**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----------o0o----------**

****

**BÁO CÁO**

**THỰC TẬP TỐT NGHIỆP**

**Đề tài: Nghiên cứu công nghệ WebRTC và áp dụng trong app gọi điện trực truyến**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn | : **Ts. Nguyễn Bá Nghiễn** |
| Sinh viên thực hiện | : Triệu Tài Niêm |
| Lớp | : ĐH KTPM1 - K11 |
| Mã sinh viên | : 1141360005 |

Hà Nội 2019-2020

**LỜI CẢM ƠN**

Không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Để có thể hoàn thành được đề tài thực tập tốt nghiệp này, em xin chân thành cảm ơn thầy **Nguyễn Bá Nghiễn**, người đã tận tình hướng dẫn, giúp đỡ, chỉ bảo và cũng là người luôn đưa ra những lời khuyên bổ ích cho em trong suốt quá trình hoàn thiện bài tập lớn này.

Trong một khoảng thời gian chưa nhiều, em đã cố gắng, nỗ lực tìm hiểu để hoàn thành bài tập lớn với đúng mục đích đề ra. Tuy nhiên, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những đóng góp phản hồi quý báu từ thầy cô và các bạn để em có thể hoàn thiện và tiếp tục phát triển đề tài. Em xin chân thành cảm ơn!

**Mục lục**

[CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 4](#_Toc33559615)

[1.1 Tên đề tài 4](#_Toc33559616)

[1.2 Lý do chọn đề tài 4](#_Toc33559618)

[1.3 Mục đích chọn đề tài 4](#_Toc33559619)

[CHƯƠNG 2. TÌM HIỂU VỀ WEBRTC 5](#_Toc33559620)

[2.1 Giới thiệu công nghệ WebRTC 5](#_Toc33559621)

[2.2 Lịch sử phát triển 5](#_Toc33559622)

[2.3 Những lợi ích của WebRTC 5](#_Toc33559623)

[2.4 Các khái niệm liên quan đến WebRTC 6](#_Toc33559624)

[2.4.1 Mạng ngang hàng (P2P) 7](#_Toc33559625)

[2.4.2 Tường lửa (Firewall) 8](#_Toc33559626)

[2.4.3 NAT - Network Address Translation 8](#_Toc33559627)

[2.4.4 STUN (Simple Traversal Of UDP Through NAT) 9](#_Toc33559628)

[2.4.5 TURN (Traversal Using NAT Relay) 9](#_Toc33559629)

[2.4.6 ICE (Interactive Communication Establishment) 10](#_Toc33559630)

[2.5 Kiến trúc của WebRTC 10](#_Toc33559631)

[2.6 Các APIs cơ bản của WebRTC 13](#_Toc33559632)

[2.7 Các bước để tạo một kết nối P2P với WebRTC 13](#_Toc33559633)

[2.8 Signaling (báo hiệu) 14](#_Toc33559634)

[2.9 Cơ chế hoạt động của WebRTC và vai trò của Signling Server 15](#_Toc33559635)

[2.10 Ứng dụng của WebRTC 16](#_Toc33559636)

[CHƯƠNG 3. XÂY DỰNG APP GỌI ĐIỆN TRỰC TUYẾN SỬ DỤNG CÔNG NGHỆ WEBRTC 18](#_Toc33559637)

[3.1 Đặt vấn đề 18](#_Toc33559638)

[3.2 Giới thiệu về Flutter và Ngôn ngữ Dart 18](#_Toc33559639)

[3.2.1 Ngôn ngữ Dart 18](#_Toc33559640)

[3.2.2 Flutter 20](#_Toc33559641)

[3.3 Giới thiệu về Websocket và ngôn ngữ Golang 22](#_Toc33559642)

[3.3.1 Websocket 22](#_Toc33559643)

[3.3.2 Ngôn ngữ lập trình Golang 24](#_Toc33559644)

[3.4 Phân tích và thiết kế hệ thống 25](#_Toc33559645)

[3.4.1 Đặc tả yêu cầu hệ thống 25](#_Toc33559646)

[3.4.2 Biểu đồ usecase 26](#_Toc33559647)

[3.4.3 Mô hình hóa dữ liệu của hệ thống 27](#_Toc33559648)

[3.4.4 Các kịch bản của Signaling Server 28](#_Toc33559649)

[3.4.5 Mockups giao diện 31](#_Toc33559650)

[KẾT LUẬN 34](#_Toc33559651)

[**Nội dụng đã thực hiện** 34](#_Toc33559652)

[Các **hạn chế cần khắc phục** 34](#_Toc33559652)

[**Hướng phát triển** 34](#_Toc33559653)

[Tài liệu tham khảo 36](#_Toc33559654)

# GIỚI THIỆU ĐỀ TÀI

## Tên đề tài

*“Nghiên cứu công nghệ WebRTC và áp dụng trong app gọi điện trực tuyến”*

## Lý do chọn đề tài

Trong thời đại công nghệ thông tin bùng nổ, việc liên lạc qua môi trường internet ngày càng trở thành xu hướng của thế giới, giúp kết nối mọi người với nhau mà chi phí lại rẻ. Qua thời gian tìm hiểu về liên lạc qua internet, em thấy công nghệ WebRTC đang là chuẩn liên lạc rất được nhiều nhà phát triển phần mềm ứng dụng vào sản phẩm của họ. Do đó em quyết định chọn đề tài *Nghiên cứu công nghệ WebRTC và áp dụng trong app gọi điện trực tuyến* làm đề tài thực tập tốt nghiệp.

## Mục đích chọn đề tài

* Tìm hiểu WebRTC - cơ chế truyền tin qua mạng ngang hàng (P2P) mới, hiện đại, đơn giản và dễ dàng tích hợp vào ứng dụng.
* Hiểu rõ được cơ chế hoạt động của WebRTC cũng những lợi ích mà WebRTC mang lại.
* Ứng dụng công nghệ WebRTC vào một dự án thực tế để hiểu rõ hơn hoạt động của nó.

# TÌM HIỂU VỀ WEBRTC

## Giới thiệu công nghệ WebRTC

WebRTC (Web Real-Time Communications) là một tập hợp bao gồm các hàm dành cho lập trình dùng cho việc liên lạc thời gian thực bằng âm thanh, video hoặc 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 hoặc cài thêm plugin gì phức tạp. Chúng ta có thể dùng để phát triển trò chơi chơi trực tiếp trong trình duyệt và rất nhiều loại ứng dụng khác.

WebRTC không chỉ là một sản phẩm mà nó là một tập hợp rất nhiều các hàm có thể được lập trình viên sử dụng cho nhiều mục đích khác nhau. Chúng có một số hàm đơn giản như yêu cầu quyền truy cập camera, microphone của máy tính, có hàm phức tạp hơn thì để thiết lập kết nối giữa hai người dùng với nhau, cũng có hàm dùng để chia sẻ màn hình với người khác. Và hàm được sử dụng nhiều nhát của WebRTC là hàm giúp cho hai người có thể gọi điện hoặc gọi video cho nhau trong thời gian thực.

WebRTC hỗ trợ hầu hết các nền các trình duyệt và nền tảng hiện nay.

## Lịch sử phát triển

Hơn một thập niên trước, việc truyền tải âm thanh, video trên web, các lập trình viên thường phải dùng đến công nghệ Flash của Adobe, nhưng vào năm 2009, nhóm kỹ sư của Google lại không muốn sử dụng công nghệ này vào sản phẩm Google Hangouts vì họ cho rằng nó kém bảo mật và không hiệu quả. Do đó các kỹ sư của Google đã bắt đầu phát triển chuẩn riêng của họ.

Đến giữa năm 2011, Google công bố một dự án nguồn mở mang tên WebRTC dành cho việc kết nối và giao tiếp thời gian thực giữa các trình duyệt với nhau và cũng trong thời gian này W3C và hiệp hội kỹ sư quốc tế (IETF) đang xây đựng một số giao thức dùng cho việc kết nối thời gian thực, do đó họ bắt tay với nhau và gộp dự án đó lại vào WebRTC.

## Những lợi ích của WebRTC

Trong những năm của thập niên trước, khi muốn xây dụng một ứng dụng web đa nền tảng, đa phương tiện những nhà phát triền thường phải sử dụng Flash hoặc Java Applet và tích hợp plugin của bên thứ ba. Việc xây dựng một ứng dụng như vậy thường rất cồng kềnh, khó triển khai rộng và hỗ trợ về sau vì phải phụ thuộc vào nhiều bên. Nhận thấy vấn đề đó, người ta luôn nghiên cứu và time kiếm giả pháp thay thế đơn giản và hiệu quả hơn trong việc phát triển ứng dụng đa phương tiện. Do đó WebRTC được ra đời để giải quyết các vấn đề này. Vậy khi sử dụng WebRTC, chúng ta sẽ được những lợi ích gì?

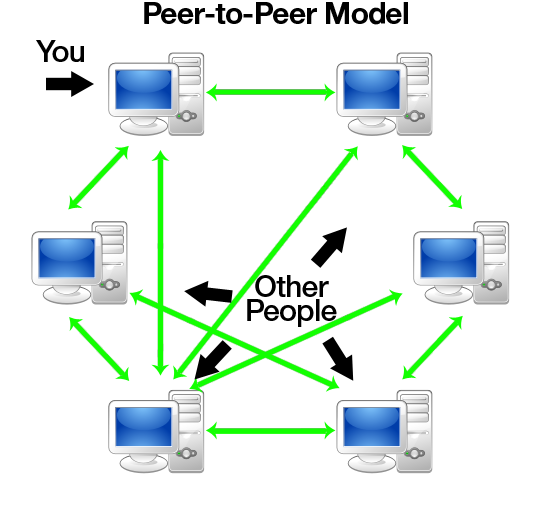
* Không plugin: Với WebRTC để xây dựng được ứng dụng Web đa phương tiện, chúng ta không cần cài thêm bất cứ plugin hay phần mềm thứ ba nào.
* Giảm giá thành: Chi phí phát triển phần mềm thấp hơn do đơn giản và không cần tích hợp plugin hay phần mềm thứ ba nào.
* Dễ sử dụng: Có thể tích hợp dễ dàng tính năng của WebRTC với các APIs được cung cấp sẵn
* Đa nền tảng: Trước đây, WebRTC chỉ hoạt động dược trên trình duyệt, tuy nhiên qua thời gian, WebRTC đã có thể hoạt động được trên hầu hết các nền tảng như Web, Android, iOS, Windows, macOS, linux…
* Truyền thông P2P (peer-to-peer): trong đa số trường hợp WebRTC thiết lập truyền thông trực tiếp giữa 2 client mà không cần thông qua bất cứ một máy chủ trung gian nào.
* Nguồn mở và miễn phí: Tất cả mọi người đều có thể sử dụng và đóng góp do dó mọi lỗi đều được cộng đồng phát hiện và vá lỗi nhanh chóng, hoàn toàn miễn phí.
* Bảo mật: Mọi dữ liệu được truyền đi đều được WebRTC mã hóa và bảo mật thông tin.

