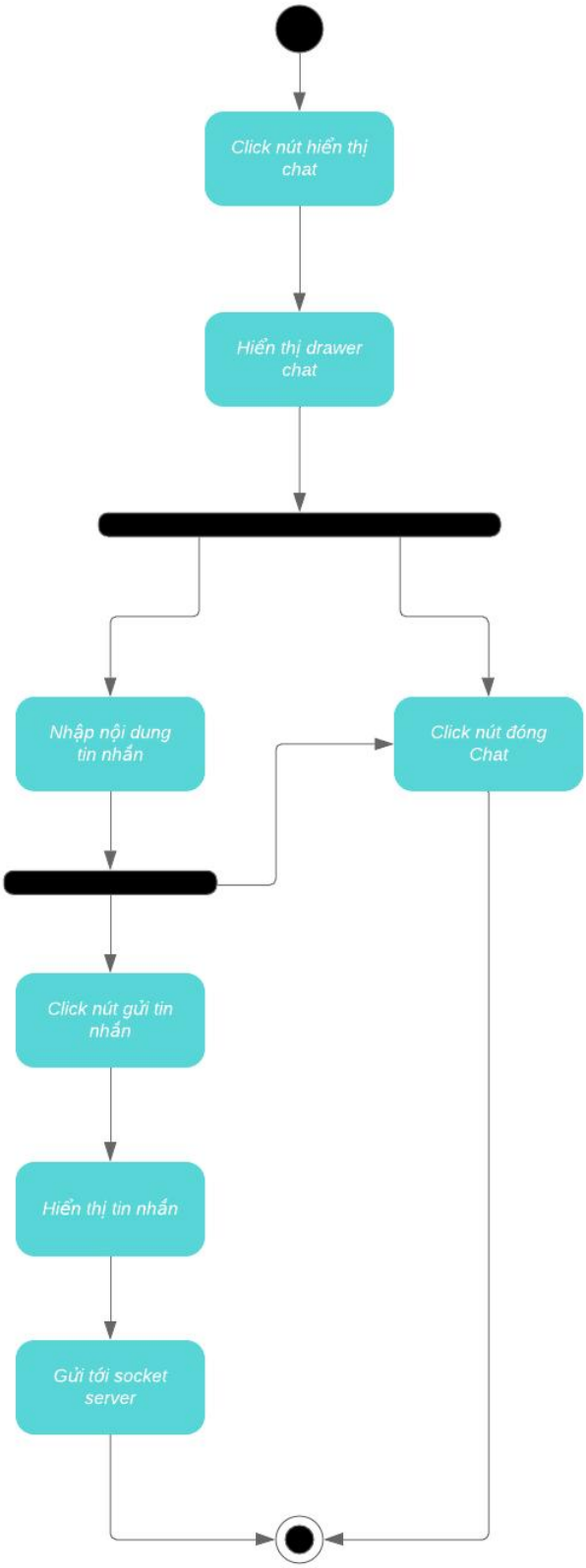


Hình 2.10: Biểu đồ hoạt động chức năng thoát phòng

2.4.2.3.4. Chức năng copy link

Hình 2.11: Biểu đồ hoạt động chức năng copy link

2.4.2.3.5. Chức năng chat



Hình 2.12: Biểu đồ hoạt động chức năng chat

2.5. Kết chương

Qua chương này chúng ta đã nắm rõ được bài toán mà hệ thống phải giải quyết, thông qua các bước phân tích, cấu trúc hệ thống, các luồng xử lý trong hệ thống được biểu diễn rõ ràng và chi tiết.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Triển khai hệ thống

3.1.1. Môi trường triển khai

Hệ thống triển khai trên nền web, chạy được trên các trình duyệt hiện đại có kết nối internet và hỗ trợ WebRTC như Chrome, Firefox,...

