

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN MÔN
LẬP TRÌNH WEB NÂNG CAO

ĐỀ TÀI:

“Web RTC”

Giảng viên hướng dẫn: ThS. MAI VĂN MẠNH

Sinh viên thực hiện: TÔ VĨNH KHANG – 51800408

BÙI QUANG KHẢI – 51800785

HÀ NGUYỄN HẢI ĐĂNG – 51800019

Lớp : 18050203

18050201

Khoa : 22

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN MÔN
LẬP TRÌNH WEB NÂNG CAO

ĐỀ TÀI:

“Web RTC”

Giảng viên hướng dẫn: ThS. MAI VĂN MẠNH

Sinh viên thực hiện: TÔ VĨNH KHANG – 51800408

BÙI QUANG KHẢI – 51800785

HÀ NGUYỄN HẢI ĐĂNG – 51800019

Lớp : 18050203

18050201

Khoa : 22

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Lập trình web nâng cao là một môn học rất hay và bổ ích, chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin và Trường Đại học Tôn Đức Thắng đã tạo điều kiện cho chúng em được học tập trong suốt thời gian qua. Chân thành cảm ơn thầy ThS. Mai Văn Mạnh đã hướng dẫn tận tình, giúp chúng em có thêm kiến thức về NodeJS. Hiểu rõ hơn về cách sử dụng công nghệ Web RTC để xây dựng một trang web cho phép người dùng gọi video trực tuyến.

Trong quá trình thực hiện bài tiểu luận này, nhóm vẫn khó tránh khỏi những sai sót không mong muốn, kính mong thầy có thể góp ý và giúp đỡ chúng em. Nhóm xin chân thành cảm ơn thầy.

BÀI TIÊU LUẬN NÀY ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm của riêng chúng tôi và được sự hướng dẫn của thầy Mai Văn Mạnh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài tập lớn này còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 20 tháng 02 năm 2020

Tác giả

(ký tên và ghi rõ họ tên)

Tô Vĩnh Khang

(Trưởng nhóm)

Bùi Quang Hải

Hà Nguyễn Hải Đăng

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

TÓM TẮT

Với đề tài “Web RTC” - Tìm hiểu về công nghệ Web RTC và xây dựng một trang web cho phép người dùng gọi video trực tuyến, nhóm đã thảo luận, tìm hiểu, phân tích yêu cầu về việc phải triển khai thiết kế một ứng dụng có thể gọi video cũng như trò chuyện với nhau trực tuyến. Để giúp cho việc sử dụng được dễ dàng, ứng dụng minh họa này phải đáp ứng được phần giao diện thân thiện, dễ sử dụng, chức năng video và chat phải theo thời gian thực (real-time). Thông qua các sơ đồ usecase, đặc tả usecase, sơ đồ lớp, sơ đồ tuần tự, sơ đồ hoạt động, mô hình thực thể ERD, mô hình dữ liệu quan hệ, đã minh họa chi tiết các chức năng cốt lõi của ứng dụng, cách thức hoạt động của từng chức năng và một số ràng buộc quan trọng. Qua đó, làm rõ được cách thức hoạt động của công nghệ web RTC.

Với đề tài hấp dẫn và thực tiễn này, nhóm mong muốn áp dụng những kiến thức đã học trong trường cùng với việc tìm hiểu nghiên cứu, phân tích hệ thống, những kiến thức từ bên ngoài cũng những trải nghiệm của cá nhân để xây dựng ứng dụng hỗ trợ được cho mọi người trong công việc lần học tập đạt được hiệu quả cao nhất.

MỤC LỤC

LỜI CẢM ƠN.....	3
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN.....	5
TÓM TẮT.....	6
MỤC LỤC.....	7
DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT.....	10
DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH.....	11
Danh mục Hình.....	11
Danh mục Bảng.....	12
CHƯƠNG I: CƠ SỞ LÝ THUYẾT.....	13
1.1. Web RTC.....	13
1.2. NodeJS.....	13
1.3. ExpressJS.....	14
1.4. PeerJS.....	14
1.5. SocketIO.....	15
1.6. Pug Engine.....	16
1.7. MongoDB.....	16
CHƯƠNG II: XÂY DỰNG TRANG WEB RTC.....	17
2.1. Phân tích yêu cầu.....	17
2.1.1. Yêu cầu chức năng.....	17
2.1.2. Yêu cầu phi chức năng.....	17
2.1.2.1. Yêu cầu về Hiệu suất.....	17
2.1.2.2. Yêu cầu về Bảo mật.....	17

2.1.2.3. Thuộc tính chất lượng phần mềm.....	17
2.1.3. Xác định Use case và các tác nhân.....	18
2.1.3.1. Xác định các tác nhân.....	18
2.1.3.2. Xác định các use case.....	18
2.1.4. Xác định phiên bản công nghệ sử dụng.....	19
2.2. Thiết kế hệ thống.....	19
2.2.1. Sơ đồ use case tổng quát.....	19
2.2.2. Sơ đồ hoạt động.....	20
2.2.2.1. Đăng nhập.....	20
2.2.2.2. Đăng xuất.....	20
2.2.2.3. Đăng kí.....	21
2.2.2.4. Tạo phòng.....	21
2.2.2.5. Tham gia phòng.....	22
2.2.2.6. Chia sẻ phòng.....	22
2.2.2.7. Video.....	23
2.2.2.8. Âm thanh.....	23
2.2.2.9. Thoát phòng.....	24
2.2.3. Sơ đồ tuần tự.....	24
2.2.3.1. Đăng nhập.....	24
2.2.3.2. Đăng xuất.....	25
2.2.3.3. Đăng kí.....	25
2.2.3.4. Tạo phòng.....	26
2.2.3.5. Tham gia phòng.....	26
2.2.3.6. Chia sẻ phòng.....	27
2.2.3.7. Video.....	27
2.2.3.8. Âm thanh.....	28
2.2.3.9. Thoát phòng.....	28

2.2.4. Sơ đồ mô hình thực thể ERD.....	28
CHƯƠNG III: HIỆN THỰC BẰNG CODE VÀ KẾT QUẢ.....	29
3.1. Hiện thực bằng code.....	29
3.1.1. Các thiết lập.....	29
3.1.1.1. Thư viện và công cụ hỗ trợ.....	29
3.1.1.2. Cơ sở dữ liệu lưu trữ.....	31
3.1.2. Các chức năng.....	32
3.1.2.1. Đăng ký tài khoản.....	32
3.1.2.2. Đăng nhập/Đăng xuất tài khoản.....	33
3.1.2.3. Tạo/Tham gia phòng họp.....	35
3.1.2.4. Phát video/Trò chuyện trực tuyến.....	35
3.1.2.5. Tắt mở Âm thanh/Video.....	39
3.1.2.6. Chia sẻ phòng họp.....	40
3.2. Kết quả.....	41
CHƯƠNG IV: TỔNG KẾT.....	45
PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ THÀNH VIÊN.....	46
Phân công công việc.....	46
Đánh giá thành viên.....	47
TÀI LIỆU THAM KHẢO.....	48

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

KKQ: Tên nhóm

ERD: Entity Relationship Diagram

DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH

Danh mục Hình

Hình 1. Hình ảnh về logo WebRTC.....	13
Hình 2. Hình ảnh về logo NodeJS.....	14
Hình 3. Hình ảnh về logo ExpressJS.....	14
Hình 4. Hình ảnh về logo ExpressJS.....	14
Hình 5. Hình ảnh về logo SocketIO.....	15
Hình 6. Hình ảnh về mô hình SocketIO.....	15
Hình 7. Hình ảnh về logo Pug Engine.....	16
Hình 8. Hình ảnh về logo MongoDB.....	16
Hình 9. Use case tổng quát.....	19
Hình 10. Sơ đồ hoạt động Đăng nhập.....	20
Hình 11. Sơ đồ hoạt động Đăng xuất.....	20
Hình 12. Sơ đồ hoạt động Đăng kí.....	21
Hình 13. Sơ đồ hoạt động Tạo phòng.....	21
Hình 14. Sơ đồ hoạt động Tham gia phòng.....	22
Hình 15. Sơ đồ hoạt động Chia sẻ phòng.....	22
Hình 16. Sơ đồ hoạt động Video.....	23
Hình 17. Sơ đồ hoạt động Âm thanh.....	23
Hình 18. Sơ đồ hoạt động Thoát phòng.....	24
Hình 19. Sơ đồ tuần tự Đăng nhập.....	24
Hình 20. Sơ đồ tuần tự Đăng xuất.....	25
Hình 21. Sơ đồ tuần tự Đăng kí.....	25
Hình 22. Sơ đồ tuần tự Tạo phòng.....	26

Hình 23. Sơ đồ tuần tự Tham gia phòng.....	26
Hình 24. Sơ đồ tuần tự Chia sẻ phòng.....	27
Hình 25. Sơ đồ tuần tự Video.....	27
Hình 26. Sơ đồ tuần tự Âm thanh.....	28
Hình 27. Sơ đồ tuần tự Thoát phòng.....	28
Hình 28. Mô hình thực thể ERD.....	28
Hình 29. Giao diện trang chủ khi chưa đăng nhập.....	41
Hình 30. Giao diện trang Đăng nhập.....	41
Hình 31. Giao diện trang Đăng ký.....	41
Hình 32. Giao diện trang chủ khi đã đăng nhập.....	42
Hình 33. Giao diện trang Chọn phòng.....	42
Hình 34. Giao diện trang Phòng họp.....	42
Hình 35. Tổ hợp chức năng có trong phát video trực tuyến.....	43
Hình 36. Chức năng trò chuyện.....	43
Hình 37. Minh họa video call có 3 người dùng trong phòng họp.....	43
Hình 38. Giao diện trang Chọn phòng được responsive trên các thiết bị khác.....	43
Hình 39. Giao diện trang Phòng họp được responsive trên các thiết bị khác.....	44

Danh mục Bảng

Bảng 1. Các use case.....	19
Bảng 2. Phân công công việc.....	47
Bảng 3. Đánh giá thành viên.....	47

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1.1. Web RTC

Web RTC (Web Real-Time Communication) là một web API được phát triển bởi W3C, khả năng hỗ trợ trình duyệt giao tiếp với nhau thông qua video call, voice call hay P2P mà không cần trình duyệt phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài. Các phần chính gồm:

- getUserMedia: Cho phép trình duyệt web truy cập vào camera hoặc microphone hoặc cả 2 để lấy dữ liệu hình ảnh, âm thanh cho việc truyền tải.
- RTCPeerConnection: Dùng để cài đặt video call hoặc voice call dùng cho việc truyền tải.
- RTCDataChannel: Cho phép trình duyệt chia sẻ dữ liệu P2P.

Có nhiều trường hợp sử dụng khác nhau cho WebRTC, từ các ứng dụng web cơ bản sử dụng máy ảnh hoặc micrô đến các ứng dụng gọi điện video và chia sẻ màn hình nâng cao hơn.



Hình 1. Hình ảnh về logo WebRTC

1.2. NodeJS

Là một hệ thống phần mềm được thiết kế để viết các ứng dụng internet có khả năng mở rộng, đặc biệt là máy chủ web. Chương trình được viết bằng JavaScript, chạy theo hướng sự kiện bất đồng bộ, sử dụng kỹ thuật điều khiển theo sự kiện, nhập/xuất không đồng bộ để tối thiểu tổng chi phí và tối đa khả năng mở rộng.



Hình 2. Hình ảnh về logo NodeJS

1.3. ExpressJS

ExpressJS là một framework được xây dựng trên nền tảng của Nodejs. Được phát hành dưới dạng phần mềm mã nguồn mở. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Nó hỗ trợ các method HTTP và middleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.



