

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA CÔNG NGHỆ PHẦN MỀM**

**HOÀNG TIẾN ĐẠT**  
**TRẦN DIỆU NHẤT HẠNH**

**KHÓA LUẬN TỐT NGHIỆP**  
**NGHIÊN CỨU COUCHBASE SERVER**  
**ĐỂ XÂY DỰNG MẠNG XÃ HỘI**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2016**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA CÔNG NGHỆ PHẦN MỀM**

**HOÀNG TIẾN ĐẠT – 11520051**

**TRẦN DIỆU NHẤT HẠNH – 11520099**

**KHÓA LUẬN TỐT NGHIỆP**  
**NGHIÊN CỨU COUCHBASE SERVER**  
**ĐỂ XÂY DỰNG MẠNG XÃ HỘI**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**  
**THS. NGUYỄN CÔNG HOAN**

**TP. HỒ CHÍ MINH, 2016**

## **DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN**

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số 16/QĐ-ĐHCNTT-ĐTĐH, ngày 19 tháng 01 năm 2016 của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. TS. Nguyễn Tấn Trần Minh Khang – Chủ tịch.
2. ThS. Huỳnh Tuấn Anh – Thư ký.
3. ThS. Phạm Thị Vương – Ủy viên.

TP. HCM, ngày 25 tháng 1 năm 2016

## NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

### (CỦA CÁN BỘ HƯỚNG DẪN)

#### Tên khóa luận:

NGHIÊN CỨU COUCHBASE SERVER ĐỂ XÂY DỰNG MẠNG XÃ HỘI

#### Nhóm SV thực hiện:

Hoàng Tiến Đạt 11520051

Trần Diệu Nhất Hạnh 11520099

#### Cán bộ hướng dẫn:

ThS. Nguyễn Công Hoan

#### Đánh giá Khóa luận:

##### 1. Về cuốn báo cáo:

Số trang	117	Số chương	5
Số bảng số liệu	32	Số hình vẽ	47
Số tài liệu tham khảo	10	Sản phẩm	1

##### Một số nhận xét về hình thức cuốn báo cáo:

Cuốn báo cáo có hình thức tốt, nội dung rõ ràng. Phần kiến trúc thiết kế, các vấn đề gãy phai và hướng giải quyết được viết rất rõ ràng, cụ thể, thể hiện rõ được vấn đề mà khóa luận cần giải quyết.

##### 2. Về nội dung nghiên cứu:

Khóa luận nghiên cứu, tập trung vào các giải pháp, công nghệ, và các vấn đề trong thiết kế và xây dựng hệ thống phân tán để đáp ứng được lượng người dùng lớn. Thông qua đó phát triển và xây dựng hệ thống mạng xã hội và ứng dụng mạng xã hội trên hệ điều hành Windows 10 Mobile. Khóa luận có có hướng nghiên cứu rõ

ràng, đạt được kết quả về việc xây dựng hệ thống lớn như đã đề ra, tạo tiền đề cho việc phát triển và đưa hệ thống vào vận hành trong thực tế.

### 3. Về chương trình ứng dụng:

Về hệ thống mạng xã hội, khóa luận đã phát triển được hệ thống có kiến trúc phân tán tốt, có khả năng dễ dàng mở rộng theo chiều ngang đáp ứng được nhu cầu về lượng lớn người sử dụng trong thực tế. Về ứng dụng mạng xã hội, khóa luận đã phát triển được các tính năng cơ bản của mạng xã hội, kết hợp với các tính năng tương tác bằng giọng nói, giúp hỗ trợ người dùng tốt hơn trong quá trình sử dụng và tham gia vào mạng xã hội:

- Chức năng SOS cảnh báo nguy hiểm
- Cập nhật trạng thái (status)
- Xem thông tin profile
- Và các chức năng có hỗ trợ điều khiển bằng giọng nói.

### 4. Về thái độ làm việc của sinh viên:

Nhóm sinh viên tích cực trong việc tìm hiểu công nghệ, kiến trúc phân tán để xây dựng hệ thống phân tán, và các công nghệ nhận dạng giọng nói để ứng dụng vào mạng xã hội. Tuy nhiên, các em vẫn chưa theo kịp được tiến độ mong muốn theo giảng viên yêu cầu. Nhóm sinh viên cần tiếp tục xây dựng và phát triển hệ thống, đưa hệ thống vào vận hành trong thực tế.

#### **Đánh giá chung:**

Khóa luận đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư chuyên ngành Công Nghệ Phần Mềm.

#### **Điểm từng sinh viên:**

Hoàng Tiến Đạt : ...../10

Trần Diệu Nhất Hạnh : ...../10

**Người nhận xét**

(Ký tên và ghi rõ họ tên)

TP. HCM, ngày 25 tháng 1 năm 2016

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP  
(CỦA CÁN BỘ PHẢN BIỆN)**

**Tên khóa luận:**

**NGHIÊN CỨU COUCHBASE SERVER ĐỂ XÂY DỰNG MẠNG XÃ HỘI**

**Nhóm SV thực hiện:**

Hoàng Tiến Đạt	11520051
Trần Diệu Nhất Hạnh	11520099

**Cán bộ phản biện:**

ThS. Phạm Thị Vương

**Danh giá Khóa luận:**

1. Về cuốn báo cáo:

Số trang	117	Số chương	5
Số bảng số liệu	32	Số hình vẽ	47
Số tài liệu tham khảo	10	Sản phẩm	1

Một số nhận xét về hình thức cuốn báo cáo:

Báo cáo có nội dung cụ thể, rõ ràng. Phần các vấn đề gấp phải và hướng giải quyết được viết rất chi tiết, thể hiện rõ ràng vấn đề đặt ra cho hệ thống.

2. Về nội dung nghiên cứu:

Khóa luận tập trung nghiên cứu về các giải pháp, công nghệ phân tán, đặc biệt là Couchbase Server vào phát triển hệ thống mạng xã hội. Hệ thống có kiến trúc phân tán tốt, có khả năng mở rộng dễ dàng để có thể đáp ứng lượng lớn người dùng của mạng xã hội.

3. Về chương trình ứng dụng:

Nhóm thực hiện đề tài nghiên cứu Couchbase Server để xây dựng mạng xã hội. Về phần hệ thống, hệ thống có kiến trúc phân tán và khả năng mở rộng tốt, có khả năng đáp ứng yêu cầu về lượng lớn người dùng của mạng xã hội. Về phần ứng dụng, ứng dụng cung cấp đầy đủ tính năng của một mạng xã hội cơ bản, có giao diện đẹp và trải nghiệm người dùng tốt.

#### 4. Về thái độ làm việc của sinh viên:

Sinh viên có thái độ làm việc nghiêm túc, chịu khó tiếp thu và tích cực giải quyết những vấn đề do giảng viên phản biện đặt ra.

#### **Đánh giá chung:**

Từ những kết quả đạt được cho thấy sinh viên năm bắt được phương pháp tự học, tự nghiên cứu tìm hiểu và giải quyết vấn đề. Kết quả khóa luận đáp ứng được yêu cầu của khóa luận tốt nghiệp kỹ sư ngành Công Nghệ Phần Mềm. Nhóm cần tiếp tục phát triển và hoàn thiện hệ thống để đưa hệ thống vào vận hành trong thực tế.

#### **Điểm từng sinh viên:**

Hoàng Tiến Đạt : ...../10

Trần Diệu Nhất Hạnh : ...../10

**Người nhận xét**

(Ký tên và ghi rõ họ tên)

## **LỜI CẢM ƠN**

Trước hết nhóm chúng em xin gửi lời cảm ơn tới các thầy, cô khoa Công Nghệ Phần Mềm nói riêng và các thầy, cô trường Đại học Công Nghệ Thông Tin – ĐHQG Tp. Hồ Chí Minh nói chung, những người đã cung cấp cho chúng em những nền tảng kiến thức cơ bản cũng như chuyên ngành để chúng em có thể hoàn thành khóa luận tốt nghiệp.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Nguyễn Công Hoan, người đã tận tình hướng dẫn và tạo điều kiện tốt nhất cho chúng em trong suốt quá trình hoàn thành khóa luận này.

Chúng em cũng xin cảm ơn thầy Phạm Thị Vương đã phản biện và có những góp ý chân thành để đề tài của nhóm có thể hoàn thiện thêm.

Cuối cùng, chúng em cũng xin gửi lời cảm ơn đến gia đình, người thân và bạn bè đã quan tâm, ủng hộ, góp ý để nhóm có thể tập trung vào việc thực hiện khóa luận tốt hơn.

Chúng em xin chân thành cảm ơn!

**TP. Hồ Chí Minh, tháng 12 năm 2015**

**Nhóm sinh viên thực hiện**

**Hoàng Tiến Đạt – Trần Diệu Nhất Hạnh**

TP. HCM, ngày 20 tháng 1 năm 2016

## ĐỀ CƯƠNG CHI TIẾT

### TÊN ĐỀ TÀI: NGHIÊN CỨU COUCHBASE SERVER ĐỂ XÂY DỰNG MẠNG XÃ HỘI

Cán bộ hướng dẫn: ThS. Nguyễn Công Hoan

Thời gian thực hiện: Từ ngày 1/10/2015 đến ngày 30/12/2015

Sinh viên thực hiện:

Hoàng Tiến Đạt – 11520051

Trần Diệu Nhất Hạnh – 11520099

Nội dung đề tài:

- Nghiên cứu Couchbase Server và ứng dụng nó vào xây dựng hệ thống kiến trúc phân tán của mạng xã hội.
- Xây dựng một hệ thống mạng xã hội có khả năng đáp ứng được một lượng lớn người dùng tại một thời điểm.
- Phát triển và tích hợp thêm các tính năng tương tác bằng giọng nói vào mạng xã hội.

Kế hoạch thực hiện:

**Giai đoạn 1 (từ ngày 1/10/2015 đến 4/11/2015):** Nghiên cứu Couchbase Server và ứng dụng Couchbase Server để xây dựng kiến trúc hệ thống mạng xã hội. Tập trung vào các vấn đề và giải pháp xây dựng hệ thống lớn, đáp ứng được lượng lớn người sử dụng. Xây dựng các tính năng cơ bản của mạng xã hội.

**Giai đoạn 2 (từ ngày 5/11/2015 đến 18/11/2015):** Hoàn thiện các tính năng của mạng xã hội.

**Giai đoạn 3 (từ ngày 19/11/2015 đến 1/12/2015):** Nghiên cứu cơ bản về các thuật toán và kỹ thuật trong nhận dạng giọng nói. Tìm hiểu và nghiên cứu các thư viện nhận dạng giọng nói trên Windows 10 Mobile.

**Giai đoạn 3 (từ ngày 2/11/2015 đến 20/12/2015):** Tập trung nghiên cứu và cài đặt các tính năng tương tác bằng giọng nói của mạng xã hội, hoàn thiện các tính năng cơ bản.

**Giai đoạn 4 (từ ngày 21/12/2015 đến 24/01/2016):** Hoàn thiện hệ thống, đề xuất và bổ sung thêm những tính năng mới.

<b>Xác nhận của CBHD</b> (Ký tên và ghi rõ họ tên)	<b>TP. HCM, ngày 20 tháng 1 năm 2016</b> <b>Sinh viên 1</b> (Ký tên và ghi rõ họ tên)
	<b>Sinh viên 2</b> (Ký tên và ghi rõ họ tên)

## MỤC LỤC

LỜI CẢM ƠN .....	viii
DANH MỤC HÌNH VẼ .....	xvii
DANH MỤC BẢNG .....	xx
DANH MỤC TỪ VIẾT TẮT .....	xxii
MỞ ĐẦU .....	1
Chương 1. GIỚI THIỆU .....	2
1.1. Lý do chọn đề tài.....	2
1.2. Tình hình nghiên cứu.....	3
1.3. Đối tượng và phạm vi nghiên cứu.....	6
1.3.1. Đối tượng nghiên cứu .....	6
1.3.2. Phạm vi nghiên cứu .....	6
1.4. Mục tiêu đề tài.....	6
1.4.1. Mục tiêu tổng quát .....	6
1.4.2. Mục tiêu cụ thể .....	7
1.5. Phạm vi công nghệ .....	7
1.6. Cấu trúc khóa luận.....	7
Chương 2. CÔNG NGHỆ ĐƯỢC SỬ DỤNG ĐỂ XÂY DỰNG ĐỀ TÀI .....	8
2.1. Nền tảng phát triển ứng dụng di động Windows 10 UWP .....	8
2.1.1. Windows Core .....	8
2.1.2. Universal Windows Platform .....	10
2.1.3. Windows 10 Mobile .....	11
2.2. Công nghệ nhận dạng giọng nói .....	12

2.2.1.	Cách thức xây dựng dịch vụ nhận dạng, điều khiển bằng giọng nói	12
2.2.2.	Mô hình triển khai công nghệ giọng nói .....	12
2.2.3.	Speech platform trên Windows 10 UWP .....	13
2.2.3.1.	Speech Synthesis.....	13
2.2.3.2.	Speech Recognition.....	13
2.2.3.3.	Cortana Interaction.....	14
2.3.	Jersey Restful Framework.....	14
2.3.1.	Giới thiệu.....	14
2.3.2.	Lý do lựa chọn.....	14
2.4.	Couchbase Server.....	15
2.4.1.	Giới thiệu.....	15
2.4.2.	Kiến trúc deployment mức cao.....	15
2.4.3.	Kiến trúc ở client .....	18
2.4.4.	Query engine .....	19
2.4.5.	Các đặc điểm nổi trội của Couchbase Server .....	23
2.4.6.	Một case study điển hình so sánh giữa Couchbase Server và MongoDB .....	24
2.4.6.1.	Giới thiệu về viber case study .....	24
2.4.6.2.	Kiến trúc ban đầu .....	25
2.4.6.3.	Kiến trúc sau khi chuyển đổi sang Couchbase: .....	26
2.5.	Netty .....	26
2.5.1.	Giới thiệu.....	26
2.5.2.	Kiến trúc .....	27
2.5.3.	Đặc tính .....	27

2.5.4. Lý do lựa chọn.....	28
2.6. Redis .....	29
2.6.1. Giới thiệu.....	29
2.6.2. Lý do lựa chọn.....	29
2.7. Amazon S3 .....	30
<b>Chương 3. CÁC VẤN ĐỀ GẶP PHẢI VÀ HƯỚNG GIẢI QUYẾT .....</b>	<b>31</b>
3.1. Khả năng đáp ứng lượng lớn người dùng của một mạng xã hội .....	31
3.1.1. Phát biểu vấn đề .....	31
3.1.2. Giải pháp giải quyết.....	32
3.2. Xây dựng news feed.....	37
3.2.1. Phát biểu vấn đề .....	37
3.2.2. Giải pháp giải quyết.....	37
3.3. Tối ưu hóa truy xuất CSDL.....	39
3.3.1. Phát biểu vấn đề .....	39
3.3.2. Giải pháp giải quyết.....	39
3.4. Đề xuất giải pháp và mô hình thiết kế kiến trúc hệ thống .....	41
3.4.1. Mô hình nhiều lớp .....	41
3.4.1.1. Giới thiệu .....	41
3.4.1.2. Phân tích lựa chọn .....	42
3.5. Tích hợp tương tác giọng nói vào mạng xã hội.....	43
<b>Chương 4. PHÂN TÍCH, THIẾT KẾ.....</b>	<b>44</b>
4.1. Mô tả hệ thống .....	44
4.2. Yêu cầu và mô hình hóa yêu cầu.....	45
4.2.1. Danh sách yêu cầu.....	45

4.2.2.	Mô hình use-case .....	47
4.2.3.	Đặc tả use-case .....	48
4.2.3.1.	Đặc tả use-case “Xem danh sách kết nối” .....	48
4.2.3.2.	Đặc tả use-case “Gửi yêu cầu tạo kết nối” .....	48
4.2.3.3.	Đặc tả use-case “Chấp nhận/hủy yêu cầu tạo kết nối” .....	49
4.2.3.4.	Đặc tả use-case “Xem danh sách các cuộc hội thoại” .....	51
4.2.3.5.	Đặc tả use-case “Xem lịch sử tin nhắn” .....	52
4.2.3.6.	Đặc tả use-case “Gửi tin nhắn thông thường” .....	52
4.2.3.7.	Đặc tả use-case “Gửi file/hình ảnh” .....	53
4.2.3.8.	Đặc tả use-case “Gửi tin nhắn thoại” .....	55
4.2.3.9.	Đặc tả use-case “Đăng status hình ảnh” .....	56
4.2.3.10.	Đặc tả use-case “Đăng status” .....	57
4.2.3.11.	Đặc tả use-case “Đăng voice status” .....	58
4.2.3.12.	Đặc tả use-case “Like status” .....	59
4.2.3.13.	Đặc tả use-case “Comment trên status” .....	60
4.2.3.14.	Đặc tả use-case “Duyệt news feed” .....	60
4.2.3.15.	Đặc tả use-case “Xem những gì xảy ra xung quanh mình” .....	61
4.2.3.16.	Đặc tả use-case “Đăng status bằng voice command” .....	62
4.2.3.17.	Đặc tả use-case “Gửi tin nhắn bằng voice command” .....	63
4.2.3.18.	Đặc tả use-case “Chuyển sang tình trạng nguy hiểm & nhờ sự trợ giúp bằng voice command” .....	64
4.2.3.19.	Đặc tả use-case “Thay đổi trạng thái bằng voice command” .....	65
4.2.3.20.	Đặc tả use-case “Truy vấn thông tin bạn bè, người thân bằng voice command” .....	66

4.2.3.21. Đặc tả use-case “Nhờ sự trợ giúp” .....	67
4.2.4. Sequence diagram.....	68
4.2.4.1. Sequence diagram cho use-case “Xem danh sách kết nối” .....	68
4.2.4.2. Sequence diagram cho use-case “Gửi yêu cầu tạo kết nối” .....	70
4.2.4.3. Sequence diagram cho use-case “Chấp nhận/hủy yêu cầu tạo kết nối”	71
4.2.4.4. Sequence diagram cho use-case “Xem danh sách các cuộc hội thoại”	73
4.2.4.5. Sequence diagram cho use-case “Xem lịch sử tin nhắn” .....	74
4.2.4.6. Sequence diagram cho use-case “Gửi tin nhắn thông thường”.....	74
4.2.4.7. Sequence diagram cho use-case “Gửi tin nhắn file/hình ảnh” .....	75
4.2.4.8. Sequence diagram cho use-case “Gửi tin nhắn thoại” .....	76
4.2.4.9. Sequence diagram cho use-case “Đăng status hình ảnh” .....	77
4.2.4.10. Sequence diagram cho use-case “Đăng status”.....	79
4.2.4.11. Sequence diagram cho use-case “Đăng voice status” .....	80
4.2.4.12. Sequence diagram cho use-case “Like status” .....	82
4.2.4.13. Sequence diagram cho use-case “Comment trên status” .....	83
4.2.4.14. Sequence diagram cho use-case “Duyệt news feed”.....	84
4.3. Thiết kế dữ liệu.....	86
4.3.1. Sơ đồ CSDL .....	86
4.3.2. Mô tả chi tiết các bảng.....	87
4.4. Kiến trúc hệ thống .....	92
4.4.1. Phân tích thiết kế kiến trúc.....	92
4.4.2. Kiến trúc hệ thống.....	94

4.4.3. Kiến trúc của Chat Server Cluster và Realtime Server Cluster.....	95
4.5. Giao diện phần mềm.....	99
4.5.1. Màn hình “Sign Up”.....	99
4.5.2. Màn hình “Sign In”.....	100
4.5.3. Màn hình “News Feed” .....	101
4.5.4. Màn hình “Messages” .....	102
4.5.5. Màn hình “Friend Requests” .....	103
4.5.6. Màn hình “Notifications” .....	104
4.5.7. Màn hình “Search” .....	105
4.5.8. Màn hình “Sharing” .....	106
4.5.9. Màn hình “Profile” .....	107
4.5.10. Màn hình “Info” .....	108
4.5.11. Màn hình “Comments” .....	109
4.5.12. Màn hình “Friends” .....	110
4.5.13. Màn hình “World Map” .....	111
4.5.14. Màn hình tương tác giọng nói .....	112
4.5.15. Màn hình “Recommend” .....	113
Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	114
5.1. Kết luận .....	114
5.2. Hướng phát triển.....	115
TÀI LIỆU THAM KHẢO .....	116

## **DANH MỤC HÌNH VẼ**

Hình 1.1 Tính năng chia sẻ giọng nói trên Zalo, Viber, Messenger .....	4
Hình 2.1 Windows Core.....	8
Hình 2.2 Các thiết bị chạy cùng một Core Windows.....	9
Hình 2.3 Tổng quan hệ điều hành Windows.....	9
Hình 2.4 Windows 10 .....	10
Hình 2.5 One Universal Windows Platform .....	11
Hình 2.6 Speech SDK features for dev .....	12
Hình 2.7 Kiến trúc deployment của Couchbase Server .....	16
Hình 2.8 Cơ chế ánh xạ id của document sang phân vùng, và server chứa nó.....	17
Hình 2.9 Luồng đi của dữ liệu trong Couchbase Server đối với một tác vụ ghi .....	19
Hình 2.10 <i>Design document</i> và <i>view</i> trong Couchbase Server .....	20
Hình 2.11 Việc thực hiện index và truy vấn được phân tán ra tất cả các node liên quan trong Couchbase Server.....	21
Hình 2.12 File index cấu trúc cây-b trong Couchbase Server .....	22
Hình 2.13 Định nghĩa index của view bằng thuộc tính name .....	22
Hình 2.14 Kiến trúc của Netty .....	27
Hình 3.1 Kiến trúc thiết kế ban đầu của hệ thống server xử lý .....	32
Hình 3.2 Kiến trúc cải tiến với sự phân tán của cụm server xử lý .....	34
Hình 3.3 Kiến trúc phân tán tất cả các thành phần trong hệ thống server xử lý .....	36
Hình 4.1 Mô hình use-case của ứng dụng .....	47
Hình 4.2 Sequence diagram “Xem danh sách kết nối” .....	68
Hình 4.3 Sequence diagram “Retrieve connections” được sử dụng bởi sequence diagram “Xem danh sách kết nối” .....	69
Hình 4.4 Sequence diagram “Gửi yêu cầu tạo kết nối” .....	70
Hình 4.5 Sequence diagram “Create Connection Request” được sử dụng bởi sequence diagram “Gửi yêu cầu tạo kết nối” .....	70
Hình 4.6 Sequence diagram “Chấp nhận/hủy yêu cầu tạo kết nối” .....	71

Hình 4.7 Sequence diagram “Update ConnectionRequest IsApproved Status” được sử dụng bởi sequence diagram “Chấp nhận/hủy yêu cầu tạo kết nối” .....	72
Hình 4.8 Sequence diagram “Xem danh sách các cuộc hội thoại” .....	73
Hình 4.9 Sequence diagram “Retrieve connections” được sử dụng bởi sequence diagram “Xem danh sách các cuộc hội thoại” .....	73
Hình 4.10 Sequence diagram “Xem lịch sử tin nhắn” .....	74
Hình 4.11 Sequence diagram “Gửi tin nhắn thông thường” .....	74
Hình 4.12 Sequence diagram gửi file/hình ảnh.....	75
Hình 4.13 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Gửi tin nhắn file/hình ảnh”.....	75
Hình 4.14 Sequence diagram “Gửi tin nhắn thoại”.....	76
Hình 4.15 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Gửi tin nhắn thoại” .....	76
Hình 4.16 Sequence diagram “Đăng status hình ảnh” .....	77
Hình 4.17 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Đăng status hình ảnh” .....	78
Hình 4.18 Sequence diagram “Post image status” được sử dụng bởi sequence diagram “Đăng status hình ảnh” .....	78
Hình 4.19 Sequence diagram “Đăng status” .....	79
Hình 4.20 Sequence diagram “Create status update” được sử dụng bởi sequence diagram “Đăng status” .....	79
Hình 4.21 Sequence diagram “Đăng voice status” .....	80
Hình 4.22 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Đăng voice status” .....	81
Hình 4.23 Sequence diagram “Post voice status” được sử dụng bởi sequence diagram “Đăng voice status”.....	81
Hình 4.24 Sequence diagram “like status”.....	82
Hình 4.25 Sequence diagram “Add additional liked-user-id of status-update” được sử dụng bởi sequence diagram “like status” .....	83

Hình 4.26 Sequence diagram “Comment trên status”.....	83
Hình 4.27 Sequence diagram “Post comment” được sử dụng bởi sequence diagram “Comment trên status” .....	84
Hình 4.28 Sequence diagram “Duyệt news feed” .....	84
Hình 4.29 Sequence diagram “Get News Feed” được sử dụng bởi sequence diagram “Duyệt news feed” .....	85
Hình 4.30 Sơ đồ CSDL của hệ thống.....	86
Hình 4.31 Sơ đồ kiến trúc hệ thống của hệ thống mạng xã hội .....	94
Hình 4.32 Mô hình kiến trúc phân tán của Chat Server và Realtime Server.....	96
Hình 4.33 Giao diện màn hình “Sign Up” .....	99
Hình 4.34 Giao diện màn hình “Sign In” .....	100
Hình 4.35 Giao diện màn hình “News Feed”.....	101
Hình 4.36 Giao diện màn hình “Message” .....	102
Hình 4.37 Giao diện màn hình “Friend Requests”.....	103
Hình 4.38 Giao diện màn hình “Notifications” .....	104
Hình 4.39 Giao diện màn hình “Search”.....	105
Hình 4.40 Giao diện màn hình “Sharing” .....	106
Hình 4.41 Giao diện màn hình “Profile”.....	107
Hình 4.42 Giao diện màn hình “Info” .....	108
Hình 4.43 Giao diện màn hình “Comments” .....	109
Hình 4.44 Giao diện màn hình “Friends” .....	110
Hình 4.45 Giao diện màn hình “World Map” .....	111
Hình 4.46 Giao diện chức năng tương tác giọng nói .....	112
Hình 4.47 Giao diện chức năng đề xuất gợi ý.....	113

## **DANH MỤC BẢNG**

Bảng 4.1 Danh sách các yêu cầu .....	46
Bảng 4.2 Đặc tả use-case “Xem danh sách kết nối” .....	48
Bảng 4.3 Đặc tả use-case “Gửi yêu cầu tạo kết nối” .....	49
Bảng 4.4 Đặc tả use-case “Chấp nhận/hủy yêu cầu tạo kết nối” .....	50
Bảng 4.5 Đặc tả use-case “Xem danh sách các cuộc hội thoại” .....	51
Bảng 4.6 Đặc tả use-case “Xem lịch sử tin nhắn” .....	52
Bảng 4.7 Đặc tả use-case “Gửi tin nhắn thông thường” .....	53
Bảng 4.8 Đặc tả use-case “Gửi file/hình ảnh” .....	55
Bảng 4.9 Đặc tả use-case “Gửi tin nhắn thoại” .....	56
Bảng 4.10 Đặc tả use-case “Đăng status hình ảnh” .....	57
Bảng 4.11 Đặc tả use-case “Đăng status” .....	58
Bảng 4.12 Đặc tả use-case “Đăng voice status” .....	59
Bảng 4.13 Đặc tả use-case “Like status” .....	59
Bảng 4.14 Đặc tả use-case “Comment trên status” .....	60
Bảng 4.15 Đặc tả use-case “Duyệt news feed” .....	61
Bảng 4.16 Đặc tả use-case “Xem những gì xảy ra xung quanh mình” .....	62
Bảng 4.17 Đặc tả use-case “Đăng status bằng voice command” .....	63
Bảng 4.18 Đặc tả use-case “Gửi tin nhắn bằng voice command” .....	64
Bảng 4.19 Đặc tả use-case “Chuyển sang tình trạng nguy hiểm & nhờ sự trợ giúp bằng voice command” .....	65
Bảng 4.20 Đặc tả use-case “Thay đổi trạng thái bằng voice command” .....	66
Bảng 4.21 Đặc tả use-case “Truy vấn thông tin bạn bè, người thân bằng voice command” .....	67
Bảng 4.22 Đặc tả use-case “Nhờ sự trợ giúp” .....	68
Bảng 4.23 Bảng mô tả các bảng trong CSDL .....	87
Bảng 4.24 Bảng User .....	88
Bảng 4.25 Bảng session .....	88
Bảng 4.26 Bảng Update .....	89

Bảng 4.27 Bảng UpdateLike .....	89
Bảng 4.28 Bảng UpdateComment.....	90
Bảng 4.29 Bảng RealTimeInfo .....	90
Bảng 4.30 Bảng Connection .....	90
Bảng 4.31 Bảng ConnectionRequest .....	91
Bảng 4.32 Bảng Conversation.....	92

## **DANH MỤC TỪ VIẾT TẮT**

<b>STT</b>	<b>Từ viết tắt, thuật ngữ tiếng Anh</b>	<b>Nội dung</b>
1	CRUD	Create-read-update-delete (Thao tác tạo-đọc-cập nhật-xóa đối với CSDL)
2	CSDL	Cơ sở dữ liệu
3	MVVM	Mô hình Model-View-ViewModel
4	NoSQL	Not Only Structured Query Language
5	UWP	Universal Windows Platform

## MỞ ĐẦU

Ngày nay, mạng xã hội đã trở thành một món ăn tinh thần không thể thiếu trong cuộc sống của gần hai tỷ người trên toàn thế giới<sup>[6]</sup>. Sự bùng nổ của mạng xã hội trong những năm gần đây đã thay đổi đáng kể cách sống và làm việc của nhiều người. Mạng xã hội là công cụ rất hữu ích, tuy nhiên nếu không biết cách sử dụng điều độ thì chúng có thể gây ra nhiều tác hại tiêu cực đối với người dùng.

Thực tế có nhiều mạng xã hội đáp ứng nhiều mục đích khác nhau, nhưng hầu hết chỉ mới dừng lại ở việc tương tác bằng tay, gây mất nhiều thời gian của người dùng.

Với những lý do trên, nhóm thực hiện đề tài với mong muốn áp dụng công nghệ nhận dạng giọng nói vào mạng xã hội, để tăng sự tương tác của người dùng, bên cạnh đó cũng giúp giảm thiểu tối đa thời gian sử dụng mạng xã hội của họ.

Ngoài ra, mạng xã hội là một trong những hệ thống có yêu cầu và đòi hỏi cao về khả năng đáp ứng nhiều người dùng tại cùng một thời điểm. Do đó, mạng xã hội rất thích hợp cho việc nghiên cứu, học hỏi, và thiết kế kiến trúc phân tán cho một hệ thống lớn.

Đề tài tương đối rộng và được thực hiện trong khoảng thời gian tương đối ngắn nên sẽ không tránh khỏi một số sai sót nào đó, rất mong quý thầy cô, các bạn thông cảm và đóng góp ý kiến để đề tài có thể hoàn chỉnh hơn.

Chúng em xin chân thành cảm ơn!

**TP. Hồ Chí Minh, tháng 12 năm 2015**

**Nhóm sinh viên thực hiện**

**Hoàng Tiến Đạt – Trần Diệu Nhất Hạnh**

## **Chương 1. GIỚI THIỆU**

### **1.1. Lý do chọn đề tài**

Hiện nay có rất nhiều loại mạng xã hội khác nhau, số người sử dụng mạng xã hội cũng ngày càng tăng cao. Nếu sử dụng đúng cách thì mạng xã hội là một công cụ rất hữu ích, nó có thể tích hợp tất cả mọi thứ mà người dùng cần. Một nơi có thể chia sẻ tâm sự, lưu lại những hình ảnh đáng nhớ, và có thể cập nhật chia sẻ thông tin với bạn bè, người thân cùng nhiều lợi ích khác. Tuy nhiên, việc chỉ ở một chỗ và sử dụng mạng xã hội không mục đích có thể gây ra nhiều ảnh hưởng tiêu cực không lường trước, điển hình như giảm bớt sự tương tác thực tế giữa người với người, các mối quan hệ có thể bị rạn nứt khi người dùng coi trọng bạn bè ảo hơn những gì ở trước mắt mà lại dành quá ít thời gian cho những người thật việc thật quanh họ. Tiếp đó nó sẽ khiến người dùng xao lãng mục tiêu cá nhân, thay vì học hỏi những kỹ năng cần thiết phát triển tương lai thì lại chỉ chăm chú để trở thành những anh hùng bàn phím. Ngoài ra nó có thể tăng nguy cơ trầm cảm, giết chết sự sáng tạo và đặc biệt gây mất thời gian của rất nhiều người.

