**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# ĐỒ ÁN TỐT NGHIỆP

# Hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung trong tổng đài thông minh

**NGUYỄN VIỆT HÙNG**
hung.nv161974@sis.hust.edu.vn

**Ngành Kỹ thuật phần mềm**

**Giảng viên hướng dẫn:** TS. Nguyễn Thị Thu Trang

**Bộ môn:** Công nghệ phần mềm
**Viện:** Công nghệ thông tin – Truyền thông

**HÀ NỘI, 01/2021**

# Lời cam kết

Họ và tên sinh viên: Nguyễn Việt Hùng

Điện thoại liên lạc: (+84) 33 937 4949 Email: hung.nv161974@sis.hust.edu.vn

Lớp: CNTT2.03 – K61 Hệ đào tạo: Chính quy

Tôi – *Nguyễn Việt Hùng* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *TS. Nguyễn Thị Thu Trang*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

*Hà Nội, ngày tháng năm*

Tác giả ĐATN

*Nguyễn Việt Hùng*

# Lời cảm ơn

Lời đầu tiên, em xin gửi lời cảm ơn chân thành nhất tới TS. Nguyễn Thị Thu Trang. Cô đã luôn nhiệt tình, tận tâm hướng dẫn và giúp đỡ em trong suốt quá trình tham gia làm đồ án tốt nghiệp.

Em cũng xin gửi lời cảm ơn tới những người anh chị, bạn bè, thành viên trên Lab 914 và Vbee, đặc biệt là anh Phạm Quang Minh đã luôn trao đổi, chia sẻ kinh nghiệm làm việc với em trong suốt thời gian qua.

Cuối cùng, em xin chân thành cảm ơn gia đình và bạn bè đã luôn giúp đỡ, động viên và tạo điều kiện tốt nhất để em có thể thực hiện đồ án tốt nghiệp này.

Trong quá trình phát triển và hoàn thiện báo cáo cũng như đồ án tốt nghiệp, em khó có thể tránh khỏi những sai sót. Em rất mong thầy cô và bạn đọc góp ý để em có thể hoàn thiện hơn nữa bản phẩm này.

Một lần nữa, em xin chân thành cảm ơn!

# Tóm tắt

Hiện nay, các hệ thống tổng đài thông minh đang dần phát triển và thay thế các tổng đài truyền thống. Tuy nhiên, các hệ thống này còn có nhược điểm là cần tổng đài viên trực liên tục để chăm sóc khách hàng và chưa có công cụ quản lý kịch bản thân thiện, dễ thao tác với người dùng. Chính vì vậy, đồ án này tập trung phát triển một hệ thống quản lý kịch bản cuộc gọi, cung cấp công cụ quản lý kịch bản trực quan, và tích hợp Voice Bot chăm sóc khách hàng tự động trong một số trường hợp.

Trong quá trình phát triển, hệ thống cần sử dụng chung thông tin xác thực và phân quyền với các hệ thống khác, ví dụ như hệ thống tổng đài gọi ra. Giải pháp đưa ra đó là phát triển một hệ thống xác thực và phân quyền tập trung, sử dụng cơ chế xác thực tập trung Central Authentication Service (CAS) và mô hình phân quyền Attribute Based Access Control (ABAC) kết hợp Role Based Access Control (RBAC).

Để dễ dàng bảo trì và mở rộng trong tương lai, hệ thống quản lý kịch bản cuộc gọi và hệ thống xác thực & phân quyền tập trung đã được thiết kế theo mô hình kiến trúc nhiều dịch vụ nhỏ Microservices, sử dụng các công nghệ ReactJS để xây dựng giao diện, NodeJS và MongoDB xây dựng hệ thống và lưu trữ dữ liệu. Hiện tại, cả hai hệ thống đang được triển khai thử nghiệm tại Công ty cổ phần Dịch vụ và Giải pháp dữ liệu Vbee với địa chỉ https://inbound.iristech.club và https://dashboard-inbound.iristech.club. Trong thời gian tới, hệ thống sẽ được phát triển thêm và đưa vào ứng dụng trong lĩnh vực tài chính, ngân hàng.

# Mục lục

**Lời cam kết** ..... **1**

**Lời cảm ơn** ..... **2**

**Tóm tắt** ..... **3**

**Mục lục** ..... **4**

**Danh mục hình vẽ** ..... **7**

**Danh mục bảng** ..... **10**

**Danh mục các từ viết tắt** ..... **12**

**Danh mục thuật ngữ** ..... **13**

**Chương 1 Giới thiệu đề tài** ..... **14**

1.1 Đặt vấn đề ..... 14

1.2 Mục tiêu và phạm vi đề tài ..... 15

1.3 Định hướng giải pháp ..... 16

1.4 Bố cục đồ án ..... 16

**Chương 2 Khảo sát và phân tích yêu cầu** ..... **18**

2.1 Hệ thống quản lý kịch bản cuộc gọi ..... 18

2.1.1 Khảo sát hiện trạng ..... 18

2.1.2 Tổng quan chức năng ..... 19

2.1.3 Đặc tả chức năng ..... 24

2.2 Hệ thống xác thực và phân quyền tập trung ..... 31

2.2.1 Khảo sát hiện trạng ..... 31

2.2.2 Tổng quan chức năng ..... 32

2.2.3 Đặc tả chức năng ..... 36

**Chương 3 Công nghệ sử dụng ..... 41**

3.1 Frontend ..... 41

3.1.1 ReactJS ..... 41

3.1.2 Ant Design ..... 42

3.1.3 Ej2 React Diagram ..... 42

3.2 Backend ..... 44

3.2.1 NodeJS ..... 44

3.2.2 ExpressJS ..... 45

3.2.3 MongoDB ..... 45

3.2.4 Redis ..... 45

3.2.5 RabbitMQ ..... 46

**Chương 4 Phát triển và triển khai ứng dụng ..... 47**

4.1 Thiết kế tổng quan ..... 47

4.1.1 Kiến trúc tổng quan hệ thống ..... 47

4.1.2 Kiến trúc tổng quan phần mềm ..... 48

4.2 Hệ thống quản lý kịch bản cuộc gọi ..... 50

4.2.1 Thiết kế kiến trúc ..... 50

4.2.2 Thiết kế chi tiết ..... 52

4.2.3 Xây dựng ứng dụng ..... 63

4.2.4 Kiểm thử và triển khai ..... 71

4.3 Hệ thống xác thực và phân quyền tập trung ..... 74

4.3.1 Thiết kế kiến trúc ..... 74

4.3.2 Thiết kế chi tiết ..... 76

4.3.3 Xây dựng ứng dụng ..... 82

4.3.4 Kiểm thử và triển khai ..... 87

**Chương 5 Các giải pháp và đóng góp nổi bật ..... 91**

5.1 Phát triển công cụ quản lý kịch bản cuộc gọi ..... 91
5.1.1 Đặt vấn đề ..... 91
5.1.2 Giải pháp ..... 92
5.1.3 Kết quả đạt được ..... 96
5.2 Thiết kế hệ thống theo kiến trúc Microservices ..... 98
5.2.1 Đặt vấn đề ..... 98
5.2.2 Giải pháp ..... 98
5.3 Giải pháp xác thực người dùng tập trung ..... 101
5.3.1 Đặt vấn đề ..... 101
5.3.2 Giải pháp ..... 101
5.3.3 Kết quả đạt được ..... 104
5.4 Giải pháp phân quyền người dùng ..... 105
5.4.1 Đặt vấn đề ..... 105
5.4.2 Giải pháp ..... 107
**Chương 6 Kết luận và hướng phát triển ..... 111**
6.1 Kết luận ..... 111
6.2 Hướng phát triển ..... 111
**Tài liệu tham khảo ..... 113**
**Phụ lục ..... A-1**
A Kịch bản kiểm thử ..... A-1
A.1 Kiểm thử chức năng thêm người dùng vào portal ..... A-1
A.2 Kiểm thử chức năng cập nhật thông tin liên hệ ..... A-1
A.3 Kiểm thử chức năng xóa số hotline ..... A-2

# Danh mục hình vẽ

**Hình 1** Biểu đồ use case tổng quan hệ thống quản lý kịch bản cuộc gọi. ..... 20

**Hình 2** Biểu đồ use case phân rã quản lý kịch bản cuộc gọi. ..... 20

**Hình 3** Biểu đồ use case phân rã quản lý liên hệ. ..... 21

**Hình 4** Biểu đồ use case phân rã quản lý người dùng. ..... 22

**Hình 5** Biểu đồ use case phân rã quản lý số hotline. ..... 22

**Hình 6** Quy trình xử lý cuộc gọi vào. ..... 23

**Hình 7** Biểu đồ use case tổng quan hệ thống xác thực và phân quyền người dùng. ..... 32

**Hình 8** Biểu đồ use case phân rã quản lý tài khoản. ..... 33

**Hình 9** Biểu đồ use case phân rã quản lý portal. ..... 33

**Hình 10** Quy trình xác thực người dùng. ..... 35

**Hình 11** Kiến trúc hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung. ..... 47

**Hình 12** Kiến trúc tổng quan hệ thống. ..... 49

**Hình 13** Thiết kế kiến trúc frontend hệ thống quản lý kịch bản cuộc gọi. ..... 50

**Hình 14** Thiết kế kiến trúc backend dịch vụ quản lý. ..... 51

**Hình 15** Thiết kế kiến trúc backend dịch vụ lưu trữ. ..... 52

**Hình 16** Thiết kế mockup hệ thống quản lý kịch bản cuộc gọi. ..... 52

**Hình 17** Biểu đồ dịch chuyển màn hình hệ thống quản lý kịch bản. ..... 54

**Hình 18** Thiết kế thành phần giao diện màn hình danh sách liên hệ. ..... 54

**Hình 19** Thiết kế thành phần giao diện màn hình quản lý người dùng. ..... 55

**Hình 20** Thiết kế thành phần giao diện màn hình tạo kịch bản cuộc gọi. ..... 56

**Hình 21** Biểu đồ trình tự tạo kịch bản cuộc gọi. ..... 57

**Hình 22** Biểu đồ thực thể liên kết hệ thống quản lý kịch bản cuộc gọi. ..... 60

**Hình 23** Biểu đồ cơ sở dữ liệu hệ thống quản lý kịch bản cuộc gọi. ..... 61

**Hình 24** Giao diện danh sách liên hệ. ..... 65

**Hình 25** Giao diện cập nhật thông tin liên hệ. ..... 66

**Hình 26** Giao diện danh sách người dùng. ..... 66

**Hình 27** Giao diện thêm người dùng vào portal. ..... 67

**Hình 28** Giao diện danh sách kịch bản cuộc gọi. ..... 67

**Hình 29** Giao diện tạo kịch bản cuộc gọi. ..... 68

**Hình 30** Giao diện tạo node mới trong biểu đồ luồng. ..... 69

**Hình 31** Giao diện cập nhật kịch bản. ..... 69

**Hình 32** Giao diện danh sách số hotline. ..... 70

**Hình 33** Giao diện thêm số hotline. ..... 70

**Hình 34** Giao diện sửa số hotline. ..... 71

**Hình 35** Thiết kế kiến trúc frontend hệ thống xác thực và phân quyền tập trung. ..... 74

**Hình 36** Thiết kế kiến trúc backend hệ thống xác thực và phân quyền tập trung. ..... 75

**Hình 37** Thiết kế mockup hệ thống xác thực và phân quyền tập trung. ..... 76

**Hình 38** Biểu đồ dịch chuyển màn hình hệ thống xác thực và phân quyền tập trung. ..... 77

**Hình 39** Thiết kế thành phần giao diện hệ thống xác thực và phân quyền tập trung. ..... 77

**Hình 40** Biểu đồ thực thể liên kết của hệ thống xác thực và phân quyền tập trung. ..... 80

**Hình 41** Biểu đồ cơ sở dữ liệu hệ thống xác thực và phân quyền tập trung. ..... 80

**Hình 42** Giao diện danh sách quyền. ..... 83

**Hình 43** Giao diện cập nhật quyền. ..... 83

**Hình 44** Giao diện danh sách tính năng. ..... 84

**Hình 45** Giao diện cập nhật tính năng. ..... 84

**Hình 46** Giao diện danh sách nhóm tính năng. ..... 85

**Hình 47** Giao diện cập nhật nhóm tính năng. ..... 85

**Hình 48** Giao diện quản lý vai trò. ..... 86

**Hình 49** Giao diện cập nhật vai trò. ..... 86

**Hình 50** Giao diện danh sách portal. ..... 87

**Hình 51** Giao diện tạo portal. ..... 87

**Hình 52** Ví dụ minh họa biểu đồ luồng trực quan hóa kịch bản cuộc gọi. ..... 92

**Hình 53** Thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi theo dạng SQL. ..... 94

**Hình 54** Thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi theo dạng NoSQL. ..... 94

**Hình 55** Công cụ quản lý kịch bản cuộc gọi. ..... 97

**Hình 56** Các hành động trong cấu hình kịch bản cuộc gọi. ..... 97

**Hình 57** Thiết kế hệ thống theo kiến trúc microservices. ..... 100

**Hình 58** Giải pháp xác thực người dùng tập trung [12]. ..... 101

**Hình 59** Biểu đồ trình tự người dùng truy cập ứng dụng lần đầu. ..... 102

**Hình 60** Biểu đồ trình tự người dùng truy cập ứng dụng lần thứ hai. ..... 103

**Hình 61** Biểu đồ trình tự người dùng truy cập ứng dụng liên kết. ..... 104

**Hình 62** Mô hình phân quyền ABAC [13]. ..... 106

**Hình 63** Chức năng thêm quyền hiển thị frontend. ..... 108

**Hình 64** Chức năng thêm quyền truy cập API. ..... 108

**Hình 65** Chức năng thêm mới tính năng. ..... 109

**Hình 66** Chức năng thêm nhóm tính năng. ..... 109

**Hình 67** Chức năng thêm vai trò. ..... 110

# Danh mục bảng

**Bảng 1** Đặc tả use case tạo kịch bản cuộc gọi ..... 24

**Bảng 2** Dữ liệu đầu vào cài đặt thông số kịch bản ..... 25

**Bảng 3** Các trường thông tin của node ..... 26

**Bảng 4** Dữ liệu đầu vào của các thẻ hành động ..... 26

**Bảng 5** Đặc tả use case tải lên tệp danh sách liên hệ ..... 27

**Bảng 6** Dữ liệu đầu vào use case tải lên tệp danh sách liên hệ ..... 28

**Bảng 7** Đặc tả use case quản lý số hotline ..... 28

**Bảng 8** Dữ liệu đầu vào khi thêm / cập nhật số hotline ..... 30

**Bảng 9** Đặc tả use case quản lý tài khoản ..... 36

**Bảng 10** Dữ liệu đầu vào chức năng thêm tài khoản ..... 37

**Bảng 11** Đặc tả use case quản lý portal ..... 38

**Bảng 12** Dữ liệu đầu vào chức năng thêm portal ..... 40

**Bảng 13** Các thuộc tính, phương thức và sự kiện chính của đối tượng DiagramComponent ..... 43

**Bảng 14** Các phương thức chính của đối tượng Connector ..... 43

**Bảng 15** Các thuộc tính chính của đối tượng Node ..... 44

**Bảng 16** Cấu hình chung cho giao diện quản lý kịch bản cuộc gọi ..... 53

**Bảng 17** Danh sách API của hệ thống quản lý kịch bản cuộc gọi ..... 58

**Bảng 18** Danh sách hàng đợi trong hệ thống quản lý kịch bản ..... 60

**Bảng 19** Thiết kế chi tiết cơ sở dữ liệu hệ thống quản lý kịch bản cuộc gọi ..... 62

**Bảng 20** Thư viện và công cụ sử dụng phát triển hệ thống quản lý kịch bản cuộc gọi ..... 63

**Bảng 21** Danh sách test case của hệ thống quản lý kịch bản cuộc gọi ..... 71

**Bảng 22** Thông số cấu hình server triển khai hệ thống quản lý kịch bản cuộc gọi ..... 73

**Bảng 23** Danh sách API của hệ thống xác thực và phân quyền tập trung ..... 78

**Bảng 24** Thiết kế chi tiết cơ sở dữ liệu hệ thống xác thực và phân quyền tập trung..... 81

**Bảng 25** Danh sách các test case của hệ thống xác thực và phân quyền tập trung ..... 88

**Bảng 26** Thông tin cấu hình server triển khai hệ thống xác thực và phân quyền tập trung 90

**Bảng 27** Danh sách thẻ hành động trong kịch bản cuộc gọi ..... 93

**Bảng 28** Thiết kế chi tiết collection Workflow ..... 94

**Bảng 29** Các trường thông tin của đối tượng Node trong Collection Workflow ..... 95

**Bảng 30** Các trường thông tin của đối tượng Action trong Collection Workflow..... 95

**Bảng 31** Ví dụ bảng phân quyền theo mô hình ACL..... 105

# Danh mục các từ viết tắt

| | |
|---|---|
| **API** | Application Programming Interface<br>Giao diện lập trình ứng dụng |
| **CNTT** | Công nghệ thông tin |
| **ĐATN** | Đồ án tốt nghiệp |
| **CSKH** | Chăm sóc khách hàng |
| **TTS** | Text to speech |
| **RBAC** | Role Based Access Control |
| **ABAC** | Attribute Based Access Control |

# Danh mục thuật ngữ

| | |
|---|---|
| **Portal** | Đơn vị tổng đài mà người dùng quản lý |
| **Service** | Dịch vụ |
| **Node** | Nút |
| **Workflow** | Kịch bản |

# Chương 1 Giới thiệu đề tài

Chương 1 đưa ra những vấn đề thực tế dẫn tới lý do chọn đề tài, tổng quan về hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung. Sau đó đưa ra mục tiêu và phạm vi của đề tài, định hướng giải pháp và bố cục trình bày của đồ án.

## 1.1 Đặt vấn đề

Chăm sóc khách hàng (CSKH) là vấn đề quan trọng thiết yếu của doanh nghiệp, quyết định đến sự tồn tại và phát triển của doanh nghiệp. Việc CSKH tốt giúp duy trì lượng khách hàng hiện tại, tạo sự hài lòng, tin tưởng, từ đó tạo ra những khách hàng trung thành. Ngoài ra, CSKH tốt còn giúp thu hút thêm khách hàng mới, giảm thiểu chi phí quảng cáo, tiếp thị. Vì vậy có thể nói CSKH là vũ khí cạnh tranh vô hình mà rất lợi hại.

Để thực hiện CSKH, doanh nghiệp thường sử dụng hệ thống tổng đài truyền thống. Tổng đài viên phải trực tổng đài khoảng 9 – 10 tiếng một ngày. Phòng trực tổng đài khá nhỏ và ồn ào. Tổng đài viên thường xuyên bị căng thẳng, mệt mỏi dẫn đến chất lượng phục vụ khách hàng chưa cao. Bên cạnh đó, tổng đài truyền thống này có chi phí tương đối cao.

Hệ thống tổng đài của doanh nghiệp được chia thành hai bộ phận chính yếu nhất đó là trung tâm gọi ra (Outbound Call Center) và trung tâm tiếp nhận cuộc gọi đến (Inbound Call Center). Trung tâm gọi ra là bộ phận chuyên để thực hiện các cuộc gọi ra từ doanh nghiệp đến khách hàng. Doanh nghiệp sử dụng các cuộc gọi ra với mục tiêu giới thiệu sản phẩm, dịch vụ mới, nhắc nhở thanh toán, điều tra, khảo sát ý kiến khách hàng. Trung tâm gọi vào là bộ phận tiếp nhận, xử lý những cuộc gọi đến từ khách hàng. Nhiệm vụ chính của trung tâm này là tư vấn sản phẩm, hỗ trợ kỹ thuật, thanh toán, xử lý đơn hàng hoặc tiếp nhận các yêu cầu nâng cấp và gia hạn dịch vụ.

Hiện nay đã có nhiều hệ thống tổng đài thông minh ra đời với mục đích thay thế tổng đài truyền thống. Tổng đài thông minh có thể dễ dàng lắp đặt và bảo trì. Doanh nghiệp chỉ cần máy tính, điện thoại kết nối internet và không cần thuê nhân viên chuyên về kỹ thuật để quản lý xuyên suốt tổng đài. Tổng đài thông minh giúp nâng cao hiệu suất công việc và quản lý tốt các nhân viên. Tổng đài thông minh còn giúp giảm thiểu chi phí nhân sự, chi phí cước gọi và có thể lựa chọn các đầu số hotline đa dạng.

Hệ thống tổng đài thông minh mới được phát triển ở những năm gần đây nên vẫn còn nhiều hạn chế. Đầu tiên là đa phần các tổng đài gọi vào, tổng đài viên đều phải túc trực như tổng đài truyền thống. Điều này có nhược điểm giống tổng đài truyền thống là hiệu suất chưa cao và tốn kém chi phí. Tiếp theo là các tổng đài thông minh đều chưa có công cụ quản lý kịch bản gọi chuyên nghiệp, trực quan, gây khó khăn trong quá trình thao tác của tổng đài viên.

Do đó, trong khuôn khổ đồ án, em đã phát triển một hệ thống quản lý cuộc gọi cho tổng đài thông minh để giải quyết các vấn đề nêu trên. Hệ thống được tích hợp voice Bot có thể tự động trả lời các cuộc gọi vào và hệ thống cung cấp một công cụ giúp trực quan hóa quá trình xây dựng kịch bản cuộc gọi.

Bên cạnh đó, trong quá trình phát triển tổng đài thông minh, có nhiều hệ thống có nhu cầu liên kết để xác thực và phân quyền tập trung như hệ thống gọi ra, hệ thống gọi vào. Chính vì thế, trong phạm vi đồ án này, em đã xây dựng một hệ thống xác thực và phân quyền tập trung cho các hệ thống khác nhau trong giải pháp tổng đài thông minh.

## 1.2 Mục tiêu và phạm vi đề tài

Để giải quyết một số vấn đề đã được nêu ra ở phần 1.1, án tốt nghiệp của em gồm hai công việc chính:

Thứ nhất, phát triển một công cụ trực quan giúp người dùng có thể dễ dàng tạo ra các kịch bản nâng cao để điều khiển luồng cho cuộc gọi vào, sử dụng công nghệ chuyển văn bản thành giọng nói (Text To Speech – TTS) để cá nhân hóa nội dung trao đổi với khách hàng, kết hợp với dùng Voice Bot giúp tiết kiệm nguồn nhân lực, nâng cao trải nghiệm người dùng.

Thứ hai, phát triển hệ thống xác thực và phân quyền tập trung, giúp (i) giảm thao tác đăng nhập trên các ứng dụng liên kết với nhau và (ii) người quản trị có thể dễ dàng quản lý và phân quyền các thành viên trong hệ thống.

Hệ thống hướng tới các đối tượng là các ngân hàng, bệnh viện, cửa hàng, dịch vụ giáo dục, bất động sản,… Phạm vi của đề tài này được giới hạn trong các chức năng của người dùng thông thường đó là:

- Trong hệ thống quản lý kịch bản cuộc gọi, người dùng thiết lập các kịch bản cuộc gọi với nội dung phù hợp. Khi khách hàng gọi đến số hotline của tổng đài, tổng đài sẽ tự động phục vụ khách hàng như nội dung đã được thiết lập sẵn.
- Trong hệ thống xác thực và phân quyền tập trung, người dùng có thể đăng nhập và sử dụng các ứng dụng liên kết với nhau. Quản trị viên có thể quản lý các tài khoản, phân quyền người dùng trong hệ thống.

## 1.3 Định hướng giải pháp

Hiện nay khi phát triển ứng dụng, có hai hướng tiếp cận chính đó là phát triển ứng dụng mobile và phát triển ứng dụng web. Để đáp ứng được nhu cầu cần cấu hình các kịch bản cuộc gọi khá chuyên sâu và phức tạp, em chọn phát triển hệ thống trên nền tảng web, theo mô hình Client-Server. Hệ thống sẽ được chia thành hai thành phần tách biệt backend và frontend, chúng giao tiếp với nhau qua REST API.

Về phía backend, có hai kiến trúc phổ biến được áp dụng đó là kiến trúc Monolithic và kiến trúc Microservices. Monolithic là kiến trúc phần mềm dạng nguyên khối, tất cả các tính năng của ứng dụng đều được gom vào một khối. Kiến trúc này giúp phát triển ứng dụng đơn giản, dễ dàng triển khai và phù hợp với khá nhiều dự án. Tuy nhiên kiến trúc này sẽ gặp khó khăn trong việc bảo trì, tính linh hoạt và khả năng mở rộng kém khi số lượng người dùng tăng, có thêm nhiều yêu cầu mới, giao tiếp với các hệ thống khác tăng. Do đó, em lựa chọn mô hình kiến trúc nhiều dịch vụ nhỏ Microservices để xây dựng hệ thống. Ứng dụng được chia ra thành nhiều service đảm nhiệm các chức năng riêng biệt và được xây dựng và triển khai độc lập với nhau. Các service này được xây dựng trên nền tảng NodeJS và sử dụng MongoDB để lưu trữ dữ liệu.

Về phía frontend, em sẽ xây dựng trang web theo hướng Single Page Application (SPA) giúp tăng hiệu năng của trang web. Toàn bộ nội dung của trang web được tải ở lần đầu tiên truy cập. Ở những lần chuyển trang sau, trang web chỉ tải lại những nội dung cần thiết. Do đó, thời gian tải lại trang giảm, tiết kiệm băng thông hơn và nâng cao trải nghiệm người dùng. Để xây dựng được một trang web như vậy, em lựa chọn sử dụng thư viện ReactJS vì ReactJS có thể phân chia trang web thành các component nhỏ để tái sử dụng và mở rộng sau này. Với yêu cầu xây dựng một công cụ thiết lập kịch bản cuộc gọi, em sử dụng thư viện Ej2 React Diagrams. Thư viện này cung cấp rất nhiều tiện ích để xây dựng một biểu đồ luồng giúp trực quan hóa các kịch bản gọi mà người dùng tạo.

Để quản lý chung thông tin người dùng trên các ứng dụng liên kết, hệ thống áp dụng cơ chế xác thực tập trung CAS. Ngoài ra, hệ thống sử dụng mô hình phân quyền RBAC kết hợp với mô hình ABAC cho phép quản trị viên có thể dễ dàng quản lý quyền truy cập của người dùng trong hệ thống.

## 1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Trong Chương 2, em sẽ khảo sát và phân tích yêu cầu, nêu tổng quan chức năng, đưa ra một số quy trình và đặc tả một số chức năng nổi bật cho hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung.

Từ các yêu cầu đã nêu ra, trong Chương 3 em sẽ đưa các các công nghệ chính để có thể phát triển và xây dựng hệ thống. Phía frontend sử dụng các công nghệ ReactJS, Ant Design và Ej2 React Diagram. Phía backend sử dụng NodeJS, ExpressJS, MongoDB, Redis và RabbitMQ.

Trong Chương 4 em trình bày chi tiết về thiết kế kiến trúc, thiết kế cơ sở dữ liệu, quá trình xây dựng ứng dụng, kiểm thử và triển khai hệ thống.

Chương 5 trình bày về những đóng góp nổi bật, những giải pháp tâm đắc mà em đưa ra trong quá trình phát triển hệ thống như công cụ quản lý kịch bản cuộc gọi, thiết kế kiến trúc hệ thống, phát triển hệ thống xác thực và phân quyền người dùng.

Cuối cùng, trong Chương 6 em tổng kết về các đóng góp chính của đồ và hướng phát triển của hệ thống trong tương lai.

# Chương 2 Khảo sát và phân tích yêu cầu

Trong Chương 1, em đã giới thiệu tổng quan về đề tài này. Tiếp theo đó, trong Chương 2 em trình bày về việc khảo sát các phần mềm liên quan và phân tích chi tiết yêu cầu của hệ thống quản lý kịch bản cuộc gọi trong mục 2.1 và hệ thống xác thực và phân quyền tập trung trong mục 2.2.

## 2.1 Hệ thống quản lý kịch bản cuộc gọi

### 2.1.1 Khảo sát hiện trạng

Hiện nay, trên thị trường đã có nhiều hệ thống tổng đài thông minh hỗ trợ cho việc chăm sóc khách hàng, nổi bật nhất là 3 ứng dụng Omicall [1], StringeeX [2] và mAiCallCenter [3]. Sau đây em xin phân tích các ưu nhược điểm của 3 ứng dụng trên.

Ứng dụng OmiCall có các chức năng cơ bản như thực hiện cuộc gọi ra, nghe cuộc gọi vào, quản lý khách hàng, quản lý nhân viên, quản lý nhóm nhân viên, báo cáo thống kê. Ngoài ra, ứng dụng có hỗ trợ người dùng tạo kịch bản cuộc gọi đơn giản: phát tệp âm thanh, chuyển tương tác nhấn phím, chuyển tiếp cuộc gọi. Ứng dụng này sẽ phù hợp với các doanh nghiệp nhỏ, chỉ cần hệ thống chăm sóc khách hàng đơn giản.

Ứng dụng StringeeX đã có các tính năng phát triển hơn so với ứng dụng OmiCall. Ngoài các tính năng đã nêu trên, StringeeX còn cho phép cấu hình kịch bản cuộc gọi theo các phím bấm, với mỗi một phím bấm sẽ có một hành động tương ứng: gửi voice mail, chuyển đến một hàng đợi, chuyển đến một số điện thoại, chuyển đến một phím khác, gửi tin nhắn đến người gọi hoặc dừng cuộc gọi. Tuy nhiên, cách cấu hình này khá đơn giản, chưa tùy biến sâu được kịch bản cuộc gọi và giao diện khó theo dõi đối với người dùng.

Ứng dụng mAiCallCenter có tính năng vượt trội hơn so với OmiCall và StringeeX. Ứng dụng cho phép người dùng có thể cấu hình kịch bản cuộc gọi theo các phím bấm được phân cấp cha con, đáp ứng được các nội dung cuộc gọi có tính phức tạp cao hơn. Ngoài ra, nội dung cuộc gọi có thể được cá nhân hóa nếu thông tin khách hàng được lưu trong hệ thống. Về giao diện hiển thị, ứng dụng vẫn chưa cung cấp cho người dùng cái nhìn trực quan về kịch bản cuộc gọi, cùng với đó là việc sử dụng Server Side Rendering làm giảm trải nghiệm của người dùng.

Cả ba ứng dụng OmiCall, StringeeX và mAiCallCenter đều chưa cung cấp được một công cụ trực quan để giúp người dùng dễ dàng quản lý kịch bản cuộc gọi. Người dùng khó có thể quan sát luồng cuộc gọi, cấu hình nâng cao cho cuộc gọi ví dụ như: chuyển cuộc gọi cho tổng đài viên khác, chuyển cuộc gọi sang kịch bản khác, phát nội dung tệp âm thanh hoặc thực hiện nhiều hành động một lúc.

Ngoài ra, tổng đài chăm sóc khách hàng luôn phải có tổng đài viên trực để chăm sóc khách hàng khi khách hàng gọi điện đến số hotline của tổng đài. Việc này gây tốn kém chi phí mà mang lại hiệu quả chưa cao. Tổng đài viên phải làm việc 9 – 10 tiếng một ngày, bị áp lực, mệt mỏi làm giảm chất lượng phục vụ khách hàng.

Từ những phân tích ở trên, em sẽ phát triển hệ thống quản lý kịch bản cuộc gọi vào cung cấp một công cụ giúp người dùng cấu hình kịch bản cuộc gọi vào qua các biểu đồ luồng. Người dùng còn có thể cấu hình chuyên sâu cho kịch bản cuộc gọi đến: (i) sử dụng công nghệ chuyển văn bản thành giọng nói (Text To Speech - TTS) với các giọng nói từ nhiều vùng miền khác nhau, (ii) kết nối đến các Voice Bot khác nhau để phục vụ các mục đích chăm sóc khách hàng khác nhau, (iii) tải lên các tệp thu âm có sẵn, (iv) kết nối đến các tổng đài viên, (v) chuyển đến một kịch bản khác, (vi) nghe lại nội dung đã phát hoặc (vii) quay lại bước lựa chọn trước. Từ đó, hệ thống giúp doanh nghiệp có thể giảm được số lượng tổng đài viên trực tổng đài, giúp giảm thiểu chi phí và nâng cao hiệu quả chăm sóc khách hàng.

### 2.1.2 Tổng quan chức năng

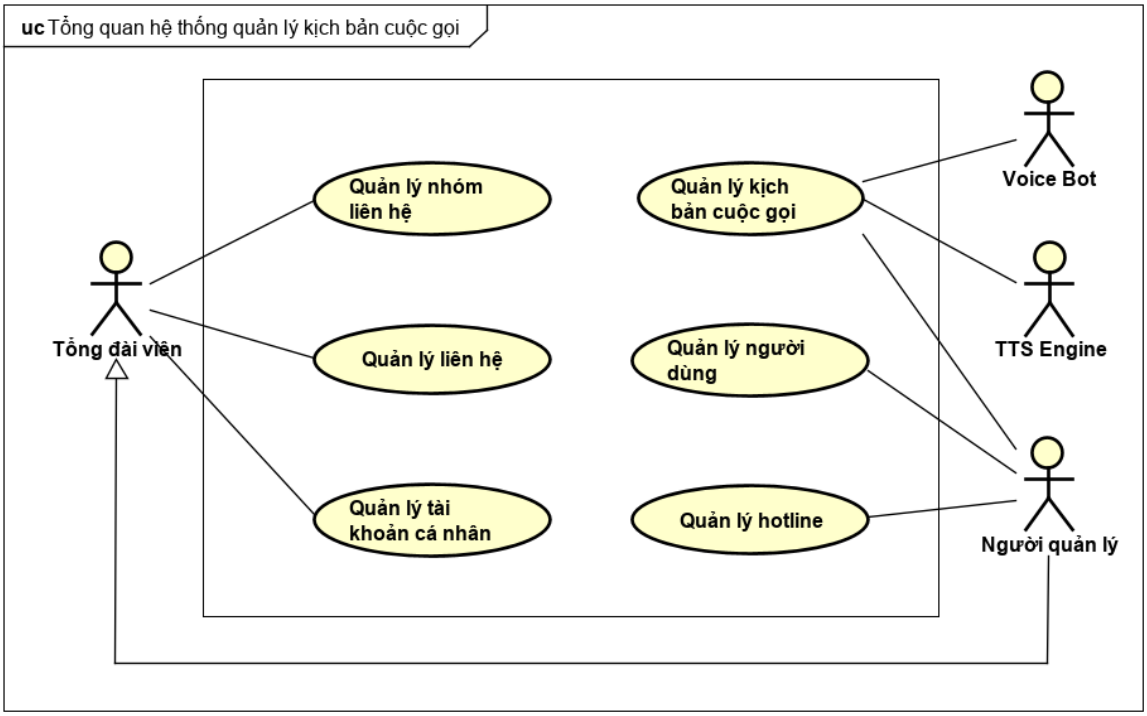
#### 2.1.2.1 Biểu đồ use case tổng quan

**Hình 1** là biểu đồ use case tổng quan của hệ thống quản lý kịch bản cuộc gọi. Hệ thống gồm bốn tác nhân, đó là: Tổng đài viên, Người quản lý, TTS Engine và Voice Bot. Tổng đài viên có thể quản lý liên hệ, quản lý nhóm liên hệ và quản lý tài khoản cá nhân. Người quản lý được kế thừa các chức năng của Tổng đài viên và có thể quản lý kịch bản cuộc gọi, quản lý người dùng trong hệ thống, và quản lý số hotline của tổng đài.

Tác nhân TTS Engine và Voice Bot là các công nghệ do Công ty cổ phần Dịch vụ và Giải pháp dữ liệu Vbee phát triển.

TTS Engine là dịch vụ chuyển văn bản thành giọng nói. Doanh nghiệp có thể cá nhân hóa nội dung cuộc gọi theo thông tin của khách hàng bằng cách thiết đặt sẵn các văn bản chứa thông tin cá nhân hóa. Hệ thống sẽ sử dụng TTS Engine để tổng hợp đoạn văn bản đó thành giọng nói và phát ra khi khách hàng gọi đến hệ thống.

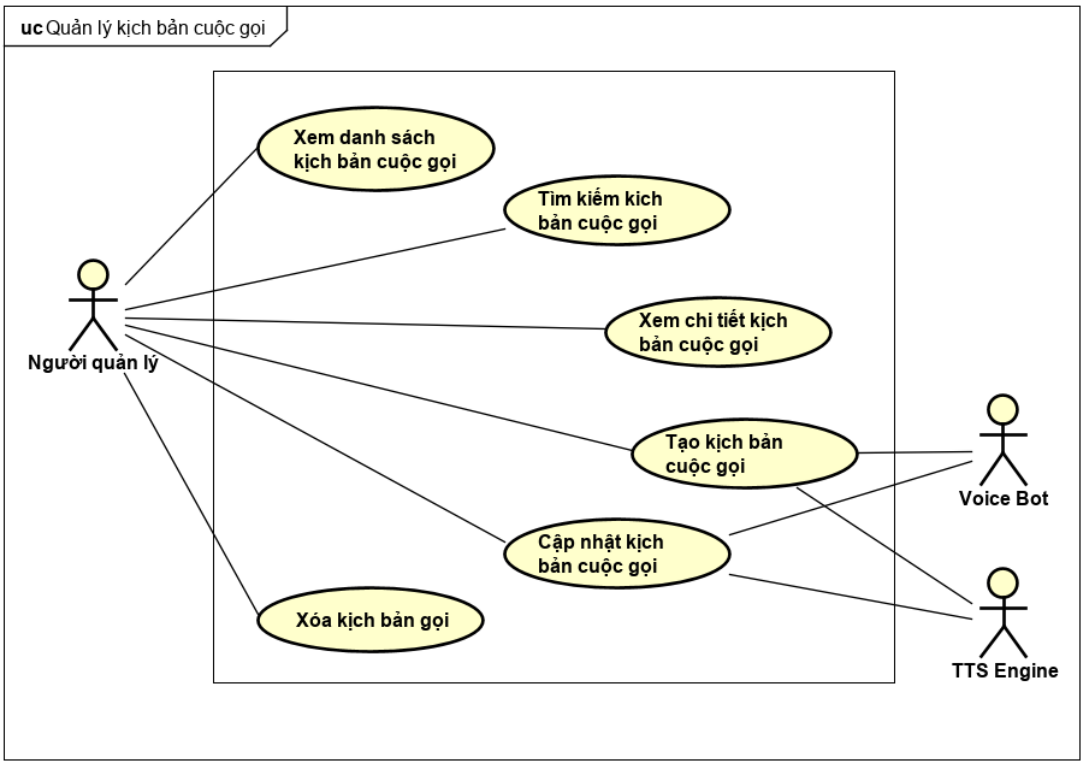
Voice Bot là một công nghệ trí tuệ nhân tạo có thể tự động nghe, hiểu và trả lời khách hàng thay thế tổng đài viên.




**Hình 1** Biểu đồ use case tổng quan hệ thống quản lý kịch bản cuộc gọi.

### 2.1.2.2 Biểu đồ use case phân rã

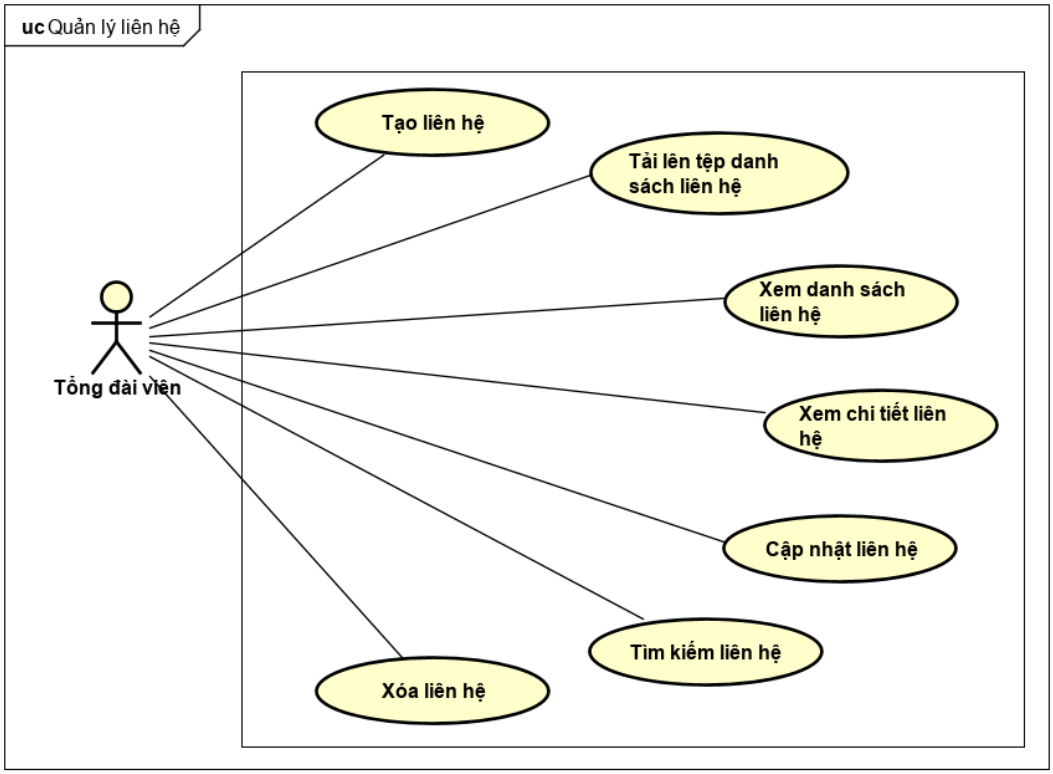
### a) Biểu đồ use case phân rã quản lý kịch bản cuộc gọi




**Hình 2** Biểu đồ use case phân rã quản lý kịch bản cuộc gọi.

**Hình 2** là đồ use case phân rã chức năng quản lý kịch bản cuộc gọi của Người quản lý. Người quản lý có thể xem danh sách, tìm kiếm, xem chi tiết, tạo, cập nhật và xóa kịch bản cuộc gọi.

