**LỜI CẢM ƠN**

Được sự hướng dẫn của Ths Kiều Tuấn Dũng và sự đồng ý của Khoa Công nghệ thông tin trường Đại Học Thủy Lợi, em đã được nhận đề tài *“***Xây dựng website livestream tương tác MEREAL***”* cho đồ án tốt nghiệp kết thúc 4,5 năm quá trình học tại trường mình.

Để có được điều này, trước hết cho phép em được gửi lời cảm ơn chân thành đến toàn thể các thầy cô giáo trong trường Đại học Thủy Lợi nói chung và đặc biệt là các thầy cô giáo trong khoa Công nghệ thông tin nói riêng, những người đã tận tình dạy dỗ, truyền đạt cho em những kiến thức, kĩ năng học tập trong suốt thời gian học tập tại trường và tạo mọi điều kiện tốt nhất để em hoàn thành các môn học tại trường cũng như là đồ án tốt ngiệp này. Đặc biệt hơn hết, em xin gửi lời cảm ơn chân thành nhất tới thầy giáo - Ths Kiều Tuấn Dũng - giảng viên khoa Công nghệ thông tin, người đã nhiệt tình chỉ bảo giúp đỡ em rất nhiều trong suốt quá trình hoàn thành đồ án từ lúc phân tích tới thiết kế và hoàn thành đồ án.

Mặc dù nhiều cố gắng để đạt được kết quả tốt cho đề tài, nhằm phản ánh thực lực trong quá trình học tập và rèn luyện tại trường Đại Học Thủy Lợi, song do những hạn chế về kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót. Em rất mong sự góp ý của các Giảng viên để em ngày một cải thiện khả năng hơn trong học tập cũng như trong công việc sau này.

Một lần nữa em xin cảm ơn chân thành tới Ths Kiều Tuấn Dũng cùng các thầy cô giáo trong Khoa đã giúp em trong suốt quá trình hoàn thành đồ án tốt nghiệp này.

Em xin chân thành cảm ơn!

Hà Nội, Tháng 10 năm 2019

Sinh viên

Nguyễn Đức Thắng

# **CHƯƠNG 1: GIỚI THIỆU CHUNG**

# **1.1. Giới thiệu đề tài**

Hiện tại livestream tương tác tặng quà như “Conffeti Việt Nam” đang rộ lên phong trào xem – trả lời câu hỏi – được phần thưởng. Vì vậy đồ án này sẽ xây dựng bộ sản phẩm dành cho website tương tác trực tiếp trên website nhằm phong phú thêm nền tảng. Bên cạnh đó, xây dựng sản phẩm này nhằm đáp ứng cung cấp dịch vụ cho các công ty, doanh nghiệp, cá nhân có nhu cầu quảng cáo sản phẩm cũng như câu hỏi liên quan đến sản phẩm của công ty, doanh nghiệp, cá nhân đó.

# **1.2. Mục tiêu đề tài**

* Livestream bằng công nghệ Web RTC
* Socket realtime phần chat trên kênh
* Tự động trao thưởng cho từng user khi kết thúc tương tác

# **CHƯƠNG 2: KIẾN THỨC CHUNG**

**Các công nghệ được sử dụng trong phát triển website:**

* **WebRTC**
* **NodeJS**
* **MongoDB**
* **Jquery**

## **2.1. WebRTC**

**WebRTC** là Web Real-Time Communication: "và là một web API được phát triển bởi World Wide Web Consortium (W3C), khả năng hỗ trợ trình duyệt (browser) giao tiếp với nhau thông qua VideoCall, VoiceCall hay transfer data "Peer-to-Peer" (P2P) mà không cần browser phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài. 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 gọi điện, tin nhắn hình, truyền dữ liệu bằng các API JavaScript." trích từ **Wikipedia**.

**WebRTC** truyền dữ liệu "Peer-to-Peer" ví dụ dễ hiểu là nếu A muốn gửi thư với B thì bước đầu tiên A và B phải biết địa chỉ nhà của nhau để mà gửi thư. Vậy việc đầu tiên cần làm là phải tạo kết nối "Peer-to-Peer"

Thông thường thì nếu gia đình của A và B đều thoải mái, cho con cháu giao lưu tứ tung thì vô tư đi A cứ gởi thư cho B và B reply cho A. Đời cứ thế mà trôi đi đơn giản như vậy. Nhưng đối với một số gia đình hơi khó khăn tý thì việc tạo kết nối "Peer-to-Peer" giữa A và B sẽ bị cản trở. Nên giống như các hệ thống VoIP, **WebRTC** cũng bị cản trở khi tạo kết nối peer-to-peer bởi tường lửa và NAT. Vậy Firewall(tường lửa) là gì và NAT là gì?

### ***2.1.1. Firewall***

Là biện pháp bảo vệ an toàn cho máy tính của bạn, nó giống như một bức tường ngăn cách máy tính của bạn hoặc mạng cục bộ (local network) với mạng khác như Internet.



Hình 1: Firewall

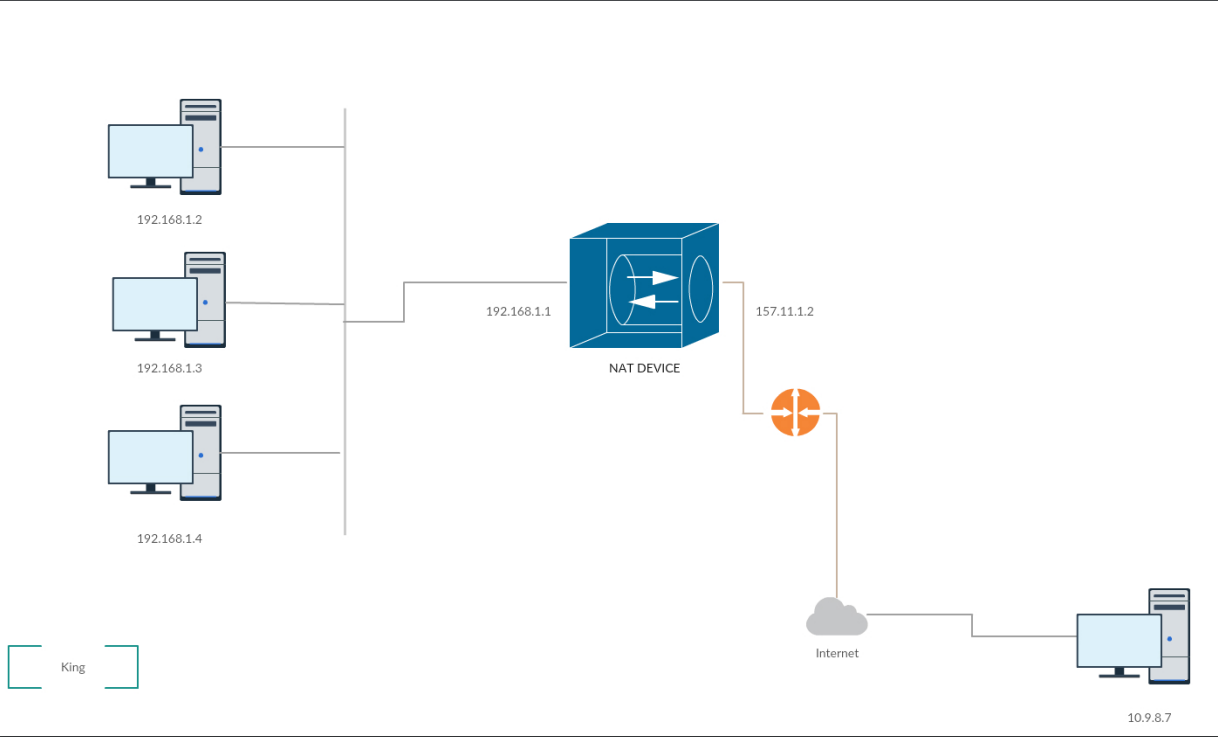
### ***2.1.2. NAT***

Như chúng ta đã biết thì có 2 loại IP là IP public và IP private, các máy trong mạng LAN được đặt IP Private vì mấy IP private k tồn tại ngoài Internet (Vd: 192.168.1.1, 192.168.1.2 etc) . Và IP public thì tất nhiên tồn tại trên Internet rồi vì chùng ta đã đăng ký với ISP(nhà cung cấp dịch vụ Internet). Thông thường NAT thường thay đổi địa chỉ thường là địa chỉ riêng (IP Private) của một kết nối mạng thành địa chỉ công cộng (IP Public).