3.1.2. Cài đặt môi trường

3.1.2.1. Cài đặt Node.js

- Thêm PPA:

```
sudo apt-get install curl python-software-properties
```

- Cài đặt Node.js:

```
sudo apt-get install nodejs
```

3.1.2.2. Cài đặt React.js

- Cài đặt từ NPM

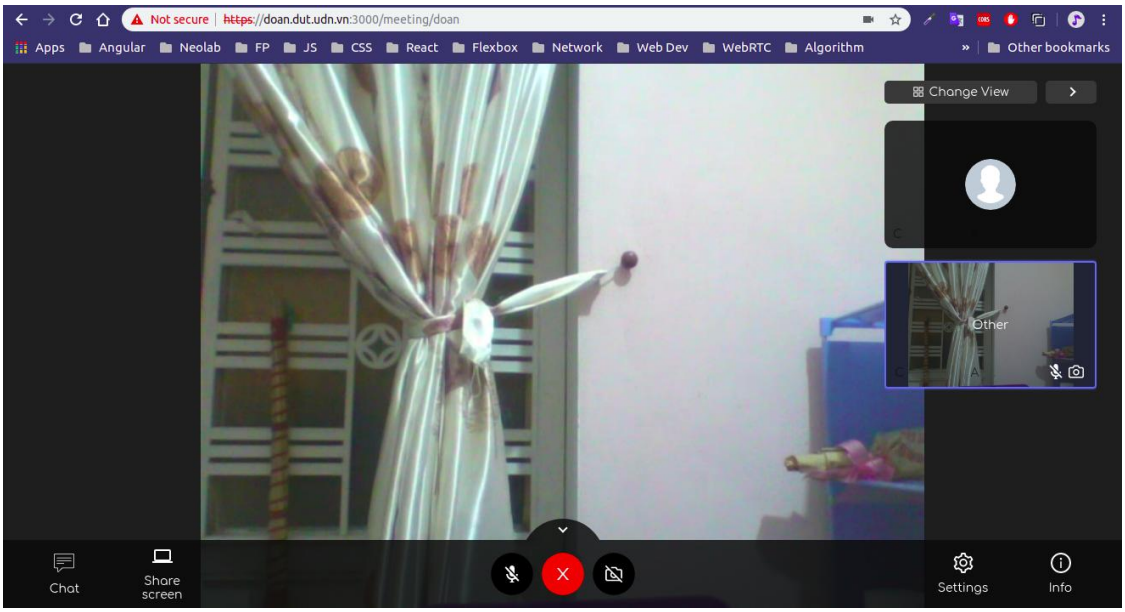
```
npm init react-app <app-name>
```

- Cài đặt từ YARN

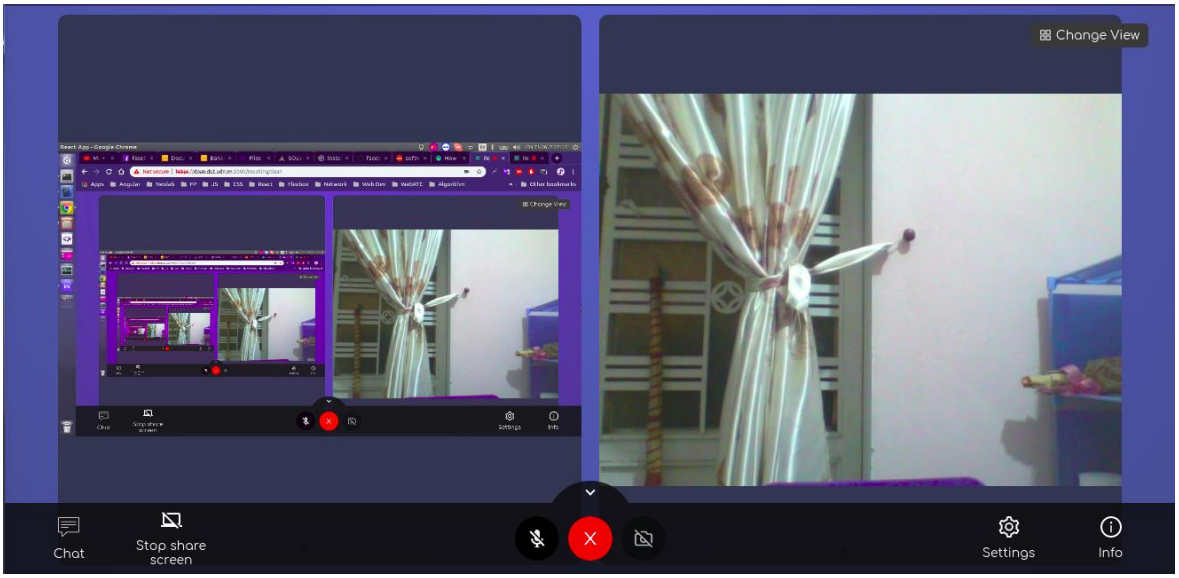
```
yarn create react-app <app-name>
```