Hình 3. Hình ảnh về logo ExpressJS

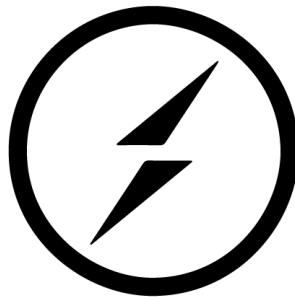
1.4. PeerJS

PeerJS là một framework của JavaScript, hoạt động như một trình bao bọc trong việc triển khai WebRTC của trình duyệt để cung cấp một API kết nối ngang hàng hoàn chỉnh, có thể định cấu hình và dễ sử dụng. Tạo một kết nối chỉ trong 3 dòng mã, cung cấp một id ngang hàng và một người ngang hàng có thể tạo kết nối luồng dữ liệu hoặc phuong tiện với một người ngang hàng từ xa.



Hình 4. Hình ảnh về logo ExpressJS

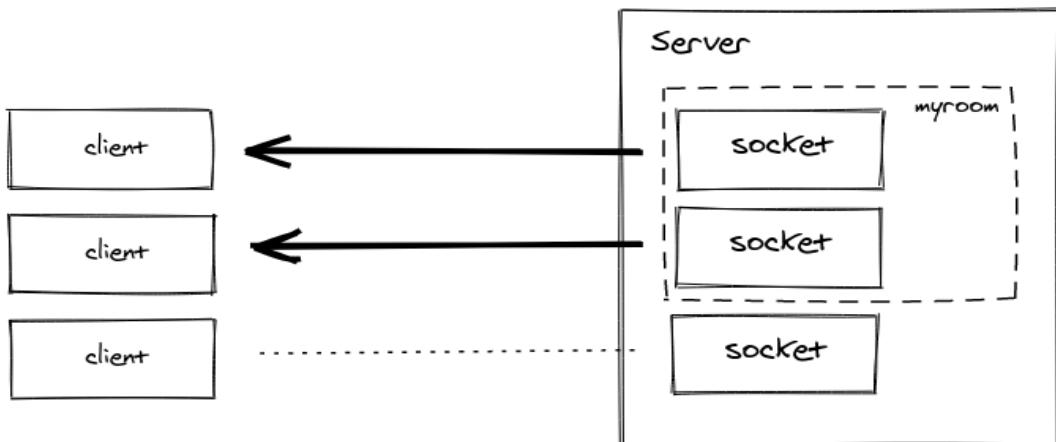
1.5. SocketIO



Hình 5. Hình ảnh về logo SocketIO

SocketIO là một framework của JavaScript dành cho các ứng dụng web thời gian thực. Cung cấp khả năng triển khai phân tích thời gian thực, phát trực tuyến nhị phân, nhắn tin tức thời và cộng tác tài liệu. Nó cho phép giao tiếp theo thời gian thực, hai chiều giữa các máy khách và máy chủ web.

Nó có 2 phần: một client-side thư viện chạy trong trình duyệt và một server-side thư viện cho Node.js. Cả 2 thành phần đều có một API gần giống nhau. Giống như Node.js, nó hướng đến sự kiện. Socket.IO xử lý kết nối một cách minh bạch.



Hình 6. Hình ảnh về mô hình SocketIO

1.6. Pug Engine

Pug Engine là một công cụ giúp làm sạch và cấu trúc code HTML. Cú pháp gọn gàng và dễ hiểu giống như ngôn ngữ lập trình Python. Pug Engine được xem là công cụ linh hoạt và đạt hiệu suất cao trong việc gói các đoạn lặp đi lặp lại, thay thế các biến trong tệp mẫu bằng các giá trị thực và chuyển đổi mẫu thành tệp HTML được gửi đến client và có thể tái sử dụng mà không cần phải viết lại. Giúp cho việc tổ chức các component được nhanh hơn.



Hình 7. Hình ảnh về logo Pug Engine

1.7. MongoDB

MongoDB là một chương trình cơ sở dữ liệu hướng tài liệu (document) đa nền tảng có sẵn nguồn. Được phân loại là một chương trình cơ sở dữ liệu NoSQL. Mỗi bộ sưu tập (collection) sẽ có các kích cỡ và các tài liệu khác nhau. Các dữ liệu được lưu trữ trong tài liệu kiểu JSON nên việc truy vấn rất nhanh. Dữ liệu lưu trữ phi cấu trúc, tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ, dữ liệu được ghi đệm lên RAM. Tuy nhiên, không có tính ràng buộc nên không được ứng dụng cho các mô hình giao dịch nào có yêu cầu độ chính xác cao, khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất cũng như yêu cầu một bộ nhớ RAM phải lớn.



Hình 8. Hình ảnh về logo MongoDB

CHƯƠNG II: XÂY DỰNG TRANG WEB RTC

2.1. Phân tích yêu cầu

2.1.1. Yêu cầu chức năng

Người dùng có thể truy cập trang web từ máy tính hoặc thiết bị di động để gọi video trực tuyến, trò chuyện trực tuyến. Các phòng họp mặt được tạo ra là riêng biệt. Người dùng tạo phòng sau đó có thể chia sẻ liên kết để mời người dùng khác vào phòng họp mặt. Hoặc có thể nhập mã phòng từ người khác để vào phòng của họ. Có thể tắt mở được âm thanh, video và thoát khỏi phòng. Điều hướng người dùng về trang đăng nhập nếu truy cập đường dẫn phòng mà chưa thực hiện bước đăng nhập.

2.1.2. Yêu cầu phi chức năng

2.1.2.1. Yêu cầu về Hiệu suất

Dữ liệu được cập nhật theo thời gian thực, nhất quán toàn vẹn; Thông tin tiếp nhận được nhanh chóng hiệu quả; Giao diện thân thiện, dễ sử dụng, tốc độ phản hồi nhanh; Có khả năng duy trì thời gian truy cập cao và ổn định.

2.1.2.2. Yêu cầu về Bảo mật

Đảm bảo dữ liệu phòng họp mặt truy cập hiển thị đúng với các tài khoản người dùng khác đã truy cập.

2.1.2.3. Thuộc tính chất lượng phần mềm

Xây dựng trang web sử dụng công nghệ Web RTC, sử dụng mọi lúc mọi nơi; Dễ dàng bảo trì, nâng cấp, phát triển phần mềm sau khi đã triển khai. Khả năng tương tác, khả năng bảo trì, tính di động, độ tin cậy, tính mạnh mẽ; Khả năng kiểm soát dữ liệu tuyệt đối, an toàn và bảo mật.

2.1.3. Xác định Use case và các tác nhân

2.1.3.1. Xác định các tác nhân

Qua tìm hiểu từ các nguồn cũng như những kiến thức sẵn có, chúng em xác định chỉ có tác nhân là người dùng (User).

2.1.3.2. Xác định các use case

Các usecase được xem là cần thiết cho hệ thống đã được nhóm đúc kết như sau:

ID	Use case	Mô tả
LI	Đăng nhập (Log In)	Người dùng đăng nhập vào ứng dụng
LO	Đăng xuất (Log Out)	Người dùng đăng xuất khỏi ứng dụng
SU	Đăng ký (Sign Up)	Người dùng đăng ký tài khoản lên hệ thống
CR	Tạo phòng (Create Room)	Người dùng tạo phòng họp mặt
JR	Tham gia phòng (Join Room)	Người dùng nhập mã phòng để vào phòng họp mặt
SR	Chia sẻ phòng (Share Room)	Người dùng chia sẻ mã phòng vào phòng họp mặt
CM	Video	Người dùng bật/tắt video
MC	Âm thanh (Audio)	Người dùng bật/tắt âm thanh
EX	Thoát phòng (Leave Room)	Người dùng thoát khỏi phòng họp mặt

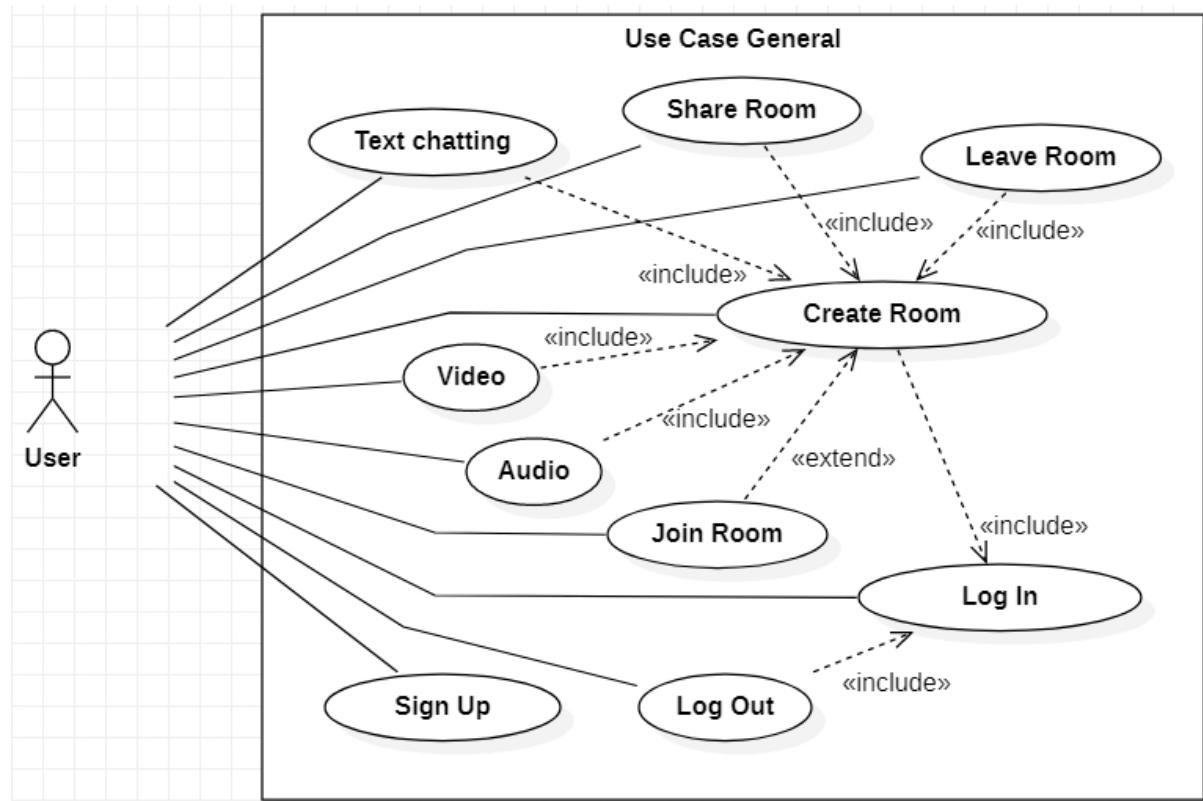
Bảng 1. Các use case

2.1.4. Xác định phiên bản công nghệ sử dụng

- NodeJS (Phiên bản 14.15.0)
- ExpressJS (Phiên bản 4.17.1)
- PeerJS (Phiên bản 0.6.1)
- SocketIO (Phiên bản 3.1.1)
- Pug Engine (Phiên bản 3.0.0)
- MongoDB (Phiên bản 4.4.4)