## Các khái niệm liên quan đến WebRTC

WebRTC là một chuẩn kết nối giúp hai client trao đổi dữ liệu trực tiếp với nhau theo mô hình mạng ngang hàng Peer-to-Peer (P2P). Do đó, để có thể bắt đầu trao đổi dữ liệu được với nhau thì các clients tham gia trao đổi dữ liệu bắt buộc phải tạo được kết nối Peer-to-Peer trước. Vậy mạng ngang hàng (P2P) là gì ?

### Mạng ngang hàng (P2P)

Mạng ngang hàng hoặc peer-to-peer viết tắt P2P là một mô hình mạng đồng đẳng, trong đó một mạng máy tính mà hoạt động của nó dựa vào khả năng tính toán và băng thông của tất cả các máy tham gia chứ không tập trung vào các máy chủ trung tâm như thông thường.



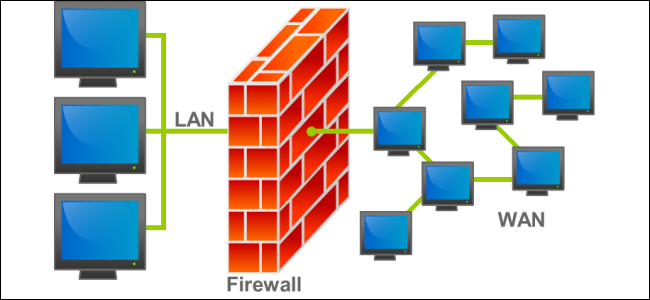
Hình 2.1 - Mô hình mạng máy tính P2P

Chính đặc tính dó nên trong mạng ngang hàng P2P không có sự khác nhau giữa máy chủ và máy khách, tất cả các máy đều bình đẳng như nhau, mỗi máy là một nút mạng đóng vai trò vừa là máy chủ vừa là máy khách đối với máy khác trong mạng lưới.

Chính vì WebRTC sử dụng kết nối peer-to-peer nên chúng ta mới có được sự tiện lợi như đã nêu ở mục 1.2. Kết nối P2P được thiết lập dễ dàng nếu các máy giao tiếp là cùng một mạng. Tuy nhiên vấn đề sẽ trở nên phức tạp hơn khi chúng ta muốn tạo kết nối này giữa hai mạng khách nhau, điển hình nhất là trong mạng internet do nó bị cản trở bởi tường lửa (Firewall) và NAT (Network Address Translation), để khắc phục được vấn đề này ta có thể sử dụng STUN (Simple Traversal Of UDP Through NAT) hoặc TURN (Traversal Using NAT Relay).

### Tường lửa (Firewall)

Tường lửa là một biện pháp bảo vệ an toàn cho máy tính, là một bức rào chắn giữa mạng cục bộ (local network) với một mạng khác (ví dụ internet).



Hình 2.2 - Tường lửa (firewall)

Một tường lửa có thể kiểm soát và lọc được các truy cập nguy hiểm như hacker hay một số loại virus tấn công, ngăn không để chúng phá hoại hay làm tê liệt hệ thống cục bộ. Ngoài ra vì các nguồn truy cập ra vào giữa mạng nội bộ và mạng khác đều phải thông qua tường lửa nên tường lửa còn có tác dụng theo dõi, phân tích các luồng lưu lượng truy cập và quyết định sẽ làm gì với những luồng lưu lượng đáng ngờ như khoá lại một số nguồn dữ liệu không cho phép.

### NAT - Network Address Translation

NAT (Network Address Translation) là một kỹ thuật cho phép chuyển đổi từ một địa chỉ IP này thành một địa chỉ IP khác. NAT được dùng phổ biến trong mạng sử dụng địa chỉ cục bộ, cần truy cập đến mạng công cộng (Internet). Khi một mạng cục bộ truy cập đến mạng internet, ta cũng có thể hiểu nó là một mạng NAT.

Thông thường một máy tính sẽ có 2 loại địa chỉ IP là private IP và public IP. Các máy tính trong một mạng cục bộ sẽ được cấp một địa chỉ private IP, địa chỉ này chỉ được dùng cho các máy trao đổi dữ liệu trong mạng đó, không tồn tại trên môi trường internet. Private IP thường bắt đầu bằng 192.168.xxx.xxx. Ngược lại với private IP, public IP là địa chỉ tồn ngoài môi trường internet.

Trong một mạng NAT, khi các máy tham gia vào mạng internet, một máy chủ sẽ sử dụng IP của chính nó làm public IP cho mỗi máy con (client) với IP riêng. Khi một máy con thực hiện kết nối hoặc gửi dữ liệu tới một máy tính nào đó trên internet, dữ liệu sẽ được gửi tới NAT, sau đó NAT sẽ thay thế địa chỉ IP gốc của máy con đó rồi gửi gói dữ liệu đi với địa chỉ IP của NAT. Máy tính từ xa hoặc máy tính nào đó trên internet khi nhận được tín hiệu sẽ gởi gói tin trở về cho máy chủ NAT bởi vì chúng nghĩ rằng máy chủ NAT là máy đã gởi những gói dữ liệu đi. NAT ghi lại bảng thông tin của những máy tính đã gởi những gói tin đi ra ngoài trên mỗi cổng dịch vụ và gửi những gói tin nhận được về đúng máy tính đó.

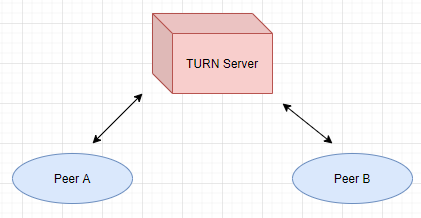
### STUN (Simple Traversal Of UDP Through NAT)

Hiểu đơn giản thì STUN server sẽ giúp cho client biết được địa chỉ IP và Port mà thiết bị NAT đang sử dụng. Và từ đó giúp cho các máy (peer) có thể lấy được địa chỉ của peer khác bao gồm địa chỉ IP, cổng giao tiếp, và thông tin về NAT và loại NAT đang sử dụng (NAT có nhiều loại) để giao tiếp và trao đổi dữ liệu với nhau.

Ưu điểm của STUN server là miễn phí và được công khai rộng rãi trên mạng. Tuy nhiên STUN có một nhược điểm là nó không hỗ trợ cho Symmetric NAT. Do đó ta không thể giao tiếp P2P với nhau nếu một trong mạng mà Peer đang sử dụng là Symmetric NAT. Giải pháp thay thế được đưa ra trong trường hợp này là sử dụng TURN server.

### TURN (Traversal Using NAT Relay)

TURN cũng giống như STUN tuy nhiên TURN hỗi trợ thêm 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.



Hình 2.3 – Cơ chế hoạt động của TURN

Nhược điểm của TURN là chi phí sử dụng lớn do sẽ có một lưu lượng băng thông lớn được sử dụng. Nhất là khi sử sụng cho gọi điện trực tuyến.

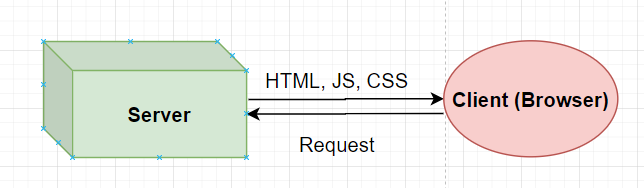
### ICE (Interactive Communication Establishment)

ICE là một giao thức được cùng để thiết lập phiên media dựa trên UDP đi qua NAT một cách nhanh nhất. Giao thức này sẽ giúp 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.

Đầu tiên nó 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 (do đang sủ dụng mạng NAT) thì ICE sẽ lấy địa chỉ bên ngoài 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 server.

## Kiến trúc của WebRTC

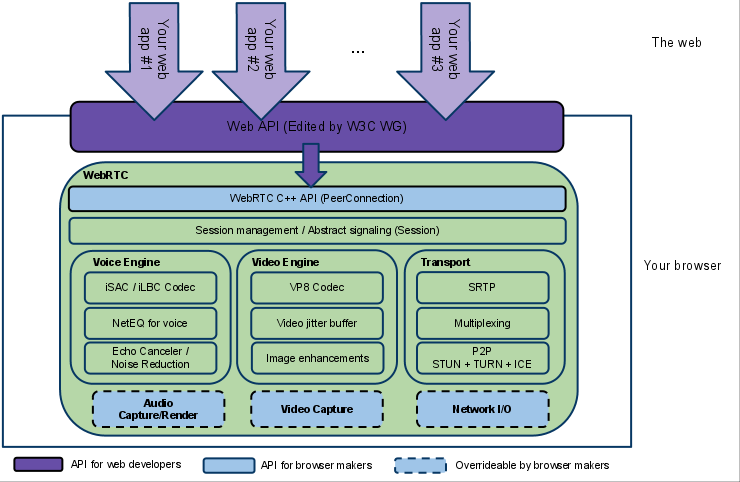
Kiến trúc web bình thông thường dựa theo mô hình client-server, trong đó trình duyệt gửi yêu cầu HTTP để lấy nội dung, máy chủ trả về nội dung cho trình duyệt dạng HTML kèm theo cả các đoạn mã JavaScript và CSS. Trong cac trường hợp phức tạp thì trình duyệt gửi XHR request (AJAX) đến máy chủ và tương tác với máy chủ qua các APIs, lúc này trình duyệt có thể trao đổi với máy bằng giao thức HTTP trên TCP hoặc WebSocket trên TCP.