Vậy khi A send B một gói tin với thông tin tóm lược như : IP nguồn (IP của A) : 192.168.1.1 , IP đích: (113.xxx.y.z)

Mà như chúng ta biết là IP private không tồn tại ngoài Internet nên chúng ta cần NAT để thay thế IP private của A thành IP public.

Hiện tại hiểu đơn giản là "NAT sử dụng IP của chính nó làm IP công cộng 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 gởi 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 NAT computer bởi vì chúng nghĩ rằng NAT computer là máy đã gửi 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 gởi những gói tin nhận được về đúng máy tính đó (client).



Hình 2: NAT

***2.1.3. STUN (Simple Traversal Of UDP Through NAT)***

Tiếp theo là STUN nhé, mấy cái khái niệm này rất quan trọng, nắm chắc thì khi implement rất dễ dàng. STUN thì là khi một máy chủ nào xài NAT (behind NAT) thì STUN server sẽ giúp cho client đó biết được địa chỉ IP và Port mà thiết bị NAT sử dụng. Và từ đó giúp cho các peer có thể lấy được địa chỉ của peer khác (IP nào, cổng mấy, NAT loại gì) để mà vượt.

Nhưng STUN có một nhược điểm là nó không support Symmetric NAT (NAT có nhiều loại), nhưng đừng lo "Không làm được thì TURN, TURN biến hình"

### ***2.1.4. TURN (Traversal Using NAT Relay)***

Cũng giống như STUN tuy nhiên TURN hỗi 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.

Nhưng cái gì cũng có hai mặt. Nhược điểm của TURN là chi phí sử dụng lớn, vì sẽ có một lưu lượng băng thông lớn được sử dụng. Nhất là với chất lượng full HD hay video HD nữa.

### ***2.1.5. 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.

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.

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 (có thể thiết bị đằng sau 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.

Nói dễ hiểu hơn là ICE xài STUN xong không được thì đi xài TURN.

## **2.2. NodeJS**

Node.js là một nền tảng chạy trên môi trường V8 JavaScript runtime - một trình thông dịch JavaScript cực nhanh chạy trên trình duyệt Chrome. Bình thường thì bạn cũng có thể tải bộ V8 và nhúng nó vào bất cứ thứ gì; Node.js làm điều đó đối với các web server. JavaScript suy cho cùng cũng chỉ là một ngôn ngữ - vậy thì không có lý do gì để nói nó không thể sử dụng trên môi trường server tốt như là trong trình duyệt của người dùng được.

Trong một môi trường server điển hình LAMP (Linux-Apache-MySQL-PHP), bạn có một web server là Apache hoặc NGINX nằm dưới, cùng với PHP chạy trên nó. Mỗi một kết nối tới server sẽ sinh ra một thread mới, và điều này khiến ứng dụng nhanh chóng trở nên chậm chạp hoặc quá tải - cách duy nhất để hỗ trợ nhiều người dùng hơn là bằng cách bổ sung thêm nhiều máy chủ. Đơn giản là nó không có khả năng mở rộng tốt. Nhưng với Node.js thì điều này không phải là vấn đề. Không có một máy chủ Apache lắng nghe các kết nối tới và trả về mã trạng thái HTTP - bạn sẽ phải tự quản lý kiến trúc lõi của máy chủ đó. May mắn thay, có một số module giúp thực hiện điều này được dễ dàng hơn, nhưng công việc này vẫn gây cho một chút khó khăn khi mới bắt đầu. Tuy nhiên, kết quả thu được là một ứng dụng web có tốc độ thực thi cao.

Một khái niệm cốt lõi của Node.js đó là các function bất đồng bộ (asynchronous functions) - vì vậy về cơ bản thì mọi thứ chạy trên nền tảng này. Với hầu hết các ngôn ngữ kịch bản máy chủ, chương trình phải đợi mỗi function thực thi xong trước khi có thể tiếp tục chạy tiếp. Với Node.js, việc xác định các function sẽ chạy để hoàn thành một tác vụ nào đó, trong khi phần còn lại của ứng dụng vẫn chạy đồng thời. Nó là một chủ đề phức tạp, nhưng đó là một trong những đặc trưng tiêu biểu của Node.js, vì vậy việc nắm vững nó là điều hết sức quan trọng.

### ***2.2.1. Ưu điểm***

Đầu tiên là ưu điểm về tốc độ thực thi và khả năng mở rộng. Node.js có tốc độ rất nhanh. Đó là một yêu cầu khá quan trọng khi bạn là một startup đang cố gắng tạo ra một sản phẩm lớn và muốn đảm bảo có thể mở rộng nhanh chóng, đáp ứng được một lượng lớn người dùng khi trang web của bạn phát triển lên.

Node.js có thể xử lý hàng ngàn kết nối đồng thời trong khi PHP sẽ chỉ có nước sụp đổ. Bên cạnh các lợi ích về tốc độ thực thi và khả năng mở rộng, có thể bạn cũng đã biết một chút về JavaScript, vì vậy tại sao lại phải phiền toái để học thêm về một ngôn ngữ lập trình hoàn toàn mới như PHP? Và sau đó bạn sẽ có một sự phấn khích khi học về một cái gì đó mới mẻ và gần như chưa được khám phá. Bạn còn nhớ cái cảm giác khi mà một cái gì đó mới xuất hiện và sau đó trở thành phổ biến khắp mọi nơi mà bạn hối tiếc đã không học về nó sớm hơn, và mãi mãi chỉ là người đến sau? Đừng phạm phải sai lầm như vậy lần này nữa. Node.js đang ngày càng trở nên lớn mạnh hơn.

### ***2.2.2. Nhược điểm***

Giống như hầu hết các công nghệ mới, việc triển khai Node.js trên host không phải là điều dễ dàng. Nếu chúng ta có một web hosting xài chung, chúng ta không thể đơn giản tải lên một ứng dụng Node.js và mong chờ nó hoạt động tốt. VPS và dedicated server là một sự lựa chọn tốt hơn – chúng ta có thể cài đặt Node.js trên chúng. Thậm chí dễ hơn là sử dụng một dịch vụ có khả năng mở rộng như là Heroku, và chúng ta có thể hoàn toàn an tâm để phát triển trang web của mình trên đó – chúng ta chỉ cần trả tiền khi cần thêm nhiều tài nguyên hơn.

Mặt khác, chúng ta rất dễ cài đặt Node.js chạy cục bộ trên máy tính của chúng ta sử dụng các hệ điều hành như Windows, Mac hoặc Linux và bắt đầu phát triển ứng dụng ngay lập tức - chỉ việc tải phiên bản Node.js tương ứng. Một điều quan trọng nên chú ý là Node.js không chỉ đơn giản là một sự thay thế cho Apache - các ứng dụng web đang tồn tại sẽ không có khả năng tương thích, và chúng ta sẽ làm việc hiệu quả với những ứng dụng phát triển từ đầu (***mặc dù có rất nhiều framework ngoài kia để giúp đỡ chúng ta với nhiều đặc trưng phổ biến***).

Một nhược điểm lớn khác của Node.js đó là nó vẫn đang trong giai đoạn phát triển ban đầu, điều này có nghĩa là một số đặc trưng sẽ thay đổi trong quá trình phát triển tiếp theo. Trong thực tế, nếu chúng ta đọc các tài liệu đi kèm, thì nó bao gồm một chỉ số ổn định (stability index), chỉ số này cho thấy mức độ rủi ro khi chúng ta sử dụng các đặc trưng hiện có.

## **2.3. MongoDB**

### ***2.3.1. NoSQL***

NoSQL là 1 dạng CSDL mã nguồn mở và được viết tắt bởi: None-Relational SQL hay có nơi thường gọi là Not-Only SQL.

NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value.

NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng,...

Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc .v.v ...

NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng.

NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn, ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable,...

### ***2.3.2. MongoDB là gì?***

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.

MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.

Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là **collection** thay vì bảng.

So với RDBMS thì trong MongoDB **collection** ứng với **table**, còn **document** sẽ ứng với **row** , MongoDB sẽ dùng các document thay cho row trong RDBMS.

Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.

Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB.

### ***2.3.3. Ưu điểm***

Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, nên bạn muốn gì thì cứ insert vào thoải mái.

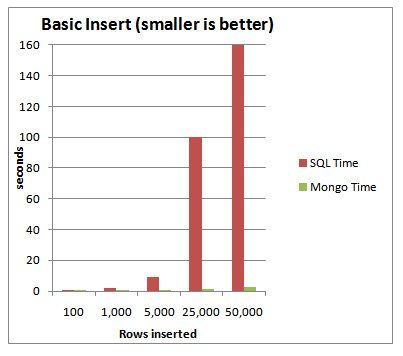
Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.

MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:

Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.



Hình 3: So sánh tốc độ insert giữa MySQL và MongoDB

## ***2.3.4. Nhược điểm***

Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.

Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dữ thừa dữ liệu.

Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điêù này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

# **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ ĐỀ TÀI**

## **3.1 Định nghĩa lại bài toán**

### ***3.1.1. Xác định vai trò người dùng***

Người dùng được chia làm 3 loại: Normal, Agent, Admin.

* Người dùng **Normal**: Có quyền lựa chọn các kênh đang phát sóng để tham gia tương tác trả lời câu hỏi và nhận phần thưởng.
* Người dùng **Agent**: Có quyền lên sóng bắt đầu cho phép người dùng **Normal** tương tác với kênh của mình.
* Người dùng **Admin**: Có quyền quản lý cao nhất về mọi mặt trên hệ thống.

### ***3.1.2 Các yêu cầu ứng dụng đối với người dùng***

* Quản lý câu hỏi, quản lý yêu cầu, quản lý, quản lý quà tặng: Đăng nhập tài khoản **Agent** để thao tác thêm, sửa, xóa.
* Duyệt yêu cầu của **Agent**, quản lý người dùng: Đăng nhập tài khoản **Admin** để thao tác duyệt, khóa tài khoản.
* Người dùng Normal chỉ sử dụng tính năng chat và thao tác những tính năng cơ bản không có quyền thao tác quản lý.

## **3.2 Phân tích và đặc tả yêu cầu**

### ***3.2.1 Đặc tả yêu cầu***

#### **3.2.1.1. Đăng nhập**

|  |  |
| --- | --- |
| Mô tả | Đăng nhập và phân quyền trên hệ thống |
| Đầu vào | Tên đăng nhập và mật khẩu |
| Xử lý | Hệ thống nhận thông tin tài khoản, mật khẩu. Kiểm tra tài khoản có tồn tại trên hệ thống hay không? Nếu tồn tại cho vào hệ thống. Ngược lại, thông báo lỗi đăng nhập cho người dùng |
| Đầu ra | Thành công: Vào sử dụng hệ thống  Thất bại: Thông báo lỗi |

#### **3.2.1.2. Thay đổi thông tin tài khoản**

|  |  |
| --- | --- |
| Mô tả | Tất cả người dùng trên hệ thống đều có quyền thay đổi thông tin tài khảon |
| Đầu vào | Id người dùng, họ tên |
| Xử lý | Hệ thống dựa vào id người dùng để cập nhật họ tên cho tài khoản người dùng |
| Đầu ra | Thành công: Thay đổi thông tin tải khoản thành công  Thất bại: Thông báo lỗi |

#### **3.2.1.3. Quản lý người dùng**

|  |  |
| --- | --- |
| Mô tả | Admin có quyền khóa, kích hoạt người dùng trên hệ thống |
| Đầu vào | Id người dùng và trạng thái cập nhật (lock, active) |
| Xử lý | Hệ thống xác định người tác động lên người dùng là admin hay không? Nếu đúng cho phép thực hiện cập nhật trạng thái người dùng theo trạng thái truyền lên. |
| Đầu ra | Thành công: Cập nhật trạng thái người dùng (lock, active)  Thất bại: Thông báo lỗi |

#### **3.2.1.4. Quản lý yêu cầu**

|  |  |
| --- | --- |
| Mô tả | **Admin** chỉ có quyền xem thông tin yêu cầu, danh sách câu hỏi, duyệt và hủy duyệt yêu cầu.  **Agent** có quyền thêm, sửa, xóa, yêu cầu của chính tài khoản đó. |
| Đầu vào | Thông tin yêu cầu |
| Xử lý | Hệ thống xác định quyền **Admin** sẽ cho phép xem thông tin yêu cầu, danh sách câu hỏi, duyệt và hủy duyệt yêu cầu.  Hệ thống xác định quyền **Agent** sẽ cho phép thêm, sửa, xóa yêu cầu của chính tài khoản đó. Nếu yêu cầu đó đã hoặc đang được phát sẽ không được chỉnh sửa |
| Đầu ra | Thành công: Thông báo thành công các thao tác  Thất bại: Thông báo lỗi |

#### **3.2.1.5. Quản lý câu hỏi**

|  |  |
| --- | --- |
| Mô tả | Chỉ **Agent** có quyền thêm câu hỏi, câu trả lời vào trong các yêu cầu livestream của **Agent** đó |
| Đầu vào | Thông tin câu hỏi, câu trả lời và id yêu cầu |
| Xử lý | Hệ thống xác định quyền **Agent** tại hệ thống cho phép thêm câu hỏi và câu trả lời. |
| Đầu ra | Thành công: Thông báo thành công các thao tác  Thất bại: Thông báo lỗi |

#### **3.2.1.6. Quản lý quà tặng**

|  |  |
| --- | --- |
| Mô tả | Chỉ **Agent** có quyền thêm, sửa, xóa quà tặng. |
| Đầu vào | Thông tin quà tặng |
| Xử lý | Hệ thống xác định quyền **Agent** tại hệ thống cho phép thêm, sửa, xóa quà tặng. |
| Đầu ra | Thành công: Thông báo thành công các thao tác  Thất bại: Thông báo lỗi |

#### **3.2.1.7. Livestream**

|  |  |
| --- | --- |
| Mô tả | **Agent**  livestream và bật câu hỏi để người dùng tương tác |
| Đầu vào | **Agent**  có yêu cầu được duyệt và camera thao tác trên web. |
| Xử lý | Hệ thống xét duyệt **Agent** có yêu cầu thuộc thời gian hiện tại hay không?. Nếu có duyệt camera của **Agent** bắt đầu cho phép **Agent** thao tác livestream trên kênh. |
| Đầu ra | Thành công: Thành công các thao tác  Thất bại: Thông báo lỗi |

#### **3.2.1.8. Đăng xuất**

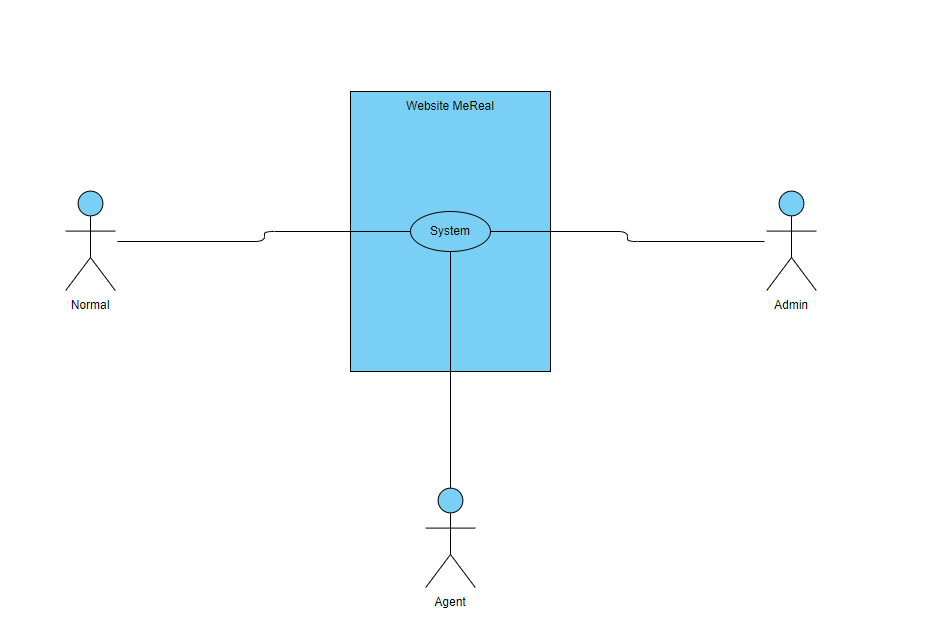
|  |  |
| --- | --- |
| Mô tả | Cho phép người dùng đăng xuất khỏi hệ thống |
| Đầu vào | Người dùng chọn nút đăng xuất |
| Xử lý | Xóa phiên đăng nhập và đăng xuất khỏi hệ thống |
| Đầu ra | Thành công: Thông báo thành công các thao tác và quay lại màn hình đăng nhập  Thất bại: Thông báo lỗi |