2.2. Thiết kế hệ thống

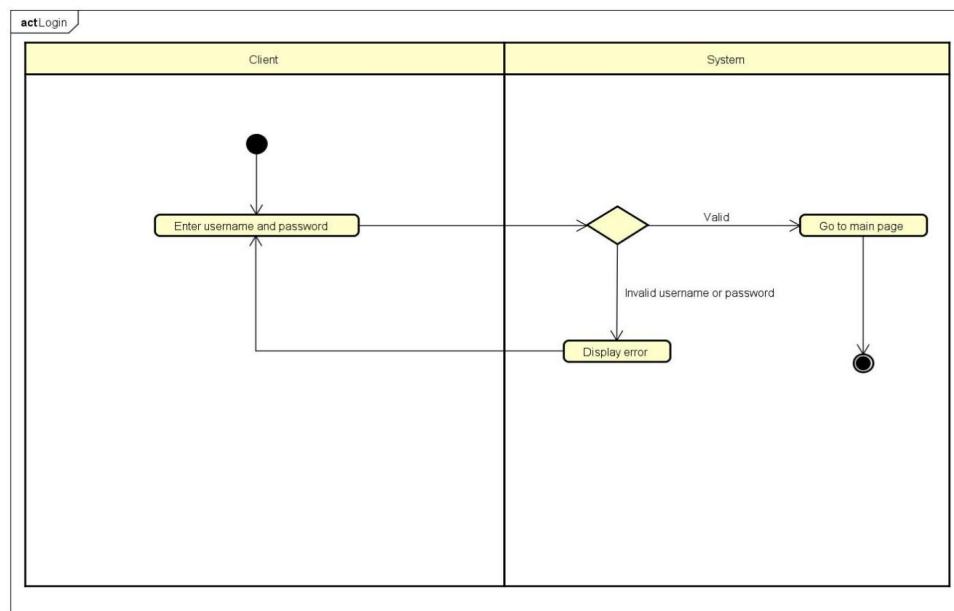
2.2.1. Sơ đồ use case tổng quát



Hình 9. Use case tổng quát

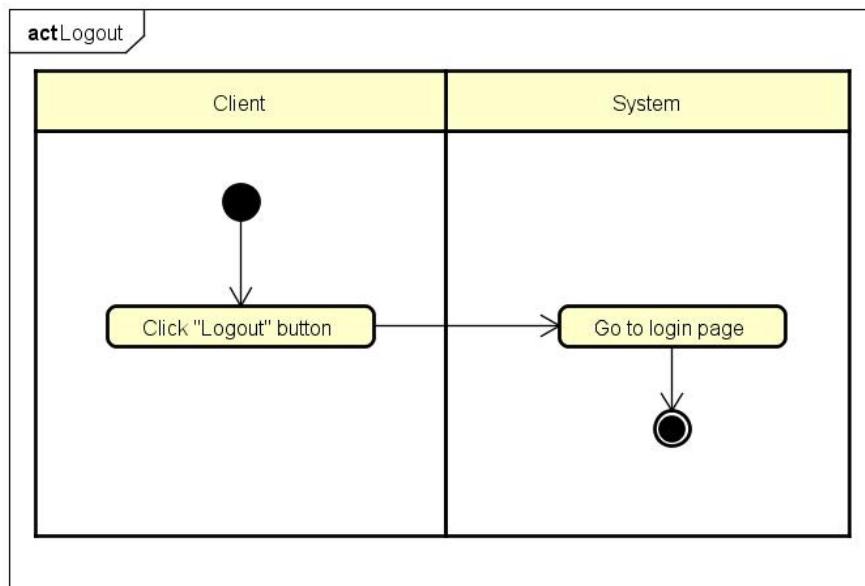
2.2.2. Sơ đồ hoạt động

2.2.2.1. Đăng nhập



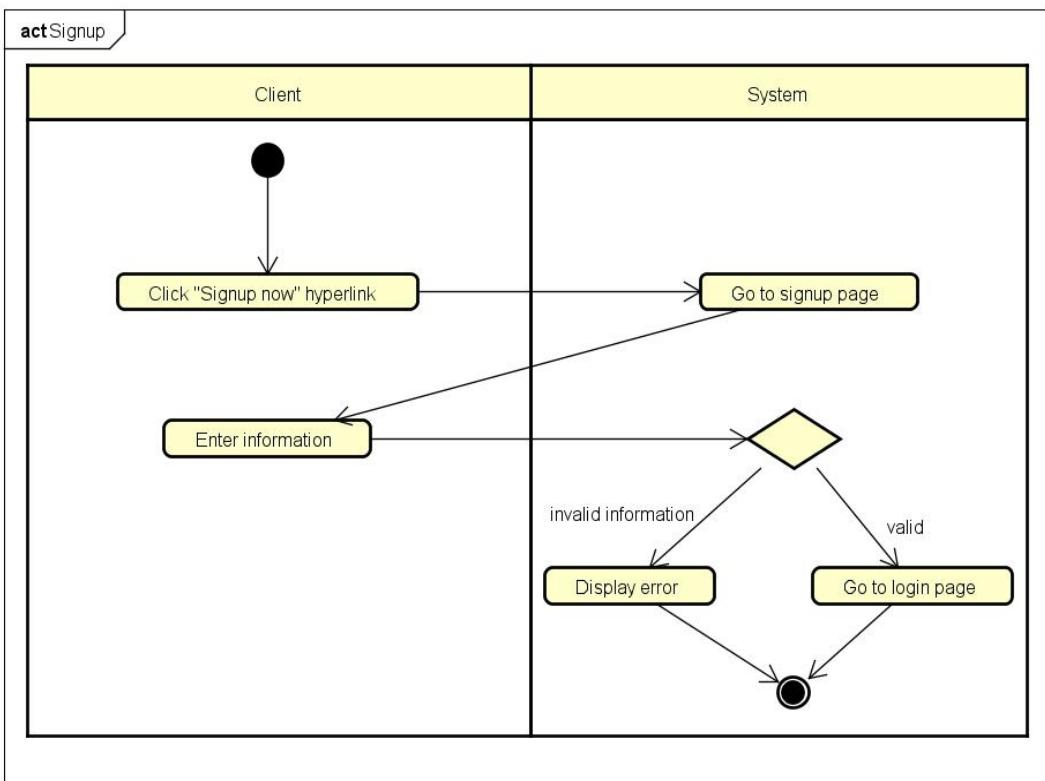
Hình 10. Sơ đồ hoạt động Đăng nhập

2.2.2.2. Đăng xuất



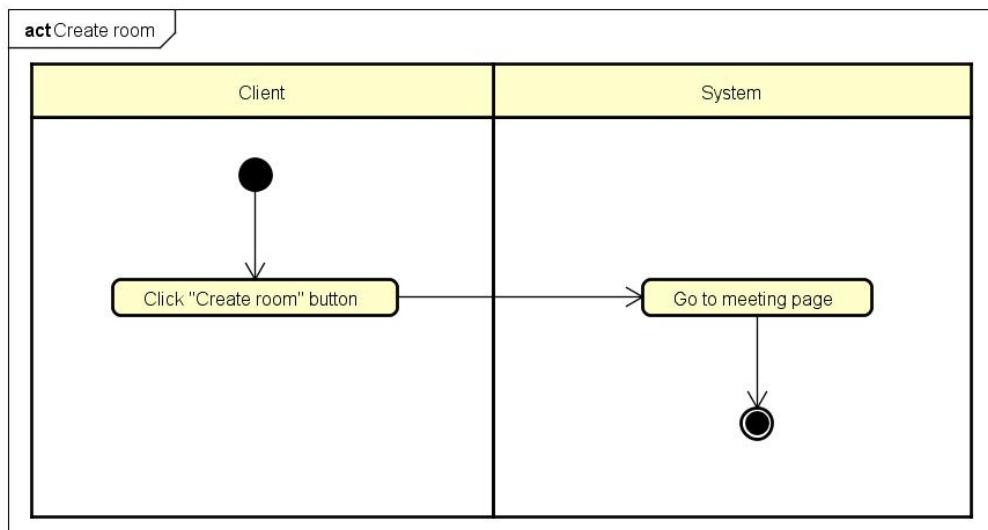
Hình 11. Sơ đồ hoạt động Đăng xuất

2.2.2.3. Đăng ký



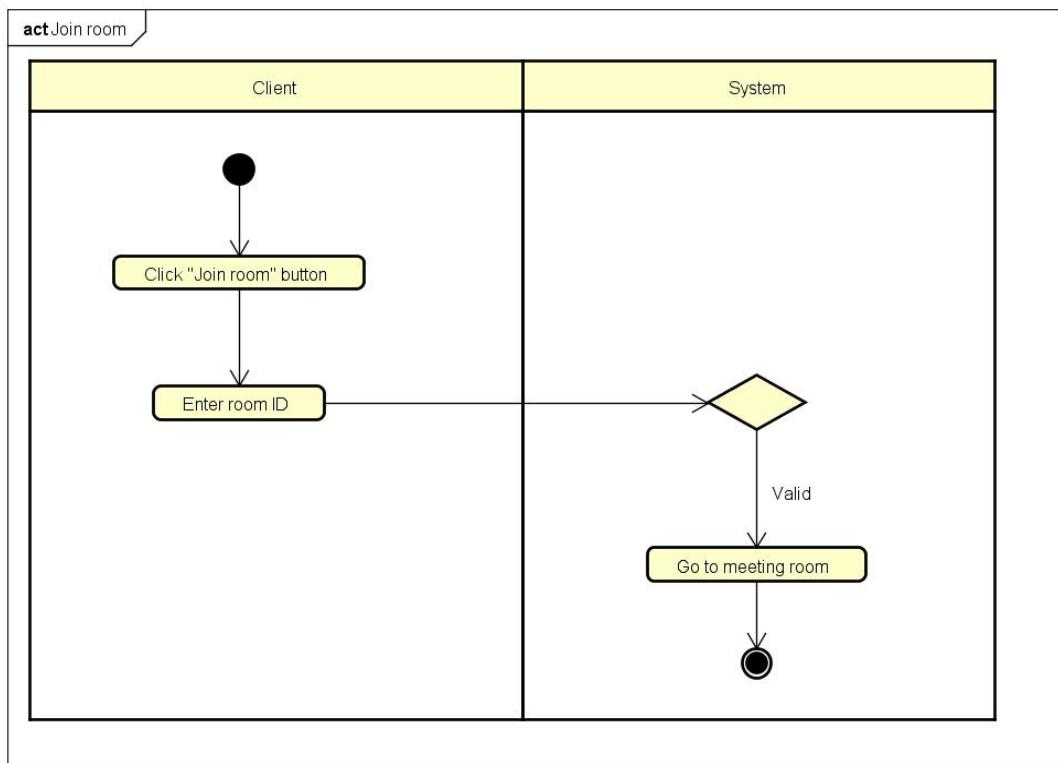
Hình 12. Sơ đồ hoạt động Đăng ký

2.2.2.4. Tạo phòng



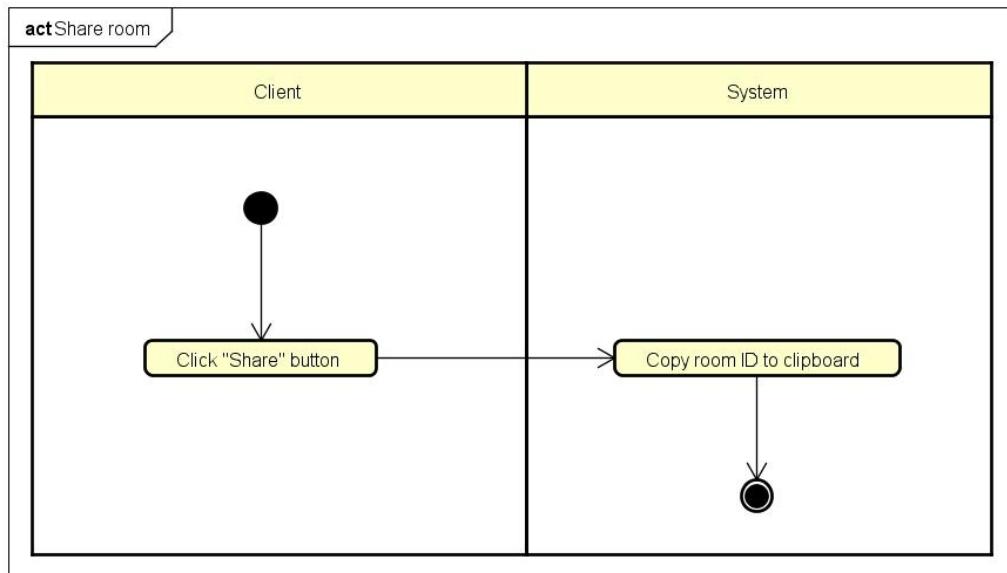
Hình 13. Sơ đồ hoạt động Tạo phòng

2.2.2.5. Tham gia phòng



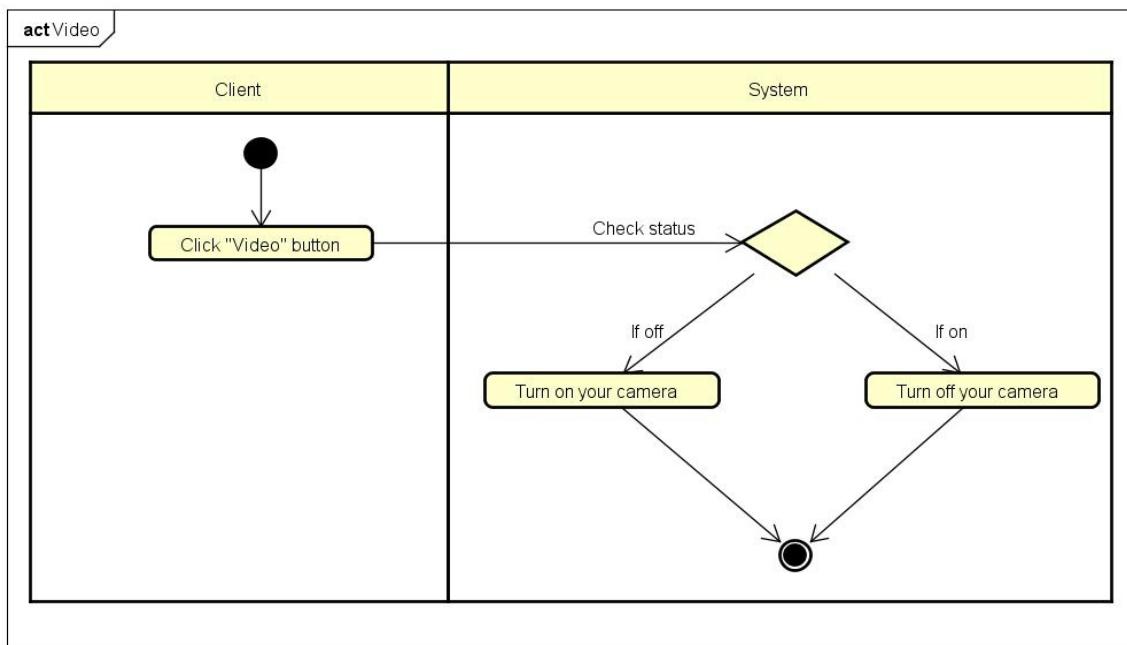
Hình 14. Sơ đồ hoạt động Tham gia phòng

2.2.2.6. Chia sẻ phòng



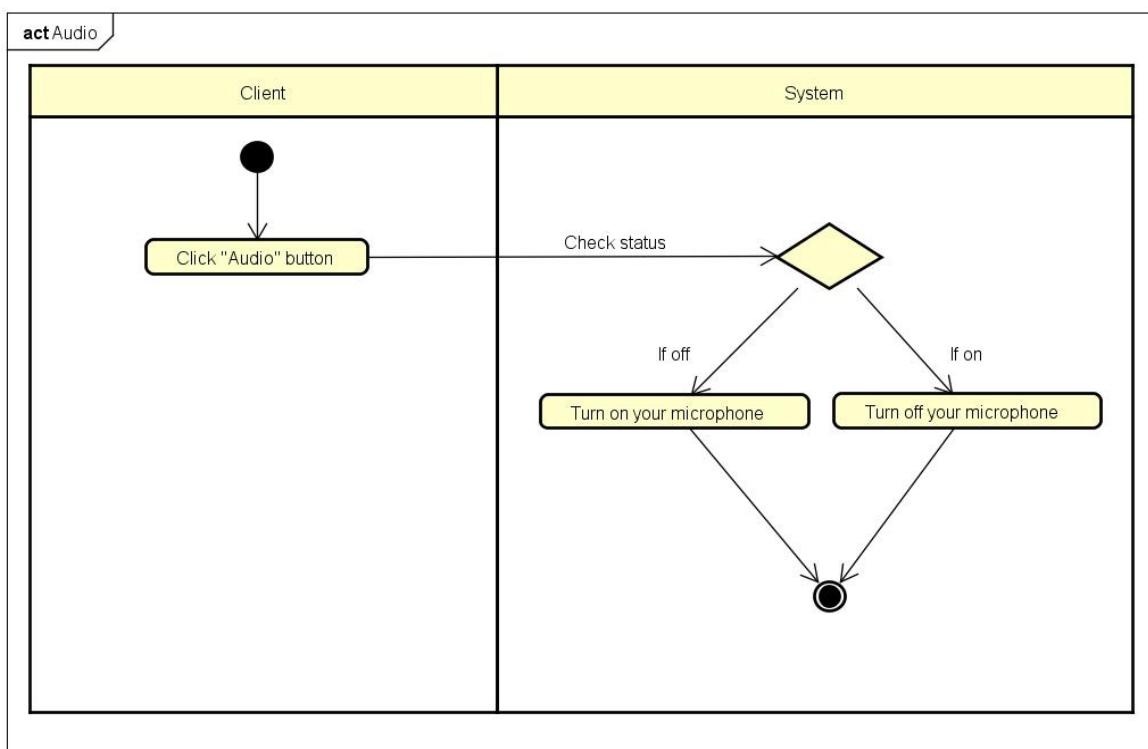
Hình 15. Sơ đồ hoạt động Chia sẻ phòng

2.2.2.7. Video



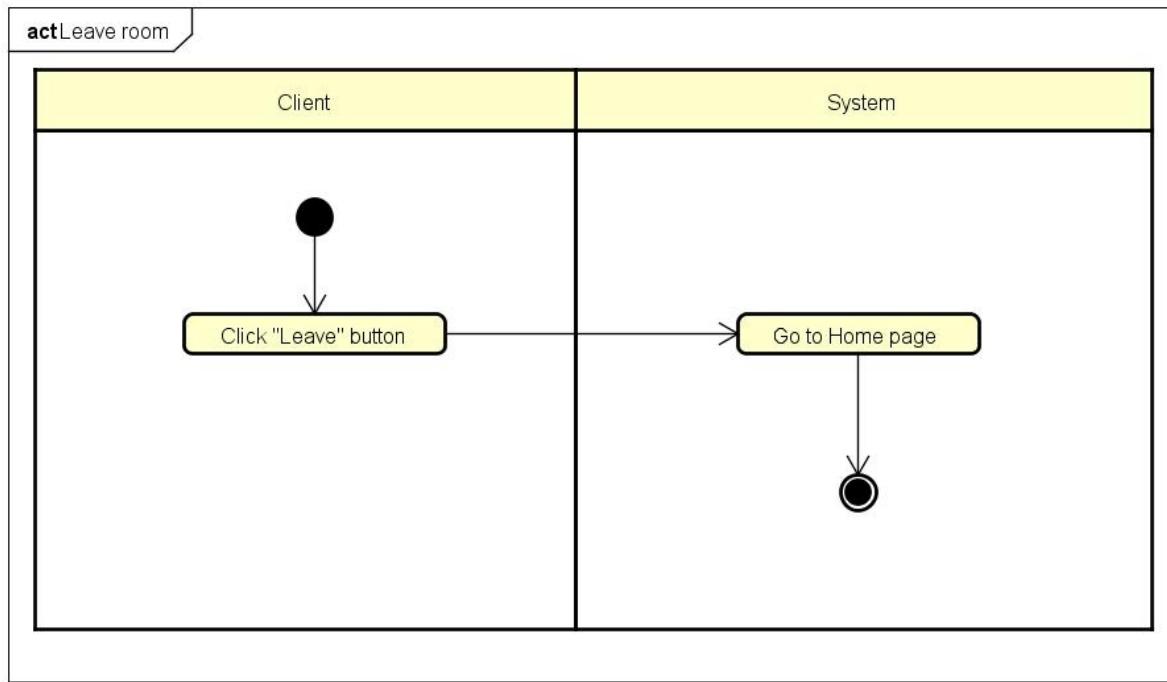
Hình 16. Sơ đồ hoạt động Video

2.2.2.8. Âm thanh



Hình 17. Sơ đồ hoạt động Âm thanh

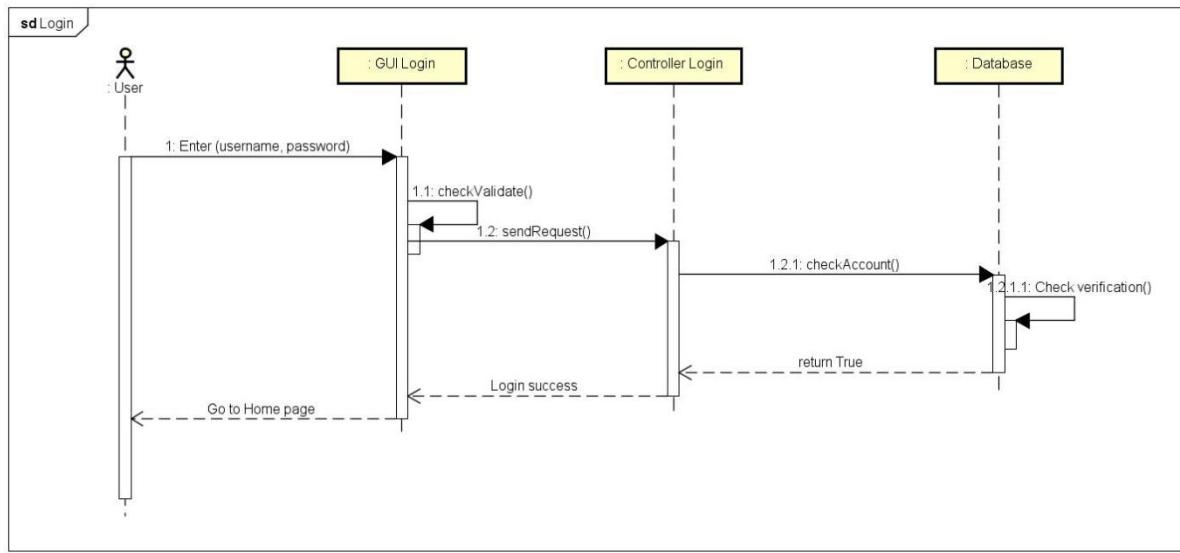
2.2.2.9. Thoát phòng



Hình 18. Sơ đồ hoạt động Thoát phòng

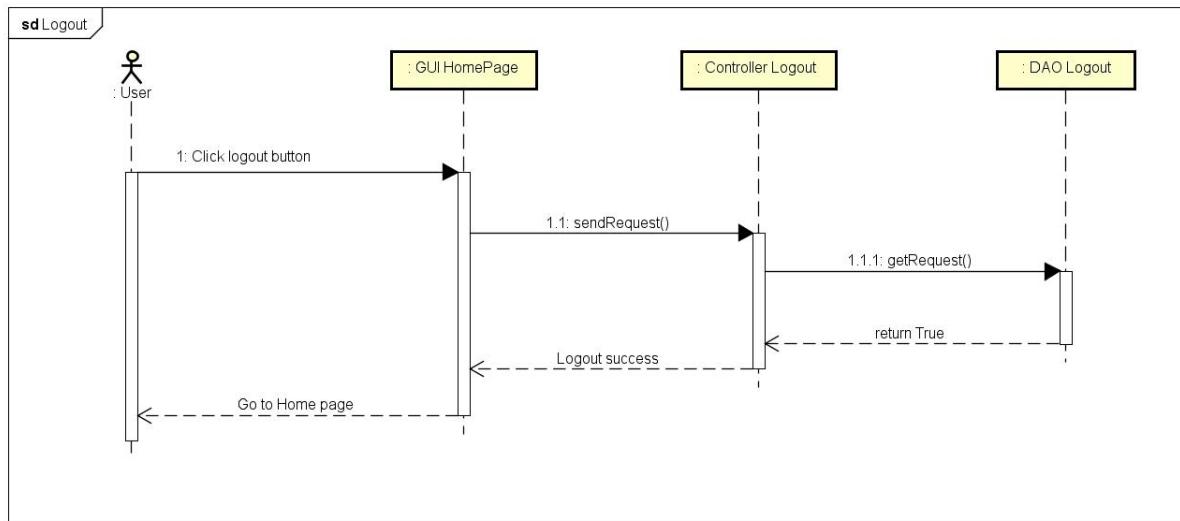
2.2.3. Sơ đồ tuần tự

2.2.3.1. Đăng nhập



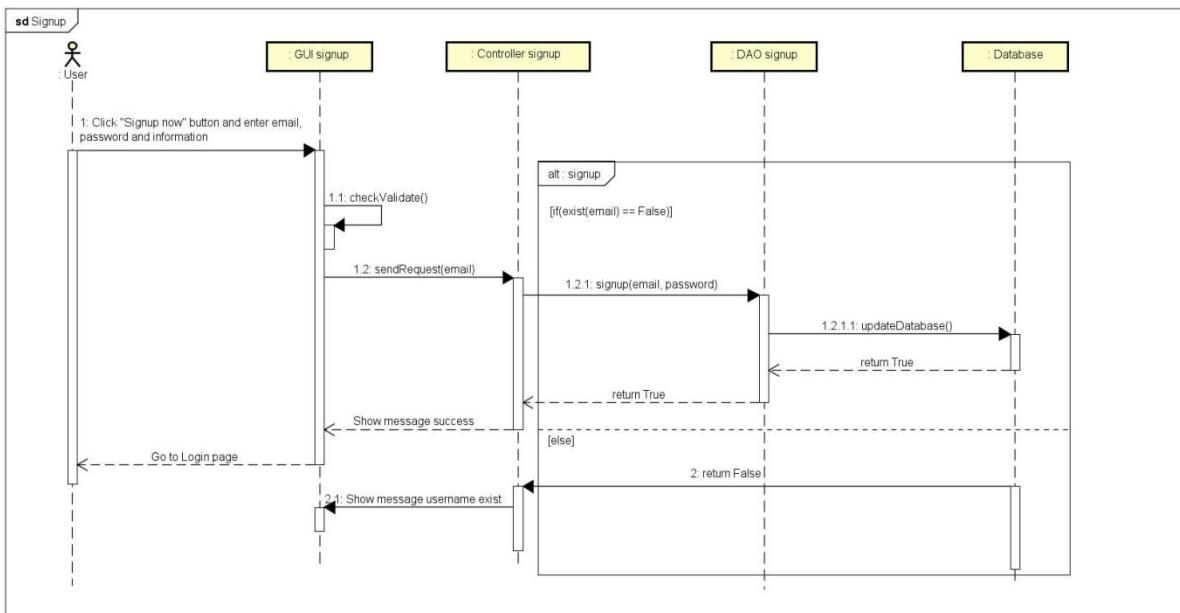
Hình 19. Sơ đồ tuần tự Đăng nhập

2.2.3.2. Đăng xuất



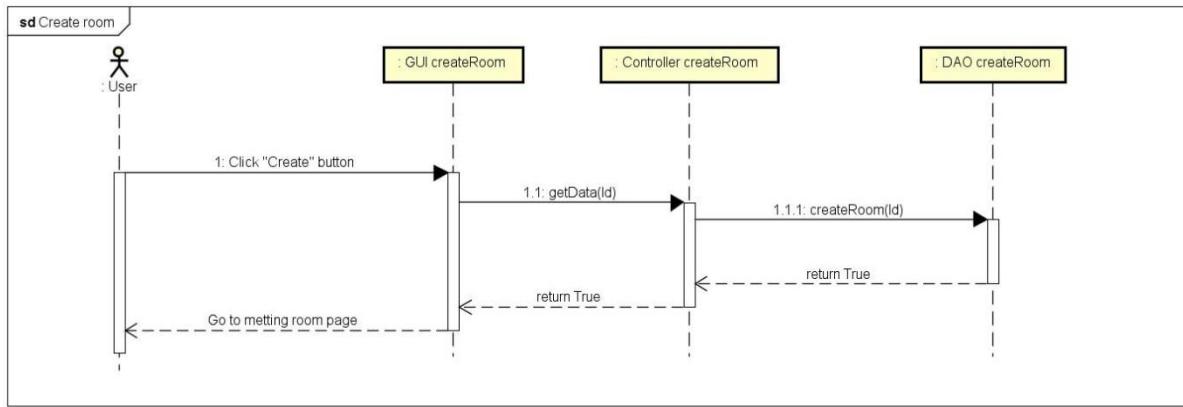
Hình 20. Sơ đồ tuần tự Đăng xuất

2.2.3.3. Đăng ký



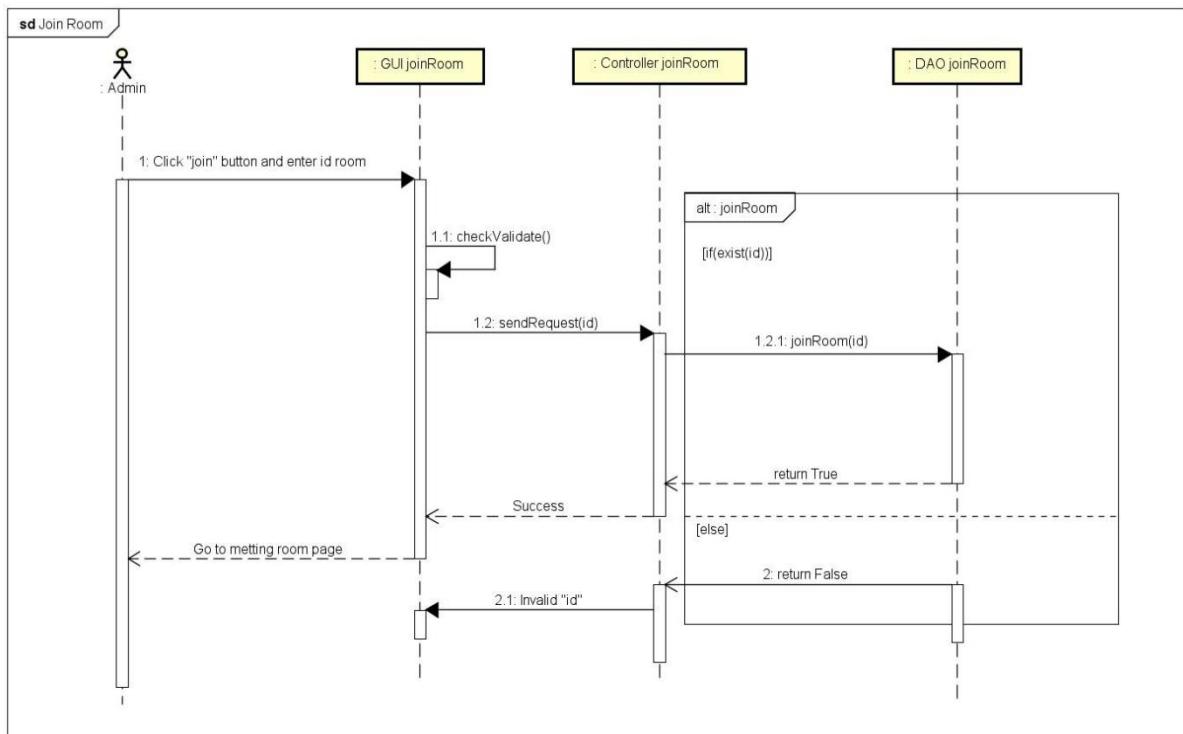
Hình 21. Sơ đồ tuần tự Đăng ký

2.2.3.4. Tạo phòng



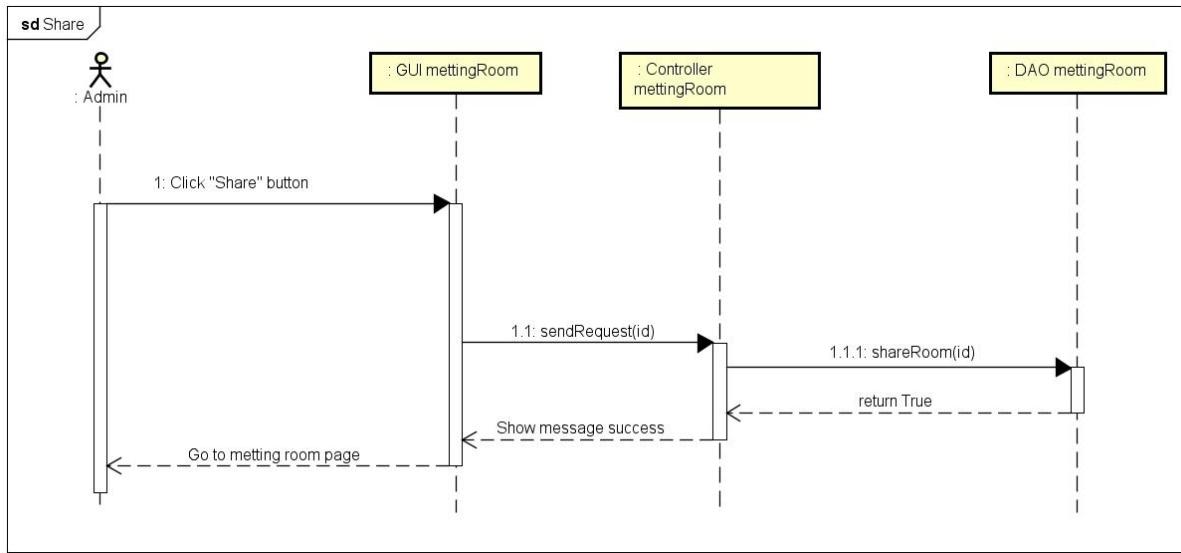
Hình 22. Sơ đồ tuần tự Tạo phòng

2.2.3.5. Tham gia phòng



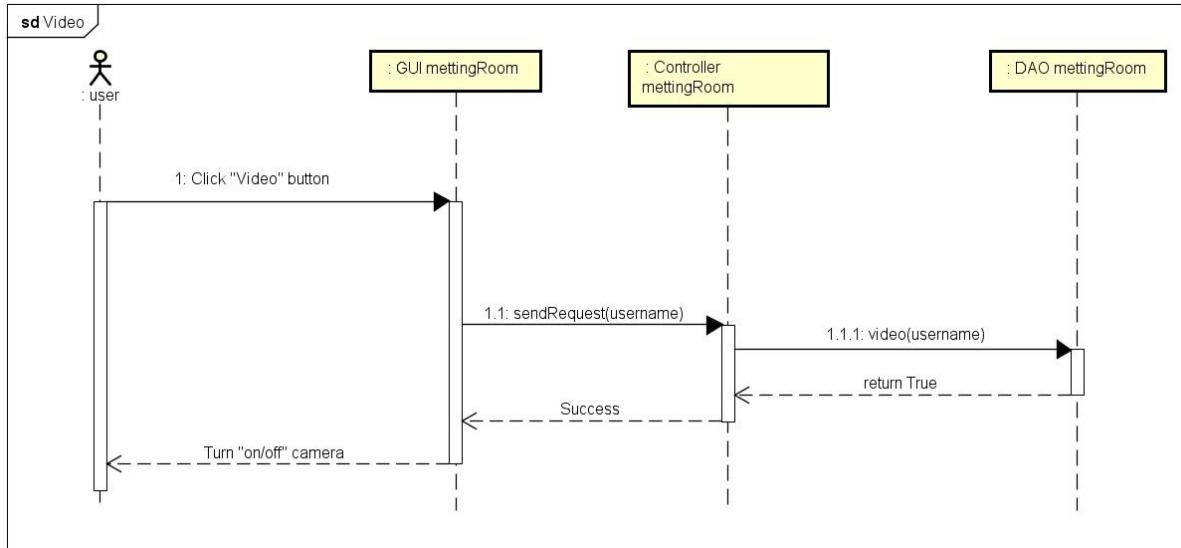
Hình 23. Sơ đồ tuần tự Tham gia phòng

2.2.3.6. Chia sẻ phòng



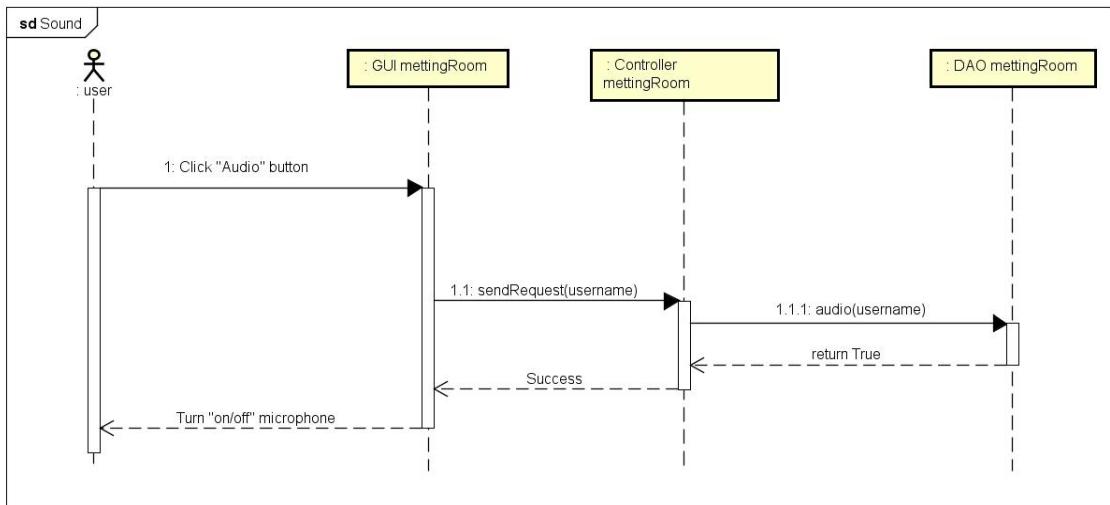
Hình 24. Sơ đồ tuần tự Chia sẻ phòng

2.2.3.7. Video



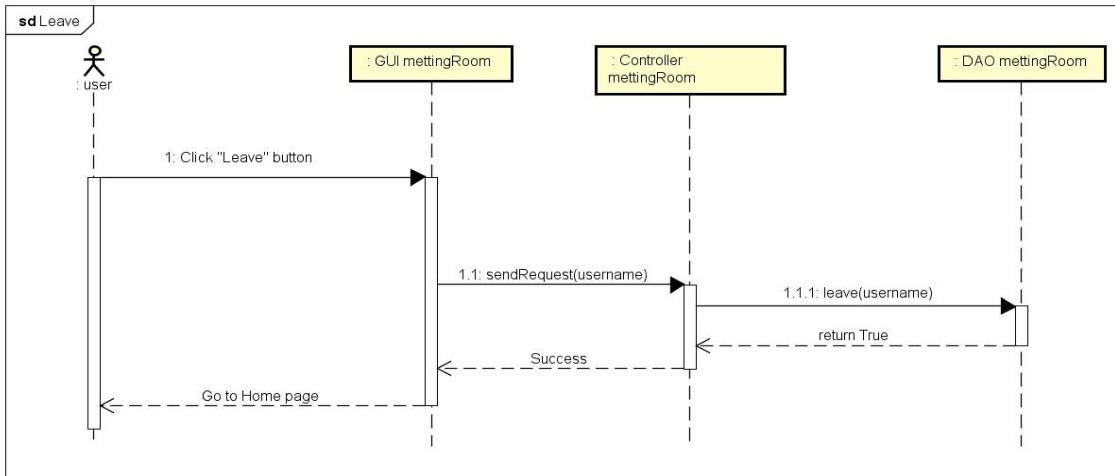
Hình 25. Sơ đồ tuần tự Video

2.2.3.8. Âm thanh



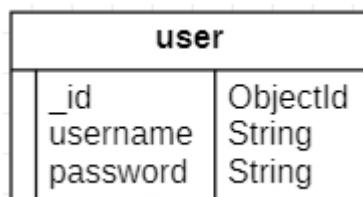
Hình 26. Sơ đồ tuần tự Âm thanh

2.2.3.9. Thoát phòng



Hình 27. Sơ đồ tuần tự Thoát phòng

2.2.4. Sơ đồ mô hình thực thể ERD



Hình 28. Mô hình thực thể ERD

CHƯƠNG III: HIỆN THỰC BẰNG CODE VÀ KẾT QUẢ

3.1. Hiện thực bằng code

3.1.1. Các thiết lập