Nhận diện giọng nói là một công nghệ, hay có thể nói là một khái niệm đang dần trở nên quen thuộc với tất cả chúng ta hiện nay. Việc áp dụng giọng nói cung cấp thêm một cách tương tác mới giúp chúng ta tiết kiệm được thời gian thực hiện các tương tác so với khi thao tác bằng tay (bởi bàn phím, chuột, màn hình cảm ứng...). Khi đó, chúng ta ra lệnh cho thiết bị thực hiện một tác vụ nào đó, mà không cần thiết phải qua nhiều bước rườm rà.

Ví dụ cụ thể tại Facebook, mạng xã hội được nhiều người sử dụng nhất tại thời điểm hiện tại, trong quá trình truy cập, người dùng đang muốn xem trạng thái của một người nào đó, nhưng khi đang thao tác bằng tay người dùng có thể dễ dàng bị cuốn theo những thông tin, trạng thái khác trên đó mà quên đi mất mục đích ban đầu của họ. Dần dần khiến họ nghiện, xao lãng và mất nhiều thời gian hơn vì nó. Nhưng khi áp dụng tương tác bằng giọng nói, nếu người dùng muốn xem trạng thái của một người nào đó, họ chỉ cần ra lệnh, yêu cầu theo ý mình thì ứng dụng sẽ trả về thông

tin họ muốn biết, giúp người dùng sử dụng có mục đích cụ thể và tiết kiệm thời gian tối đa.

Nhóm chọn đề tài “**Nghiên cứu Couchbase Server để xây dựng mạng xã hội**” nhằm phát triển một mạng xã hội trên nền tảng công nghệ di động Windows 10 Mobile, nhằm nghiên cứu các công nghệ (diễn hình là Couchbase Server), kỹ thuật, kiến trúc phân tán để xây dựng hệ thống lớn, ngoài ra mạng xã hội mà nhóm xây dựng còn giúp tăng tính tương tác của người dùng với thiết bị, giúp đáp ứng nhu cầu của người dùng nhanh hơn, tiết kiệm thời gian hơn thông qua việc tương tác bằng giọng nói. Bên cạnh đó cũng sẽ khắc phục nhiều hạn chế của mạng xã hội hiện tại.

Ngoài ra, nhóm cũng muốn thử sức mình thiết kế và xây dựng một hệ thống có thể đáp ứng được một lượng lớn người dùng. Với đặc trưng của mạng xã hội, người dùng có thể dùng mọi lúc mọi nơi, tương tác liên tục với nhau, đòi hỏi hệ thống mạng xã hội phải đáp ứng được một lượng lớn người dùng sử dụng tại một thời điểm. Do đó, mạng xã hội là một hệ thống điển hình và phù hợp để nghiên cứu, áp dụng các kỹ thuật, mô hình, kiến trúc phân tán, nhằm xây dựng hệ thống có thể đáp ứng lượng lớn người dùng tại một thời điểm.

## 1.2. Tình hình nghiên cứu

### *Hiện trạng các mạng xã hội, các công trình nghiên cứu liên quan đến đề tài:*

- Về tình hình các mạng xã hội, có rất nhiều loại hình mạng xã hội với các mục đích khác nhau: Social News (Digg, Sphim, Newsvine,...) cho phép chúng ta có thể đọc tin tức từ các topic sau đó có thể vote hoặc comment; Social Sharing (Flickr, Snapfish, Youtube,...) cho phép chúng ta có thể tạo, chia sẻ các hình ảnh, video với tất cả mọi người; Social Networks (Facebook, MySpace, Twitter,...) cho phép chúng ta có thể tìm kiếm bạn bè dựa trên thông tin cá nhân, sở thích hay lĩnh vực quan tâm, từ đó kết nối, chia sẻ với nhau mà không phân biệt không gian thời gian, Social Bookmarking (Delicious, Faves, StumbleUpon, BlogMarks, Diigo,...) cho phép chúng ta có thể chia sẻ hoặc bookmark các site quan tâm.

Trong phạm vi khóa luận, nhóm chủ yếu tập trung nghiên cứu về loại hình Social Networks trên thiết bị di động

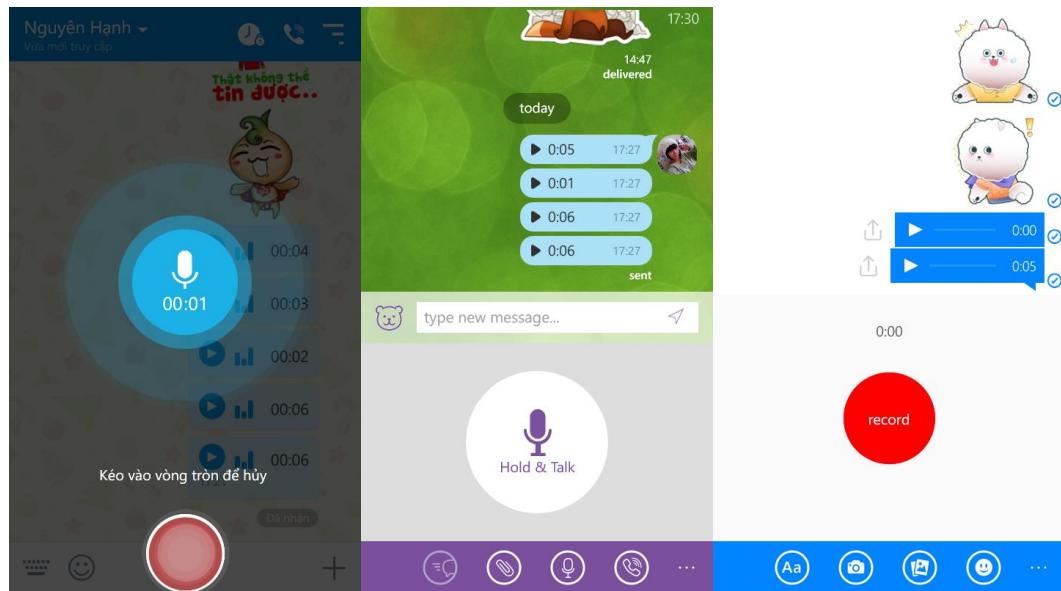
- Về công nghệ nhận dạng giọng nói, đã có nhiều nghiên cứu triển khai hoặc giới thiệu thành công những ứng dụng giọng nói.

Ask.com là một dịch vụ cho phép người dùng hỏi và nhận câu trả lời, đã tích hợp công nghệ nhận dạng giọng nói của Nuance phát triển vào ứng dụng iOS và Android. Sự liên kết này cho phép người dùng hỏi, trả lời, cũng như đăng tải các bình luận.

Amazon cũng cập nhật app Kindle trên iOS để hỗ trợ tính năng VoiceOver trong iOS. VoiceOver sẽ tự động đọc nội dung trên màn hình để giúp cho việc xem sách của những người bị khiếm thị được dễ dàng và thuận tiện hơn.

Siri, Google Now, Cortana là những ứng dụng được phát triển bởi các tập đoàn công nghệ lớn với tiềm lực mạnh mẽ. Bộ ba phần mềm này giúp người dùng thiết bị di động có thể tương tác với thiết bị của mình một cách thông minh hơn, từ việc đặt câu hỏi, nhận câu trả lời cho đến điều chỉnh các thông số máy và khởi chạy ứng dụng. Mọi thao tác đều được thực hiện một cách tự động.

Facebook, Messenger, Zalo, Viber, WhatsApp cũng đã có tính năng chia sẻ giọng nói, tích hợp tính năng chat giọng nói vào sản phẩm của mình.



Hình 1.1 Tính năng chia sẻ giọng nói trên Zalo, Viber, Messenger

Tuy nhiên, đa số mạng xã hội chỉ mới dừng lại ở việc sử dụng giọng nói để gửi tập tin âm thanh, mà chưa có phần phiên dịch từ giọng nói sang văn bản. Việc gõ từ bàn phím trở nên quá lâu khi cần nói những nội dung dài, vậy tại sao không nói cho nhanh? Đây là suy nghĩ của hầu hết người dùng sử dụng các loại thiết bị di động.

Một khảo sát của Forrester đã chỉ ra sự gia tăng của ứng dụng điều khiển bằng giọng nói. Một lượng lớn người dùng xài công nghệ này để gửi tin nhắn, 46% dùng cho việc tìm kiếm, 40% dùng giọng nói để tìm đường đi và 38% dùng để ghi chú. Đó là những con số khá lớn tính năng 1168 người tham gia cuộc nghiên cứu [10].

### **Đánh giá những công trình, hệ thống đã công bố:**

#### *Ưu điểm:*

- Các mạng xã hội hiện tại đa số đều đáp ứng nhu cầu của người sử dụng về mặt ý nghĩa thông tin liên lạc.
- Có tích hợp phiên dịch phần cập nhật trạng thái hay tin nhắn sang ngôn ngữ người dùng mong muốn ngay cả khi mạng xã hội chỉ sử dụng một ngôn ngữ như tiếng Anh, giúp mọi người có thể hiểu và giao tiếp trao đổi với nhau tốt hơn.
- Sự phát triển và áp dụng công nghệ nhận dạng giọng nói giúp biến những chiếc smartphone khô khan thành người bạn có thể trò chuyện bất kỳ lúc nào.

#### *Nhược điểm:*

- Đa số các mạng xã hội đều gây ảo hóa cho người dùng, giảm thiểu khả năng tương tác thực tế giữa các người dùng với nhau, khiến người dùng ngày càng thụ động.
- Chưa đáp ứng về mặt tiện lợi tiết kiệm thời gian.
- Việc áp dụng tính năng chia sẻ giọng nói trên các ứng dụng di động có giới hạn thời gian một lần ghi âm tin nhắn, và người dùng phải luôn giữ phím ghi âm mới có thể ghi âm tin nhắn được.
- Công nghệ nhận diện giọng nói tuy đã phát triển và cải tiến không ngừng nghỉ nhưng không phải lúc nào máy cũng nhận diện đúng giọng của người dùng.

### **1.3. Đối tượng và phạm vi nghiên cứu**

#### **1.3.1. Đối tượng nghiên cứu**

Mạng xã hội.

#### **1.3.2. Phạm vi nghiên cứu**

*Trong phạm vi thực hiện đề tài, khóa luận này sẽ bao gồm:*

- Phát triển mạng xã hội tích hợp tương tác thông qua giọng nói trên thiết bị di động, hệ điều hành Windows Phone, cụ thể là Windows 10 Mobile.
- Nghiên cứu các kỹ thuật, mô hình kiến trúc để xây dựng một hệ thống có thể đáp ứng lượng lớn người dùng, có hiệu suất cao và độ trễ thấp.
- Nghiên cứu và ứng dụng nhận dạng giọng để xây dựng các tính năng tăng khả năng tương tác của người dùng với ứng dụng mạng xã hội thông qua các lệnh bằng giọng nói của người dùng.
- Do giới hạn về mặt thời gian, nhóm sẽ tập trung nghiên cứu và sử dụng các thư viện nhận dạng giọng nói có sẵn, thay vì tự phát triển bộ thư viện nhận dạng giọng nói để sử dụng trong ứng dụng. Nhóm sẽ giới hạn ngôn ngữ được hỗ trợ, chỉ hỗ trợ nhận dạng bằng tiếng Anh.

### **1.4. Mục tiêu đề tài**

#### **1.4.1. Mục tiêu tổng quát**

- Xây dựng một hệ thống và ứng dụng có đầy đủ các tính năng cơ bản của các mạng xã hội thông thường.
- Xây dựng một hệ thống phân tán có khả năng mở rộng để đáp ứng lượng lớn người dùng.
- Xây dựng một ứng dụng mạng xã hội có hỗ trợ tính năng tương tác bằng giọng nói.
- Nghiên cứu các công nghệ, kỹ thuật, và kiến trúc phân tán để áp dụng vào việc xây dựng hệ thống.

### **1.4.2. Mục tiêu cụ thể**

- Tìm hiểu các công nghệ, kỹ thuật để xây dựng hệ thống hiệu suất cao, độ trễ thấp.
- Nghiên cứu CSDL Couchbase Server để ứng dụng trong thiết kế hệ thống phân tán của mạng xã hội.
- Nghiên cứu các mô hình kiến trúc phân tán, kiến trúc hướng-dịch-vụ, mô hình xử lý hướng sự kiện bất đồng bộ.
- Nghiên cứu ứng dụng các thư viện nhận dạng giọng nói trên C# và Windows 10 Mobile.
- Nghiên cứu các bài viết về “Mạng xã hội thế hệ tiếp theo”, “Giao diện tương tác thế hệ tiếp theo”.

### **1.5. Phạm vi công nghệ**

Về phía client, nhóm sẽ sử dụng công nghệ Windows UWP. Về phía server, nhóm sẽ sử dụng các công nghệ của Java để cài đặt. Có 2 loại CSDL được sử dụng là CSDL NoSQL phân tán Couchbase Server được dùng để lưu trữ dữ liệu trên mạng xã hội, và Redis được sử dụng để cache và lưu trữ lịch sử tin nhắn, thời điểm online, cũng như các dữ liệu thời gian thực của người dùng (như vị trí của người dùng, vị trí SOS).

### **1.6. Cấu trúc khóa luận**

Cấu trúc của đề tài bao gồm:

Chương 1: Giới thiệu đề tài.

Chương 2: Công nghệ được sử dụng để xây dựng đề tài.

Chương 3: Các vấn đề gấp phải và hướng giải quyết.

Chương 4: Phân tích, thiết kế.

Chương 5: Kết luận và hướng phát triển.

## **Chương 2. CÔNG NGHỆ ĐƯỢC SỬ DỤNG ĐỂ XÂY DỰNG ĐỀ TÀI**

### **2.1. Nền tảng phát triển ứng dụng di động Windows 10 UWP**

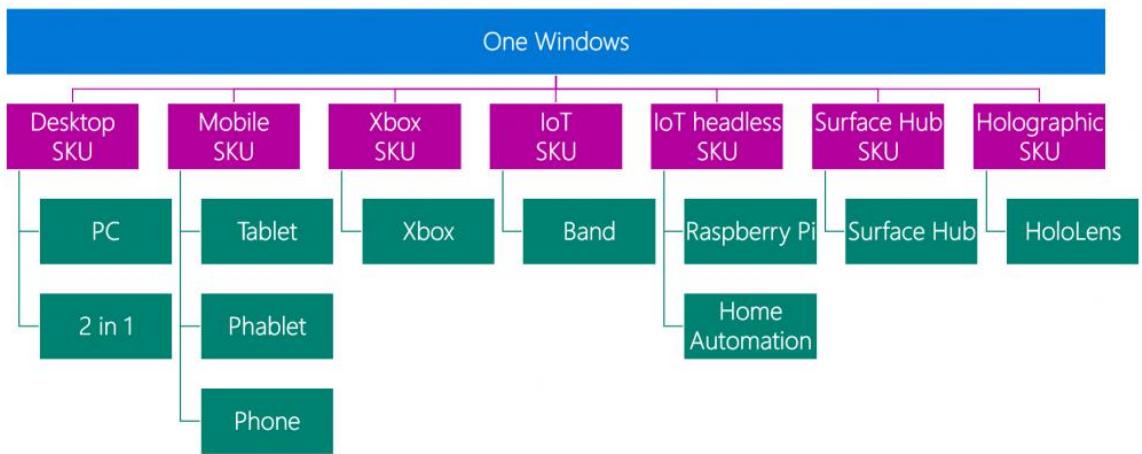
#### **2.1.1. Windows Core**



Hình 2.1 Windows Core

Windows Core là cấu trúc hạ tầng được xây dựng phát triển từ khi Windows ra đời và tới nay trên tất cả các thiết bị từ thiết bị IoT, Desktop, Phone, tới Xbox, cả thiết bị lớn như Surface Hub và kính thực tế ảo HoloLens đều cài đặt một phiên bản Windows. Tính chung khoảng 10 loại thiết bị cài đặt Windows Core, tuy nhiên với Windows Core thì chưa đủ để thực hiện các chức năng của chúng, chưa nói đến việc tạo “thực tế ảo”. Nói cách khác Windows Core là nơi kết nối các thành phần phần cứng, phần mềm và tạo môi trường triển khai hiệu quả để có thể khai thác tất cả các tính năng của thiết bị.

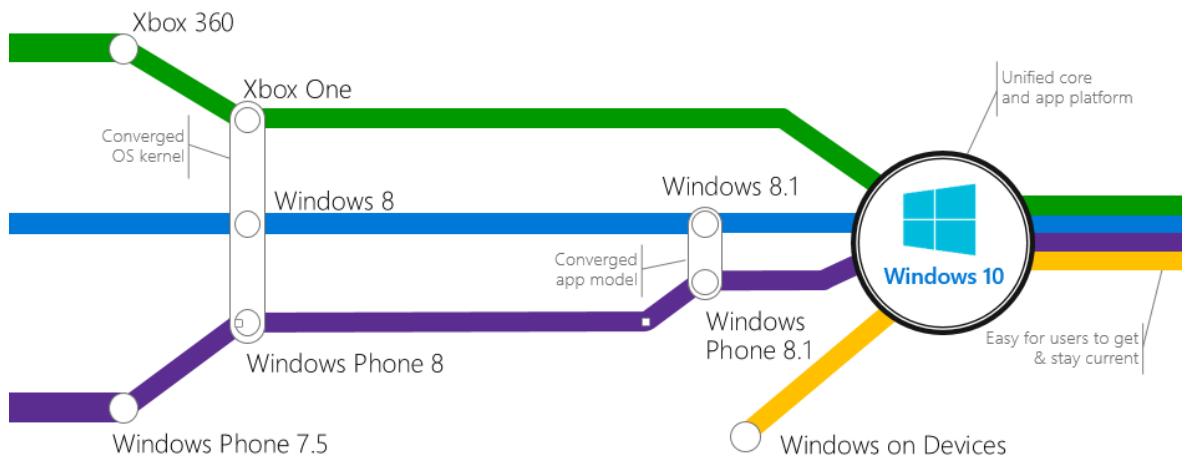
Đối với nhà phát triển thì Windows Core mang đến một lợi ích rất lớn đó là tất cả các thiết bị điều chạy cùng một hệ điều hành nên việc phát triển, lập trình chúng là hoàn toàn giống nhau.



Hình 2.2 Các thiết bị chạy cùng một Core Windows

Qua đó ta thấy được sự kế thừa hạ tầng Windows trên hầu hết tất cả các thiết bị công nghệ, mỗi loại lại được xây dựng theo các nhóm tính năng riêng. Ví dụ khi nói về holographic thì Microsoft đã tạo ra một phiên bản Windows holographic và cài đặt lên Hololens, chứ không phải xây dựng một phiên bản Windows cho Hololens. Tuy bây giờ chỉ có mỗi Hololens chạy công nghệ này.

Nhưng nhìn từ hướng nhà phát triển thì khi xây dựng một ứng dụng Windows 10 thì chúng ta thường hướng tới một loại thiết bị cụ thể trong số chúng nhiều hơn. Đó là cách mà Windows tạo nên sự chuyên biệt hóa về trải nghiệm trong mỗi sản phẩm.



Hình 2.3 Tổng quan hệ điều hành Windows

Nói về sự đồng nhất về hệ điều hành Windows ta có sơ đồ minh họa như trên, năm 2012 Microsoft cho ra đời Windows Phone 7, nó là một hệ điều hành hoàn toàn khác và độc lập, cùng lúc đó chúng ta cũng có XBox 360 và nó lại chạy trên một hệ điều hành khác nữa. Nhưng rồi Windows 8, Xbox One, Windows Phone 8 ra đời thì chúng ta đã thấy bắt đầu có sự đồng nhất khi chúng đều sử dụng nhân Windows NT. Tiếp theo là Windows 8.1 ra đời hứa hẹn đầu tiên trong việc đồng nhất app model, tạo một môi trường tương đồng (không giống nhau hoàn toàn) khi phát triển ứng dụng trên cả Phone và Windows. Và cuối cùng Windows 10 đột ngột xuất hiện mở ra sự đồng nhất hoàn toàn, không chỉ ở lớp hệ điều hành mà đồng nhất cả về nền tảng phát triển.



Hình 2.4 Windows 10

### 2.1.2. Universal Windows Platform

Sự xuất hiện của Windows 10 đánh dấu bước tiến mới của Microsoft trong nỗ lực phát triển hệ sinh thái Windows của mình. Điểm khác biệt cốt lõi của Windows 10 so với Windows 8/8.1 và Windows Phone nằm ở khả năng tương thích với các ứng dụng trên nền tảng Universal Windows Platform với hệ thống API mạnh và hiệu quả hơn, giúp ứng dụng chạy với hiệu năng cao, mang lại trải nghiệm tốt nhất cho người dùng.



Hình 2.5 One Universal Windows Platform

Universal App không phải là một khái niệm mới, nó đã được Microsoft giới thiệu trên Windows và Windows Phone 8.1. Tuy vậy UWP vẫn có nhiều điểm khác biệt so với các ứng dụng Universal trên các phiên bản hệ điều hành trước. Nếu như trên Windows và Windows Phone 8.1, ứng dụng Universal cần nhiều giao diện được thiết kế riêng để phù hợp với màn hình của từng thiết bị thì với UWP, giao diện được tự động điều chỉnh theo độ phân giải của màn hình. Điều này có nghĩa công việc của các nhà phát triển ứng dụng được giảm bớt khỏi lượng công việc phải làm, đồng thời giao diện app cũng trở nên đẹp mắt và thân thiện hơn. Universal Windows Platform kế thừa nền tảng WinRT của Windows 8/8.1, đồng thời bổ sung những API mới của Windows 10. Vì vậy ứng dụng UWP sẽ có hiệu năng cao đáng kể và mang lại những trải nghiệm tốt nhất cho người dùng.

### 2.1.3. Windows 10 Mobile

Không chỉ đem tới một hệ sinh thái đồng nhất, Windows 10 trên di động còn tích hợp khá nhiều tính năng đáng giá nhằm nâng cao trải nghiệm người dùng.

Đặc biệt là sự phát triển của trợ lý ảo Cortana, hỗ trợ tìm kiếm bằng giọng nói và thực thi tác vụ được yêu cầu. Trợ lý ảo này có khả năng ghi nhớ và học thói quen của người dùng để thực thi mệnh lệnh nhanh chóng hơn về sau.

Ngoài ra, chúng ta còn có thể lập trình tạo ra các ứng dụng tích hợp tương tác điều khiển trực tiếp với Cortana.

Speech SDK Features for Devs	Windows Phone	Windows Store	iOS (iPhone/iPad)	Android (Phone/Tablet)
 Built-in Personal Assistant with Speech	Cortana*	N/A	Siri*	Google Now Launcher*
 Personal Assistant Extensions in third-party apps using Voice Commands	Yes	N/A	No (first-party apps only)	No (first-party apps only)
 Speech Synthesis SDK for Devs (Text-to-Speech)	Yes	Yes	Yes	Yes
 In-app Speech Recognition SDK for Devs	Yes	Yes*	No*	Yes

Hình 2.6 Speech SDK features for dev

## 2.2. Công nghệ nhận dạng giọng nói

### 2.2.1. Cách thức xây dựng dịch vụ nhận dạng, điều khiển bằng giọng nói

Thông thường một bộ máy giọng nói sẽ có hai phần. Phần thứ nhất gọi là speech synthesizer (còn gọi là Text to Speech hay TTS). Đây là một trình tổng hợp giọng nói mà thiết bị hoặc ứng dụng xài để tương tác với người dùng, ví dụ như đọc văn bản trên màn hình, thông báo về tiến độ chạy của một tác vụ nào đó.

Phần thứ hai là một công nghệ nhận dạng cho phép ứng dụng biết được người dùng nói gì, từ đó chuyển thể thành lệnh để thiết bị thực thi hoặc chuyển đổi thành các ký tự nhập liệu. Nói cách khác, đây là thứ thay thế cho bàn phím của chúng ta.

Một ứng dụng nhận dạng giọng nói lý tưởng sẽ bao gồm cả hai bộ phận nói trên. Siri, Google Now, Cortana là ví dụ của những phần mềm tương tác giọng nói lý tưởng. Còn Facebook, Messenger, Zalo, WhatsApp là các phần mềm chỉ sử dụng giọng nói cho chiều nhập liệu, không có chiều phản hồi.

### 2.2.2. Mô hình triển khai công nghệ giọng nói

Có nhiều cách thức mà các công ty hiện nay đang triển khai voice technology, có thể kể đến 2 phương pháp phổ biến như sau:

**Điện toán đám mây:** Trong trường hợp này, việc nhận dạng, xử lý ngôn ngữ sẽ diễn ra trên máy chủ của các công ty cung cấp dịch vụ. Phương pháp đám mây giúp việc nhận dạng được chính xác hơn, ứng dụng thì có dung lượng nhỏ, nhưng bù lại thì thiết bị ở phía người dùng phải luôn kết nối với Internet. Độ trễ trong quá trình gửi giọng nói từ máy lên server rồi trả kết quả từ server về lại máy cũng là những thứ đáng cân nhắc. Siri, Google Now, Cortana hiện đang xài cách này.

**Tích hợp thẳng vào app:** Với phương thức này, quá trình xử lý giọng nói sẽ diễn ra trong nội bộ ứng dụng, không cần giao tiếp với bên ngoài, chính vì thế tốc độ sẽ nhanh hơn. Người dùng cũng không bắt buộc phải kết nối vào mạng thường trực. Tuy nhiên, giải pháp này gặp nhược điểm đó là khi có cập nhật hoặc thay đổi gì đó về bộ máy nhận dạng, nhà sản xuất sẽ phải cập nhật lại cả một app, trong khi với phương thức đám mây thì những thay đổi đó chỉ cần làm ở phía server. Kích thước ứng dụng cũng sẽ tăng lên, có thể lên tới cả vài trăm MB. Hiện có Nuance và một vài app nhỏ là xài phương pháp tích hợp. Apple, Google cũng có bổ sung tùy chọn offline cho một số ngôn ngữ nhất định dùng trong việc chuyển văn bản thành chữ viết.

### 2.2.3. Speech platform trên Windows 10 UWP

#### 2.2.3.1. Speech Synthesis

Cung cấp cho các ứng dụng khả năng có thể nói, phát âm, tương tác với người dùng qua việc sử dụng thư viện Windows.Media.SpeechSynthesis.

Các SpeechSynthesis API hỗ trợ khởi tạo và cấu hình một bộ tổng hợp giọng nói để chuyển đổi một chuỗi văn bản thành âm thanh. Đặc điểm giọng nói, phát âm, cường độ, tốc độ, điểm nhấn trọng âm có thể tùy chỉnh thông qua Speech Synthesis Markup Language (SSML).

#### 2.2.3.2. Speech Recognition

Cung cấp cho các ứng dụng khả năng có thể hiểu người dùng nói gì thông qua việc sử dụng thư viện Windows.Media.SpeechRecognition.

Các SpeechRecognition API hỗ trợ cho phép ứng dụng nhận dạng khi người dùng đang nói, có thể sử dụng hoặc không sử dụng giao diện. Đặc biệt, nó sẽ tự động phát hiện khi nào người dùng ngừng nói và trả về một chuỗi ký tự cho ứng dụng.

#### **2.2.3.3. Cortana Interaction**

Sử dụng thư viện Windows.ApplicationModel.VoiceCommands và Windows.ApplicationModel.AppService để có thể phát triển các ứng dụng tích hợp tương tác điều khiển thông qua giọng nói từ Cortana mà không phải trực tiếp khởi chạy ứng dụng.

### **2.3. Jersey Restful Framework**

#### **2.3.1. Giới thiệu**

REST là chuẩn web service chung, thống nhất và được sử dụng rộng rãi nhất trong việc xây dựng và cài đặt các hệ thống server mà có thể đáp ứng được các tương tác từ client.

Jersey là một framework được xây dựng nhằm giúp các nhà phát triển có thể dễ dàng xây dựng được một web service tuân thủ theo chuẩn REST và JAX-RS một cách đơn giản và dễ dàng, nhằm giúp nhà phát triển có thể tập trung hơn vào xử lý business logic, ngoài ra còn giúp họ có thể tạo ra một web service có cấu trúc tốt hơn, tuân thủ theo đúng chuẩn REST.

Jersey là một RESTful web service mã nguồn mở, được phát triển bởi cộng đồng Java, và đủ độ ổn định để có thể đưa vào môi trường production.

#### **2.3.2. Lý do lựa chọn**

Jersey là một Micro Framework, nó chỉ chứa những thành phần cần thiết, và thiết yếu nhất để xây dựng nên một Restful Webservice. Vì Jersey chỉ chứa những thành phần cần thiết và thiết yếu nhất, nên nó dễ dàng đạt được hiệu suất cao hơn những framework công kênh, hỗ trợ nhiều tính năng.

## **2.4. Couchbase Server**

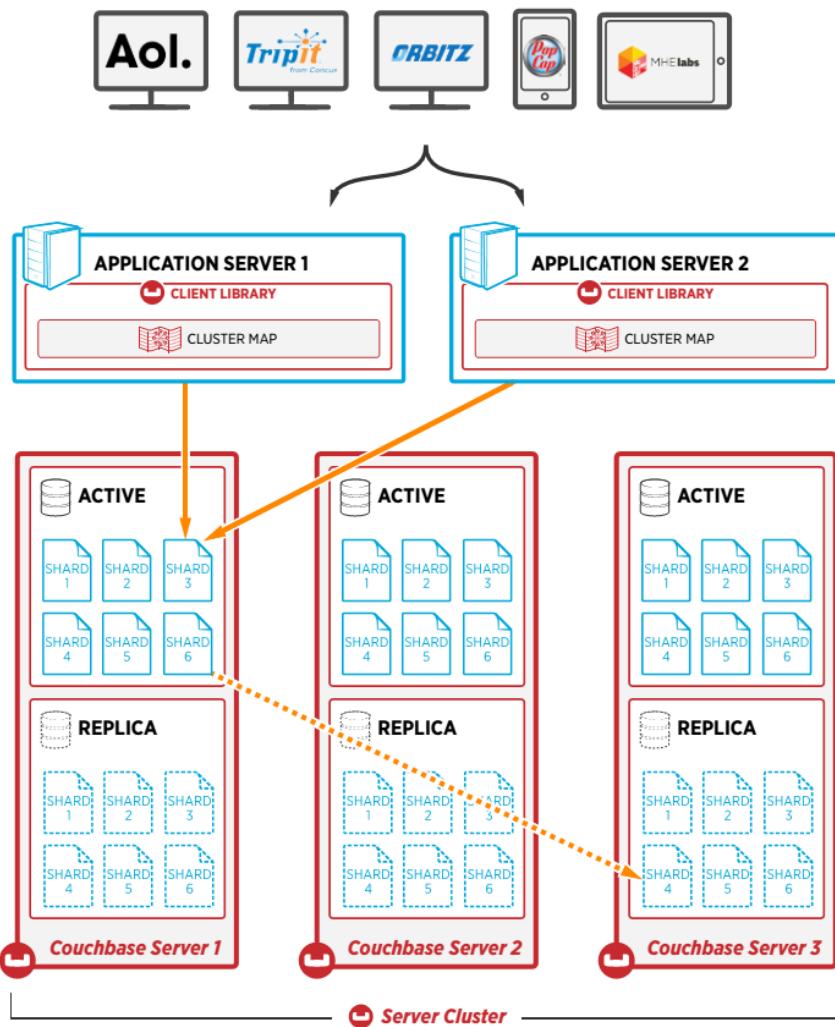
### **2.4.1. Giới thiệu**

Couchbase Server là một loại CSDL NoSQL phân tán hướng document và mã nguồn mở được sử dụng cho các ứng dụng tương tác, và đặc biệt thích hợp cho những ứng dụng cần một mô hình dữ liệu mềm dẻo, dễ mở rộng, hiệu suất cao ổn định, và khả năng đáp ứng 24x365.

Couchbase cung cấp một loại CSDL NoSQL có hiệu suất cao nhất, có khả năng mở rộng (scale) dễ dàng nhất và mang đầy đủ tính năng nhất trên thế giới. Couchbase Server được thiết kế từ tư tưởng đơn giản: Xây dựng một CSDL NoSQL đa dụng, đầu tiên và tốt nhất. Nhờ mục tiêu đó mà Couchbase Server đã trở thành một giải pháp hàng đầu bao gồm: kiến trúc không-chia-sẻ-bất-kì-thứ-gì giữa các node trong cluster, tầng caching được xây dựng sẵn, cơ chế chia nhỏ dữ liệu tự động. Couchbase Server là một dự án mã nguồn mở. Các khách hàng lớn của Couchbase bao gồm Amadeus, Bally's, Beats Music, Cisco, Comcast, Concur, Disney, eBay/ Paypal, Neiman Marcus, Orbitz, Rakuten/ Viber, Sky, Tencent và Verizon, cũng như nhiều khách hàng lớn khác.