3.1.3. Kết quả thực nghiệm

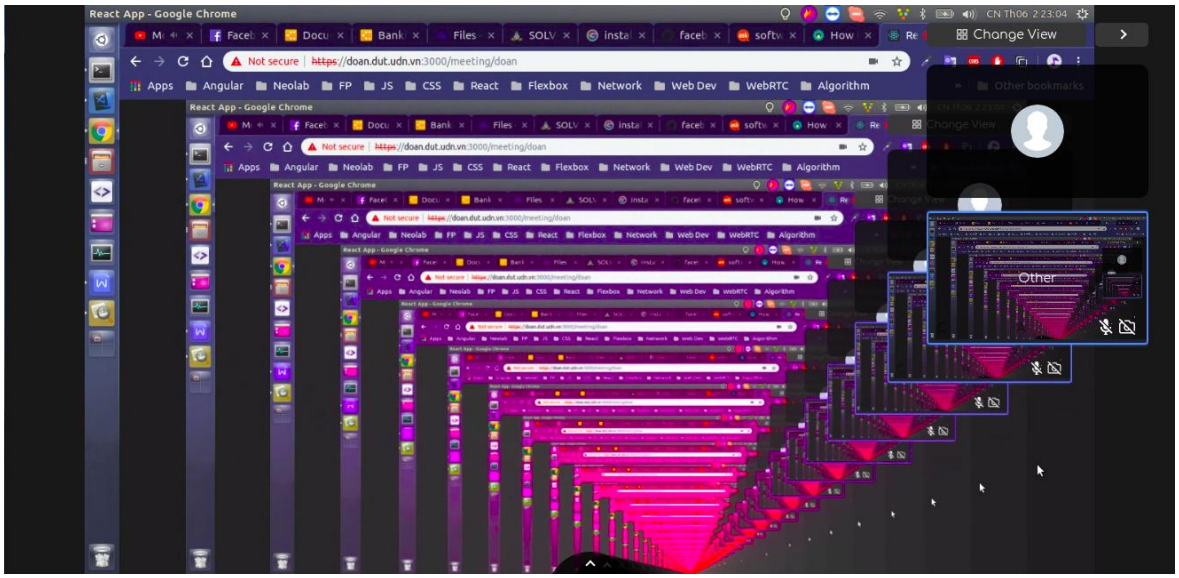
3.1.3.1. Giao diện người dùng



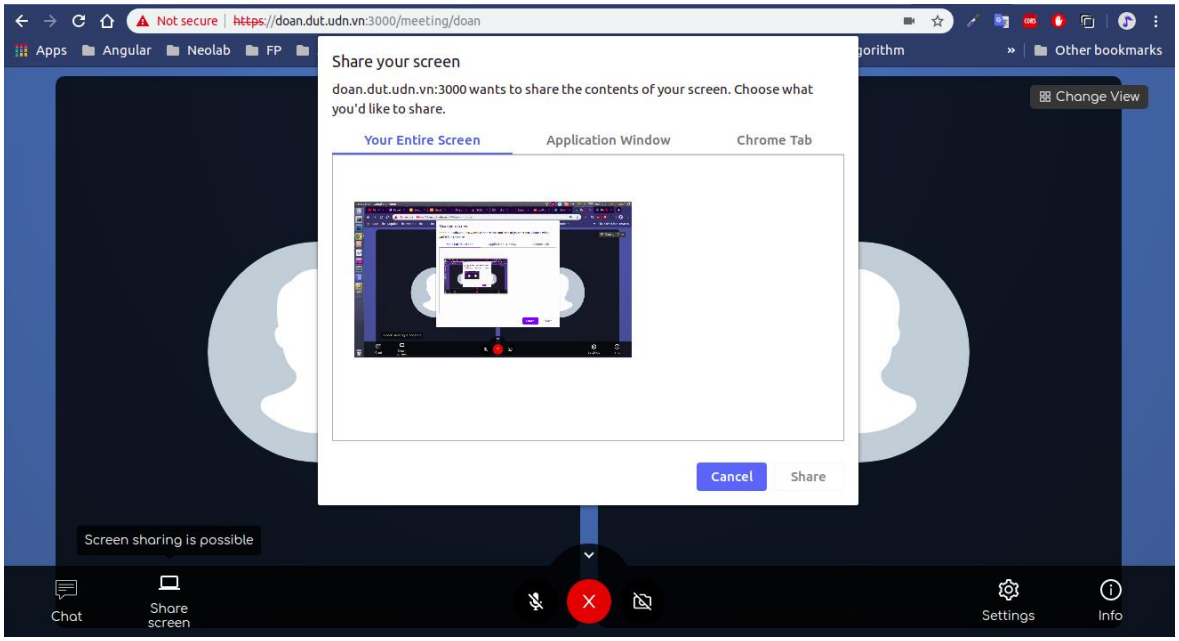