3.1.1.1. Thư viện và công cụ hỗ trợ

```
require("dotenv").config();
const express = require("express");
const { v4: v4UniqueId } = require("uuid");
const http = require("http");
const { ExpressPeerServer } = require("peer");
const app = express();
const port = process.env.PORT || 7000;
const httpServer = http.Server(app);
const io = require("socket.io")(httpServer);
const peerServer = ExpressPeerServer(httpServer, {debug: true,});
const bcrypt = require("bcrypt");
const path = require("path");
const session = require("express-session");
const bodyParser = require("body-parser");
const SESSION_NAME = process.env.SESSION_NAME;
const mongoose = require("mongoose");
```

Thiết lập các thư viện ở phía server của ứng dụng. Các thư viện cơ bản để cài đặt và sử dụng như Express Framework, Express Session, biến môi trường (dotenv), tạo unique id (v4 của uuid), tạo kết nối tới cơ sở dữ liệu mongodb (mongoose) và mã hóa mật khẩu bằng Bcrypt. Ngoài ra, còn cần các thư viện chính để có thể khởi tạo và thiết lập nền cơ chế của một ứng dụng có sử dụng WebRTC để thực hiện gọi video và trò chuyện trực tuyến là HTTP, PeerJS và Socket.io.

```

app.set("view engine", "pug");
app.set("views", path.join(__dirname, "views"));
app.use(express.static("public"));
app.use("/peerjs", peerServer);
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(
  session({
    name: SESSION_NAME,
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: true,
    cookie: { maxAge: 1000 * 60 * 60 * 24 },
  })
);

```

Sử dụng view engine Pug và các middlewares cho ứng dụng. Thiết lập đường ống “/peerjs” liên kết từ xa với phía client có cùng đường dẫn. Biến cookie dùng để xác định thời gian hiệu lực của session là 24 giờ. Tại file meetingRoom.js Thiết lập kết nối với Socket.io và PeerJS từ phía client có cùng đường ống dẫn “/peerjs” khi tham gia vào phòng họp.

```

const socket = io("/");
let peer = new Peer(undefined, {
  host: "/",
  // For development
  port: "7000",
  // For production
  // port: "443",
  path: "/peerjs",
});

```

```
peer.on("open", (userId) => {
    socket.emit("Entry room", meetingRoomId, userId);