### **2.4.2. Kiến trúc deployment mức cao**

Couchbase Server có một kiến trúc không-chia-sẻ (Là một kiến trúc mà mỗi node bất kỳ trong cluster là độc lập, có thể tự vận hành, và không phụ thuộc vào bất kỳ node nào khác trong cluster – trái ngược với mô hình master-slave). Nó có thể được dùng như là một cụm các server phía sau các server của ứng dụng (back-end).

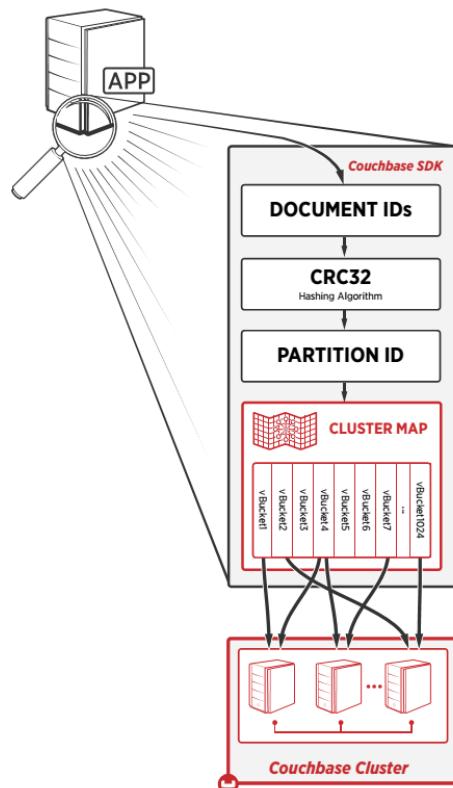


Hình 2.7 Kiến trúc deployment của Couchbase Server

Dữ liệu có thể được lưu trữ trên CouchbaseServer dưới dạng tài liệu JSON hoặc dữ liệu nhị phân. Các document được phân phối trên các node của cluster dựa trên phân phối chuẩn, và được chứa trong các Bucket. Bucket cung cấp một nhóm logic các tài nguyên vật lý trong một cluster. Ví dụ, khi ta tạo một bucket trong Couchbase Server, ta có thể cấp phát cho nó một lượng RAM nhất định cho việc cache dữ liệu và cấu hình số lượng bản sao.

Theo thiết kế, mỗi bucket được chia ra thành nhiều phân vùng logic gọi là vBucket. Các phân vùng được map tới các server trong cluster. Và việc map này được lưu trữ trong một cấu trúc tra cứu được gọi là cluster map. Các ứng dụng sẽ sử dụng thư viện

thông minh của Couchbase để tương tác với server. Khi cần truy xuất đến một document bất kỳ, thư viện thông minh này sẽ thực hiện việc băm id của document bằng thuật toán CRC32 để có được id của phân vùng chứa document đó, sau đó sẽ thông qua cluster map để lấy được thông tin của server node đang chứa nó và truy xuất đến đó.



Hình 2.8 Cơ chế ánh xạ id của document sang phân vùng, và server chứa nó

Một bucket có thể được cấu hình để có thể có đến 3 bản sao trong cùng một cluster. Vì Couchbase Server chia các bucket ra thành nhiều phân vùng, và một server sẽ chỉ quản lý một phần của các phân vùng này. Điều này có nghĩa là, tại bất kỳ thời điểm nào, đối với một phân vùng nhất định, chỉ có một bản chính là hoạt động, và các phân vùng bản sao của nó sẽ nằm trên các server khác. Nếu một server lưu trữ phân vùng chính bị hỏng, cluster sẽ chuyển một trong các bản sao của phân vùng đó trên server khác thành bản chính, để đảm bảo ứng dụng có thể tiếp tục truy xuất đến dữ liệu mà không có bất kỳ khoảng thời gian chết nào.

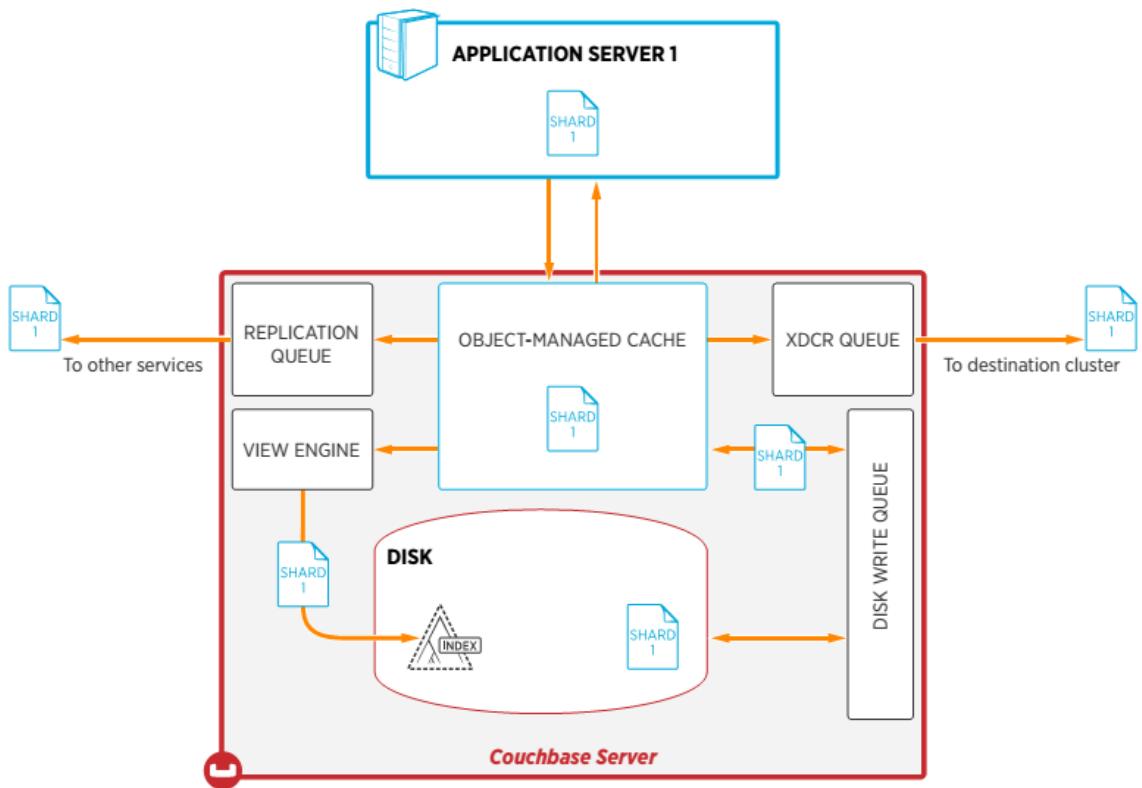
### **2.4.3. Kiến trúc ở client**

Để làm việc được với Couchbase Server, các ứng dụng sẽ sử dụng các SDK tương thích với Memcached (Vì Couchbase Server là sự kết hợp giữa CouchDB và Memcached, SDK dùng có Couchbase Server hoàn toàn tương thích với Memcached) trên nhiều ngôn ngữ như Java, .Net, PHP, Python, Node.js, C và C++. Những thư viện client này nắm rõ cơ chế hoạt động của cluster và có thể tự cập nhật lại cluster map. Nó sẽ tự động gửi các request từ ứng dụng server thích hợp. Phiên bản Couchbase Server 3.0 bổ sung thêm khả năng liên lạc với cluster thông qua SSL.

Sau khi client kết nối với cluster, nó sẽ yêu cầu lấy cluster map từ Couchbase Server cluster và giữ một kết nối luôn mở với server để có thể nhận được các cập nhật của cluster map khi cần thiết. Cluster map được chia sẻ giữa tất cả các node trong cluster và với các client của Couchbase Server. Luồng đi của dữ liệu từ một Couchbase client tới server được tiến hành theo các bước:

- Ứng dụng sẽ liên lạc với Couchbase Server thông qua bộ SDK thông minh.
- Couchbase client lấy id của document và băm nó để chuyển thành id của phân vùng. Dựa vào id của phân vùng và cluster map, client có thể tra cứu ra được document đó thuộc về phân vùng nào nằm trên server nào. Dựa vào đó client sẽ cập nhật document này trên server đó.
- Khi một document đã được gửi tới cluster. Couchbase Server sẽ sao document đó ra để lưu trữ trên các bản sao của phân vùng chứa document đó.

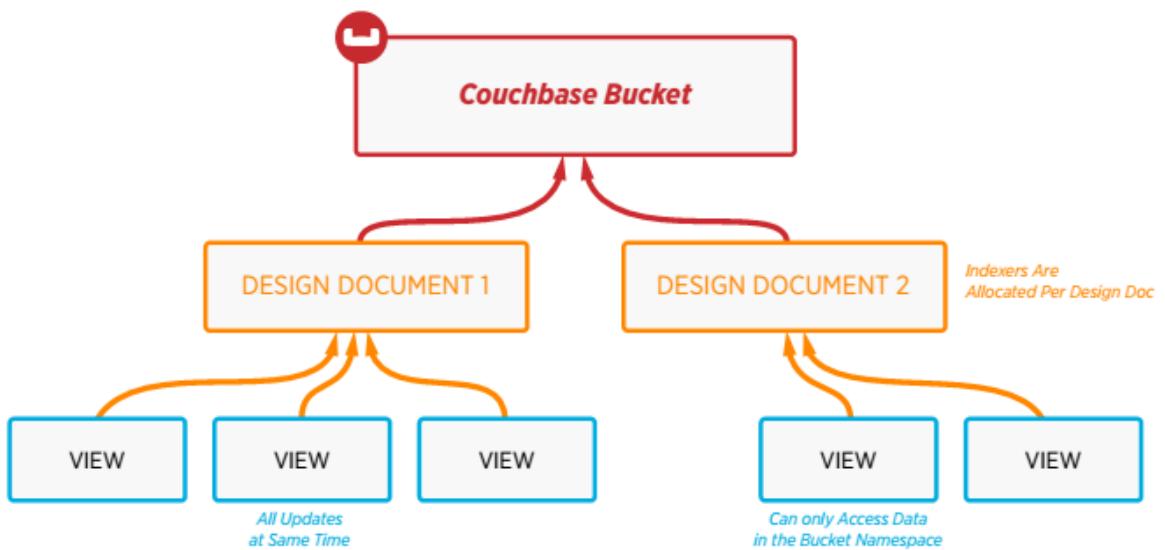
Trong Couchbase Server, sự thay đổi xảy ra ở mức document. Các client sẽ lấy các document từ server, thay đổi một số trường nào đó, và gửi lại cho Couchbase Server.



Hình 2.9 Luồng đi của dữ liệu trong Couchbase Server đối với một tác vụ ghi

#### 2.4.4. Query engine

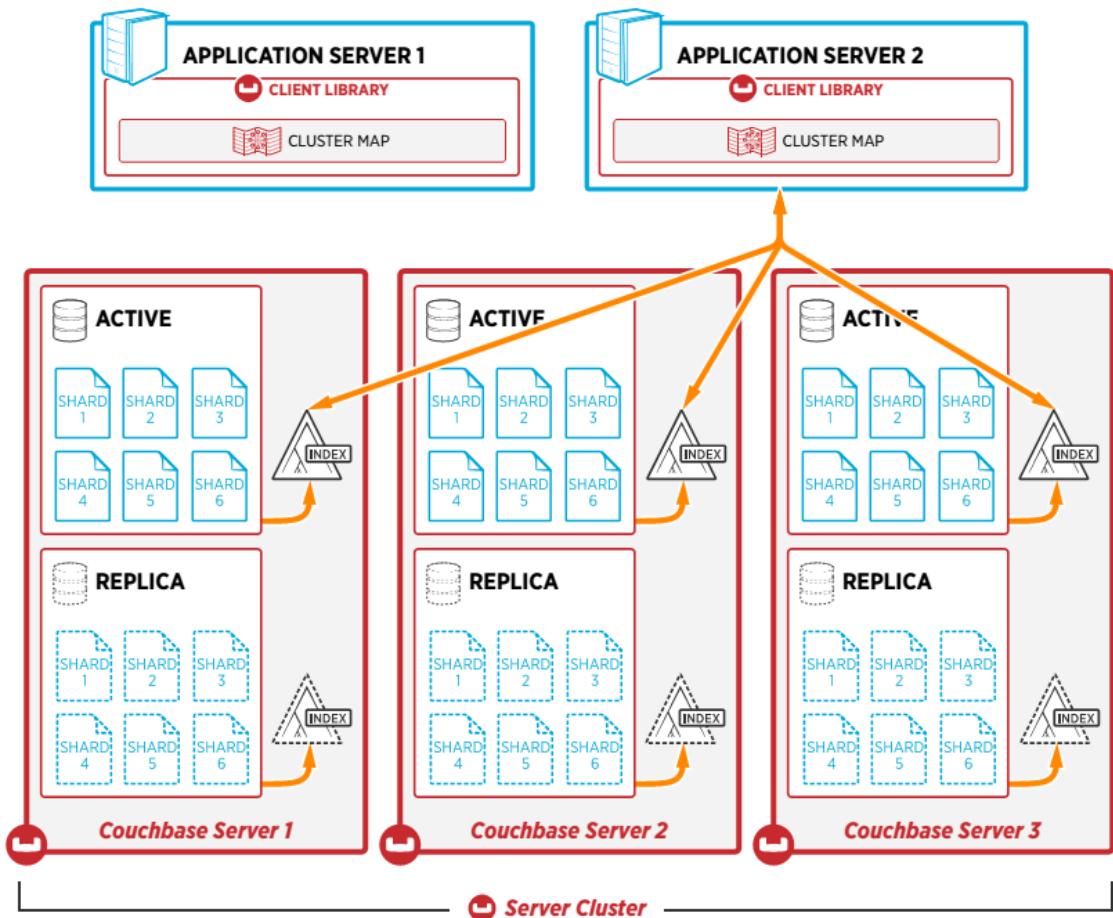
Với Couchbase Server, ta có thể dễ dàng thực hiện việc index và truy vấn các tài liệu JSON. Như được thể hiện trong hình 2.12, các index của view được định nghĩa bằng *design document* và *view*. Mỗi *design document* có thể có nhiều view, và mỗi bucket của Couchbase Server có thể có nhiều *design document*. *Design document* cho phép người dùng có thể gom nhóm các view lại với nhau. Các view nằm trong một *design document* được xử lý song song và các *design document* khác nhau có thể được xử lý song song tại những thời điểm khác nhau.



Hình 2.10 *Design document* và *view* trong Couchbase Server

View trong Couchbase Server được định nghĩa bằng Javascript sử dụng hàm map, hàm này có nhiệm vụ lấy dữ liệu từ document và với một hàm reduce tùy chọn để kết dữ liệu được tạo ra từ hàm map. Hàm map định nghĩa những thuộc tính nào mà dựa trên đó index được xây dựng.

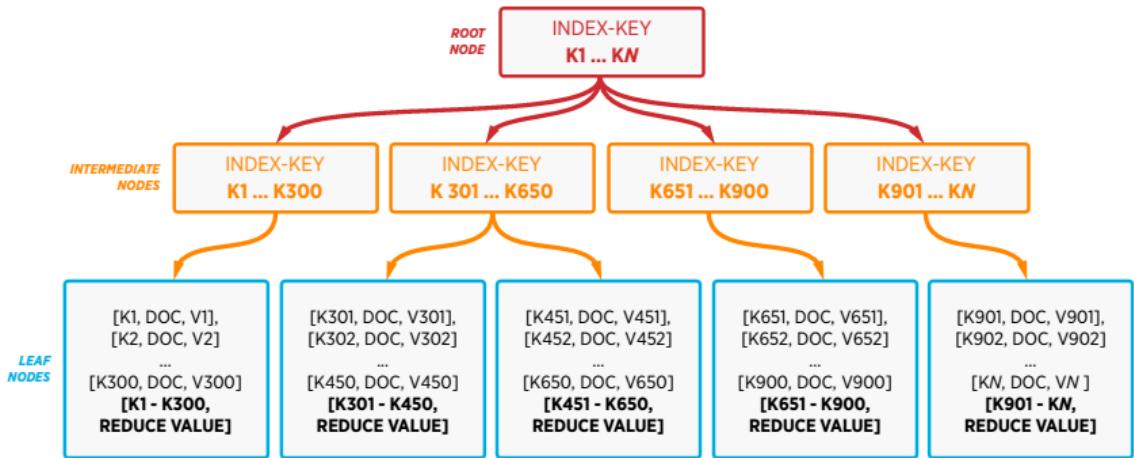
Index trong Couchbase server cũng được phân tán. Mỗi server sẽ thực hiện index trên những dữ liệu mà nó chứa. Như trong hình 2.11, index được tạo ra từ các document.



Hình 2.11 Việc thực hiện index và truy vấn được phân tán ra tất cả các node liên quan trong Couchbase Server

Ứng dụng có thể truy vấn Couchbase Server sử dụng Couchbase SDK, kể cả trong quá trình tái cân bằng. Truy vấn được gửi qua port 8092 tới một server ngẫu nhiên nào đó trong cluster. Server nào nhận được truy vấn này sẽ gửi truy vấn này đến cho những server khác trong cùng một cluster, và kết các kết quả về lại trong khi đang truyền kết quả truy vấn cho client.

Trong quá trình xây dựng view ban đầu, Couchbase Server đọc các file dữ liệu phân vùng trên mỗi server, và thực thi hàm map trên tất cả các document.



Hình 2.12 File index cấu trúc cây-b trong Couchbase Server

Giống như việc index khóa chính, ta có thể bóc một hoặc nhiều thuộc tính của document ra làm khóa, và việc truy vấn dựa trên khóa này có thể sẽ cho ra một hoặc nhiều kết quả (do có thể có sự trùng lặp, vì bất kỳ thuộc tính nào trong document đều có thể làm khóa của view). Ví dụ, như trong hình 2.13, index của view được định nghĩa bằng cách sử dụng thuộc tính name.

```

▼ VIEW CODE
Map
1 function (doc, meta) {
2   if(doc.name)
3     emit(doc.name, null);
4 }

```

Key      Value

Index Definition

Hình 2.13 Định nghĩa index của view bằng thuộc tính name

Index trong Couchbase Server được xây dựng theo hướng “lớn lên dần”. Nếu một view được truy xuất lần đầu tiên, các hàm map và reduce sẽ được thực thi trên tất cả các document trong bucket. Mỗi truy xuất tới view chỉ xử lý những document nào mới được thêm vào, cập nhật, hay xóa kể từ lần cập nhật lần cuối của view. Mặc định, mỗi 5 giây, view engine sẽ kiểm tra xem đã có nhiều hơn 5000 document bị thay đổi từ lần cập nhật trước chưa, nếu đúng thì sẽ thực hiện tác vụ cập nhật view.

#### **2.4.5. Các đặc điểm nổi trội của Couchbase Server**

Couchbase Server là một sự kết hợp giữa hệ CSDL NoSQL phân tán CouchDB, và Memcached. Nó mang nhiều đặc điểm nổi trội để được ứng dụng vào các hệ thống phân tán lớn, có khả năng đáp ứng cao và độ trễ thấp.

##### Kiến trúc không-chia-sẻ-gì

Couchbase Server được thiết kế dựa trên kiến trúc không-chia-sẻ-gì, do đó tất cả các node trong Couchbase Cluster là ngang hàng nhau, và có thể hoạt động độc lập. Mỗi node đều bao gồm một hệ quản lý dữ liệu và hệ quản lý cluster. Khi một node trong cluster gặp trục trặc, hư hỏng hoặc khi thêm một node mới vào cluster thì Couchbase Server chỉ tái thiết lập lại cluster map, và hệ thống hoàn toàn có thể hoạt động bình thường.

##### Cơ chế sharding tự động dựa trên cơ chế băm khóa

Khi được deploy thành một cluster, Couchbase Server sẽ hỗ trợ cơ chế sharding tự động. Đơn vị tương tác cơ bản của Couchbase server là document. Mỗi document được liên kết với một khóa. Không gian khóa của Couchbase Server được phân vùng dựa trên hàm băm CRC32 thành những đơn vị lưu trữ nhỏ hơn được gọi là vBucket. VBucket được ánh xạ vào các node trên một cluster, và được lưu trữ trong một cấu trúc truy vấn được gọi là cluster map. Cluster map được chia sẻ giữa tất cả các node trong Couchbase Cluster, cũng như giữa những client của Couchbase. Số lượng vBucket là một hằng số (theo thiết kế là 1024), dù trong cluster có bao nhiêu node.

Khi số lượng node trong cluster thay đổi (Mở rộng thêm node, hoặc một vài node bị hư hỏng), vBucket được tái phân phối lại giữa các node trong cluster để dữ liệu giữa các node được cân bằng tương đối với nhau, đảm bảo cho cluster có hiệu suất đọc/ghi cao nhất. Việc này được thực hiện thông qua một tác vụ được gọi là tái cân bằng.

##### Tính sẵn và khả năng dự phòng cao

Tính sẵn sàng cao trong Couchbase Server cluster đạt được thông qua việc sao chép dự phòng ở mức vBucket. Couchbase giữ nhiều bản sao chép của dữ liệu trong cluster

bằng cách sao chép dự phòng các vBucket (tạm gọi là A – vBucket bản chính) thành các bản sao chép (tạm gọi là R – vBucket bản dự phòng) và lưu trữ các bản sao chép này trên các node khác trong cluster. Nếu một node trong cluster bị crash hoặc trở nên không sẵn sàng (do hỏng hóc, lỗi phần cứng...), bộ điều phối của cluster hiện tại sẽ thông báo tới tất cả những node khác trong cluster, và những vBucket bản dự phòng của các vBucket bản chính trên node bị hư hỏng được chuyển thành bản chính (Khi đọc dữ liệu từ Couchbase, dữ liệu chỉ được đọc từ vBucket bản chính). Sau đó, cluster map sẽ được tái cập nhật trên các node trong cluster cũng như trên các client. Quá trình hiệu lực hóa các bản sao này là cơ chế dự phòng của Couchbase Server. Dựa trên một số giới hạn về kiến trúc của Couchbase, cơ chế dự phòng tự động chỉ có thể hoạt động nếu cluster hiện tại có hơn 3 node (sau khi một số node bị hư hỏng và không còn sẵn sàng nữa). Nếu số lượng node của server nhỏ hơn 3, cơ chế dự phòng sẽ cần phải được chạy một cách thủ công bởi người quản trị khi một node bị crash.

#### Khả năng cache tự động

Couchbase Server cung cấp khả năng cache dữ liệu một cách tự động, giúp tạo nên hiệu suất cao khi đọc/ghi những dữ liệu thường được truy xuất. Couchbase Server sử dụng cơ chế cache của Memcached (Vì được hợp nhất giữa CouchDB và Memcached), do đó đạt được hiệu suất rất cao. Những dữ liệu được truy xuất thường xuyên sẽ luôn được nằm trong cache.

### **2.4.6. Một case study điển hình so sánh giữa Couchbase Server và MongoDB**

#### **2.4.6.1. Giới thiệu về Viber case study**

Được bắt đầu vận hành và đưa vào hoạt động vào khoảng 3 năm trước, Viber là một ứng dụng giúp gửi tin nhắn bằng text, tin nhắn thoại, hỗ trợ gọi điện, chat qua video... Vào ngày 2 tháng 4 năm 2014, Viber có khoảng 300 triệu người dùng, và số lượng người dùng tăng lên khoảng 1 triệu mỗi ngày. Vài tỷ tin nhắn được gửi mỗi tháng.

#### **2.4.6.2. Kiến trúc ban đầu**

Kiến trúc ban đầu của Viber bao gồm một tầng các server ứng dụng, đằng sau sử dụng một loại CSDL được lưu trữ trong bộ nhớ.

Đến năm 2011, Viber chuyển sang sử dụng MongoDB, vào thời điểm đó, các phiên bản đầu tiên của MongoDB đã có hỗ trợ cơ chế sharding. Ngoài MongoDB, Viber còn sử dụng thêm một tầng caching bằng Redis nằm trên tầng MongoDB bên dưới. Nhưng dần dần, Viber gặp phải các vấn đề về performance đối với MongoDB, do đó dần dần Viber phải phụ thuộc hơn vào Redis. Nhưng vào thời điểm đó, Redis không hỗ trợ sharding, do vậy họ đã tự tạo một cơ chế sharding trên tầng front-end (server ứng dụng). Mongo vẫn tiếp tục gây nên vấn đề về performance, do đó Viber đã bắt đầu chuyển toàn bộ dữ liệu qua Redis vào năm 2013.

Những điểm đã ảnh hưởng tới việc Viber phải bỏ MongoDB và chuyển toàn bộ dữ liệu hết qua Redis:

- MongoDB không thể đáp ứng được 10 ngàn tác vụ trong một giây, nhưng cái Viber cần là khoảng vài trăm ngàn tác vụ trong một giây. Tập dữ liệu lớn cũng gây cho MongoDB nhiều vấn đề.
- MongoDB không mở rộng tốt lắm khi có quá nhiều server ứng dụng, vì mỗi connection của MongoDB được quản lý trên một thread riêng, do đó nó gây tốn nhiều tài nguyên về CPU và bộ nhớ.
- Cơ chế sharding do Viber định nghĩa để sử dụng cho Redis không dễ dàng mở rộng.

Vì lý do đó mà, hệ thống server của Viber vào thời điểm đó bao gồm:

- Một MongoDB cluster với khoảng 150 server (thiết lập dự phòng: 1 master – 2 slave).
- Ba Redis cluster với tổng khoảng 150 server.

Do vậy, Viber quyết định chuyển đổi sang một CSDL NoSQL khác.

#### **2.4.6.3. Kiến trúc sau khi chuyển đổi sang Couchbase:**

Yêu cầu sau khi chuyển đổi:

- Có performance rất cao, và có thể đáp ứng khoảng một triệu tác vụ trên giây.
- Có thể quản lý những tập dữ liệu lớn.
- Có khả năng thay đổi số lượng node trong cluster mà không gây ảnh hưởng tới performance.
- Vì Viber sử dụng Amazon Web Service, vì vậy sự hư hỏng của các node trong cluster đôi khi hay xảy ra, do đó hệ thống phải có khả năng giải quyết sự hư hỏng của các node mà không làm ảnh hưởng tới sự hoạt động của hệ thống.
- Có khả năng backup dễ dàng.
- Có tính sẵn sàng cao.
- Dễ dàng giám sát.

Sau khi chuyển đổi, Viber có khoảng 400 server ứng dụng, và:

- 7 Couchbase cluster, một số cluster có số lượng node lên tới 60, mỗi server có khoảng 60GB RAM.
- Tổng cộng có khoảng 150 Couchbase server (nhỏ hơn một nửa so với số lượng server MongoDB và Redis ban đầu).

### **2.5. Netty**

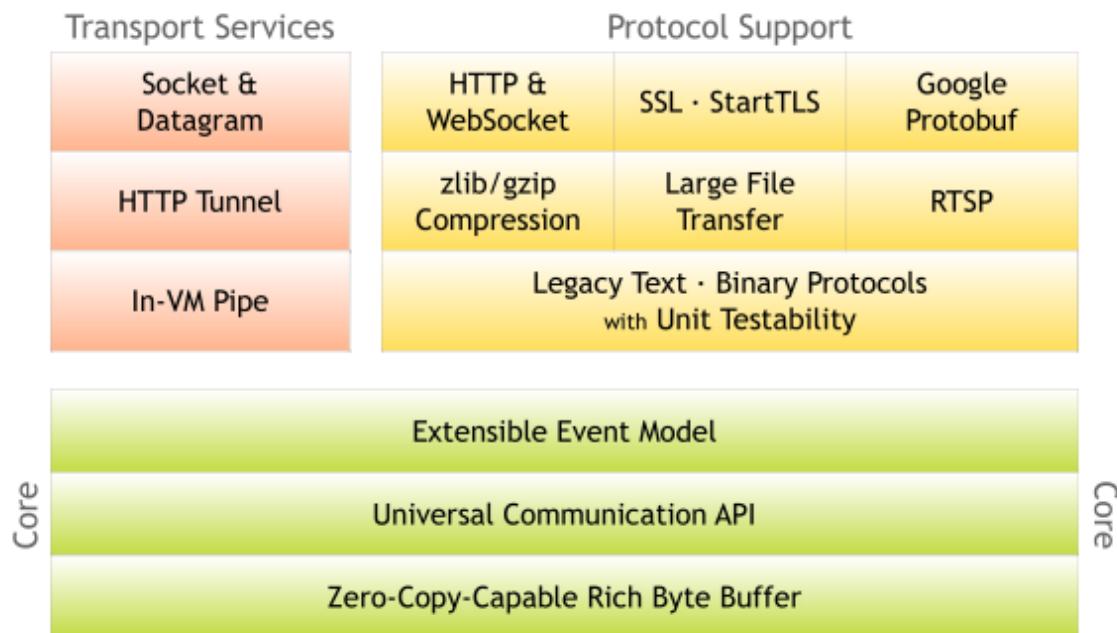
#### **2.5.1. Giới thiệu**

Netty là một framework phát triển ứng dụng mạng theo mô hình hướng sự kiện bất đồng bộ, dùng để tăng tốc độ khi phát triển các ứng dụng client-server có performance cao và dễ dàng bảo trì. Nó giúp đơn giản hóa các công việc liên quan đến lập trình mạng như: xây dựng các server sài TCP socket hoặc UDP socket.

Khi nói đến “nhanh và dễ dàng” ta thường nghĩ ngay đến kết quả là ứng dụng gấp những vấn đề về performance hoặc khả năng bảo trì. Nhưng Netty đã được viết và thiết kế một cách rất kĩ càng, nhờ vào kinh nghiệm từ việc cài đặt các giao thức như FTP, SMTP, HTTP và những giao thức dựa-trên-chuỗi hoặc giao thức nhị phân khác.

Và kết quả là, Netty đã rất thành công trong việc tìm ra cách để đạt được sự đơn giản khi phát triển, hiệu suất, tính ổn định, và tính uyển chuyển mà không phải hy sinh đi khác khía cạnh quan trọng khác của một framework.

### 2.5.2. Kiến trúc



Hình 2.14 Kiến trúc của Netty

### 2.5.3. Đặc tính

Về thiết kế

- Netty cung cấp một chuẩn API thông nhất cho nhiều phương thức truyền tải dữ liệu: blocking và non-blocking.
- Dựa trên mô hình sự kiện uyển chuyển, dễ mở rộng, cho phép phân chia hệ thống ra các thành phần rõ ràng.
- Mô hình thread dễ tùy biến – Có thể sử dụng 1 thread, nhiều thread hoặc thread pool.
- Hỗ trợ giao thức truyền tải gói dữ liệu không-kết-nối.

Về tính dễ dàng khi phát triển

- Cung cấp nhiều ví dụ, hướng và Javadoc được mô tả kĩ.
- Không phụ thuộc vào bất kỳ thư viện nào khác, chỉ JDK5 (cho Netty 3.x) hoặc JDK6 (cho Netty 4.x) là đủ.

Về hiệu suất

- Hiệu suất cao, độ trễ thấp.
- Ít tiêu tốn tài nguyên.
- Giảm thiểu sự copy dữ liệu không cần thiết trên bộ nhớ.

Về tính bảo mật

- Hỗ trợ đầy đủ chuẩn bảo mật StartTLS hoặc SSL/TLS.

Về cộng đồng

- Netty luôn được đảm bảo release sớm và có chu kỳ release ổn định.
- Tác giả của Netty có kinh nghiệm phát triển nhiều framework tương tự từ năm 2003.

#### 2.5.4. Lý do lựa chọn

Hệ thống mạng xã hội bao gồm nhiều tính năng, và một vài trong số những tính năng đó đòi hỏi client và server phải liên tục giao tiếp với nhau. Việc giao tiếp này nếu được thực hiện thông qua một Restful Webservice sẽ gây tốn rất nhiều chi phí, làm tăng độ trễ của server cũng như làm số lượng operation/s giảm nhanh chóng khi có một lượng lớn người truy cập.

Do vậy, để đảm bảo được hiệu suất và để server có thể đáp ứng được một lượng lớn người dùng thì cần phải có một server xử lý real-time, tương tác trực tiếp với client thông qua kết nối TCP được giữ trong suốt quá trình hoạt động của client.

Netty là một framework phát triển ứng dụng mạng được thiết kế tốt dựa trên các “best design practices” trong thực tế, có performance cao, và cộng đồng lớn. Do vậy, nhóm lựa chọn Netty để cài đặt server để đáp ứng một số xử lý real-time của mạng xã hội.