#### **3.2.1.9. Quên mật khẩu/Đổi mật khẩu**

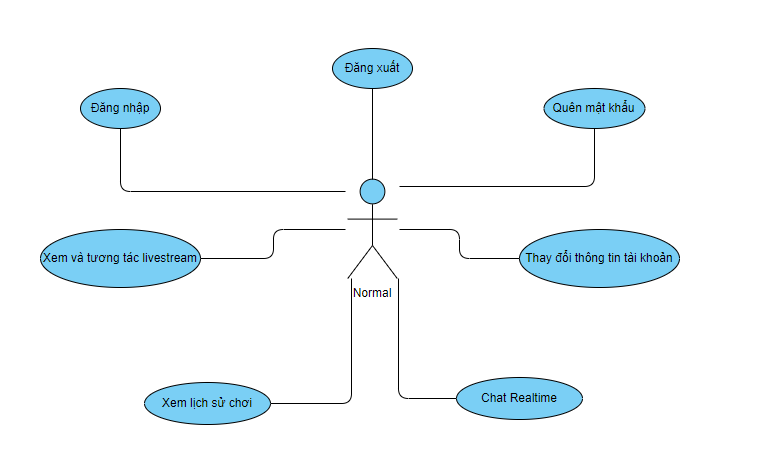
|  |  |
| --- | --- |
| Mô tả | Cho phép người dùng quên mật khẩu và đổi mật khẩu |
| Đầu vào | Email người dùng, tên đăng nhập, mật khẩu mới |
| Xử lý | Quên mật khẩu: Hệ thống sẽ gửi email đường dẫn đổi mật khẩu theo email người dùng truyền lên.  Đổi mật khẩu: Hệ thống sẽ đổi mật khẩu nếu mật khẩu cũ đúng |
| Đầu ra | Thành công: Thông báo thành công các thao tác và quay lại màn hình đăng nhập  Thất bại: Thông báo lỗi |

### ***3.2.2. Phân tích yêu cầu***

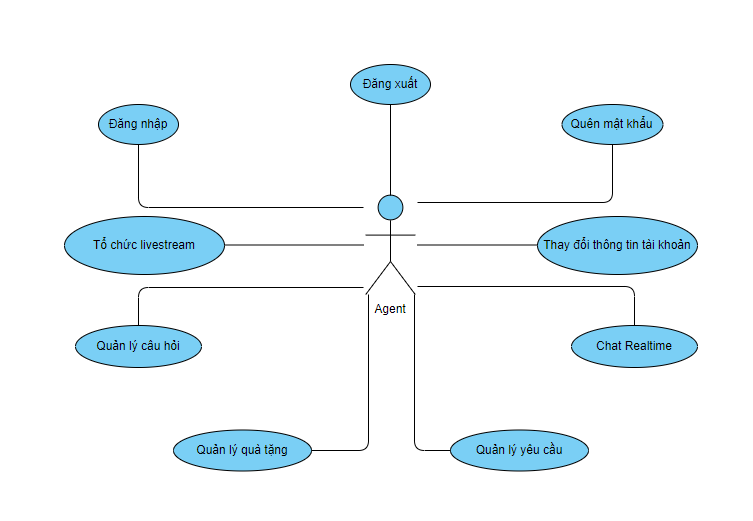
Usercase tổng quát



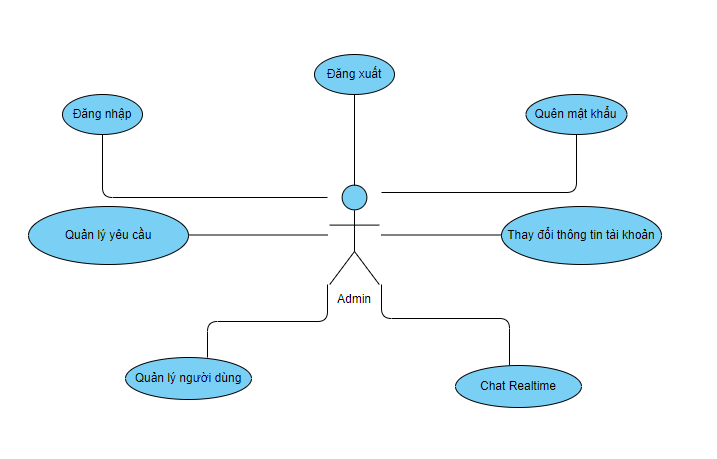
**a.** Phía người dùng **“Normal”**

****

**b.** Phía người dùng **“Agent”**

****

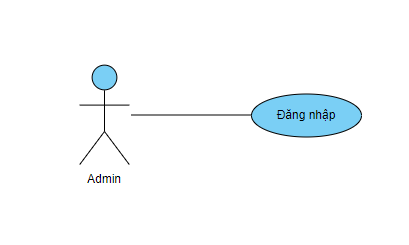
**c.** Phía người dùng **“Admin”**

****

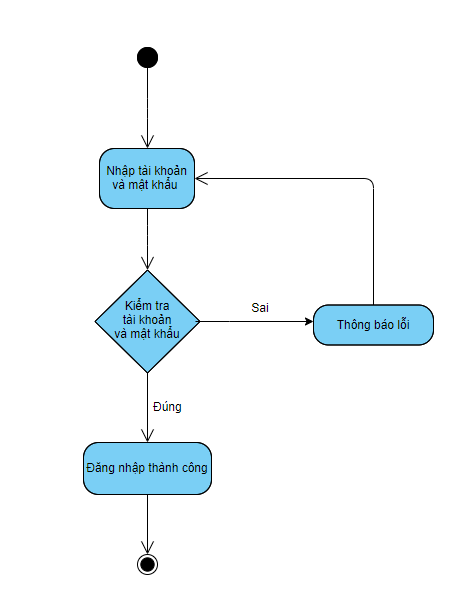
#### **3.2.2.1. Chức năng của người dùng**

**a. Đăng nhập**

Mô tả: Chức năng cho phép người dùng đăng nhập vào hệ thống thông qua tài khoản và mật khẩu.

****

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhập tài khoản và mật khẩu.
  + **Bước 2**: Kiểm tra tài khoản và mật khẩu trên hệ thống. Nếu đúng đến bước 3, nếu sai thông báo lỗi và quay lại bước 1
  + **Bước 3**: Đăng nhập thành công. Hệ thống chuyển màn hình trang chủ
  + **Bước 4**: Kết thúc.