});
```

Tiếp theo là tạo 2 custom middlewares “checkLogin” và “preventWhenLogged” để tiện cho việc kiểm tra đã đăng nhập hoặc chưa đăng nhập và xử lý các điều hướng trang phù hợp:

```
const checkLogin = (req, res, next) => {
    if (!req.session.username) { res.redirect("/login"); } else { next(); }
};

const preventWhenLogged = (req, res, next) => {
    if (req.session.username) { res.redirect("/"); } else { next(); }
};
```

3.1.1.2. Cơ sở dữ liệu lưu trữ

```
const AccountModel = require("./models/account");
mongoose.connect(process.env.MONGODBLOCAL_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
});
```

Thiết lập kết nối từ ứng dụng đến cơ sở dữ liệu Mongodb trên máy chủ localhost và gọi Account Model để có thể tương tác với collection user của cơ sở dữ liệu mongodb.

```
mongoose.connect(process.env.MONGODBONLINE_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
});
```

Tương tự ta cũng sẽ có thiết lập kết nối cho cơ sở dữ liệu Mongodb Atlas để sử dụng cho production.

3.1.2. Các chức năng

3.1.2.1. Đăng ký tài khoản

```
app.get("/signup", preventWhenLogged, (req, res) => {
  res.render("signup");
});
```

Thiết lập đường dẫn trang đăng ký tài khoản cho người dùng với phương thức GET khi truy cập vào trang đăng ký tài khoản. Nếu người dùng đã đăng nhập thì sẽ không thể quay lại trang đăng ký vì đã có middleware kiểm tra.

```
app.post("/signup", preventWhenLogged, (req, res) => {
  var username = req.body.username;
  var password = req.body.password;
  var confirmPassword = req.body.confirmPassword;
  let encryptedPassword = "";
  if(confirmPassword != password) {
    res.render("signup", {
      error: "Confirm password does not match",
    });
  } else {
    bcrypt.genSalt(10, (err, salt) => {
      bcrypt.hash(password, salt, function (err, hash) {
        encryptedPassword = hash;
        AccountModel.findOne({
          username: username,
        })
        .then((data) => {
          if (data) {
```

```

    res.render("signup", {
      error: "User has already existed",
    });
  } else {
    AccountModel.create({
      username: username,