## 2.6. Redis

### 2.6.1. Giới thiệu

Redis là một kho lưu trữ cấu trúc dữ liệu trong bộ nhớ, được dùng như một CSDL, cache hoặc một hệ thống phân phối thông điệp. Nó hỗ trợ các cấu trúc dữ liệu như chuỗi, mảng băm, danh sách liên kết, tập hợp, tập hợp được sắp xếp... Redis hỗ trợ sẵn tính năng sao chép, scripting bằng Lua, transaction và nhiều cấp độ lưu trữ dưới đĩa khác nhau, cung cấp khả năng đáp ứng cao thông qua “Redis Sentinel” và tự động phân vùng với Redis Cluster.

Để đảm bảo Redis có thể được truy xuất bởi nhiều thread, nhiều tiến trình, hoặc nhiều client cùng một lúc, Redis đảm bảo các operation trên các cấu trúc dữ liệu của nó như chèn vào chuỗi, tăng giá trị trong một mảng băm, thêm một phần tử vào danh sách liên kết, tính toán hợp/giao/sự tương quan... là atomic.

Để có thể đạt được hiệu suất vượt trội, Redis làm việc trực tiếp với các tập dữ liệu trong bộ nhớ. Tùy thuộc vào mục đích của người sử dụng mà họ có thể tùy chỉnh việc lưu trữ dữ liệu vào bộ nhớ để phòng trường hợp mất dữ liệu khi server bị sự cố hoặc tắt đột ngột. Tính năng lưu trữ dữ liệu xuống đĩa cứng có thể được tắt đi, nếu người sử dụng chỉ muốn dùng Redis như một bộ cache lưu trữ trên bộ nhớ.

Redis hỗ trợ hầu hết các ngôn ngữ được sử dụng trong môi trường production như: C, C++, C#, Common Lisp, Dart, Erlang, Go, Java, Javascript, Lua, Objective-C, Perl, PHP, Pure Data, Python, R, Racket, Ruby, Rust, Scala, Smalltalk và tcl.

### 2.6.2. Lý do lựa chọn

Redis là một trong những CSDL NoSQL có hiệu suất cao nhất và được sử dụng rộng rãi nhất. Dữ liệu được Redis lưu trữ trực tiếp trên các cấu trúc dữ liệu trong bộ nhớ, do đó nó đạt hiệu suất rất cao. Đối với những trường hợp đòi hỏi nhiều request đọc/ghi liên tục và không đòi hỏi quá cao về việc đảm bảo dữ liệu không bị mất, thì Redis phù hợp hơn nhiều so với các CSDL quan hệ truyền thống hoặc CSDL NoSQL sử

dụng đĩa cứng làm nơi lưu trữ dữ liệu chính. Do vậy, nhóm lựa chọn Redis để lưu trữ chat log và các dữ liệu cần được lưu trữ/xử lý thời gian thực khác.

## 2.7. Amazon S3

Amazon S3 là một dịch vụ lưu trữ được cung cấp bởi Amazon, giúp người dùng có thể lưu trữ hay truy xuất dữ liệu của mình dù bất kỳ độ lớn nào, bất kỳ thời điểm nào, và tại bất kỳ đâu.

Kiến trúc của Amazon S3 được thiết kế để thân thiện với các ngôn ngữ lập trình, sử dụng các interface của Amazon S3 để lưu trữ và lấy các object (Mỗi file lưu trữ trên Amazon S3 sẽ được gọi là object).

Amazon cung cấp 2 interface tương tác chính là interface theo chuẩn REST và interface theo chuẩn SOAP.

Amazon S3 REST API là một interface giao tiếp với Amazon S3 thông qua giao thức HTTP. Sử dụng REST API, người dùng có thể sử dụng các request cơ bản của giao thức HTTP để thực hiện việc tạo, lấy, cập nhật hoặc xóa các object.

## **Chương 3. CÁC VẤN ĐỀ GẶP PHẢI VÀ HƯỚNG GIẢI QUYẾT**

### **3.1. Khả năng đáp ứng lượng lớn người dùng của một mạng xã hội**

#### **3.1.1. Phát biểu vấn đề**

Khác với các ứng dụng, và hệ thống thông thường. Ứng dụng mạng xã hội là một loại hình ứng dụng mà người dùng thực hiện rất nhiều tác vụ/tương tác trong một thời gian ngắn như: Lướt news feed, like một status, comment lên một status, vào xem comment, liên tục chat với bạn bè... Do vậy, lượng request mà một client gửi lên là khá lớn.

Khi một mạng xã hội trở nên thông dụng, nó có xu hướng sẽ lan truyền rất nhanh (những người sử dụng sẽ có xu hướng giới thiệu bạn bè mình sử dụng – vì như vậy họ có thể tương tác với bạn bè của mình trên mạng xã hội, và những người chưa sử dụng MXH bao giờ nhìn thấy bạn bè mình dùng sẽ muốn sử dụng để có thể kết nối với bạn bè của mình qua một kênh khác). Do vậy, lượng người dùng đăng ký sử dụng sẽ gia tăng nhanh chóng. Do đó, đòi hỏi hệ thống phải có khả năng quản lý và đáp ứng lượng lớn người dùng.

Ngoài ra, ứng dụng mạng xã hội là một loại hình ứng dụng mà các người dùng liên tục thực hiện các tương tác ảo với nhau, và người dùng có thể sử dụng mọi lúc, mọi nơi – nhất là đối với những ứng dụng mạng xã hội trên di động. Do vậy, so với các ứng dụng/hệ thống thông thường, số lượng người dùng tại một thời điểm của mạng xã hội là khá lớn.

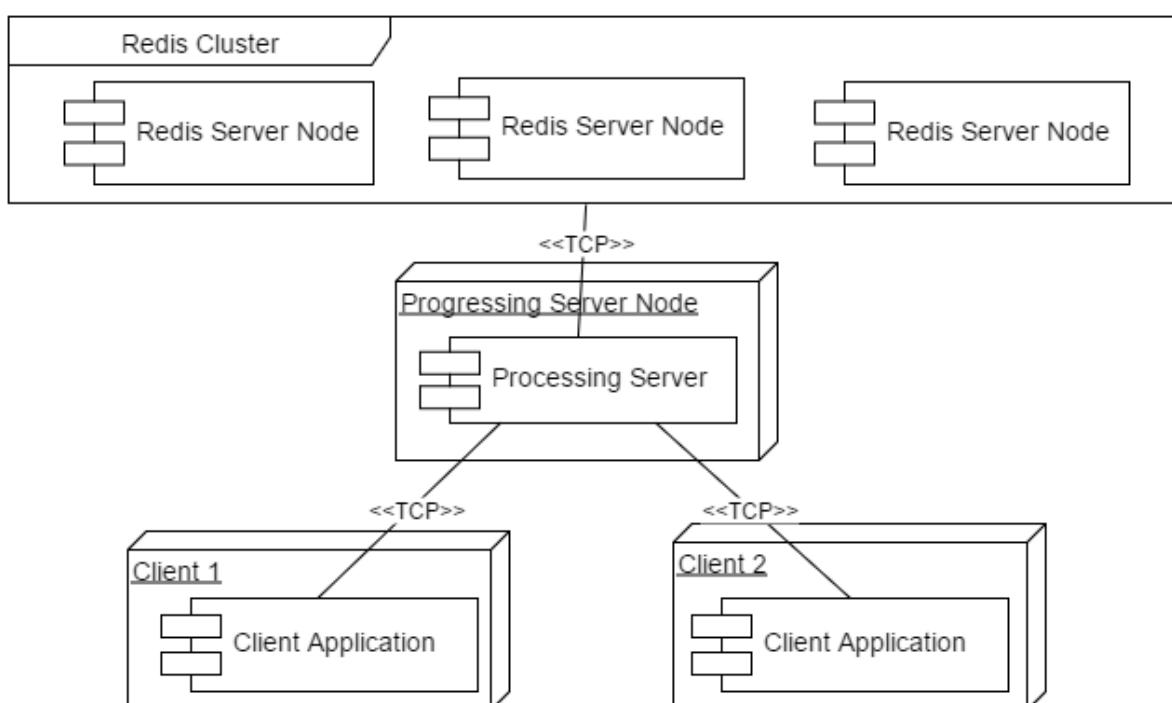
Do vậy, có 3 vấn đề đòi hỏi hệ thống mạng xã hội phải có khả năng đáp ứng một lượng lớn người dùng là:

- Số lượng request tại một thời điểm mà ứng dụng mạng xã hội gửi lên hệ thống mạng xã hội lớn hơn nhiều, nếu so sánh với các ứng dụng thông thường.
- Lượng người/tài khoản đăng ký và sử dụng lớn, khi mạng xã hội đó trở nên thông dụng.
- Số lượng người sử dụng đồng thời tại một thời điểm khá lớn.

### 3.1.2. Giải pháp giải quyết

Đối với Chat Server và Realtime Server của hệ thống mạng xã hội mà nhóm xây dựng, thành phần chịu tải nặng nhất là CSDL, vì hầu hết các xử lý của Chat Server Realtime Server xoay quanh việc đọc/ghi liên tục CSDL, để có thể cung cấp cho client dữ liệu thời gian thực tại bất kỳ thời điểm nào. Do vậy, nhóm bắt đầu từ việc phân tán server CSDL (Cụ thể là Redis). Vì Redis là một CSDL NoSQL (Tuy thường được dùng cho cache, nhưng Redis có khả năng đáp ứng rất tốt cho việc lưu trữ dữ liệu đòi hỏi truy xuất với hiệu suất cao), việc phân tán của Redis sẽ dễ dàng hơn so với các CSDL quan hệ. Mặc khác, bản thân Redis đã tích hợp khả năng phân tán một cách tự động, vì vậy việc thực hiện phân tán Server Redis được thực hiện khá dễ dàng.

Mô hình kiến trúc đầu tiên mà nhóm thiết kế:



Hình 3.1 Kiến trúc thiết kế ban đầu của hệ thống server xử lý

Đối với mô hình kiến trúc này, hệ thống có thể chịu tải và đáp ứng cho một lượng người dùng khá lớn, vì thành phần chịu tải nặng nhất của server đã được phân tán thành một cụm server, để việc đọc/ghi dữ liệu được chia sẻ ra nhiều Redis Node. Nhưng mô hình kiến trúc này vẫn còn một điểm giới hạn có thể dễ dàng nhận ra, khi

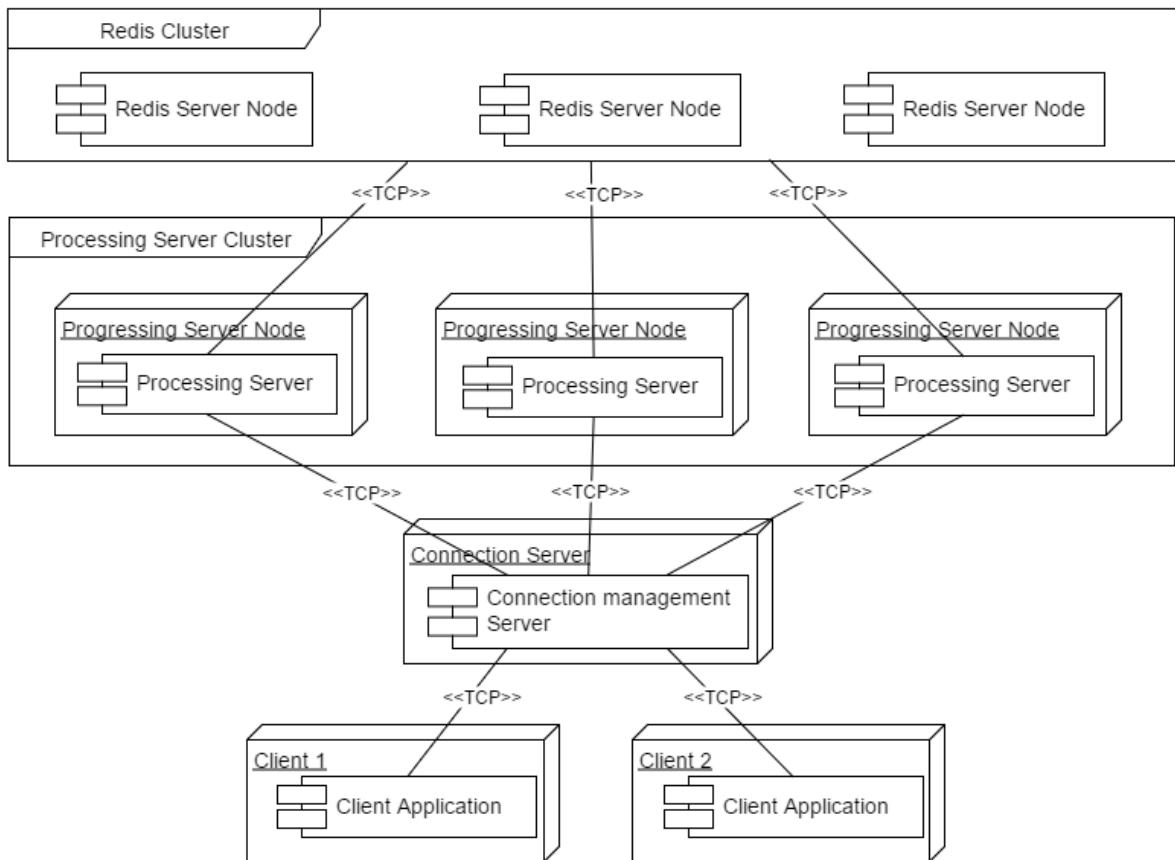
lượng người dùng trở nên quá lớn và đạt tới một lượng nhất định, một server xử lý sẽ không thể nào chịu tải nỗi, đòi hỏi cần phải phân tán server xử lý.

Việc phân tán server xử lý gặp phải một vấn đề lớn, đó là việc quản lý kết nối TCP từ tất cả các client. Nếu phần việc xử lý được phân tán ra nhiều server xử lý, và một client bất kỳ sẽ kết nối tới một server xử lý bất kỳ, thì sẽ gặp phải trường hợp: Server xử lý này muốn gửi thông điệp/lệnh tới một client đang kết nối tới một server xử lý khác.

Giải pháp đơn giản nhất cho vấn đề này là cho phép các server xử lý có thể liên lạc với nhau bằng phương thức broadcast (Khi một server xử lý bất kỳ muốn gửi một thông điệp/lệnh đến một client bất kỳ, thì nó sẽ gửi đến tất cả các server xử lý khác thông điệp “Hãy gửi <thông điệp/lệnh> này cho <client này>”). Nhưng việc liên lạc giữa các server xử lý bằng phương pháp broadcast sẽ trở thành điểm giới hạn phân tán của hệ thống này, vì khi số lượng server xử lý trong cluster càng lớn, chi phí để thực hiện broadcast sẽ càng lớn dần lên. Và khi số lượng server trong cluster đạt tới một mức nào đó, việc thêm một server vào cluster cũng không làm tăng thêm performance của toàn bộ hệ thống (đôi khi còn làm giảm), do chi phí để thực hiện broadcast trở nên quá lớn.

Do đó, nhóm đã chọn giải pháp tách thành phần quản lý kết nối tcp ra một server riêng. Khi đó, connection server này (server chịu trách nhiệm quản lý tất cả các kết nối tcp của client với hệ thống) sẽ làm cầu nối trung gian để liên lạc giữa client với hệ thống. Khi client gửi một thông điệp/lệnh nào đó lên hệ thống, connection server này sẽ chọn ngẫu nhiên một server bằng hàm random theo phân phối chuẩn, sau đó sẽ gửi thông điệp/lệnh của client cho server đó xử lý. Khi một server xử lý bất kỳ muốn gửi thông điệp/lệnh đến một user bất kỳ, thì sẽ gửi lệnh đó (kèm theo yêu cầu gửi lệnh tới user này) tới connection server, và connection server sẽ thực hiện truyền tải thông điệp/lệnh này đến những client liên quan (Một user có thể đăng nhập trên nhiều client cùng một lúc, khi đó thì thông điệp/lệnh này sẽ được chuyển tới nhiều client).

Mô hình kiến trúc phân tán của nhóm sau khi tách phần quản lý kết nối ra một connection server riêng:



Hình 3.2 Kiến trúc cải tiến với sự phân tán của cụm server xử lý

Connection server này chỉ đảm nhiệm 2 nhiệm vụ:

- Chuyển thông điệp/lệnh của client tới một server xử lý (Tương tự như một load balancer).
- Chuyển thông điệp/lệnh của một server xử lý bất kỳ tới một user (Những client) tương ứng.

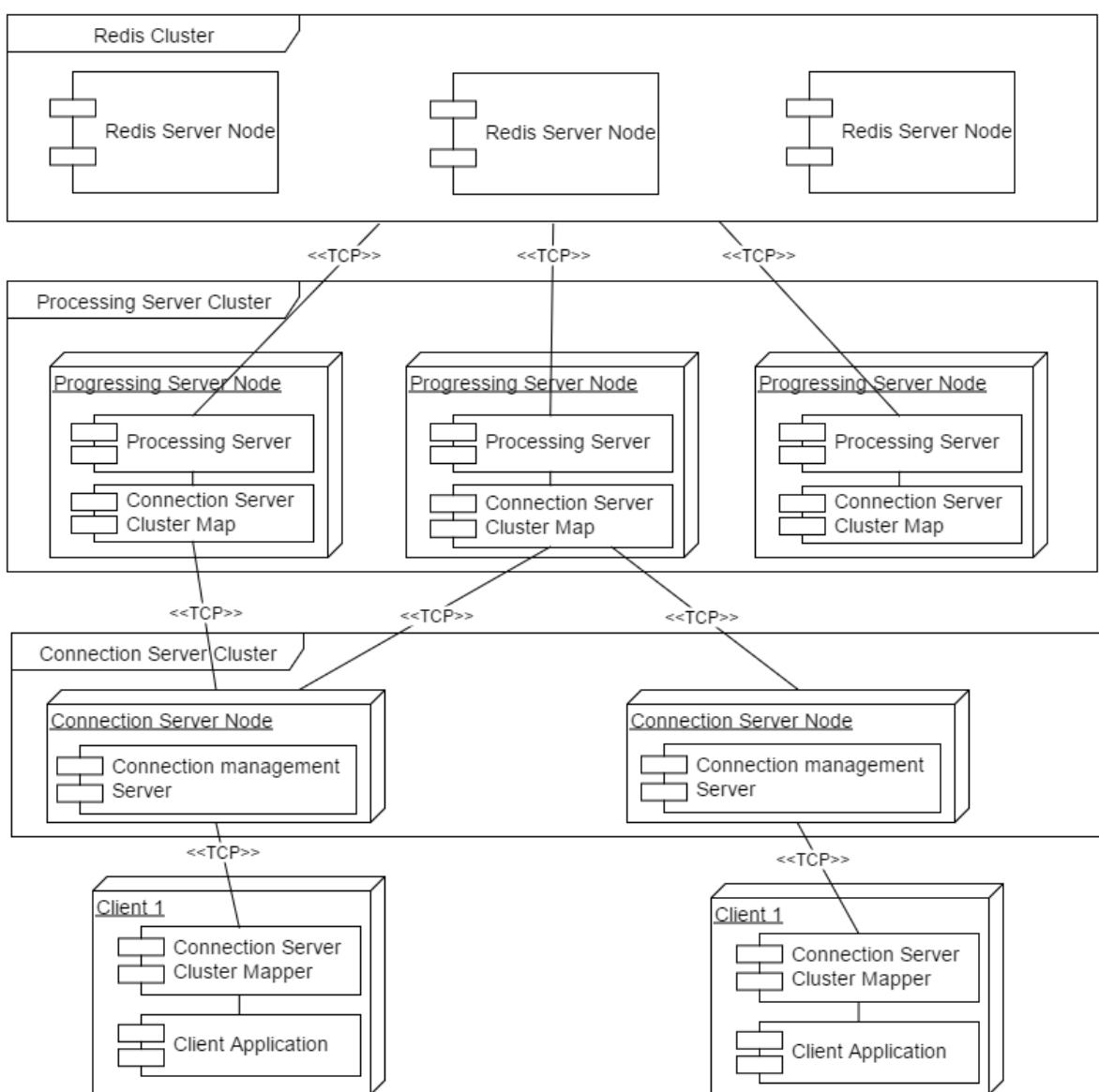
Với mô hình kiến trúc phân tán này, hệ thống đã có khả năng đáp ứng được cho một lượng rất lớn người dùng, chỉ duy nhất connection server trong hệ thống là không có khả năng phân tán, nhưng vì nhiệm vụ của connection server chỉ là cầu nối trung gian giữa client và cụm server xử lý, nó có thể dễ dàng quản lý được lượng lớn kết nối.

Nhưng khi lượng người dùng sử dụng đồng thời quá lớn, lớn tới mức một connection server không thể quản lý được kết nối của tất cả các client kết nối đến nó, khi đó hệ

thống sẽ không thể nào đáp ứng nổi, và đòi hỏi ta phải tìm ra giải pháp để phân tán connection server.

Qua quá trình nghiên cứu các mô hình phân tán của các CSDL NoSQL, cũng như các mô hình và giải pháp phân tán của các hệ thống lớn. Nhóm quyết định áp dụng mảng-băm-phân-tán (Distributed hash table – Một cơ chế phân tán cũng được áp dụng trong Couchbase Server) để giúp phân tán connection server.

Mô hình kiến trúc của server, sau khi áp dụng mảng-băm-phân-tán để phân tán connection server:



### Hình 3.3 Kiến trúc phân tán tất cả các thành phần trong hệ thống server xử lý

Trong mô hình này, mỗi server xử lý và mỗi client sẽ có một Connection Server Cluster Mapper (là một dạng mảng-băm-phân-tán) nhằm giúp client và server xử lý biết được Connection Server nào sẽ chịu trách nhiệm quản lý kết nối của Client nào. Connection Server Cluster Mapper là một hàm băm để ánh xạ giữa user id người dùng với địa chỉ và port của connection server quản lý kết nối của user đó (Ánh xạ:  $\text{CRC16(user\_id)} \% \text{số\_lượng\_server} \rightarrow (\text{địa chỉ}, \text{port})$  của connection server quản lý user đó). Khi một client muốn kết nối đến hệ thống, nó sẽ băm user-id của người dùng bằng CRC16, rồi lấy giá trị đó tra cứu trong Connection Server Cluster Mapper để biết được địa chỉ và port server mà client cần kết nối đến. Tương tự như vậy, mỗi khi server xử lý muốn gửi thông điệp cho một user nhất định, server xử lý cũng sẽ băm user-id của người dùng bằng CRC16, rồi lấy giá trị đó tra cứu trong Connection Server Cluster Mapper để biết địa chỉ và port của connection server chịu trách nhiệm quản lý kết nối của user này, và yêu cầu connection server đó truyền thông điệp/lệnh cho user này.

Với mô hình này, tất cả các thành phần trong hệ thống đều có khả năng mở rộng theo chiều ngang. Và khi client gửi lên bất kỳ một thông điệp/lệnh nào thì nó chỉ được tiếp nhận bởi một connection server, được xử lý bởi một server xử lý, trong quá trình xử lý không tồn thêm bất kỳ chí phí nào. Do vậy, hệ thống này gần như có khả năng mở rộng vô hạn, và có khả năng đáp ứng gần như “vô số” người dùng, miễn số lượng server node trong cluster đủ để chịu tải cho lượng người dùng này.

Cách thức hoạt động của hệ thống này sẽ được mô tả rõ hơn trong phần kiến trúc của hệ thống.

## **3.2. Xây dựng news feed**

### **3.2.1. Phát biểu vấn đề**

News feed là một trong những tính năng quan trọng nhất của mạng xã hội. Giúp người dùng có thể cập nhật được những thông tin, trạng thái từ bạn bè cũng như thông tin của những gì xảy ra trên mạng, hay ngoài đời một cách đơn giản và dễ dàng nhất.

Mục tiêu chính của news feed là đem tới cho người dùng những thông tin mà họ cần và những thông tin mà họ muốn đọc nhất. Do đó, việc tối ưu news feed để chỉ đưa lên những thông tin mà người dùng cần là một trong những yếu tố cực kì quan trọng. Vì khi lượng kết nối, bạn bè của người dùng lớn dần lên, thì news feed sẽ chứa rất nhiều thông tin rác, nhiều thông tin rác, và sẽ làm ảnh hưởng tới chất lượng của mạng xã hội.

### **3.2.2. Giải pháp giải quyết**

Giải pháp đơn giản nhất để sinh news feed là lấy cập nhật trạng thái của tất cả kết nối/bạn bè của người dùng, sau đó sắp xếp các cập nhật trạng thái theo thời gian. Ưu điểm của phương pháp này là đơn giản, dễ cài đặt, và chi phí thực hiện thấp, nhưng nó lại mang lại trải nghiệm không tốt cho người dùng. Vì khi lượng kết nối/bạn bè của người dùng quá lớn, số lượng kết nối/bạn bè thân thiết trong số ít, thì hầu như news feed của người dùng sẽ tràn lan những thông tin mà người dùng không muốn xem.

Giải pháp thứ 2 để sinh news feed là lấy cập nhật trạng thái của tất cả các kết nối/bạn bè của người dùng, sau đó sắp xếp theo 2 yếu tố là thời gian và độ thân thiết của mối quan hệ giữa 2 người, nhưng ưu tiên sắp xếp theo độ thân thiết. Nếu có cập nhật trạng thái từ bạn thân thì cập nhật trạng thái này sẽ được ưu tiên hơn cập nhật trạng thái của những người khác, dù thời gian đăng trạng thái này lâu đài hơn các cập nhật trạng thái khác. Giải pháp này khi mới nhìn vào, ta thấy nó rất hiệu quả. Nhưng thật sự, còn có rất nhiều yếu tố khác ảnh hưởng tới mức độ quan tâm của người dùng tới một cập nhật trạng thái, và chỉ dựa vào mức độ thân thiết thì chưa thật sự nói lên được điều

gì. Mặt khác, chỉ dựa vào hoạt động của người dùng trên mạng xã hội thì rất khó để đo được mức độ thân thiết giữa 2 người dùng với nhau.

Sự quan tâm của người dùng tới một cập nhật trạng thái phụ thuộc vào rất nhiều yếu tố khác nhau. Do vậy, giải pháp hiệu quả nhất là kết hợp tất cả yếu tố đó lại với nhau.

Giả sử ta gọi mức độ quan tâm của một người tới một cập nhật trạng thái nhất định là “Mức quan tâm”. “Mức quan tâm” này được ảnh hưởng bởi nhiều yếu tố khác nhau (Ta viết tắt là  $f$  – factor). Mỗi yếu tố ( $f$ ) sẽ có một mức độ ảnh hưởng nhất định tới “Mức quan tâm” của người dùng lên một cập nhật trạng thái, ta sẽ gọi mức độ ảnh hưởng là trọng số của yếu tố này ( $w$  – weight). Khi đó, ta có thể tính được mức độ quan tâm của người dùng dựa vào các yếu tố ảnh hưởng bằng công thức:

$$\text{Mức quan tâm} = \sum_{i=1}^n w_i \times f_i$$

Khi đó, bài toán sinh news feed trở thành bài toán đi tìm: Các yếu tố ảnh hưởng, công thức tính giá trị của yếu tố đó, và trọng số của yếu tố đó so với các yếu tố còn lại. Sau khi tính toán được *Mức quan tâm* của từng cập nhật trạng thái, ta sẽ sắp xếp các cập nhật trạng thái theo thứ tự *Mức quan tâm* giảm dần.

Tham số  $w$  (trọng số) có ảnh hưởng lớn tới kết quả mà thuật toán mang lại. Tham số này có thể được tối ưu bằng cách đặt ra giá trị giả định ban đầu, sau đó mang đi chạy thử trên các tập dữ liệu trong thực tế để đánh giá độ hiệu quả, và điều chỉnh các tham số  $w$  sao cho thuật toán mang lại kết quả cao nhất (Dự đoán gần đúng nhất mức quan tâm tương đối của người dùng với một cập nhật trạng thái so với các cập nhật trạng thái khác).

Yếu tố đầu tiên mà ta có thể xét đến là Mức quan tâm của người dùng đối với những cập nhật trạng thái trước đó của cùng tác giả (Ví dụ, để xét mức quan tâm của người dùng A với cập nhật trạng thái mới nhất của người dùng B, ta sẽ xét mức quan tâm của người dùng A đối với những cập nhật trạng thái trước đó trong quá khứ của B). Yếu tố này dựa vào giả định: Nếu một người dùng A có xu hướng quan tâm tới các

cập nhật trạng thái của người B trong quá khứ, thì người A cũng sẽ có xu hướng quan tâm tới các cập nhật trạng thái trong tương lai của người B. Mức độ quan tâm đối với những cập nhật trạng thái trong quá khứ của B sẽ được đánh giá thông qua số lượng like và comment của A với những cập nhật trạng thái của B trong quá khứ.

Một số yếu tố khác ta có thể đưa vào để xét Mức quan tâm là:

- Thời gian đăng của cập nhật trạng thái: Đây là một yếu tố cực kì quan trọng và không thể thiếu khi xét Mức quan tâm của người dùng. Người dùng sẽ có xu hướng muốn xem những tin tức, cập nhật trạng thái gần đây, hơn là những tin tức, cập nhật trạng thái cách đây 3-4 năm.
- Loại hình nội dung của cập nhật trạng thái: Thông thường nội dung về hình ảnh sẽ được người sử dụng để ý hơn so với nội dung chỉ toàn chữ.
- Số lượng lượt like và comment: Yếu tố này dựa vào giả định “Ta có xu hướng quan tâm hơn tới những vấn đề nóng hổi trong xã hội, và những vấn đề được nhiều người quan tâm”.

### **3.3. Tối ưu hóa truy xuất CSDL**

#### **3.3.1. Phát biểu vấn đề**

Đối với các hệ thống có lượng người dùng lớn, đòi hỏi số lượng tác vụ đọc cao như mạng xã hội, việc tối ưu hóa hiệu suất truy xuất dữ liệu từ CSDL đóng một vai trò rất quan trọng để giúp đảm bảo tính ổn định và giảm thiểu sự quá tải của hệ thống.

#### **3.3.2. Giải pháp giải quyết**

Couchbase Server hỗ trợ 2 phương pháp để truy xuất đọc dữ liệu là: Truy xuất thông qua key và truy vấn.

Tuy là một CSDL phân tán có dữ liệu được chia ra thành các phần nhỏ trên tất cả các node trong một cluster, việc truy xuất thông qua key của Couchbase Server cực kì hiệu quả. Thay vì để server thực hiện toàn bộ công việc, SDK “thông minh” của Couchbase Server sẽ thực hiện một phần việc trên client. Khi truy xuất đến bất kỳ document có key nào, trước tiên client sẽ thực hiện băm key đó bằng CRC32 để lấy

id của phân vùng chứa document (`partition_id = CRC32(document_id) % 1024`). Sau đó dựa vào cluster map để xác định node nào đang chứa document này và truy xuất trực tiếp tới node đó để lấy document này về thông qua key. Do vậy, khi client thực hiện truy xuất lấy một document bất kỳ, chỉ duy nhất một node trên cluster thực hiện xử lý. Do đó, nếu xét về tính tương đối (trường hợp dữ liệu trên tất cả các node trong cluster cân bằng nhau), số-lượng-operation/s Couchbase đáp ứng được sẽ tăng lên  $n$  lần nếu cluster có  $n$  node, khi so sánh với số-lượng-operation/s khi cluster có 1 node.

Khác với truy xuất thông qua key, việc truy vấn dữ liệu trở nên phức tạp hơn nhiều khi dữ liệu bị phân ra thành từng phần nhỏ trên từng node của cluster. Để có thể truy vấn được thì đòi hỏi phải gom tất cả những dữ liệu liên quan về một nơi và thực hiện truy vấn trên tập dữ liệu đó. Do đó, khi client muốn thực hiện một câu truy vấn bất kỳ, nó sẽ gửi câu lệnh truy vấn lên một node bất kỳ trong cluster. Node này sẽ thực hiện yêu cầu lấy tất cả các dữ liệu liên quan trên tất cả các node (thay vì chỉ một node như truy vấn thông qua key), sau đó gom lại, thực hiện truy vấn trên tập dữ liệu đó và trả về dữ liệu kết quả cho client. Ngoài chi phí để thực hiện truy vấn, ta còn phải chịu thêm nhiều chi phí khác như: chi phí truy xuất tất cả dữ liệu liên quan trên tất cả các node, chi phí truyền dữ liệu từ tất cả các node về node đang xử lý (vì tất cả dữ liệu liên quan là khá lớn nên chi phí này khá đáng kể), chi phí để parse tài liệu json để có thể thực hiện các so sánh trong câu truy vấn. Do vậy, chi phí để thực hiện truy vấn lớn hơn rất nhiều so với chi phí thực hiện truy xuất dữ liệu thông qua key.