**b) Biểu đồ use case phân rã quản lý liên hệ**

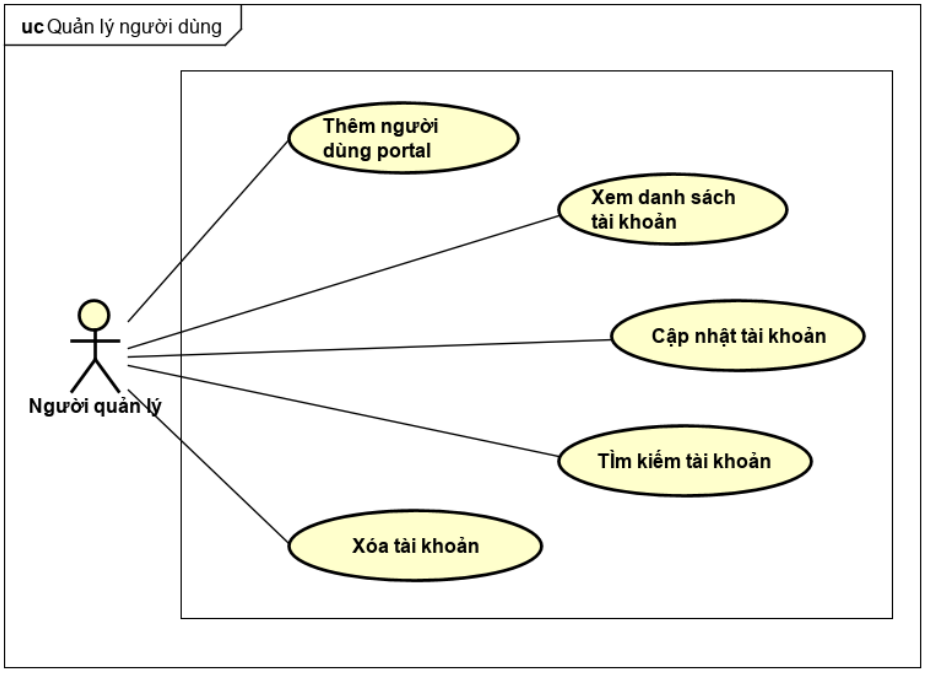



**Hình 3** Biểu đồ use case phân rã quản lý liên hệ.

Hệ thống quản lý kịch bản cho phép tổng đài viên quản lý danh sách liên hệ trong hệ thống như **Hình 3**. Tổng đài viên có thể xem danh sách, tạo, xem chi tiết, sửa và xóa liên hệ. Ngoài ra, hệ thống còn cung cấp thêm chức năng tải lên tệp danh sách liên hệ giúp người dùng có thể tải lên cùng lúc nhiều liên hệ vào hệ thống.

**c) Biểu đồ use case phân rã quản lý người dùng**

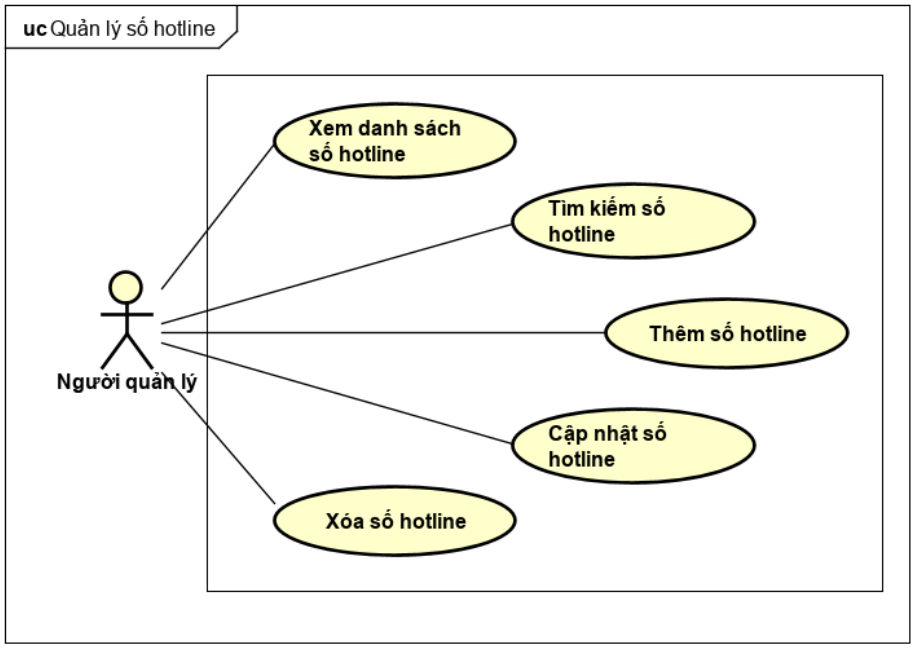
**Hình 4** là biểu đồ phân rã cho use case quản lý người dùng. Người quản lý có thể thêm người dùng vào sử dụng hệ thống, xem danh sách các tài khoản trong hệ thống, cập nhật thông tin cơ bản của tài khoản người dùng (ví dụ như tên, số điện thoại, vai trò), tìm kiếm tài khoản, xóa tài khoản ra khỏi hệ thống.




**Hình 4** Biểu đồ use case phân rã quản lý người dùng.

**d) Biểu đồ use case phân rã quản lý số hotline**

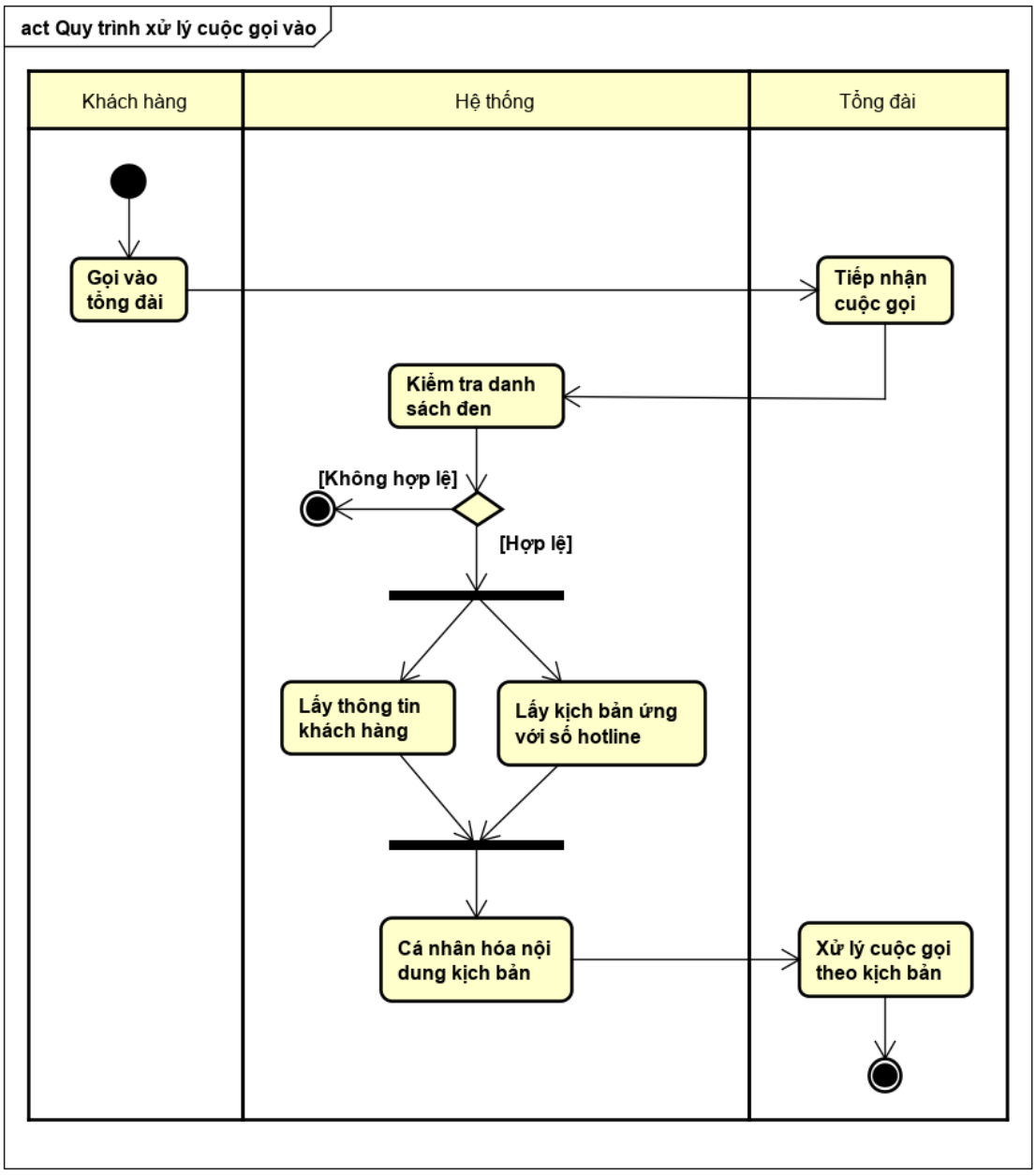
**Hình 5** là biểu đồ use case phân rã cho chức năng quản lý hotline của Người quản lý. Người quản lý có thể xem danh sách, tìm kiếm, thêm, cập nhật và xóa số hotline.




**Hình 5** Biểu đồ use case phân rã quản lý số hotline.

### 2.1.2.3 Quy trình nghiệp vụ

Trong hệ thống quản lý kịch bản cuộc gọi có nhiều quy trình nghiệp vụ. Dưới đây em xin trình bày quy trình nghiệp vụ chính đó là Quy trình xử lý cuộc gọi vào như **Hình 6**. Khi khách hàng gọi đến số hotline của tổng đài, tổng đài sẽ tiếp nhận cuộc gọi. Hệ thống kiểm tra xem số điện thoại của khách hàng có nằm trong danh sách đen hay không. Nếu có, cuộc gọi sẽ kết thúc. Nếu số điện thoại của khách hàng là hợp lệ, hệ thống lấy thông tin khách hàng, lấy kịch bản tương ứng rồi thực hiện cá nhân hóa nội dung kịch bản cuộc gọi. Sau đó hệ thống trả thông tin kịch bản cho tổng đài. Tổng đài thực hiện theo kịch bản mà người dùng đã cấu hình cho đến khi kết thúc cuộc gọi.




**Hình 6** Quy trình xử lý cuộc gọi vào.

### 2.1.3 Đặc tả chức năng

#### 2.1.3.1 Đặc tả use case tạo kịch bản cuộc gọi

**Bảng 1** Đặc tả use case tạo kịch bản cuộc gọi

<table>
<tr><td>Mã use case</td><td>UC001</td><td>Tên use case</td><td>Use case Tạo kịch bản gọi</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Người quản lý</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng đăng nhập vào hệ thống quản lý kịch bản cuộc gọi với vai trò là Người quản lý</td></tr>
<tr><td>Luồng sự kiện chính (Thành công)</td><td colspan="3">
<table>
<tr><th>STT</th><th>Thực hiện bởi</th><th>Hành động</th></tr>
<tr><td>1.</td><td>Người quản lý</td><td>Chọn chức năng tạo kịch bản gọi</td></tr>
<tr><td>2.</td><td>Hệ thống</td><td>Hiển thị giao diện tạo kịch bản gọi</td></tr>
<tr><td>3.</td><td>Người quản lý</td><td>Cài đặt các thông số của kịch bản (mô tả phía dưới *)</td></tr>
<tr><td>4.</td><td>Người quản lý</td><td>Thực hiện lặp lại các thao tác tạo node và cấu hình cho node để tạo ra luồng kịch bản cuộc gọi</td></tr>
<tr><td>4.1.</td><td>Người quản lý</td><td>Yêu cầu tạo node cho luồng kịch bản</td></tr>
<tr><td>4.2.</td><td>Hệ thống</td><td>Hiển thị giao diện tạo node cho luồng kịch bản</td></tr>
<tr><td>4.3.</td><td>Người quản lý</td><td>Nhập các thông tin của node (mô tả phía dưới **)</td></tr>
<tr><td>4.4.</td><td>Người quản lý</td><td>Thực hiện lặp lại các thao tác thêm hành động và cài đặt hành động cho node</td></tr>
<tr><td>4.4.1.</td><td>Người quản lý</td><td>Chọn thẻ hành động (danh sách các thẻ hành động được mô tả phía dưới ***)</td></tr>
<tr><td>4.4.2.</td><td>Người quản lý</td><td>Nhập các trường thông tin tương ứng của thẻ hành động (mô tả phía dưới ****)</td></tr>
<tr><td>4.5.</td><td>Người quản lý</td><td>Gửi yêu cầu tạo node cho luồng kịch bản</td></tr>
<tr><td>4.6.</td><td>Hệ thống</td><td>Kiểm tra thông tin của node</td></tr>
</table>
</td></tr>
</table>

| | | | |
|---|---|---|---|
| | 4.7. | Hệ thống | Thêm node mới vào luồng kịch bản |
| | 5. | Người quản lý | Yêu cầu tạo kịch bản |
| | 6. | Hệ thống | Kiểm tra các thông tin của kịch bản |
| | 7. | Hệ thống | Lưu thông tin kịch bản vào cơ sở dữ liệu |
| | 8. | Hệ thống | Thông báo tạo kịch bản thành công |
| **Luồng sự kiện thay thế** | **STT** | **Thực hiện bởi** | **Hành động** |
| | 4.7a. | Hệ thống | Thông báo lỗi: cần nhập đủ các trường thông tin cần thiết của node nếu Người quản lý nhập thiếu |
| | 4.7b. | Hệ thống | Thông báo lỗi: cần chọn ít nhất một thẻ hành động cho node nếu Người quản lý chưa chọn thẻ hành động nào cho node |
| | 6a. | Hệ thống | Thông báo lỗi: cần nhập đủ các thông tin cần thiết của kịch bản gọi |
| **Hậu điều kiện** | Không | | |

* Dữ liệu đầu vào các cài đặt của kịch bản:

**Bảng 2** Dữ liệu đầu vào cài đặt thông số kịch bản

| STT | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| 1. | Tên kịch bản | | Có |
| 2. | Nhạc nền | Nhạc nền được phát khi khách hàng gọi đến tổng đài | Không |
| 3. | Giọng nói | Loại giọng nói sử dụng khi sử dụng chuyển văn bản thành giọng nói | Có |
| 4. | Tốc độ đọc | Tốc độ đọc của người đọc khi sử dụng chuyển văn bản thành giọng nói | Có |

| | | | |
|---|---|---|---|
| 5. | Kịch bản nâng cao | Kích hoạt chế độ kịch bản nâng cao | Không |

** Các trường thông tin của node:

**Bảng 3** Các trường thông tin của node

| STT | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| 1. | Tên node | | Có |
| 2. | Phím bấm | Phím bấm của người dùng để kích hoạt node | Có |

*** Danh sách các thẻ hành động của node:

Danh sách các thẻ hành động được mô tả chi tiết tại **Bảng 27**, trong mục 5.1.2.2.

**** Dữ liệu đầu vào của các thẻ hành động

**Bảng 4** Dữ liệu đầu vào của các thẻ hành động

| Thẻ hành động | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| Nhập văn bản | Số lần lặp lại | Số lần lặp lại khi phát âm thanh giọng nói | Có |
| | Thời gian chờ nhấn phím | Thời gian tổng đài chờ khách hàng nhấn phím bấm. Hết thời gian này tổng đài sẽ không ghi nhận phím bấm nữa | Có |
| | Nội dung | Nội dung của văn bản | Có |
| | Nội dung thay thế | Nội dung văn bản sử dụng để thay thế khi hệ thống không có thông tin của khách hàng | Có |
| Chuyển đến kịch bản khác | Kịch bản | Hệ thống chuyển đến kịch bản này để xử lý cuộc gọi | Có |
| Tải lên tệp âm thanh | Tệp âm thanh | Tệp âm thanh có định dạng mp3 hoặc wav | Có |
| Kết nối tổng đài viên | Hàng đợi | Một tổng đài viên sẽ được chọn ra từ hàng đợi này để tiếp nhận cuộc gọi của khách hàng | Có |
| Kết nối Bot | Địa chỉ kết nối với Bot | Địa chỉ websocket để kết nối với Bot | Có |

| | | | |
|---|---|---|---|
| | Bot token | Token định danh cho Bot | Có |
| Nghe lại | Không | | |
| Quay lại | Không | | |
| Kết thúc cuộc gọi | Mã kết thúc cuộc gọi | Mã kết thúc cuộc gọi phục vụ cho việc tra cứu sau này | |

### 2.1.3.2 Đặc tả use case tải lên tệp danh sách liên hệ

**Bảng 5** Đặc tả use case tải lên tệp danh sách liên hệ

<table>
<tr><td>Mã use case</td><td>UC002</td><td>Tên use case</td><td>Use case Tải lên tệp danh sách liên hệ</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Tổng đài viên, Giám sát viên hoặc Người quản lý</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng đăng nhập vào hệ thống quản lý kịch bản cuộc gọi với vai trò là Tổng đài viên, Giám sát viên hoặc Người quản lý</td></tr>
<tr><td>Luồng sự kiện chính (Thành công)</td><td colspan="3">
<table>
<tr><th>STT</th><th>Thực hiện bởi</th><th>Hành động</th></tr>
<tr><td>1.</td><td>Người dùng</td><td>Chọn chức năng tải tệp danh sách liên hệ</td></tr>
<tr><td>2.</td><td>Người dùng</td><td>Chọn tệp danh sách liên hệ cần tải lên</td></tr>
<tr><td>3.</td><td>Hệ thống</td><td>Kiểm tra tệp danh sách liên hệ (mô tả phía dưới *)</td></tr>
<tr><td>4.</td><td>Hệ thống</td><td>Đọc tệp danh sách liên hệ và lưu thông tin các liên hệ</td></tr>
<tr><td>5.</td><td>Hệ thống</td><td>Thông báo tải lên danh sách liên hệ thành công và hiển thị danh sách các liên hệ</td></tr>
</table>
</td></tr>
<tr><td>Luồng sự kiện thay thế</td><td colspan="3">
<table>
<tr><th>STT</th><th>Thực hiện bởi</th><th>Hành động</th></tr>
<tr><td>3a.</td><td>Hệ thống</td><td>Thông báo lỗi: Định dạng tệp không hợp lệ nếu người dùng tải lên tệp sai định dạng</td></tr>
</table>
</td></tr>
</table>

| | | | |
|---|---|---|---|
| | 3b. | Hệ thống | Thông báo lỗi: Kích thước tệp tải lên quá lớn nếu người dùng tải lên tệp có kích thước lớn hơn quy định |
| | 5a. | Hệ thống | Thông báo lỗi: Liên hệ tải lên bị trùng nếu thông tin liên hệ trong tệp trùng với liên hệ đã có trong hệ thống |
| **Hậu điều kiện** | Không | | |

* Dữ liệu đầu vào use case tải lên tệp danh sách liên hệ:

**Bảng 6** Dữ liệu đầu vào use case tải lên tệp danh sách liên hệ

| STT | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| 1. | Tệp danh sách liên hệ | Tệp excel danh sách các liên hệ, có kích thước nhỏ hơn 5Mb. | Có |

### 2.1.3.3 Đặc tả use case quản lý số hotline

**Bảng 7** Đặc tả use case quản lý số hotline

| **Mã use case** | UC003 | **Tên use case** | Use case Quản lý số hotline |
|---|---|---|---|
| **Tác nhân** | Người quản lý | | |
| **Tiền điều kiện** | Khách đã đăng nhập với vai trò Người quản lý | | |

**Thêm (C):**

| | STT | Thực hiện bởi | Hành động |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người quản lý | Chọn chức năng thêm số hotline |
| | 2. | Hệ thống | Hiển thị hộp thoại thêm số hotline |
| | 3. | Người quản lý | Nhập các thông tin số hotline (mô tả phía dưới *) |
| | 4. | Người quản lý | Yêu cầu thêm số hotline |
| | 5. | Hệ thống | Kiểm tra các trường thông tin bắt buộc |

| | | | |
|---|---|---|---|
| | 6. | Hệ thống | Lưu thông tin số hotline |
| | 7. | Hệ thống | Thông báo thêm số hotline thành công |
| **Luồng sự kiện thay thế** | 6a. | Hệ thống | Thông báo lỗi: cần nhập đủ các trường thông tin cần thiết nếu Người quản lý nhập thiếu |
| | 7a. | Hệ thống | Thông báo lỗi: không thể thêm số hotline nếu có lỗi xảy ra trong quá trình thêm số hotline |

**Xem (R):**

| | **STT** | **Thực hiện bởi** | **Hành động** |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người quản lý | Chọn chức năng Xem danh sách số hotline |
| | 2. | Hệ thống | Lấy danh sách tất cả các số hotline trong hệ thống theo trang |
| | 3. | Hệ thống | Hiển thị giao diện xem danh sách số hotline |
| **Luồng sự kiện thay thế** | 4a1. | Người quản lý | Thay đổi số trang hiện tại |
| | 4a2. | Hệ thống | Lấy danh sách các số hotline theo trang được chọn |

**Cập nhật (U):**

| | **STT** | **Thực hiện bởi** | **Hành động** |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người quản lý | Chọn chức năng cập nhật số hotline |
| | 2. | Hệ thống | Hiển thị hộp thoại chỉnh sửa số hotline |
| | 3. | Người quản lý | Nhập các thông tin chỉnh sửa số hotline (mô tả phía dưới *) |
| | 4. | Người quản lý | Yêu cầu cập nhật số hotline |
| | 5. | Hệ thống | Kiểm tra các trường thông tin bắt buộc |
| | 6. | Hệ thống | Lưu thông tin số hotline |

| | | | |
|---|---|---|---|
| | 7. | Hệ thống | Thông báo chỉnh sửa số hotline thành công |
| | 5a. | Hệ thống | Kết thúc use case nếu Người quản lý chọn chức năng hủy thao tác |
| **Luồng sự kiện thay thế** | 6a. | Hệ thống | Thông báo lỗi: cần nhập đủ các trường thông tin cần thiết nếu Người quản lý nhập thiếu |
| | 7a. | Hệ thống | Thông báo lỗi: không thể cập nhật số hotline nếu có lỗi xảy ra trong quá trình cập nhật số hotline |

**Xóa (D):**

| | **STT** | **Thực hiện bởi** | **Hành động** |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người quản lý | Chọn một số hotline và yêu cầu xóa |
| | 2. | Hệ thống | Hiển thị cảnh báo yêu cầu Người quản lý xác nhận xóa |
| | 3. | Người quản lý | Xác nhận xóa số hotline |
| | 4. | Hệ thống | Thực hiện xóa số hotline |
| | 5. | Hệ thống | Thông báo xóa số hotline thành công |
| **Luồng sự kiện thay thế** | 4a. | Hệ thống | Kết thúc use case nếu Người quản lý xác nhận không xóa số hotline |
| | 5a. | Hệ thống | Thông báo lỗi: không thể xóa được số hotline nếu có lỗi xảy ra khi xóa số hotline |

| | |
|---|---|
| **Hậu điều kiện** | Không |

* Dữ liệu đầu vào khi thêm / cập nhật số hotline:

**Bảng 8** Dữ liệu đầu vào khi thêm / cập nhật số hotline

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc** |
|---|---|---|---|
| 1. | Số hotline | Số hotline của tổng đài chăm sóc khách hàng | Có |
| 2. | Tên số hotline | | Có |

| | | | |
|---|---|---|---|
| 3. | Kịch bản gọi vào | Kịch bản gọi vào ứng với số hotline | Có |

## 2.2 Hệ thống xác thực và phân quyền tập trung

### 2.2.1 Khảo sát hiện trạng

Xác thực và phân quyền luôn là thành phần không thể thiếu của một hệ thống. Xác thực là việc hệ thống xác minh xem người dùng đó là ai. Người dùng cần nhập thông tin đăng nhập hoặc sử dụng các ứng dụng khác như Google, Facebook,… để xác thực. Phân quyền để xác định xem người dùng đó có thể làm gì với các tài nguyên của hệ thống (ví dụ người dùng có quyền xem, chỉnh sửa mà không có quyền xóa tệp).

Thông thường, khi người dùng truy cập vào ứng dụng, hệ thống sẽ kiểm tra xem người dùng đã đăng nhập hay chưa. Nếu chưa đăng nhập, hệ thống sẽ hiển thị một form đăng nhập để người dùng nhập thông tin tài khoản bao gồm tên người dùng / email / số điện thoại và mật khẩu. Sau đó hệ thống xác thực người dùng, nếu thành công, hệ thống sẽ cho phép người dùng truy cập tài nguyên của hệ thống. Với các ứng dụng khác nhau, người dùng cần có các tài khoản khác nhau, mỗi lần truy cập vào đều phải đăng nhập. Điều này gây khó chịu với người dùng vì phải nhớ nhiều tài khoản và phải đăng nhập nhiều lần.

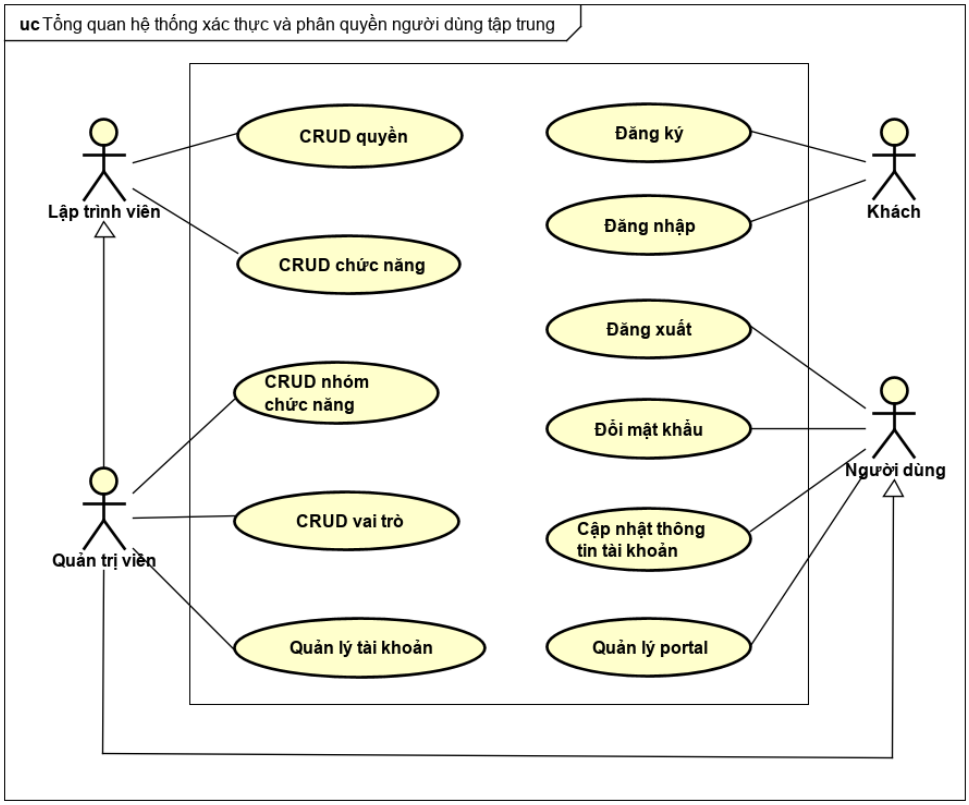
Tuy nhiên, với các hệ thống được tích hợp với nhau, việc sử dụng mỗi tài khoản khác nhau cho một hệ thống là không phù hợp. Người dùng chỉ cần một tài khoản duy nhất mà có thể sử dụng cho tất cả các hệ thống đã tích hợp. Điều này giúp khắc phục các vấn đề đã nêu ra ở trên. Do đó, yêu cầu đặt ra là phát triển một hệ thống đăng nhập chung cho tất cả các hệ thống được tích hợp. Người dùng chỉ cần đăng nhập duy nhất một lần trên hệ thống này, sau đó có thể sử dụng các hệ thống tích hợp khác mà không phải đăng nhập lại. Đây được gọi là cơ chế Single Sign On – SSO.

Sau quá trình xác thực, hệ thống đã biết người dùng đó là ai. Việc tiếp theo hệ thống cần thực hiện là xác định xem người dùng dùng này có thể thực hiện những hành động nào, tác động lên tài nguyên nào của hệ thống. Để thực hiện được việc đó, hệ thống cần xây dựng một hệ thống luật và kiểm tra mỗi khi người dùng yêu cầu truy cập vào tài nguyên hệ thống.

Do đó, nhu cầu đặt ra là xây dựng hệ thống xác thực và phân quyền tập trung cho các hệ thống liên kết với nhau. Hệ thống này cung cấp cơ chế Single Sign On và cho phép quản trị viên có thể quản lý quyền tập trung cho toàn bộ hệ thống.

### 2.2.2 Tổng quan chức năng

#### 2.2.2.1 Biểu đồ use case tổng quan



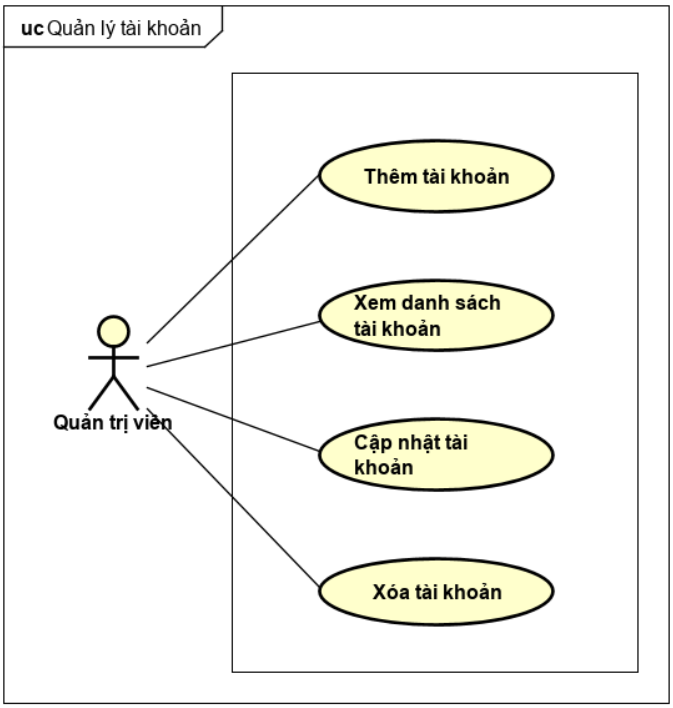

**Hình 7** Biểu đồ use case tổng quan hệ thống xác thực và phân quyền người dùng.

Biểu đồ use case tổng quan hệ thống xác thực và phân quyền người dùng được thể hiện qua Hình 7. Hệ thống gồm có bốn tác nhân chính là: Khách, Người dùng, Lập trình viên và Quản trị viên. Để sử dụng các chức năng của hệ thống, khách cần đăng ký một tài khoản sau đó đăng nhập vào hệ thống. Người dùng có thể cập nhật thông tin tài khoản của mình, đổi mật khẩu và quản lý portal. Lập trình viên có thể quản lý các quyền và chức năng trong hệ thống. Quản trị viên kế thừa các chức năng của người dùng và Lập trình viên. Ngoài ra, quản trị viên còn có các chức năng phục vụ cho việc phân quyền người dùng như CRUD nhóm chức năng, CRUD vai trò và quản lý tài khoản trong hệ thống.

Quản trị viên có thể định ra các quyền trong hệ thống. Sau đó, quản trị viên có thể tạo ra các chức năng bằng cách thêm các quyền vào chức năng. Tiếp theo, quản trị viên có thể tạo ra các nhóm chức năng, trong đó gồm nhiều chức năng. Ví dụ, trong nhóm chức năng quản lý liên hệ gồm các chức năng là tạo mới liên hệ, xem liên hệ, chỉnh sửa liên hệ và xóa liên hệ. Cuối cùng, quản trị viên sẽ tạo ra các vai trò trong đó chứa các nhóm chức năng đã được định nghĩa. Ví dụ vai trò Tổng đài viên có thể có các nhóm chức năng là quản lý liên hệ, quản lý cuộc gọi và quản lý tài khoản cá nhân.

#### 2.2.2.2 Biểu đồ use case phân rã

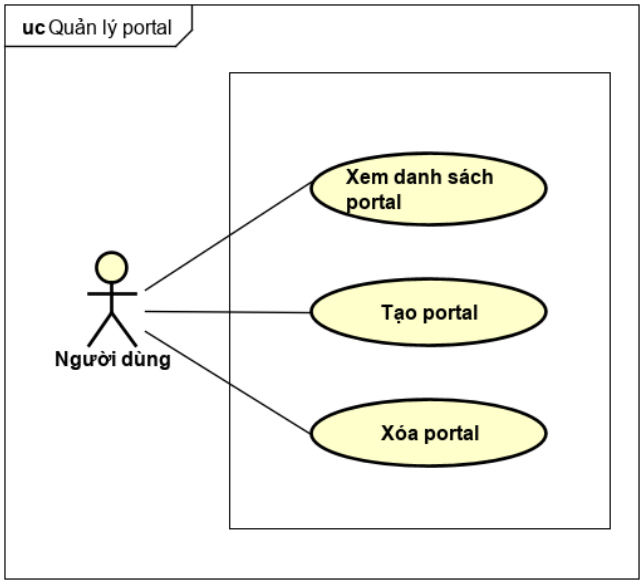
**a) Biểu đồ use case phân rã quản lý tài khoản**




**Hình 8** Biểu đồ use case phân rã quản lý tài khoản.

Quản trị viên có thể quản lý các tài khoản trong hệ thống như **Hình 8**. Hệ thống cho phép quản trị viên có thể thêm, xem danh sách, cập nhật và xóa tài khoản.

**b) Biểu đồ use case phân rã quản lý portal**




**Hình 9** Biểu đồ use case phân rã quản lý portal.

Người dùng có thể quản lý portal của mình như **Hình 9** biểu diễn. Người dùng có thể tạo portal là các trang với tên miền phụ riêng, có thể xem danh sách hoặc xóa portal mà mình tạo ra.

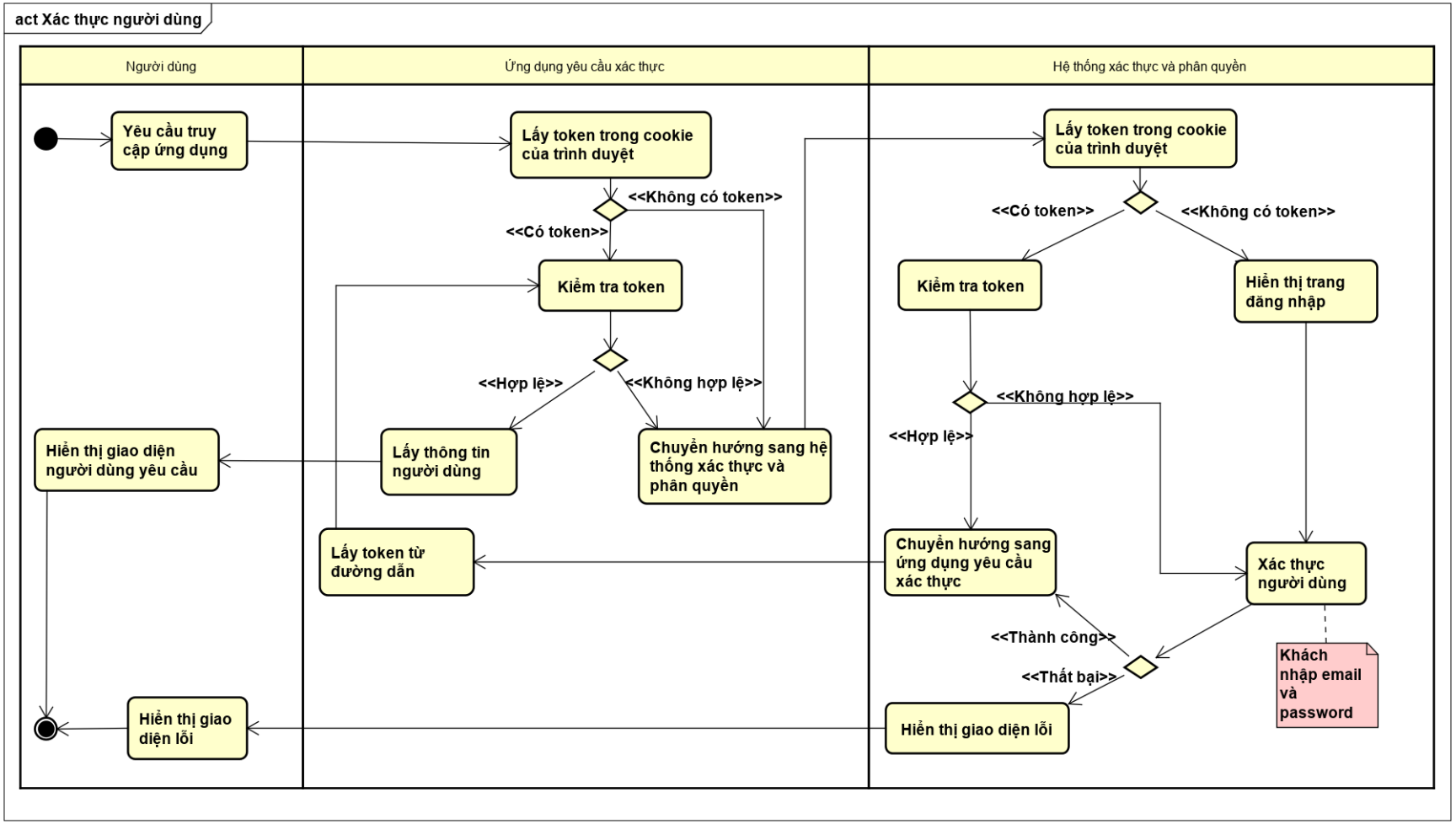
#### 2.2.2.3 Quy trình nghiệp vụ

Trong phần này, em sẽ trình bày quy trình xác thực của người dùng khi người dùng truy cập vào ứng dụng có yêu cầu xác thực.

**Hình 10** biểu diễn quy trình xác thực người dùng trong hệ thống xác thực và phân quyền người dùng tập trung.

Nếu người dùng truy cập vào ứng dụng yêu cầu xác thực lần đầu tiên thì ứng dụng sẽ chuyển hướng sang hệ thống xác thực và phân quyền kèm theo url hiện tại. Tại đây người dùng cập nhập vào thông tin xác thực là tên người dùng / email / số điện thoại và mật khẩu. Hệ thống sẽ kiểm tra thông tin đăng nhập. Nếu hợp lệ, hệ thống sẽ sinh ra một token để lưu vào cookie của trình duyệt và chuyển hướng về ứng dụng mà người dùng yêu cầu truy cập kèm theo token. Ứng dụng sẽ nhận được token đó qua url của trình duyệt và kiểm tra. Nếu hợp lệ, ứng dụng sẽ lưu token vào cookie của trình duyệt và hiển thị giao diện mà người dùng yêu cầu hiển thị.

Khi người dùng truy cập vào ứng dụng lần thứ 2, ứng dụng sẽ lấy token trong cookie ra và kiểm tra. Nếu token hợp lệ thì ứng dụng sẽ hiển thị luôn giao diện người dùng yêu cầu mà không cần phải đăng nhập lại. Nếu token không hợp lệ thì ứng dụng sẽ điều hướng sang hệ thống xác thực và phân quyền như mô tả ở trên.