      password: encryptedPassword,
    });
    res.redirect("/login");
  }
})
.catch((err) => {
  res.render("signup", {
    error: "User has already existed",
  });
});
}
);
}
);

```

Thiết lập phương thức POST cho đường dẫn đăng ký tài khoản để khi người dùng đăng ký tài khoản thì dữ liệu của người dùng sẽ được gửi lên server để kiểm tra và xác nhận. Nếu thông tin được xác nhận thành công thì dữ liệu của người dùng sẽ được lưu vào cơ sở dữ liệu và mật khẩu sẽ được mã hóa Bcrypt. Nếu người dùng đã đăng nhập thì sẽ không thể thực hiện phương thức này vì đã có middleware kiểm tra.

3.1.2.2. Đăng nhập/Đăng xuất tài khoản

```

app.get("/login", preventWhenLogged, (req, res) => {
  res.render("login");
});

```

Thiết lập đường dẫn trang đăng nhập cho người dùng với phương thức GET khi truy cập vào trang đăng nhập. Nếu người dùng đã đăng nhập thì sẽ không thể quay lại trang đăng nhập vì đã có middleware kiểm tra.

```
app.post("/login", preventWhenLogged, (req, res) => {
  AccountModel.findOne({username: req.body.username,})
    .then((data) => {
      bcrypt
        .compare(req.body.password, data.password)
        .then(function (result) {
          if (result) {req.session.username = req.body.username;
          res.redirect("/");
        } else {
          res.render("login", { error: "Invalid email or password",});
        }
      });
    }).catch((err) => {
      res.render("login", {error: "Invalid email or password",});
    });
});
```

Thiết lập phương thức POST cho đường dẫn đăng nhập để khi người dùng đăng nhập thì dữ liệu của người dùng sẽ được gửi lên server để kiểm tra và xác nhận. Nếu thông tin được xác nhận thành công thì người dùng sẽ được điều hướng về trang chủ. Nếu người dùng đã đăng nhập thì sẽ không thể thực hiện phương thức này vì đã có middleware kiểm tra.

```
app.get("/logout", checkLogin, (req, res) => {
  req.session.destroy((error) => {
    if (error) { res.redirect("/");}
    else {res.clearCookie(SESSION_NAME);res.redirect("/");}
  });
});
```

```
});  
});
```

Đăng xuất sẽ xóa cookie và loại bỏ session, điều hướng về lại trang chủ.