Do đó, để có thể tối ưu hóa việc truy xuất dữ liệu cho hệ thống mạng xã hội, việc truy xuất phải được thực hiện hoàn toàn thông qua key, và hoàn toàn không dùng đến truy vấn.

Couchbase Server cũng hỗ trợ khái niệm View như trong CSDL quan hệ. Nhưng cơ chế hoạt động của View trong Couchbase Server khác rất nhiều trong CSDL quan hệ, và nó hiệu quả hơn nhiều. View trong CSDL quan hệ chỉ đơn thuần là một “virtual table” với dữ liệu là kết quả của câu truy vấn sql, mỗi khi người dùng truy vấn trên View thì hệ quản trị CSDL lại thực hiện câu truy vấn để доставить dữ liệu vào View, do vậy

hầu như View không giúp cải thiện performance cho việc truy xuất dữ liệu từ CSDL quan hệ. Khác với CSDL quan hệ, View trong Couchbase Server được sử dụng cơ chế “cập nhật gia tăng” (incremental update), với cơ chế này, chỉ lần đầu tiên được tạo ra, Couchbase Server mới cần truy xuất tất cả các document để tính toán và đổ dữ liệu vào View. Sau lần cập nhật đầu tiên, khi nào một document được tạo ra, xóa hoặc cập nhật thì Couchbase Server sẽ chỉ cập nhật thêm document mới đó vào view thông qua hai hàm map-reduce. Dữ liệu của view sẽ được lưu lại sau mỗi lần cập nhật, do vậy, việc truy xuất thông qua view trên Couchbase Server cực kì tối ưu.

Vậy, để có thể đạt được hiệu suất truy xuất dữ liệu cao nhất qua CouchbaseServer, hệ thống sẽ chỉ truy xuất dữ liệu thông qua key, và không dùng tới câu truy vấn. Để làm được điều này thì cần tạo ra các View có index (key của view) thích hợp để có thể truy xuất tất cả các tập dữ liệu cần thiết trong hệ thống mạng xã hội thông qua key.

### **3.4. Đề xuất giải pháp và mô hình thiết kế kiến trúc hệ thống**

#### **3.4.1. Mô hình nhiều lớp**

##### **3.4.1.1. Giới thiệu**

Mô hình nhiều lớp là một trong những mô hình kiến trúc phần mềm thông dụng và phổ biến nhất, nó chia phần mềm thành nhiều thành phần nắm những trách nhiệm riêng biệt. Việc phân chia này tương tự như việc chia để trị, giúp phân rã hệ thống thành những module nhỏ hơn, thực hiện các chức năng khác nhau. Nhờ vậy giúp hệ thống có thể được xây dựng dễ dàng hơn, có cấu trúc rõ ràng hơn cũng như hệ thống sẽ trở nên dễ bảo trì và mở rộng. Tư tưởng của việc phân chia này tương tự tư tưởng của 2 nguyên tắc quan trọng trong thiết kế phần mềm là “Separation of concern” và “Single responsibility principle”.

Trong mô hình nhiều lớp có 3 mô hình con phổ biến nhất là:

- Mô hình 2 lớp: Chia làm 2 lớp chính là lớp Application và lớp DataAccess.

- Mô hình 3 lớp: Chia làm 3 lớp chính là lớp Presentation, lớp Business Logic và lớp DataAccess.
- Mô hình 4 lớp: Chia làm 4 lớp chính là lớp Presentation, lớp Application, lớp Business Logic và lớp DataAccess.

Thông thường số lớp này (việc lựa chọn giữa mô hình 2 lớp, 3 lớp và 4 lớp) sẽ phụ thuộc vào độ lớn của hệ thống.

#### **3.4.1.2. Phân tích lựa chọn**

Thông thường trong thiết kế phần mềm có 2 tư tưởng để phân chia thành phần của phần mềm thành những thành phần nhỏ hơn chịu những trách nhiệm riêng biệt là: phân chia module hóa và phân chia thành nhiều lớp. Phân chia module hóa chỉ đơn giản là việc phân chia chương trình/hệ thống thành những thành phần tách biệt nhau về chức năng/xử lý/trách nhiệm. Còn việc phân chia thành nhiều lớp là phân chia phần mềm/hệ thống thành những thành phần có sự phụ thuộc rõ ràng và thông thường các thành phần sẽ phụ thuộc nhau theo chiều từ trên xuống dưới (Các lớp high-level sẽ phụ thuộc vào các lớp low-level hơn): Lớp bên trên chỉ phụ thuộc vào duy nhất lớp bên dưới của nó (Ví dụ như, trong mô hình 3 lớp Presentation-BusinessLogic-DataAccess, Lớp Presentation chỉ phụ thuộc vào lớp Business Logic, lớp BusinessLogic chỉ phụ thuộc vào lớp DataAccess). Nhờ sự phân chia trách nhiệm và phụ thuộc rõ ràng giữa các lớp với nhau, mà mô hình nhiều lớp giúp phần mềm/hệ thống có thể dễ dàng mở rộng hơn, cũng như giảm thiểu tối đa thay đổi. Vì lớp bên trên chỉ phụ thuộc duy nhất vào lớp bên dưới kè nó, nên khi một lớp bất kỳ thay đổi thì thông thường nó chỉ làm ảnh hưởng tới các lớp liền kề nó.

Nhưng đôi khi việc cứng nhắc, hay lạm dụng lại làm tăng chi phí phát triển. Mỗi lớp trong mô hình nhiều lớp tương tự như một Abstraction Layer (Lớp trừu tượng), nó giúp che đi cài đặt chi tiết của những thành phần bên dưới, nhờ đó mà abstraction layer sẽ giúp bảo vệ các lớp high-level khỏi sự thay đổi của các lớp low-level. Nhưng việc lạm dụng hay dư thừa abstraction layer sẽ làm tăng chi phí và thời gian phát triển.

Do vậy, việc lựa chọn nên chia chương trình/hệ thống thành bao nhiêu lớp sẽ phụ thuộc vào độ lớn của chương trình/hệ thống. Chương trình/hệ thống càng lớn và càng có nhiều chức năng thì ta càng cần nhiều lớp để làm đơn giản nó (bằng cách chia để trị).

Tư tưởng của việc phân chia thành nhiều lớp không chỉ được ứng dụng trong phần mềm, mà nó còn được ứng dụng để thiết kế các giao thức mạng, hạ tầng các hệ thống...

Nhóm sẽ không áp dụng máy móc bất kỳ mô hình con nào của mô hình nhiều lớp, mà chỉ áp dụng tư tưởng của mô hình nhiều lớp để chia hệ thống thành những thành phần nhỏ hơn, để hệ thống có thể dễ dàng bảo trì, nâng cấp, mở rộng hay sửa đổi hơn.

### **3.5. Tích hợp tương tác giọng nói vào mạng xã hội**

Ứng dụng được tích hợp tương tác giọng nói với các chức năng:

- Bật/Tắt chế độ voice command
- Gửi tin nhắn cho bạn bè
- Cập nhật chia sẻ trạng thái, bình luận các bài đăng
- Truy vấn xem trang cá nhân, thông tin của người dùng
- Hiển thị nhanh các thông báo, danh sách lời mời kết bạn, danh sách bạn bè
- Cập nhật và truy vấn nhanh về các trạng thái hiện thời (mood status), vị trí của người dùng, từ đó đưa ra chỉ dẫn hay lời khuyên như gửi tin nhắn để rủ đi ăn
- Tìm kiếm bạn bè qua tên
- Check-in chia sẻ vị trí

## **Chương 4. PHÂN TÍCH, THIẾT KẾ**

*Trong quá trình cài đặt và triển khai hệ thống mạng xã hội, để có thể đảm bảo hệ thống có kiến trúc tốt, đảm bảo được yêu cầu ban đầu, phát triển theo đúng hướng đòi hỏi cần có những tài liệu mô tả cụ thể về chức năng, luồng xử lý, mô hình dữ liệu, cũng như kiến trúc bên trong và giao diện.*

*Chương này, nhóm sẽ trình bày chi tiết các vấn đề khi thiết kế hệ thống, để từ đó có thể tiến hành cài đặt một cách dễ dàng và đảm bảo hệ thống có thể vận hành được trong thực tế.*

### **4.1. Mô tả hệ thống**

Hệ thống mạng xã hội lifeSocial (tên được kết hợp giữa 2 từ life và social network) là một hệ thống mạng xã hội mang đầy đủ các tính năng của mạng xã hội cơ bản như: đăng cập nhật trạng thái, like comment trên cập nhật trạng thái, duyệt news feed, chat, kết bạn... Nó còn bao gồm các tính năng để tăng sự tương tác giữa người dùng và ứng dụng thông qua giọng nói, cũng như các tính năng để tăng sự tương tác thực tế của người dùng.

Nhóm các tính năng của mạng xã hội cơ bản bao gồm:

- Xem danh sách kết nối.
- Gửi yêu cầu tạo kết nối.
- Chấp nhận/hủy yêu cầu tạo kết nối.
- Xem danh sách các cuộc hội thoại.
- Xem lịch sử tin nhắn.
- Gửi tin nhắn thông thường.
- Gửi tin nhắn file/hình ảnh.
- Đăng status nội dung text.
- Đăng status hình ảnh.
- Đăng voice status.
- Like một status.

- Comment trên một status.
- Duyệt news feed.

Nhóm các tính năng tương tác bằng giọng nói:

- Đăng status bằng voice command.
- Gửi tin nhắn bằng voice command.
- Thông báo trạng thái nguy hiểm & yêu cầu sự giúp đỡ từ người thân.
- Thay đổi trạng thái bằng voice command.
- Truy vấn thông tin bạn bè bằng voice command.
- Các tính năng điều khiển ứng dụng thông qua voice command (Go back, go home, xem profile...).

Nhóm các tính năng nâng cao tương tác giữa người dùng với người dùng:

- Thông báo trạng thái nguy hiểm & yêu cầu sự giúp đỡ từ người thân & những người gần đó.

## 4.2. Yêu cầu và mô hình hóa yêu cầu

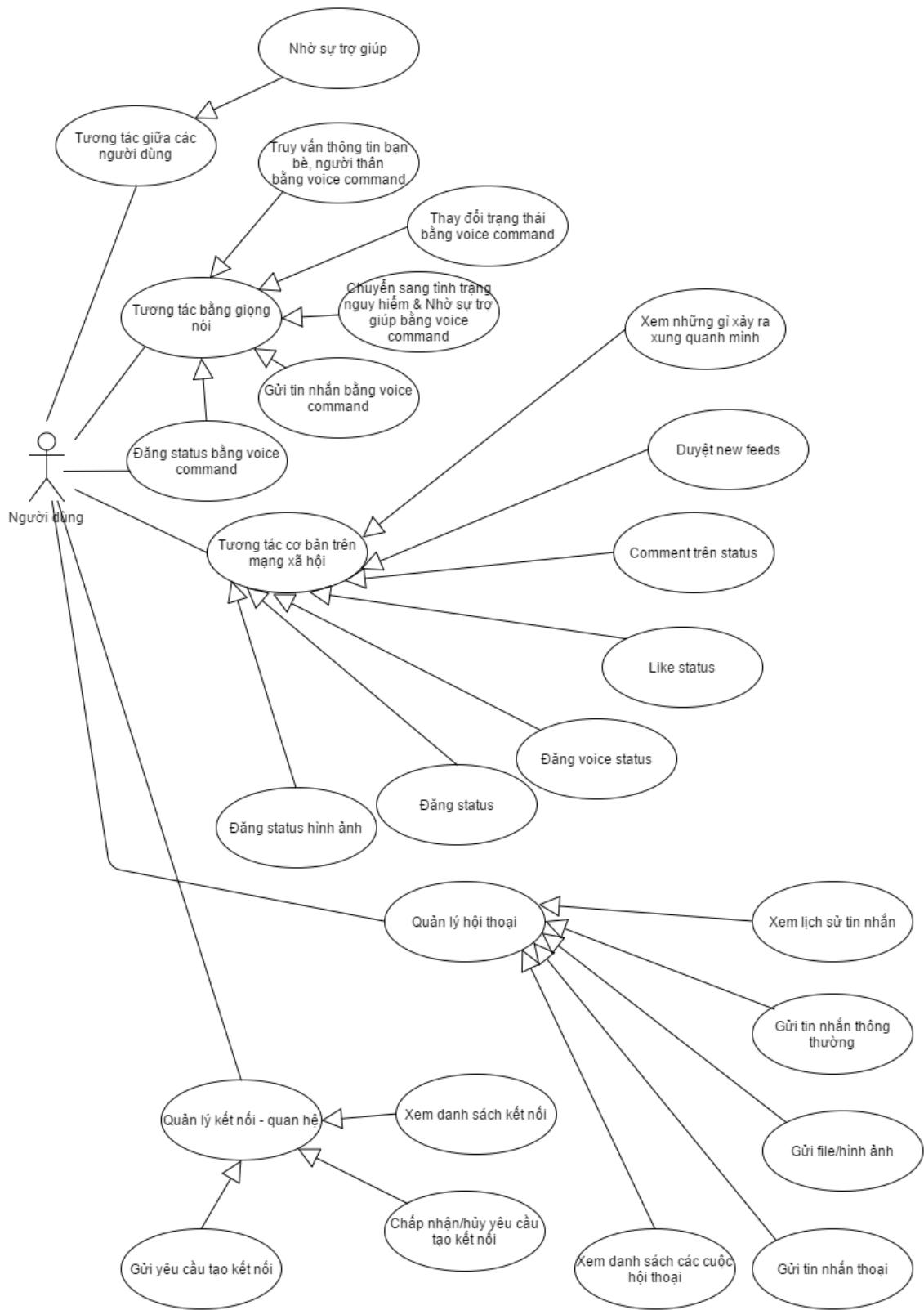
### 4.2.1. Danh sách yêu cầu

STT	Yêu cầu
1	Xem danh sách kết nối
2	Gửi yêu cầu tạo kết nối
3	Chấp nhận/hủy yêu cầu tạo kết nối
4	Xem danh sách các cuộc hội thoại
5	Xem lịch sử tin nhắn
6	Gửi tin nhắn thông thường
7	Gửi file/hình ảnh
8	Gửi tin nhắn thoại
9	Đăng status hình ảnh
10	Đăng status
11	Đăng voice status

12	Like status
13	Comment trên status
14	Duyệt news feed
15	Xem những gì xảy ra xung quanh mình
16	Đăng status bằng voice command
17	Gửi tin nhắn bằng voice command
18	Chuyển sang tình trạng nguy hiểm & nhờ sự trợ giúp bằng voice command
19	Thay đổi trạng thái bằng voice command
20	Truy vấn thông tin bạn bè, người thân bằng voice command
21	Nhờ sự trợ giúp

Bảng 4.1 Danh sách các yêu cầu

#### **4.2.2. Mô hình use-case**



Hình 4.1 Mô hình use-case của ứng dụng

### 4.2.3. ĐẶC TẢ USE-CASE

#### 4.2.3.1. ĐẶC TẢ USE-CASE “XEM DANH SÁCH KẾT NỐI”

STT use-case	1
Tên use-case	Xem danh sách kết nối
Mô tả	Xem danh sách những người dùng đã kết nối với người dùng này
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<ol style="list-style-type: none"> <li>1. Người dùng mở ứng dụng và vào trang chính.</li> <li>2. Người dùng nhấn nút</li> <li>3. Ứng dụng gửi yêu cầu lấy danh sách bạn bè lên server.</li> <li>4. Server tìm và trả về danh sách bạn bè của người dùng.</li> <li>5. Ứng dụng hiển thị danh sách bạn bè của người dùng kèm theo trạng thái online hiện tại.</li> </ol>
Dòng sự kiện thay thế	4b. Nếu người dùng không có bạn bè nào: <ol style="list-style-type: none"> <li>1. Ứng dụng hiển thị thông báo “Bạn không có bạn bè nào”.</li> </ol>
Điều kiện sau khi thực hiện	Không có

Bảng 4.2 Đặc tả use-case “Xem danh sách kết nối”

#### 4.2.3.2. ĐẶC TẢ USE-CASE “GỬI YÊU CẦU TẠO KẾT NỐI”

STT use-case	2
Tên use-case	Gửi yêu cầu tạo kết nối
Mô tả	Gửi yêu cầu tạo kết nối giữa 2 người dùng với nhau
Actor	Người sử dụng

Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Người dùng đã vào trang profile của đối tượng cần kết bạn. (Sau khi thực hiện use-case “Xem profile của bạn bè”).</li> </ol>
Dòng sự kiện chính	<p>Khi người dùng đang ở trang Profile:</p> <ol style="list-style-type: none"> <li>1. Người dùng nhấn nút “Add Friend”.</li> <li>2. Ứng dụng gửi request “Tạo kết nối” lên server.</li> <li>3. Server kiểm tra giữa 2 người đã có kết nối chưa.</li> <li>4. Server lưu trữ thông tin yêu cầu tạo kết nối vào CSDL.</li> <li>5. Server trả về thông tin “Kết nối”.</li> <li>6. Ứng dụng hiển thị thông báo “Gửi yêu cầu kết bạn thành công”.</li> </ol>
Dòng sự kiện thay thế	<p>3b. Khi giữa 2 người đã có kết nối (Trong trường hợp người dùng đăng nhập cùng một tài khoản trên 2 máy khác nhau, và thực hiện use-case này tại cùng một thời điểm):</p> <ol style="list-style-type: none"> <li>1. Server trả về lỗi “connection_request_already_existed”.</li> <li>2. Ứng dụng hiển thị thông báo “Bạn đã kết nối với người này”.</li> </ol>
Điều kiện sau khi thực hiện	Dữ liệu về yêu cầu tạo “kết nối” được lưu trữ trong CSDL.

Bảng 4.3 Đặc tả use-case “Gửi yêu cầu tạo kết nối”

#### 4.2.3.3. Đặc tả use-case “Chấp nhận/hủy yêu cầu tạo kết nối”

STT use-case	3
Tên use-case	Chấp nhận/Hủy yêu cầu tạo kết nối
Mô tả	Chấp nhận yêu cầu tạo kết nối
Actor	Người sử dụng

Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Người dùng đã nhận được yêu cầu tạo kết nối bởi một người dùng khác.</li> </ol>
Dòng sự kiện chính	<ol style="list-style-type: none"> <li>1. Người dùng vào trang chính của ứng dụng.</li> <li>2. Người dùng chọn tab “Connection requests”.</li> <li>3. Người dùng lựa chọn Chấp Nhận hoặc Hủy đối với yêu cầu kết bạn.</li> <li>4. Ứng dụng gửi request Chấp Nhận/Hủy yêu cầu kết bạn lên server.</li> <li>5. Server kiểm tra yêu cầu kết bạn đó còn tồn tại không.</li> <li>6. Server xử lý request và cập nhật dữ liệu vào CSDL.</li> <li>7. Server trả về response.</li> <li>8. Ứng dụng hiển thị thông báo “Bạn đã Chấp nhận/Hủy yêu cầu kết bạn thành công”.</li> </ol>
Dòng sự kiện thay thế	<p>5b. Khi yêu cầu kết bạn đó không còn tồn tại (Trong trường hợp người dùng sử 2 máy khác nhau để đăng nhập cùng 1 tài khoản và thực hiện use-case này tại cùng 1 thời điểm)</p> <ol style="list-style-type: none"> <li>1. Server trả về lỗi “connection_request_not_existed”.</li> <li>2. Ứng dụng hiển thị thông báo “Yêu cầu tạo kết nối này không tồn tại”.</li> </ol>
Điều kiện sau khi thực hiện	Trạng thái của “yêu cầu tạo kết nối” được chuyển sang <u>đã_chấp_nhận/hủy</u> . Một record “connection” mới được tạo ra trong CSDL dựa trên “yêu cầu tạo kết nối” này nếu người dùng lựa chọn “Chấp nhận”.

Bảng 4.4 Đặc tả use-case “Chấp nhận/hủy yêu cầu tạo kết nối”

#### 4.2.3.4. Đặc tả use-case “Xem danh sách các cuộc hội thoại”

STT use-case	4
Tên use-case	Xem danh sách tất cả các cuộc hội thoại
Mô tả	Xem danh sách tất cả các cuộc hội thoại với những người mà người dùng từng chat, ứng với mỗi cuộc hội thoại sẽ hiển thị tin nhắn cuối cùng và thời điểm nhận tin nhắn cuối cùng của cuộc hội thoại đó.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>Người dùng đã đăng nhập vào ứng dụng.</li> <li>Ứng dụng kết nối được tới internet.</li> <li>Ứng dụng đã tạo kết nối tcp với Chat Server.</li> </ol>
Dòng sự kiện chính	<ol style="list-style-type: none"> <li>Người dùng vào trang chính của ứng dụng.</li> <li>Người dùng chọn tab “Messages”.</li> <li>Ứng dụng gửi request lên server để lấy danh sách tất cả các cuộc hội thoại.</li> <li>Server truy xuất CSDL để lấy dữ liệu cần thiết.</li> <li>Server trả về danh sách tất cả các cuộc hội thoại của người dùng.</li> <li>Ứng dụng gửi tcp request lên Chat Server để lấy danh sách các tin nhắn cuối cùng của các cuộc hội thoại.</li> <li>Ứng dụng hiển thị danh sách tất cả các cuộc hội thoại của người dùng kèm theo tin nhắn cuối cùng và thời điểm của tin nhắn cuối cùng.</li> </ol>
Dòng sự kiện thay thế	<p>5b. Nếu danh sách các cuộc hội thoại trả về từ server là rỗng:</p> <ol style="list-style-type: none"> <li>Ứng dụng hiển thị “Bạn không có bất kỳ cuộc hội thoại nào”.</li> </ol>
Điều kiện sau khi thực hiện	Không có

Bảng 4.5 Đặc tả use-case “Xem danh sách các cuộc hội thoại”

#### 4.2.3.5. Đặc tả use-case “Xem lịch sử tin nhắn”

STT use-case	5
Tên use-case	Xem lịch sử tin nhắn
Mô tả	Xem lịch sử các tin nhắn giữa 2 người dùng
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Ứng dụng đã tạo kết nối tcp với Chat Server (Được thực hiện ngay khi mở ứng dụng).</li> <li>4. Người dùng từng có ít nhất một cuộc hội thoại với người dùng khác.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi người dùng đang ở trong trang “Danh sách lịch sử tất cả các cuộc hội thoại”:</p> <ol style="list-style-type: none"> <li>1. Người dùng nhấn chọn một cuộc hội thoại mình muốn xem lịch sử tin nhắn.</li> <li>2. Ứng dụng gửi request lên chat server để lấy lịch sử tin nhắn.</li> <li>3. Khi nhận được lịch sử tin nhắn, ứng dụng hiển thị lịch sử tin nhắn lên giao diện.</li> </ol>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.6 Đặc tả use-case “Xem lịch sử tin nhắn”

#### 4.2.3.6. Đặc tả use-case “Gửi tin nhắn thông thường”

STT use-case	6
Tên use-case	Gửi tin nhắn thông thường
Mô tả	Gửi tin nhắn dưới dạng text thông thường

Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Ứng dụng đã tạo kết nối tcp với Chat Server (Được thực hiện ngay khi mở ứng dụng).</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi người dùng đang ở trong trang “Hội thoại”:</p> <ol style="list-style-type: none"> <li>1. Người dùng nhập nội dung tin nhắn cần gửi.</li> <li>2. Người dùng nhấn nút gửi.</li> <li>3. Ứng dụng gửi request tạo tin nhắn mới trong cuộc hội thoại hiện tại lên server.</li> <li>4. Ứng dụng đưa tin nhắn mới vừa gửi vào danh sách các tin nhắn của cuộc hội thoại, và đặt nó ở trạng thái “Đang gửi”.</li> <li>5. Ứng dụng chờ tín hiệu đã nhận được request từ server.</li> <li>6. Ứng dụng chuyển trạng thái tin nhắn vừa gửi trong “danh sách các tin nhắn của cuộc hội thoại” sang trạng thái đã gửi.</li> </ol>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.7 Đặc tả use-case “Gửi tin nhắn thông thường”

#### 4.2.3.7. Đặc tả use-case “Gửi file/hình ảnh”

STT use-case	7
Tên use-case	Gửi tin nhắn file/hình ảnh
Mô tả	Gửi tin nhắn dưới dạng file hoặc hình ảnh
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Ứng dụng đã tạo kết nối tcp với Chat Server (Được thực hiện ngay khi mở ứng dụng).</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi người dùng đang ở trong trang “Hội thoại”:</p> <ol style="list-style-type: none"> <li>1. Người nhấn nút gửi file/hình ảnh.</li> <li>2. Người dùng chọn file/hình ảnh cần gửi.</li> <li>3. Người dùng nhấn gửi.</li> <li>4. Ứng dụng encode binary của file/hình ảnh thành base64 rồi request API upload file lên server.</li> <li>5. Server convert dữ liệu vừa nhận được thành binary và request upload file lên Amazon S3.</li> <li>6. Server trả về link của file vừa mới upload (Đường dẫn tới file đó trên Amazon S3).</li> <li>7. Ứng dụng gửi request tạo tin nhắn file/hình ảnh kèm theo đường link của file/hình ảnh vừa upload tới Chat Server.</li> <li>8. Ứng dụng đưa tin nhắn file/hình ảnh mới vừa gửi vào danh sách các tin nhắn của cuộc hội thoại, và đặt nó ở trạng thái “Đang gửi”.</li> <li>9. Ứng dụng chờ tín hiệu đã nhận được request từ server.</li> <li>10. Ứng dụng chuyển trạng thái tin nhắn vừa gửi trong “danh sách các tin nhắn của cuộc hội thoại” sang trạng thái đã gửi.</li> </ol>

Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.8 Đặc tả use-case “Gửi file/hình ảnh”

#### 4.2.3.8. Đặc tả use-case “Gửi tin nhắn thoại”

STT use-case	8
Tên use-case	Gửi tin nhắn thoại
Mô tả	Gửi tin nhắn thoại của người dùng
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> <li>3. Ứng dụng đã tạo kết nối tcp với Chat Server (Được thực hiện ngay khi mở ứng dụng).</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi người dùng đang ở trong trang “Hội thoại”:</p> <ol style="list-style-type: none"> <li>1. Người nhắn vào biểu tượng tạo tin nhắn thoại.</li> <li>2. Ứng dụng thông báo cho người dùng đọc nội dung cần gửi.</li> <li>3. Ứng dụng ghi âm lại và encode thành một file audio.</li> <li>4. Ứng dụng encode binary của audio đó thành base64 rồi request API upload file lên server.</li> <li>5. Server convert dữ liệu vừa nhận được thành binary và request upload file lên Amazon S3.</li> <li>6. Server trả về link của file vừa mới upload (Đường dẫn tới file đó trên Amazon S3).</li> <li>7. Ứng dụng gửi request tạo tin nhắn thoại kèm theo đường link của file audio vừa upload lên Chat Server.</li> </ol>

	<p>8. Ứng dụng đưa tin nhắn thoại mới vừa gửi vào danh sách các tin nhắn của cuộc hội thoại, và đặt nó ở trạng thái “Đang gửi”.</p> <p>9. Ứng dụng chờ tín hiệu đã nhận được request từ server.</p> <p>10. Ứng dụng chuyển trạng thái tin nhắn vừa gửi trong “danh sách các tin nhắn của cuộc hội thoại” sang trạng thái đã gửi.</p>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.9 Đặc tả use-case “Gửi tin nhắn thoại”

#### 4.2.3.9. Đặc tả use-case “Đăng status hình ảnh”

STT use-case	9
Tên use-case	Đăng status hình ảnh
Mô tả	Người dùng đăng hình ảnh lên timeline của mình.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang chạy ở trang chia sẻ trạng thái:</p> <p>1. Người dùng chọn hình ảnh và nhấn đăng ảnh</p> <p>2. Ứng dụng encode binary của file/hình ảnh thành base64 rồi request API upload file lên server.</p> <p>3. Server convert dữ liệu vừa nhận được thành binary và request upload file lên Amazon S3.</p> <p>4. Server trả về link của file vừa mới upload (Đường dẫn tới file đó trên Amazon S3).</p>

	5. Ứng dụng gửi request đăng status hình ảnh 6. Ứng dụng hiển thị thông báo đã đăng ảnh thành công
Dòng sự kiện thay thế	2b. Nếu người dùng chọn ảnh nhưng không nhấn đăng 1. Ứng dụng thông báo status bị hủy
Điều kiện sau khi thực hiện	Không có

Bảng 4.10 Đặc tả use-case “Đăng status hình ảnh”

#### 4.2.3.10. Đặc tả use-case “Đăng status”

STT use-case	10
Tên use-case	Đăng status
Mô tả	Người dùng đăng status với nội dung là một đoạn text lên timeline của mình.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	1. Người dùng đã đăng nhập vào ứng dụng. 2. Ứng dụng kết nối được tới internet.
Dòng sự kiện chính	Use-case có thể bắt đầu khi ứng dụng đang chạy ở trang chia sẻ trạng thái: 1. Người dùng nhập trạng thái và nhấn chia sẻ 2. Ứng dụng gửi request đăng status lên server 3. Status vừa gửi được chuyển sang trạng thái đang chia sẻ 4. Ứng dụng chờ tín hiệu đã nhận được request từ server. 5. Ứng dụng chuyển trạng thái status vừa chia sẻ sang trạng thái đã chia sẻ. 6. Ứng dụng hiển thị thông báo đã đăng status thành công
Dòng sự kiện thay thế	Không có

Điều kiện sau khi thực hiện	Không có
-----------------------------	----------

Bảng 4.11 Đặc tả use-case “Đăng status”

#### 4.2.3.11. Đặc tả use-case “Đăng voice status”

STT use-case	11
Tên use-case	Đăng voice status
Mô tả	Người dùng đăng status bằng giọng nói của mình lên timeline.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ul style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ul>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi người dùng đang ở chia sẻ trạng thái:</p> <ul style="list-style-type: none"> <li>1. Người nhấn vào biểu tượng tạo voice status.</li> <li>2. Ứng dụng thông báo cho người dùng đọc nội dung cần chia sẻ.</li> <li>3. Ứng dụng ghi âm lại và encode thành một file audio.</li> <li>4. Ứng dụng encode binary của audio đó thành base64 rồi request API upload file lên server.</li> <li>5. Server convert dữ liệu vừa nhận được thành binary và request upload file lên Amazon S3.</li> <li>6. Server trả về link của file vừa mới upload (Đường dẫn tới file đó trên Amazon S3).</li> <li>7. Ứng dụng gửi request tạo voice status kèm theo đường link của file audio vừa upload lên server.</li> <li>8. Ứng dụng đặt voice status mới vừa chia sẻ ở trạng thái “Đang chia sẻ”.</li> <li>9. Ứng dụng chờ tín hiệu đã nhận được request từ server.</li> </ul>

	10. Ứng dụng chuyển trạng thái voice status vừa chia sẻ sang trạng thái đã chia sẻ.
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.12 Đặc tả use-case “Đăng voice status”

#### 4.2.3.12. Đặc tả use-case “Like status”

STT use-case	12
Tên use-case	Like status
Mô tả	Người dùng thực hiện like một status nào đó trên news feed của mình.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>Người dùng đã đăng nhập vào ứng dụng.</li> <li>Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang ở trang News feed hoặc Comments:</p> <ol style="list-style-type: none"> <li>Người dùng nhấn like của một status</li> <li>Ứng dụng gửi request update like lên server</li> <li>Ứng dụng cập nhật trạng thái thành đã like và tăng số lượng like của cập nhật trạng thái</li> </ol>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.13 Đặc tả use-case “Like status”

#### 4.2.3.13. Đặc tả use-case “Comment trên status”

STT use-case	13
Tên use-case	Comments trên status
Mô tả	Người dùng thực hiện comment lên một status trên news feed.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang ở trang Comments:</p> <ol style="list-style-type: none"> <li>1. Người dùng nhập nội dung comment cho status và nhấn Post comment</li> <li>2. Ứng dụng gửi request post comment lên server</li> <li>3. Ứng dụng hiển thị comment của người dùng trong danh sách comment</li> </ol>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.14 Đặc tả use-case “Comment trên status”

#### 4.2.3.14. Đặc tả use-case “Duyệt news feed”

STT use-case	14
Tên use-case	Duyệt news feed
Mô tả	Người dùng duyệt qua các cập nhật trạng thái của bạn bè và người thân của mình trên news feed.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang ở trang News feed:</p>