**Hình 10** Quy trình xác thực người dùng.

### 2.2.3 Đặc tả chức năng

#### 2.2.3.1 Đặc tả use case quản lý tài khoản

**Bảng 9** Đặc tả use case quản lý tài khoản

| **Mã use case** | UC001 | **Tên use case** | Use case Quản lý tài khoản |
|---|---|---|---|
| **Tác nhân** | Quản trị viên | | |
| **Tiền điều kiện** | Khách đã đăng nhập với vai trò Quản trị viên | | |

**Thêm (C):**

| | **STT** | **Thực hiện bởi** | **Hành động** |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Quản trị viên | Chọn chức năng thêm mới tài khoản |
| | 2. | Hệ thống | Hiển thị hộp thoại thêm mới tài khoản |
| | 3. | Quản trị viên | Nhập các thông tin tài khoản (mô tả phía dưới *) |
| | 4. | Quản trị viên | Yêu cầu thêm tài khoản |
| | 5. | Hệ thống | Kiểm tra các trường thông tin bắt buộc |
| | 6. | Hệ thống | Kiểm tra các trường thông tin để đảm bảo không trùng lặp trong hệ thống |
| | 7. | Hệ thống | Lưu thông tin tài khoản vào cơ sở dữ liệu |
| | 8. | Hệ thống | Thông báo tạo tài khoản thành công và gửi thông tin tài khoản đến email đăng ký |
| **Luồng sự kiện thay thế** | 6a. | Hệ thống | Thông báo lỗi: cần nhập đủ các trường thông tin cần thiết nếu quản trị viên nhập thiếu |
| | 7a. | Hệ thống | Thông báo cụ thể các trường thông tin bị trùng nếu quản trị viên nhập trùng |

**Xem (R):**

| | STT | Thực hiện bởi | Hành động |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Quản trị viên | Yêu cầu Xem danh sách tài khoản |
| | 2. | Hệ thống | Lấy danh sách tất cả các tài khoản trong hệ thống theo trang |
| | 3. | Hệ thống | Hiển thị giao diện xem danh sách tài khoản |
| **Luồng sự kiện thay thế** | 4a1. | Quản trị viên | Thay đổi số trang hiện tại |
| | 4a2. | Hệ thống | Lấy danh sách các tài khoản theo trang được chọn |

**Xóa (D):**

| | STT | Thực hiện bởi | Hành động |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Quản trị viên | Chọn một tài khoản và yêu cầu xóa |
| | 2. | Hệ thống | Hiển thị cảnh báo yêu cầu người dùng xác nhận xóa |
| | 3. | Quản trị viên | Xác nhận xóa tài khoản |
| | 4. | Hệ thống | Xóa tài khoản trong cơ sở dữ liệu |
| | 5. | Hệ thống | Thông báo xóa tài khoản thành công |
| **Luồng sự kiện thay thế** | 4a. | Hệ thống | Kết thúc use case nếu người dùng xác nhận không xóa tài khoản |
| | 5a. | Hệ thống | Thông báo lỗi: không thể xóa được tài khoản nếu có lỗi xảy ra khi xóa tài khoản |

| | |
|---|---|
| **Hậu điều kiện** | Không |

* Dữ liệu đầu vào khi thêm tài khoản:

**Bảng 10** Dữ liệu đầu vào chức năng thêm tài khoản

| STT | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| 1. | Email | | Có |

| | | | |
|---|---|---|---|
| 2. | Mật khẩu | | Có |
| 3. | Tên | | Có |
| 4. | Số điện thoại | | Có |

### 2.2.3.2 Đặc tả use case quản lý portal

**Bảng 11** Đặc tả use case quản lý portal

| **Mã use case** | UC002 | **Tên use case** | Use case Quản lý portal |
|---|---|---|---|
| **Tác nhân** | Người dùng | | |
| **Tiền điều kiện** | Khách đã đăng nhập với vai trò người dùng | | |

**Tạo (C):**

| | **STT** | **Thực hiện bởi** | **Hành động** |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người dùng | Chọn chức năng thêm mới portal |
| | 2. | Hệ thống | Hiển thị hộp thoại thêm mới portal |
| | 3. | Người dùng | Nhập các thông tin portal (mô tả phía dưới *) |
| | 4. | Người dùng | Yêu cầu thêm portal |
| | 5. | Hệ thống | Kiểm tra các trường thông tin bắt buộc |
| | 6. | Hệ thống | Kiểm tra các trường thông tin để đảm bảo không trùng lặp trong hệ thống |
| | 7. | Hệ thống | Lưu thông tin tài khoản vào cơ sở dữ liệu |
| | 8. | Hệ thống | Thông báo thêm mới portal thành công và hiển thị ra danh sách |
| | 6a. | Hệ thống | Thông báo lỗi: cần nhập đủ các trường thông tin cần thiết nếu người dùng nhập thiếu |

| | | | |
|---|---|---|---|
| **Luồng sự kiện thay thế** | 7a. | Hệ thống | Thông báo cụ thể các trường thông tin bị trùng nếu người dùng nhập trùng |

**Xem (R):**

| | STT | Thực hiện bởi | Hành động |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người dùng | Đăng nhập thành công |
| | 2. | Hệ thống | Lấy danh sách tất cả các portal trong hệ thống |
| | 3. | Hệ thống | Hiển thị giao diện xem danh sách portal |
| **Luồng sự kiện thay thế** | 3a. | Người dùng | Thông báo tài khoản chưa có portal |

**Xóa (D):**

| | STT | Thực hiện bởi | Hành động |
|---|---|---|---|
| **Luồng sự kiện chính** | 1. | Người dùng | Chọn một portal và yêu cầu xóa |
| | 2. | Hệ thống | Hiển thị cảnh báo yêu cầu người dùng xác nhận xóa |
| | 3. | Người dùng | Xác nhận xóa portal |
| | 4. | Hệ thống | Xóa portal trong cơ sở dữ liệu |
| | 5. | Hệ thống | Thông báo xóa portal thành công |
| **Luồng sự kiện thay thế** | 4a. | Hệ thống | Kết thúc use case nếu người dùng xác nhận không xóa portal |
| | 5a. | Hệ thống | Thông báo lỗi: không thể xóa được portal nếu có lỗi xảy ra khi xóa tài khoản |

| | |
|---|---|
| **Hậu điều kiện** | Không |

* Dữ liệu đầu vào khi thêm portal:

**Bảng 12** Dữ liệu đầu vào chức năng thêm portal

| STT | Trường dữ liệu | Mô tả | Bắt buộc |
|---|---|---|---|
| 1. | Tên portal | | Có |
| 2. | Subdomain của portal | Phục vụ việc tạo tên miền cho portal | Có |
| 3. | Tiền tố của portal | Phục vụ việc sinh id của dữ liệu | Có |

Qua việc khảo sát các hệ thống tổng đài thông minh, em sẽ dựa trên đó và phát triển một hệ thống quản lý kịch bản cuộc gọi có tính năng mới là công cụ tạo kịch bản cuộc gọi, giúp cho doanh nghiệp có thể tạo được các kịch bản chuyên nghiệp hơn, từ đó giảm thiểu chi phí và nâng cao hiệu quả chăm sóc khách hàng. Bên cạnh đó, em cũng sẽ phát triển một hệ thống xác thực và phân quyền tập trung để người dùng có thể sử dùng chung một tài khoản trong các ứng dụng liên kết và người quản trị dễ dàng phân quyền người dùng. Để phát triển hai hệ thống trên, em sẽ tìm hiểu các công nghệ và trình bày chi tiết trong Chương 3.

# Chương 3 Công nghệ sử dụng

Chương 2 đã khảo sát và phân tích yêu cầu hệ thống. Trong Chương 3, em trình bày về các công nghệ sử dụng để phát triển hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung. Hệ thống được chia thành hai thành phần frontend và backend. Phía frontend sử dụng thư viện ReactJS và Ant Design để xây dựng giao diện. Để tạo ra các biểu đồ luồng cho kịch bản cuộc gọi, em sử dụng thư viện Ej2 React Diagram. Phía backend sử dụng nền tảng NodeJS kết hợp với ExpressJS và MongoDB, ngoài ra còn sử dụng Redis để cache dữ liệu và RabbitMQ để động bộ dữ liệu trong các hệ thống liên quan.

## 3.1 Frontend

### 3.1.1 ReactJS

ReactJS [4] là một thư viện Javascript được sử dụng cho việc xây dựng các thành phần giao diện UI có khả năng tái sử dụng. Nó được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook. ReactJS lần đầu được triển khai cho ứng dụng Newsfeed của Facebook năm 2011, sau đó được triển khai cho Instagram năm 2012. ReactJS có thể tạo ra những ứng dụng chạy trên nền tảng Web với tốc độ rất nhanh và đạt hiệu suất cao. Dưới đây là một số tính năng quan trọng và nổi bật của ReactJS:

JSX (viết tắt của cụm từ Javascript XML) - là một loại cú pháp mở rộng dành cho ngôn ngữ Javascript viết theo kiểu XML, điều này cho phép lập trình viên có thể viết các đoạn mã HTML ngay trong mã nguồn Javascript để thể hiện các thành phần giao diện người dùng. Khi biên dịch sang mã Javascript, JSX cũng thực hiện việc tối ưu hoá, các lỗi sẽ sớm được phát hiện ngay trong quá trình biên dịch, qua đó giúp cho việc gỡ lỗi trở nên dễ dàng hơn.

Component-Based – React được xây dựng dựa trên Component. Mỗi component đều được thiết kế với một logic cụ thể. Do đó, thay vì phải quan tâm đến toàn bộ ứng dụng, ta chỉ cần tập trung vào các phần nhỏ. Điều này giúp cho việc phát triển và gỡ lỗi trở nên dễ dàng hơn. Có 2 khái niệm chính đối với Component: props và state. Props là một đối tượng, nó lưu trữ các giá trị của các attribute (thuộc tính) của một thẻ. Đây là cách là component có thể nhận được các giá trị của thuộc tính truyền vào từ bên ngoài, và là cách mà các component có thể giao tiếp với nhau. State cũng giống như props, nó lưu trữ thông tin về component, nhưng là lưu trữ dữ liệu động của một component, khi state thay đổi thì toàn bộ component sẽ được

cập nhật lại. Ngoài ra, một ưu điểm nữa của việc sử dụng component là tăng khả năng tái sử dụng code.

Luồng dữ liệu một chiều (One-way Data Binding) - ReactJS được thiết kế tuân theo luồng dữ liệu một chiều từ cha đến các con. Lợi ích của luồng dữ liệu một chiều cho phép kiểm soát tốt hơn trong toàn ứng dụng. Nếu luồng dữ liệu theo hướng khác thì nó yêu cầu các chức năng bổ sung. Đó là bởi vì các Component được cho là bất biến và dữ liệu trong chúng không thể thay đổi từ bên ngoài.

Virtual DOM - đây là một tính năng cho phép ReactJS xây dựng các ứng dụng đạt hiệu suất cao, đem lại trải nghiệm tốt cho người dùng. ReactJS duy trì 2 loại DOM song song là DOM thực và DOM ảo (Virtual DOM). Khi giao diện có một sự thay đổi nào đó, toàn bộ những phần thay đổi này sẽ được kết xuất trong DOM ảo. Sau đó, nó sẽ tiến hành kiểm tra sự khác nhau giữa DOM hiện tại và DOM trước đó. Nếu có sự khác nhau, DOM thực sẽ chỉ cập nhật những phần thay đổi, còn các phần không thay đổi vẫn được giữ nguyên. Điều này làm cho ứng dụng trở nên nhanh hơn và không bị lãng phí bộ nhớ.

Vì ReactJS là một thư viện javascript nhẹ và chỉ hỗ trợ các thành phần UI nên có phải được bổ sung với các công cụ, thư viện, framework hoặc công nghệ khác để xây dựng thành một kiến trúc hoàn chỉnh.

### 3.1.2 Ant Design

Ant Design [5] là thư viện hỗ trợ việc xây dựng UI, UX một cách hiệu quả và nhanh chóng. Nó là tập hợp các components của React được xây dựng theo chuẩn thiết kế của Ant UED Team. Tương tự như chuẩn Material Design, Ant cung cấp hầu hết các component thông dụng trong một ứng dụng Web hiện đại như: Layout, Menu, Button,…

Không chỉ dừng lại ở mức giao diện, Ant Design cũng hỗ trợ sẵn việc thao tác và xử lý các sự kiện đối với các thành phần giao diện người dùng một cách dễ dàng.

Hiện tại, Ant Design là một trong những thư viện hỗ trợ việc xây dựng giao diện phổ biến nhất hiện nay với hơn 25k stars trên Github.

### 3.1.3 Ej2 React Diagram

Ej2 React Diagram [6] là thư viện chuyên sử dụng để biểu diễn các thành phần trực quan, như là các sơ đồ, biểu đồ luồng, sơ đồ tư duy.

Để tạo ra một biểu đồ luồng, cần sử dụng ba đối tượng chính sau mà thư viện Ej2 React Diagram cung cấp: DiagramComponent, Node và Connector.

DiagramComponent là đối tượng hỗ trợ tạo ra khung của biểu đồ luồng và cung cấp nhiều phương thức, sự kiện giúp lập trình viên. Danh sách các thuộc tính, phương thức và sự kiện hình được đưa ra trong **Bảng 13**.

**Bảng 13** Các thuộc tính, phương thức và sự kiện chính của đối tượng DiagramComponent

| STT | Nhóm | Tên | Mô tả |
|---|---|---|---|
| 1. | Thuộc tính | addInfo | Cho phép người dùng lưu một số thông tin thêm về biểu đồ |
| 2. | | backgroundColor | Đặt màu nền cho biểu đồ |
| 3. | | connectorDefaults | Cài đặt các thuộc tính mặc định cho connector |
| 4. | | connectors | Định nghĩa các kết nối trong biểu đồ |
| 5. | | constraints | Định nghĩa các ràng buộc trong biểu đồ, ví dụ như cho phép undo, redo, zoom,… |
| 6. | | nodeDefaults | Cài đặt các thuộc tính mặc định cho node |
| 7. | | nodes | Danh sách các node ở trong biểu đồ |
| 8. | | selectedItems | Định nghĩa tập các item được chọn |
| 9. | | width | Chiều rộng của biểu đồ |
| 10. | | height | Chiều cao của biểu đồ |
| 11. | Phương thức | add | Thêm các connector hoặc node vào biểu đồ |
| 12. | | drag | Kéo một đối tượng đến vị trí nhất định |
| 13. | Sự kiện | positionChange | Vị trí đối tượng thay đổi |

Connector là đối tượng kết nối giữa hai node trong biểu đồ luồng. Connector có các thuộc tính chính được đưa ra trong **Bảng 14**.

**Bảng 14** Các phương thức chính của đối tượng Connector

| STT | Tên | Mô tả |
|---|---|---|
| 1. | addInfo | Cho phép người dùng lưu một số thông tin thêm về connector |

| | | |
|---|---|---|
| 2. | constraints | Định nghĩa các ràng buộc của connector, ví dụ cho phép kéo thả hay không, có thể xóa hay không |
| 3. | sourceID | Cài đặt điểm bắt đầu của connector |
| 4. | targetID | Cài đặt điểm kết thúc của connector |
| 5. | type | Định nghĩa kiểu connector |

Node là những nút trong biểu đồ. Các thuộc tính chính của Node được liệt kê ở

**Bảng 15** Các thuộc tính chính của đối tượng Node

| STT | Tên | Mô tả |
|---|---|---|
| 1. | addInfo | Cho phép người dùng lưu một số thông tin thêm về node |
| 2. | constraints | Định nghĩa các ràng buộc của node, ví dụ cho phép kéo thả hay không, có thể xóa hay không |
| 3. | width | Cài đặt độ rộng của node |
| 4. | height | Cài đặt chiều cao của node |
| 5. | shape | Cài đặt hình dạng của node |
| 6. | offsetX | Vị trí theo trục X trong biểu đồ |
| 7. | offsetY | Vị trí theo trục Y trong biểu đồ |
| 8. | backgroundColor | Màu nền của node |

## 3.2 Backend

### 3.2.1 NodeJS

NodeJS [7] là một nền tảng được xây dựng dựa trên V8 Javascript Engine - trình thông dịch thực thi mã Javascript, giúp cho việc xây dựng các ứng dụng web trở nên đơn giản và dễ dàng mở rộng.

NodeJS được phát triển bởi Ryan Dahl vào năm 2009 và có thể chạy được trên những hệ điều hành khác nhau như: MacOS, Windows, Linux,…

NodeJS được viết bằng Javascript với cộng đồng người dùng lớn mạnh. Nhờ cơ chế xử lý bất đồng bộ (non-blocking) nên tốc độ xử lý của NodeJS rất nhanh, nó có thể xử lý hàng nghìn kết nối cùng một lúc mà không gặp phải bất cứ vấn đề nào. Ngoài ra, NodeJS cũng có khả năng dễ dàng mở rộng, đây là một lợi thế quan trọng

### 3.2.2 ExpressJS

ExpressJS [8] là một framework được xây dựng trên nền tảng của NodeJS. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. ExpressJS hỗ trợ các phương thức HTTP và các middleware, qua đó giúp cho việc tạo ra một API trở nên dễ dàng và nhanh chóng.

Về hiệu suất, ExpressJS cung cấp thêm các tính năng giúp cho việc lập trình hiệu quả hơn chứ không làm giảm tốc độ của NodeJS.

ExpressJS là một trong những framework phía Backend phổ biến nhất hiện nay. Các framework nổi tiếng của NodeJS hiện nay đều sử dụng ExpressJS làm core như: SailsJS, MEAN,…

### 3.2.3 MongoDB

MongoDB [9] là một trong những cơ sở dữ liệu mã nguồn mở NoSQL phổ biến được viết bằng C++. MongoDB không lưu trữ dữ liệu theo dạng bảng, cột như mô hình dữ liệu quan hệ (SQL) mà dữ liệu được lưu dưới dạng bộ sưu tập (collections) và tài liệu (documents). Nó cho phép các document có các trường và cấu trúc khác nhau, linh hoạt trong việc lưu trữ dữ liệu.

Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau. Do vậy, việc thêm, sửa hay xoá sẽ không cần mất thời gian kiểm tra xem có thoả mãn các ràng buộc dữ liệu như trong các hệ cơ sở dữ liệu quan hệ.

MongoDB có hiệu năng cao, tốc độ truy vấn nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ. Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ thêm bản ghi của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

Do không có tính chất ràng buộc nên khi thao tác dữ liệu với MongoDB cũng phải rất cẩn thận, nếu không sẽ dẫn tới việc sai lệch trong dữ liệu.

### 3.2.4 Redis

Redis - (Remote Dictionary Server) [10] là một mã nguồn mở được dùng để lưu trữ dữ liệu có cấu trúc, nó có thể được sử dụng như một loại cơ sở dữ liệu NoSQL. Redis lưu trữ dữ liệu dạng key-value.

Ưu điểm nổi bật nhất của Redis là dữ liệu được lưu trữ trên RAM, nên việc truy xuất dữ liệu diễn ra rất nhanh chóng. Ngoài ra, nó cũng hỗ trợ tính năng lưu trữ dữ liệu trên đĩa cứng, cho phép hệ thống có thể phục hồi dữ liệu khi gặp sự cố.

### 3.2.5 RabbitMQ

RabbitMQ [11] là một message broker sử dụng giao thức AMQP (Advanced Message Queuing Protocol - Giao thức giao nhận tin nhắn sử dụng hàng đợi). Đây là một chương trình trung gian đóng vai trò lưu trữ và điều phối các yêu cầu (message) giữ người nhận (consumer) và người gửi (producer).

Đối với các hệ thống sử dụng kiến trúc Microservices thì việc giao tiếp giữa các service diễn ra khá nhiều, khiến cho luồng xử lý trở nên phức tạp. Điều này làm cho việc lập trình trở nên khó khăn. RabbitMQ được sử dụng để giải quyết những vấn đề như vậy.

Trong RabbitMQ, producer gửi tin nhắn đến các hàng đợi có trong message broker và consumer sẽ lắng nghe tin nhắn từ các hàng đợi này. Producer và consumer có thể được xây dựng khác nhau về mặt ngôn ngữ nhưng giao tiếp vẫn thành công do chúng kết nối với nhau thông qua message broker.

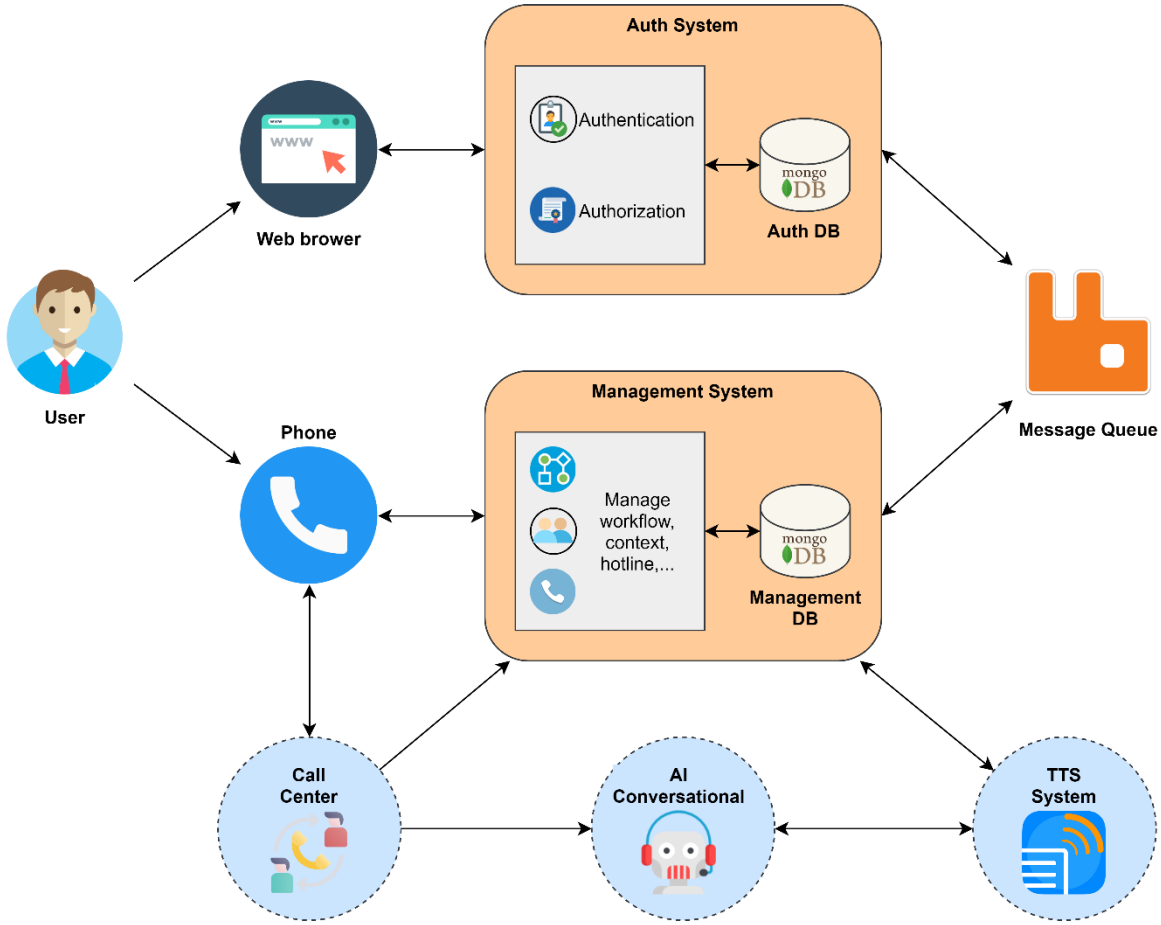
Trong Chương 3 em đã trình bày về các công nghệ để phát triển hệ thống. Phía frontend sử dụng thư viện ReactJS, Ant Design và Ej2 React Diagram. Phía backend sử dụng nền tảng NodeJS kết hợp với ExpressJS, MongoDB, Redis và RabbitMQ. Để hiểu rõ hơn về cấu trúc và cách thức hệ thống hoạt động, em xin trình bày cụ thể trong Chương 4.

# Chương 4 Phát triển và triển khai ứng dụng

Trong các chương trước, em đã khảo sát, phân tích yêu cầu phần mềm và tìm hiểu các công nghệ liên quan. Trong Chương 4, em trình bày từ việc thiết kế tổng quan cho đến thiết kế chi tiết từng hệ thống. Cùng với đó, em trình bày về kết quả đạt được và việc kiểm thử, triển khai hệ thống.

## 4.1 Thiết kế tổng quan

### 4.1.1 Kiến trúc tổng quan hệ thống




**Hình 11** Kiến trúc hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung.

**Hình 11** mô tả kiến trúc tổng quan của hệ thống hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung. Các hệ thống chính trong kiến trúc là (i) Hệ thống xác thực và phân quyền tập trung (Auth System), (ii) Hệ thống quản lý kịch bản cuộc gọi, (iii) hệ

thống gọi điện (Call Center), (iv) Hệ thống Bot (AI Conversational), và (v) Hệ thống chuyển văn bản thành giọng nói (TTS System). Trong đó, ba hệ thống (iii), (iv) và (v) được phát triển bởi Công ty cổ phần Dịch vụ và Giải pháp dữ liệu Vbee.

Hệ thống gọi điện có chức năng thực hiện các cuộc gọi ra và nhận các cuộc gọi đến từ khách hàng theo các kịch bản đã được cài đặt sẵn.

Hệ thống Bot là hệ thống có thể tự động giao tiếp, thực hiện chăm sóc khách hàng thay thế các tổng đài viên.

Hệ thống TTS có chức năng chuyển văn bản đầu vào thành giọng nói. Người dùng có thể điều chỉnh tốc độ của giọng nói và có thể chọn các loại giọng nói khác nhau. Hệ thống cung cấp một số loại giọng nói của nam, nữ đến từ Hà Nội và Sài Gòn.

Hệ thống xác thực và phân quyền tập trung có nhiệm vụ xác thực và phân quyền chung cho người dùng trong hệ thống tổng đài thông minh. Người dùng chỉ cần sử dụng một tài khoản duy nhất. Tài khoản này sẽ được dùng chung cho các ứng dụng được liên kết trong tổng đài như hệ thống gọi ra Out Bound và hệ thống gọi vào In Bound. Bên cạnh đó, người quản trị có thể tạo ra các vai trò với các quyền khác nhau để gán cho người dùng.

Hệ thống quản lý kịch bản cuộc gọi có chức năng quản lý liên hệ, quản lý các nhóm liên hệ, quản lý các đầu số hotline của tổng đài và quản lý kịch bản cuộc gọi. Khi người dùng gọi điện đến số hotline của tổng đài. Hệ thống gọi điện sẽ lấy kịch bản cuộc gọi từ Hệ thống quản lý cuộc gọi và xử lý cuộc gọi theo kịch bản đó.

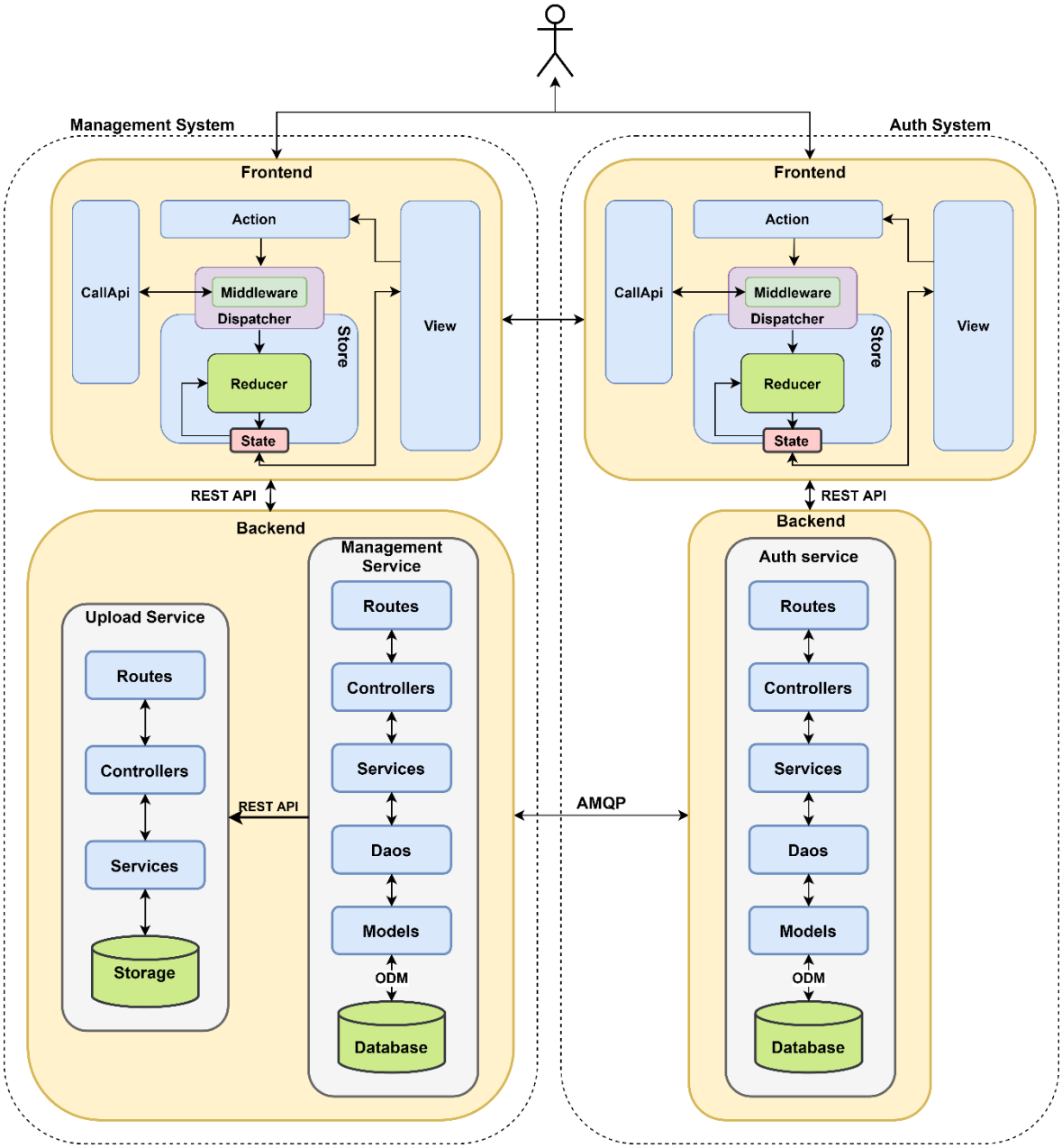
Hai hệ thống xác thực và phân quyền và hệ thống quản lý kịch bản cuộc gọi đồng bộ các thông tin về người dùng, portal và quyền qua hệ thống Message Queue. Thông tin chi tiết về hàng đợi em sẽ trình bày trong phần 4.2.2.2.

Trong phạm vi của đồ án, em thực hiện phát triển hai hệ thống: Hệ thống quản lý kịch bản cuộc gọi và Hệ thống xác thực và phân quyền tập trung. Khi người dùng truy cập vào hệ thống quản lý kịch bản, hệ thống xác thực và phân quyền sẽ kiểm tra xem người dùng đó là ai, có quyền gì trong hệ thống, từ đó hệ thống quản lý kịch bản sẽ hiển thị nội dung tương ứng với quyền hạn của người dùng.

### 4.1.2 Kiến trúc tổng quan phần mềm

Kiến trúc phần mềm của Hệ thống quản lý kịch bản cuộc gọi và Hệ thống xác thực và phân quyền tập trung được thể hiện ở **Hình 12**. Mỗi hệ thống được chia ra thành hai phần frontend và backend riêng biệt, giao tiếp với nhau qua REST API. Chính vì thế, việc thay đổi kiến trúc hoặc thay đổi công nghệ ở backend không ảnh hưởng đến frontend. Tương tự, việc thay

đổi kiến trúc hoặc công nghệ ở frontend cũng không ảnh hưởng đến backend. Từ đó, việc phát triển hệ thống trở nên linh hoạt hơn, có thể dễ dàng mở rộng hệ thống sau này.



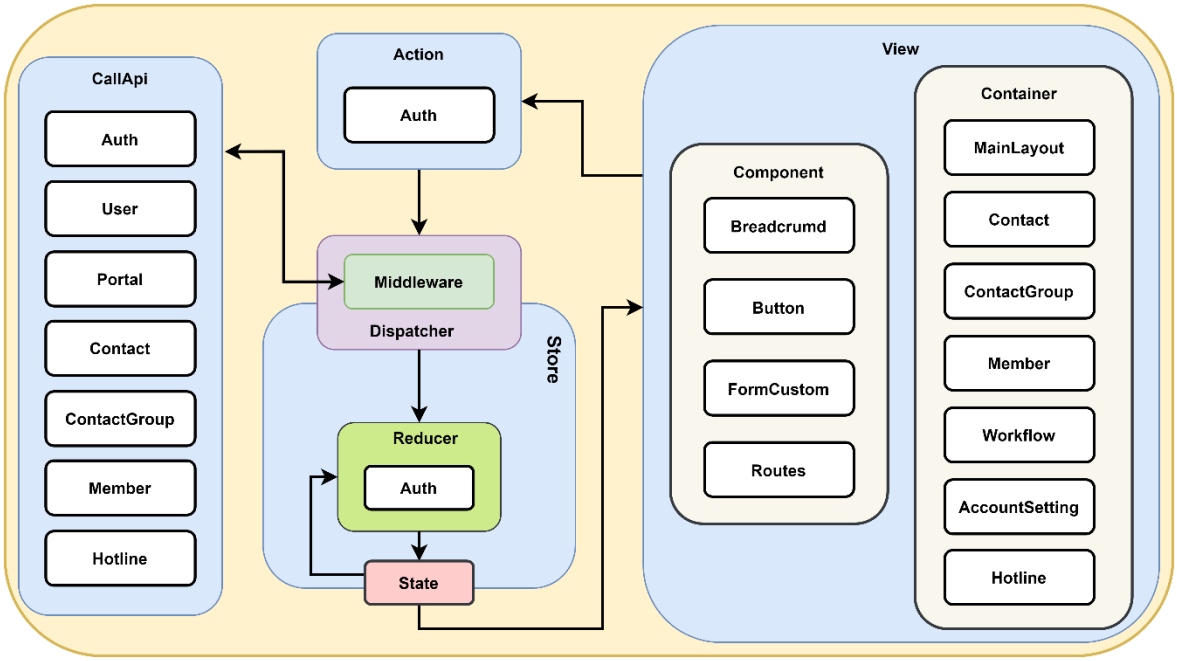

**Hình 12** Kiến trúc tổng quan hệ thống.

Em sẽ trình bày chi tiết hệ thống quản lý kịch bản cuộc gọi trong phần 4.2, hệ thống xác thực và phân quyền tập trung trong phần 4.3.

## 4.2 Hệ thống quản lý kịch bản cuộc gọi

### 4.2.1 Thiết kế kiến trúc

#### 4.2.1.1 Thiết kế kiến trúc frontend




**Hình 13** Thiết kế kiến trúc frontend hệ thống quản lý kịch bản cuộc gọi.

Kiến trúc frontend của hệ thống quản lý kịch bản cuộc gọi được thể hiện qua **Hình 13**. Hệ thống sử dụng thư viện React kết hợp với Redux để quản lý trạng thái của ứng dụng. Khi cần thay đổi trạng thái ứng dụng, các thành phần giao diện của View sẽ gọi đến các hành động được định nghĩa sẵn ở Action. Khi đó, Reducer sẽ nhận được sự kiện và cập nhật trạng thái tương ứng. Nếu hành động đó cần tương tác với máy chủ, Middleware sẽ thực hiện gọi và nhận dữ liệu rồi trả về cho Reducer xử lý.

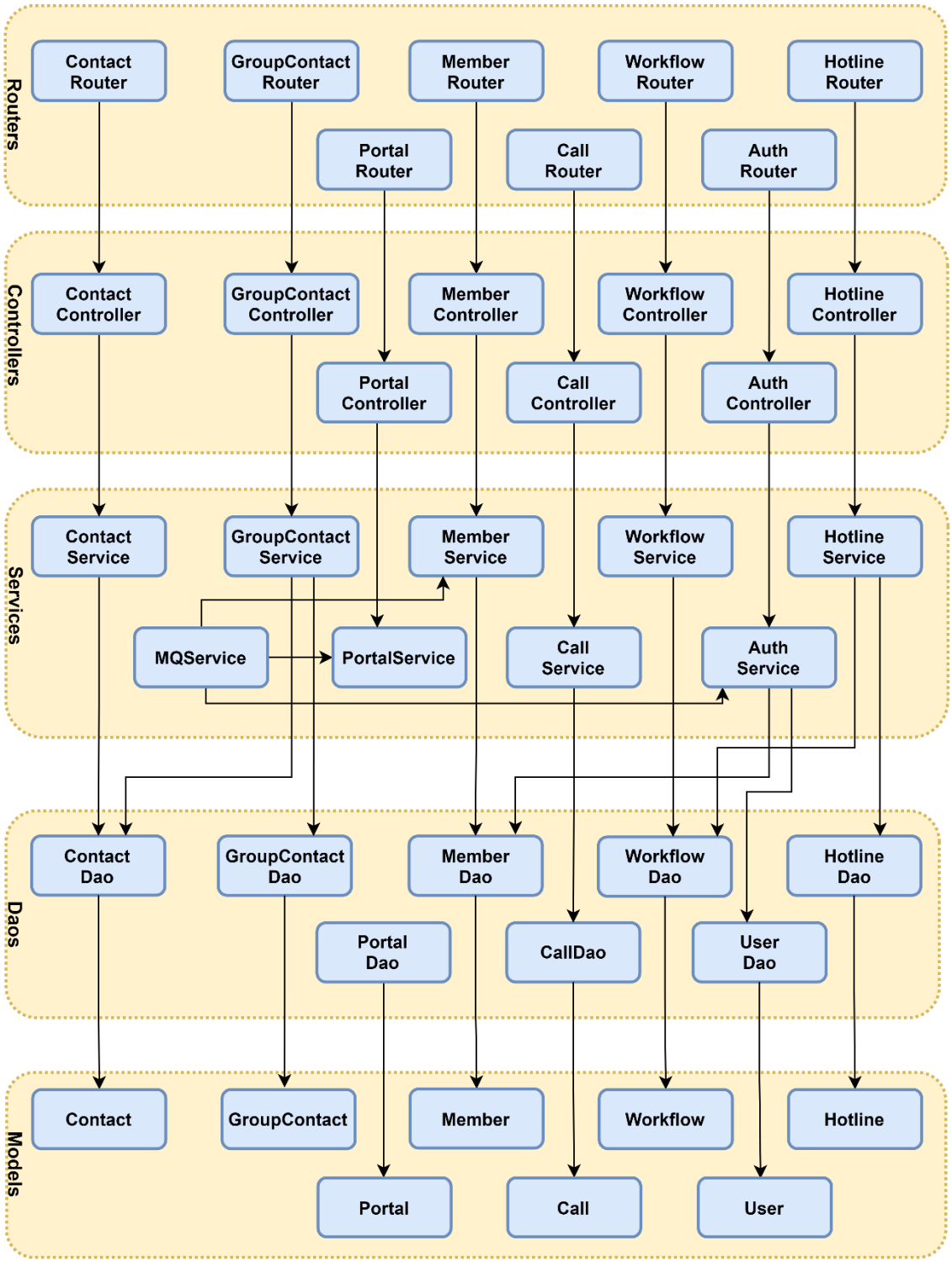
#### 4.2.1.2 Thiết kế kiến trúc backend

**a) Thiết kế kiến trúc backend dịch vụ quản lý**

**Hình 14** mô tả kiến trúc tổng quan backend của hệ thống quản lý kịch bản cuộc gọi. Phần backend được xây dựng bằng NodeJS, được chia thành các phần: Routers, Controllers, Services, Daos và Models. Phần backend là nơi xử lý các request mà client gửi đến.

Khi client gửi request đến server, các Routers có nhiệm vụ định tuyến, gọi đến các Controllers tương ứng để xử lý. Các Controllers tiếp nhận yêu cầu từ Routers, sau đó gọi đến các Services để xử lý về mặt nghiệp vụ. Daos là nơi định nghĩa các phương thức chung để thao tác với cơ sở dữ liệu. Vì vậy, các Services không gọi thẳng đến các Models để tương

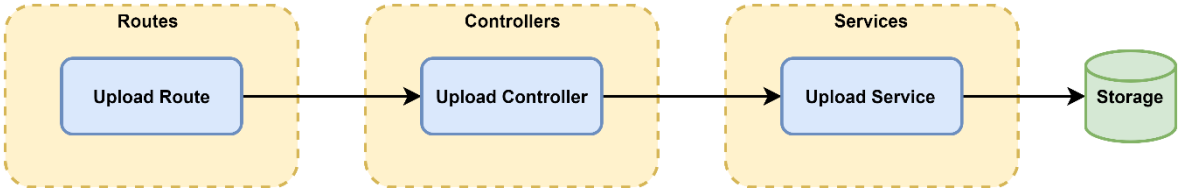
tác với cơ sở dữ liệu mà gọi qua thành phần Daos. Điều này giúp phân tách logic cho các thành phần, các services chỉ thực hiện xử lý về nghiệp vụ, các thành phần Daos chỉ thực hiện thao tác với cơ sở dữ liệu.