3.1.2.3. Tạo/Tham gia phòng họp

```
app.get("/roomoption", checkLogin, (req, res) => {  
    res.render("roomOption");  
});
```

Hiển thị trang chọn chức năng trước khi tham gia vào phòng họp, nếu chưa đăng nhập thì sẽ không vào được trang này.

```
app.get("/create", checkLogin, (req, res) => {  
    res.redirect(`/meetingroom/${v4UniqueId()}`);  
});
```

Chức năng tạo phòng họp với unique id, nếu chưa đăng nhập thì sẽ không thực hiện được chức năng này.

```
app.post("/join", checkLogin, (req, res) => {  
    res.redirect(`/meetingroom/${req.body.joinRoomId}`);  
});
```

Chức năng tham gia phòng họp với unique id được lấy từ form input, nếu chưa đăng nhập thì sẽ không thực hiện được chức năng này.

3.1.2.4. Phát video/Trò chuyện trực tuyến

```
io.on("connection", (socket) => {  
    socket.on("Entry room", (meetingRoomId, userId) => {
```

```

socket.join(meetingRoomId);

socket
    .to(meetingRoomId)
    .broadcast.emit("New user has connected", userId);

socket.on("New message", (newMessage, username) => {
    io.to(meetingRoomId).emit("Add new message", newMessage,
    username);
});

});

```

Server sẽ lắng nghe lời nhắn tham gia phòng họp “Entry room” từ phía client để thực hiện tạo kết nối, sau đó server sẽ gửi lời nhắn “New user has connected” để thông báo người dùng mới đã tham gia vào phòng họp cho tất cả các người dùng khác có trong phòng họp biết được. Bên cạnh đó, khi các người dùng trò chuyện và gửi tin nhắn thì server cũng sẽ lắng nghe lời nhắn “New message” từ phía client mỗi khi 1 tin nhắn được gửi để xác nhận và gửi lời nhắn “Add new message” để cho phép client có thể nhìn thấy được tin nhắn.

```

const addStreamingVideo = (video, stream) => {
    video.className = "col-6 mb-2 col-md-4 mb-md-0";
    video.srcObject = stream;
    video.addEventListener("loadedmetadata", () => {video.play();});
    document.getElementById("videoContainer").append(video);
};

```

Tại phía client, hàm tạo khuôn video nhận tham số là video HTML element và luồng stream, khuôn video này sẽ hiển thị responsive dưới dạng col-4 ở màn hình các thiết bị như laptop, tablet và col-6 ở màn hình mobile. Video HTML element sẽ nhận vào đường dẫn src là luồng stream. Video sẽ được play khi có event “loadedmetadata” xảy ra. “loadedmetadata” là event được thực hiện khi metadata được tải lên.

```
let selfStream; let selfVideo = document.createElement("video");
selfVideo.muted = true;
```

Khởi tạo video cá nhân, đồng thời chúng ta cũng phải tắt âm video cá nhân để tránh nghe vọng âm của chính mình trong cuộc gọi.

```
const newUserEntered = (userId, stream) => {
  let peerCall = peer.call(userId, stream);
  let otherUserVideo = document.createElement("video");
  peerCall.on("stream", (otherUserStream) => {
    addStreamingVideo(otherUserVideo, otherUserStream);
  });
};
```

Hàm khởi tạo video của các người dùng khác khi tham gia cuộc gọi, tham số nhận vào là id của người dùng và luồng stream. Hàm sẽ thực hiện “cuộc gọi” (Peer call) đến mọi người dùng trong phòng họp theo luồng stream và gọi hàm tạo khuôn video.

```
navigator.mediaDevices
  .getUserMedia({video: true, audio: true})
  .then((stream) => {
    selfStream = stream;
    addStreamingVideo(selfVideo, stream);
    socket.on("New user has connected", (userId) => {
      newUserEntered(userId, stream);
    });
    peer.on("call", (call) => {
      let video = document.createElement("video");
      call.answer(stream);
      call.on("stream", (userStream) => {
        addStreamingVideo(video, userStream);
      });
    });
});
```

```

    });
});

}).catch((error) => {console.error("Error: ", error);});

```

Tạo giao thức để người dùng có thể cho phép ứng dụng được phép truy cập vào thiết bị camera và micro của mình. Khi truy cập thành công sẽ khởi tạo và hiển thị trong phòng họp video cá nhân của người dùng và nếu có những người dùng khác thì họ sẽ lắng nghe lời nhắn “New user has connected” từ phía server để thực hiện khởi tạo video của họ bên trong phòng họp. Để mọi người trong phòng họp có thể nhìn thấy video của nhau thì họ đều đồng thời phải thực hiện “cuộc gọi” và “trả lời” nhau thông qua giao thức của PeerJS. Bên cạnh việc phát video trực tuyến còn có chức năng chat trực tuyến, nhóm làm cho cả thiết bị di động và PC/Laptop nên có 2 khôi sự kiện sau:

```

inputMessage.addEventListener("keydown", (event) => {
  if (inputMessage.value !== "" && event.keyCode === 13) {
    socket.emit( "New message",inputMessage.value,
      meetingUsername.innerHTML
    ); inputMessage.value = "";
  }
});

```

Kiểm tra và lấy dữ liệu tin nhắn từ form input khi người dùng nhấn “Enter” từ bàn phím, lúc này sẽ gửi lời nhắn “New message” lên server.

```

buttonInputMessage.addEventListener("click", () => {
  if (inputMessage.value !== "") {
    socket.emit("New message",inputMessage.value,
      meetingUsername.innerHTML
    ); inputMessage.value = "";
  }
});

```

Hàm này có thể hỗ trợ cho các thiết bị di động, người dùng chỉ việc nhấn “Icon mũi tên” thì mọi thao tác sẽ được thực hiện tương tự như khi nhấn “Enter” trên PC/Laptop.

```
socket.on("Add new message", (newMessage, username) => {
    let messageElement = document.createElement("li");
    let timeline = new Date().toLocaleDateString() + ", " + new
    Date().toLocaleTimeString();
    messageElement.innerHTML = `<b>${username}</b> - <span
    class="meetingroom__timeline">${timeline}</span><br /> +
    newMessage + "<br /><br />";
    document.getElementById("chatMessages").append(messageElement);
    document.getElementById("chatBox").scrollTop = document.getElementById(
        "chatBox").scrollHeight;
});
```

Phía client lắng nghe lời nhắn “Add new message” từ server để có thể hiển thị tin nhắn bao gồm tên người dùng, thời gian gửi và nội dung lên khung trò chuyện. Nếu lượng tin nhắn nhiều thì khung trò chuyện sẽ tự động kéo xuống.

3.1.2.5. Tắt mở Âm thanh/Video

```
document.getElementById("audioButton").addEventListener("click", () => {
    if (selfStream.getAudioTracks()[0].enabled === true) {
        selfStream.getAudioTracks()[0].enabled = false;
        document.getElementById("audioIcon").classList.remove("fa-volume-up");
        document.getElementById("audioIcon").classList.add("fa-volume-mute");
    }
    else { selfStream.getAudioTracks()[0].enabled = true;
        document.getElementById("audioIcon").classList.remove("fa-volume-mute");
        document.getElementById("audioIcon").classList.add("fa-volume-up");
    }
});
```

Tắt âm thanh cá nhân trong cuộc gọi dựa vào luồng stream cá nhân và thay đổi biểu tượng của nút tắt âm thanh. Có thể bật lại nếu muốn.

```
document.getElementById("videoButton").addEventListener("click", () => {
    if (selfStream.getVideoTracks()[0].enabled === true) {
        selfStream.getVideoTracks()[0].enabled = false;
        document.getElementById("videoIcon").classList.remove("fa-video");
        document.getElementById("videoIcon").classList.add("fa-video-slash");
    }
    else { selfStream.getVideoTracks()[0].enabled = true;
        document.getElementById("videoIcon").classList.remove("fa-video-slash");
        document.getElementById("videoIcon").classList.add("fa-video");
    }
});
```

Tắt video cá nhân trong cuộc gọi dựa vào luồng stream cá nhân và thay đổi biểu tượng của nút tắt video. Có thể bật lại nếu muốn.

3.1.2.6. Chia sẻ phòng họp

```
document.getElementById("shareButton").addEventListener("click", () => {
    let meetingRoomIdToShare =
    document.getElementById("meetingRoomIdToShare");
    meetingRoomIdToShare.select();
    meetingRoomIdToShare.setSelectionRange(0, 5000);
    document.execCommand("copy");
    document.getElementById("shareIcon").classList.remove("fa-copy");
    document.getElementById("shareIcon").classList.add("fa-check-square");
    document.getElementById("shareIcon").classList.remove("text-light");
    document.getElementById("shareIcon").classList.add("text-success");
    alert("You have just copied the room id");
});
```

Thực hiện sao chép mã id của phòng họp đang tham gia khi bấm nút chia sẻ.

3.2. Kết quả



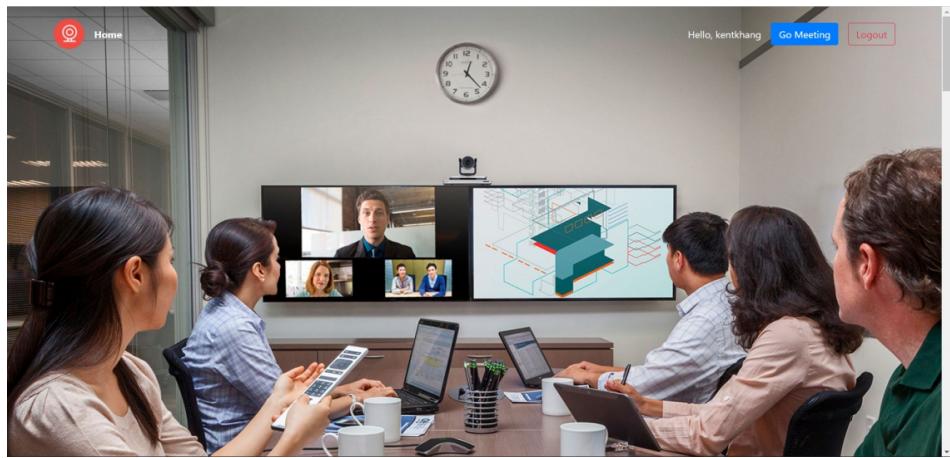
Hình 29. Giao diện trang chủ khi chưa đăng nhập

A screenshot of the login page for a "VIDEO CALL APP". At the top center is a red circular logo with a white camera icon. Below it, the text "VIDEO CALL APP" is written in red. There are two input fields: "Username:" and "Password:", each with a corresponding input box below it. To the right of the password field is a "login" button. Below the input fields is a link "Have no account? Signup now!".

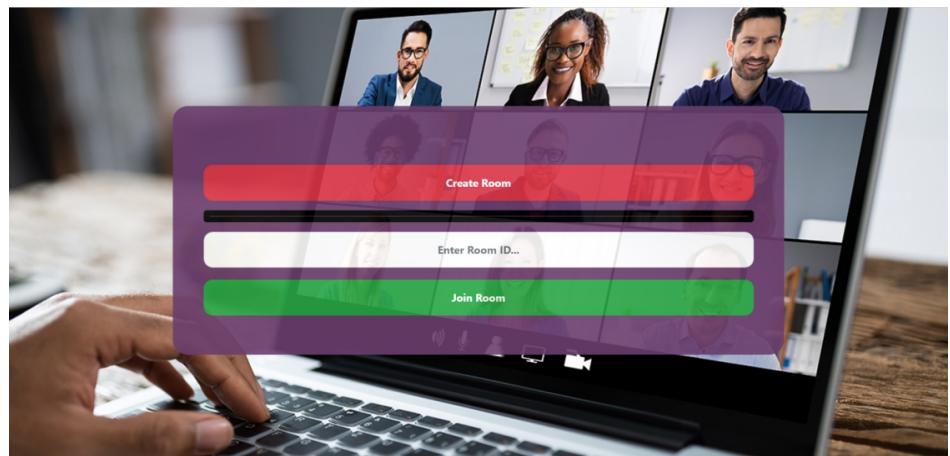
Hình 30. Giao diện trang Đăng nhập

A screenshot of the sign-up page for a "VIDEO CALL APP". At the top center is a red circular logo with a white camera icon. Below it, the text "SIGN UP" is written in red. There are three input fields: "Username:", "Password:", and "Confirm Password:", each with a corresponding input box below it. At the bottom left is a "Back" button, and at the bottom right is a "Signup" button.