	<ol style="list-style-type: none"> <li>1. Khi người dùng mở ứng dụng, trang news feed sẽ hiện ra ở trang chủ</li> <li>2. Ứng dụng gửi request get newsfeed lên server</li> <li>3. Server xử lý và lấy dữ liệu new feeds của người dùng.</li> <li>4. Server phản hồi trả về danh sách news feed</li> <li>5. Ứng dụng hiển thị các cập nhật trạng thái trong news feed.</li> </ol>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.15 Đặc tả use-case “Duyệt news feed”

#### 4.2.3.15. Đặc tả use-case “Xem những gì xảy ra xung quanh mình”

STT use-case	15
Tên use-case	Xem những gì xảy ra xung quanh mình
Mô tả	Chức năng cho phép người dùng xem những câu chuyện, tin tức, cập nhật trạng thái xung quanh vị trí hiện tại của người dùng
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang ở trang đầu của ứng dụng:</p> <ol style="list-style-type: none"> <li>1. Người dùng chọn chức năng “Những gì xảy ra xung quanh bạn” trong menu.</li> <li>2. Ứng dụng gửi request kèm theo thông tin vị trí hiện tại của người dùng để lấy các cập nhật trạng thái được đăng xung quanh vị trí hiện tại của người dùng.</li> </ol>

	<p>3. Server xử lý và lấy các cập nhật trạng thái được tạo ra xung quanh vị trí của người dùng.</p> <p>4. Server phản hồi trả về danh sách cập nhật trạng thái.</p> <p>5. Ứng dụng hiển thị các danh sách các cập nhật trạng thái được đăng xung quanh vị trí hiện tại của người dùng.</p>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.16 Đặc tả use-case “Xem những gì xảy ra xung quanh mình”

#### 4.2.3.16. Đặc tả use-case “Đăng status bằng voice command”

STT use-case	16
Tên use-case	Đăng status bằng voice command
Mô tả	Người dùng cập nhật trạng thái của mình bằng voice command
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang chạy ở bất kỳ trang nào:</p> <p>1. Người dùng nhấn nút “Assistive Touch” hoặc đọc lệnh “turn on command” (hay lệnh do người dùng tự xét) để kích hoạt voice command</p> <p>2. Người dùng ra lệnh yêu cầu “new status” bằng giọng nói.</p> <p>3. Ứng dụng yêu cầu người dùng đọc status muốn đăng.</p> <p>4. Ứng dụng xác nhận người dùng có muốn đăng status hay không.</p>

	<p>5. Status sẽ được đăng theo use case “Đăng status” nếu người dùng xác nhận muốn đăng status.</p> <p>6. Ứng dụng thông báo status đã đăng thành công.</p>
Dòng sự kiện thay thế	<p>5b. Nếu người dùng xác nhận không muốn đăng status</p> <p>1. Ứng dụng thông báo status đã bị hủy</p>
Điều kiện sau khi thực hiện	Không có

Bảng 4.17 Đặc tả use-case “Đăng status bằng voice command”

#### 4.2.3.17. Đặc tả use-case “Gửi tin nhắn bằng voice command”

STT use-case	17
Tên use-case	Gửi tin nhắn bằng voice command
Mô tả	Người dùng gửi tin nhắn cho bạn bè hoặc người thân của mình thông qua voice command
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p> <p>3. Ứng dụng đã tạo kết nối tcp với Chat Server (Được thực hiện ngay khi mở ứng dụng).</p>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang ở trang chi tiết tin nhắn hoặc trang khác:</p> <p>1. Người dùng nhấn nút “Assistive Touch” hoặc đọc lệnh “turn on command” (hay lệnh do người dùng tự xét) để kích hoạt voice command</p> <p>2. Người dùng ra lệnh yêu cầu tạo tin nhắn mới bằng giọng nói.</p> <p>3. Ứng dụng yêu cầu người dùng đọc tin nhắn muốn gửi và đối tượng người dùng sẽ nhận.</p>

	<p>4. Ứng dụng xác nhận người dùng có muốn gửi tin nhắn hay không.</p> <p>5. Tin nhắn sẽ được gửi nếu người dùng xác nhận muốn gửi.</p> <p>6. Ứng dụng thông báo tin nhắn đã gửi thành công.</p>
Dòng sự kiện thay thế	<p>5b. Nếu người dùng xác nhận không muốn gửi tin nhắn</p> <p>1. Ứng dụng thông báo tin nhắn đã bị hủy</p>
Điều kiện sau khi thực hiện	Không có

Bảng 4.18 Đặc tả use-case “Gửi tin nhắn bằng voice command”

#### 4.2.3.18. Đặc tả use-case “Chuyển sang tình trạng nguy hiểm & nhờ sự trợ giúp bằng voice command”

STT use-case	18
Tên use-case	Chuyển sang tình trạng nguy hiểm và nhờ sự trợ giúp bằng voice command
Mô tả	Tính năng này nhằm giúp người dùng thông báo đến người thân và nhờ sự giúp đỡ của người thân khi người dùng đang trong tình trạng nguy hiểm và cần sự giúp đỡ.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang chạy ở bất kỳ trang nào:</p> <p>1. Người dùng nhấn nút “Assistive Touch” hoặc đọc lệnh “turn on command” (hay lệnh do người dùng tự xét) để kích hoạt voice command</p> <p>2. Người dùng ra lệnh yêu cầu cập nhật chuyển sang tình trạng nguy hiểm và nhờ sự giúp đỡ.</p>

	<p>3. Ứng dụng gửi request lên server yêu cầu chuyển tình trạng của người dùng sang “đang trong tình trạng nguy hiểm”.</p> <p>4. Server chuyển trạng thái của người dùng sang “đang trong tình trạng nguy hiểm”.</p> <p>5. Server gửi thông tin yêu cầu sự hỗ trợ đến tất cả những kết nối thân thiết (những người trong danh sách kết nối có quan hệ bạn thân hoặc họ hàng).</p> <p>6. Ứng dụng trạng thái người dùng sang “Đang trong tình trạng nguy hiểm” và liên tục gửi tọa độ hiện tại của người dùng lên server (30s một lần).</p>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.19 Đặc tả use-case “Chuyển sang tình trạng nguy hiểm & nhờ sự trợ giúp bằng voice command”

#### 4.2.3.19. Đặc tả use-case “Thay đổi trạng thái bằng voice command”

STT use-case	19
Tên use-case	Thay đổi trạng thái bằng voice command
Mô tả	
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p>
Dòng sự kiện chính	Use-case có thể bắt đầu khi ứng dụng đang chạy ở bất kỳ trang nào:

	<ol style="list-style-type: none"> <li>1. Người dùng nhấn nút AlwaysOnTop hoặc dùng lệnh “turn on command” (hay lệnh do người dùng tự xét) để kích hoạt voice command</li> <li>2. Người dùng ra lệnh yêu cầu thay đổi trạng thái.</li> <li>3. Ứng dụng xác nhận người dùng có muốn cập nhật thay đổi hay không.</li> <li>4. Ứng dụng cập nhật thay đổi trạng thái nếu người dùng xác nhận muốn cập nhật.</li> <li>5. Ứng dụng thông báo đã cập nhật thành công.</li> </ol>
Dòng sự kiện thay thế	<p>4b. Nếu người dùng xác nhận không muốn cập nhật</p> <ol style="list-style-type: none"> <li>1. Ứng dụng thông báo hủy cập nhật thay đổi trạng thái</li> </ol>
Điều kiện sau khi thực hiện	Không có

Bảng 4.20 Đặc tả use-case “Thay đổi trạng thái bằng voice command”

#### 4.2.3.20. Đặc tả use-case “Truy vấn thông tin bạn bè, người thân bằng voice command”

STT use-case	20
Tên use-case	Truy vấn thông tin bạn bè, người thân bằng voice command
Mô tả	
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<ol style="list-style-type: none"> <li>1. Người dùng đã đăng nhập vào ứng dụng.</li> <li>2. Ứng dụng kết nối được tới internet.</li> </ol>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang chạy ở bất kỳ trang nào:</p> <ol style="list-style-type: none"> <li>1. Người dùng nhấn nút AlwaysOnTop hoặc dùng lệnh “turn on command” (hay lệnh do người dùng tự xét) để kích hoạt voice command</li> </ol>

	<p>2. Người dùng ra lệnh yêu cầu xem, truy vấn thông tin bạn bè.</p> <p>3. Ứng dụng hiển thị thông tin người dùng muốn xem.</p>
Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

Bảng 4.21 Đặc tả use-case “Truy vấn thông tin bạn bè, người thân bằng voice command”

#### 4.2.3.21. Đặc tả use-case “Nhờ sự trợ giúp”

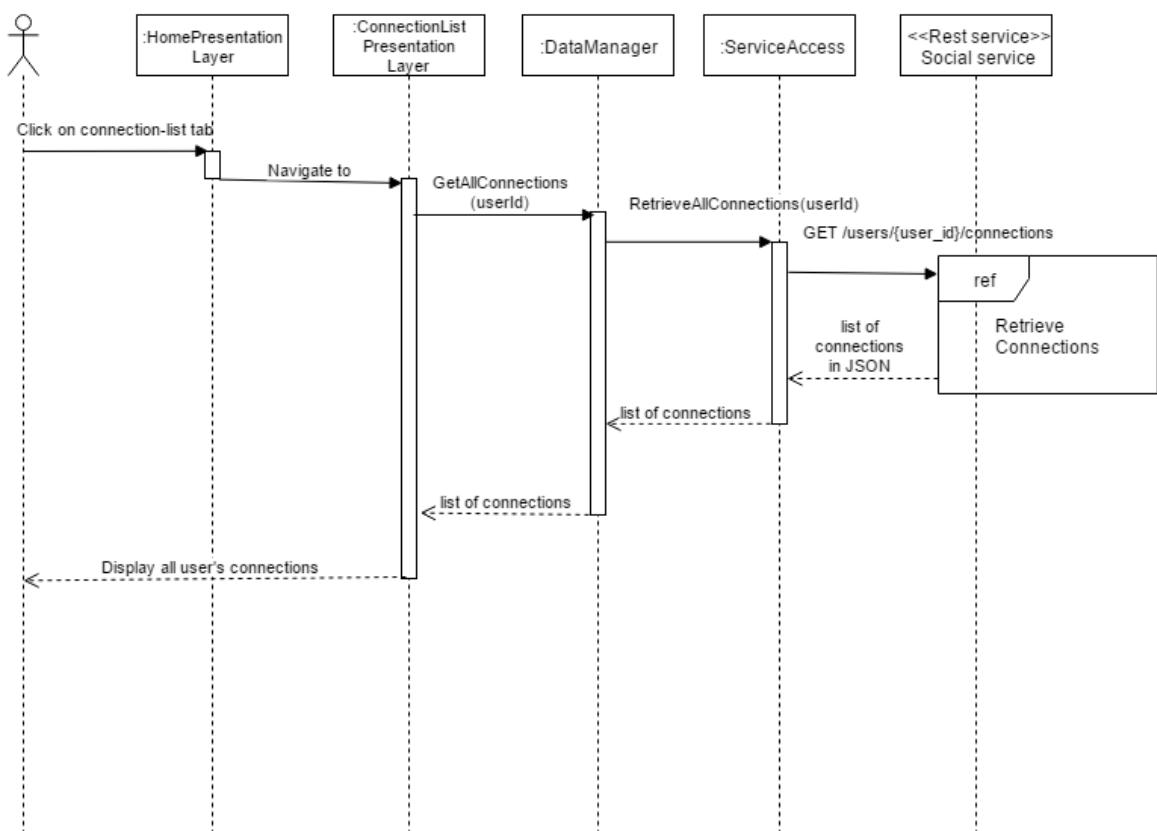
STT use-case	21
Tên use-case	Nhờ sự trợ giúp
Mô tả	Chức năng cho phép người dùng có thể nhờ sự hỗ trợ từ những người bạn hay người thân của mình khi gặp phải chuyện gì đó.
Actor	Người sử dụng
Điều kiện trước khi thực hiện	<p>1. Người dùng đã đăng nhập vào ứng dụng.</p> <p>2. Ứng dụng kết nối được tới internet.</p>
Dòng sự kiện chính	<p>Use-case có thể bắt đầu khi ứng dụng đang được mở:</p> <p>1. Người dùng nhấn vào “Assistive Touch” và lựa chọn tính năng “Nhờ sự trợ giúp” và lựa chọn mức độ “tình trạng nguy hiểm” mà người dùng đang gặp phải (Chia làm 3 mức độ: “Nhờ sự trợ giúp”, “Trong tình trạng nguy hiểm”, “Trong tình trạng nguy hiểm &amp; Cần sự trợ giúp”)</p> <p>2. Ứng dụng gửi request “Nhờ sự trợ giúp” lên server.</p> <p>3. Server gửi thông báo đến tất cả bạn bè và người thân của người dùng về để yêu cầu sự trợ giúp kèm theo vị trí hiện tại trên bảng đồ của người dùng.</p>

Dòng sự kiện thay thế	Không có
Điều kiện sau khi thực hiện	Không có

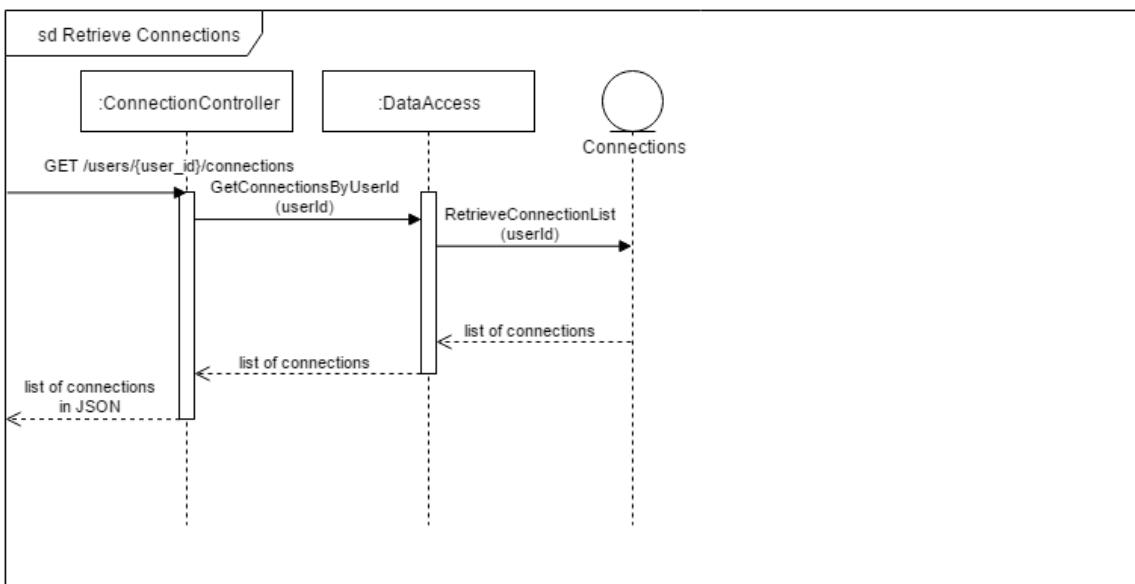
Bảng 4.22 Đặc tả use-case “Nhờ sự trợ giúp”

#### 4.2.4. Sequence diagram

##### 4.2.4.1. Sequence diagram cho use-case “Xem danh sách kết nối”

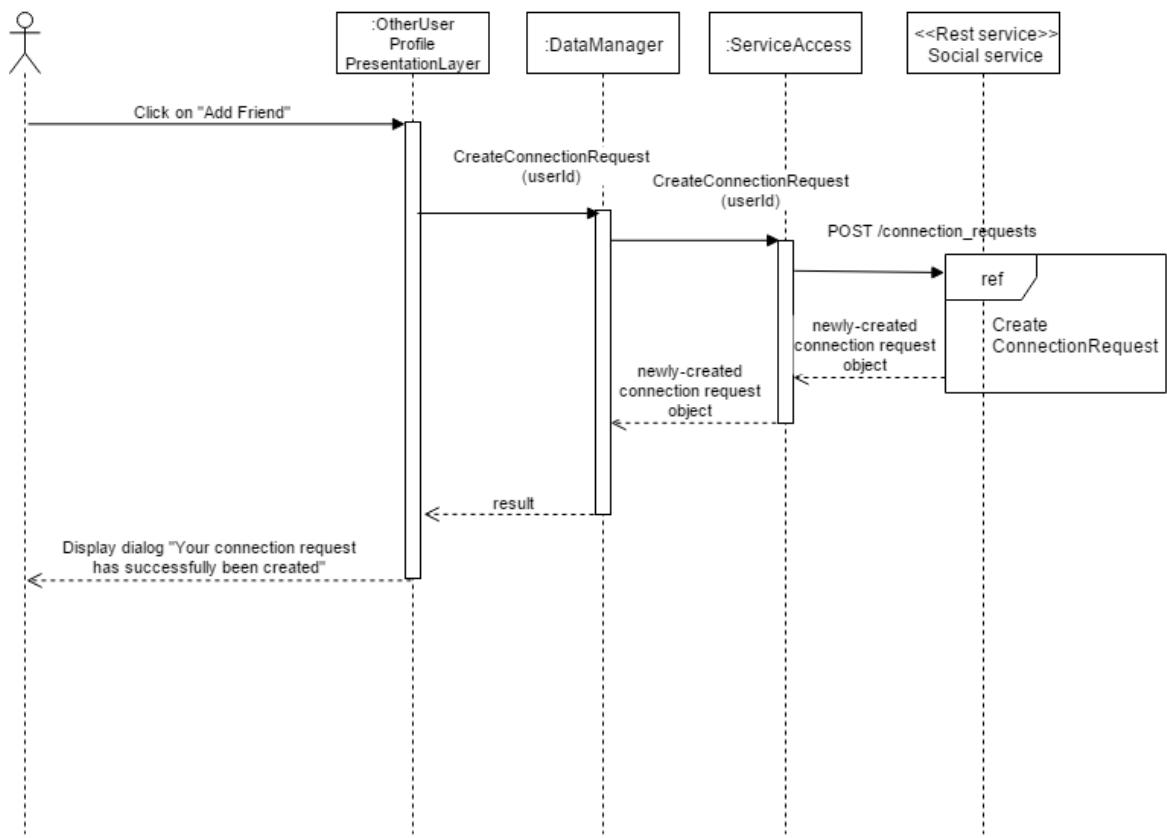


Hình 4.2 Sequence diagram “Xem danh sách kết nối”.

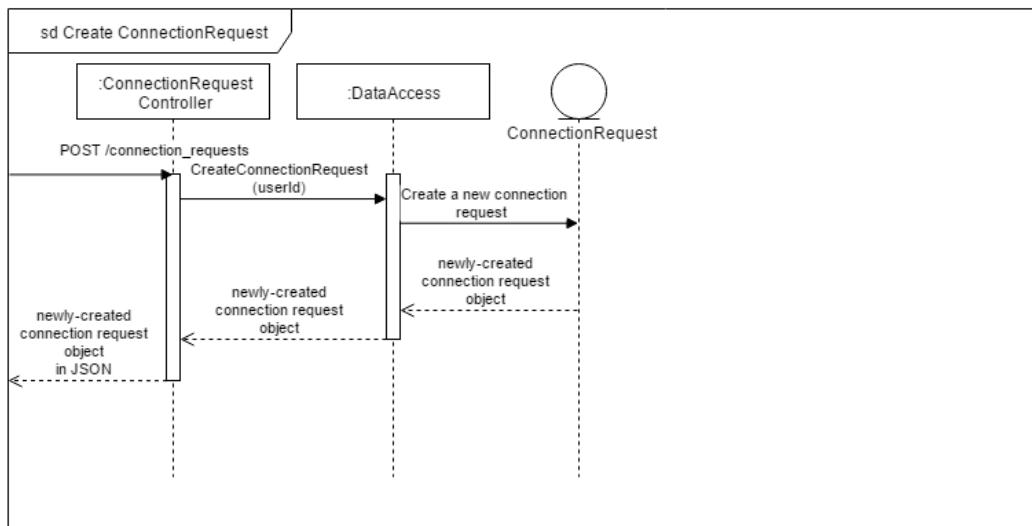


Hình 4.3 Sequence diagram “Retrieve connections” được sử dụng bởi sequence diagram “Xem danh sách kết nối”

#### 4.2.4.2. Sequence diagram cho use-case “Gửi yêu cầu tạo kết nối”

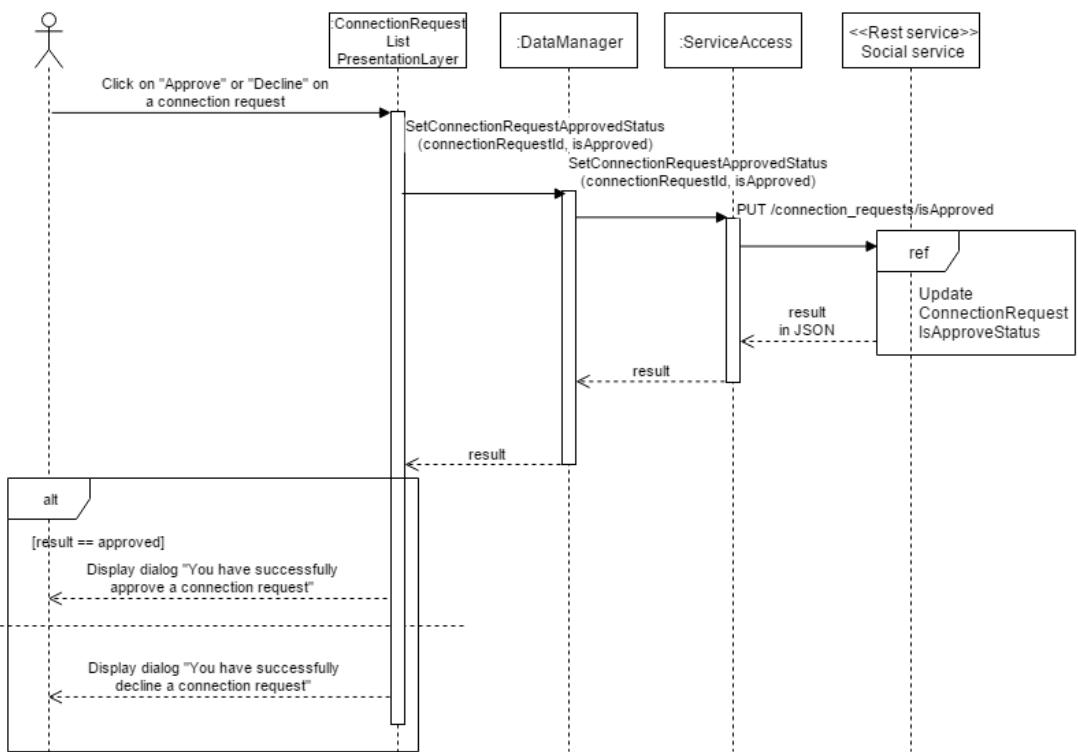


Hình 4.4 Sequence diagram “Gửi yêu cầu tạo kết nối”

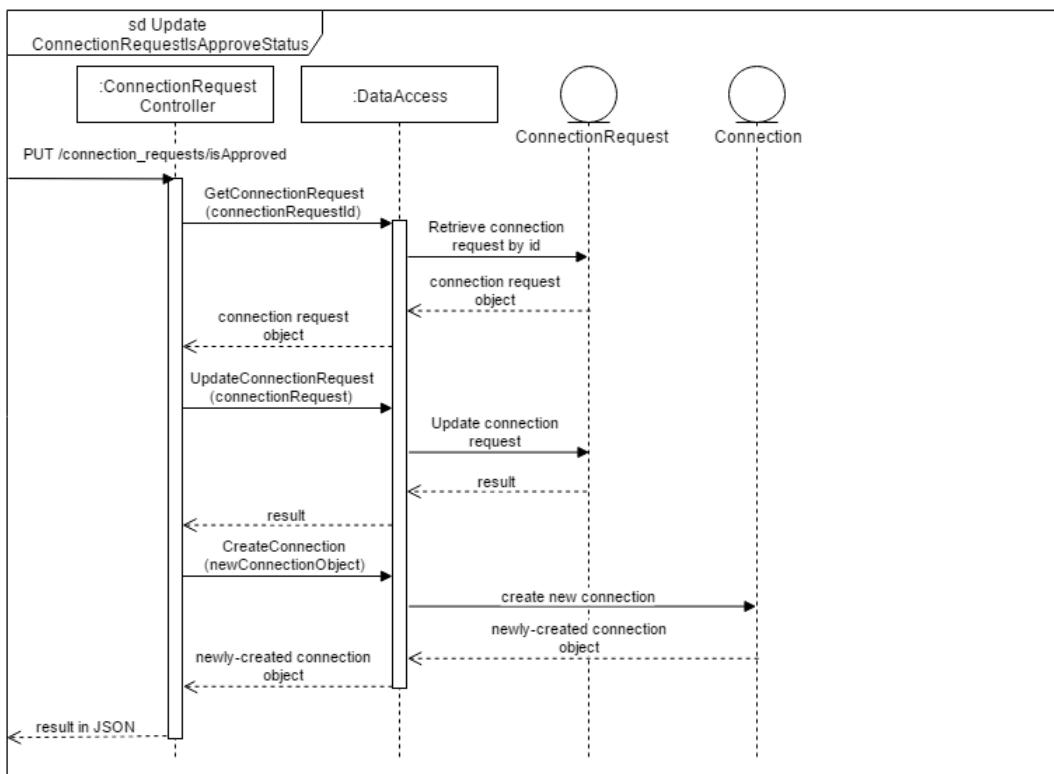


Hình 4.5 Sequence diagram “Create Connection Request” được sử dụng bởi sequence diagram “Gửi yêu cầu tạo kết nối”

#### 4.2.4.3. Sequence diagram cho use-case “Chấp nhận/hủy yêu cầu tạo kết nối”

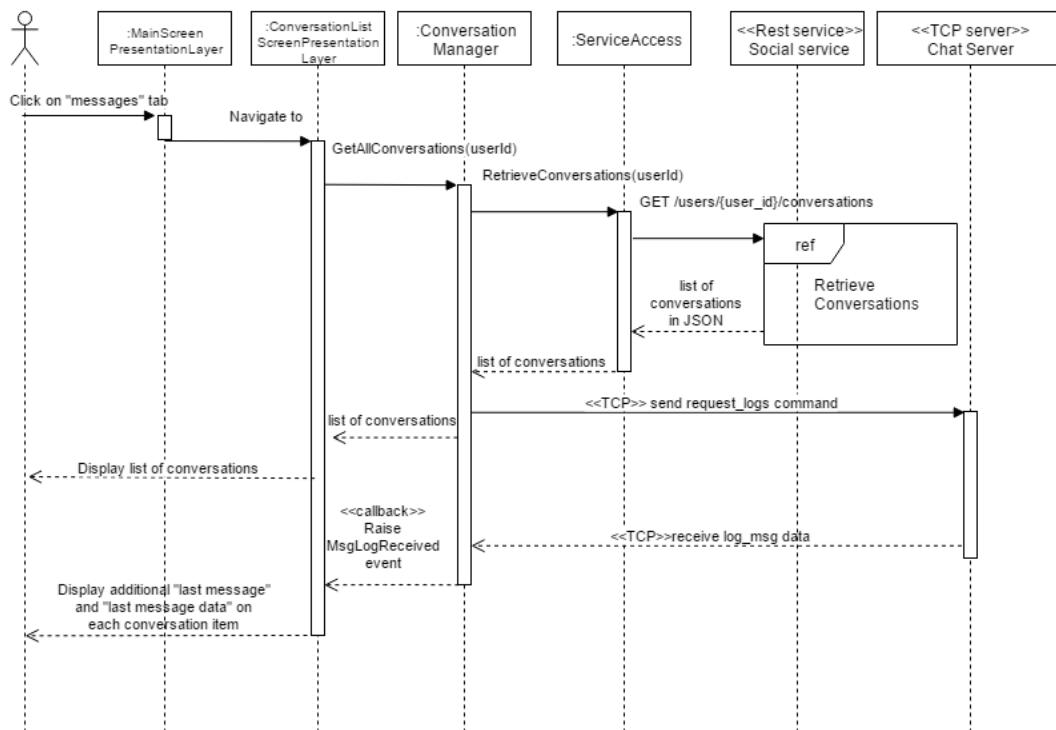


Hình 4.6 Sequence diagram “Chấp nhận/hủy yêu cầu tạo kết nối”

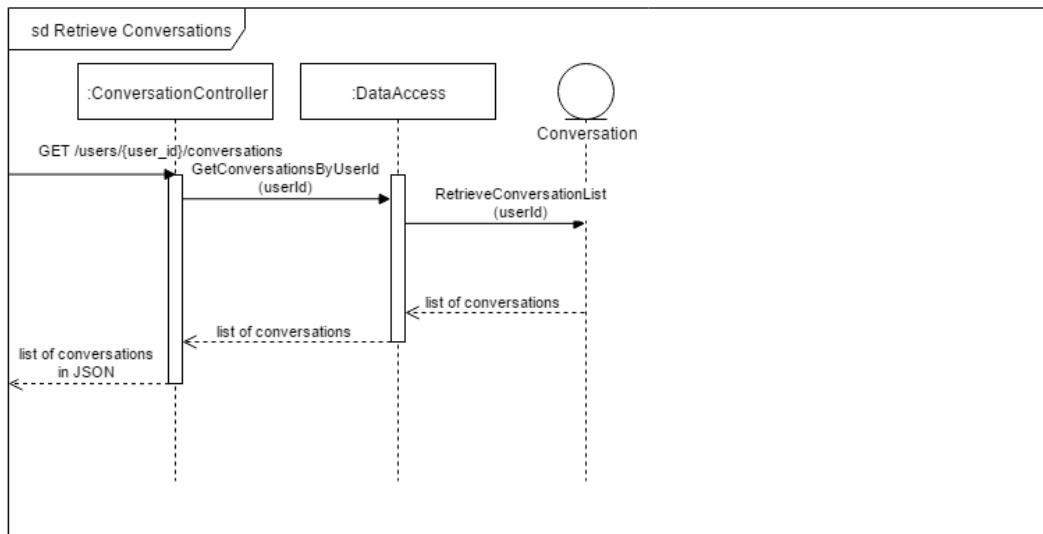


Hình 4.7 Sequence diagram “Update ConnectionRequest IsApproved Status” được sử dụng bởi sequence diagram “Chấp nhận/hủy yêu cầu tạo kết nối”

#### 4.2.4.4. Sequence diagram cho use-case “Xem danh sách các cuộc hội thoại”

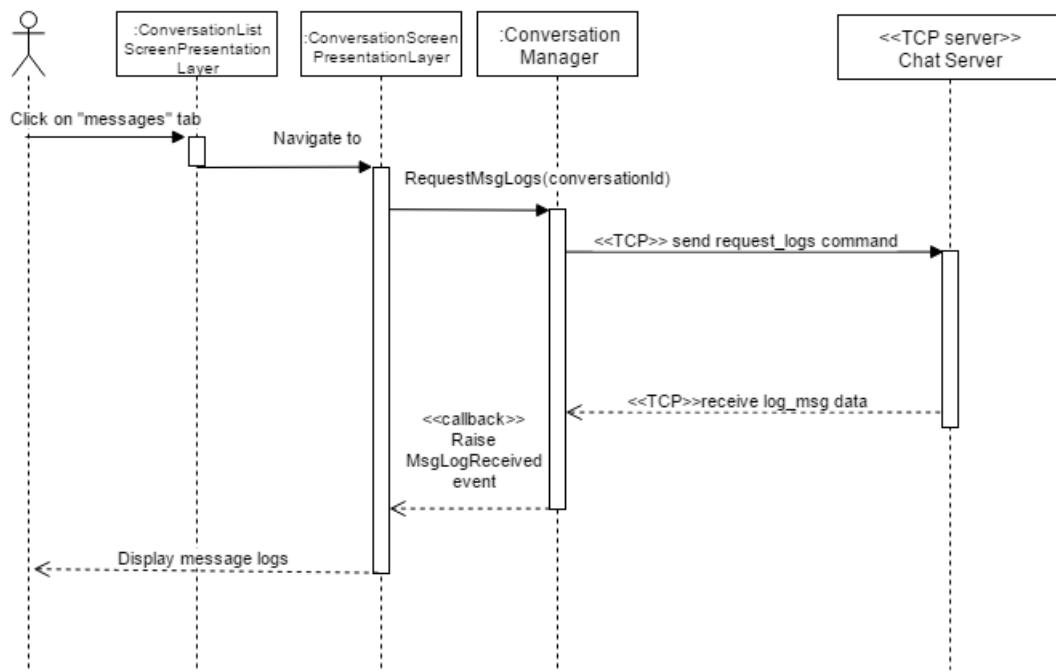


Hình 4.8 Sequence diagram “Xem danh sách các cuộc hội thoại”