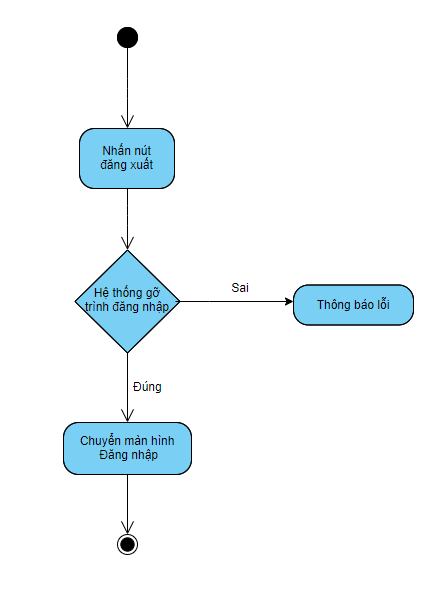
Biểu đồ hoạt động

**b. Đăng xuất**

Mô tả: Hệ thống cho phép người dùng đăng xuất khỏi hệ thống

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Đăng xuất”
  + **Bước 2**: Hệ thống gỡ đăng nhập của người dùng. Nếu trong quá trình gỡ bị lỗi sẽ hiển thị thông báo ngược lại sẽ thực hiện bước 3.
  + **Bước 3**: Hệ thống chuyển màn hình Đăng nhập
  + **Bước 4**: Kết thúc.

Biểu đồ hoạt động

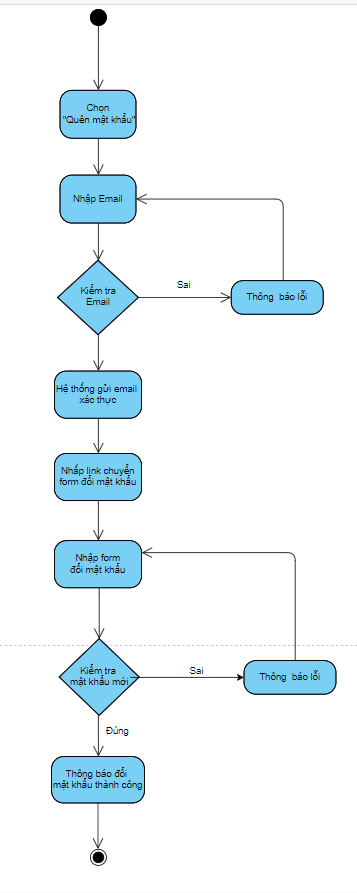


**c. Quên mật khẩu**

Mô tả: Hệ thống cho phép người dùng quên mật khẩu qua email đã đăng ký. Hệ thống gửi một email tới và xác thực chuyển form đổi mật khẩu mới.

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quên mật khẩu”
  + **Bước 2**: Người dùng nhập email đã sử dụng để đăng ký tài khoản.
  + **Bước 3**: Hệ thống kiểm tra email nếu đúng thực hiện bước 4. Nếu sai thông báo lỗi.
  + **Bước 4**: Hệ thống gửi một email xác thực.
  + **Bước 5:** Người dùng nhấp chọn link trong email hệ thống chuyển form đổi mật khẩu mới.
  + **Bước 6:** Người dùng nhập mật khẩu mới tối thiểu 8 ký tự.
  + **Bước 7:** Hệ thống kiểm tra tính hợp lệ của mật khẩu. Nếu đúng thực hiện bước 8. Ngược lại quay lại bước 6.
  + **Bước 8:** Thông báo đổi mật khẩu thành công.
  + **Bước 9:** Kết thúc

Biểu đồ hoạt động

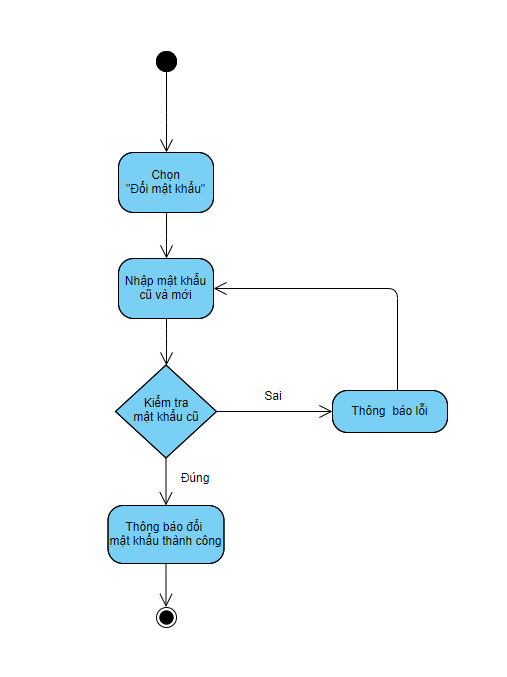


**d. Đổi mật khẩu**

Mô tả: Hệ thống cho phép người dùng đổi mật khẩu.

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Đổi mật khẩu”
  + **Bước 2**: Người dùng nhập mật khẩu cũ và mới.
  + **Bước 3**: Hệ thống kiểm tra cũ nếu đúng thực hiện bước 4. Nếu sai thông báo lỗi.
  + **Bước 4:** Thông báo đổi mật khẩu thành công.
  + **Bước 5:** Kết thúc

Biểu đồ hoạt động

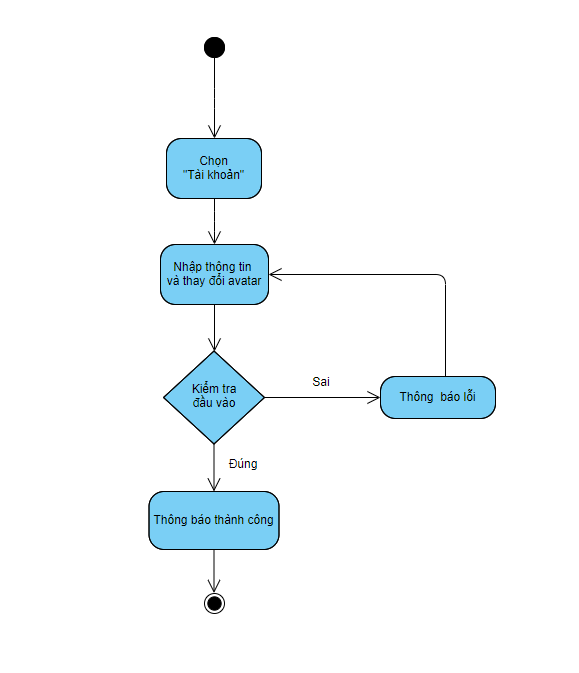
****

**e. Thay đổi thông tin tài khoản**

Mô tả: Hệ thống cho phép người dùng đổi mật khẩu.

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Đổi mật khẩu”
  + **Bước 2**: Người dùng thông tin và thay đổi ảnh đại diện (avatar)
  + **Bước 3**: Hệ thống kiểm tra đầu vào nếu đúng thực hiện bước 4. Nếu sai thông báo lỗi.
  + **Bước 4:** Thông báo thành công.
  + **Bước 5:** Kết thúc

Biểu đồ hoạt động

****

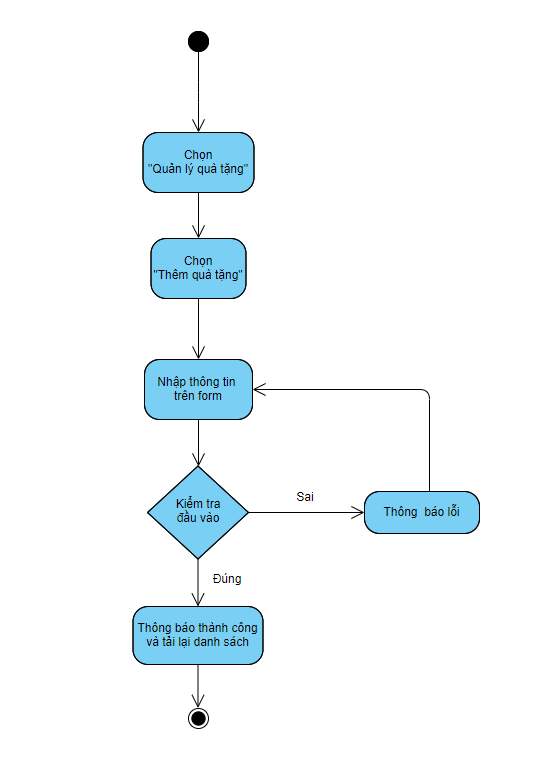
**f. Quản lý quà tặng**

Mô tả: Hệ thống cho phép người dùng **“Agent”** thêm, sửa, xóa quà tặng.