Hình 2.4 – Kiến trúc của Web cổ điển

Tuy nhiên, với WebRTC, định nghĩa mô hình client-server được mở rộng bởi truyền thông peer-to-peer giữa các trình duyệt, thêm máy chủ *signaling* (báo hiệu) và thành phần chức năng truyền thông thời gian thực (Real Time Cummunication, viết tắt là RTC) cuả trình duyệt. Úng dụng WebRTC tương tác với trình duyệt qua những WebRTC APIs đang được chuẩn hóa, cho phép nó điều khiển và khai thác hợp lý chức năng thời gian thực của trình duyệt. Ứng dụng WebRTC có thể tương tác với các APIs chuẩn hóa một cách chủ động (truy vấn khả năng của trình duyệt) hoặc bị động (tiếp nhận thông báo khởi tạo bời trình duyệt). Dó đó WebRTC phài cung cấp một tập các chức năng quản lý kết nối, chức năng điều khiển media, hỗ trợ vượt qua NATs, tường lửa, encoding/decoding…

Hiện nay, kiến trúc của WebRTC có nhiều chuẩn khác nhau, bao gồm cả úng dụng và các APIs trình duyệt. Trong WebRTC, trình duyệt có khả năng truy cập vào phần cứng hệ thống để lấy video và audio qua các APIs. Những video, audio này được xử lý để gia tăng chất lượng, đồng bộ, ngoài ra ’bitrate’ được điều chỉnh theo sự tăng giảm của băng thông và độ trễ giữa các client. Ở phía còn lại, quá trình giải mã sẽ được thực hiện ngược lại, client phải thực hiện giải mã media theo thời gian thực.



Hình 2.5 – Kiến trúc của WebRTC

WebRTC có 3 lớp APIs, bao gồm :

* APIs cho lập trình viên : Chứa các APIs mà lập trinh viên cần, gồm các đối tượng chính là WebRTCDataChannel, MediaStream
* APIs cho nhà phát triển sử dụng
* Overridable APIs: các APIs mà nhà phát triển trình duyệt có thể thay đổi

Ở hình 2.5 thành phần *Video Engine* là framework xử lý video từ camera đến mạng và từ mạng ra màn hình, nó sử dụng video codec là VP8 – một dạng nén video mờ, miễn phí của Google và VP9, hỗ trợ tính nâng cao chất lượng video tổng thể, nâng cao chất lượng hình ảnh bằng cách khử nhiễu được lấy từ camera.

Thành phần *Audio Engine* cũng là framework xử lý audio từ card âm thanh, nó sử dụng codec iSAC (internet Speech Audio Codec), iLBC (internet Low Birate Codec) và Opus, thành phần này cũng có chức năng xử lý tiếng ồn trong audio giúp cải thiện âm thanh.

Ngoài ra, thành phần vận chuyển/quản lý phiên (session) rất quan trong, cho phép thiết lập và quản lý kết nối P2P qua các loại mạng khác nhau.

## Các APIs cơ bản của WebRTC

WebRTC bao gồm các giao thức, APIs liên quan và làm việc với nhau để hổ trợ trao đổi dữ liệu giữa các trình duyệt. Ở mức độ cơ bản thì ứng dụng WebRTC thực hiện các công việc chính bao gồm:

* Lấy dữ liệu video, audio hoặc dữ liệu khác trên máy.
* Lấy thông tin mạng bao gồm IP, port,.. vào trao đổi thông tin này với client (peer) còn lại để bắt đầu thiết lập kết nối kể cả qua Firewall hay NATs
* Khởi tạo hoặc đóng phiên kết nối
* Trao đổi về khả năng hỗ trợ media của từng peers như độ phân giải, codecs,..
* Streaming video, audio hoặc data giữa hai peers

Để thực hiện được các công việc trên, WebRTC được chuẩn hóa và sử dụng các APIs quanh các khái niệm chính:

* RTCPeerConnection: Thiết lập kết nối cho cuộc gọi video, audio hoặc data
* MediaStream: Truy cập vào media như camera hay microphone của máy
* DataChannel: Giao tiếp peer-to-peer cho các dử liệu không phải media

## Các bước để tạo một kết nối P2P với WebRTC

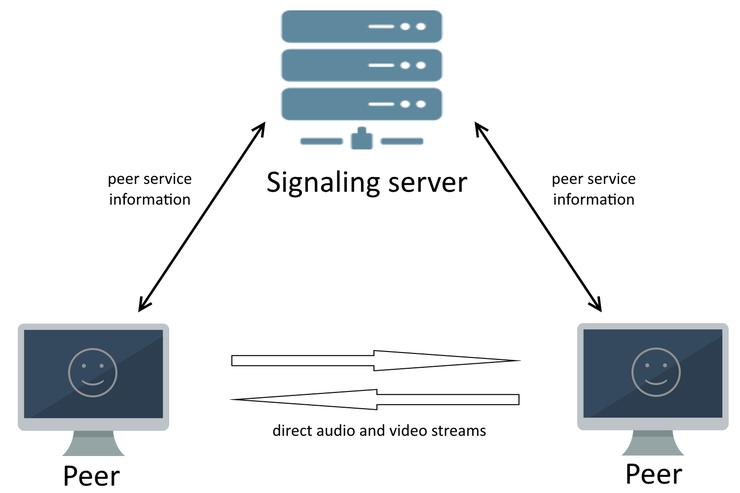
Dựa vào các APIs mà WebRTC cung cấp ta có thể thực hiện một kết nối P2P theo các bước dưới đây:

* *Bước 1:* Sử dụng API getUserMedia để truy cập vào camera và microphone
* *Bước 2:* Lấy thông tin mạng đang sử dụng như địa chỉ IP, ports và trao đổi thông tin đó với peer mà mình muốn kết nối tới để tạo connection dù cho có bị ngăn cản bởi NATs hay Firewalls
* *Bước 3:* Sau đó thì dùng RTCPeerConnection và RTCDataChannel để voice call/ video call hoặc chia sẻ dữ liệu sau khi đã có kết nối peer-to-peer.

Trong WebRTC, mỗi một MediaStream đại diện cho sự đồng bộ dữ liệu âm thanh và hình ảnh. Nó được khởi tạo bằng cách gọi hàm getUserMedia. Sau khi một kết nối WebRTC tới một máy tính khác được thiết lập, chúng ta có khả năng truy cập vào stream của máy tính đó. Mỗi peer sẽ có một local media stream riêng. Một MediaStream sẽ có input/output. Input là sẽ lấy những dữ liệu âm thanh và hình ảnh của local và output dùng để hiển thị lên giao diện hoặc được RTCPeerConnection sử dụng.

## Signaling (báo hiệu)

WebRTC sử dụng RTCPeerConnection để truyền dữ liệu giao tiếp giữa hai peers với nhau. Nhưng trước đó cả hai đều chưa có thông tin gì về nhau, không biết địa chỉ của nhau nên không thể giao tiếp và truyền dữ liệu qua lại với nhau. Việc này tương tự như khi ta muốn gửi thư liên lạc cho một ai đó mà không biết được địa chỉ của người kia vậy. Signaling sẽ giúp ta giải quyết vấn đề đó và thường được xây đựng bằng Socket.



Hình 2.5 – Mô hình Sinaling server

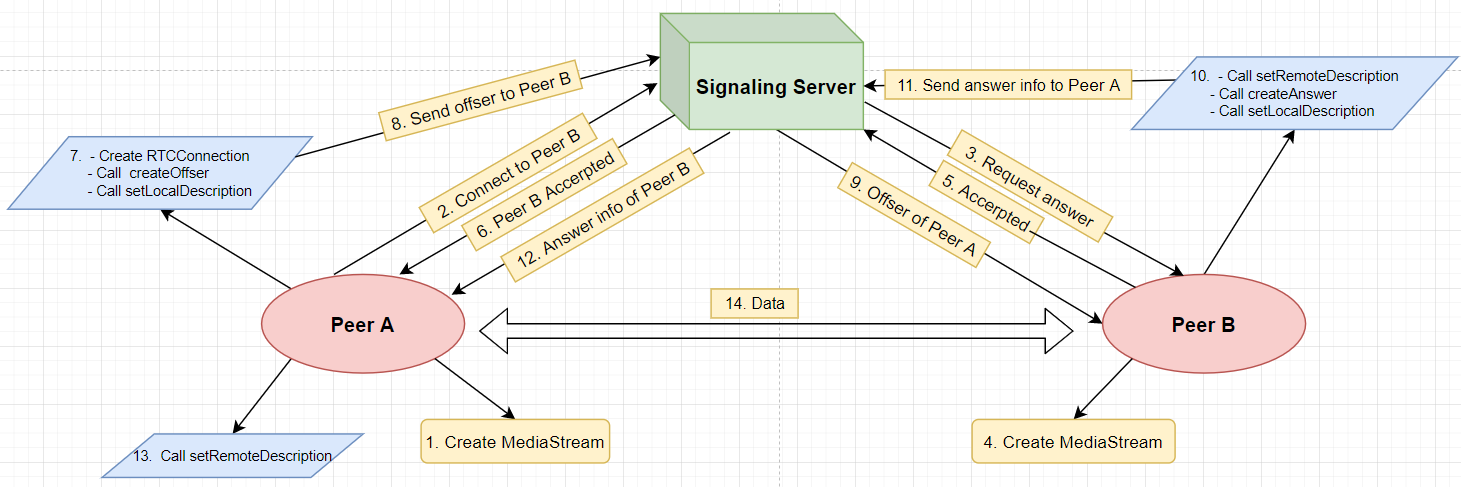
Vậy Signaling dùng để làm gì?