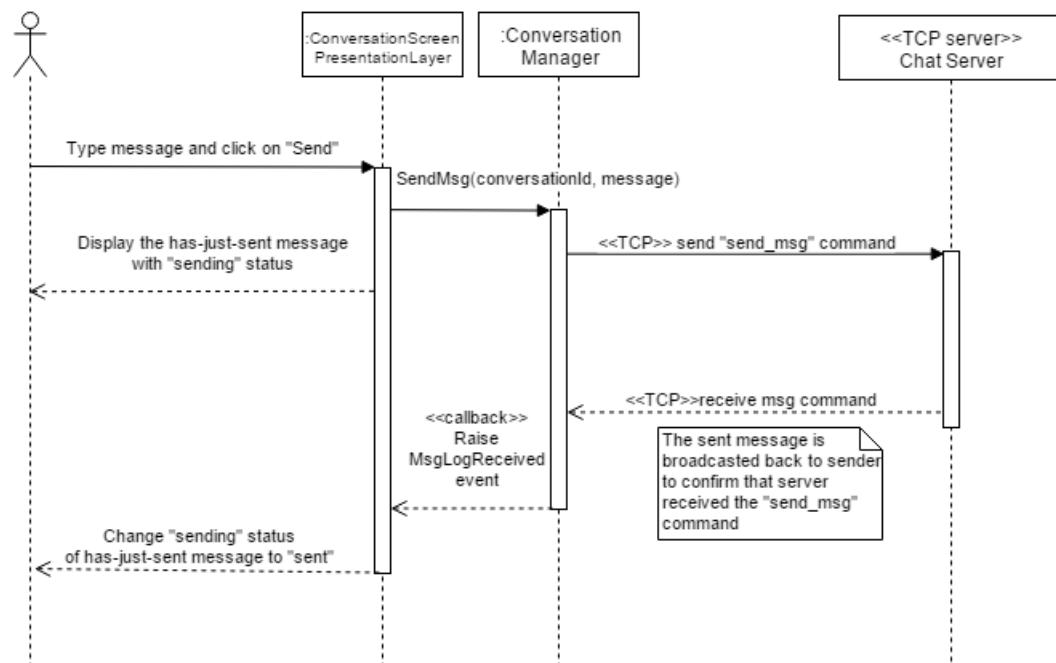
Hình 4.9 Sequence diagram “Retrieve connections” được sử dụng bởi sequence diagram “Xem danh sách các cuộc hội thoại”

#### 4.2.4.5. Sequence diagram cho use-case “Xem lịch sử tin nhắn”



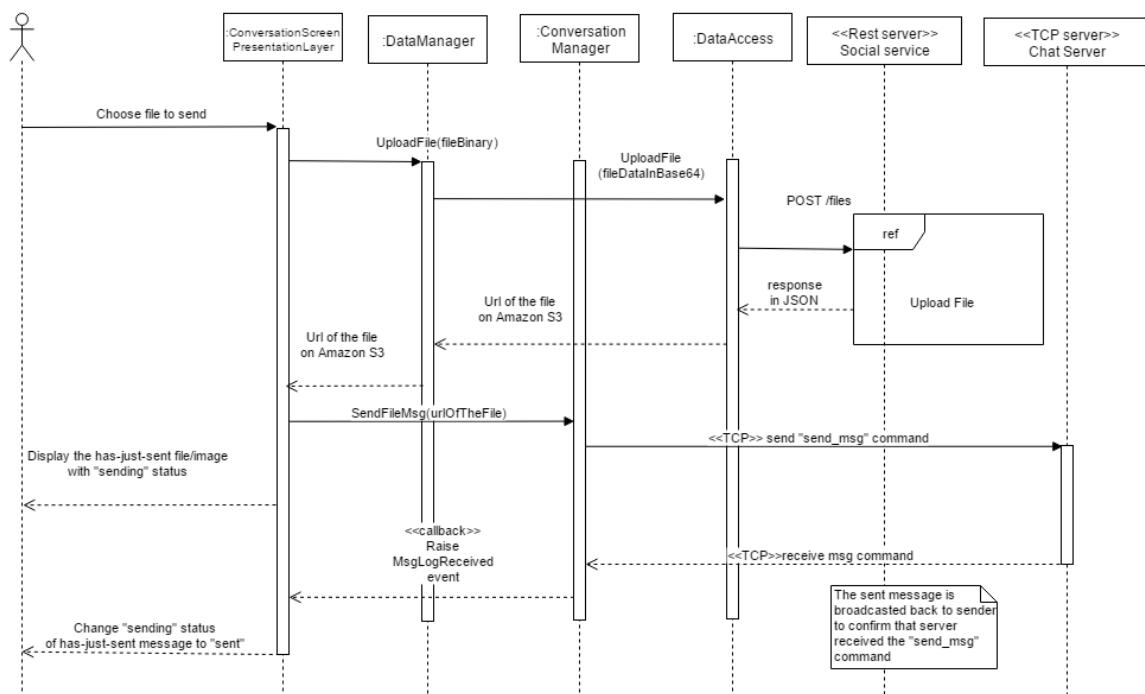
Hình 4.10 Sequence diagram “Xem lịch sử tin nhắn”

#### 4.2.4.6. Sequence diagram cho use-case “Gửi tin nhắn thông thường”

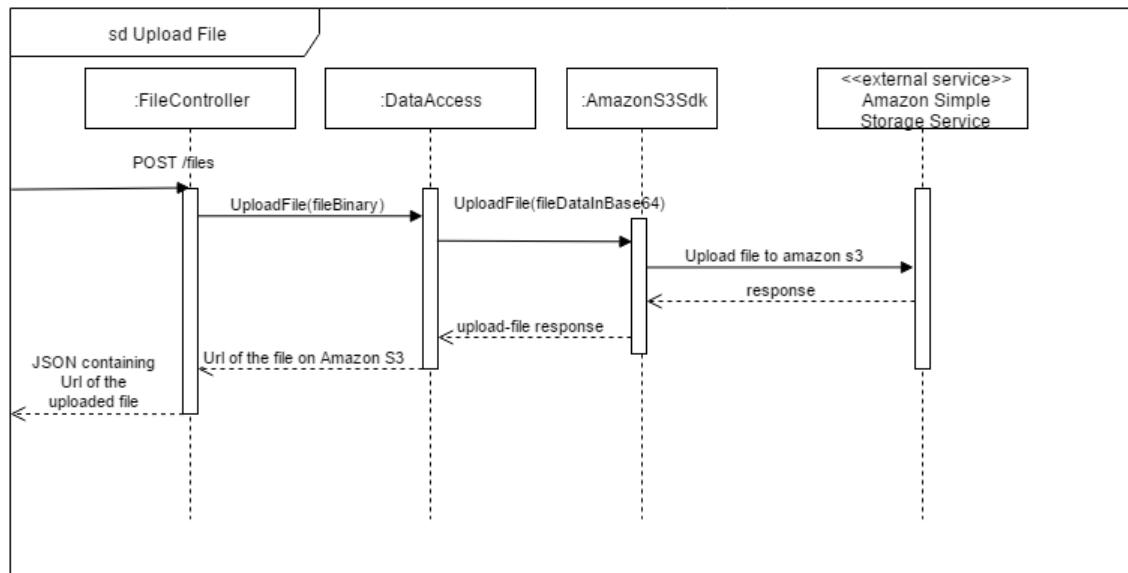


Hình 4.11 Sequence diagram “Gửi tin nhắn thông thường”

#### 4.2.4.7. Sequence diagram cho use-case “Gửi tin nhắn file/hình ảnh”

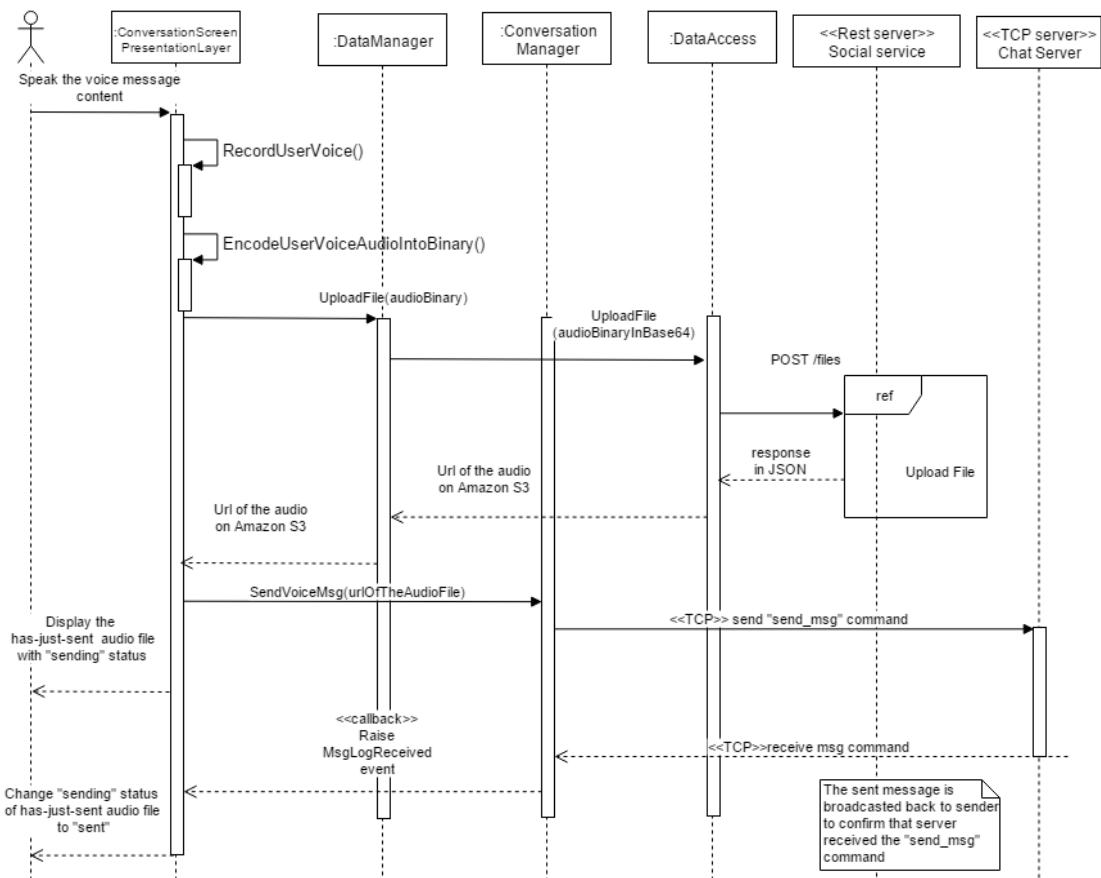


Hình 4.12 Sequence diagram gửi file/hình ảnh.

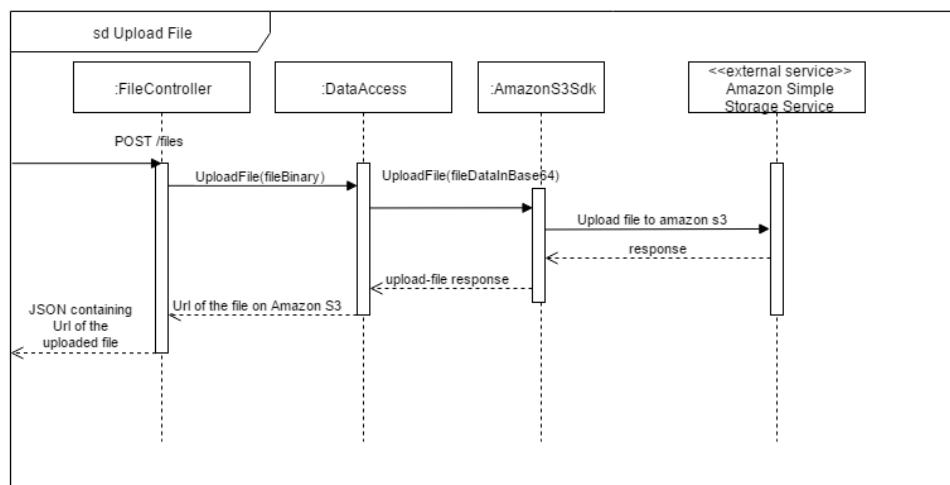


Hình 4.13 Sequence diagram “Upload file” được sử dụng bởi sequence diagram  
“Gửi tin nhắn file/hình ảnh”

#### 4.2.4.8. Sequence diagram cho use-case “Gửi tin nhắn thoại”

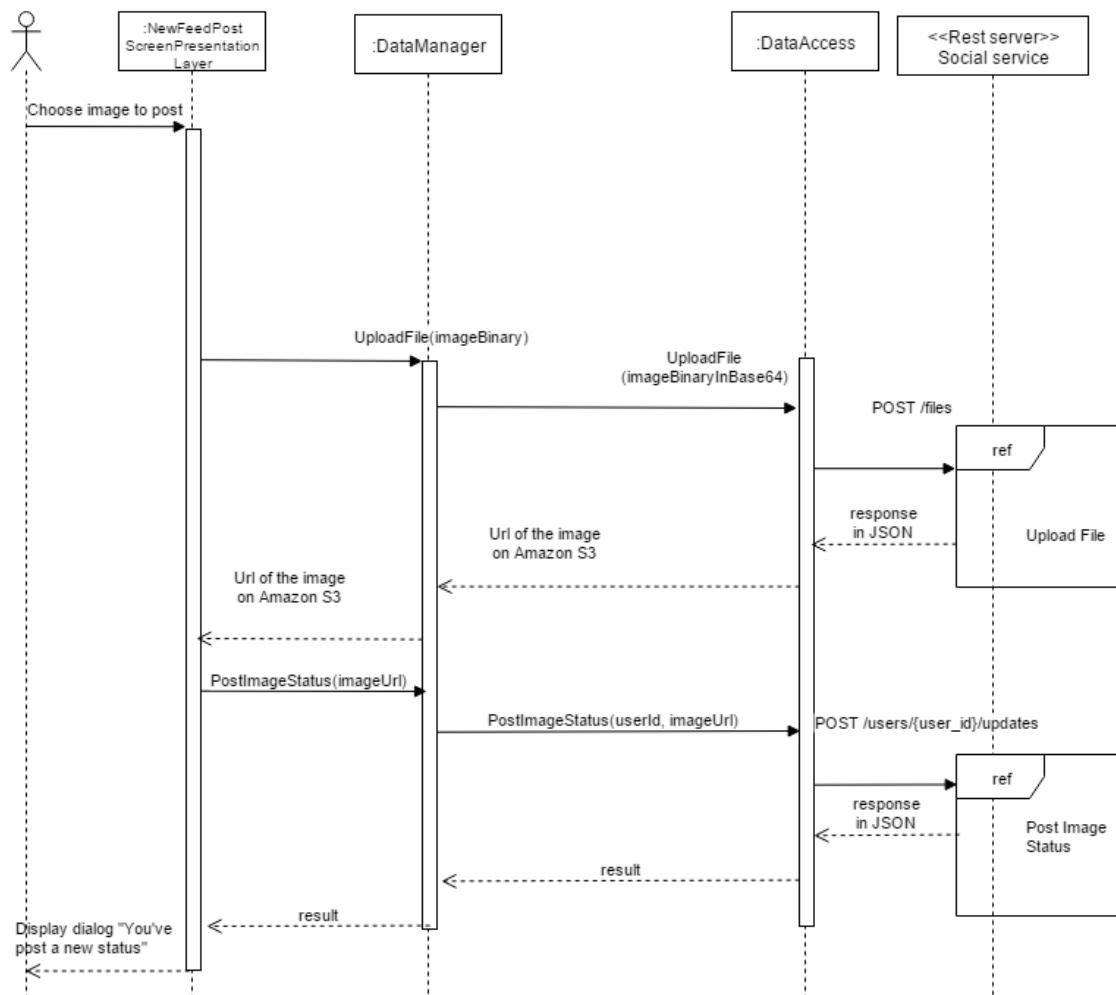


Hình 4.14 Sequence diagram “Gửi tin nhắn thoại”

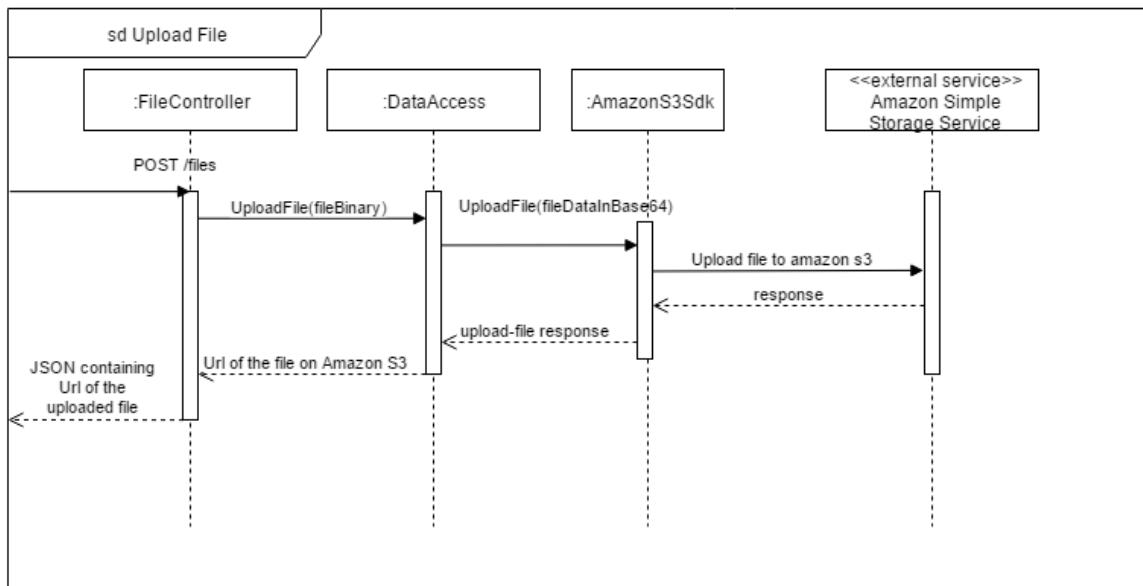


Hình 4.15 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Gửi tin nhắn thoại”

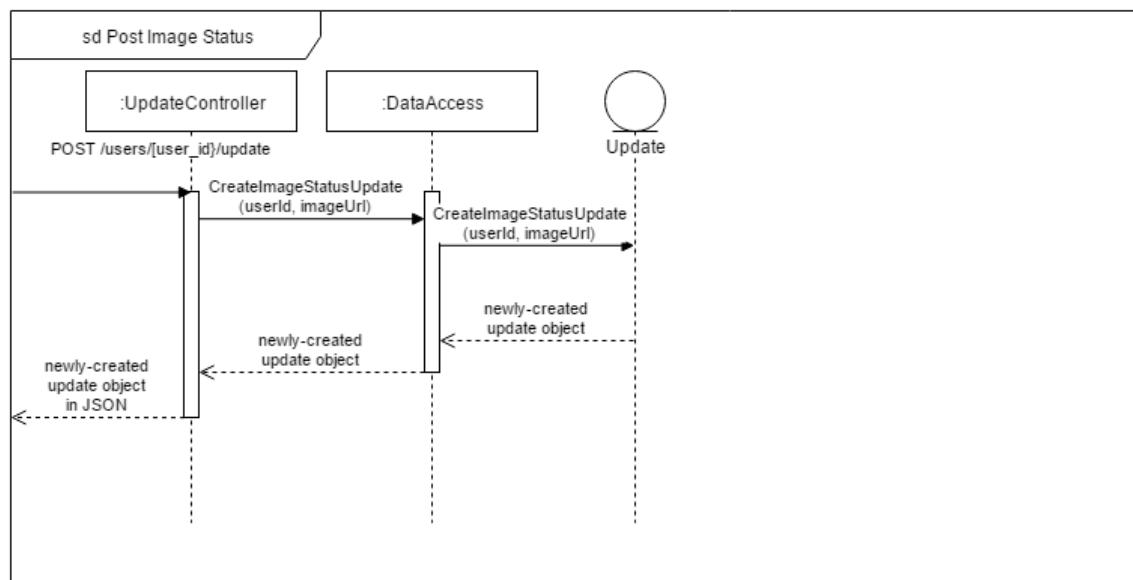
#### 4.2.4.9. Sequence diagram cho use-case “Đăng status hình ảnh”



Hình 4.16 Sequence diagram “Đăng status hình ảnh”

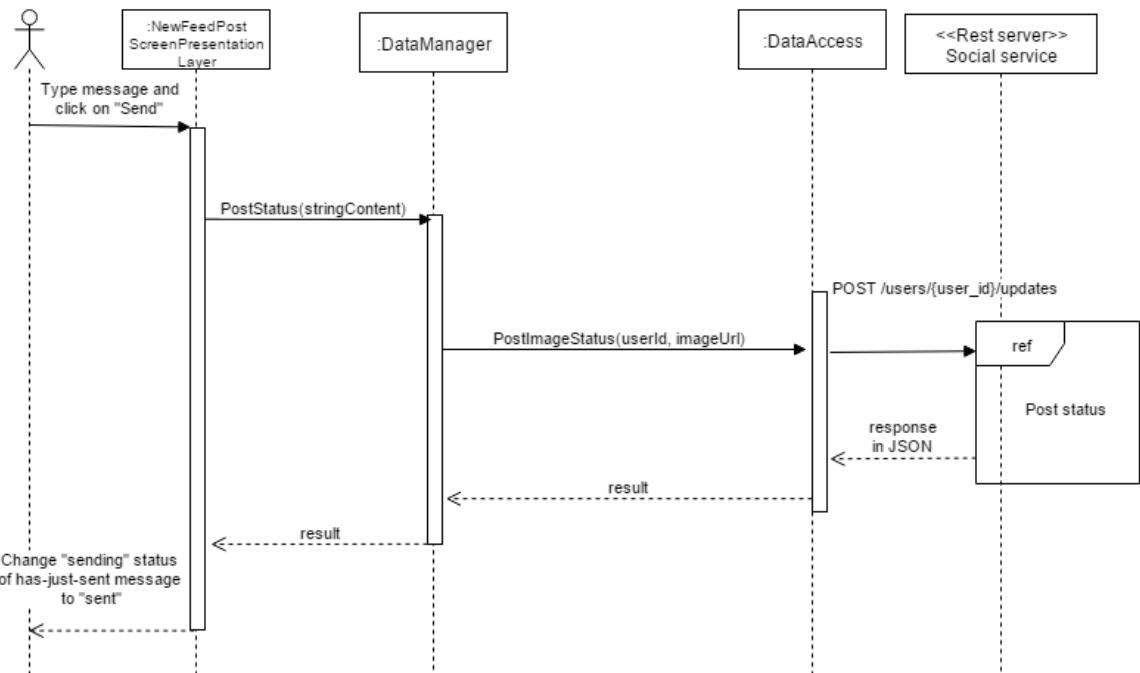


Hình 4.17 Sequence diagram “Upload file” được sử dụng bởi sequence diagram  
“Đăng status hình ảnh”

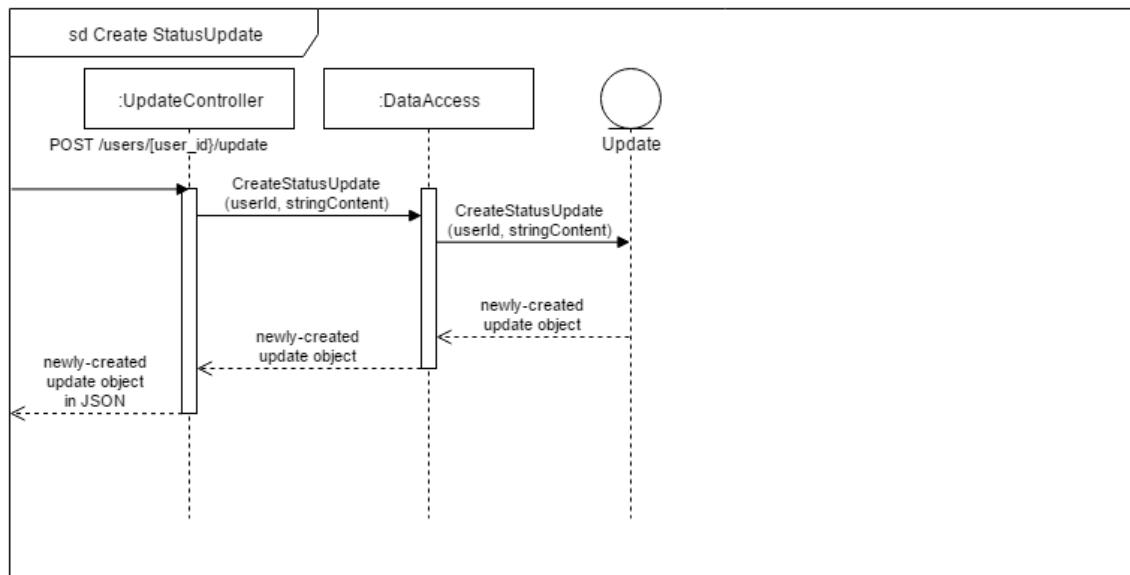


Hình 4.18 Sequence diagram “Post image status” được sử dụng bởi sequence  
diagram “Đăng status hình ảnh”

#### 4.2.4.10. Sequence diagram cho use-case “Đăng status”

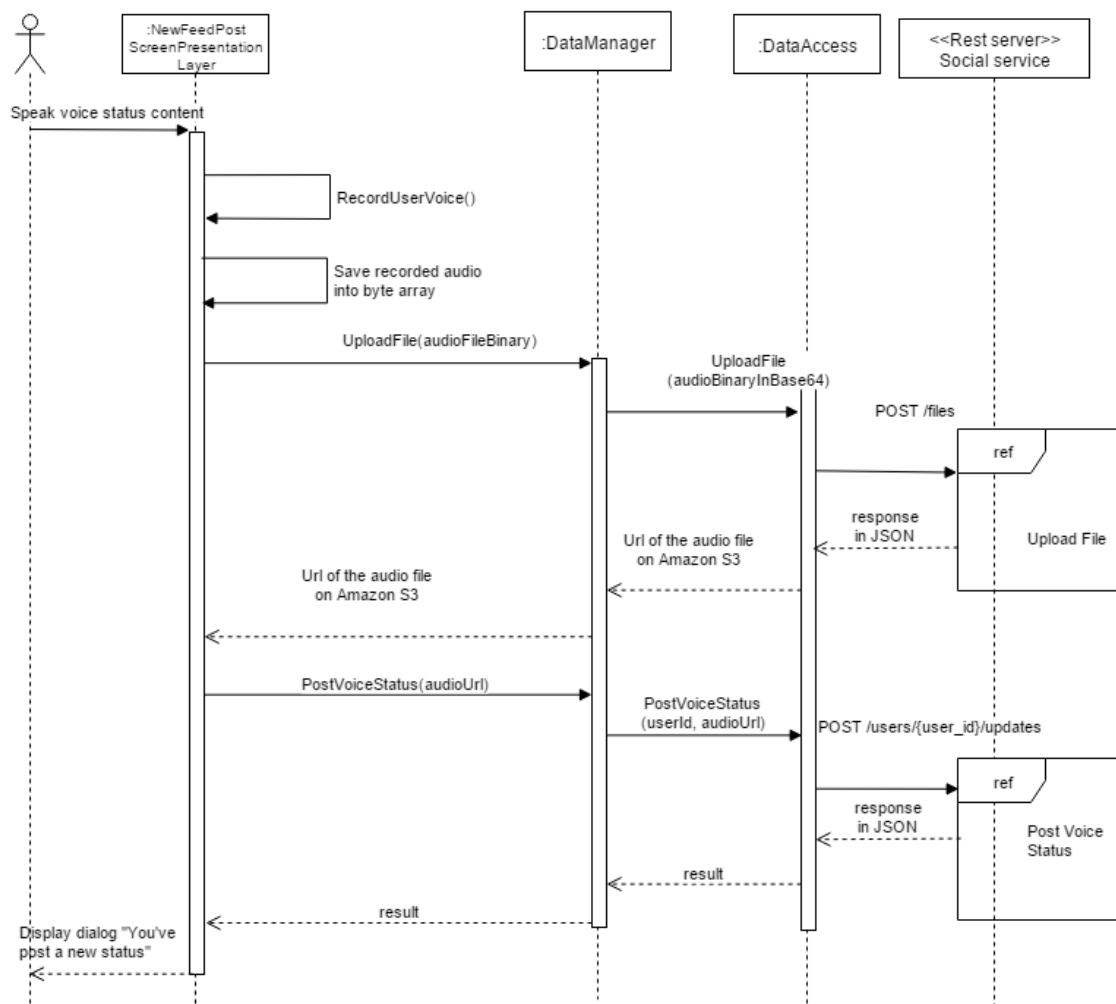


Hình 4.19 Sequence diagram “Đăng status”

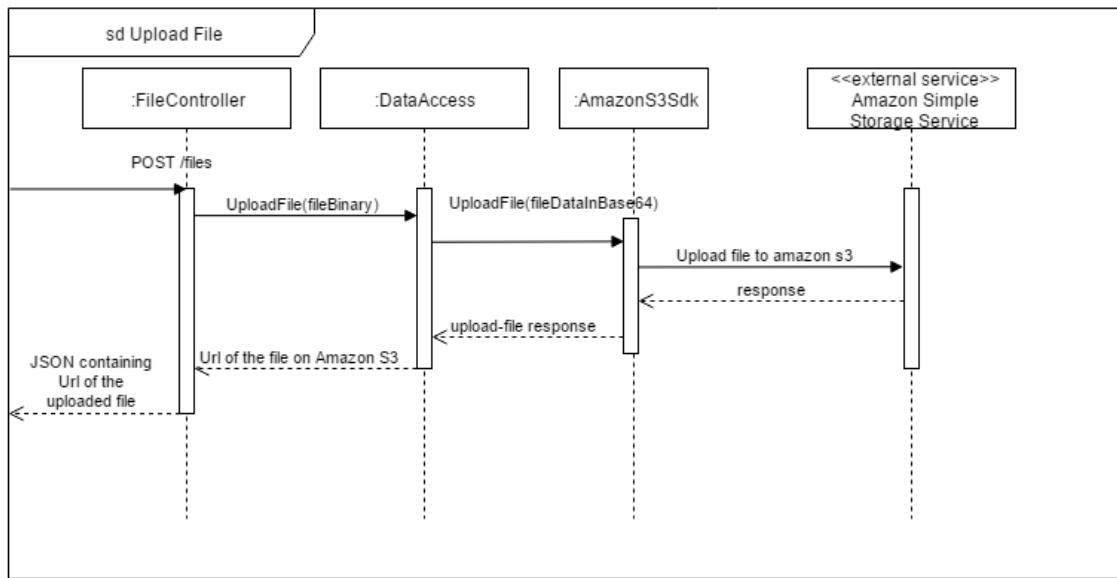


Hình 4.20 Sequence diagram “Create status update” được sử dụng bởi sequence diagram “Đăng status”

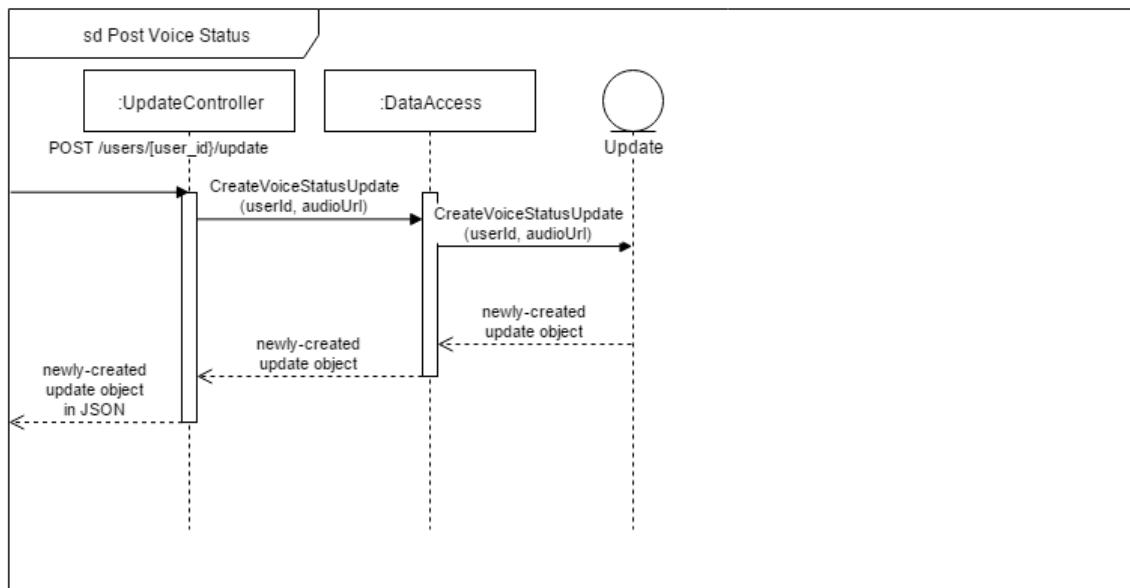
#### 4.2.4.11. Sequence diagram cho use-case “Đăng voice status”