- Thêm quà tặng:

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quản lý quà tặng”.
  + **Bước 2**: Người dùng nhấn nút “Thêm quà tặng”.
  + **Bước 3**: Người dùng nhập thông tin tại form.
  + **Bước 4:** Hệ thống kiểm tra đầu vào. Nếu đúng thực hiện bước 5, ngược lại quay lại bước 3.
  + **Bước 5:** Thông báo thành công và tải lại danh sách quà tặng
  + **Bước 6:** Kết thúc

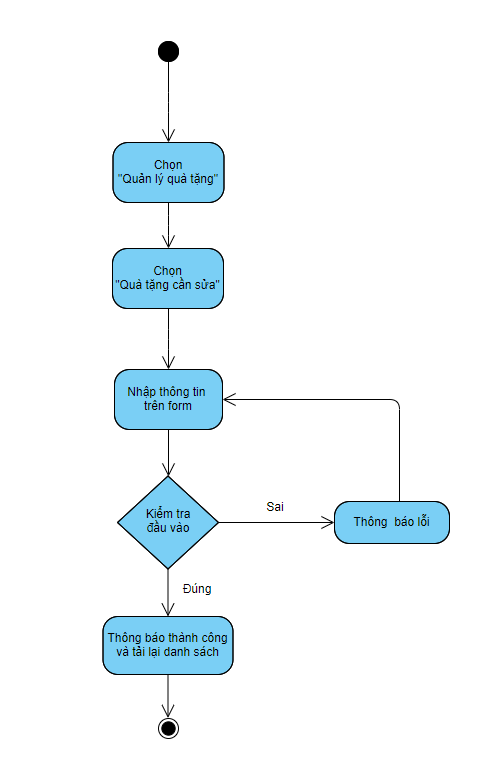
Biểu đồ hoạt động

****

**-** Sửa quà tặng

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quản lý quà tặng”.
  + **Bước 2**: Người dùng chọn quà tặng và nhấn nút “Sửa quà tặng”.
  + **Bước 3**: Người dùng nhập thông tin tại form.
  + **Bước 4:** Hệ thống kiểm tra đầu vào. Nếu đúng thực hiện bước 5, ngược lại quay lại bước 3.
  + **Bước 5:** Thông báo thành công và tải lại danh sách quà tặng
  + **Bước 6:** Kết thúc

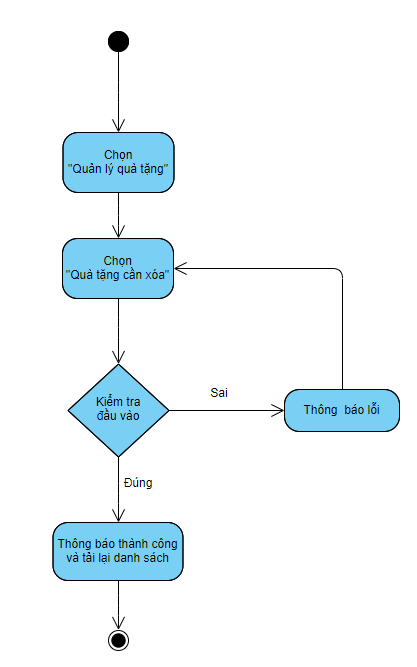
Biểu đồ hoạt động



- Xóa quà tặng

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quản lý quà tặng”.
  + **Bước 2**: Người dùng chọn quà tặng và nhấn nút “Xóa quà tặng”.
  + **Bước 3:** Hệ thống kiểm tra đầu vào. Nếu đúng thực hiện bước 4, ngược lại quay lại bước 2.
  + **Bước 4:** Thông báo thành công và tải lại danh sách quà tặng
  + **Bước 5:** Kết thúc

Biểu đồ hoạt động

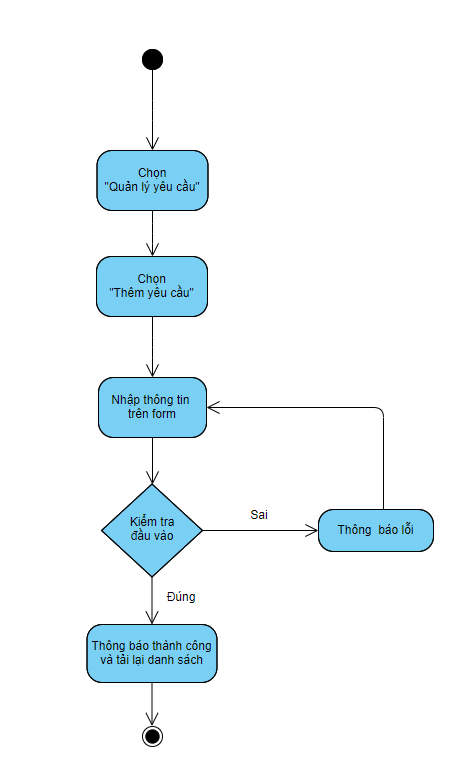


**g. Quản lý yêu cầu**

**-** Thêm yêu cầu:

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quản lý yêu cầu”.
  + **Bước 2**: Người dùng nhấn nút “Thêm yêu cầu”.
  + **Bước 3**: Người dùng nhập thông tin tại form.
  + **Bước 4:** Hệ thống kiểm tra đầu vào. Nếu đúng thực hiện bước 5, ngược lại quay lại bước 3.
  + **Bước 5:** Thông báo thành công và tải lại danh sách quà tặng
  + **Bước 6:** Kết thúc

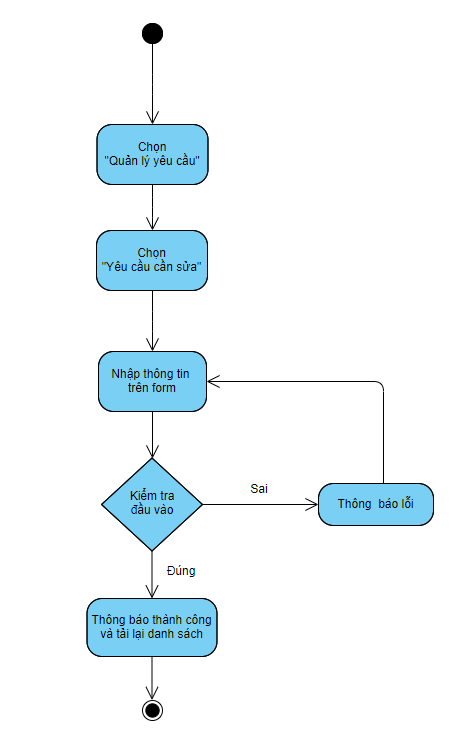
Biểu đồ hoạt động

****

**-** Sửa yêu cầu:

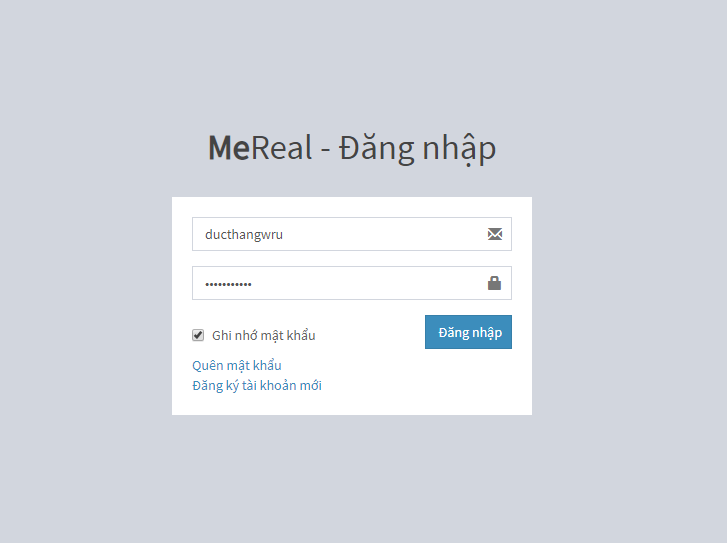
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhấp chọn “Quản lý yêu cầu”.
  + **Bước 2**: Người dùng chọn yêu cầu cần sửa và nhấn nút “sửa yêu cầu”.
  + **Bước 3**: Người dùng nhập thông tin tại form.
  + **Bước 4:** Hệ thống kiểm tra đầu vào. Nếu đúng thực hiện bước 5, ngược lại quay lại bước 3.
  + **Bước 5:** Thông báo thành công và tải lại danh sách quà tặng
  + **Bước 6:** Kết thúc