Hình 3.1: Giao diện dạng list



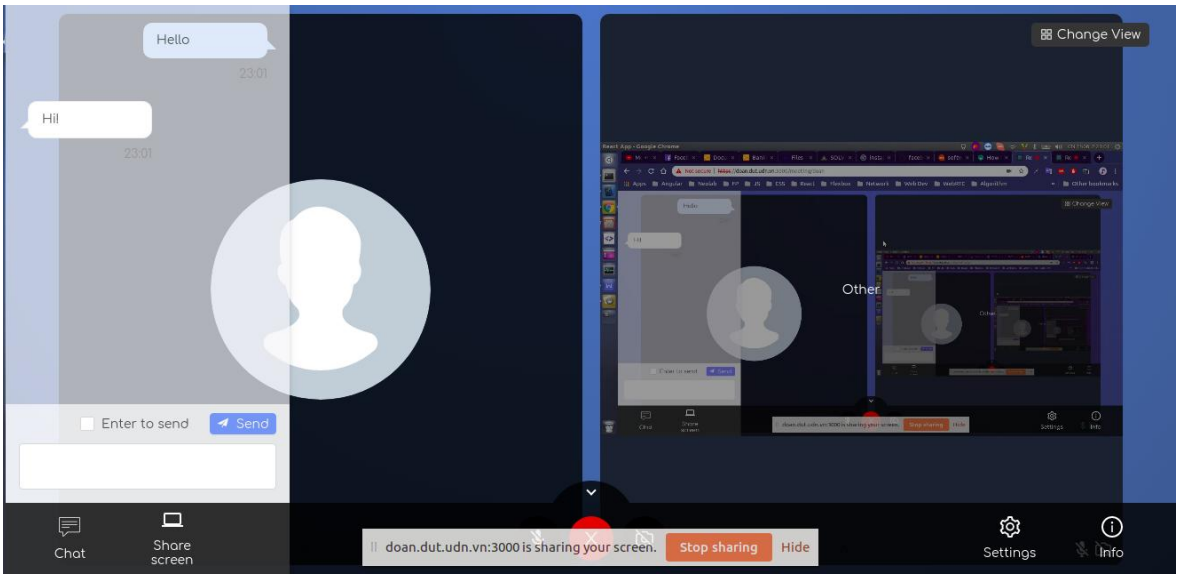
Hình 3.2: Giao diện dạng lưới



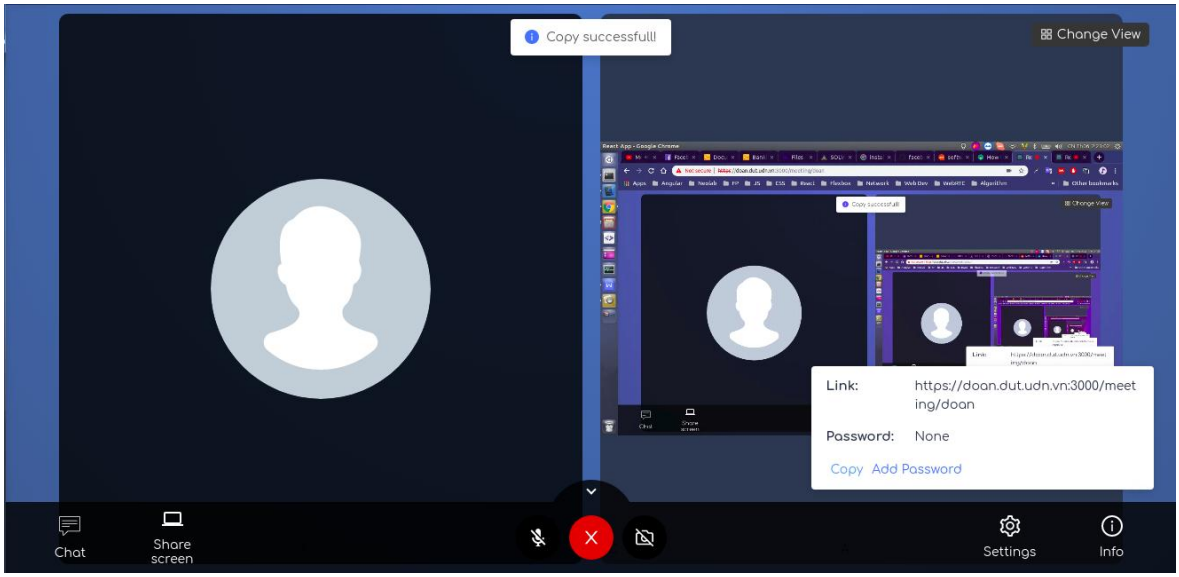