* Session control messages: dùng để khởi tạo hoặc chấm dứt một kết nối
* Network configuration: dùng để lấy thông tin địa chỉ public IP, port,… của peer kết nối từ STUN hoặc STURN server
* Media capabilities: các loại codecs và resolutions cần thiết cho việc voice call/video call

Để có thể bắt đầu gửi và nhận data qua WebRTC giữa hai peers thì bắt buộc phải thực hiện thành công những phần trên.

## Cơ chế hoạt động của WebRTC và vai trò của Signling Server

Cơ chế hoạt động của WebRTC và vai trò của Signling Server được thể hiện trong hình vẽ dưới đây.



Hình 2.6 – Vai trò của Signaling server và cơ chế hoạt động của WebRTC

Giả sử peer A muốn thực hiện một cuộc gọi qua WebRTC tới peer B, ta cần phải thực hiện các bước đưới đây:

* Bước 1: *Peer A* khởi tạo MediaStream.
* Bước 2: *Peer A* gửi yêu cầu kết nối tới *peer B* qua *Signaling server.*
* Bước 3: *Signling Server* gửi đến *peer B* message rằng *“peer A đang cần kết nối, có muốn kết nối không?”.*
* Bước 4: Nếu *peer B* chấp nhận yêu cầu kết nối của *peer A*, *peer B* sẽ tiến hành khởi tạo MediaSctream tương tự như *bước 1* ở *peer A.*
* Bước 5: *Peer B* gửi message đến *Signaling server* rằng *“tôi chấp nhận yêu cầu kết nối từ* *peer A”.*
* Bước 6: *Signaling server* gửi message đến peer A *“peer B đã chấp chận yêu cầu kết nối”.*
* Bước 7: Sau khi *peer A* biết rằng *peer B* đã sẵn sàng kết nối, *peer A* tiến hành khởi tạo một phiên kết nối *RTCConnection*, khởi tạo offer bằng cách gọi hàm *createOffer* và lưu nó lại bằng hàm *setLocalDescription*.
* Bước 8: *Peer A* gửi thông tin offer vừa khởi cho *Signaling server* và yêu cầu *Signaling server* gửi nó cho *peer B.*
* Bước 9: *Signaling server* gửi offer mới nhận được từ *peer A* cho *peer B*
* Bước 10: *Peer B* khởi tạo một *RTCConnection*, lưu lại thông tin của peer A bằng hàm *setRemoteDescription*, tiếp theo peer B tạo thông tin trả lời bằng *createAnswer* và lưu nó lại với hàm *setLocalDescription.*
* Bước 11: *Peer B* gửi thông tin trả lời cho *Signaling server* và yêu cầu *Signaling server* gửi nó cho *peer A.*
* Bước 12: *Signaling server* gửi thông tin trả lời nhận được từ *peer B* cho *peer A*
* Bước 13: Peer lưu lại và xử lý thông tin trả lời nhận được bằng hàm setRemoteDescription
* Bước 14: Lúc này cả Peer A và peer B đã có được thông tin liên lạc của nhau. Cả hai tiến hành gửi data (candidates) cho nhau với thông tin liên lạ đang có. Nếu cả hai peers nhận được candidates thì có nghĩa 2 peer đã được kết nối thành công.

Sau khi thực hiện xong các bước trên, 2 peers có đã có thể trao đổi trực tiếp với nhau. Nếu một peer muốn chấm dứt kết nối thì sẽ, hủy session connection hiện tại, ngắt *MediaStream* và gửi một message ngắt kết nối cho peer còn lại thông qua *Signaling Server,* sau khi nhận được message ngắt kết nối, peer còn lại cũng sẽ thực hiện hủy *connection* và *MediaStream.*

## Ứng dụng của WebRTC

Ta có thể sử dụng WebRTC trong rất nhiều ứng dụng khác nhau để mang tới một cách kết nối đơn giản tiện lợi. WebRTC có thể tạo ra các webphone, ứng dụng hội nghị trực tuyến, chăm sóc khách hàng, hướng dẫn trực tuyến, giáo dục, chăm sóc sức khỏe, tư vấn bảo hiểm, game,…

Ngày nay, hầu hết các công ty đều có Website cung cấp các thông tin và thường là điểm đầu tiên giúp khách hàng liên hệ với doanh nghiệp. Nếu khách hàng không tìm thấy đủ thông tin trên Website hoặc cần phản ánh một vấn đề liên quan đến sản phẩm thì họ chỉ có cách gọi điện thoại. Tuy nhiên, cách này có thể làm mất các ngữ cảnh trên website hay không mô tả rõ ràng được vấn đề. WebRTC có thể ứng dụng để giúp thực hiện cuộc gọi ngay trên Website và cho phép chia sẻ màn hình của khách hàng, từ đó các nhân viên hỗ trợ có thể nhanh chóng nhận ra và giải quyết vấn đề của khách hàng.

Một số app phổ biến hiện nay như Facebook Messager, Skype, Telegram,… cũng ứng dụng cho WebRTC cho phần gọi điện. Nó cho thấy tiềm năng to lớn mà WebRTC mang lại.

# XÂY DỰNG APP GỌI ĐIỆN TRỰC TUYẾN SỬ DỤNG CÔNG NGHỆ WEBRTC

## Đặt vấn đề

Ở chương này, chúng ta cần làm rõ làm rõ được cơ chế hoạt động và ứng dụng của WebRTC như thế nào trong một phần mềm thực tế. Do dó, em lựa chọn việc xây dựng một app di dộng gọi điện trực tuyến để hiểu rõ hơn về cơ chế hoạt động của WebRTC cũng như vai trò của Signaling server.

Để đơn giản trong quá trình xây dựng app, em sử dụng Flutter Framework mới của Google cho việc xây dựng app. Sử dụng ngôn ngữ Golang và giao thức WebSocket để xây dựng Signaling server.

## Giới thiệu về Flutter và Ngôn ngữ Dart

### Ngôn ngữ Dart

* *Giới thiệu về ngôn ngữ Dart*

Dart là ngôn ngữ lập trình mới, được phát triển bởi Google và ra mắt ở Hội thảo GOTO ngày 10-12 Tháng Mười năm 2011 tại Aarhus. Hiện Dart đã được chấp thuận bởi tổ chức Ecma. Dart có thể được sử dụng để xây dựng web, server và các ứng dụng di động (IOS và Android với công cụ Flutter).



Hình 3.1 – Ngôn ngữ lập trình Dart

Tương tự như Java, Dart là ngôn ngữ trình thuần hướng đối tượng, cú pháp kiểu C và ngắn gọn hơn Java, mã code Dart cũng có có thể biên dịch thành JavaSript để chạy trên trình duyệt. Dart là ngôn ngữ nguồn mở, miễn phí, nó hỗ trợ tất cả các kỹ thuật lập trình hướng đối tượng hiện tại như interface, mixin, abstract, generic, static typing và sound type… Phiên bản mới nhất tính đến thời điểm hiện tại của Dart là 2.7 với rất nhiều cả tiến, trong đó được chú ý nhất là từ phiên bản 2.6, Dart ra mắt công cụ dart2native cho phép chạy các đoạn mã mà không cần cài đặt Dart SDK.

Vào năm 2017, Google cho ra mắt SDK Futter sử dụng ngôn ngữ Dart phát triển ứng dụng di động chạy đa nền tảng (Một lần viết code build ra ứng dụng cho iSO, Android, Linux, Windows và cả macOS).

* *Đặc điểm của ngôn ngữ Dart*

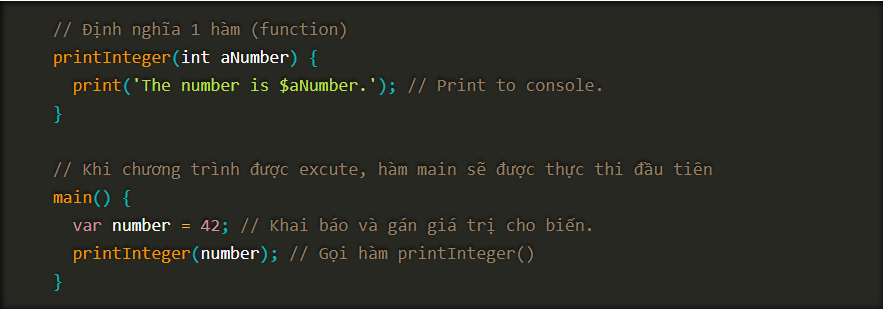
Ngay từ khi cho ra mắt, Google định hướng cho Dart là một ngôn ngữ cải tiến hơn Javascript do đó nó hỗ trợ cả 2 cách biên dịch phổ biến là Just In Time (JIT) và Ahead Of Time (AOT) với đặc điểm của 2 cách biên dịch trên là:

* Just In Time (JIT): Cho phép *hot reloading* hoạt động, giúp phát triển sản phẩm tiện dụng và nhanh hơn. JIT giúp ít rất nhiều trong việc debug. Với các ngôn ngữ khác, ví dụ như với Java, hoặc C++ để debug hoặc có xem các thay đổi thì ta phải biên dịch xong chạy file mới biên dịch. Như với cơ chế biên dịch JIT của Dart ta có thể xem luôn thay đổi của chương trình mà không cần thực hiện biên dịch toàn bộ chương trình.
* Ahead Of Time (AOT): Với AOT thì trình biên dịch chuyển ngôn ngữ Dart thẳng sang Native Code giúp hiệu năng tốt nhất có thể khi chạy trong môi trường production (khi chạy chương trình, Dart sẽ biên dịch từ đầu đến cuối tương tự C/C++, Java, Go…)

Những đặc điểm của Dart có thể kể đến như sau:

* Dart là một ngôn ngữ thuần hướng đối tượng: Trong Dart mọi thứ đều là đối tượng, bao gồm cả số, chuỗi, null
* Dart gần giống Java, được cải tiến lên từ Javascript, cũng là hệ C nên cú pháp tương tự nhau
* Kiểu dữ liệu của một biến là không nhất thiết phải khai báo bởi vì Dart có thể tự suy ra kiểu dựa trên giá trị truyền vào cho biến (tương tự JavaScript)
* Từ khóa **dynamic** dùng để chỉ định một biến được khai báo khi chưa xác định được kiểu dữ liệu.
* Dart hỗ trợ **top-level function** (giống như hàm main()), đồng nghĩa chúng ta có thể sử dụng hàm đó ở bất cứ đâu mà không cần thông qua tên **class** hay bất kì **instance** của class nào cả
* Như các ngôn ngữ khác, **Dart** hỗ trợ **generic type**, như **List** nhưng Dart có thể chấp nhận mọi kiểu dữ liệu.
* Dart cho phép tạo hàm bên trong hàm (còn gọi là **nested** **function** hoặc **local function**) và tương tự Dart cũng có **top-level variable**
* Khác với với **Java** và **C++**, **Dart** không hỗ trợ **public**, **protected** và **private**. Nếu như  tên biến, hàm,... bắt đầu với dấu gạch dưới ( **\_** ), thì Dart sẽ hiểu nó là **private**.
* Với công cụ **Dart tools** có thể báo cho chúng ta loại vấn đề bao gồm **warning** và **errors**.