**Hình 14** Thiết kế kiến trúc backend dịch vụ quản lý.

**b) Thiết kế kiến trúc backend dịch vụ lưu trữ**

Dịch vụ lưu trữ có chức năng lưu trữ các tệp tin đa phương tiện. Thiết kế kiến trúc backend của dịch vụ này được thể hiện trong **Hình 15**. Dịch vụ được chia ra thành ba phần đó là Routes, Controllers và Services tương tự như dịch vụ quản lý. Dịch vụ này không kết nối đến cơ sở dữ liệu mà sử dụng kho lưu trữ (Storage) để lưu trữ dữ liệu.

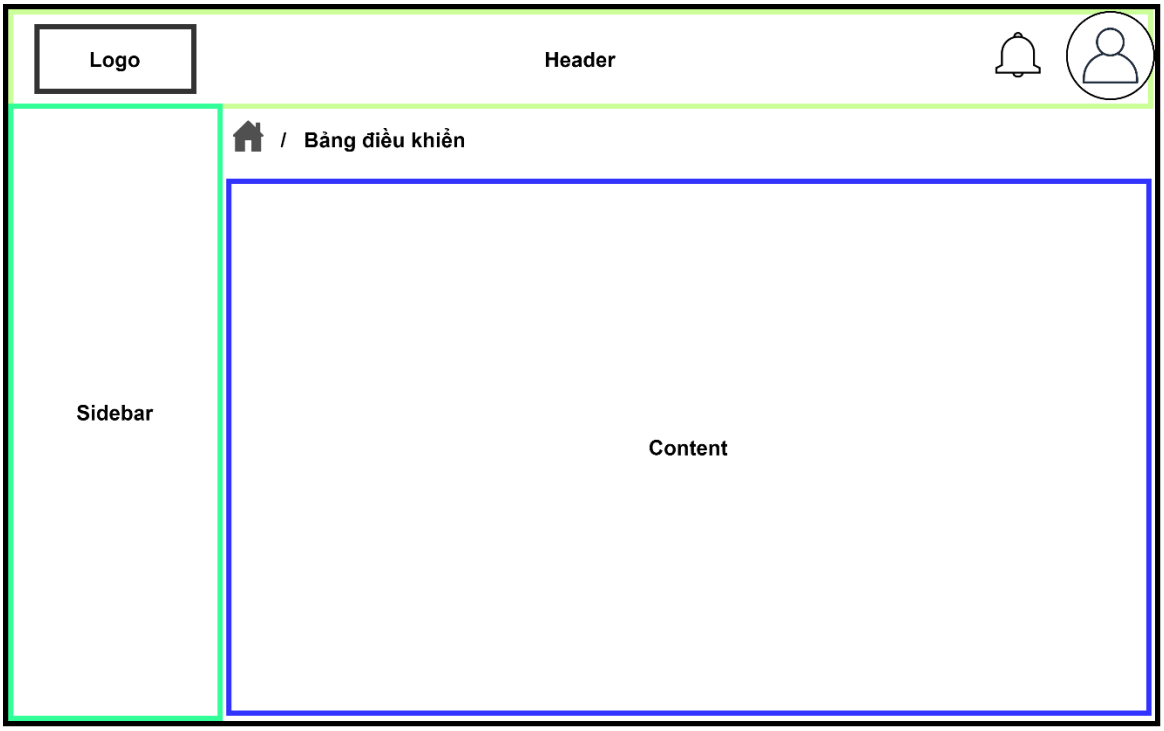



**Hình 15** Thiết kế kiến trúc backend dịch vụ lưu trữ.

### 4.2.2 Thiết kế chi tiết

#### 4.2.2.1 Thiết kế chi tiết frontend

**a) Thiết kế mockup**




**Hình 16** Thiết kế mockup hệ thống quản lý kịch bản cuộc gọi.

Giao diện hệ thống quản lý kịch bản cuộc gọi sử dụng thư viện ReactJS nên sẽ được phân chia thành các component khác nhau. Em sử dụng độ phân giải phổ biến, thường dùng cho các thiết bị hiện nay là 1920 x 1080 để thiết kế. Bố cục trang web được chia thành 3 phần chính đó là Header, Sidebar và Content như **Hình 16**.

Tất cả các màn hình của hệ thống quản lý kịch bản cuộc gọi đều sử dụng thống nhất hai phần Header và Sidebar, chỉ phần Content thay đổi để hiển thị nội dung tương ứng. Header là thanh trên cùng của trang web. Nó hiển thị logo của trang web, các thông báo và các thông tin cơ bản của người dùng. Sidebar là phần phía bên trái của trang web. Ở đây người dùng có thể dễ dàng điều hướng trang web đến các chức năng mong muốn.

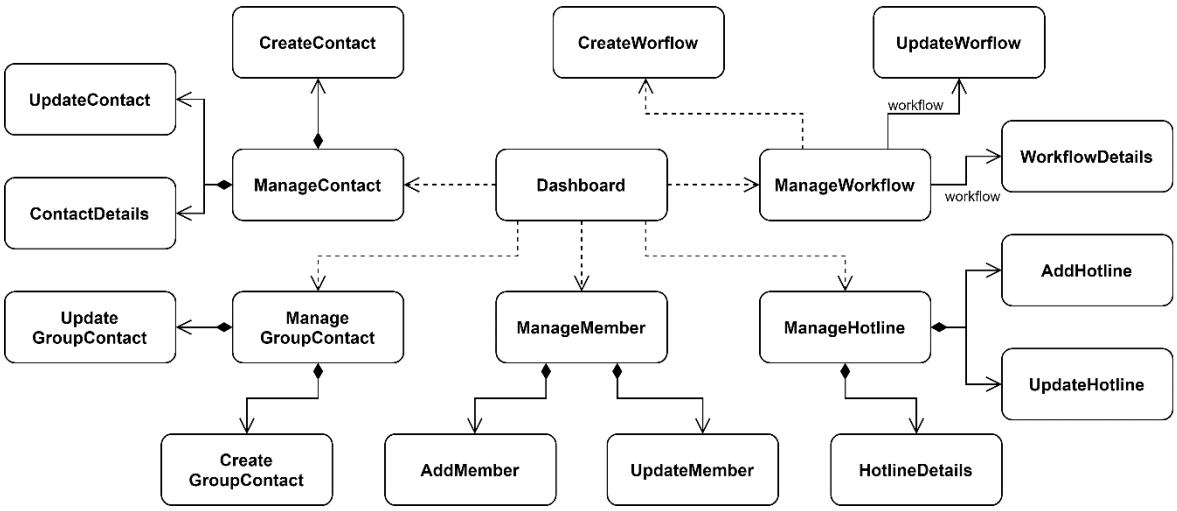
Toàn bộ trang web được sử dụng chung các cấu hình thuộc tính như **Bảng 16** để đảm bảo tính nhất quán.

**Bảng 16** Cấu hình chung cho giao diện quản lý kịch bản cuộc gọi

| Thuộc tính | Cấu hình |
|---|---|
| Màu sắc | Màu chính: #2B6CB0<br>Màu thông báo thành công: #48BB78<br>Màu cảnh báo: #ECC94B<br>Màu thông báo lỗi: #E53E3E |
| Font chữ | Kiểu chữ: -apple-system, BlinkMacSystemFont, Segoe UI<br>Cỡ chữ phần nội dung: 14px |
| Đổ bóng | 0 0px 8px 4px rgba(0, 0, 0, 0.15) |
| Bo góc | 1px |
| Vị trí hiển thị thông báo | Góc trên bên phải màn hình |

**b) Biểu đồ dịch chuyển màn hình**

Từ màn hình chính, người dùng có thể điều hướng sang các màn hình khác nhau tùy vào quyền của mình trong hệ thống như **Hình 17**. Hệ thống có một số màn hình chính như: màn hình điều khiển, màn hình quản lý liên hệ, quản lý nhóm liên hệ, quản lý người dùng, quản lý số hotline và quản lý kịch bản cuộc gọi.

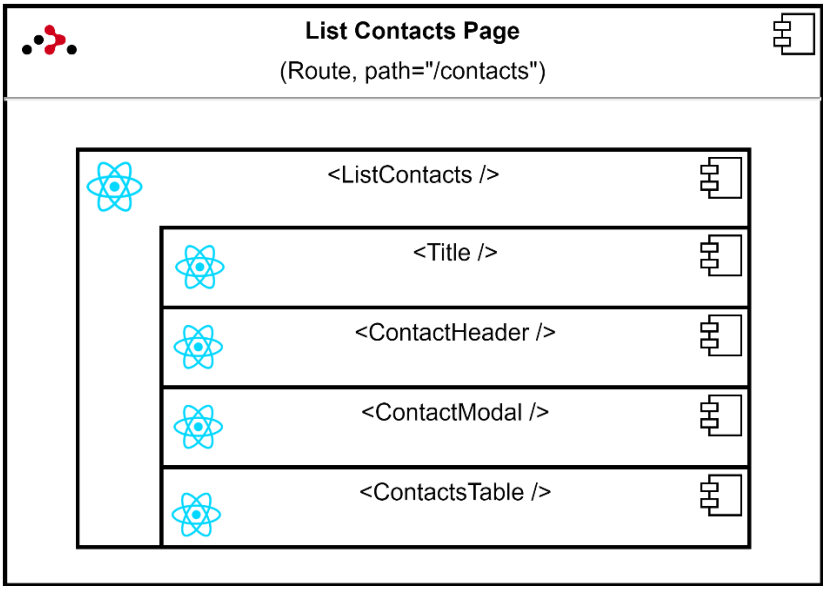



**Hình 17** Biểu đồ dịch chuyển màn hình hệ thống quản lý kịch bản.

### c) Thiết kế thành phần giao diện

Trong phần này, em sẽ trình bày thiết kế giao diện của ba màn hình chính, đó là: danh sách liên hệ, quản lý người dùng và tạo kịch bản cuộc gọi.

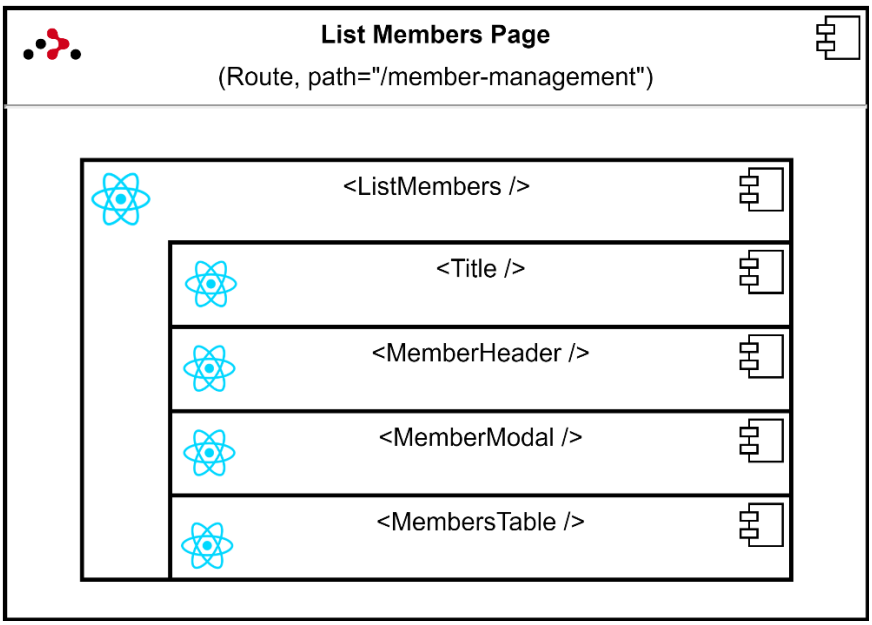
Màn hình danh sách liên hệ bao gồm bốn thành phần chính như **Hình 18**. Thành phần Title hiển thị tiêu đề của trang. ContactHeader là thành phần chứa thanh tìm kiếm, nút tải tệp danh sách liên hệ, nút tạo liên hệ mới. ContactModal là thành phần để hiển thị form xem chi tiết, thêm hoặc cập nhật thông tin liên hệ. Thành phần ContactTable có chức năng hiển thị danh sách các liên hệ dưới dạng bảng.




**Hình 18** Thiết kế thành phần giao diện màn hình danh sách liên hệ.

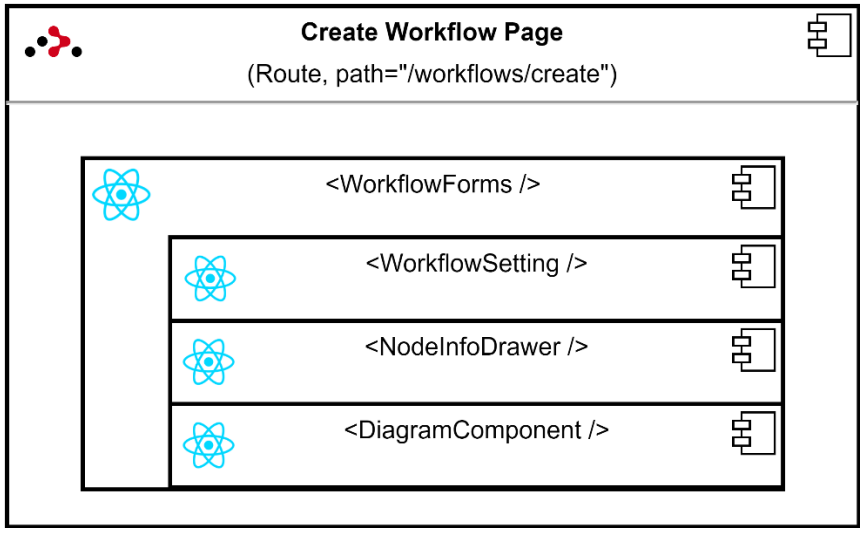
Màn hình quản lý người dùng gồm bốn thành phần chính như **Hình 19**, đó là: Title, MemberHeader, MemberModal, MembersTable. Thành phần MemberHeader nằm dưới tiêu

đề trang (Title) là nơi hiển thị thanh tìm kiếm, nút mời người dùng vào portal. MemberModal là form có chức năng thêm, cập nhật thông tin người dùng. Thành phần MembersTable có chức năng hiển thị danh sách người dùng trong portal dưới dạng bảng.




**Hình 19** Thiết kế thành phần giao diện màn hình quản lý người dùng.

**Hình 20** thể hiện các thành phần giao diện của màn hình tạo kịch bản cuộc gọi. WorkflowSetting là một form để người dùng nhập các thông tin cấu hình cho kịch bản cuộc gọi, bao gồm: tên kịch bản, nhạc nền, giọng nói, tốc độ đọc, có sử dụng chức năng kịch bản nâng cao hay không. NodeInfoDrawer là thành phần sẽ hiển thị khi người dùng yêu cầu thêm node vào kịch bản. NodeInfoDrawer chứa các thông tin: tên node, phím bấm và các thẻ hành động được chứa trong node đó. Thành phần DiagramComponent có chức năng hiển thị kịch bản cuộc gọi dưới dạng biểu đồ luồng. Người dùng có thể kéo thả, nhấn chuột để tạo ra kịch bản cuộc gọi mong muốn.




**Hình 20** Thiết kế thành phần giao diện màn hình tạo kịch bản cuộc gọi.

#### 4.2.2.2 Thiết kế chi tiết backend

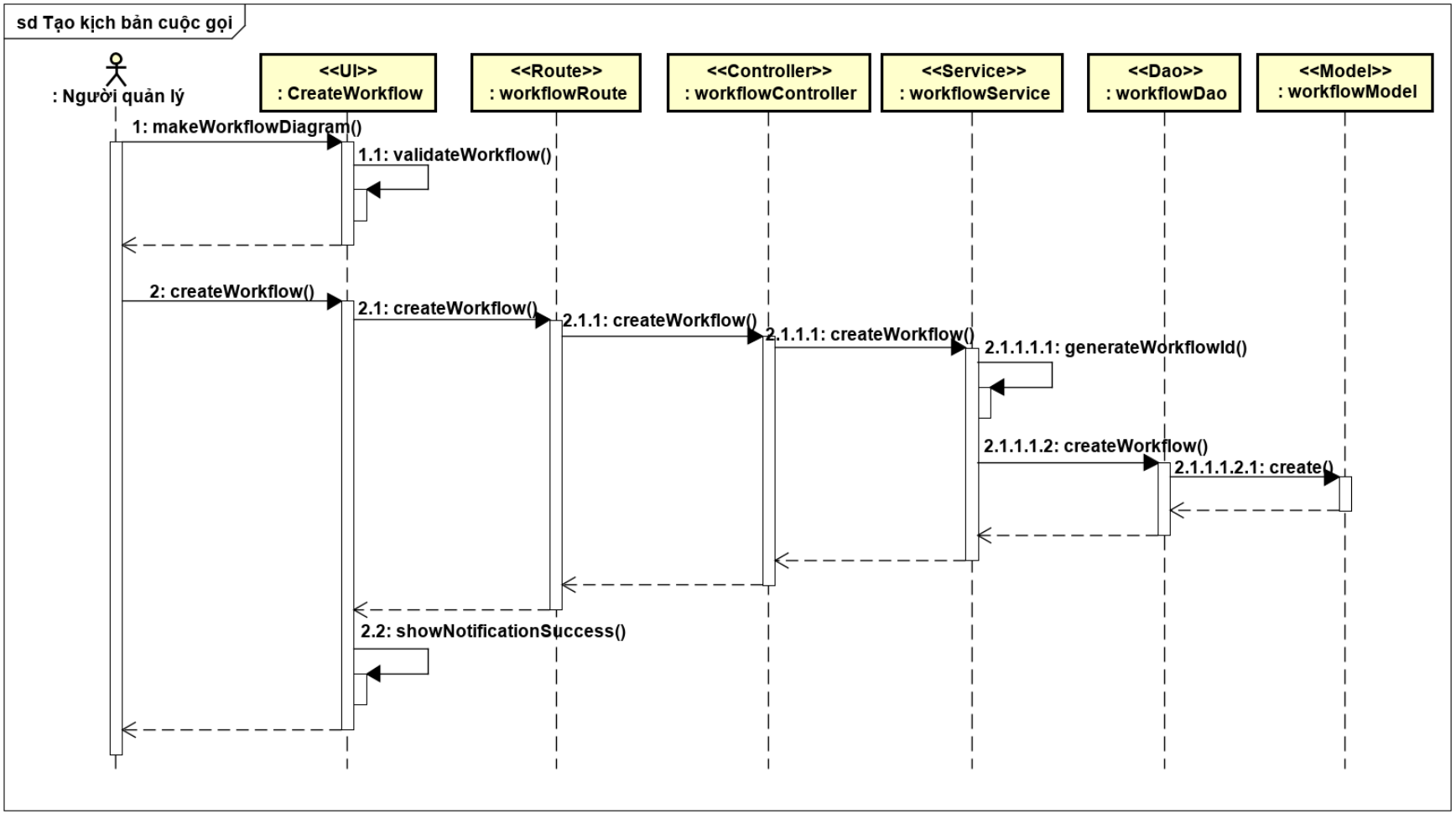
**a) Luồng hoạt động**

Trong phần này, em xin trình bày luồng hoạt động khi người quản lý tạo kịch bản cuộc gọi như **Hình 21**.

Trước hết, người dùng thao tác trên giao diện để tạo ra một biểu đồ luồng. Người dùng cần nhập thông tin của kịch bản, sau đó tạo ra các node, cấu hình thông tin cho node đó, tiếp tục như vậy cho đến khi hoàn tất biểu đồ luồng. Trong quá trình thao tác, các thành phần UI sẽ tự động kiểm tra dữ liệu và hiển thị thông báo lỗi nếu có.

Sau khi người dùng gửi yêu cầu tạo kịch bản cuộc gọi, các thông tin về kịch bản sẽ được gửi về phần backend qua giao thức HTTP để xử lý. Các Router sẽ định tuyến và gọi đến workfowController để xử lý, sau đó Controller gọi đến Service, Service gọi đến Dao và sau cùng là Model để tạo kịch bản cuộc gọi. Thông tin kịch bản sẽ được lưu vào trong cơ sở dữ liệu.

Sau khi backend xử lý xong, response được gửi trả về cho phía giao diện. Giao diện hiển thị thông báo tạo kịch bản cuộc gọi thành công cho người dùng và chuyển hướng sang giao diện danh các kịch bản cuộc gọi, kịch bản mới tạo sẽ được đưa lên đầu danh sách.




**Hình 21** Biểu đồ trình tự tạo kịch bản cuộc gọi.

**b) Thiết kế API**

Như đã trình bày trong mục 4.1.24.1, các dịch vụ trong hệ thống quản lý kịch bản người dùng giao tiếp với nhau qua REST API. Cụ thể, trong **Bảng 17**, em sẽ đưa ra danh sách các API chính mà hệ thống cung cấp.

**Bảng 17** Danh sách API của hệ thống quản lý kịch bản cuộc gọi

| STT | Nhóm API | Mục đích | Phương thức | Địa chỉ |
|---|---|---|---|---|
| **Base url:** /api/v1 | | | | |
| 1. | Quản lý kịch bản cuộc gọi | Tạo kịch bản cuộc gọi | POST | /workflows |
| 2. | | Lấy danh sách kịch bản cuộc gọi | GET | /workflows |
| 3. | | Lấy thông tin chi tiết kịch bản cuộc gọi | GET | /workflows/:id |
| 4. | | Cập nhật thông tin kịch bản cuộc gọi | PUT | /workflows/:id |
| 5. | | Xóa kịch bản cuộc gọi | DELETE | /workflows/:id |
| 6. | | Lấy kịch bản cuộc gọi với nội dung cá nhân hóa ứng với khách hàng | GET | /workflow/personalized |
| 7. | Quản lý người dùng | Thêm người dùng vào hệ thống | POST | /members |
| 8. | | Lấy danh sách người dùng trong hệ thống | GET | /members |

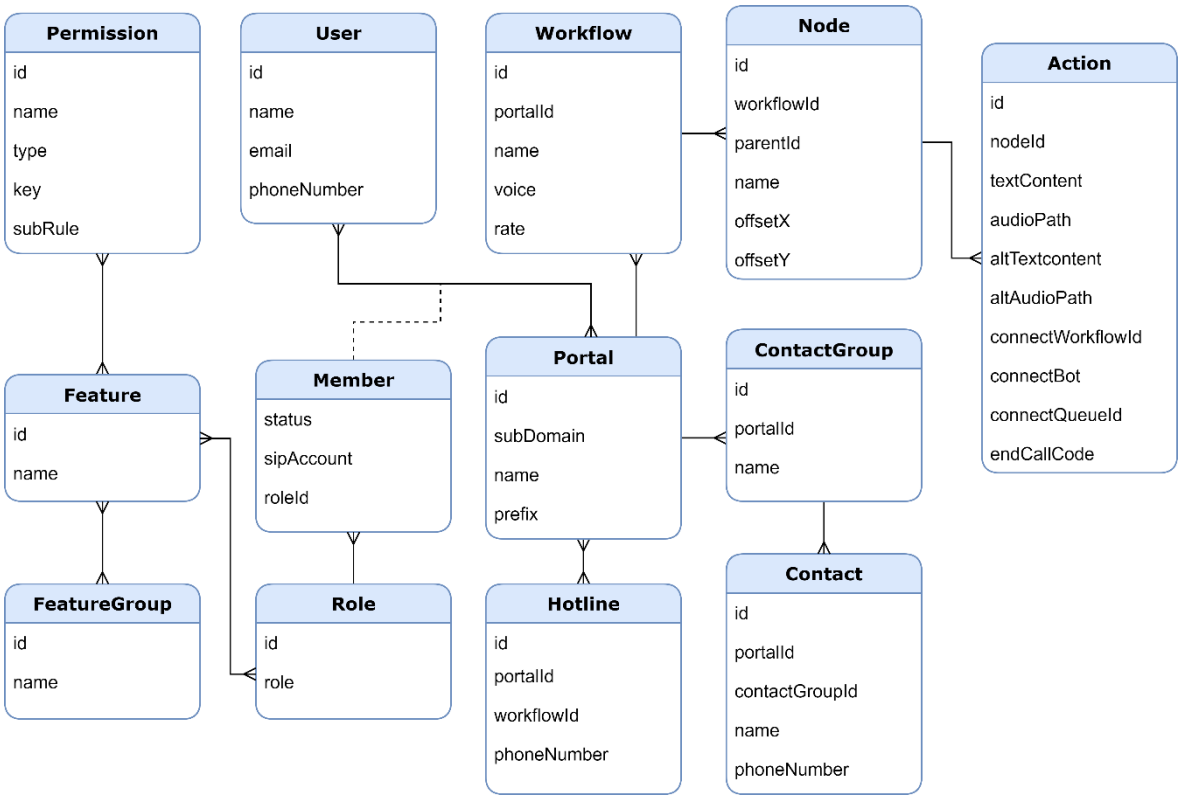
| STT | Nhóm API | Mục đích | Phương thức | Địa chỉ |
|---|---|---|---|---|
| 9. | | Lấy thông tin chi tiết người dùng | GET | /members/:id |
| 10. | | Cập nhật thông tin người dùng | PUT | /members/:id |
| 11. | | Xóa người dùng khỏi hệ thống | DELETE | /members/:id |
| 12. | Quản lý liên hệ | Thêm liên hệ mới | POST | /contacts |
| 13. | | Lấy danh sách liên hệ | GET | /contacts |
| 14. | | Xem thông tin chi tiết liên hệ | GET | /contacts/:id |
| 15. | | Cập nhật thông tin liên hệ | PUT | /contacts/:id |
| 16. | | Xóa liên hệ | DELETE | /contacts/:id |
| 17. | | Tải lên tệp danh sách liên hệ | POST | /contacts/upload |
| 18. | Quản lý nhóm liên hệ | Tạo nhóm liên hệ | POST | /group-contacts |
| 19. | | Lấy danh sách nhóm liên hệ | GET | /group-contacts |
| 20. | | Xem chi tiết nhóm liên hệ | GET | /group-contacts/:id |
| 21. | | Cập nhật nhóm liên hệ | PUT | /group-contacts/:id |
| 22. | | Xóa nhóm liên hệ | DELETE | /group-contacts/:id |

Dịch vụ quản lý và dịch vụ xác thực & phân quyền giao tiếp với nhau thông qua hệ thống Message Queue như phần 4.1.2 em đã mô tả. Trong **Bảng 18**, em đưa ra danh sách các hàng đợi sử dụng để trao đổi dữ liệu giữa hai dịch vụ quản lý và dịch vụ xác thực & phân quyền.

**Bảng 18** Danh sách hàng đợi trong hệ thống quản lý kịch bản

| STT | Hàng đợi | Mô tả |
|---|---|---|
| 1. | USER_QUEUE | Hàng đợi đồng bộ thông tin người dùng |
| 2. | PORTAL_QUEUE | Hàng đợi đồng bộ thông tin portal |
| 3. | PERMISSION_QUEUE | Hàng đợi đồng bộ thông tin quyền |
| 4. | FEAUTURE_QUEUE | Hàng đợi đồng bộ thông tin tính năng |
| 5. | FEATURE_GROUP_QUEUE | Hàng đợi đồng bộ thông tin nhóm tính năng |
| 6. | ROLE_QUEUE | Hàng đợi đồng bộ thông tin vai trò |

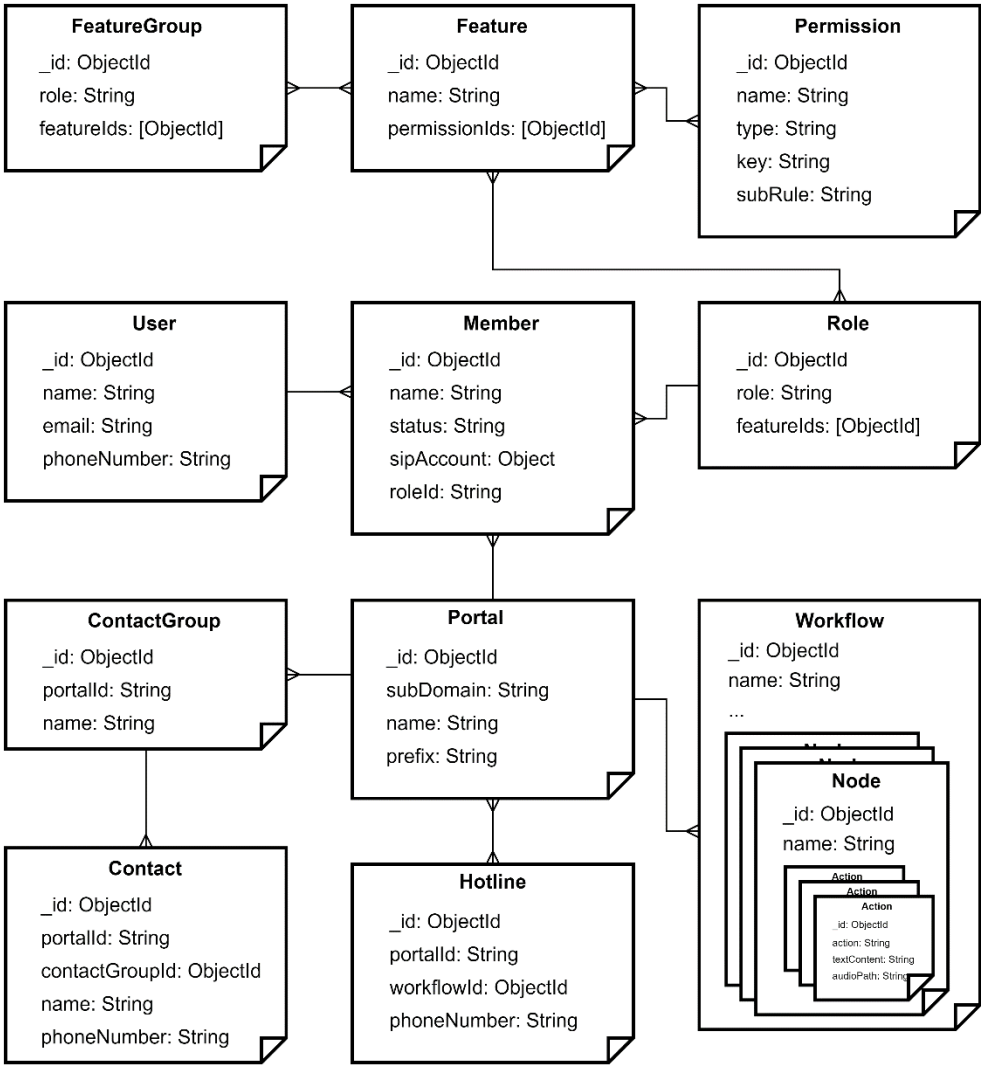
### 4.2.2.3 Thiết kế chi tiết cơ sở dữ liệu




**Hình 22** Biểu đồ thực thể liên kết hệ thống quản lý kịch bản cuộc gọi.

**Hình 22** là biểu đồ thực thể liên kết của hệ thống quản lý cuộc cuộc gọi. Một người dùng có thể quản lý nhiều portal khác nhau. Trong portal, người dùng có thể quản lý các liên hệ và nhóm liên hệ, quản lý kịch bản, quản lý số hotline của tổng đài.

Từ biểu đồ thực thể liên kết, em đã thiết kế biểu đồ cơ sở dữ liệu dạng NoSQL cho cơ sở dữ liệu MongoDB như **Hình 23**.




**Hình 23** Biểu đồ cơ sở dữ liệu hệ thống quản lý kịch bản cuộc gọi.

Cơ sở dữ liệu hệ thống quản lý kịch bản cuộc gọi gồm 5 collection chính. Đó là: Member, Hotline, ContactGroup, Contact và Workflow. Các collection khác như Portal, User, Permission, Feature, FeatureGroup và Role được đồng bộ từ hệ thống xác thực và phân quyền tập trung, em sẽ đặc tả trong phần 4.3.2.3. Thiết kế chi tiết cơ sở dữ liệu cho hệ thống quản lý kịch bản cuộc gọi được trình bày trong **Bảng 19**.

**Bảng 19** Thiết kế chi tiết cơ sở dữ liệu hệ thống quản lý kịch bản cuộc gọi

| Collection | Tên trường | Kiểu dữ liệu | Ý nghĩa |
|---|---|---|---|
| User | _id | ObjectId | ID của người dùng trong hệ thống |
| | name | String | Tên người người dùng |
| | email | String | Email của người dùng |
| | phoneNumber | String | Số điện thoại của người dùng |
| Portal | _id | String | ID của portal trong hệ thống |
| | name | String | Tên của portal |
| | subDomain | String | Tên miền phụ của portal |
| | prefix | String | Tiền tố của portal, phục vụ cho việc tạo ra các ID sau này |
| Contact Goup | _id | ObjectId | ID của nhóm người dùng |
| | portalId | ObjectId | ID của portal chứa nhóm người dùng |
| | Name | String | Tên nhóm người dùng |
| Contact | _id | String | ID của liên hệ |
| | groupId | ObjectId | ID của nhóm liên hệ |
| | portalId | ObjectId | ID của portal mà liên hệ thuộc |
| | phoneNumber | String | Số điện thoại của liên hệ |
| | name | String | Tên liên hệ |
| Hotline | _id | ObjectId | ID của số hotline |
| | portalId | ObjectId | ID của portal sử dụng số hotline |

| | | | |
|---|---|---|---|
| | workflowId | ObjectId | ID của kịch bản tương ứng với số hotline |
| | phoneNumber | String | Số điện thoại của hotline |
| Member | _id | ObjectId | ID của thành viên |
| | Name | String | Tên của thành viên |
| | sipAccount | Object | Tài khoản sip của thành viên để thực hiện các cuộc gọi |
| | roleId | String | ID của vai trò mà thành viên được gán |

Đối với collection Workflow, em sẽ trình bày chi tiết trong mục 5.1.2.3.

### 4.2.3 Xây dựng ứng dụng

#### 4.2.3.1 Thư viện và công cụ sử dụng

Trong quá trình thực hiện xây dựng hệ thống quản lý kịch bản cuộc gọi, em có sử dụng một số thư viện và công cụ hỗ trợ được nêu ra ở **Bảng 20**.

**Bảng 20** Thư viện và công cụ sử dụng phát triển hệ thống quản lý kịch bản cuộc gọi

| | **Công cụ** | **Mục đích** | **Địa chỉ URL** |
|---|---|---|---|
| Công cụ chung | Visual Studio Code | IDE lập trình | https://code.visualstudio.com |
| | Docker | Triển khai hệ thống | https://www.docker.com |
| | Axios | Thư viện gửi HTTP request | https://github.com/axios/axios |
| | Javascript | Ngôn ngữ lập trình | https://developer.mozilla.org/en-US/docs/Web/JavaScript |
| Công cụ sử dụng phía backend | NodeJS | Nền tảng xây dựng backend | https://nodejs.org |
| | Express | Framework xây dựng backend | https://expressjs.com |

| | | | |
|---|---|---|---|
| | Mongoose | Thư viện mô hình hóa đối tượng thao tác với CSDL MongoDB | https://mongoosejs.com |
| | Robo3T | Công cụ quản trị cơ sở dữ liệu | https://robomongo.org |
| | Express | Framework xây dựng backend | https://expressjs.com |
| | Amqplib | Thư viện hỗ trợ kết nối với RabbitMQ | https://www.npmjs.com/package/amqplib |
| Công cụ sử dụng phía frontend | ReactJS | Thư viện xây dựng frontend | https://ReactJS.org |
| | ReduxJS | Thư viện quản lý trạng thái cho ứng dụng Javascript | https://redux.js.org |
| | Styled Components | Thư viện viết CSS cho ứng dụng React | https://styled-components.com |
| | Moment | Thư viện xử lý thời gian | https://momentjs.com |
| | EJ2 React Diagrams | Thư viện hỗ trợ vẽ biểu đồ cho ứng dụng React | https://ej2.syncfusion.com/ |

#### 4.2.3.2 Kết quả đạt được

Trong quá trình thực hiện đồ án, em đã phát triển được hệ thống quản lý kịch bản cuộc gọi cho tổng đài thông minh. Khi khách hàng gọi đến số hotline, nếu thông tin khách hàng có sẵn trong hệ thống, nội dung cuộc gọi có thể được cá nhân hóa. Hệ thống có một số chức năng đặc biệt trong quá trình cấu hình như: (i) sử dụng công nghệ chuyển văn bản thành giọng nói để tạo ra thông tin cá nhân hóa với từng khách hàng, (ii) kết nối đến các voice Bot để trao đổi, (iii) sử dụng tài liệu âm thanh có sẵn, (iv) kết nối đến tổng đài viên và (v) chuyển đến một kịch bản khác. Hệ thống giúp người dùng có thể dễ dàng tạo ra các kịch bản cho

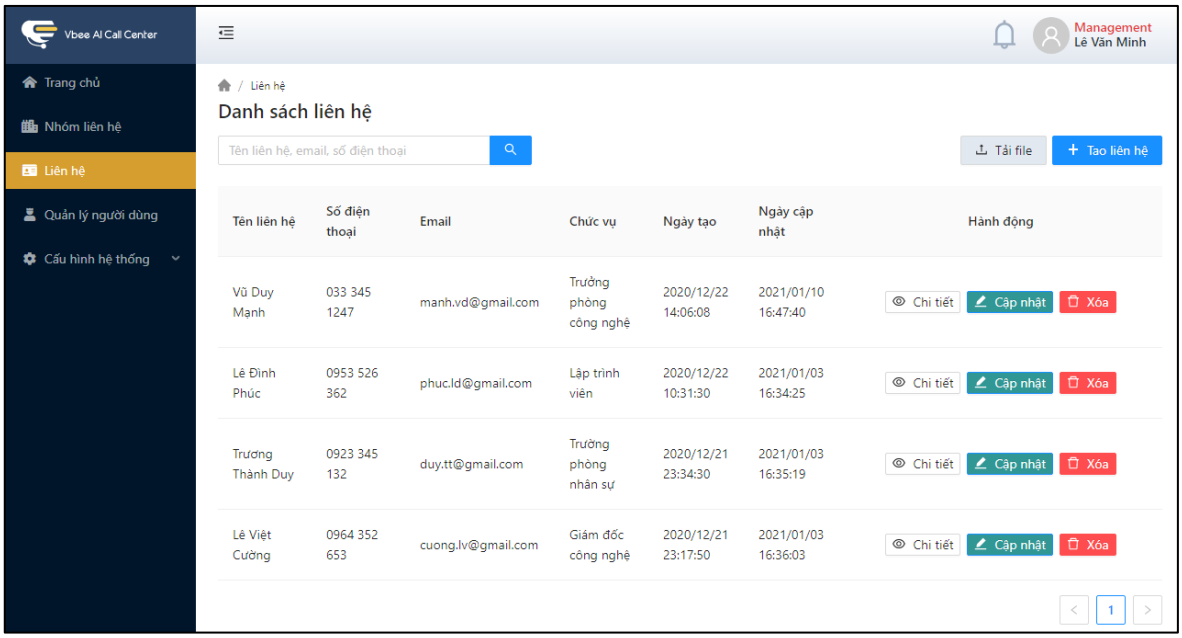
cuộc gọi, quản lý liên hệ, quản lý tổng đài viên, xem các báo cáo thống kê. Hiện tại, hệ thống được triển khai thử nghiệm tại https://dashboard-inbound.iristech.club.

#### 4.2.3.3 Minh họa các chức năng chính

Trong phần này, em sẽ trình bày về một số chức năng chính của hệ thống quản lý kịch bản cuộc gọi, đó là: quản lý liên hệ, quản lý người dùng, quản lý kịch bản cuộc gọi và quản lý số hotline.