Hình 3.3: Giao diện khi ẩn toolbar



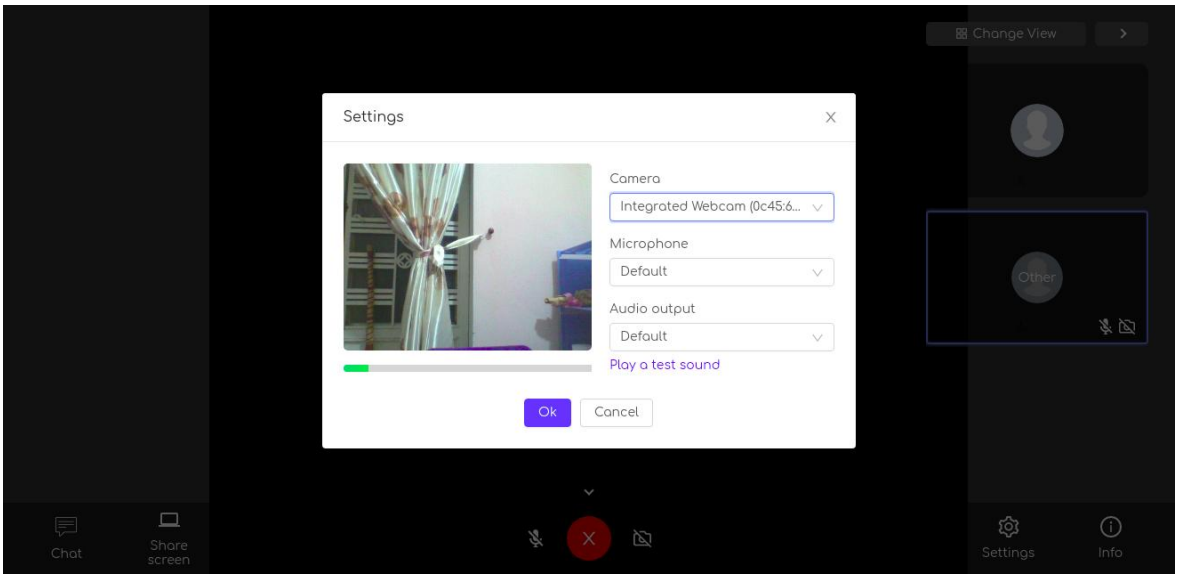
Hình 3.4: Giao diện khi chia sẻ màn hình