Một chương trình Dart cơ bản:



Hình 3.2 – Một chương trình cơ bản viết bằng ngôn ngữ Dart

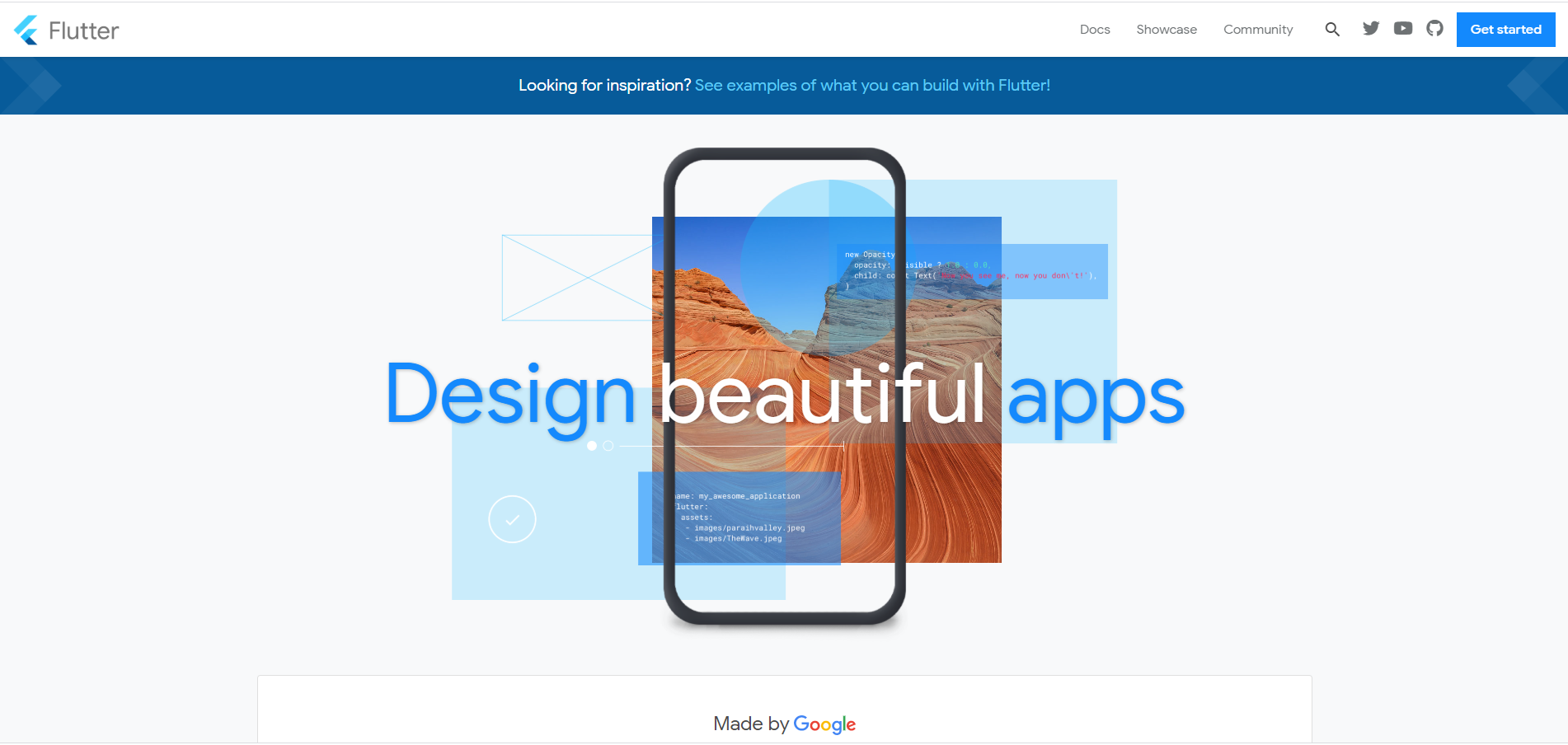
### Flutter

* *Giới thiệu*

Flutter là một công cụ do Google xây dựng cho phép các nhà phát triển xây dựng các ứng dụng đa nền tảng có thể được thực hiện trong các hệ thống khác nhau như Android, iOS, Website thậm chí cả ứng dụng trên máy tính bao gồm cả macOS, Linux và Windows chỉ với một bộ source code chung.



Flutter được xây dựng trong C và C ++, nó cung cấp một cơ chế rendering 2D, một funtional-reactive framework là React-inspired, và một tập hợp các “Material Design” widget cho Android và “Cupertino Design” cho iOS, điều này cho phép các nhà phát triền có thể tạo ra các ứng dụng với giao diện và hiệu xuất tương tương với ứng dụng native trên nền tảng đó.



Hình 3.3 – Trang chủ của Flutter - flutter.dev

Không giống như các công cụ phát triển ứng dụng đa nền tảng khác như React-Native, VueNative,… sử dụng JavaScript, Flutter lại sử dụng Dart làm ngôn ngữ phát triển.

* *Cơ chế reandering của Flutter*

Ứng dụng sẽ không thực thi mã Dart trực tiếp. Theo thời gian, một ứng dụng được release xây dựng, mã sẽ được biên dịch để local nhận được hiệu suất như một kết quả tốt hơn và đáp ứng giao diện người dùng tốt hơn. Trong khi phát triển trong chế độ debug (kiểm tra đối với các lỗi tiềm năng) Flutter cũng thực hiện một số nhiệm vụ mà có thể làm cho các ứng dụng chạy chậm hơn. Nếu trải qua tình trạng này, Flutter sẽ cho biết cách đặt một dải ruy băng màu đỏ đầu ngay trong màn hình với dòng chữ "Slow Mode" ở trên đó.

* *Lợi ích khi sử dụng Flutter để phát triển ứng dụng:*
* Phát triển ứng dụng nhanh chóng với Hot Reload: Tính năng hot reload của Flutter giúp nhanh chóng và dễ dàng thử nghiệm, thêm tính năng và sửa lỗi nhanh và xây dựng giao diện người dùng nhanh hơn. Giúp tải lại lần thứ hai, mà không làm mất trạng thái, trên emulator, simulator và device cho iOS và Android.
* UI đẹp: Flutter cung cấp các widget built-in đẹp theo Material Design và Cupertino (iOS-flavor). Các API chuyển động phong phú
* Framework hiện đại: Flutter giúp chúng ta dễ dàng tạo giao diện người dùng, tập hợp các platform, layout và widget phong phú
* Truy cập các tính năng và SDK native: Flutter cho phép sử dụng lại mã Java, Swift và Objective C hiện tại và truy cập các tính năng và SDK native trên iOS và Android.

## Giới thiệu về Websocket và ngôn ngữ Golang

### Websocket

Hiện nay với sự phát triển từng ngày của công nghệ, các ứng dụng web đã khác xa so với ngày đầu nó xuất hiện, cùng với đó là vô số các kỹ thuật mới được áp dụng nhằm đem lại trải nghiệm mới mẻ và phần tiện dụng cho người dùng. Công nghệ thời gian thực (realtime) ngày càng trở nên phổ biến.

Chúng ta có thể xây dựng một ứng dựng thời gian thực với nhiều công nghệ như:

* Ajax Long-Polling:
* Server Sent Events (SSE)
* Comet
* Websocket
* …

Trong đó WebSocket với sự hổ trợ của HTML5 là nổi bật nhất được đông đảo cộng đồng lập trình viên sử dụng.

* *Websocket là gì?*

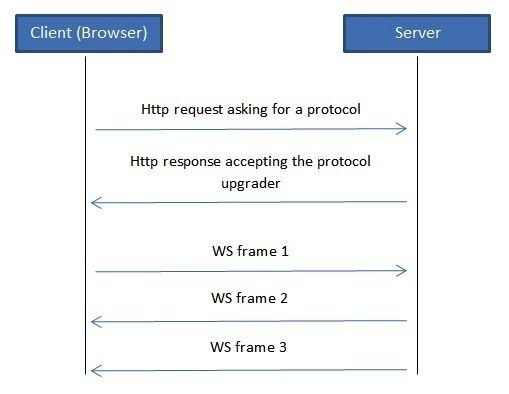
WebSocket là một giao thức giúp truyền dữ liệu hai chiều giữa server-client với một kết nối TCP duy nhất. Nó được thiết kế để truyền dữ liệu bằng cách sử dụng cổng 80 và cổng 443 và Websocket cũng có thể hiểu là một phần của HTML5.Do đó, WebSocket có thể hoạt động trên các cổng web tiêu chuẩn, nên không có rắc rối về việc mở cổng cho các ứng dụng, lo lắng về việc bị chặn bởi các tường lửa hay proxy server.

Khi sử dụng giao thức HTTP client cần chủ động gửi yêu cầu (request) cho server, sau khi server xử lý thì sẽ trả lại phản hồi (response) cho client. Tuy nhiên, với giao thức Websocket thì server có thể chủ động gửi thông tin đến client mà không cần phải có yêu cầu từ client. Tương tự client có thể gửi thông điệp đến server bất cứ lúc nào.