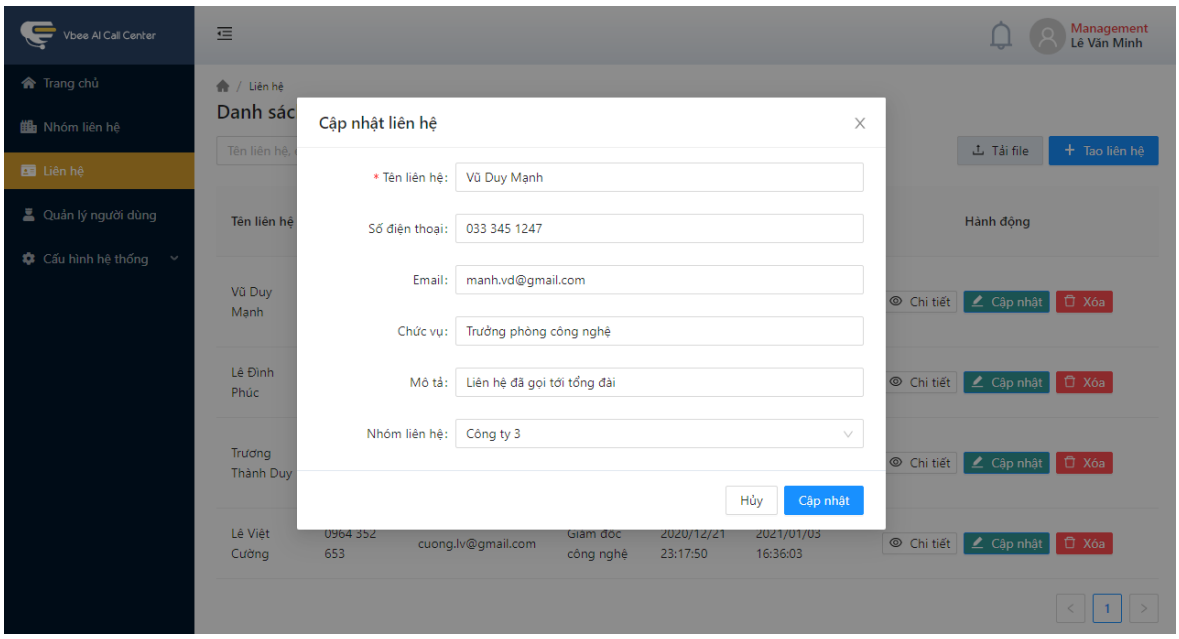
**a) Quản lý liên hệ**

**Hình 24** mô tả giao diện của màn hình danh sách liên hệ. Danh sách liên hệ được trình bày dưới dạng bảng. Phía trên của bảng là trường dữ liệu phục vụ cho chức năng tìm kiếm và các nút bấm cho thao tác tạo liên hệ bằng file Excel hoặc bằng cách nhập thông tin. Bảng trình bày các thông tin của liên hệ bao gồm tên liên hệ, số điện thoại, email, chức vụ, ngày tạo và ngày cập nhật. Bên phải của bảng là các thao tác bao gồm xem chi tiết, cập nhật liên hệ hoặc xóa liên hệ. Bảng được phân trang, người dùng có thể lựa chọn số trang cần theo dõi ở thanh bên phải phía dưới của bảng.



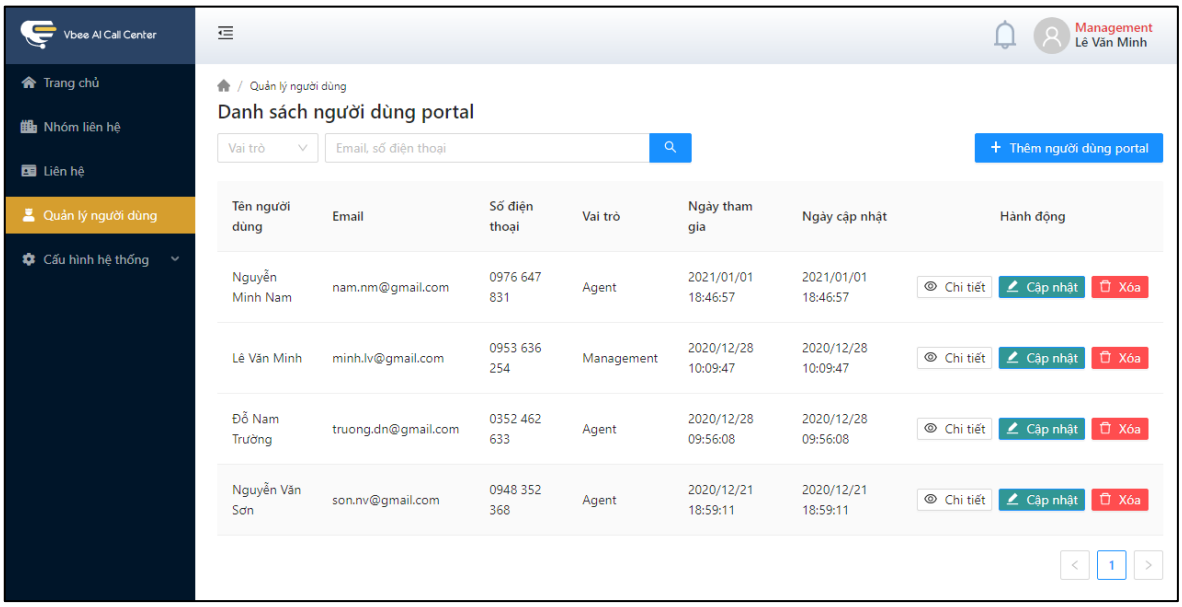

**Hình 24** Giao diện danh sách liên hệ.

**Hình 25** mô tả giao diện của hộp thoại tạo liên hệ sau khi người chọn chức năng Cập nhật liên hệ trên màn hình danh sách. Các trường thông tin cho người dùng nhập bao gồm tên liên hệ, số điện thoại, email, chức vụ, mô tả. Email là trường thông tin bắt buộc. Trường nhóm liên hệ được lựa chọn từ danh sách. Các trường thông tin được xác thực ngay trên giao diện trước khi gọi API cập nhật liên hệ.



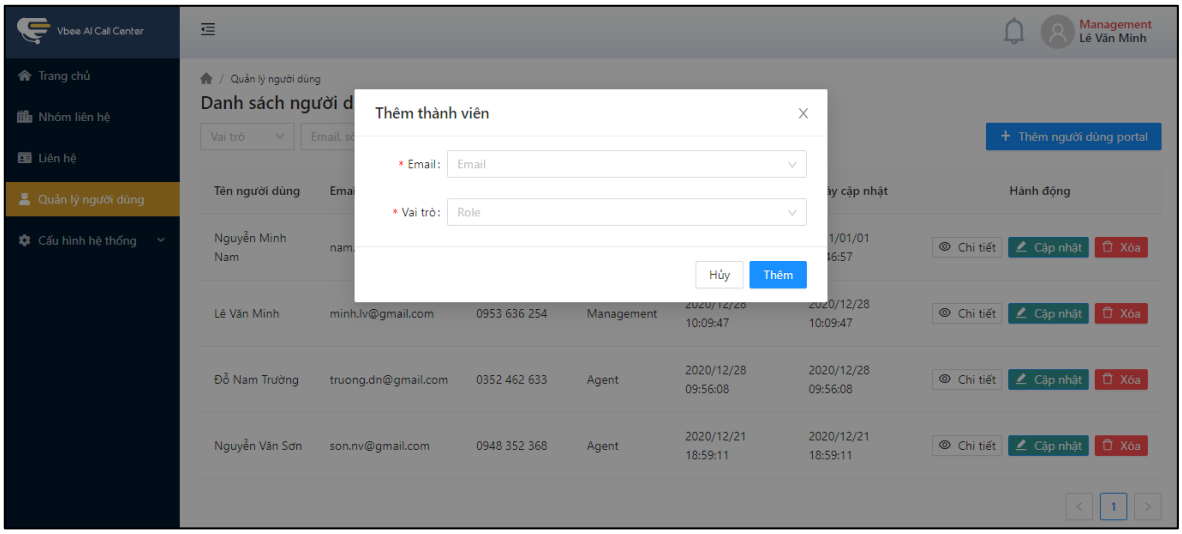

**Hình 25** Giao diện cập nhật thông tin liên hệ.

### b) Quản lý người dùng




**Hình 26** Giao diện danh sách người dùng.

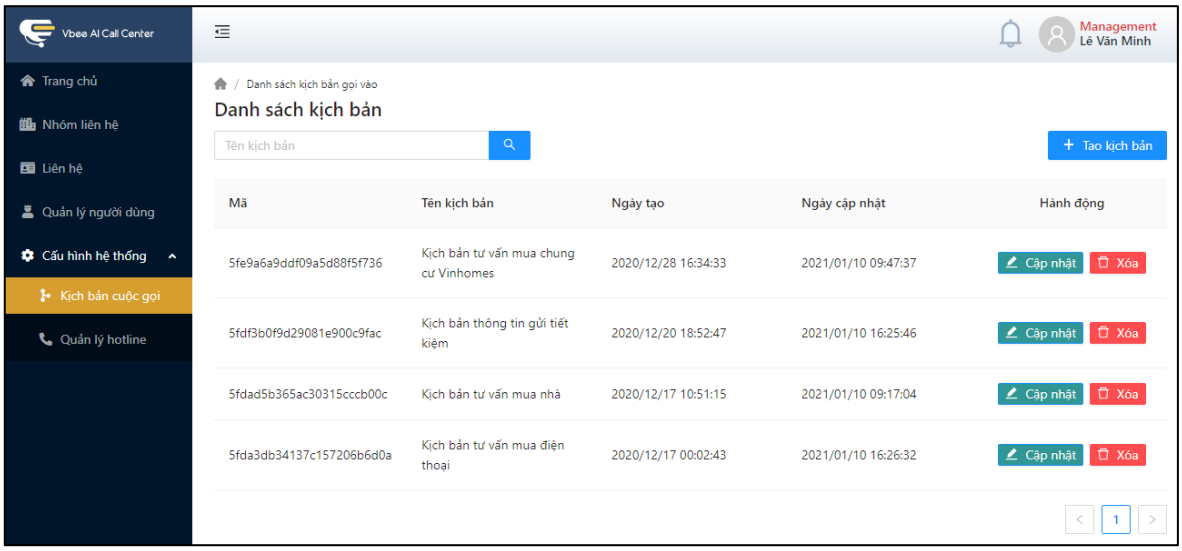
Chức năng Quản lý người dùng dành cho người quản lý. **Hình 26** mô tả giao diện của màn hình danh sách người dùng trong portal. Màn hình hỗ trợ chức năng tìm kiếm người dùng theo vai trò hoặc email, số điện thoại. Bảng trình bày các thông tin của người dùng portal bao gồm tên người dùng, email, số điện thoại, vai trò, ngày tham gia portal và ngày cập nhật. Bên phải của bảng là các thao tác bao gồm xem chi tiết, cập nhật hoặc xóa người dùng khỏi hệ thống.




**Hình 27** Giao diện thêm người dùng vào portal.

Khi chọn chức năng Thêm người dùng portal, hộp thoại được hiện lên như **Hình 27**. Các trường thông tin bắt buộc bao gồm email và vai trò người dùng. Sau khi thêm, một lời mời được gửi qua email đến người dùng. Các trường thông tin được xác thực ngay trên giao diện trước khi gọi API thêm người dùng.

### c) Quản lý kịch bản cuộc gọi

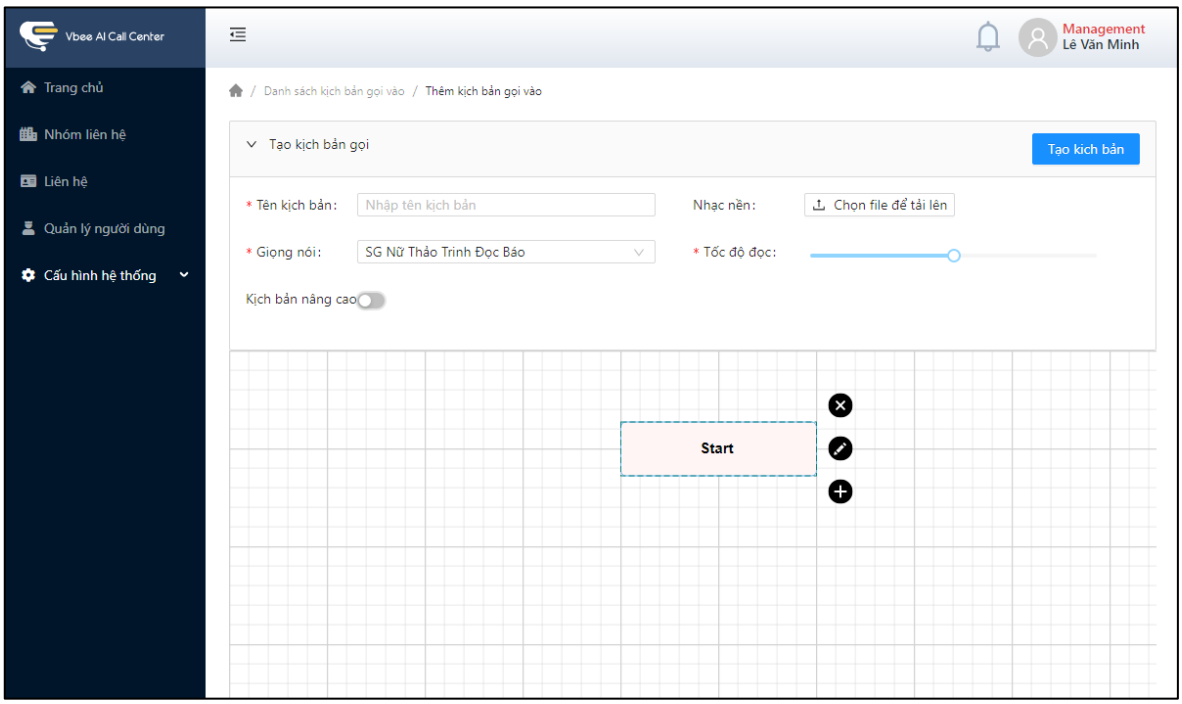



**Hình 28** Giao diện danh sách kịch bản cuộc gọi.

**Hình 28** là giao diện danh sách kịch bản cuộc gọi. Bảng thể hiện thông tin một số thông tin của các kịch bản cuộc gọi bao gồm: mã, tên kịch bản, ngày tạo, ngày cập nhật. Bên phải của bảng là các hành động cập nhật và xóa kịch bản.

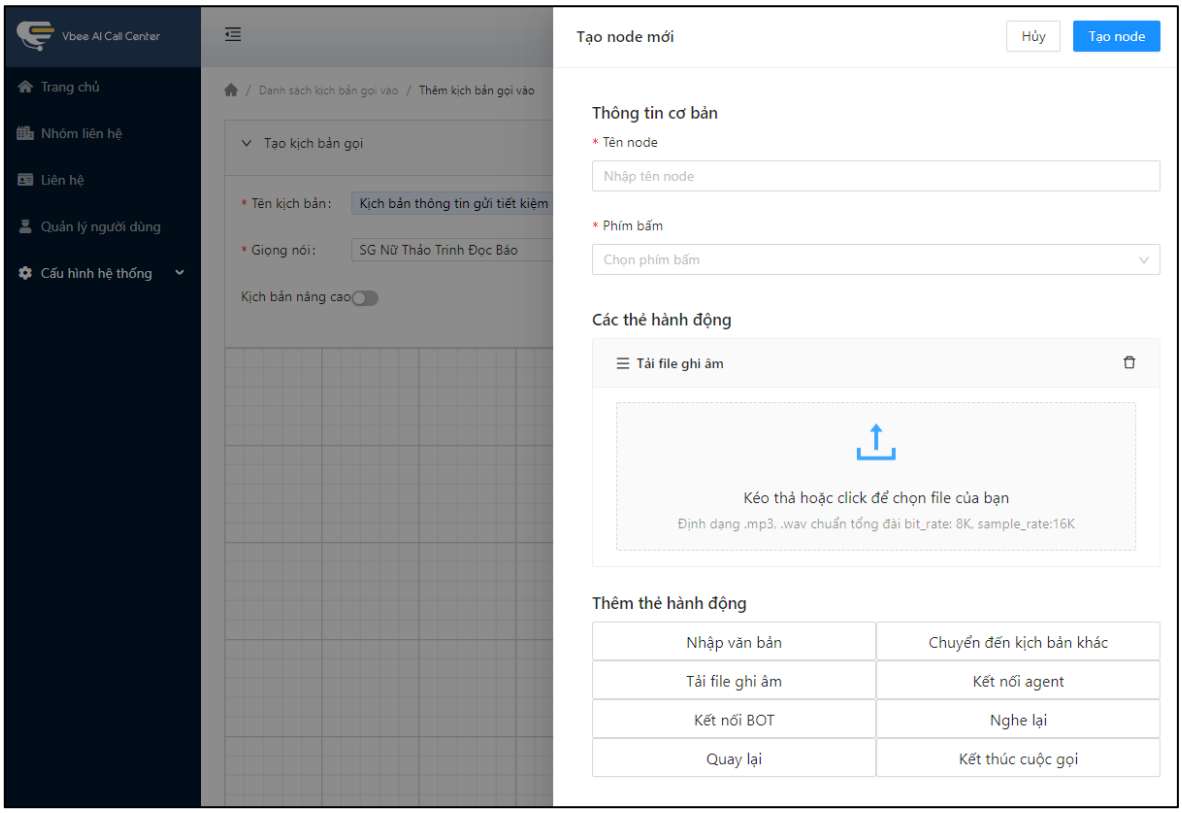
Phía trên bảng là trường dữ liệu cho phép tìm kiếm kịch bản cuộc gọi theo tên. Phía trên góc phải là nút tạo kịch bản. Khi nhấn vào nút tạo kịch bản, hệ thống sẽ hiển thị giao diện tạo kịch bản cuộc gọi như **Hình 29**. Giao diện tạo kịch bản gồm hai phần, một phần là form chứa thông tin của kịch bản, một phần là biểu đồ luồng hoạt động của kịch bản.

Người dùng cần nhập vào các thông tin của kịch bản như tên, giọng nói, tốc độ đọc, nhạc nền,… Sau đó, người dùng thao tác để tạo ra luồng kịch bản phù hợp. Khi chọn vào một node trên biểu đồ, ở cạnh bên phải của node đó sẽ hiện lên các tùy chọn: xóa node, sửa cấu hình node và thêm node con của node đó.



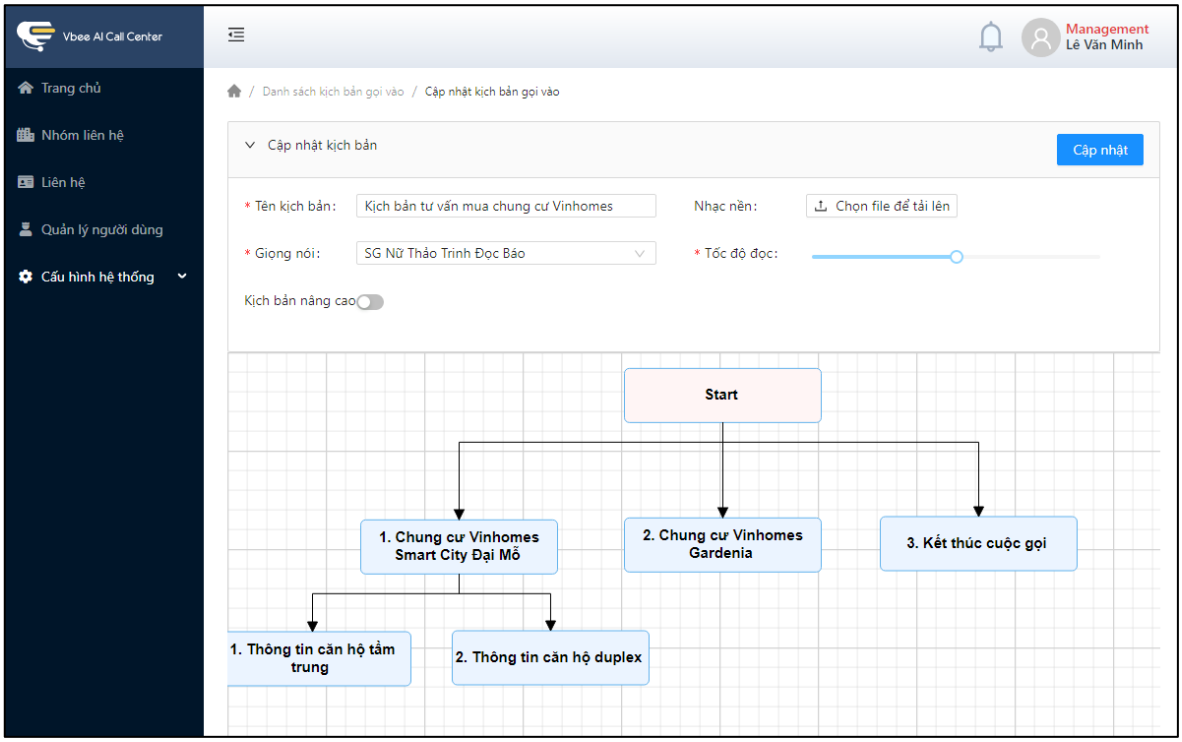

**Hình 29** Giao diện tạo kịch bản cuộc gọi.

Trong trường hợp người dùng chọn tạo node mới, hệ thống sẽ hiển thị một form để cấu hình cho node đó như **Hình 30**. Các trường thông tin bắt buộc là tên node, phím bấm tương ứng với node đó. Tiếp theo, người dùng cài đặt các thẻ hành động cho node đó. Sau khi hoàn thành, người dùng bấm tạo node để tạo ra một node mới trên biểu đồ luồng.




**Hình 30** Giao diện tạo node mới trong biểu đồ luồng.

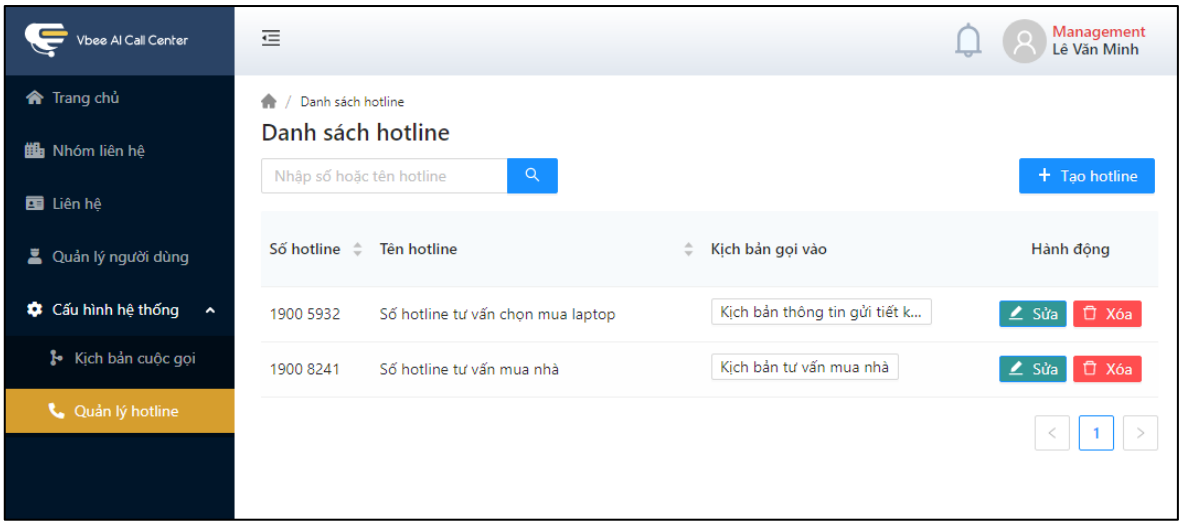
**Hình 31** là giao diện cập nhật thông tin của kịch bản. Giao diện này tương tự với giao diện tạo kịch bản.




**Hình 31** Giao diện cập nhật kịch bản.

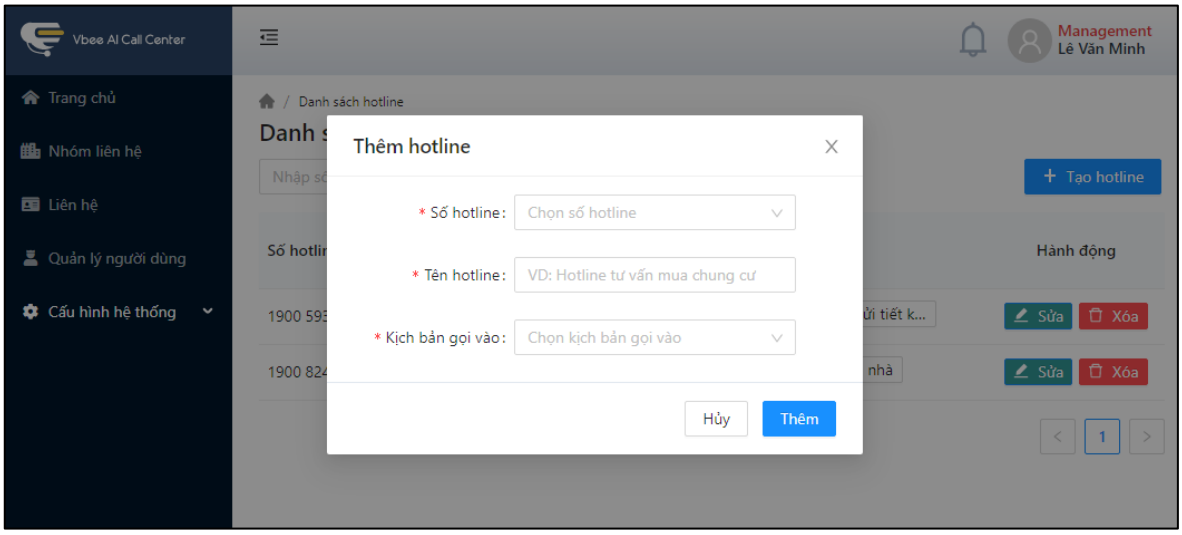
### d) Quản lý số hotline

Chức năng Quản lý số hotline dành cho người quản lý. **Hình 32** mô tả giao diện của màn hình danh sách hotline. Bảng trình bày các thông tin bao gồm số hotline, tên số hotline, kịch bản gọi vào tương ứng. Có hai thao tác với số hotline bao gồm sửa và xóa.



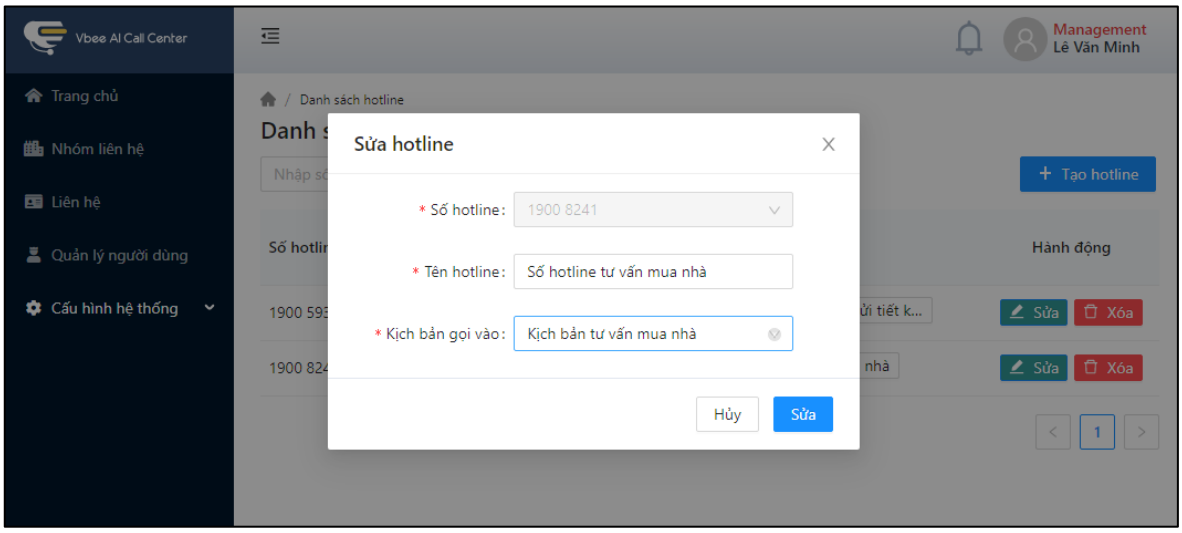

**Hình 32** Giao diện danh sách số hotline.

**Hình 33** mô tả giao diện của hộp thoại thêm hotline sau khi người dùng chọn chức năng Tạo hotline. Các trường thông tin là bắt buộc. Người dùng lựa chọn số hotline và kịch bản gọi vào từ danh sách. Các trường thông tin được xác thực ngay trên giao diện trước khi gọi API thêm hotline.




**Hình 33** Giao diện thêm số hotline.

**Hình 34** mô tả giao diện của hộp thoại sửa hotline sau khi người dùng chọn chức năng Sửa hotline. Trường số hotline không cho phép sửa. Các trường thông tin được xác thực ngay trên giao diện trước khi gọi API sửa hotline.




**Hình 34** Giao diện sửa số hotline.

### 4.2.4 Kiểm thử và triển khai

Em sử dụng kỹ thuật kiểm thử hộp đen dựa trên các màn hình hiển thị để kiểm thử cho hệ thống quản lý kịch bản cuộc gọi. Em đã đưa ra 31 test case kiểm thử, các test case này được trình bày trong **Bảng 21**. Chi tiết nội dung kiểm thử của các test case em sẽ trình bày trong phần Phụ lục.

**Bảng 21** Danh sách test case của hệ thống quản lý kịch bản cuộc gọi

| STT | Nhóm chức năng | Tên test case |
|---|---|---|
| 1. | Quản lý liên hệ | Kiểm tra hiển thị danh sách liên hệ |
| 2. | | Kiểm tra thêm liên hệ mới |
| 3. | | Kiểm tra tải lên tệp danh sách liên hệ |
| 4. | | Kiểm tra xem chi tiết liên hệ |
| 5. | | Kiểm tra cập nhật liên hệ |
| 6. | | Kiểm tra tìm kiếm liên hệ |

| | | |
|---|---|---|
| 7. | | Kiểm tra xóa liên hệ |
| 8. | Quản lý nhóm liên hệ | Kiểm tra hiển thị danh sách nhóm liên hệ |
| 9. | | Kiểm tra tạo nhóm liên hệ |
| 10. | | Kiểm tra xem thông tin chi tiết nhóm liên hệ |
| 11. | | Kiểm tra cập nhật nhóm liên hệ |
| 12. | | Kiểm tra tìm kiếm nhóm liên hệ |
| 13. | | Kiểm tra xóa nhóm liên hệ |
| 14. | Quản lý kịch bản cuộc gọi | Kiểm tra danh sách kịch bản cuộc gọi |
| 15. | | Kiểm tra tạo kịch bản cuộc gọi |
| 16. | | Kiểm tra xem thông tin kịch bản cuộc gọi |
| 17. | | Kiểm tra cập nhật kịch bản cuộc gọi |
| 18. | | Kiểm tra tìm kiếm kịch bản cuộc gọi |
| 19. | | Kiểm tra xóa kịch bản cuộc gọi |
| 20. | Quản lý người dùng | Kiểm tra thêm người dùng vào portal |
| 21. | | Kiểm tra danh sách tài khoản trong portal |
| 22. | | Kiểm tra cập nhật thông tin tài khoản |
| 23. | | Kiểm tra tìm kiếm tài khoản |
| 24. | | Kiểm tra xóa tài khoản khỏi portal |
| 25. | Quản lý hotline | Kiểm tra danh sách số hotline |
| 26. | | Kiểm tra thêm số hotline |
| 27. | | Kiểm tra tìm kiếm số hotline |

| | | |
|---|---|---|
| 28. | | Kiểm tra cập nhật số hotline |
| 29. | | Kiểm tra xóa số hotline |
| 30. | Quản lý tài khoản cá nhân | Kiểm tra xem thông tin cá nhân |
| 31. | | Kiểm tra cập nhật thông tin cá nhân |

Hệ thống quản lý kịch bản cuộc gọi đang được triển khai thử nghiệm tại https://dashboard-inbound.iristech.club. Cụ thể một số thông tin về cấu hình server để cài đặt và triển khai được nêu ra ở **Bảng 22**.

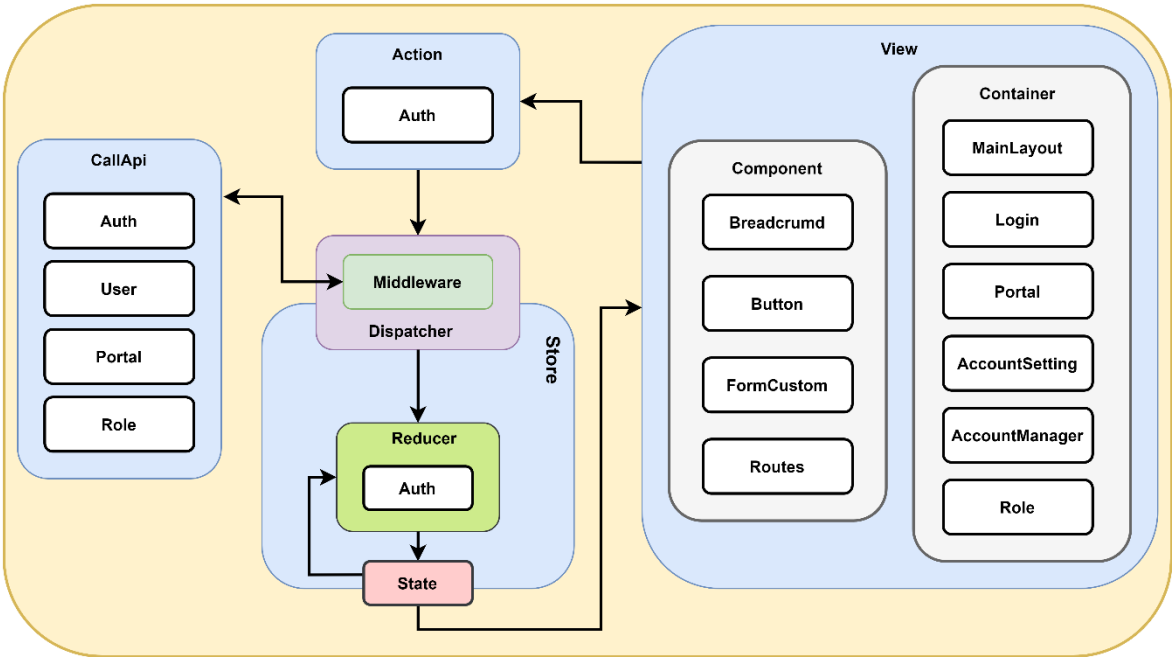
**Bảng 22** Thông số cấu hình server triển khai hệ thống quản lý kịch bản cuộc gọi

| **Tên cấu hình** | **Thông số** |
|---|---|
| Hệ điều hành | Ubuntu 16.04 |
| CPU | Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz |
| RAM | 2 GB |

## 4.3 Hệ thống xác thực và phân quyền tập trung

### 4.3.1 Thiết kế kiến trúc

#### 4.3.1.1 Thiết kế kiến trúc frontend




**Hình 35** Thiết kế kiến trúc frontend hệ thống xác thực và phân quyền tập trung.

Tương tự như kiến trúc frontend của hệ thống quản lý kịch bản cuộc gọi, kiến trúc frontend của hệ thống xác thực và phân quyền tập trung bao gồm bốn thành phần chính đó là: View, Store, Action và CallApi, được thể hiện ở **Hình 35**.

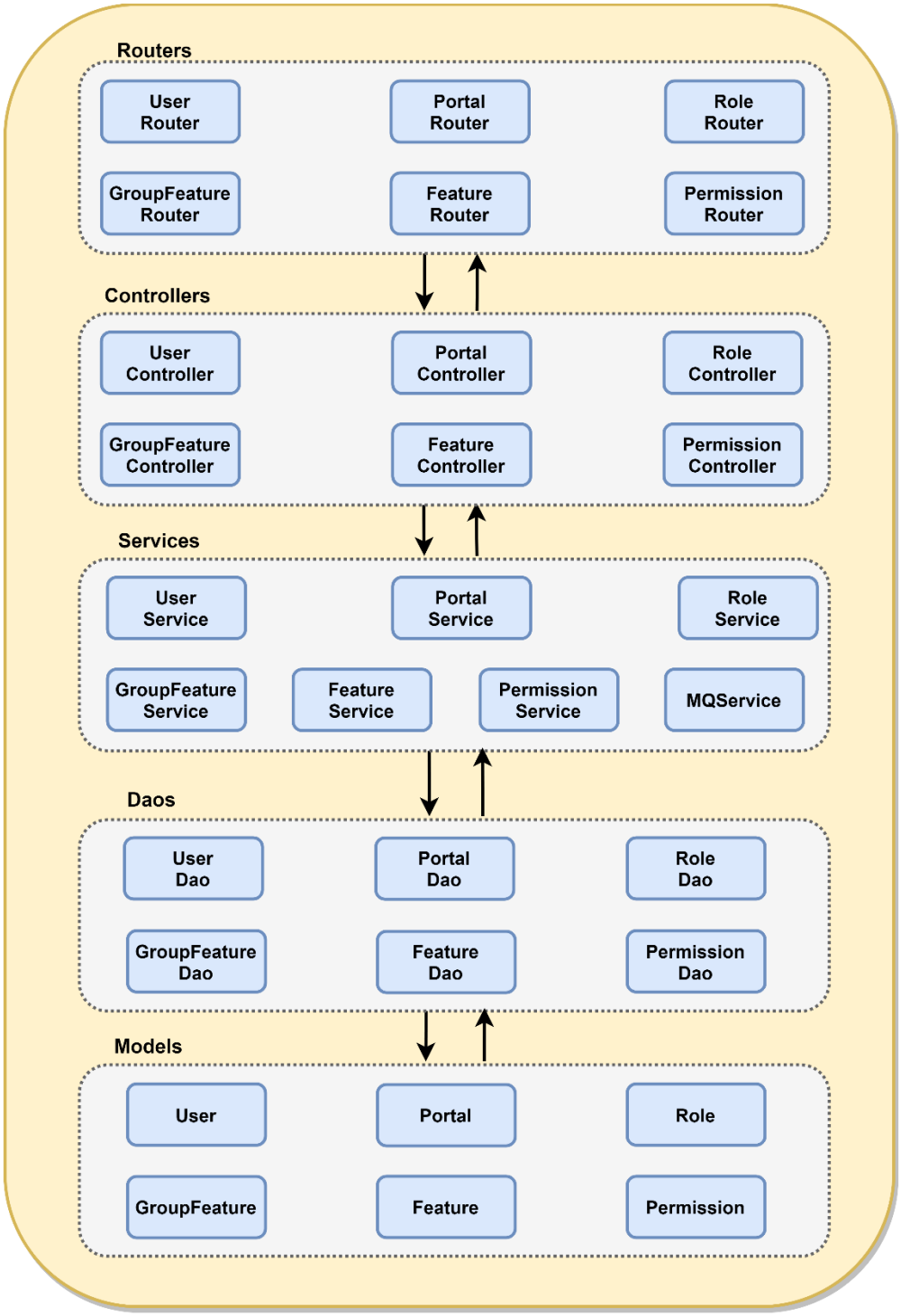
Khi muốn thay đổi trạng thái ứng dụng, thành phần giao diện View sẽ gọi đến các Action. Khi đó, Reducer sẽ nhận được sự kiện và cập nhật trạng thái tương ứng. Nếu cần tương tác với máy chủ, các Middleware sẽ thực hiện gửi yêu cầu và nhận dữ liệu từ máy chủ về cho Reducer xử lý.

#### 4.3.1.2 Thiết kế kiến trúc backend

Kiến trúc backenđ của hệ thống xác thực và phân quyền tập trung tương tự với kiến trúc backend của hệ thống quản lý kịch bản cuộc gọi. **Hình 36** thể hiện kiến trúc backend của hệ thống xác thực và phân quyền người dùng, gồm năm thành phần chính là: Routers, Controllers, Serivces, Daos và Models.

Khi client gửi request đến server, các Routers có nhiệm vụ định tuyến, gọi đến các Controllers tương ứng để xử lý. Các Controllers nhận yêu cầu từ Routers, sau đó gọi đến các

Services để xử lý về mặt nghiệp vụ. Các Services gọi đến Daos để thực hiện thao tác với cơ sở dữ liệu.