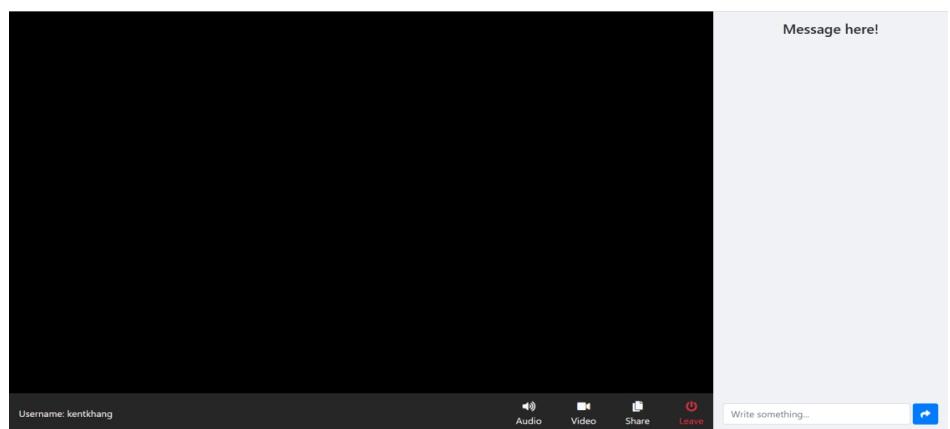
Hình 31. Giao diện trang Đăng ký



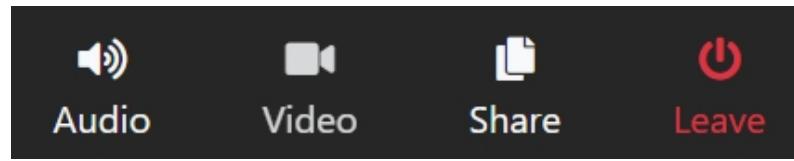
Hình 32. Giao diện trang chủ khi đã đăng nhập



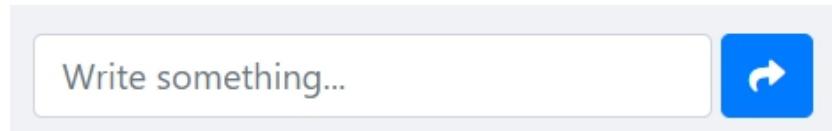
Hình 33. Giao diện trang Chọn phòng



Hình 34. Giao diện trang Phòng họp



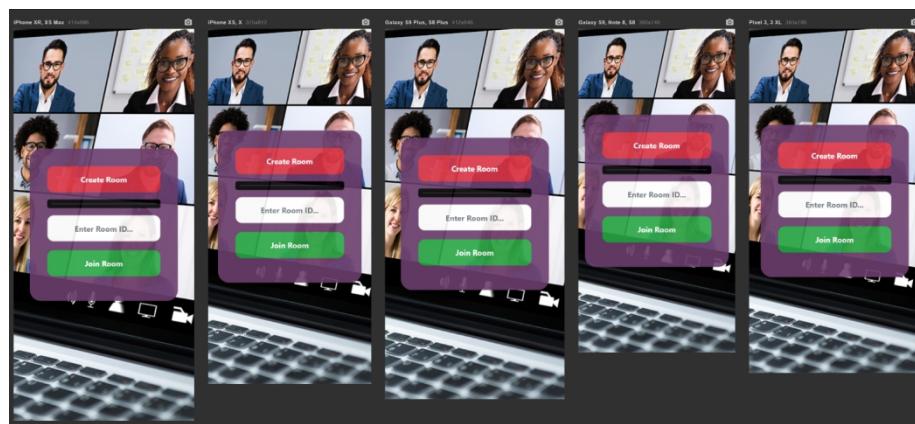
Hình 35. Tổ hợp chức năng có trong phát video trực tuyến



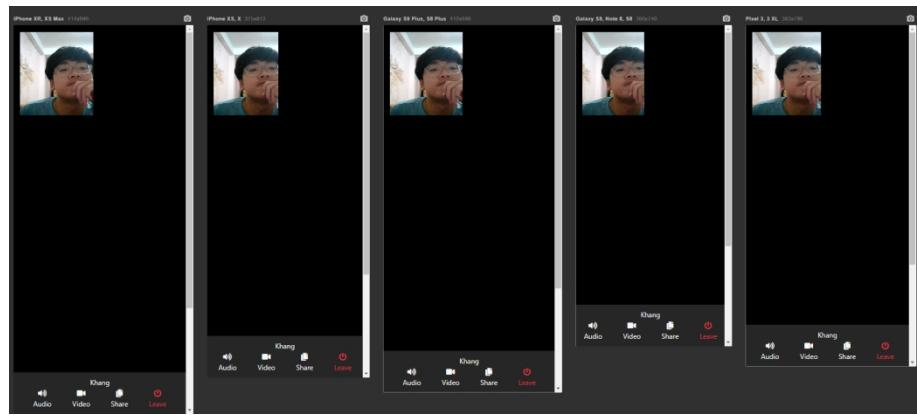
Hình 36. Chức năng trò chuyện



Hình 37. Minh họa video call có 3 người dùng trong phòng họp



Hình 38. Giao diện trang Chọn phòng được responsive trên các thiết bị khác



Hình 39. Giao diện trang Phòng họp được responsive trên các thiết bị khác

Link mã nguồn trên Github:

<https://github.com/K-Kyler/video-meeting-app>

Link sản phẩm đã hosting bằng Heroku: <https://stark-lowlands-76556.herokuapp.com/>

Do nhóm sử dụng Heroku là miễn phí nên khi sử dụng link này lần đầu đôi khi cần đợi một vài giây để server hoạt động trở lại do nó đang trong trạng thái ngủ.

CHƯƠNG IV: TỔNG KẾT

Bài báo cáo này đã phân tích yêu cầu và thiết kế một trang web thời gian thực Web RTC, sử dụng NodeJS phía back-end, với các framework chính như ExpressJS, SocketIO, PeerJS và lưu trữ dữ liệu thông qua MongoDB, các sơ đồ và bảng biểu đã miêu tả chi tiết các chức năng cốt lõi của hệ thống, cách thức hoạt động của từng chức năng và một số ràng buộc quan trọng.

Với đề tài “Web RTC - Tìm hiểu về công nghệ Web RTC và xây dựng một trang web cho phép người dùng gọi video trực tuyến”, nhóm đã học hỏi thêm được rất nhiều thông qua việc tìm hiểu từ các tài liệu từ nhiều nguồn, các chức năng hỗ trợ tiện lợi từ framework có sẵn để góp phần đưa ra ý tưởng cho việc thiết kế ứng dụng này qua cái nhìn tổng thể, bao quát nhất. Việc triển khai trang Web RTC giúp cho việc gọi video và chat trực tuyến trở nên dễ dàng hơn. Tuy nhiên, nếu deploy thực tế cần có server mạnh để việc phản hồi video, audio được nhanh chóng. Ở đây, nhóm còn có sử dụng Heroku và database trực tuyến - Mongo Atlas để hosting lên một web thực tế để nhiều người truy cập nên việc phản hồi video, audio bị delay nhưng chức năng chat vẫn đảm bảo hoạt động tốt. Ứng dụng này về cơ bản đã có giao diện với màu sắc và bố cục sắp xếp các thành phần mà theo cảm nhận của nhóm là phù hợp và thân thiện với người dùng. Đồng thời trang web đã bổ sung đầy đủ về khả năng responsive trên mọi thiết bị.

Nhu cầu sử dụng ứng dụng gọi video và nhắn tin trực tuyến này đang ngày càng được ưa chuộng, việc tìm hiểu, nghiên cứu và triển khai các ứng dụng tương tự như vậy sẽ giúp cho lĩnh vực phần mềm ngày càng phát triển cũng như những mong đợi mà nhóm đang theo đuổi.

PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ THÀNH VIÊN

Phân công công việc

Họ tên	Thời gian	Nội dung công việc
Tô Vĩnh Khang	16/02/2020 - 20/02/2020	Thảo luận nhóm, thống nhất ý tưởng và công nghệ sử dụng, phân công công việc. Tìm hiểu về công nghệ WebRTC. Trình bày nội dung báo cáo Chương I, Chương II (2.1, 2.2.1, 2.2.4, 2.2.5).
	20/02/2020 - 28/02/2020	Thiết kế trang chủ, đăng nhập, đăng ký, chọn phòng. Trình bày nội dung báo cáo Chương III (3.2), Chương IV.
	28/02/2020 - 06/03/2020	Quay video Demo. Tổng hợp tất cả nội dung, kiểm tra, cập nhật mục lục và nộp bài tiểu luận.
Bùi Quang Khải	16/02/2020 - 20/02/2020	Thảo luận nhóm, thống nhất ý tưởng và công nghệ sử dụng và tạo repository trên Git. Tìm hiểu về công nghệ WebRTC. Thiết kế phòng họp mặt.
	20/02/2020 - 28/02/2020	Thực hiện chức năng tạo phòng họp, phát video và trò chuyện trực tuyến, chia sẻ, thoát phòng họp.
	28/02/2020 - 06/03/2020	Trình bày nội dung báo cáo Chương III (3.1).
Hà	16/02/2020 -	Thảo luận nhóm, thống nhất ý tưởng và công nghệ sử

Nguyễn Hải Đăng	20/02/2020	dụng. Tìm hiểu về công nghệ WebRTC.
	20/02/2020 - 28/02/2020	Thực hiện chức năng đăng nhập, đăng ký, đăng xuất.
	28/02/2020 - 06/03/2020	Trình bày nội dung báo cáo Chương II (2.2.2, 2.2.3).

Bảng 2. Phân công công việc

Đánh giá thành viên

Họ tên	Số công việc đã nhận (100%)	Mức độ hoàn thành
Tô Vĩnh Khang	33.33%	Đúng tiến độ đã phân công
Bùi Quang Khải	33.34%	Đúng tiến độ đã phân công
Hà Nguyễn Hải Đăng	33.33%	Đúng tiến độ đã phân công

Bảng 3. Đánh giá thành viên

TÀI LIỆU THAM KHẢO

Tài liệu Tiếng Anh:

- [1] Basarat Ali Syed, [2014], Beginning Node.js, Melbourne, Australia
- [2] Vu D.Hong, [2020], Slide lecture Web Advanced Programming, Ton Duc Thang University, Vietnam
- [3] Rohit Rai, [2013], Socket IO Realtime Web App Development, Mumbai, India.
- [4] J.W. Satzinger, R.B. Jackson, S.D. Burd, [2011], Systems Analysis and Design in a Changing World, 6th edition, Course Technology, Australia.

Tài liệu Tiếng Việt:

- [5] N.M.Vương, N.T.T.Sang, N.T.Sơn, D.T.T.Vân, [2017], Phân tích và thiết kế hệ thống thông tin, Đại học quốc gia Thành phố Hồ Chí Minh, Thành phố Hồ Chí Minh.

Tài liệu Internet:

- [6] <https://www.w3schools.com/nodejs/>
- [7] <https://expressjs.com/en/guide/routing.html>
- [8] <https://www.toptal.com/webrtc/taming-webrtc-with-peerjs>
- [9] <https://socket.io/docs/v3>
- [10] <https://pugjs.org/api/getting-started.html>
- [11] <https://docs.mongodb.com/manual/>
- [12] <https://www.mongodb.com/cloud/atlas>