WebSocket có thể giúp giảm độ trễ bởi vì một khi kết nối WebSocket được thành lập, server không cần phải chờ đợi cho một yêu cầu từ client. Tất cả dữ liệu giao tiếp giữa client-server sẽ được gửi trực tiếp qua một kết nối cố định làm cho thông tin được gửi đi nhanh chóng và liên tục khi cần thiết.

Để sử dụng được Websocket thì không phải chỉ cần client hỗ trợ mà còn phải có server WebSocket. Server WebSocket có thể được tạo ra bằng bất kỳ ngôn ngữ server-side nào, ví dụ NodeJS, Golang,…

* *Cơ chế hoạt động của WebSocket*



Hình 3.4 – Cơ chế hoạt động của WebSocket

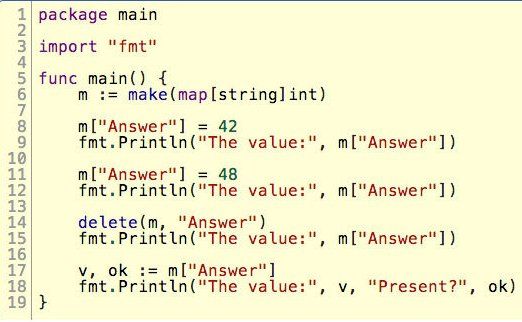
Giao thức WebSocket có hai phần gồm: Bắt tay và truyền dữ liệu. Ban đầu client sẽ gửi yêu cầu khởi tạo kết nối Websocket đến server, Server kiểm tra và gửi trả kết quả chấp nhận kết nối. Nếu kết nối thành công quá trình gửi dữ liệu có thể được thực hiện, dữ liệu chính là các WS frame ở trong hình trên.

### Ngôn ngữ lập trình Golang

Go hay còn được biết đến như Golang là một ngôn ngữ lập trình mã nguồn mở (open source), dạng biên dịch (compiled) kiểu tĩnh (statically type) được phát triển bởi Google. Ra mắt lần đầu vào năm 2017. Mục đích chính của Golang là hỗ trợ phát triển các phần mềm có tính sẵn sàng cao đồng thời giúp việc mở rộng nhanh và dễ dàng hơn.

Go cũng như các ngôn ngữ khác như Java, Python, Ruby, NodeJS… dùng để lập trình phía Server. Tuy nhiên Go lại có các ưu điểm như:

* Concurrency (cơ chế đồng thời) là một phần có sẵn của Go, do đó viết một chương trình đa luồng sẽ rất dễ dàng. Các tiến trình đa luồng được lưu trữ bởi goroutines và channels
* Golang là một ngôn ngữ biên dịch, mã nguồn sẽ được biên dịch sang mã nhị phân (binary). Giúp cho việc chạy chương trình nhanh hơn rất nhiều.
* Go vô cùng đơn giản, toàn bộ các đặc điểm có thể tóm gọn lại trong một vài trang giấy.
* Bộ biên dịch của Go hỗ trợ liên kết tĩnh (static linking), tất cả code có thể được kết nối tĩnh thông qua một đoạn mã nhị phân và có thể triển khai trên server đám mây dễ dàng mà không cần lo lắng đến các điều kiện phụ thuộc.
* Và đương nhiên Go cũng hỗ trợ đa nề tảng
* Tốc độ thực thi cao, linh hoạt và khả năng tương thích tốt.



Hình 3.5 – Một chương trình đơn giản viết bằng Golang

Do Go được thiết kế với tiêu chí là ngôn ngữ đơn giản, nhỏ gọn, và dễ học nên Go không có một số đặc tính quan trọng mà các ngôn ngữ khác có như ***generics*** - kiểu biểu diễn của ***Types*** dưới dạng tham số khi định nghĩa lớp, hàm và interfaces.

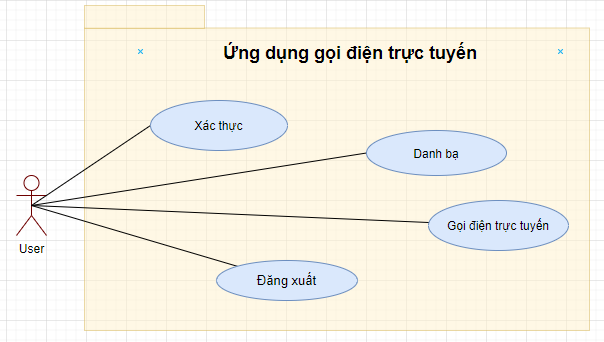
## Phân tích và thiết kế hệ thống

### Đặc tả yêu cầu hệ thống

Để làm rõ được sự hoạt động của WebRTC, ta cần phải xây dựng hệ thống Signaling Server và hệ thống API cho việc xác thực và sử dụng các chức năng cơ bản phục vụ cho việc thực hiện cuộc gọi như đã trình bày ở chương I. Dự vào các yêu cầu cơ bản đó, một hệ thống Server phục vụ cho *app gọi điện* *trực truyến* cần có các chức năng cơ bản sau:

* Xác thực: Đăng nhập tài khoản user
* Xem danh bạ
* Cuộc gọi: gọi/kết thúc cuộc gọi
* Đăng xuất

### Biểu đồ usecase



Hình 3.6 – Biểu đồ ca sử dụng của hệ thống

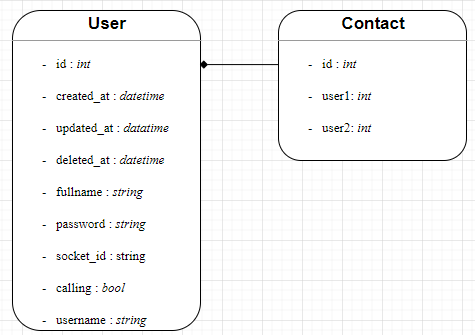
Bảng mô tả các usecases:

|  |  |  |
| --- | --- | --- |
| **#** | **Tên usecase** | **Mô tả** |
| 1 | Xác thực | Cho phép actors đăng nhập vào hệ thống để sử dụng các chức năng |
| 2 | Danh bạ | Cho phép actors quản lý danh bạn cá nhân |
| 3 | Gọi điện trực tuyến | Actors có thể gọi điện trực tuyến cho các users khác trên hệ thống |
| 4 | Đăng xuất | Thoát tải khoản hiện tại và quay về đăng nhập |

### Mô hình hóa dữ liệu của hệ thống

Từ yêu cầu đặc tả người dùng và các user case chính của hệ thống ta có mô hình hóa dữ liệu của hệ thống như sau:

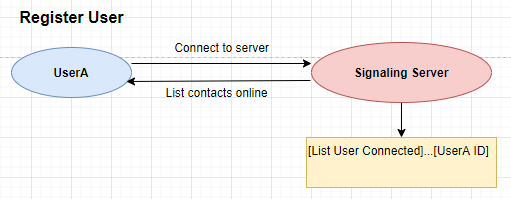
* Lớp ***User*** :
* Id : *int*
* created\_at : *datetime*  – thời gian tạo tài khoản
* updated\_at : *datatime* – thời gian cập nhật tài khoản
* deleted\_at : *datetime* – thời gian xóa tài khoản
* fullname : *string* – Tên hiển thị
* password : *string* – mật khẩu đã mã hóa
* socket\_id : string – mã phiên của WebSocket
* calling : *bool* – true nếu user đang có cuộc gọi khác
* username : *string* – tên đang nhập
* Lớp ***Contact :***
* Id : *int*
* User1 : *int*
* User2 : *int*



Hình 3.7 – Mối quan hệ giữa các lớp trong hệ thống

### Các kịch bản của Signaling Server

* *Đăng ký hàng chờ*



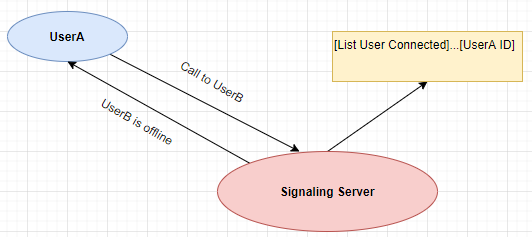
Hình 3.7 – Đăng ký hàng chờ khi user trực tuyến

Mỗi khi một user nào đó trực tuyến, app sẽ gửi một thông điệp đăng ký lên Server thông qua giao thức WebSocket, Server tiến hành lưu lại ID của user đó vào một danh sách đăc biệt (user online list), danh sách này thể hiện rằng các users trong danh sách đã được kết nối thành công tới hệ thống và đã sẵn sàng thực hiện hoặc nhận cuộc gọi. Đồng thời, Server cũng trả về một danh sách các liên hệ của user mới đăng nhập đang online cho client.

Trong trường hợp user ngoại tuyến (mất kết nối tới server), ID của user sẽ bị xóa khỏi danh sách.

Trong một ứng dụng gọi điện trực tuyến sử dụng WebRTC,ngoài các yếu tố khách quan mà một cuộc gọi qua WebRTC không thể được kết nối thành công như Firewall, NATs,… thì vẫn còn các yếu tố chủ quan mà người dùng đóng vai trò quyết định, dưới đây là các trường hợp mà một cuộc gọi không thể được thực hiện hoặc chấm dứt một cuộc gọi do nguyên nhân chủ quan.

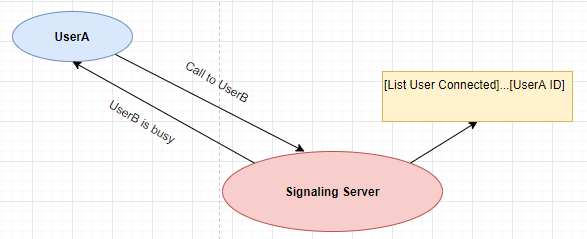
* *Thực hiện cuộc gọi đến một người dùng đang ngoại tuyến*



Hình 3.8 – Thực hiện cuộc gọi đến một user đang ngoại tuyến

UseA gửi thông điệp lên Signaling Server yêu cầu được thực hiện cuộc đến UserB, trong trường hợp UserB ngoại tuyến hoặc mất kết nối (ID userB không nằm trong danh sách các users đang sẵn sàng kết nối), Signaling server sẽ gửi lại thông điệp *UserB đang không trực tuyến* đến UserA.