Biểu đồ hoạt động

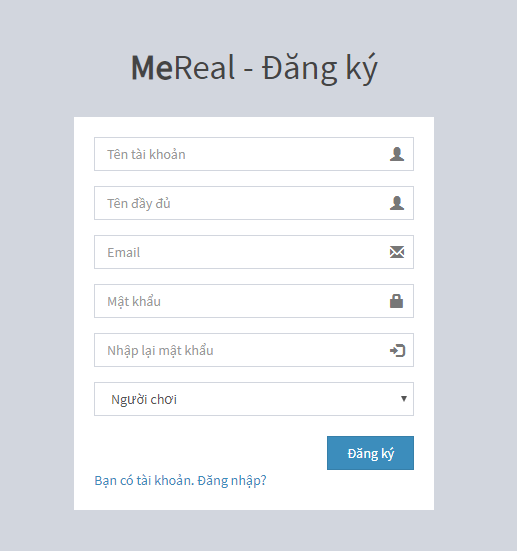


# **CHƯƠNG 5: GIỚI THIỆU GIAO DIỆN ỨNG DỤNG**

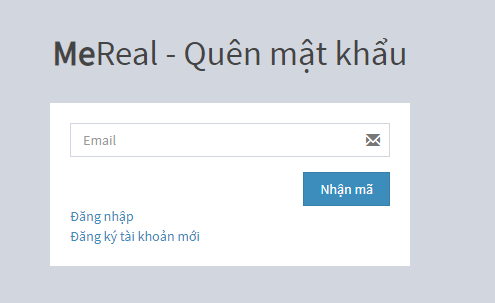
## ***5.1. Màn hình đăng nhập , đăng ký, quên mật khẩu***