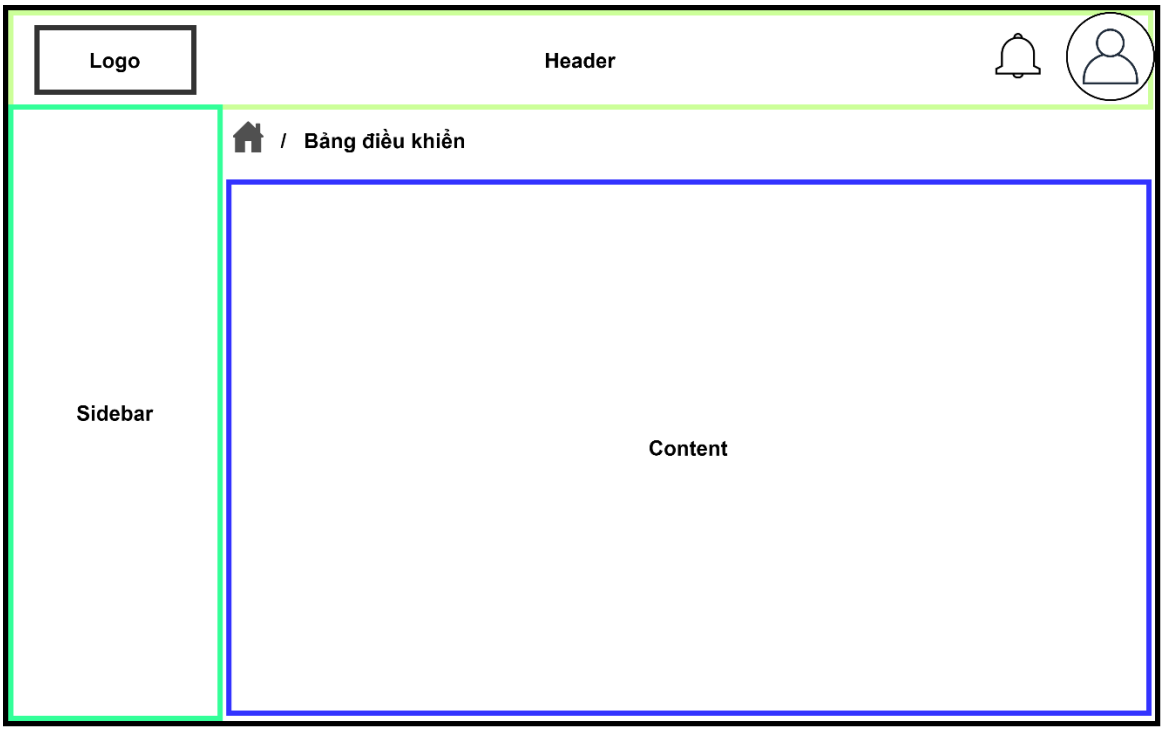



**Hình 36** Thiết kế kiến trúc backend hệ thống xác thực và phân quyền tập trung.

### 4.3.2 Thiết kế chi tiết

#### 4.3.2.1 Thiết kế chi tiết frontend

**a) Thiết kế mockup**

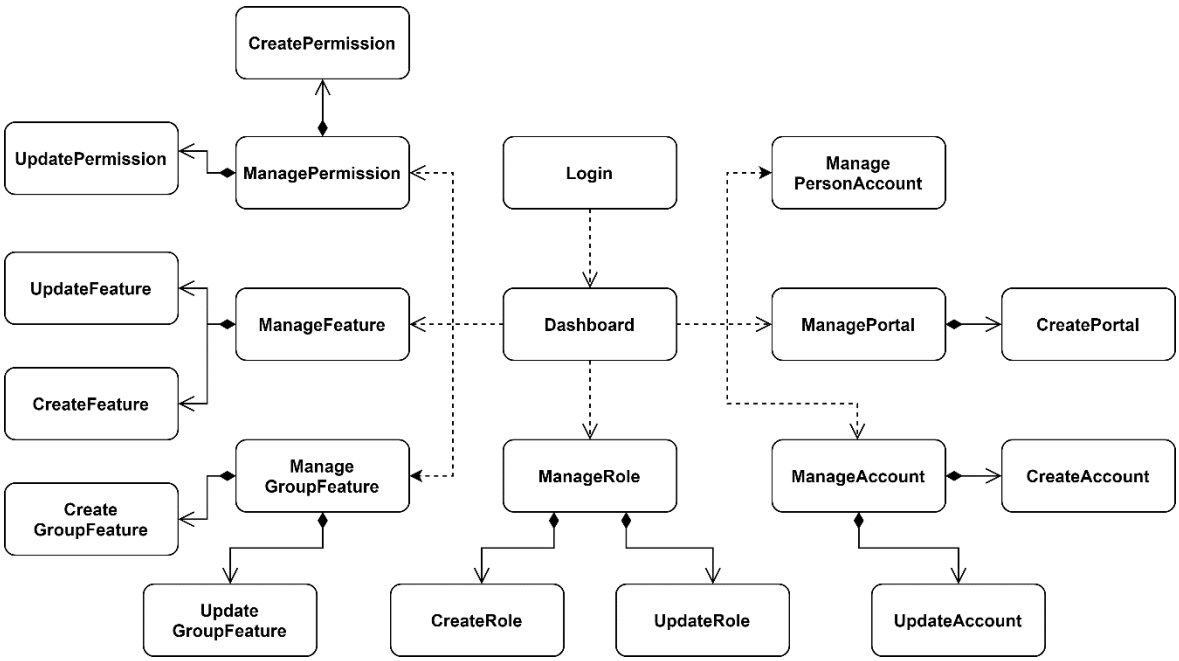



**Hình 37** Thiết kế mockup hệ thống xác thực và phân quyền tập trung.

Tương tự như giao diện của hệ thống quản lý kịch bản cuộc gọi, các màn hình của hệ thống xác thực và phân quyền đều sử dụng thống nhất hai phần Header và Sidebar, chỉ phần Content thay đổi để hiển thị nội dung tương ứng. Header là thanh trên cùng của trang web. Nó hiển thị logo của trang web, các thông báo và các thông tin cơ bản của người dùng. Sidebar là phần phía bên trái của trang web. Ở đây người dùng có thể dễ dàng điều hướng trang web đến các chức năng mong muốn.

**b) Biểu đồ dịch chuyển màn hình**

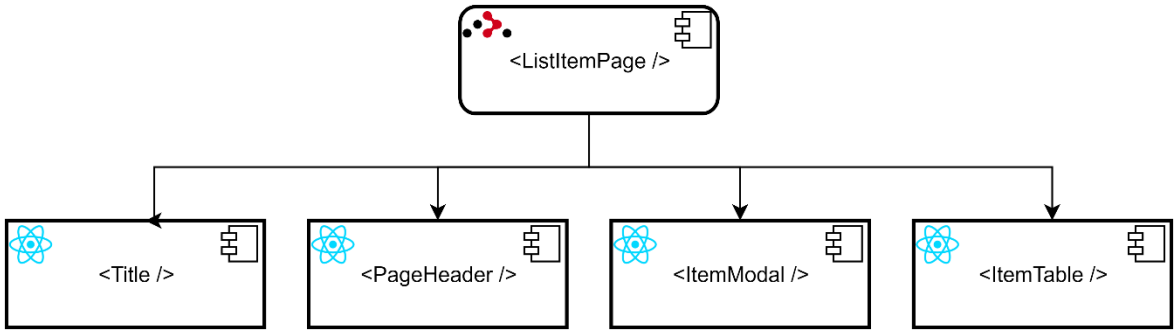
**Hình 38** là biểu đồ dịch chuyển màn hình hệ thống xác thực và phân quyền tập trung. Sau khi đăng nhập vào hệ thống, người dùng được chuyển đến màn hình Dashboard. Tiếp theo đó, người dùng có thể chuyển đến các màn hình quản lý liên quan đến quyền, màn hình quản lý portal, quản lý tài khoản hệ thống và quản lý tài khoản tài khoản các nhân.




**Hình 38** Biểu đồ dịch chuyển màn hình hệ thống xác thực và phân quyền tập trung.

### c) Thiết kế thành phần giao diện

Trong phần này, em trình bày thiết kế các thành phần giao diện chung cho các màn hình quản lý quyền, quản lý tính năng, quản lý nhóm tính năng và quản lý vai trò.




**Hình 39** Thiết kế thành phần giao diện hệ thống xác thực và phân quyền tập trung.

**Hình 39** thể hiện các thành phần giao diện của các trang quản lý trên. Một trang quản lý bao gồm bốn thành phần con, đó là: Title, PageHeader, ItemModal và ItemTable. Thành phần Title hiển thị tiêu đề của trang. PageHeader là thành phần chứa thanh tìm kiếm, các nút chức năng như thêm quyền, thêm vai trò. Thành phần Item Modal là một form sẽ được hiển thị ra khi người dùng chọn chức năng thêm mới hoặc chỉnh sửa. ItemTable thành phần có chức năng hiển thị dữ liệu dưới dạng bảng.

### 4.3.2.2 Thiết kế chi tiết backend

**Bảng 23** đưa ra danh sách các API của hệ thống xác thực và phân quyền tập trung.

**Bảng 23** Danh sách API của hệ thống xác thực và phân quyền tập trung

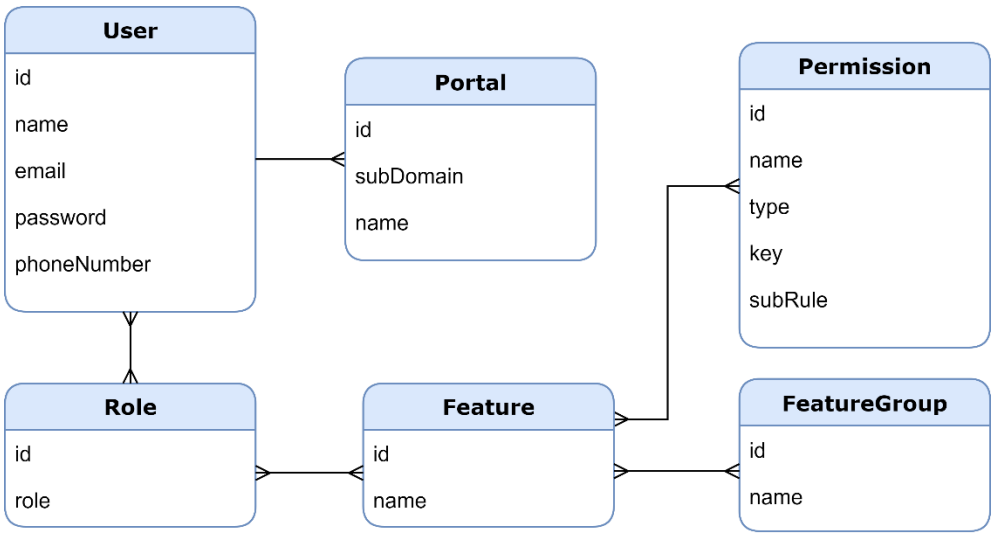
| STT | Nhóm API | Mục đích | Phương thức | Địa chỉ |
|---|---|---|---|---|
| **Base url:** /api/v1 | | | | |
| 1. | Xác thực và phân quyền | Đăng ký tài khoản | POST | /auths/register |
| 2. | | Đăng nhập | POST | /auths/login |
| 3. | | Xác thực người dùng | GET | /auths/verify |
| 4. | Người dùng | Lấy danh sách người dùng hệ thống | GET | /users |
| 5. | | Lấy thông tin chi tiết người dùng | GET | /users/:id |
| 6. | | Cập nhật thông tin người dùng | PUT | /users/:id |
| 7. | | Xóa người dùng | DELETE | /users/:id |
| 8. | Quản lý quyền | Tạo quyền mới | POST | /permissions |
| 9. | | Lấy danh sách các quyền | GET | /permissions |
| 10. | | Lấy thông tin chi tiết của quyền | GET | /permissions/:id |
| 11. | | Cập nhật quyền | PUT | /permissions/:id |
| 12. | | Xóa quyền | DELETE | /permissions/:id |
| 13. | Quản lý chức năng | Thêm mới chức năng | POST | /features |
| 14. | | Lấy danh sách các chức năng | GET | /features |
| 15. | | Xem thông tin chi tiết của chức năng | GET | /features/:id |

| STT | Nhóm API | Mục đích | Phương thức | Địa chỉ |
|---|---|---|---|---|
| 16. | | Cập nhật chức năng | PUT | /features/:id |
| 17. | | Xóa chức năng | DELETE | /features/:id |
| 18. | Quản lý nhóm chức năng | Tạo nhóm chức năng | POST | /features-group |
| 19. | | Lấy danh sách nhóm chức năng | GET | /features-group |
| 20. | | Xem chi tiết nhóm chức năng | GET | /features-group/:id |
| 21. | | Cập nhật nhóm chức năng | PUT | /features-group/:id |
| 22. | | Xóa nhóm chức năng | DELETE | /features-group/:id |
| 23. | Quản lý vai trò | Tạo vai trò | POST | /roles |
| 24. | | Lấy danh sách vai trò | GET | /roles |
| 25. | | Xem chi tiết vai trò | GET | /roles/:id |
| 26. | | Cập nhật vai trò | PUT | /roles/:id |
| 27. | | Xóa vai trò | DELETE | /roles/:id |
| 28. | Quản lý portal | Tạo portal | POST | /portals |
| 29. | | Kiểm tra prefix của portal tồn tại hay chưa | POST | /portals/prefix |
| 30. | | Kiểm tra tên miền phụ tồn tại hay chưa | POST | /portals/sub-domain |

#### 4.3.2.3 Thiết kế chi tiết cơ sở dữ liệu

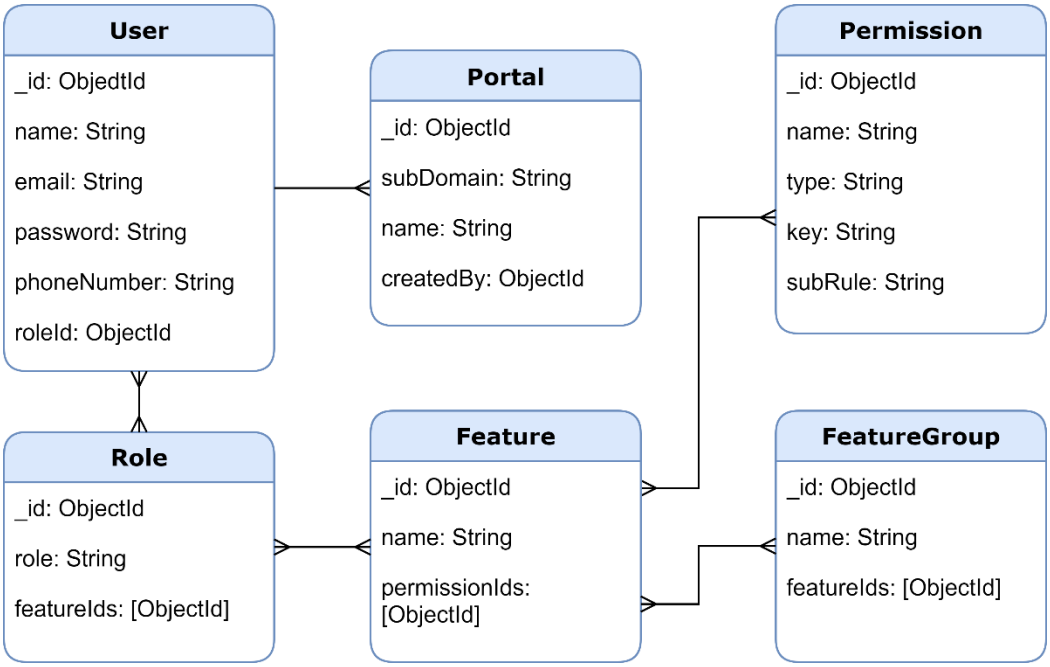
**Hình 40** biểu diễn các thực thể trong hệ thống xác thực và phân quyền người dùng tập trung và mối quan hệ giữa chúng. Một người dùng có thể sở hữu nhiều Portal. Quyền của người dùng được định nghĩa bởi thực thể Role. Trong hệ thống tổng đài có các Role là Người quản

lý và Tổng đài viên. Mỗi Role có quyền thực hiện các nhóm chức năng riêng. Nhóm chức năng này sẽ gồm nhiều chức năng được định nghĩa ở Feature, ví dụ như chức năng thêm liên hệ, xem danh sách liên hệ, xóa liên hệ. Các chức năng này bao gồm nhiều quyền (Permission), đó là các quyền truy cập vào các API, quyền hiển thị các thành phần trên giao diện.




**Hình 40** Biểu đồ thực thể liên kết của hệ thống xác thực và phân quyền tập trung.

Từ biểu đồ thực thể liên kết, em đã thiết kế biểu đồ cơ sở dữ liệu dạng NoSQL cho cơ sở dữ liệu MongoDB như **Hình 41**.




**Hình 41** Biểu đồ cơ sở dữ liệu hệ thống xác thực và phân quyền tập trung.

Hệ thống xác thực và phân quyền tập trung có 6 collections là User, Portal, Role, FeatureGroup, Feature và Permission. Thiết kế chi tiết từng collection em xin trình bày trong **Bảng 24**.

**Bảng 24** Thiết kế chi tiết cơ sở dữ liệu hệ thống xác thực và phân quyền tập trung

| **Collection** | **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** |
|---|---|---|---|
| User | _id | ObjectId | ID của người dùng trong hệ thống |
| | name | String | Tên người người dùng |
| | email | String | Email của người dùng |
| | phoneNumber | String | Số điện thoại của người dùng |
| | password | String | Mật khẩu đã được mã hóa của người dùng |
| | roleId | String | ID của vai trò |
| Portal | _id | String | ID của portal trong hệ thống |
| | name | String | Tên của portal |
| | subDomain | String | Tên miền phụ của portal |
| | createdBy | ObjectId | ID của người tạo ra portal |
| Role | _id | ObjectId | ID của vai trò |
| | role | String | Tên vai trò |
| | featureIds | [ObjectId] | Mảng ID cá chức năng mà vai trò này có quyền thực hiện |
| FeatureGroup | _id | ObjectId | ID của nhóm chức năng |
| | name | String | Tên nhóm chức năng |

| | | | |
|---|---|---|---|
| | featureIds | [ObjectId] | Mảng ID các chức năng có trong nhóm chức năng này |
| Feature | _id | String | ID của chức năng |
| | name | String | Tên chức năng |
| | permissionIds | [ObjectId] | Mảng ID của các quyền tương ứng với chức năng |
| Permission | _id | ObjectId | ID của quyền |
| | name | String | Tên của quyền |
| | type | String | Loại quyền: quyền thực hiện bên frontend hoặc quyền thực hiện bên backend |
| | key | String | Key của quyền |
| | subRule | String | Tập luật do người dùng định nghĩa |

### 4.3.3 Xây dựng ứng dụng

#### 4.3.3.1 Thư viện và công cụ sử dụng

Trong quá trình phát triển hệ thống xác thực và phân quyền tập trung, em sử dụng các thư viện và công cụ giống hệ thống quản lý kịch bản cuộc gọi. Chi tiết em đã đưa ra ở **Bảng 20**.

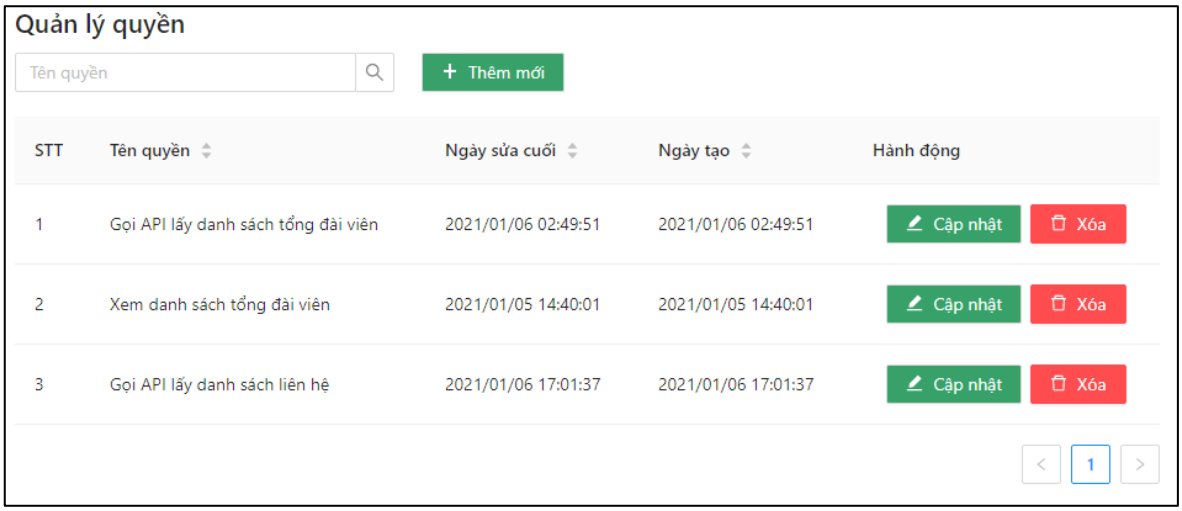
#### 4.3.3.2 Kết quả đạt được

Em đã xây dựng và triển khai hệ thống trên server và gắn được tên miền. Địa chỉ truy cập của hệ thống là https://inbound.iristech.club. Cùng với đó, hệ thống quản lý kịch bản cuộc gọi cũng đã được tích hợp vào hệ thống.

#### 4.3.3.3 Minh họa các chức năng chính

Trong phần này em trình bày một số chức năng chính của hệ thống xác thực và phân quyền tập trung, đó là: quản lý quyền, quản lý tính năng, quản lý nhóm tính năng, quản lý vai trò và quản lý portal.

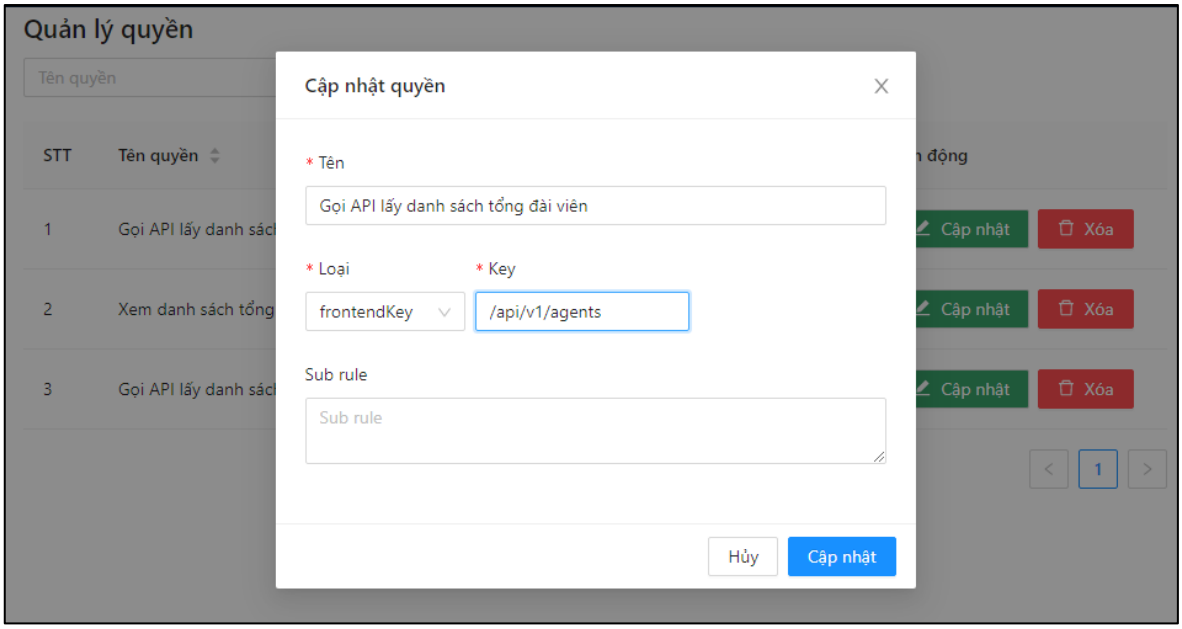
**a) Quản lý quyền**




**Hình 42** Giao diện danh sách quyền.

**Hình 42** là giao diện danh sách các quyền. Danh sách được hiển thị dưới dạng bảng. Phía trên bảng là trường dữ liệu phục vụ cho việc tìm kiếm và nút bấm thêm mới quyền. Các thông tin của quyền được hiển thị là: tên quyền, ngày tạo và ngày sửa đổi. Phần bên phải của bảng là các nút ứng với thao tác cập nhật và xóa quyền. Danh sách quyền được hiển thị theo dạng phân trang, Người dùng có thể chọn số trang ở góc dưới bên phải.

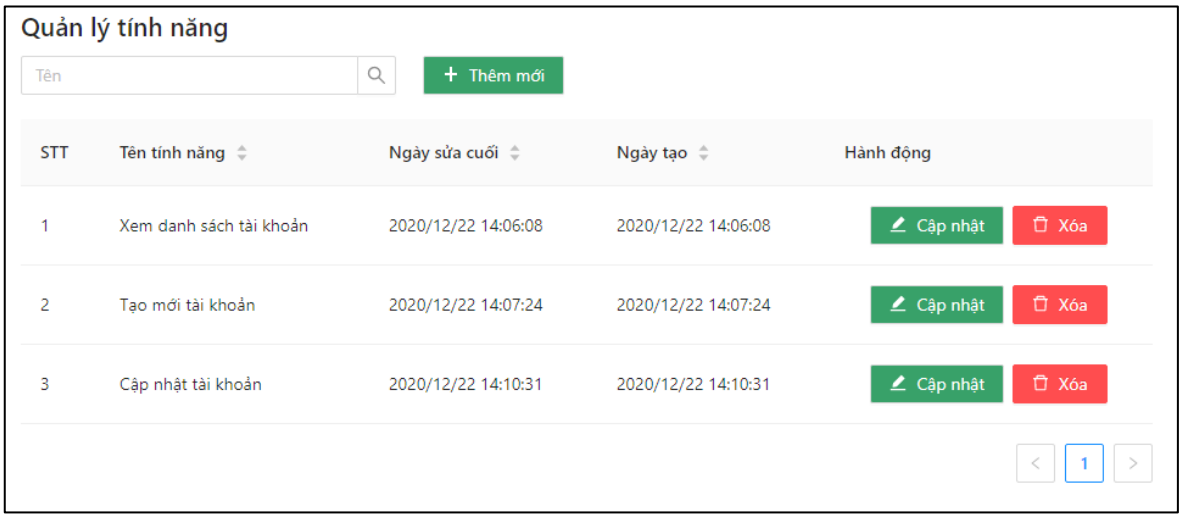
Giao diện cập nhật quyền được thể hiện ở **Hình 43**. Người dùng cần nhập vào các trường thông tin: tên quyền, loại quyền, key và có thể nhập subRule là tập luật đi kèm.




**Hình 43** Giao diện cập nhật quyền.

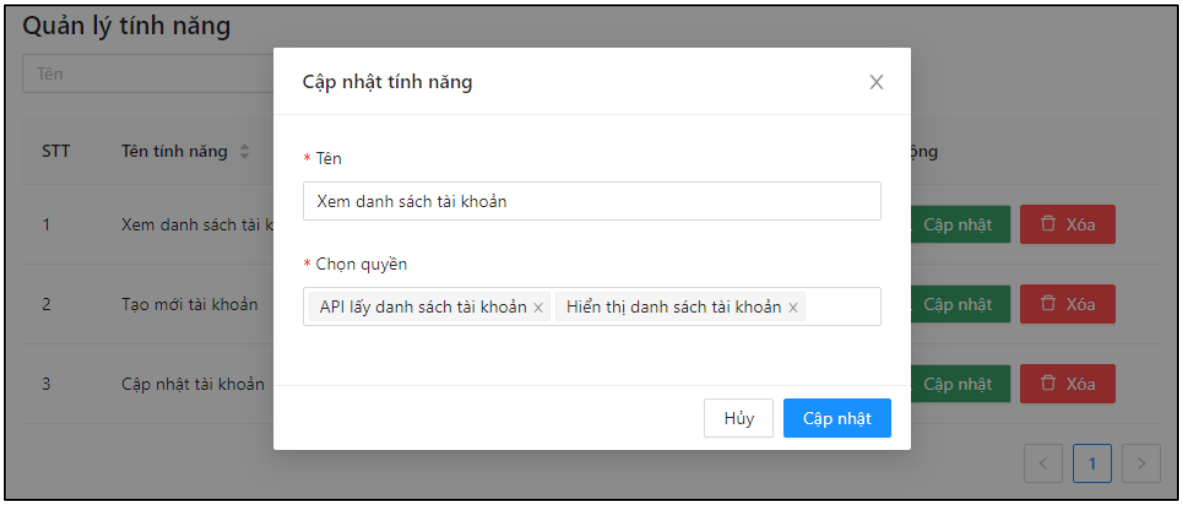
**b) Quản lý tính năng**

**Hình 44** là giao diện danh sách các tính năng. Tương tự như danh sách quyền, danh sách tính năng được hiển thị dưới dạng bảng được phân trang. Các hành động được đặt phía bên phải bảng, bao gồm cập nhật và xóa tính năng.




**Hình 44** Giao diện danh sách tính năng.

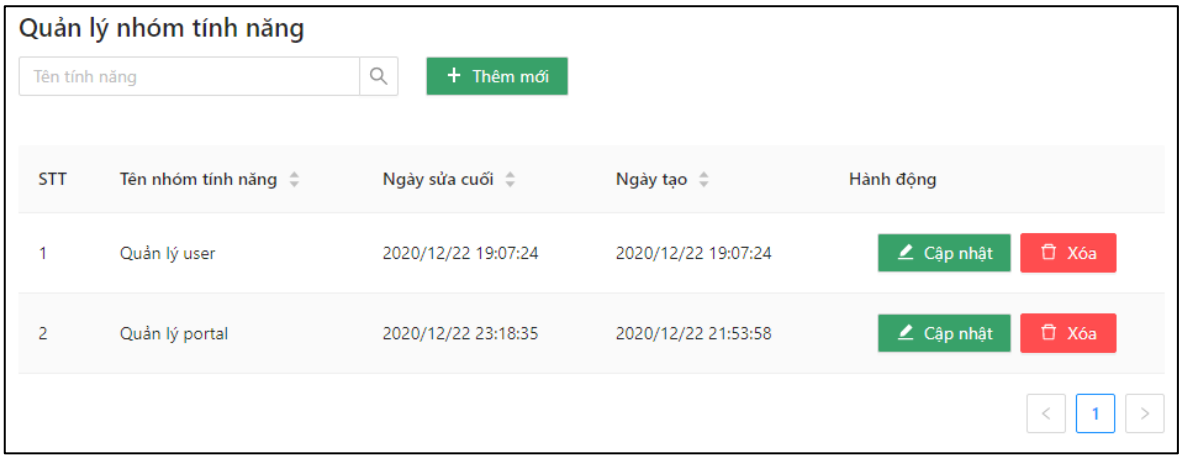
Khi chọn vào cập nhật tính năng, giao hiện sẽ hiển thị như **Hình 45**. Người dùng cần nhập vào tên tính năng và chọn các quyền ứng với tính năng đó.




**Hình 45** Giao diện cập nhật tính năng.

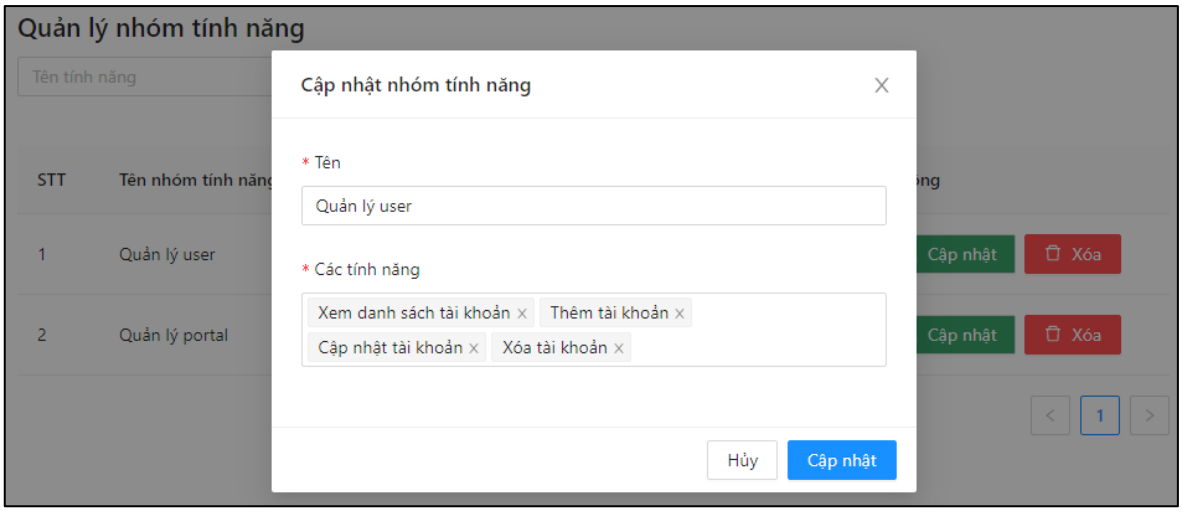
### c) Quản lý nhóm tính năng

**Hình 46** là giao diện danh sách nhóm tính năng. Giao diện này có cấu trúc tương tự các giao diện đã nêu trên.




**Hình 46** Giao diện danh sách nhóm tính năng.

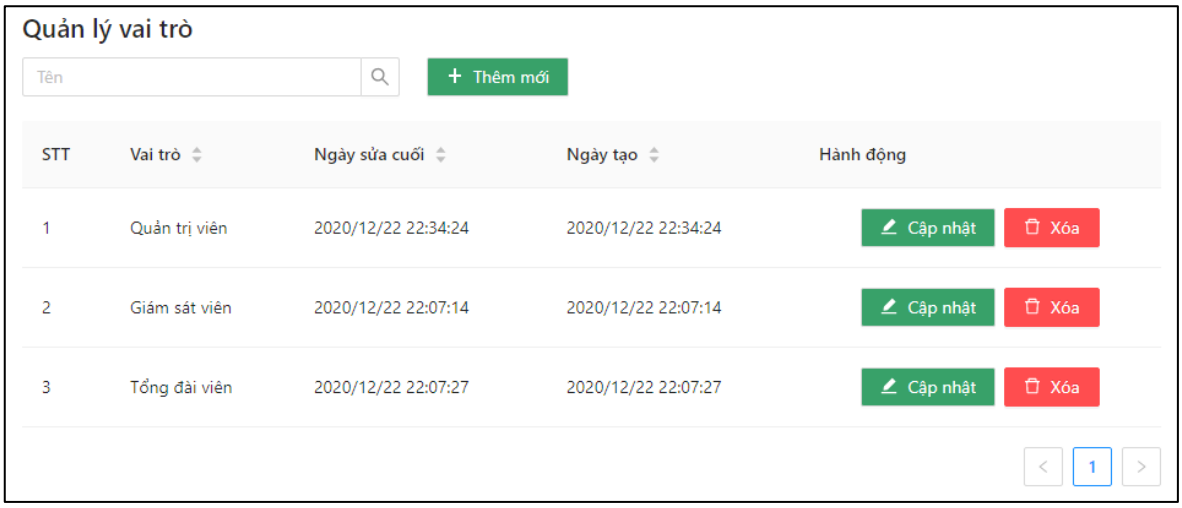
Khi người dùng nhấn nút cập nhật, hộp thoại như **Hình 47** sẽ hiện ra. Người dùng được cập nhật tên nhóm tính năng và các tính năng trong nhóm tình năng.



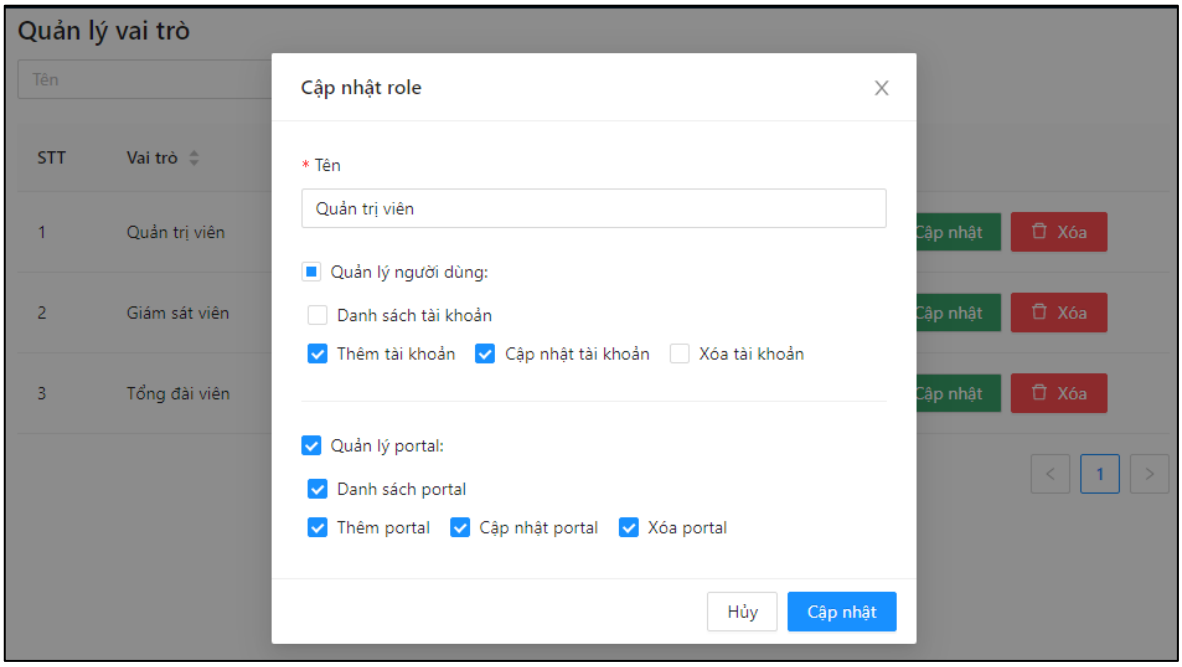

**Hình 47** Giao diện cập nhật nhóm tính năng.

**d) Quản lý vai trò**

**Hình 48** mô tả giao diện danh sách các vai trò. Các vai trò được hiển thị dưới dạng bảng có thể phân trang. Phía bên phải của bảng là các nút ứng với chức năng cập nhật và xóa vai trò. Khi chọn nút cập nhật vai trò, một hộp thoại sẽ hiển thị ra như **Hình 49**. Hộp thoại đó chứa danh sách các quyền được hiển thị theo nhóm. Người dùng có thể ấn chọn vào tên nhóm quyền để chọn toàn bộ quyền trong nhóm, hoặc có thể chọn từng quyền một.



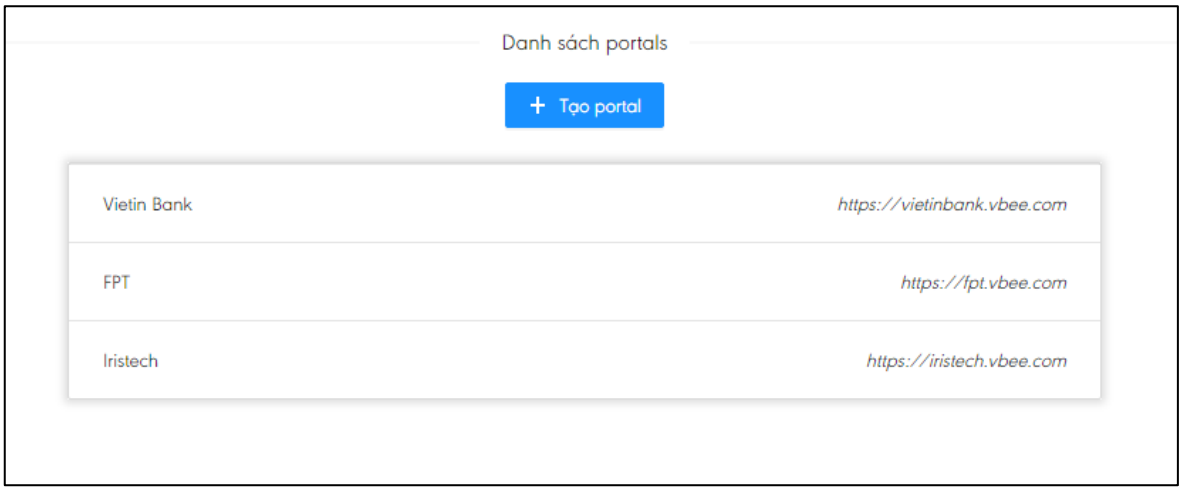

**Hình 48** Giao diện quản lý vai trò.




**Hình 49** Giao diện cập nhật vai trò.

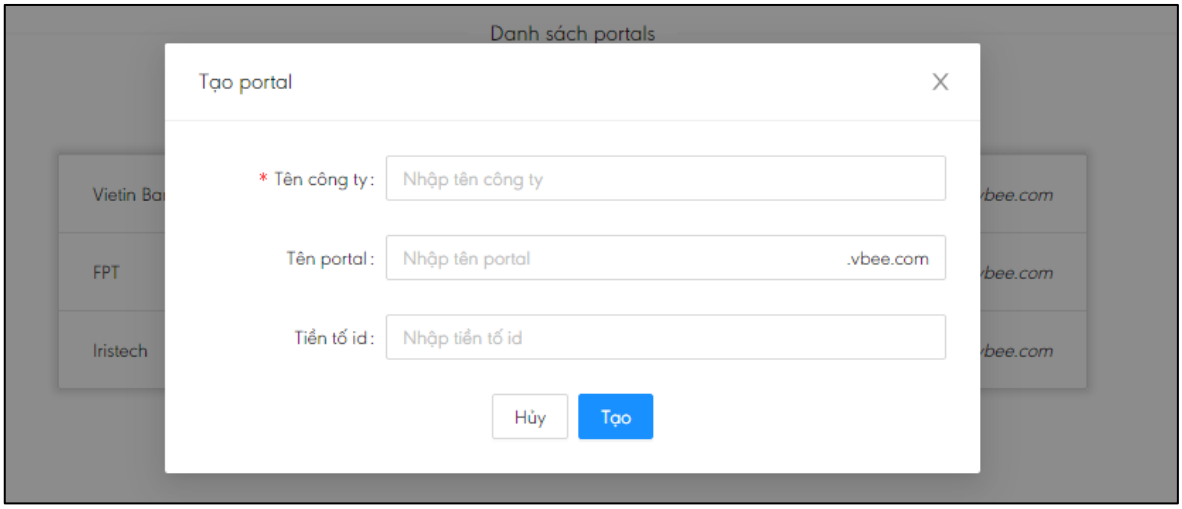
### e) Quản lý portal

**Hình 50** mô tả danh sách các portal mà người dùng sở hữu. Một portal được hiển thị với các thông tin: tên công ty và đường dẫn tới portal đó. Người dùng có thể tạo portal bằng cách nhấn vào nút tạo portal trên giao diện. Một hộp thoại tạo portal sẽ xuất hiện như **Hình 51**.