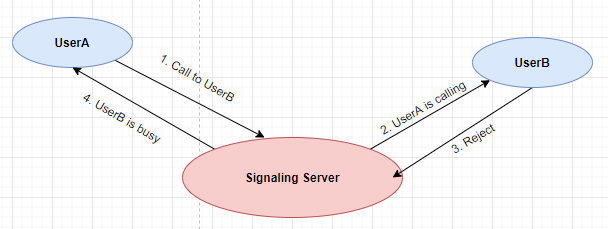
* *Thực hiện cuộc gọi đến người dùng đang trong cuộc gọi khác*



Hình 3.9 – Thực hiện cuộc gọi đến một user đang trong cuộc gọi khác

Trong quá trình một user thực hiện cuộc gọi đến một user khác, nếu user nhận cuộc gọi đang trong một cuộc gọi khác, Signaling Server sẽ gửi một thông điệp thông báo user nhận yêu cầu cuộc gọi đang trong một cuộc gọi khác (máy bận).

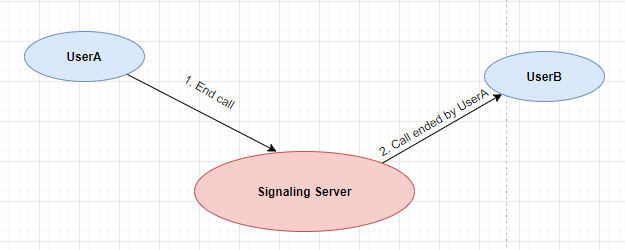
* *Người nhận từ chối cuộc gọi*



Hình 3.10 – Người nhận từ chối cuộc gọi

UserA thực hiện cuộc gọi đến UserB bằng cách gửi một thông điệp yêu cầu kết nối đến UserB đến Signaling Server (1), Server gửi tiếp thông điệp *UserA đang gọi, yêu cầu trả lời* tới UserB (2). UserB không muốn nghe máy nên thực hiện gửi thông điệp *không trả lời/từ chối* đến Server (3), Server chuyển lại thông điệp này về UserA (B).

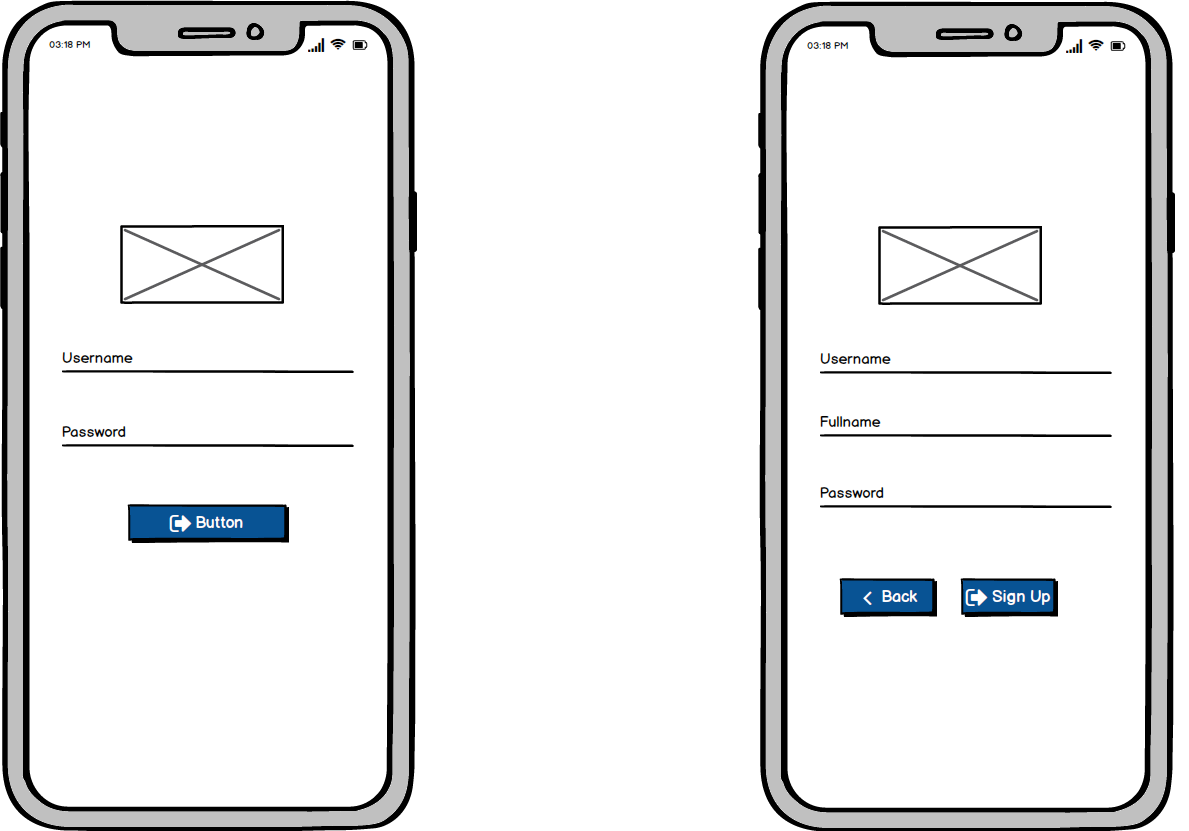
* *Kết thức một cuộc gọi*



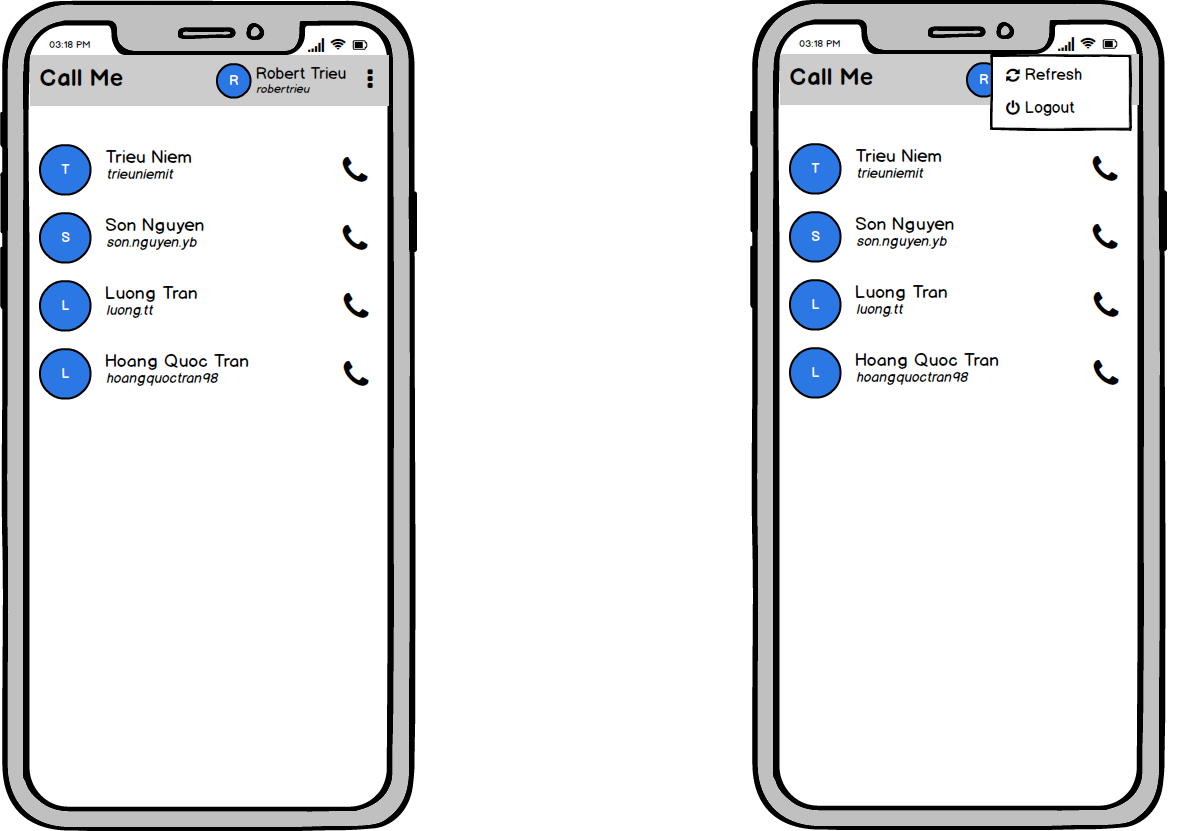
Hình 3.10 – Chấm dứt một cuộc gọi

Trong khi đang thực hiện cuộc gọi, giả sử UserA muốn chấm dứt cuộc gọi với UserB, UserA tiến hành ngắt kết nối P2P với APIs do WebRTC cung cấp, sau đó UserA gửi một thông điệp *ngắt cuộc gọi* đến Sinaling Server và thoát khỏi màn hình cuộc gọi (1), Sinaling Server chuyển tiếp thông điệp này đến UserB (2). Sau khi nhận được thông điệp *ngắt cuộc gọi* từ UserA, UserB sẽ thực hiện xóa *RTCConnectSession* qua các APIs WebRTC cung cấp và thoát khỏi màn hình cuộc gọi. Cuộc gọi được chấm dứt.