Hình 4: Màn hình đăng nhập

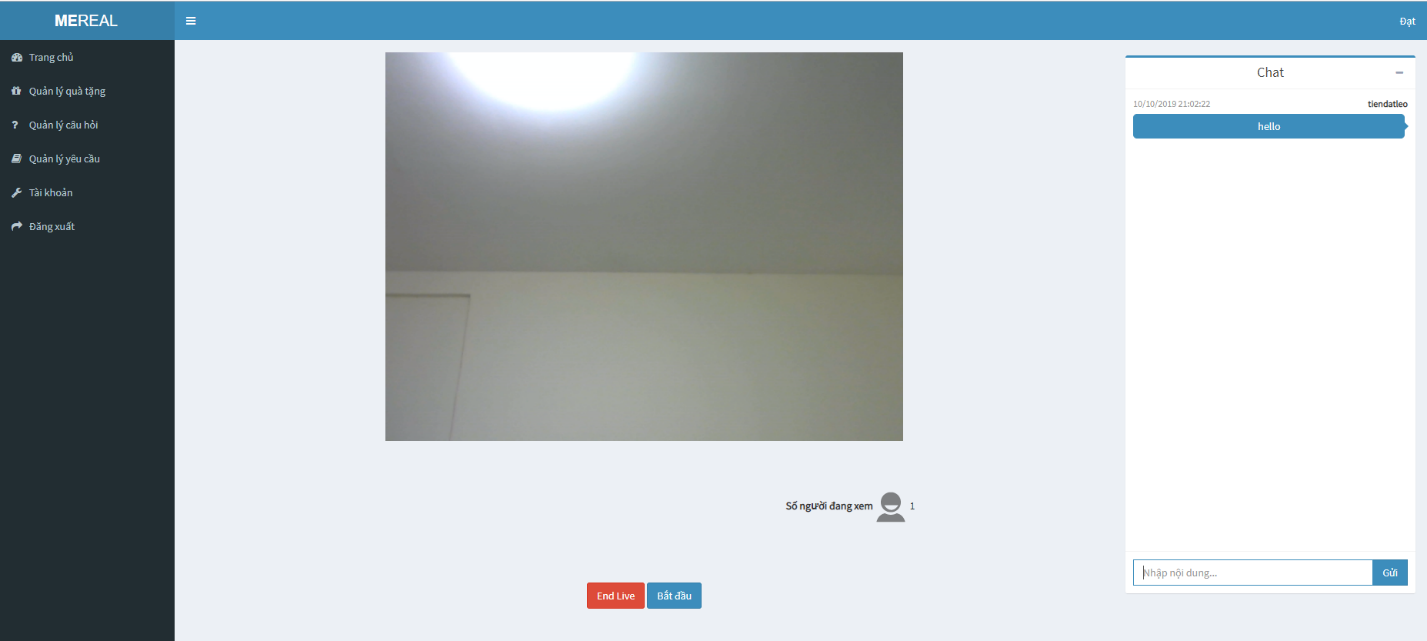


Hình 5: Màn hình đăng ký

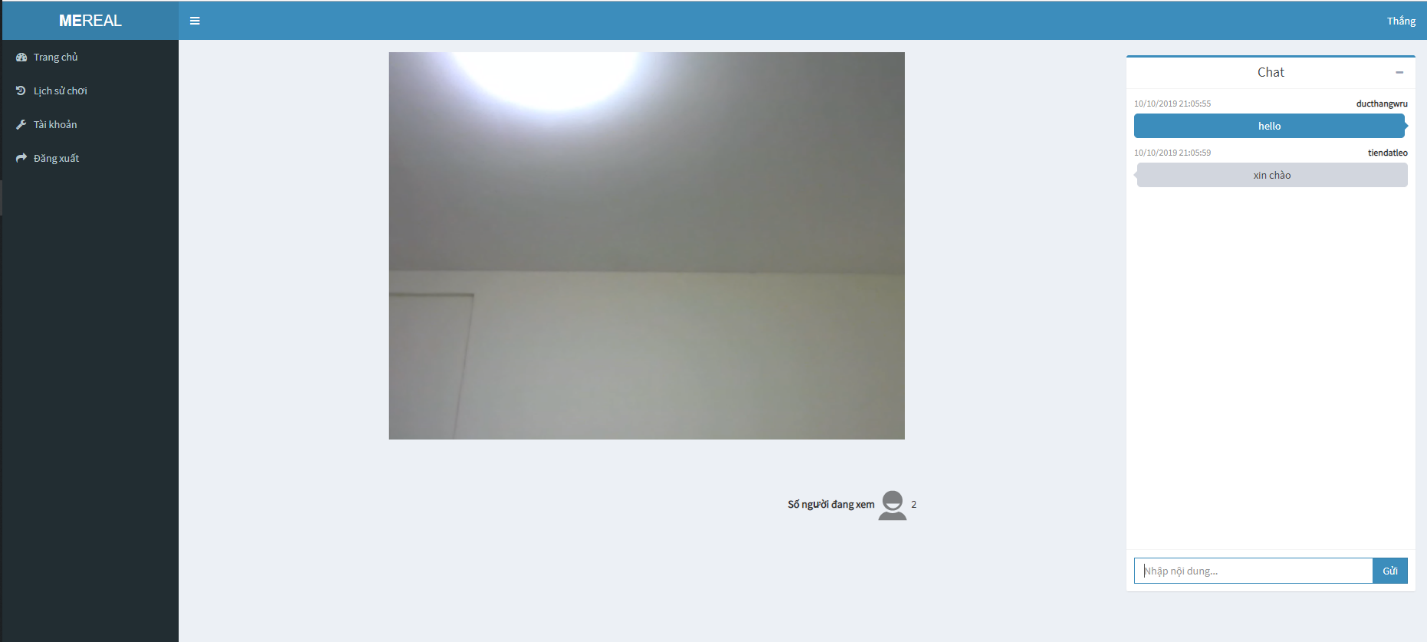


Hình 6: Quên mật khẩu

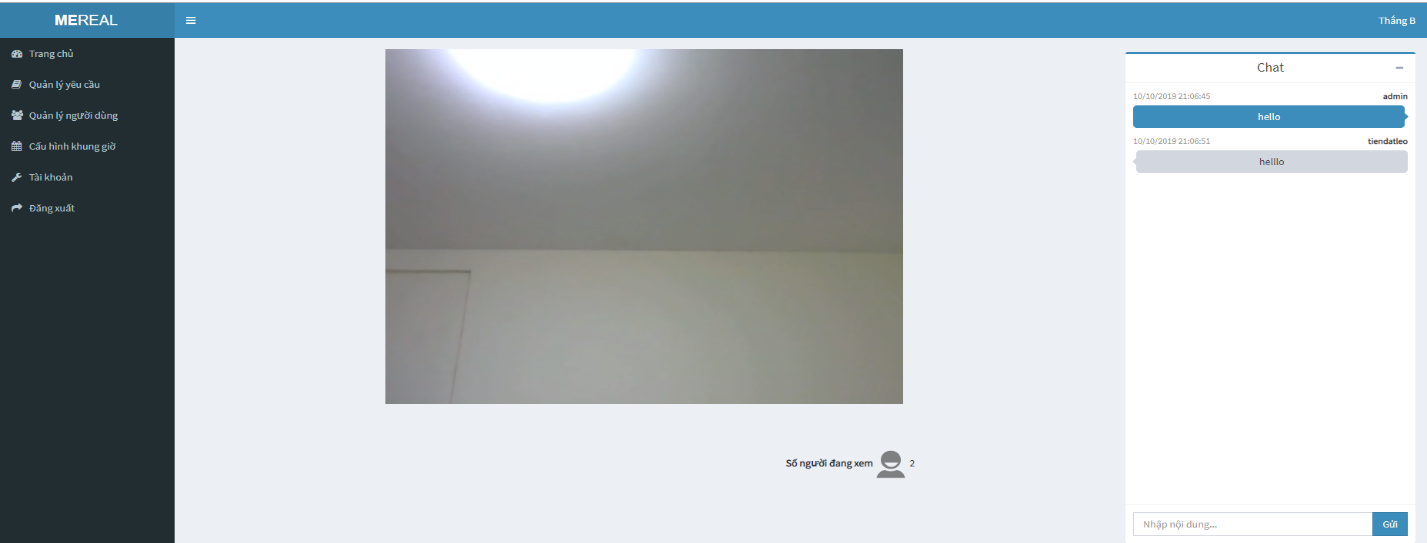
## ***5.2. Màn hình trang chủ***



Hình 7: Màn hình trang chủ dành cho Agent



Hình 8: Màn hình trang chủ dành cho Normal



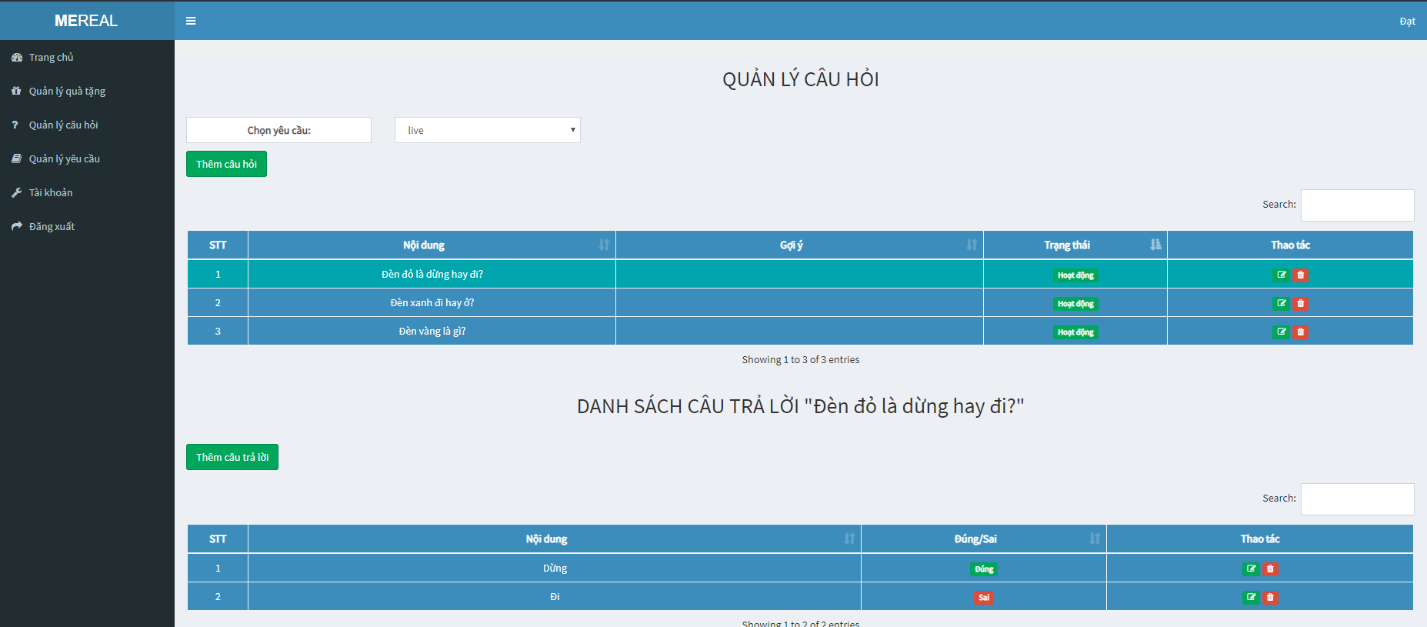
Hình 9: Màn hình trang chủ dành cho Admin

## ***5.3. Màn hình quản lý quà tặng***



Hình 10: màn hình quản lý quà tặng của Agent

## ***5.4. Màn hình quản lý câu hỏi***



Hình 11: Màn hình quản lý câu hỏi dành cho Agent

## ***5.5. Màn hình quản lý yêu cầu***



Hình 12: Màn hình quản lý yêu cầu dành cho Agent



Hình 13: Màn hình quản lý yêu cầu dành cho Admin

## ***5.6. Màn hình quản lý người dùng***



Hình 14: Màn hình quản lý người dùng của Admin

## ***5.7. Màn hình cấu hình khung giờ***



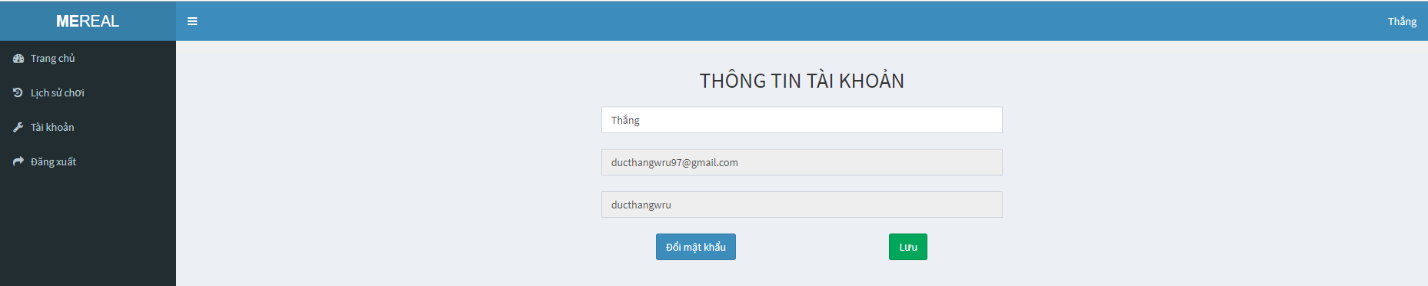
Hình 15: Màn hình xem danh sách cấu hình khung giờ của Admin

## ***5.8. Màn hình xem danh sách lịch sử chơi***



Hình 16: Màn hình xem danh sách lịch sử chơi

## ***5.8. Màn hình thay đổi thông tin tài khoản***



Hình 17: Màn hình cập nhật thông tin tài khoản