Hình 3.5: Giao diện chat



Hình 3.6: Giao diện thông tin phòng



Hình 3.7: Giao diện thay đổi cài đặt thiết bị

3.1.4. Nhận xét và đánh giá kết quả

Bảng 3.1: Đánh giá kết quả

Chức năng	Tình trạng	Mô tả
Đăng nhập, đăng xuất	Hoàn thành	
Đăng ký	Hoàn thành	
Xác thực email	Hoàn thành	
Tạo mới địa điểm	Hoàn thành	
Xem chỉ dẫn tới địa điểm	Hoàn thành	
Đánh giá, bình luận địa điểm	Hoàn thành	
Tìm kiếm địa điểm	Hoàn thành	
Lọc các địa điểm trong khu vực	Hoàn thành	
Đặt món online	Hoàn thành	
Quản lý người dùng	Hoàn thành	

Quản lý địa điểm	Hoàn thành	
Thống kê hệ thống	Hoàn thành	

3.2. Kết chương

Chương này đã cung cấp được các bước triển khai thực tế của đồ án, các hình ảnh về sản phẩm lúc hoạt động thực tế.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

Trong thời gian tìm hiểu, nghiên cứu cơ sở lý thuyết và triển khai ứng dụng công nghệ, đồ án đã đạt được những kết quả sau:

a. Về mặt lý thuyết, đồ án đã đạt được:

Hiểu được lý thuyết cơ bản của lập trình web.

Các bước triển khai hệ thống để hoàn thiện một sản phẩm trên thực tế.

Tìm hiểu nhu cầu của người dùng về lĩnh vực ẩm thực.

Phân tích thiết kế hệ thống, các luồng hoạt động cũng như các ca hoạt động cho một ứng dụng web nhiều chức năng.

Cách ứng dụng framework vào một dự án.

b. Về mặt thực tiễn ứng dụng, đồ án đã đạt được:

Cung cấp được một kênh thông tin để có thể chia sẻ và chỉ dẫn các địa điểm ẩm thực, các chức năng cơ bản và cần thiết cho một hệ thống.

Hệ thống đã hoàn thiện các chức năng như:

Đăng nhập, đăng xuất

Đăng ký

Xác thực email

Tạo mới địa điểm

Xem chỉ dẫn tới địa điểm

Đánh giá, bình luận địa điểm

Tìm kiếm địa điểm

Lọc các địa điểm trong khu vực

Đặt món online

Quản lý người dùng

Quản lý địa điểm

Thống kê hệ thống

2. Những điều chưa đạt được

Giao diện người dùng còn khá đơn giản.

Các chức năng như thông báo còn chưa hoàn thiện.

Các cài đặt, chức năng cá nhân cho thành viên còn ít.

3. Hướng nghiên cứu và phát triển

Một số số hướng nghiên cứu và phát triển của đề tài như sau:

Cải thiện giao diện người dùng

Hoàn thiện các tính năng đang có của hệ thống

Bổ sung chức năng quảng bá địa điểm, thông báo về địa điểm có khuyến mãi cho thành viên

Thêm các phương thức thanh toán cho chức năng đặt món

Bổ sung chức năng đặt bàn và thêm bình luận video cho địa điểm

TÀI LIỆU THAM KHẢO

[1] Trang chủ Ruby: https://www.ruby-lang.org/vi/

[2] Trang chủ Ruby on Rails: http://rubyonrails.org/

[3] Trang Stackoverflow: http://stackoverflow.com/

[4] Trang chia sẻ kiến thức công nghệ Viblo: https://viblo.asia /

[5] Ruby on Rails Tutorial (Rails 5): https://www.railstutorial.org/book /

[6] Google maps Javascript api:
https://developers.google.com/maps/documentation/javascript/directions

https://developers.google.com/maps/documentation/javascript/examples/map-geolocation

[7] Wikipedia: https://wikipedia.org/