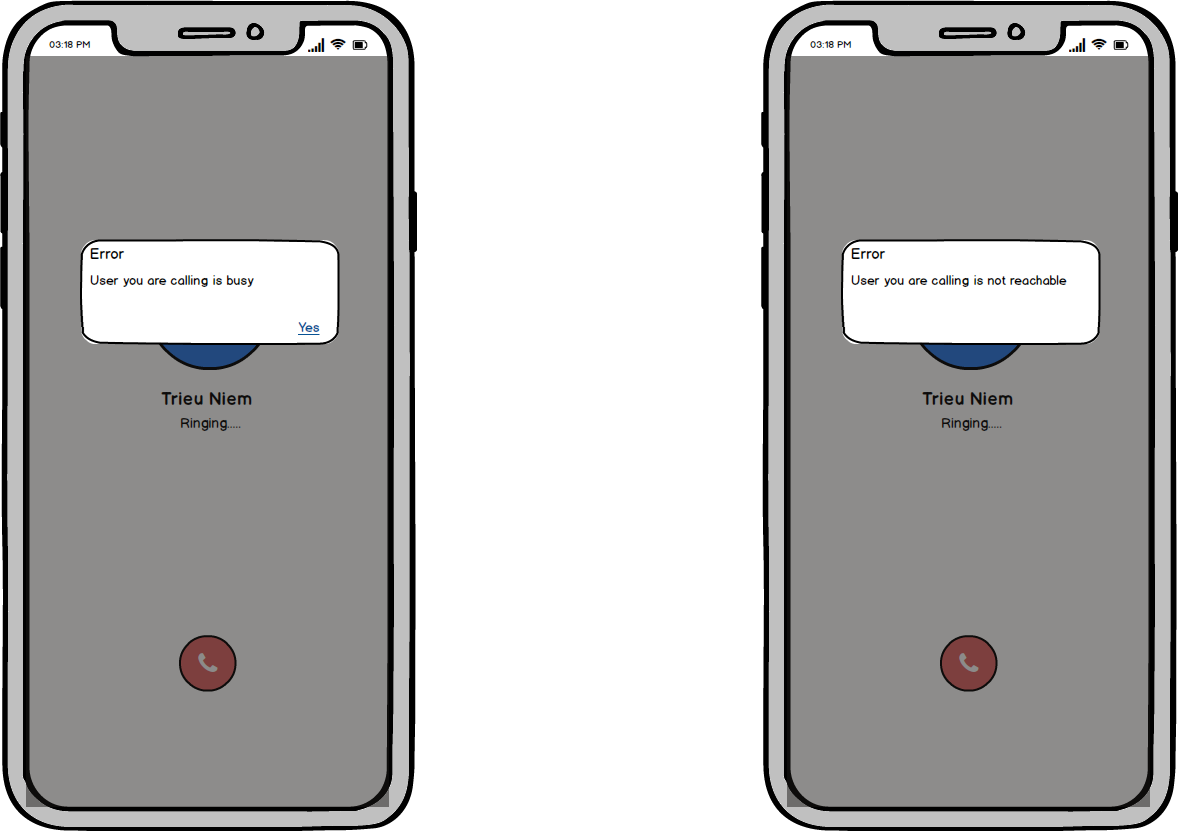
### Mockups giao diện



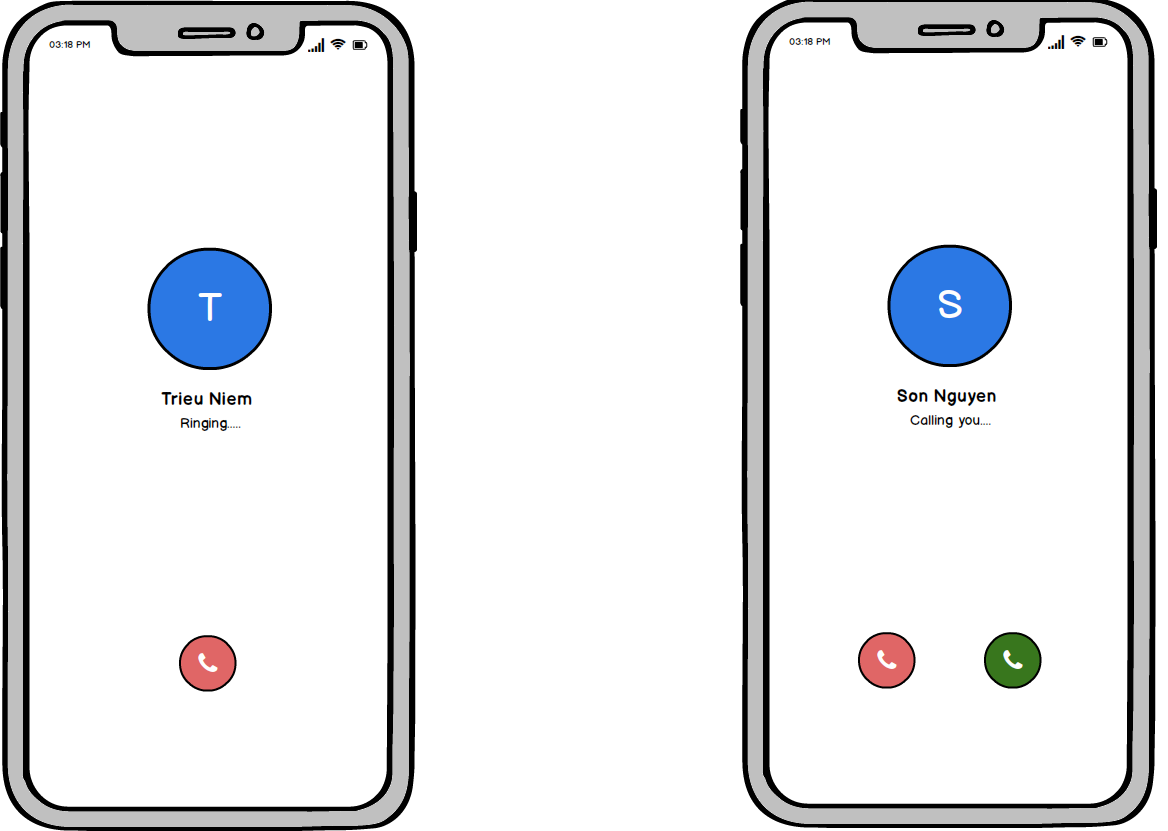
*Hình 3.7 – Mockup giao diện đăng nhập (trái) và đăng ký(phải)*



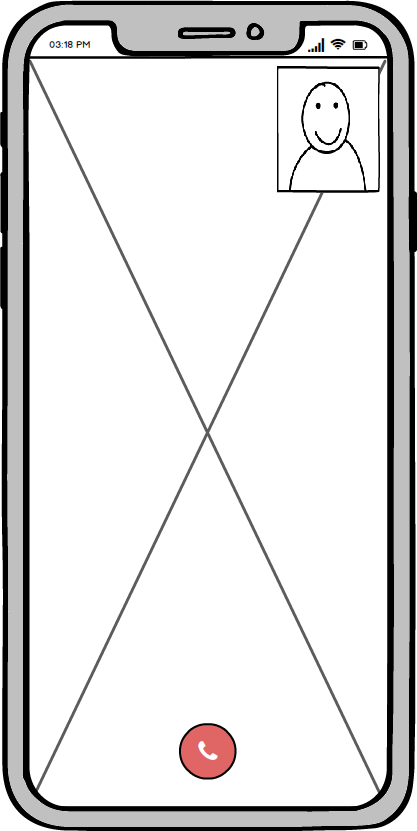
*Hình 3.8 – Mockup giao diện trang chủ sau khi đăng nhập thành công*



*Hình 3.9 – Mockup giao diện máy bận (trái) và giao diện user không trực tuyến (phải)*

**

*Hình 3.10 – Mockup giao diện chờ nghe máy (trái) và giao diện có cuộc gọi đến(phải)*



*Hình 3.11 – Mockup giao diện kết nối thành công (đang trong cuộc gọi)*

# KẾT LUẬN

Thông qua việc tìm hiểu và nghiên cứu đề tài này, em đã có cái nhìn toàn diện hơn về ứng dựng của của WebRTC trong phát triển sản phẩm công nghệ thông tin nói chung và phát triển phần mềm di động nói chung. Nó không chỉ giúp chúng ta giảm thiểu thời gian và chi phí xây dựng phần mềm. Đặc biệt giúp giảm thiểu số lượng công việc đối với người làm phần mềm.

### **Nội dụng đã thực hiện**

Trong thời gian hoàn thiện đề tài, em đã hiểu rõ được cơ chế hoạt động của WebRTC, vai trò của Signaling Server và từ đó xây dựng được một app di động có các chức năng cơ bản của một app gọi điện như:

* Xác thực user.
* Quản lý được danh bạ cá nhân
* Thực hiện cuộc gọi video giữa các user
* **Các hạn chế cần khắc phục**

Do thời gian hoàn thiện đề tài ngắn và kiến thức cá nhân của em còn hạn chế nên app hiên nay mới chỉ thực hiện được cuộc gọi trong mạng nội bộ do vấn đề về NATs và Firewall đã trình bày ở chương I. Bên cạnh đó việc thuê một Server thật để cài đặt và làm TURN hoạt động trên môi trường internet cũng là một khoản phí không nhỏ đối với sinh viên như em. Do đó đề tài này vẫn chưa thực sự hoàn thiện do nó vẫn chưa thể hoạt động tốt trong môi trường internet.

### **Hướng phát triển**

Từ những hạn chế trên của đề tài, chúng ta có thể có hướng phát triển của phần mềm như :

* Khắc phục lỗi không hoạt động được trên môi trường internet bằng cách xây dựng TURN server
* Thêm chức năng quản lý lịch sử cuộc gọi
* Tích hợp thêm AI nhận dạng khuôn mặt vào ứng dụng để ghép các sticker. Từ đó, có thể phát triển trở thành một mạng xã hội.

# Tài liệu tham khảo

[1] Bài viết *«Cơ bản về WebRTC»* của tác giả**khoatd92** tại blog<https://kipalog.com/>

[2] Wikipedia *– WeRTC* <https://vi.wikipedia.org/wiki/WebRTC>

[3] Luận văn thạc sĩ «*Nghiên cứu ứng dụng công nghệ WebRTC cho giải pháp cộng tác và chia sẽ dữ liệu đa phương tiện tại trung tâm MVAS-TCT viễn thông Mobifone»*  - **Nguyễn Viết Thắng**

[3] Các tài liệu tham khảo khác