**Hình 50** Giao diện danh sách portal.

Các thông tin mà người dùng cần nhập để tạo portal là tên công ty, tên portal và tiền tố id. Các thông tin sẽ được kiểm tra ngay trong khi người dùng nhập vào ô input và sẽ hiển thị cảnh báo lỗi nếu tên portal hoặc tiền tố id đã tồn tại trong hệ thống.




**Hình 51** Giao diện tạo portal.

### 4.3.4 Kiểm thử và triển khai

Trong quá trình phát triển hệ thống, em sử dụng kỹ thuật kiểm thử hộp đen để kiểm thử các chức năng cho hệ thống. Em đã đưa ra 37 test case kiểm thử, cụ thể được trình bày trong **Bảng 25**. Chi tiết nội dung kiểm thử của các test case em sẽ trình bày trong phần Phụ lục.

**Bảng 25** Danh sách các test case của hệ thống xác thực và phân quyền tập trung

| STT | Nhóm chức năng | Tên test case |
|---|---|---|
| 1. | Quản lý quyền | Kiểm tra hiển thị danh sách quyền |
| 2. | | Kiểm tra thêm quyền mới |
| 3. | | Kiểm tra xem chi tiết quyền |
| 4. | | Kiểm tra cập nhật quyền |
| 5. | | Kiểm tra tìm kiếm quyền |
| 6. | | Kiểm tra xóa quyền |
| 7. | Quản lý chức năng | Kiểm tra hiển thị danh sách chức năng |
| 8. | | Kiểm tra thêm chức năng |
| 9. | | Kiểm tra xem thông tin chi tiết chức năng |
| 10. | | Kiểm tra cập nhật chức năng |
| 11. | | Kiểm tra tìm kiếm chức năng |
| 12. | | Kiểm tra xóa chức năng |
| 13. | Quản lý nhóm chức năng | Kiểm tra danh sách nhóm chức năng |
| 14. | | Kiểm tra thêm nhóm chức năng |
| 15. | | Kiểm tra xem thông tin nhóm chức năng |
| 16. | | Kiểm tra cập nhật nhóm chức năng |
| 17. | | Kiểm tra tìm kiếm nhóm chức năng |
| 18. | | Kiểm tra xóa nhóm chức năng |
| 19. | Quản lý vai trò | Kiểm tra danh sách vai trò |

| STT | Nhóm chức năng | Tên test case |
|---|---|---|
| 20. | | Kiểm tra thêm vai trò |
| 21. | | Kiểm tra xem thông tin vai trò |
| 22. | | Kiểm tra cập nhật vai trò |
| 23. | | Kiểm tra tìm kiếm vai trò |
| 24. | | Kiểm tra xóa vai trò |
| 25. | Quản lý tài khoản | Kiểm tra danh sách tài khoản |
| 26. | | Kiểm tra thêm tài khoản |
| 27. | | Kiểm tra tìm kiếm tài khoản |
| 28. | | Kiểm tra cập nhật tài khoản |
| 29. | | Kiểm tra xóa tài khoản |
| 30. | Quản lý tài khoản cá nhân | Kiểm tra đăng ký |
| 31. | | Kiểm tra đăng nhập |
| 32. | | Kiểm tra đăng xuất |
| 33. | | Kiểm tra đổi mật khẩu |
| 34. | | Kiểm tra cập nhật thông tin tài khoản |
| 35. | Quản lý portal | Kiểm tra danh sách portal |
| 36. | | Kiểm tra thêm portal |
| 37. | | Kiểm tra xóa portal |

Hiện tại hệ thống đang được triển khai thử nghiệm với địa chỉ https://inbound.iristech.club.

Thông tin cấu hình server để triển khai hệ thống được đưa ra ở **Bảng 26**.

**Bảng 26** Thông tin cấu hình server triển khai hệ thống xác thực và phân quyền tập trung

| **Tên cấu hình** | **Thông số** |
|---|---|
| Hệ điều hành | Ubuntu 16.04 |
| CPU | Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz |
| RAM | 2 GB |

Trong Chương 4, em đã áp dụng các công nghệ đã tìm hiểu ở Chương 3 để thiết kế và xây dựng hệ thống quản lý kịch bản, xác thực và phân quyền tập trung. Tiếp theo đó, trong Chương 5 em sẽ trình bày cụ thể những đóng góp nổi bật của mình trong suốt quá trình làm đồ án tốt nghiệp.

# Chương 5 Các giải pháp và đóng góp nổi bật

Trong các chương trước, em đã trình bày chi tiết từ đặt vấn đề, khảo sát phân tích yêu cầu đến thiết kế, xây dựng, triển khai và kiểm thử hệ thống. Tuy nhiên, những đóng góp nổi bật, những giải pháp mà em tâm đắc trong quá trình làm đồ án chưa được làm rõ. Chính vì thế, trong Chương 5 em sẽ trình bày chi tiết về các vấn đề trên. Cụ thể, mục 5.1 em sẽ trình bày về cách mình phát triển công cụ quản lý kịch bản cuộc gọi. Mục 5.2 em sẽ trình bày về cách mình thiết kế kiến trúc của hệ thống. Trong mục 5.3 và 5.4, em sẽ trình bày các giải pháp về xác thực và phân quyền người dùng.

## 5.1 Phát triển công cụ quản lý kịch bản cuộc gọi

### 5.1.1 Đặt vấn đề

Trong hệ thống tổng đài thông minh, kịch bản gọi có thể coi là phần quan trọng nhất. Nó giúp điều hướng luồng cuộc gọi theo cấu hình mà người quản trị thiết đặt sẵn. Như đã trình bày ở mục 2.1.1, các công cụ để quản lý kịch bản cuộc gọi của các hệ thống hiện nay còn rất nhiều điểm hạn chế, em xin tổng hợp lại như sau.

Thứ nhất là giao diện không trực quan, khó hình dung. Khi người dùng vào xem nội dung kịch bản, rất khó có thể hình dung tổng quan xem các luồng thực hiện như nào, sự phân cấp các phím bấm như nào. Để xem được các node con có trong một node, người dùng phải nhấn vào xem chi tiết node đó .

Thứ hai là người dùng khó kiểm soát khi số lượng node nhiều. Khi số lượng node tăng lên, việc kiểm soát chính xác các luồng càng khó khăn hơn.

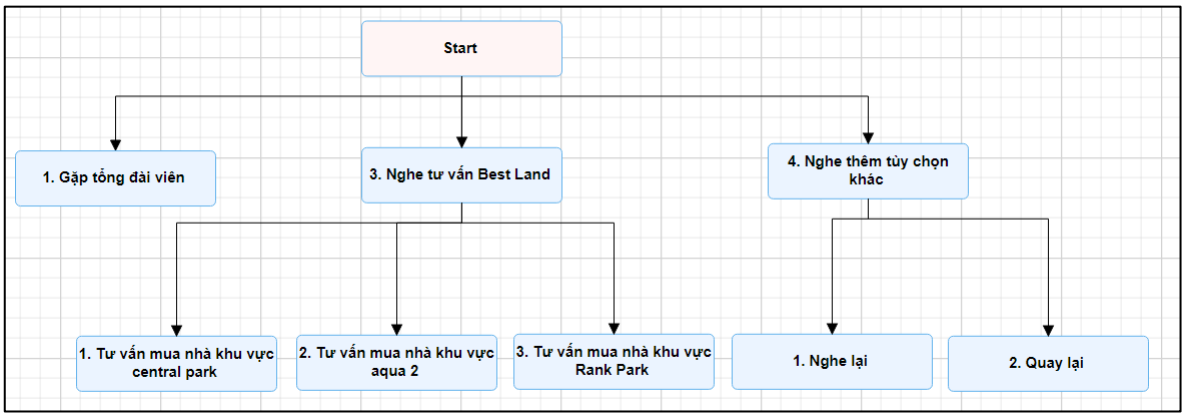
Thứ ba là với những kịch bản gọi với độ phức tạp cao hơn thì ứng dụng sẽ không đáp ứng được. Các ứng dụng hiện tại chỉ cho phép ứng với mỗi phím bấm của người dùng, hệ thống sẽ thực hiện một hành động tương ứng, ví dụ phát một đoạn âm thanh hoặc kết nối đến một tổng đài viên. Nếu người dùng muốn sau khi bấm một phím, trước hết hệ thống sẽ phát một đoạn âm thanh, sau đó kết nối đến tổng đài viên thì hệ thống không đáp ứng được.

### 5.1.2 Giải pháp

Từ những điểm hạn chế đã nêu ở mục 5.1.1, em sẽ phát triển công cụ quản lý kịch bản để giải quyết các vấn đề đó.

#### 5.1.2.1 Lựa chọn cách hiển thị kịch bản và thư viện phù hợp

Để giúp cho người dùng có cái nhìn trực quan nhất về luồng cuộc gọi, sự phân cấp giữa các phím bấm trong kịch bản, em sẽ hiển thị kịch bản dưới dạng một biểu đồ luồng thay vì hiển thị dạng danh sách như các ứng dụng khác đã làm. Mỗi phím bấm của người dùng sẽ tương ứng với một node trong biểu đồ. Khi người dùng thêm phím bấm tương ứng với thêm một node trong biểu đồ. Hệ thống sẽ hiển thị ra form để người dùng cài đặt chi tiết cho node đó.




**Hình 52** Ví dụ minh họa biểu đồ luồng trực quan hóa kịch bản cuộc gọi.

**Hình 52** là một ví dụ minh họa về biểu đồ luồng cho kịch bản cuộc gọi tư vấn mua nhà. Khi khách hàng gọi vào tổng đài, hệ thống sẽ phát ra âm thanh hướng dẫn người dùng bấm chọn các phím số theo kịch bản. Khách hàng bấm phím số 1 để gặp tổng đài viên, phím số 3 để nghe tư vấn mua nhà của Best Land, bấm phím số 4 để nghe các tùy chọn khác. Việc hiển thị kịch bản cuộc gọi ở dạng biểu đồ luồng hoạt động như trên giúp cho người dùng có cái nhìn tổng quan về kịch bản của cuộc gọi, đồng thời giúp người dùng có thể dễ dàng thực hiện các thao tác thêm, cập nhật, xóa các node trong kịch bản.

Thư viện em lựa chọn để vẽ ra biểu đồ luồng này là Ej2 React Diagrams. Đây là thư viện rất mạnh để giúp tạo ra công cụ vẽ luồng kịch bản. Lập trình viên có thể tùy biến màu sắc, hình dạng của các khối, các liên kết trong biểu đồ luồng. Thư viện còn hỗ trợ các thao tác undo, redo, sao chép, cắt và dán có thể giúp người dùng dễ dàng thao tác trong quá trình sử dụng. Ngoài ra, sau khi tạo xong biểu đồ, thư viện còn hỗ trợ xuất biểu đồ dưới dạng chuỗi để có thể lưu vào cơ sở dữ liệu. Thêm nữa, thư viện còn có tài liệu hướng dẫn, tài liệu API đầy đủ giúp thuật tiện cho việc tra cứu và sử dụng.

#### 5.1.2.2 Thiết kế các hành động

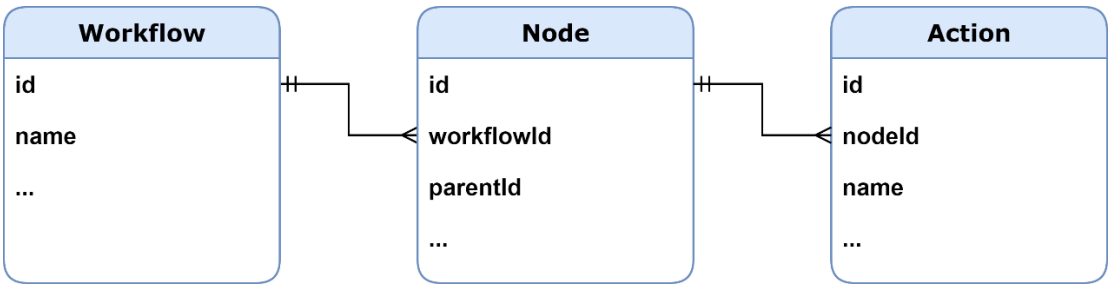
Trong kịch bản cuộc gọi sẽ có các node tương ứng với các phím bấm của khách hàng khi gọi vào tổng đài. Với mỗi node, tổng đài sẽ thực hiện hành động để tương tác với khách hàng. Để khắc phục hạn chế chỉ chọn được một hành động ứng với một phím bấm như đã nêu ở mục 5.1.1, em đã thiết kế một node sẽ có một danh sách các hành động được sắp xếp có thứ tự. Khi khách hàng nhấn phím tương ứng với node đó, các hành động này sẽ được thực hiện lần lượt từ trên xuống dưới. **Bảng 27** là danh sách các thẻ hành động và mô tả chi tiết của các thẻ đó.

**Bảng 27** Danh sách thẻ hành động trong kịch bản cuộc gọi

| STT | Trường dữ liệu | Mô tả |
|---|---|---|
| 1. | Nhập văn bản | Hệ thống sẽ chuyển văn bản thành giọng nói và phát ra âm thanh giọng nói |
| 2. | Chuyển đến kịch bản khác | Hệ thống chuyển đến một kịch bản cuộc gọi khác để xử lý |
| 3. | Tải lên tệp âm thanh | Hệ thống thực hiện phát tệp âm thanh |
| 4. | Kết nối tổng đài viên | Hệ thống chuyển cuộc gọi của khách hàng đến tổng đài viên |
| 5. | Kết nối Bot | Hệ thống chuyển cuộc gọi của khách hàng đến Bot |
| 6. | Nghe lại | Hệ thống thực hiện phát lại âm thanh của node trước |
| 7. | Quay lại | Hệ thống thực hiện quay lại một node trước trong kịch bản cuộc gọi |
| 8. | Kết thúc cuộc gọi | Hệ thống kết thúc cuộc gọi |

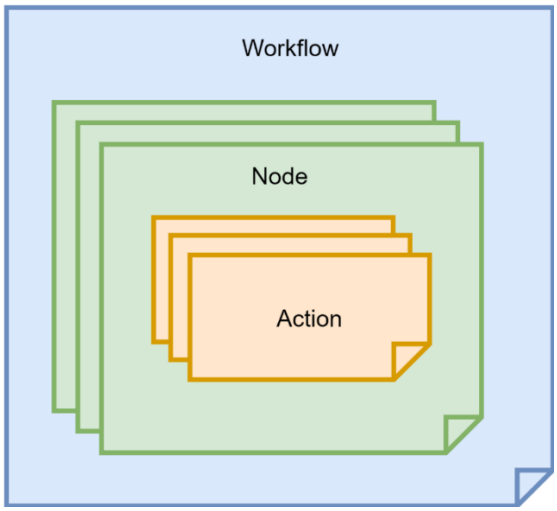
#### 5.1.2.3 Thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi

Mỗi kịch bản cuộc gọi có một danh sách các node khác nhau, trong mỗi node có danh sách các hành động khác nhau. Nếu thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi theo dạng NoSQL như **Hình 53**, thì khi lấy một kịch bản, hệ thống cần thực hiện JOIN hai bảng Workflow và bảng Node để lấy được tất cả các node trong kịch bản. Sau đó với từng node, hệ thống phải JOIN bảng Node với bảng Action để lấy được tất cả các Action trong node đó.




**Hình 53** Thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi theo dạng SQL.

Vì vậy, để tránh việc phải JOIN nhiều bảng như trên, em đã thiết kế cơ sở dữ liệu kịch bản cuộc gọi (Workflow) theo dạng NoSql. Collection Workflow chứa một mảng các node, trong mỗi node chứa một mảng các hành động được sắp xếp có thứ tự như **Hình 54**.




**Hình 54** Thiết kế cơ sở dữ liệu cho kịch bản cuộc gọi theo dạng NoSQL.

Thiết kế chi tiết các trường dữ liệu được mô tả qua **Bảng 28**.

**Bảng 28** Thiết kế chi tiết collection Workflow

| **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** |
|---|---|---|
| _id | ObjectId | ID của kịch bản cuộc gọi |
| name | String | Tên của kịch bản cuộc gọi |
| portalId | ObjectId | ID của portal chứa kịch bản cuộc gọi |
| voice | String | Loại giọng nói để áp dụng công nghệ chuyển văn bản thành giọng nói |

| | | |
|---|---|---|
| rate | Number | Tốc độ đọc của người đọc khi sử dụng chuyển văn bản thành giọng nói |
| bgMusicPath | String | Địa chỉ URL dẫn đến tệp âm thanh nhạc nền khi khách hàng gọi đến tổng đài |
| nodes | [Object] | Mảng các đối tượng Node của kịch bản cuộc gọi (mô tả chi tiết phía dưới *) |

* Đối tượng Node gồm các trường thông tin được mô tả ở **Bảng 29**.

**Bảng 29** Các trường thông tin của đối tượng Node trong Collection Workflow

| **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** |
|---|---|---|
| id | String | ID của node |
| parentId | String | ID của node cha |
| digit | String | Phím bấm tương ứng với node |
| name | String | Tên của node tương ứng với phím bấm |
| offsetX | Number | Vị trí của node ứng với trục x trong biểu đồ luồng |
| offsetY | Number | Vị trí của node ứng với trục y trong biểu đồ luồng |
| actions | [Object] | Mảng các đối tượng Action là các thẻ hành động trong Node (mô tả chi tiết phía dưới **) |

** Đối tượng Action gồm các trường thông tin được mô tả ở **Bảng 30**.

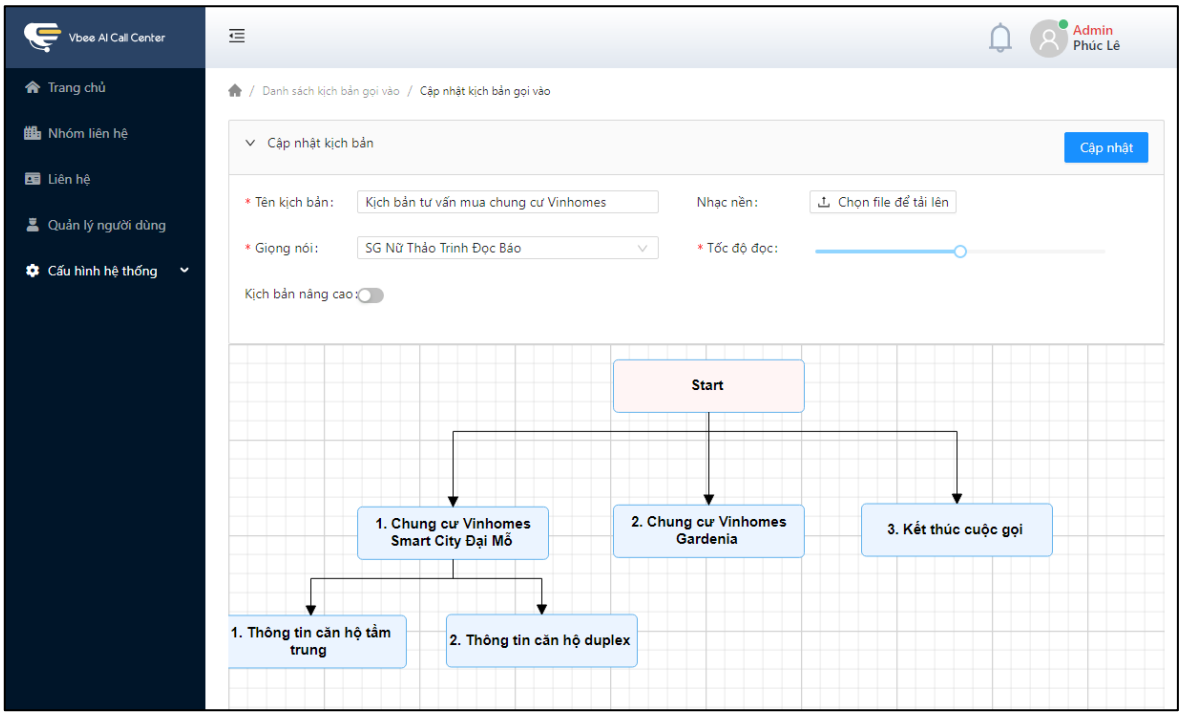
**Bảng 30** Các trường thông tin của đối tượng Action trong Collection Workflow

| **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** |
|---|---|---|
| action | String | Tên của hành động |
| textContent | String | Văn bản để sử dụng công nghệ chuyển âm thanh thành giọng nói |

| | | |
|---|---|---|
| audioPath | String | Địa chỉ URL của tệp âm thanh đã được chuyển từ văn bản (lấy từ trường textContent) thành giọng nói |
| alternativeTextContent | String | Văn bản dùng để thay thế khi trường textContent không sử dụng được nội dung cá nhân hóa |
| alternativeAudioPath | String | Địa chỉ URL của tệp âm thanh đã được chuyển từ văn bản (lấy từ trường alternativeTextContent) thành giọng nói |
| pressKeyTimeOut | Number | Thời gian tổng đài chờ người dùng nhấn phím |
| repeatTime | Number | Số lần tổng đài phát lặp lại nội dung của tệp âm thanh |
| forwardWorkflowId | ObjectId | ID của kịch bản cuộc gọi muốn chuyển hướng tới |
| recordPath | String | Địa chỉ URL của tệp âm thanh người dùng tải lên |
| connectQueueId | ObjectId | ID của hàng đợi. Hệ thống sẽ chọn ra một tổng đài viên trong hàng đợi này để kết nối với khách hàng |
| connectBot | Object | Đối tượng chứa địa chỉ websocket kết nối Bot và token để định danh |
| endCallCode | String | Mã kết thúc cuộc gọi |

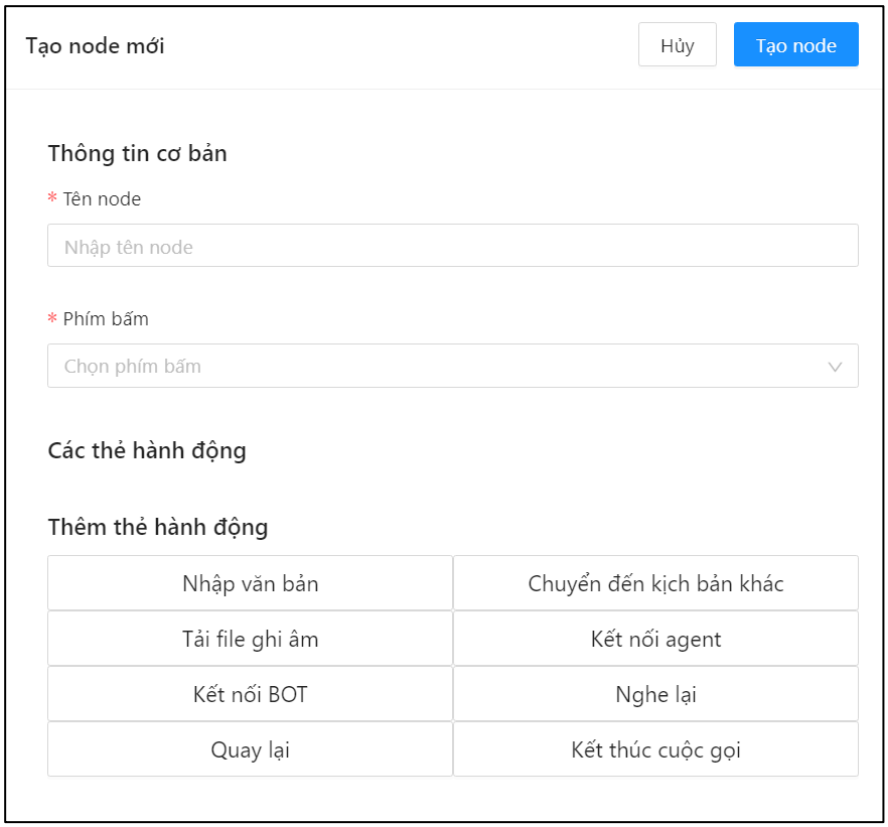
### 5.1.3 Kết quả đạt được

Em đã phát triển được công cụ trực quan giúp người dùng có thể dễ dàng quan sát được luồng của kịch bản cuộc gọi như **Hình 55**.




**Hình 55** Công cụ quản lý kịch bản cuộc gọi.

Người dùng có thể cấu hình các kịch bản có độ phức tạp cao, có thể lựa chọn nhiều hành động trong một node như **Hình 56**.




**Hình 56** Các hành động trong cấu hình kịch bản cuộc gọi.

## 5.2 Thiết kế hệ thống theo kiến trúc Microservices

### 5.2.1 Đặt vấn đề

Hiện nay khi thiết kế hệ thống, có hai kiến trúc phổ biến là kiến trúc dạng khối tập trung (monolithic) và kiến trúc nhiều dịch vụ nhỏ (microservices).

Kiến trúc theo dạng khối tập trung thường phù hợp để phát triển các ứng dụng vừa và nhỏ. Kiến trúc này giúp phát triển ứng dụng nhanh chóng, dễ triển khai. Tuy nhiên khi số lượng người dùng và số lượng yêu cầu chức năng tăng lên sẽ làm cho ứng dụng ngày càng phức tạp, cồng kềnh, khó bảo trì, khi ứng dụng phát sinh lỗi rất khó để kiểm tra và sửa lỗi. Đồng thời, mỗi cập nhật nhỏ của hệ thống đều phải deploy đầy đủ. Ngoài ra, kiến trúc này gây rào cản không nhỏ về mặt công nghệ. Các ứng dụng dạng khối thường chỉ có thể sử dụng một hoặc một vài ngôn ngữ và công nghệ. Việc thêm mới hoặc thay đổi công nghệ thường phải viết lại toàn bộ ứng dụng, việc này rất tốn kém chi phí và thời gian cho đội ngũ phát triển.

Kiến trúc microservices ra đời theo cách chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ có kết nối với nhau để giải quyết các vấn đề của kiến trúc monolithic. Mô hình khối truyền thống có thể được áp dụng trong các dịch vụ nhỏ. Vì vậy, kiến trúc microservices giúp giảm thiểu sự phức tạp của các hệ thống lớn, từ đó dễ dàng phát triển, bảo trì, có thể áp dụng và thay đổi công nghệ một cách linh hoạt, dễ dàng mở rộng hệ thống.

Chính vì thế mà em đã thiết kế kiến trúc hệ thống quản lý kịch bản cuộc gọi theo kiến trúc Microservices. Hệ thống sẽ được chia nhỏ làm nhiều dịch vụ, mỗi dịch vụ có các chức năng khác nhau và không phụ thuộc quá nhiều vào các dịch vụ khác.

### 5.2.2 Giải pháp

Em đã áp dụng kiến trúc microservices vào để thiết kế kiến trúc hệ thống quản lý kịch bản cuộc gọi như **Hình 57**. Hệ thống bao gồm 3 dịch vụ đó là dịch vụ xác thực và phân quyền người dùng (Authentication & Authorization Service), dịch vụ quản lý (Management Service) và dịch vụ lưu trữ (Upload Service). Các dịch vụ kết nối với nhau qua REST API.

Dịch vụ quản lý chủ yếu đảm nhiệm chức năng quản lý trong hệ thống, bao gồm quản lý kịch bản, quản lý liên hệ, quản lý nhóm liên hệ, quản lý nhóm tổng đài viên, quản lý hàng đợi, quản lý hotline và quản lý tài khoản cá nhân.

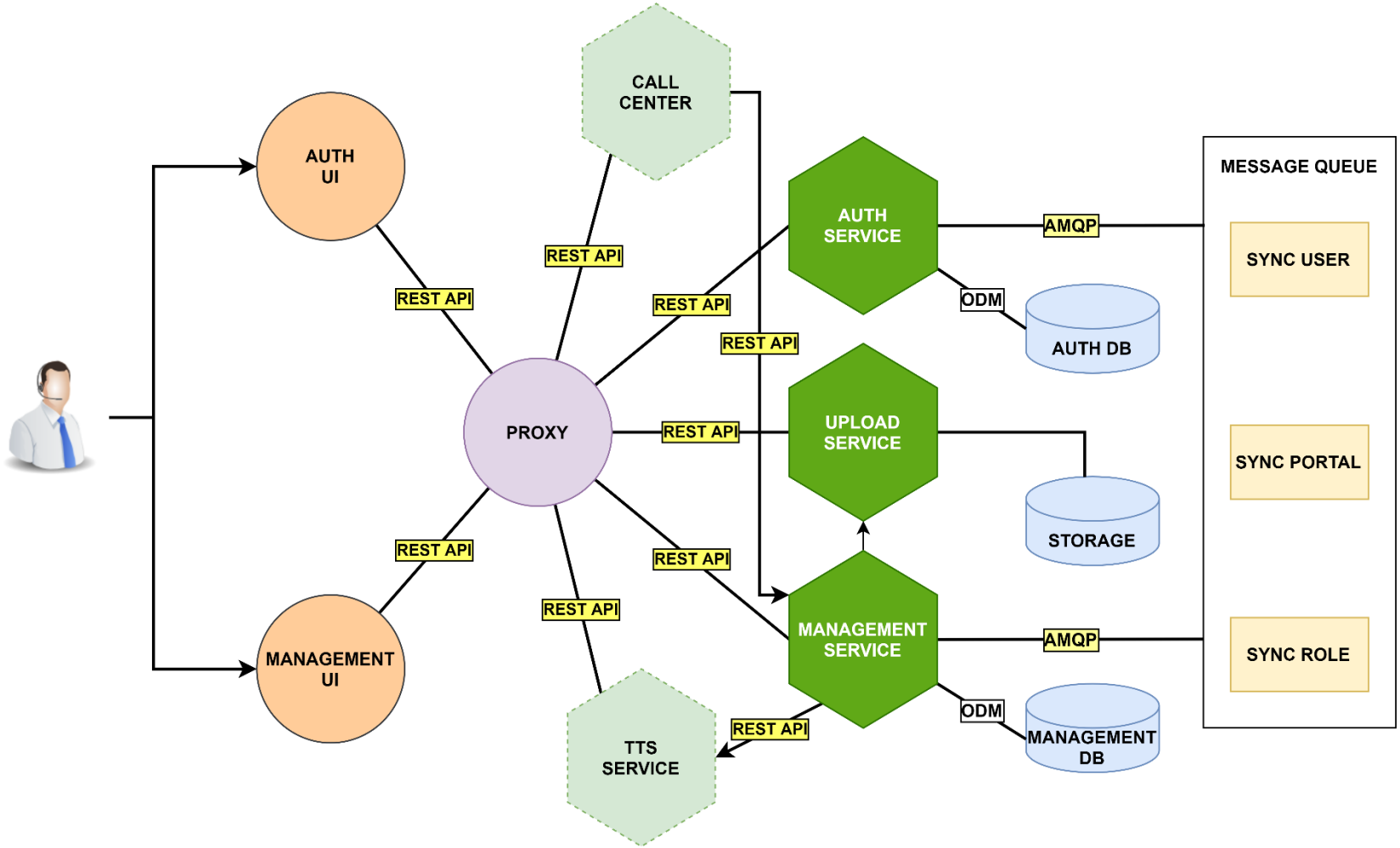
Dịch vụ xác thực và phân quyền người dùng cung cấp chức năng giúp quản lý người dùng tập trung. Khi người dùng truy cập vào các ứng dụng liên kết, ứng dụng sẽ chuyển đến dịch vụ này để xác thực và kiểm tra quyền truy cập, nếu hợp lệ thì người dùng mới được sử dụng các chức năng tương ứng với vai trò trong hệ thống.

Dịch vụ lưu trữ đảm nhiệm chức năng lưu trữ các tập tin của hệ thống bao gồm các tập tin mà người dùng tải lên (hình ảnh, âm thanh, tệp excel danh sách người dùng,…), các tệp âm thanh được chuyển thử văn bản thành giọng nói của các kịch bản,…

Ngoài ra, hệ thống còn tương tác với hai dịch vụ ngoài là TTS Service và Call Center. Hai dịch vụ này em đã trình bày ở mục 4.1.1.

Dịch vụ quản lý và dịch vụ xác thực và phân quyền người dùng đều không truy vấn trực tiếp vào cơ sở dữ liệu MongoDB mà thực hiện thông qua Object Data Model (ODM). ODM hỗ trợ ánh xạ giữa đối tượng trong Javascript với các document trong cơ sở dữ liệu ở dưới. Thêm nữa, ODM còn hỗ trợ cho việc kiểm tra, chuyển đổi kiểu dữ liệu phù hợp. Từ đó, việc truy vấn trở nên đơn giản và dễ dàng hơn.

Thông tin về người dùng, portal, quyền của dịch vụ quản lý và dịch vụ xác thực & phân quyền người dùng được đồng bộ qua hàng đợi Message Queue được xây dựng bằng RabbitMQ. Thông tin các hàng đợi em đã trình bày trong phần 4.2.2.2.




**Hình 57** Thiết kế hệ thống theo kiến trúc microservices.

## 5.3 Giải pháp xác thực người dùng tập trung

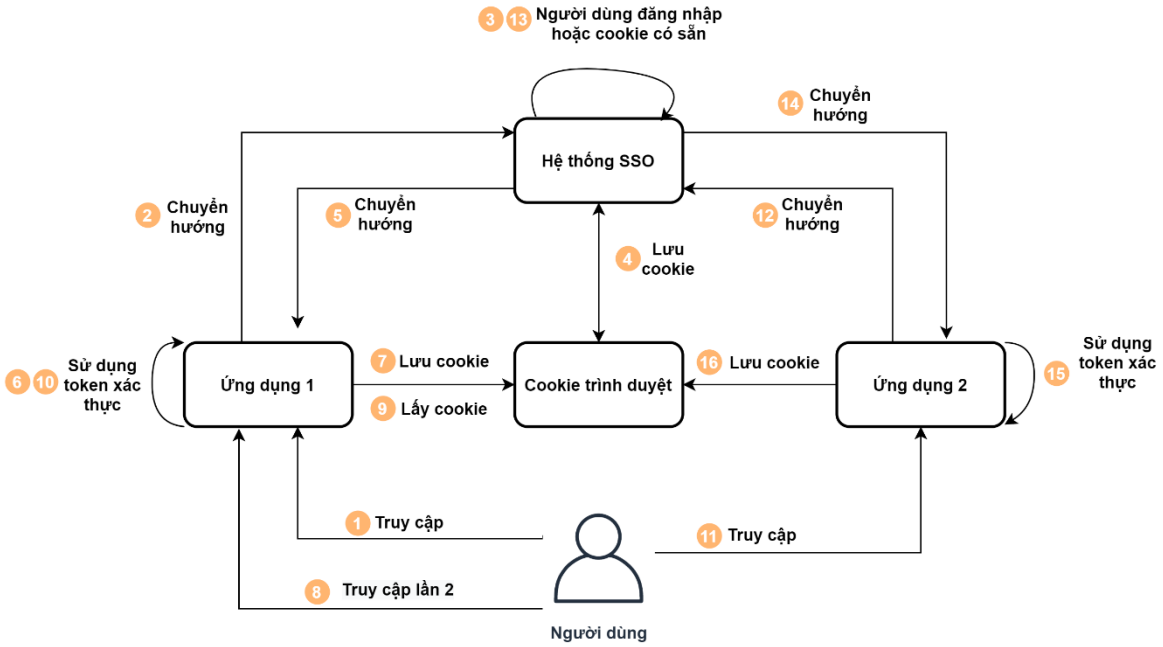
### 5.3.1 Đặt vấn đề

Thông thường, để bảo mật thì người dùng cần phải xác thực trước khi truy cập vào tài nguyên của một trang web. Người dùng có thể xác thực bằng các ứng dụng Google, Facebook… hoặc sử dụng tài khoản đăng nhập. Mỗi lần truy cập vào ứng dụng, hệ thống sẽ kiểm tra xem người dùng đã đăng nhập chưa. Nếu đã đăng nhập thì hệ thống cho phép truy nhập, nếu không thì người dùng sẽ phải đăng nhập. Do đó, với mỗi ứng dụng, người dùng đều phải có tài khoản đăng nhập.

Tuy nhiên điều này là chưa phù hợp đối với các ứng dụng liên kết với nhau. Người dùng chỉ cần sử dụng một tài khoản và đăng nhập một lần duy nhất là có thể sử dụng các ứng dụng khác.

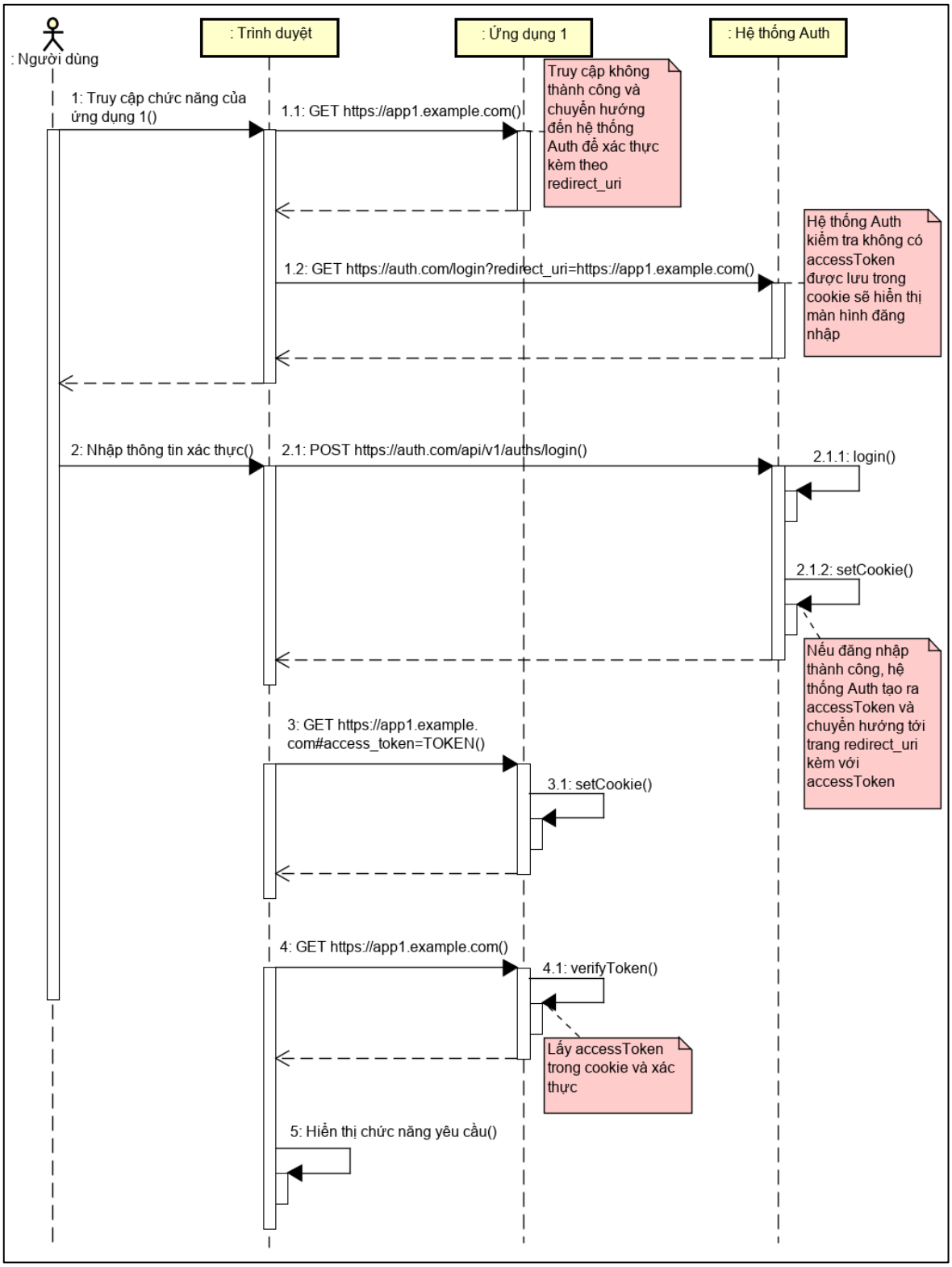
### 5.3.2 Giải pháp

Giải pháp của em đưa ra là xây dựng một hệ thống xác thực tập trung, tất cả các ứng dụng liên kết đều sử dụng hệ thống này để xác thực.




**Hình 58** Giải pháp xác thực người dùng tập trung [12].

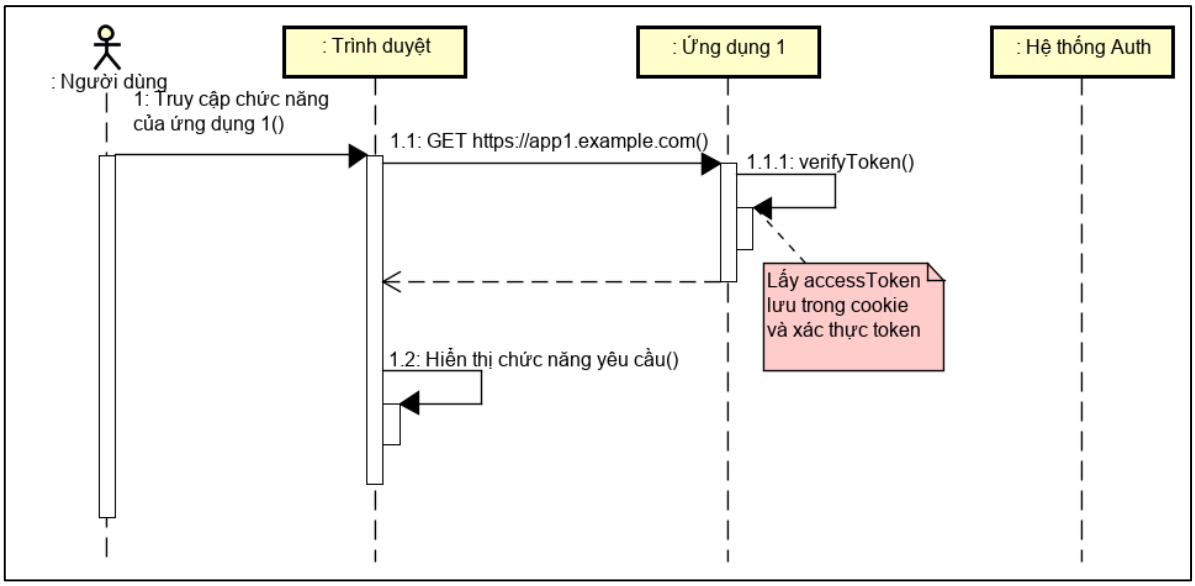
**Hình 58** mô tả khi người dùng truy cập vào ứng dụng 1 lần đầu tiên, ứng dụng sẽ chuyển hướng người dùng sang hệ thống xác thực tập trung. Sau khi đăng nhập, hệ thống sẽ lưu token vào cookie của trình duyệt, sau đó chuyển hướng người dùng về ứng dụng 1 kèm theo token. Ứng dụng sẽ kiểm tra token đó, nếu token hợp lệ thì ứng dụng sẽ lưu token vào cookie của trình duyệt và cho phép người dùng truy cập vào ứng dụng (**Hình 59**).




**Hình 59** Biểu đồ trình tự người dùng truy cập ứng dụng lần đầu.

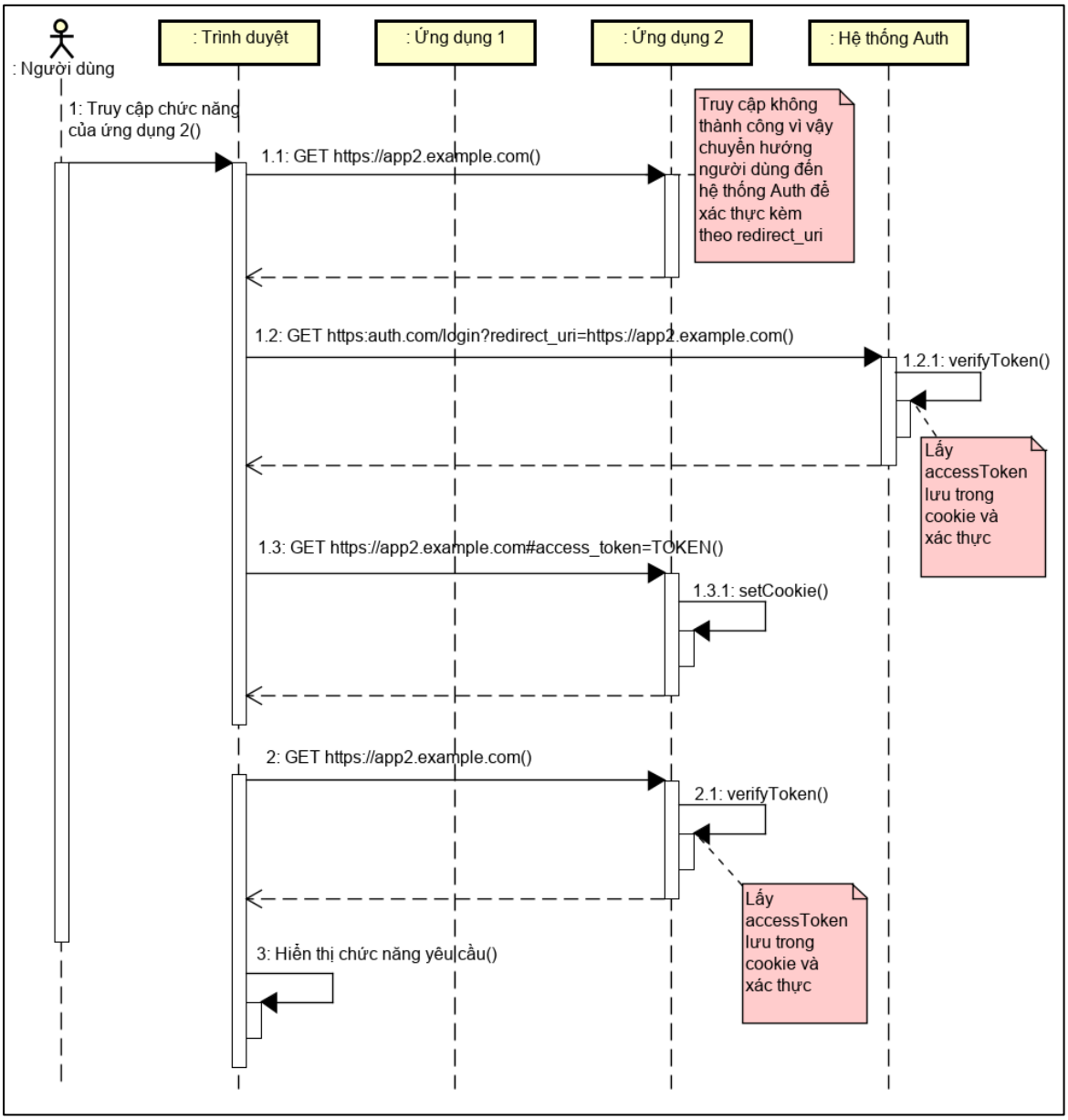
Khi người dùng truy cập vào ứng dụng 1 lần thứ hai, ứng dụng sẽ lấy token trong cookie ra và kiểm tra. Nếu token đó hợp lệ, người dùng sẽ được truy cập vào ứng dụng. Nếu không có

token hoặc token không hợp lệ, ứng dụng sẽ chuyển hướng người dùng đến hệ thống xác thực tập trung như mô tả ở trên (**Hình 60**).




**Hình 60** Biểu đồ trình tự người dùng truy cập ứng dụng lần thứ hai.

Khi người dùng truy cập vào ứng dụng 2, ứng dụng sẽ chuyển hướng người dùng đến hệ thống xác thực tập trung. Do trong cookie đã có token nên hệ thống kiểm tra token này, nếu hợp lệ sẽ chuyển hướng về ứng dụng 2 và trả về kèm theo token. Ứng dụng 2 kiểm tra token, nếu hợp lệ, ứng dụng sẽ lưu token vào cookie và cho phép người dùng truy cập ứng dụng như **Hình 61**.




**Hình 61** Biểu đồ trình tự người dùng truy cập ứng dụng liên kết.

### 5.3.3 Kết quả đạt được

Với giải pháp xác thực tập trung, chỉ cần một tài khoản và một lần đăng nhập, người dùng có thể truy cập vào tất cả các hệ thống liên kết.

## 5.4 Giải pháp phân quyền người dùng

### 5.4.1 Đặt vấn đề

#### 5.4.1.1 Một số mô hình phân quyền

Sau khi thực hiện xác thực, việc tiếp theo hệ thống cần làm là xác định xem người dùng đó có quyền thực hiện các hành động nào, được truy cập các tài nguyên nào trong hệ thống. Để thực hiện được điều đó, người quản trị cần xây dựng các hệ thống luật để phân quyền. Khi yêu cầu về nghiệp vụ tăng cao thì việc phân quyền càng trở lên phức tạp.

Có một vài cách tiếp cận để xây dựng hệ thống phân quyền, đó là (i) Access Control List - ACL, (ii) Role-Based Access Control – RBAC và (iii) Attribute-Based Access Control (ABAC).

Mô hình ACL xây dựng hệ thống quyền theo dạng ma trận của đối tượng và quyền, giống như cách quản lý tệp tin trên hệ điều hành Linux.

**Bảng 31** Ví dụ bảng phân quyền theo mô hình ACL

| | **Tạo bài viết** | **Xem bài viết** | **Sửa bài viết** | **Xóa bài viết** |
|---|---|---|---|---|
| Nguyễn Văn A | | x | | |
| Trần Văn B | x | x | x | |
| Nguyễn Văn C | x | x | x | x |

**Bảng 31** là ví dụ về cách phân quyền theo mô hình ACL. A có thể xem bài viết. B có quyền tạo bài viết, xem bài viết và sửa bài viết. C có thể tạo, xem, sửa và xóa bài viết.

Để phân quyền, người quản trị cần phân quyền cho từng người dùng trong hệ thống. Khi hệ thống lớn lên, số lượng người dùng trong hệ thống tăng lên, các tài nguyên trong hệ thống tăng lên, ma trận phân quyền cũng sẽ lớn lên theo cấp số nhân, việc quản lý quyền sẽ trở nên rất phức tạp. Do đó, mô hình ACL thường phù hợp với hệ thống có ít đối tượng và không có quá nhiều chức năng.

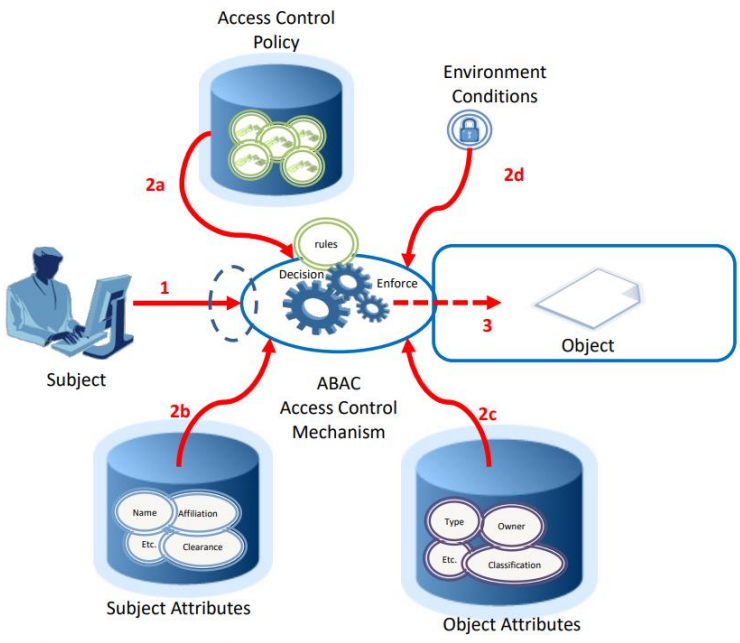
Trong mô hình RBAC, những người sử dụng giống nhau dựa trên nhiệm vụ công việc, vai trò xử lý thông tin sẽ được gán vào một vai trò (role), tương ứng với mỗi vai trò sẽ có các quyền thực hiện khác nhau. So với mô hình ACL, mô hình RBAC giúp giảm thiểu độ phức tạp khi số lượng đối tượng và quyền tăng cao. Tuy nhiên, trong một số trường hợp, mô hình RBAC không giải quyết được, ví dụ khi muốn cấp quyền chỉ người tạo bài viết mới được

sửa hoặc xóa bài viết, hoặc với cùng một vai trò, người dùng ở phòng ban A có quyền khác với người dùng ở phòng ban B, hoặc khi muốn chặn quyền truy cập của một người dùng có địa chỉ IP cụ thể.

Mô hình điều khiển truy cập dựa trên thuộc tính Attribute Based Access Control (ABAC) là phương thức điều khiển truy cập tài nguyên được xác định bởi một tập các thuộc tính (attribute) của chủ thể truy cập (subject), đối tượng truy cập (object) và môi trường (environment) dựa theo tập các luật (policy) được tạo thành từ việc kết hợp các thuộc tính trên với nhau.

Ví dụ với hệ thống tổng đài, subject là Tổng đài viên, Người quản lý. Object là các tài nguyên của hệ thống tổng đài, như danh sách liên hệ, danh sách kịch bản cuộc gọi. Enviroment là bối cảnh khi mà yêu cầu truy cập xảy ra, có thể là địa chỉ của người dùng, thời gian hiện tại, ngày làm việc trong tuần. Operation là các thao tác mà subject muốn thực hiện lên object, đó có thể là yêu cầu lấy danh sách liên hệ, yêu cầu xóa liên hệ, yêu cầu xem chi tiết kịch bản cuộc gọi.

**Hình 62** mô tả cách thức hoạt động của mô hình ABAC. Khi chủ thể subject gửi yêu cầu muốn truy cập đến đối tượng object, hệ thống xác định quyền truy cập bằng cách tính toán dựa trên các luật (rules), các thuộc tính của chủ thể (subject attributes), các thuộc tính của đối tượng (object attributes) và điều kiện môi trường (enviroment conditions). Nếu hợp lệ, chủ thể được cấp phép truy cập. Ngược lại, yêu cầu truy cập bị từ chối. Chính vì vậy, mô hình ABAC có thể phân quyền "mịn" hơn, phù hợp với nhiều ngữ cảnh và nghiệp vụ khác nhau.




**Hình 62** Mô hình phân quyền ABAC [13].

Theo kiến trúc đã nêu ra ở mục 5.3, hệ thống gồm phần frontend và phần backend trong đó chứa các service. Cả hai thành phần backend và frontend đều cần thực hiện việc phân quyền. Mô hình ACL không thể áp dụng trong trường hợp này vì lượng tài nguyên cần quản lý quyền truy cập trong hệ thống là lớn, và còn có thể tiếp tục tăng khi hệ thống mở rộng sau này. Nếu áp dụng mô hình RBAC, độ phức tạp của việc phân quyền được giảm đi đáng kể so với mô hình ACL. Tuy nhiên, mô hình này vẫn còn các nhược điểm như đã trình bày ở trên. Vì vậy em sẽ sử dụng kết hợp hai mô hình phân quyền RBAC và ABAC để vừa có thể gom nhóm các quyền lại, vừa có thể phân chia quyền chuyên sâu hơn.

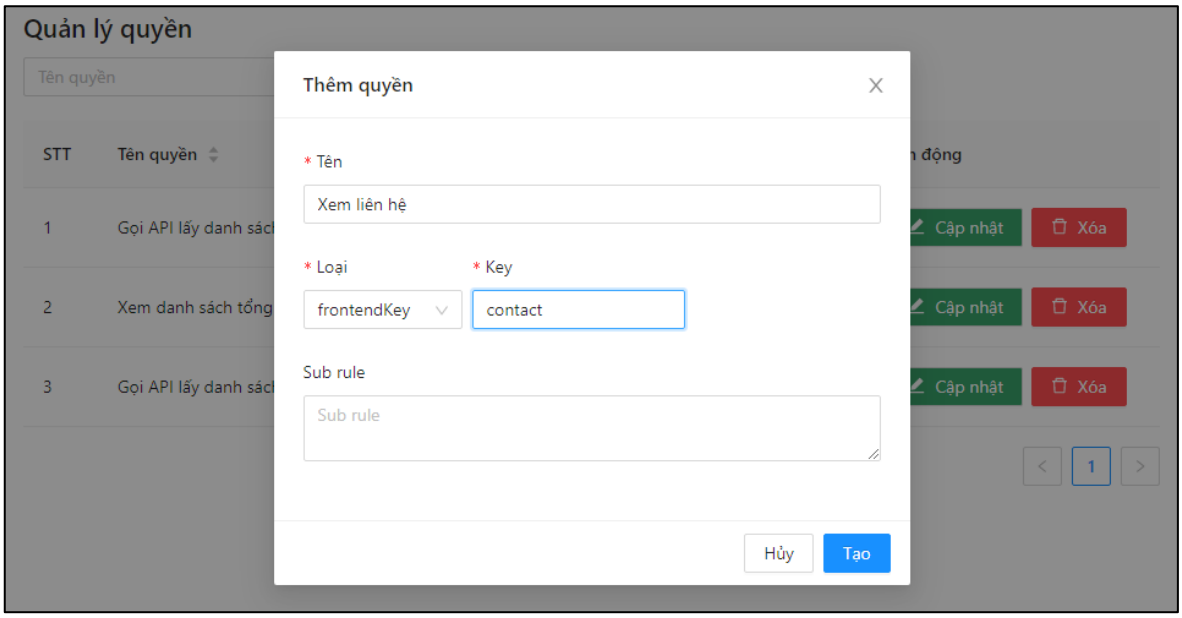
#### 5.4.1.2 Phân quyền tĩnh và phân quyền động

Bên cạnh đó, khi xây dựng hệ thống, trong phần mã nguồn, lập trình viên sẽ thưởng "phân quyền tĩnh". "Phân quyền tĩnh" trong mã nguồn backend là việc lập trình viên viết "cứng" trong đoạn mã nguồn nếu người dùng A có vai trò là X thì sẽ được thực hiện quyền Y và không được thực hiện quyền Z. "Phân quyền tĩnh" trong mã nguồn phía frontend là việc lập trình viên viết "cứng" trong đoạn mã nguồn nếu người dùng A có vai trò X thì sẽ hiển thị giao diện Y, ngược lại thì A không có quyền xem giao diện Y". Khi thêm quyền mới hoặc thay đổi logic việc phân quyền, nếu hệ thống sử dụng "phân quyền tĩnh" như trên thì toàn bộ mã nguồn phân quyền đó đều phải thay đổi, dẫn đến tốn kém chi phí nâng cấp và bảo trì.

### 5.4.2 Giải pháp

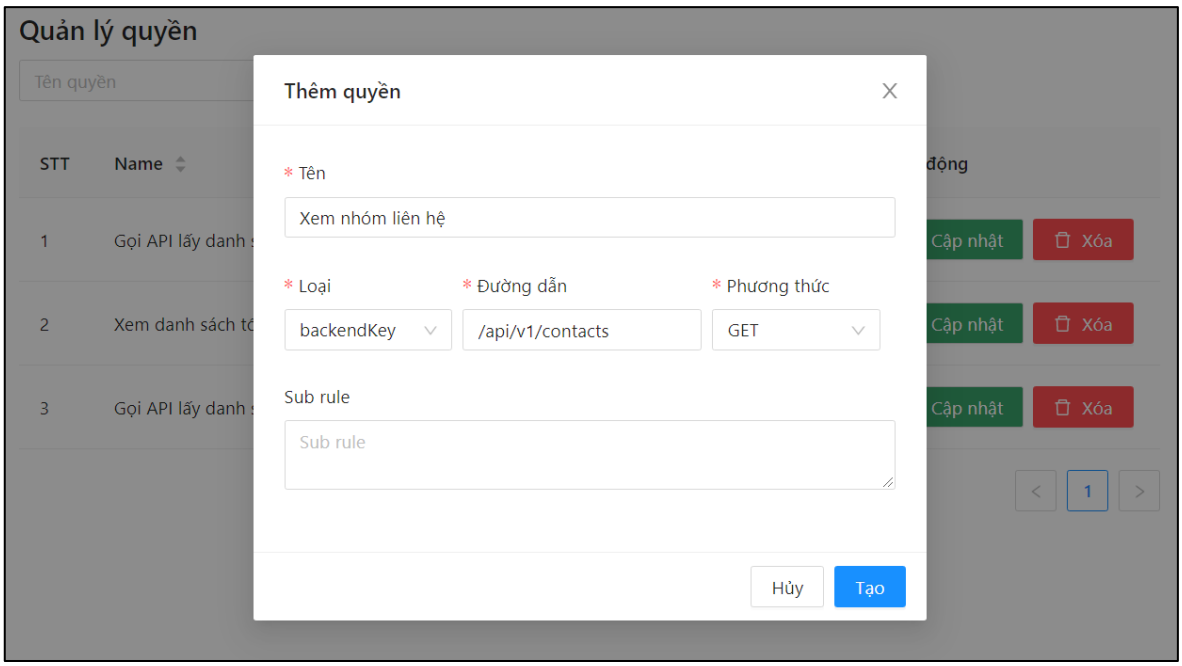
Ở phía backend, các API được định danh bằng tên phương thức (method) và đường dẫn (path). Ở phía frontend, trang web được chia nhỏ thành các thành phần (component) được định danh bằng key của component. Vì vậy, em thiết kế các quyền để gán cho người dùng. Một quyền truy cập có các thuộc tính là: (i) tên (name), (ii) loại (type) để xác định đối tượng truy cập thuộc về phía frontend hay phía backend, (iii) key để định danh component nếu là frontend và method + path nếu là phía backend, (iv) tập các luật mà người dùng định nghĩa (subRule). subRule là các điều kiện môi trường (environment conditions) để thu hẹp phạm vi của quyền mà người dùng tự định nghĩa ra, có thể là chỉ cho phép một dải địa chỉ IP nhất định, hoặc chỉ cho phép người tạo ra liên hệ có quyền xóa liên hệ đó, v.v.

Người dùng có thể cài đặt các quyền hiển thị các thành phần ở phía frontend như **Hình 63**.




**Hình 63** Chức năng thêm quyền hiển thị frontend.

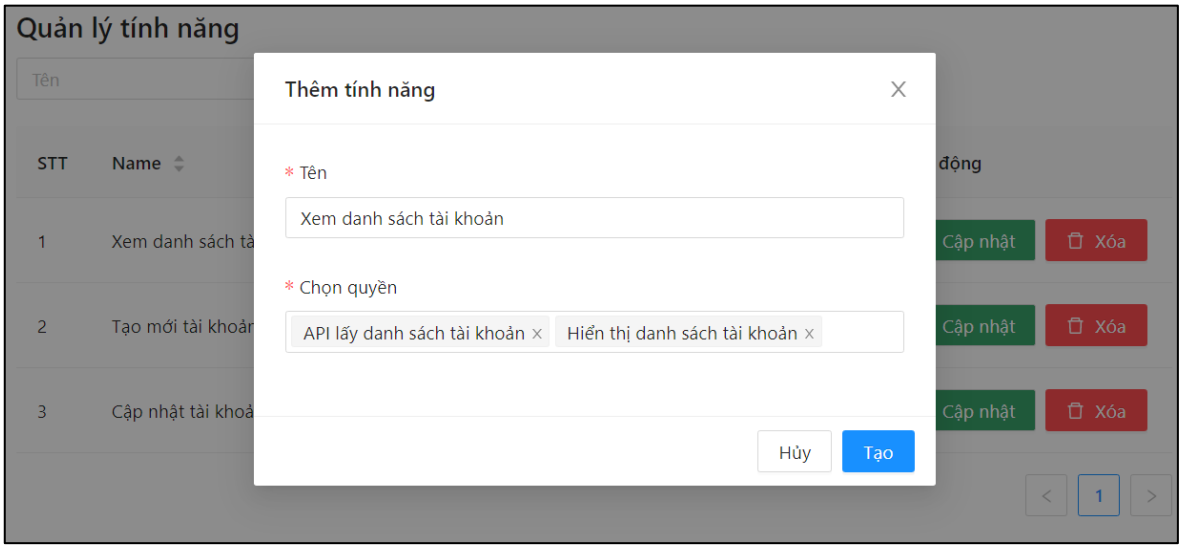
Nếu cài đặt quyền truy cập đến API phía backend thì người dùng cần cài đặt phương thức và đường dẫn cho API đó như **Hình 64**.




**Hình 64** Chức năng thêm quyền truy cập API.

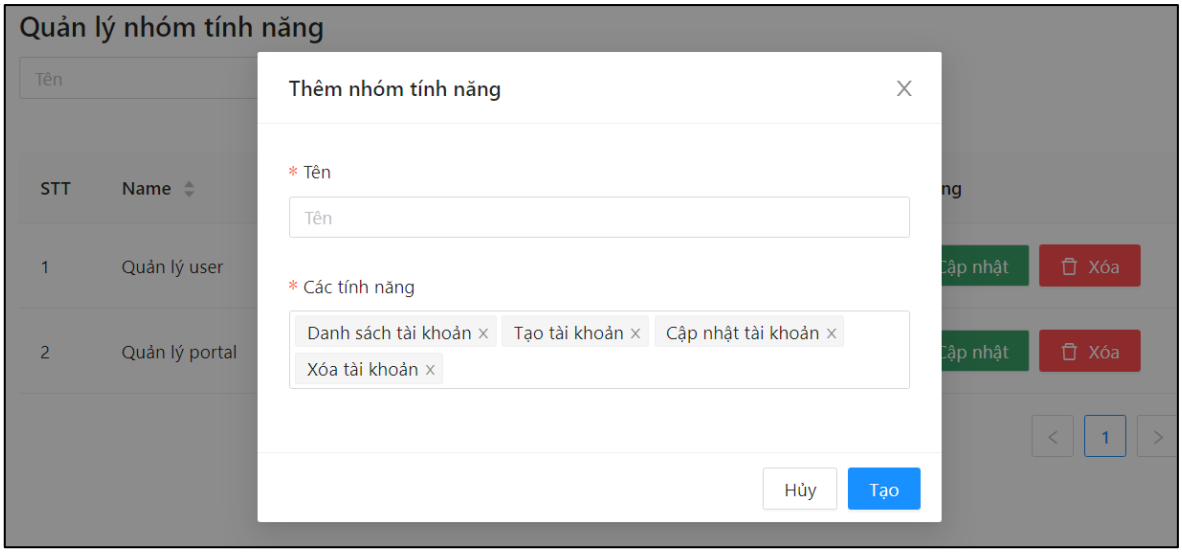
Lập trình viên là người xây dựng ra hệ thống và biết được các key của các thành phần của frontend, các phương thức và đường dẫn truy cập của các API phía backend. Quản trị viên là người quản trị hệ thống nên chỉ quan tâm đến các chức năng mà không cần biết tài nguyên đó thuộc frontend hay backend. Chính vì vậy, em đã thiết kế một tính năng Feature bao gồm

nhiều quyền bao gồm cả frontend và backend. Người dùng có thể tạo ra các tính năng bằng cách thêm các quyền ứng với các tính năng đó như **Hình 65**.

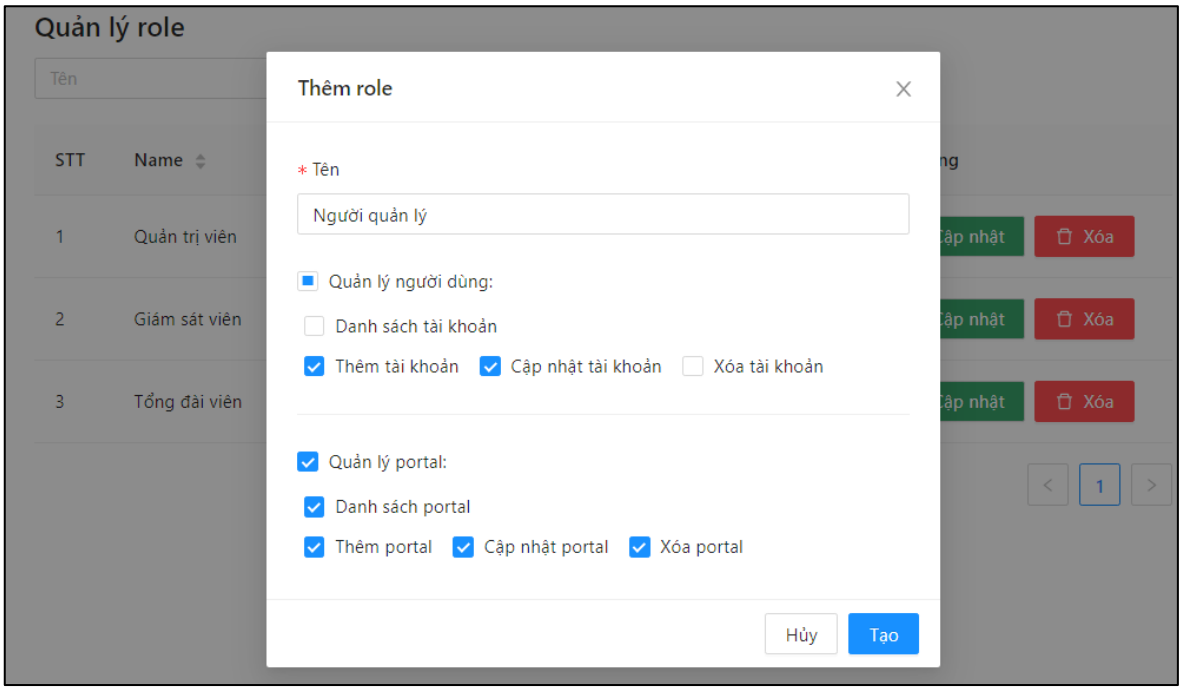



**Hình 65** Chức năng thêm mới tính năng.

Ngoài ra, để thuận tiện cho người dùng trong quá trình phân quyền, em thiết kế ra các Nhóm tính năng (FeatureGroup) và các Vai trò (Role) bao gồm nhiều nhóm tính năng và các tính năng. Khi người dùng tạo ra vai trò mới, người dùng sẽ chọn ra các nhóm tính năng và có thể thêm bớt các tính năng trong nhóm đó. **Hình 66** thể hiện chức năng thêm nhóm tính năng và **Hình 67** thể hiện chức năng thêm vai trò vào hệ thống.




**Hình 66** Chức năng thêm nhóm tính năng.




**Hình 67** Chức năng thêm vai trò.

Các hệ thống liên quan, ví dụ hệ thống quản lý kịch bản cuộc gọi cũng nhận được các luật tương ứng do chúng chia sẻ với nhau qua hệ thống hàng đợi đã được trình bày ở phần 5.2. Khi đó, người quản lý portal sẽ có thể gán vai trò cho các thành viên để thực hiện việc phân quyền.

Khi người dùng truy cập đến các chức năng của hệ thống, hệ thống sẽ độc lập kiểm tra xem người dùng có quyền truy cập hay không mà không cần gọi ra hệ thống bên ngoài. Nếu không có quyền, hệ thống sẽ không hiển thị các giao diện tương ứng với quyền và cũng không được phép truy cập các API ứng với tính năng đó.

Trong Chương 5, em trình bày về những đóng góp nổi bật mà em tâm đắc nhất trong quá trình làm đồ án. Để xây dựng một công cụ trực quan giúp người dùng dễ dàng hình dung được kịch bản cuộc gọi, em đã sử dụng thư viện Ej2 React Diagram để tạo ra các biểu đồ luồng có thể thao tác trực tiếp trên đó. Cùng với đó, em thiết kế hệ thống theo mô hình kiến trúc Microservices để hệ thống dễ dàng bảo trì và mở rộng về sau. Cùng với đó, em đã áp dụng cơ chế xác thực tập trung CAS, mô hình phân quyền RBAC kết hợp ABAC để xây dựng hệ thống xác thực và phân quyền tập trung.

# Chương 6 Kết luận và hướng phát triển

## 6.1 Kết luận

Sau quá trình làm đồ án tốt nghiệp, em đã phát triển được hệ thống quản lý kịch bản cuộc gọi, xác thực và phân quyền tập trung. Hệ thống được thiết kế theo kiến trúc Microservices để có thể linh hoạt sửa đổi mà mở rộng sau này. Hiện tại, hệ thống đang được triển khai thử nghiệm tại địa chỉ https://inbound.iristech.club và https://dashboard-inbound.iristech.club.

Để phát triển hệ thống quản lý kịch bản cuộc gọi, em đã tìm hiểu, phân tích các hệ thống tổng đài thông minh trên thị trường, tìm hiểu các thư viện và công cụ hỗ trợ. Qua đó, em đã phát triển một công cụ quản lý kịch bản cuộc gọi bằng thư viện Ej2 React Diagram. Thay vì việc hiển thị kịch bản cuộc gọi ở dạng danh sách, hệ thống thể hiện dưới dạng biểu đồ luồng để người dùng có cái nhìn tổng quan về luồng cuộc gọi. Người dùng có thể cấu hình những kịch bản cuộc gọi phức tạp, chọn nhiều hành động cho một phím bấm. Cùng với đó, hệ thống sử dụng công nghệ chuyển văn bản thành giọng nói, có thể cá nhân hóa nội dung trao đổi với khách hàng. Thêm nữa, hệ thống có thể kết nối đến Voice Bot để chăm sóc khách hàng tự động mà không cần sự tham gia của tổng đài viên.

Trong quá trình phát triển hệ thống, em cũng xây dựng một hệ thống xác thực và phân quyền tập trung. Hệ thống áp dụng cơ chế xác thực tập trung CAS để quản lý chung thông tin người dùng trên các ứng dụng liên kết. Ngoài ra, hệ thống sử dụng mô hình phân quyền RBAC kết hợp với mô hình ABAC cho phép quản trị viên có thể dễ dàng quản lý quyền truy cập của người dùng trong hệ thống.

Em đã học hỏi được rất nhiều kiến thức, kinh nghiệm sau khi hoàn thành đồ án. Đó là kỹ năng phân tích, thiết kế, xây dựng hệ thống, kỹ năng làm việc nhóm, viết báo cáo. Đây là những kinh nghiệm vô cùng quý báu với em sau khi tốt nghiệp.

## 6.2 Hướng phát triển

Hệ thống quản lý kịch bản, xác thực và phân quyền người dùng tập trung đang được hoàn thiện, cải tiến và phát triển trong tương lai.

Trước hết, em sẽ hoàn thiện hệ thống về phần UI/UX để nâng cao trải nghiệm của người dùng.Tiếp theo đó, em sẽ tích hợp hệ thống quản lý kịch bản cuộc gọi với hệ thống gọi ra Call In sẵn có để hỗ trợ quản lý kịch bản cuộc gọi cho hệ thống này. Cuối cùng, em sẽ tích

hợp hệ thống xác thực và phân quyền tập trung với một số hệ thống khác để quản lý người dùng tập trung.

# Tài liệu tham khảo


[1] "OMICall – Tổng Đài Cloud Thông Minh". https://omicall.com (truy cập tháng 1 02, 2021).

[2] "StringeeX - OmniChannel Contact Center. Customer relationship management call center. Better sales, lower cost." https://stringeex.com/ (truy cập tháng 1 02, 2021).

[3] "mAICallCenter", *Trung tâm CNTT MobiFone*. https://itc.mobifone.vn/giai_phap/maicallcenter/ (truy cập tháng 1 02, 2021).

[4] "React – A JavaScript library for building user interfaces". https://reactjs.org/ (truy cập tháng 1 10, 2021).

[5] "Ant Design - The world's second most popular React UI framework". https://ant.design/ (truy cập tháng 1 10, 2021).

[6] "React Diagram component - Syncfusion". https://ej2.syncfusion.com/react/documentation/diagram/ (truy cập tháng 1 10, 2021).

[7] Node.js, "Node.js", *Node.js*. https://nodejs.org/en/ (truy cập tháng 1 10, 2021).

[8] "Express - Node.js web application framework". https://expressjs.com/ (truy cập tháng 1 10, 2021).

[9] "The most popular database for modern apps", *MongoDB*. https://www.mongodb.com (truy cập tháng 1 10, 2021).

[10] "Redis". https://redis.io/ (truy cập tháng 1 10, 2021).

[11] "Messaging that just works — RabbitMQ". https://www.rabbitmq.com/ (truy cập tháng 1 10, 2021).

[12] Auth0, "Single Sign-On", *Auth0 Docs*. https://auth0.com/docs/ (truy cập tháng 1 04, 2021).

[13] V. C. Hu *và c.s.*, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations.(2014)", *NIST Spec. Publ. SP*, 2014.

# Phụ lục

## A Kịch bản kiểm thử

### A.1 Kiểm thử chức năng thêm người dùng vào portal

| STT | Mô tả | Quy trình kiểm thử | Kết quả mong muốn | Kết quả thực tế | Trạng thái |
|---|---|---|---|---|---|
| 1. | Xác thực các trường thông tin | Bỏ trống các trường thông tin<br>Nhấn nút Thêm | Hiển thị lỗi: Các trường thông tin là bắt buộc | Hiển thị lỗi: Các trường thông tin là bắt buộc | Đạt |
| 2. | Xác thực trường email | Nhập email sai định dạng, trường vai trò chọn đầy đủ<br>Nhấn nút Thêm | Hiển thị lỗi: Email không đúng định dạng | Hiển thị lỗi: Email không đúng định dạng | Đạt |
| 3. | Xác thực trường vai trò | Bỏ trống vai trò, trường email nhập đầy đủ và đúng định dạng<br>Nhấn nút Thêm | Hiển thị lỗi: Vai trò người dùng không được để trống | Hiển thị lỗi: Vai trò người dùng không được để trống | Đạt |
| 4. | Kiểm tra mail lời mời | Nhập đủ các trường thông tin, đúng định dạng | Nhận được lời mời qua email | Nhận được lời mời qua email | Đạt |
| 5. | Thêm mới thành công người dùng | Nhấn vào đường link trong lời mời | Người dùng được thêm vào danh sách | Người dùng được thêm vào danh sách | Đạt |

### A.2 Kiểm thử chức năng cập nhật thông tin liên hệ

| STT | Mô tả | Quy trình kiểm thử | Kết quả mong muốn | Kết quả thực tế | Trạng thái |
|---|---|---|---|---|---|
| 1. | Xác thực trường tên liên hệ | Bỏ trống trường tên liên hệ, các trường khác nhập đầy đủ, đúng định dạng<br>Nhấn nút Cập nhật | Hiển thị lỗi: Liên hệ là bắt buộc | Hiển thị lỗi: Liên hệ là bắt buộc | Đạt |
| 2. | Xác thực trường số điện thoại | Nhập số điện thoại sai định dạng, các trường khác nhập đầy đủ, đúng định dạng<br>Nhấn nút Cập nhật | Hiển thị lỗi: Số điện thoại không đúng định dạng | Hiển thị lỗi: Số điện thoại không đúng định dạng | Đạt |
| 3. | Xác thực trường email | Nhập email sai định dạng, các trường khác nhập đầy đủ, đúng định dạng<br>Nhấn nút Cập nhật | Hiển thị lỗi: Email không đúng định dạng | Hiển thị lỗi: Email không đúng định dạng | Đạt |
| 4. | Cập nhật liên hệ thành công | Nhập đủ các trường thông tin, đúng định dạng | Liên hệ được thêm vào danh sách | Liên hệ được thêm vào danh sách | Đạt |

## A.3 Kiểm thử chức năng xóa số hotline

| STT | Mô tả | Quy trình kiểm thử | Kết quả mong muốn | Kết quả thực tế | Trạng thái |
|---|---|---|---|---|---|
| 1. | Kiểm tra hộp thoại xác nhận | Nhấn nút xóa liên hệ trên màn danh sách | Hiển thị hộp thoại xác nhận | Hiển thị hộp thoại xác nhận | Đạt |
| 2. | Xóa số hotline thành công | Nhấn nút đồng ý trên hộp thoại xác nhận | Số hotline bị xóa khỏi danh sách | Số hotline bị xóa khỏi danh sách | Đạt |