Hình 4.21 Sequence diagram “Đăng voice status”

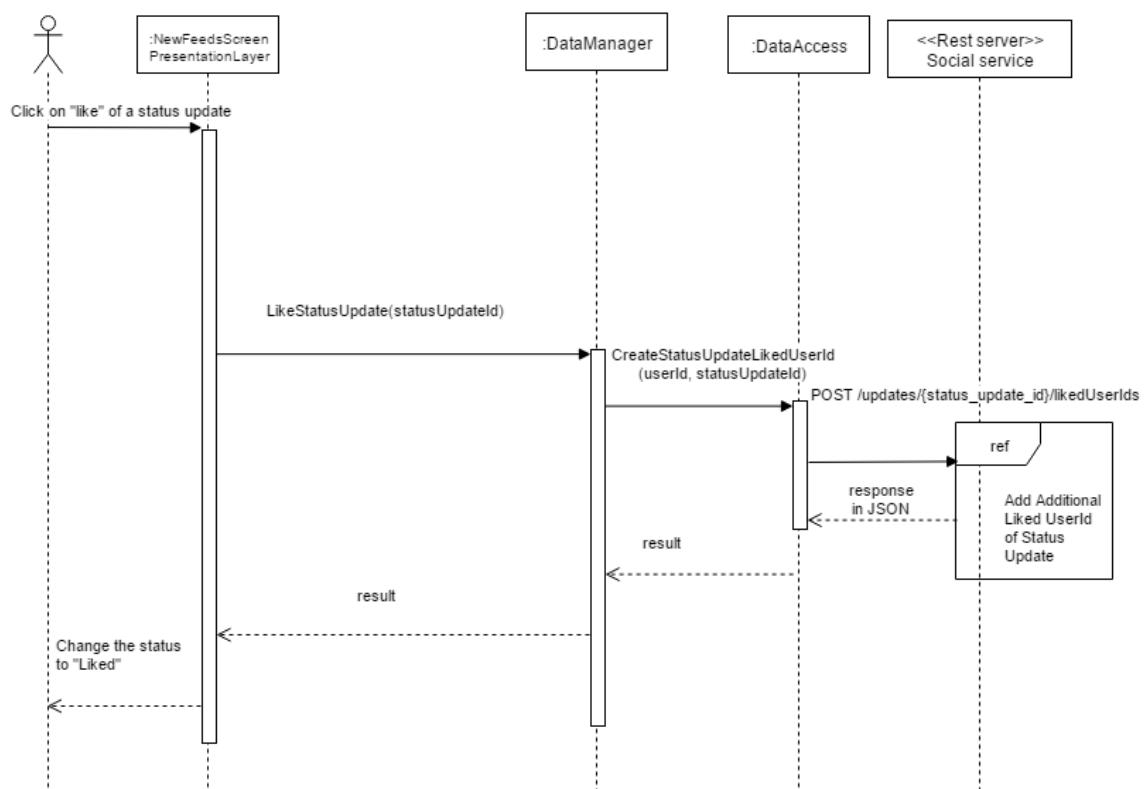


Hình 4.22 Sequence diagram “Upload file” được sử dụng bởi sequence diagram “Đăng voice status”

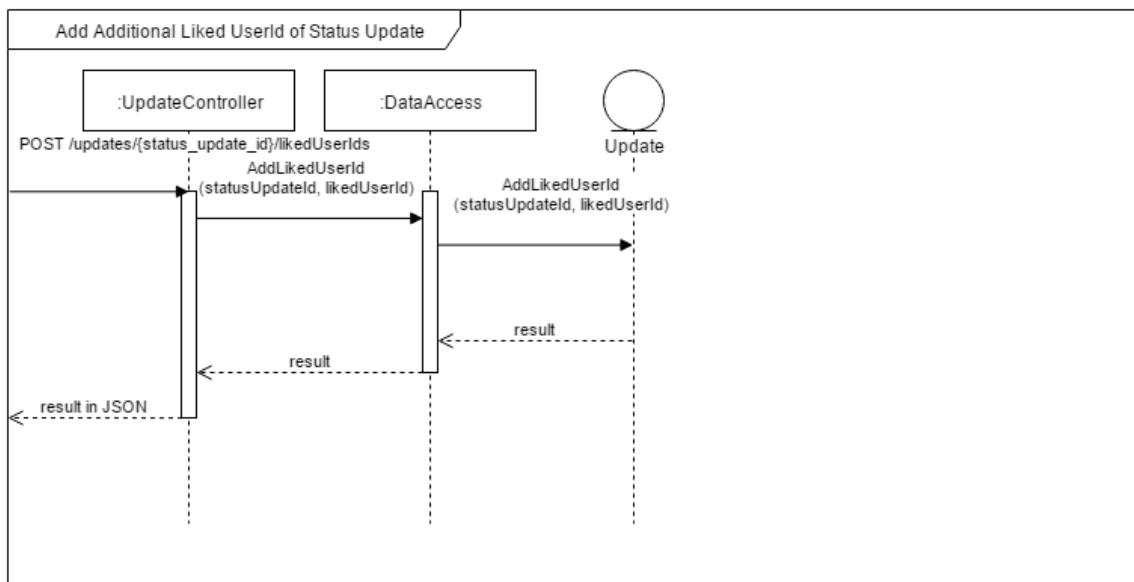


Hình 4.23 Sequence diagram “Post voice status” được sử dụng bởi sequence diagram “Đăng voice status”

#### 4.2.4.12. Sequence diagram cho use-case “Like status”

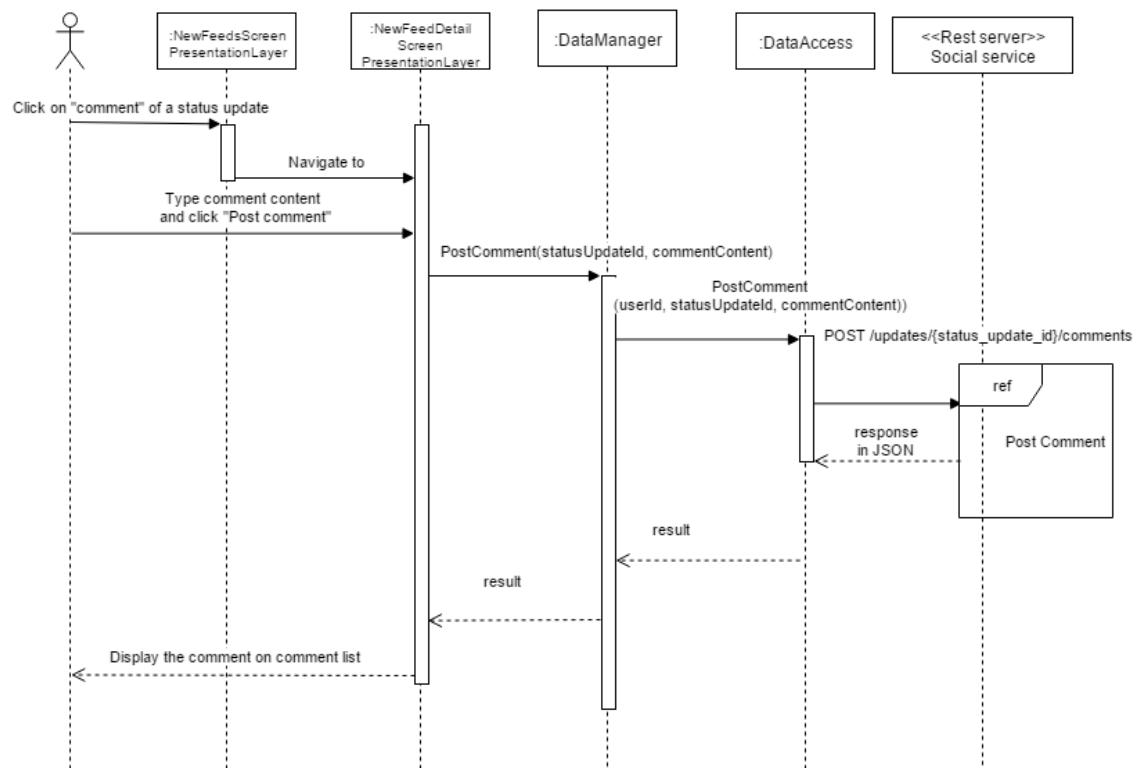


Hình 4.24 Sequence diagram “like status”

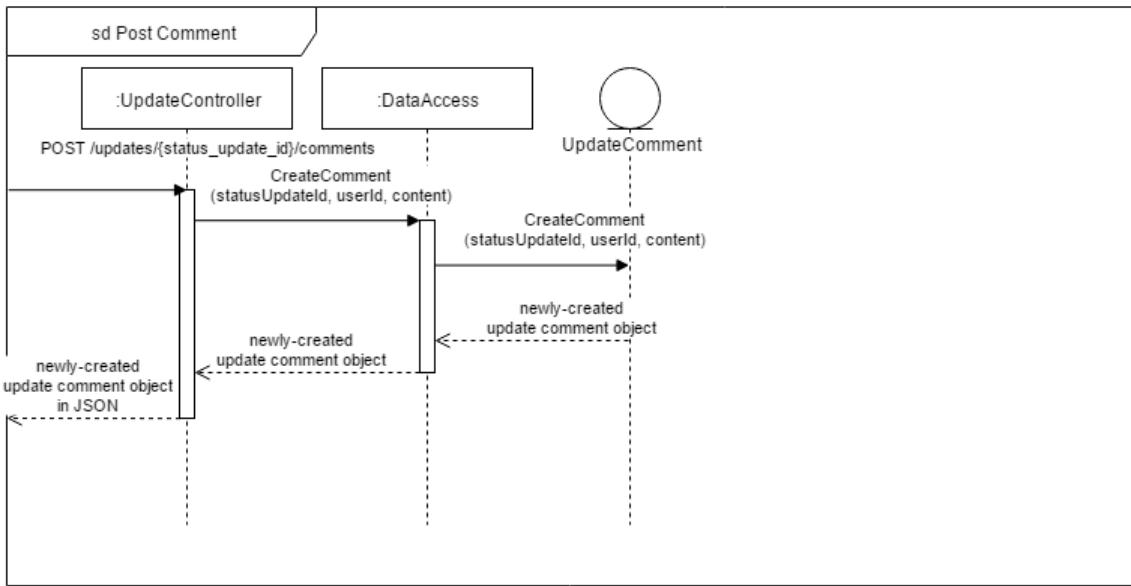


Hình 4.25 Sequence diagram “Add additional liked-user-id of status-update” được sử dụng bởi sequence diagram “like status”

#### 4.2.4.13. Sequence diagram cho use-case “Comment trên status”

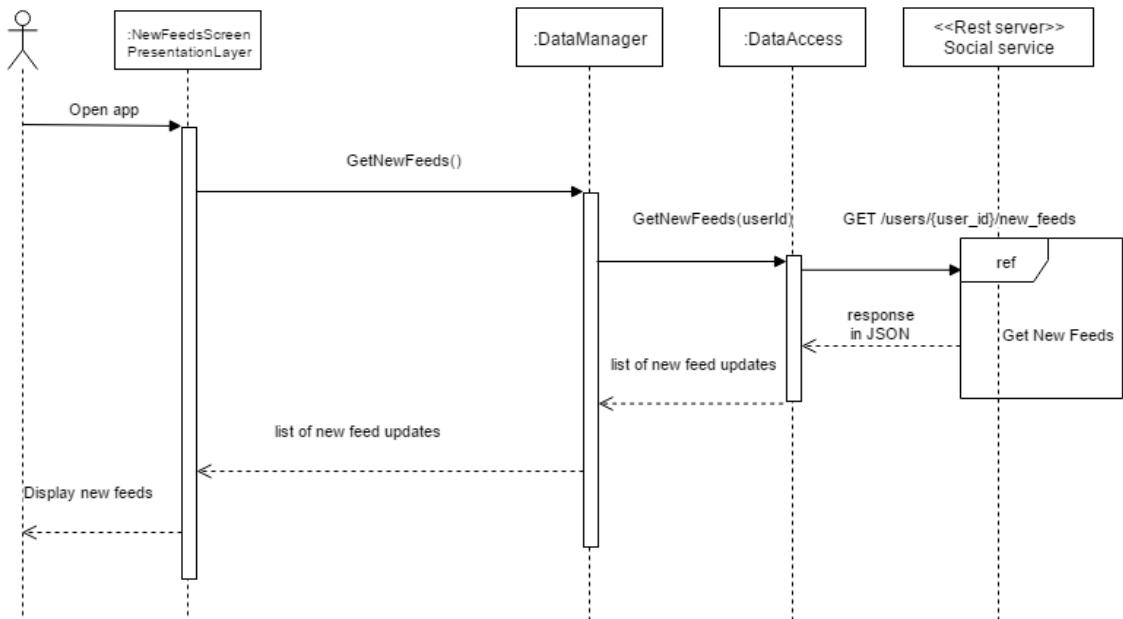


Hình 4.26 Sequence diagram “Comment trên status”

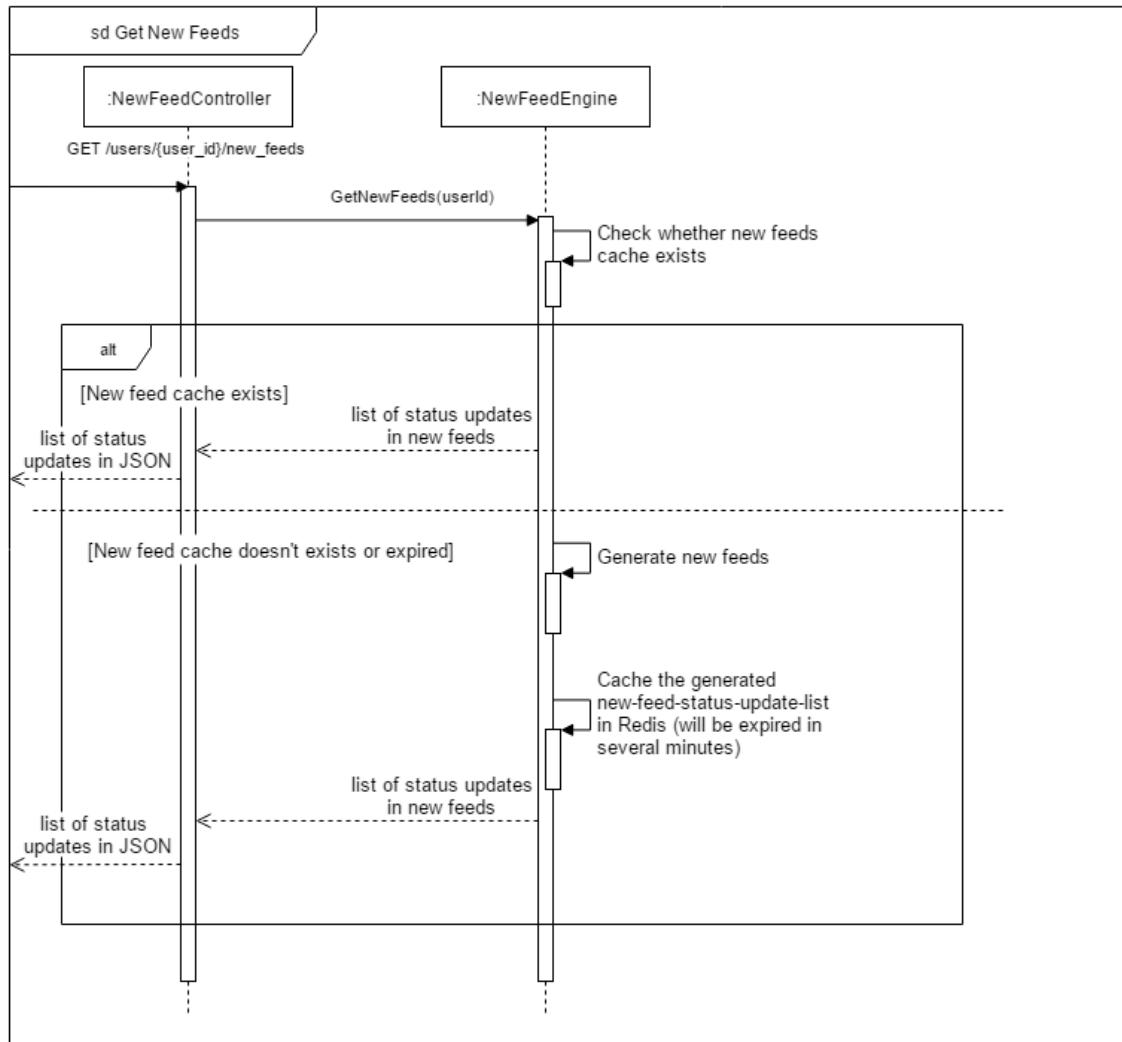


Hình 4.27 Sequence diagram “Post comment” được sử dụng bởi sequence diagram “Comment trên status”

#### 4.2.4.14. Sequence diagram cho use-case “Duyệt news feed”



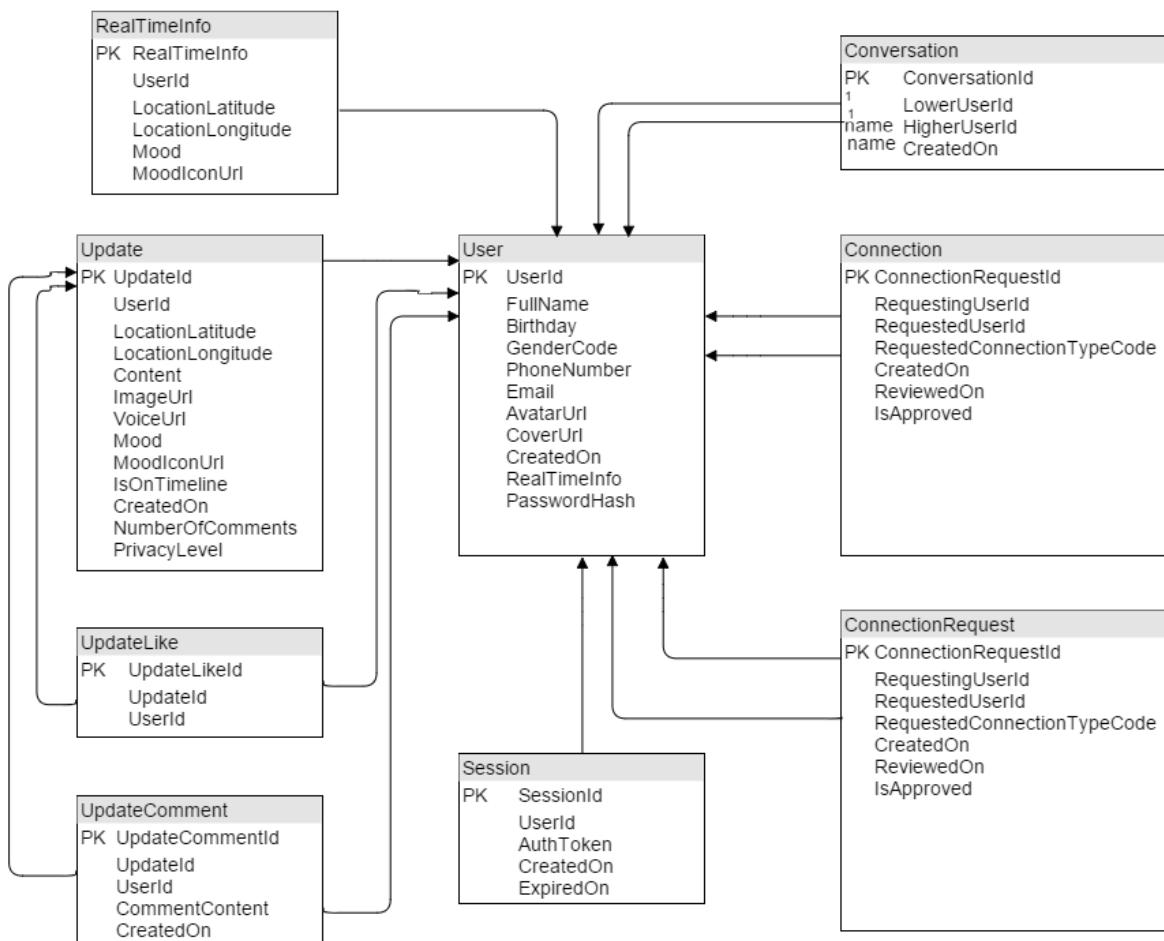
Hình 4.28 Sequence diagram “Duyệt news feed”



Hình 4.29 Sequence diagram “Get News Feed” được sử dụng bởi sequence diagram  
“Duyệt news feed”

### 4.3. Thiết kế dữ liệu

#### 4.3.1. Sơ đồ CSDL



Hình 4.30 Sơ đồ CSDL của hệ thống

STT	Tên Bảng	Diễn giải	Ghi chú
1	User	Chứa thông tin người dùng	
2	Session	Chứa thông tin về session đăng nhập	
3	Update	Chứa thông tin các cập nhật trạng thái của người dùng	
4	UpdateLike	Lưu giữ thông tin user đã like cập nhật trạng thái	

5	UpdateComment	Lưu giữ thông tin comment của người dùng trên cập nhật trạng thái	
6	RealTimeInfo	Chứa một số thông tin realtime của người dùng	
7	Connection	Chứa thông tin về kết nối (quan hệ) giữa 2 người dùng	
8	ConnectionRequest	Chứa thông tin về yêu cầu tạo kết nối hoặc yêu cầu thay đổi mối quan hệ	
9	Conversation	Chứa thông tin về cuộc hội thoại giữa 2 người dùng. Social service chỉ lưu giữ thông tin người dùng đó đã từng có hội thoại với những người dùng nào. Dữ liệu message log được lưu trữ trên CSDL Redis của Chat Server	

Bảng 4.23 Bảng mô tả các bảng trong CSDL

#### 4.3.2. Mô tả chi tiết các bảng

Bảng User: Lưu trữ thông tin của người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
UserId	Integer	Khóa chính
FullName	VARCHAR(50)	Tên người dùng
Birthday	DateTime	Ngày sinh
GenderCode	VARCHAR(10)	Mã giới tính của người dùng (“male” là nam, “female” là nữ)

PhoneNumber	VARCHAR(20)	SĐT của người dùng
Email	VARCHAR(50)	Địa chỉ email
AvatarUrl	VARCHAR(100)	Url tới hình avatar của người dùng
CoverUrl	VARCHAR(100)	Url tới hình cover của người dùng
CreatedOn	DateTime	Thời gian tạo tài khoản
PasswordHash	VARCHAR(100)	Mật khẩu của người dùng

Bảng 4.24 Bảng User

Bảng Session: Lưu trữ thông tin về session đăng nhập của người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
SessionId	Integer	Khóa chính
UserId	Integer	Khóa ngoại
AuthToken	Token để xác thực người dùng	
CreatedOn	DateTime	Thời gian tạo session
ExpiredOn	DateTime	Thời gian session hết hạn

Bảng 4.25 Bảng session

Bảng Update: Lưu trữ thông tin về các cập nhật trạng thái của người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
UpdateId	Integer	Khóa chính
UserId	Integer	Khóa ngoại
LocationLatitude	Double	Kinh độ tại vị trí đăng cập nhật trạng thái
LocationLongitude	Double	Vĩ độ tại vị trí đăng cập nhật trạng thái
Content	VARCHAR(1000)	Nội dung của cập nhật trạng thái

ImageUrl	VARCHAR(100)	Url hình ảnh của cập nhật trạng thái
VoiceUrl	VARCHAR(100)	Url file audio ghi âm voice status của người dùng
Mood	VARCHAR(20)	Tâm trạng
MoodIconUrl	VARCHAR(100)	Url hình ảnh icon đại diện cho tâm trạng của người dùng
IsOnTimeline	Boolean	Xác định xem cập nhật trạng thái này có được đưa lên timeline không
CreatedOn	DateTime	Thời gian tạo
NumberOfComments	Integer	Số lượng comment
PrivacyLevel	Integer	Mức độ bảo mật (Giới hạn chỉ những người thân thiết mới được thấy)

Bảng 4.26 Bảng Update

Bảng UpdateLike: Lưu trữ thông tin các user đã like cập nhật trạng thái.

Trường giá trị	Kiểu dữ liệu	Mô tả
UpdateLikeId	Integer	Khóa chính
UpdateId	Integer	Khóa ngoại
UserId	Integer	Khóa ngoại

Bảng 4.27 Bảng UpdateLike

Bảng UpdateComment: Lưu trữ thông tin về các bình luận của người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
UpdateCommentId	Integer	Khóa chính
UpdateId	Integer	Khóa ngoại

UserId	Integer	Khóa ngoại
CommentContent	VARCHAR(500)	Nội dung bình luận
CreatedOn	DateTime	Thời gian tạo

Bảng 4.28 Bảng UpdateComment

Bảng RealTimeInfo: Lưu trữ một số thông tin realtime của người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
RealTimeInfoId	Integer	Khóa chính
UserId	Integer	Khóa ngoại
LocationLatitude	Double	Vị trí hiện tại của người dùng theo kinh độ
LocationLongitude	Double	Vị trí hiện tại của người dùng theo vĩ độ
Mood	VARCHAR(20)	Tâm trạng hiện tại của người dùng
MoodIconUrl	VARCHAR(100)	Url hình ảnh icon đại diện cho tâm trạng của người dùng

Bảng 4.29 Bảng RealTimeInfo

Bảng Connection: Lưu trữ thông tin về kết nối (quan hệ) giữa 2 người dùng.

Trường giá trị	Kiểu dữ liệu	Mô tả
ConnectionId	Integer	Khóa chính
FirstUserId	Integer	Id của người dùng thứ nhất
SecondUserId	Integer	Id của người dùng thứ 2
ConnectionTypeCode	VARCHAR(50)	Mã quan hệ giữa 2 người dùng
CreatedOn	DateTime	Thời gian tạo

Bảng 4.30 Bảng Connection

Bảng ConnectionRequest: Lưu trữ thông tin về yêu cầu tạo kết nối giữa 2 người dùng (Tạo mới quan hệ) hoặc yêu cầu thay đổi loại quan hệ.

Trường giá trị	Kiểu dữ liệu	Mô tả
ConnectionRequestId	Integer	Khóa chính
RequestingUserId	Integer	Id người gửi yêu cầu tạo mới quan hệ/thay đổi loại quan hệ
RequestedUserId	Integer	Id người được gửi yêu cầu
RequestConnectionTypeCode	VARCHAR(50)	Mã của loại quan hệ mà yêu cầu này muốn tạo/thay đổi
CreatedOn	DateTime	Thời điểm tạo
ReviewOn	DateTime	Thời điểm người dùng lựa chọn đồng ý hay hủy bỏ
IsApproved	Boolean	Cho biết người dùng đã đồng ý hay hủy bỏ connection request này. Giá trị của trường này chỉ có hiệu lực khi ReviewOn mang giá trị khác null.

Bảng 4.31 Bảng ConnectionRequest

Bảng Conversation: Lưu trữ thông tin về những user nào đã từng có hội thoại với nhau, nội dung lịch sử hội thoại sẽ được lưu trữ trên CSDL (Redis) của Chat Server.

Trường giá trị	Kiểu dữ liệu	Mô tả
ConversationId	Integer	Khóa chính
LowerUserId	Integer	Id của user có giá trị nhỏ hơn (Nếu kết nối giữa 2 người có id lần lượt là 4, 2 thì LowerUserId có giá trị là 2). Việc này sẽ giúp loại bỏ trùng

		lắp và tăng hiệu suất khi truy vấn.
HigherUserId	Integer	Id của user có giá trị cao hơn.
CreatedOn	DateTime	Ngày tạo

Bảng 4.32 Bảng Conversation

## 4.4. Kiến trúc hệ thống

### 4.4.1. Phân tích thiết kế kiến trúc

Để hệ thống có thể đáp ứng được lượng lớn người dùng thì một đặc điểm quan trọng nhất của hệ thống là ngoài việc nó phải đảm bảo được hiệu suất cao, nó còn phải có khả năng mở rộng (scale) theo chiều ngang.

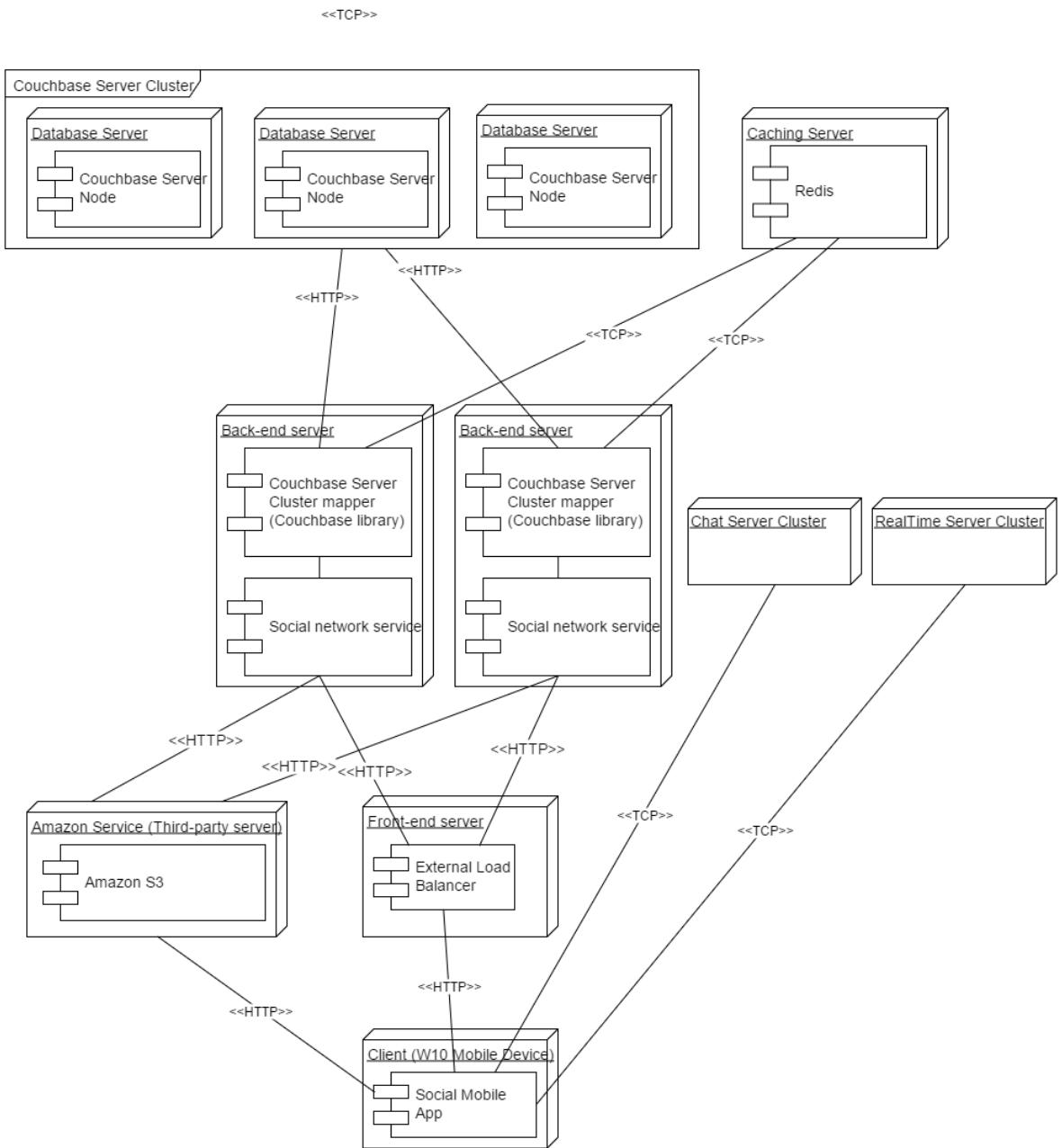
Bản thân Couchbase Server là một hệ quản trị CSDL NoSQL có khả năng phân tán, việc mở rộng (scale) theo chiều ngang trở nên rất đơn giản. Ta chỉ cần set up một Couchbase Server Cluster, sau đó cài đặt các database server khác kết nối tới Cluster này. Cơ chế phân tán của Couchbase Server dựa trên cơ chế phân phối khóa (key), khi người dùng insert document vào thì Cluster Map của thư viện CouchbaseClient sẽ tự động băm key đó ra để xem nó sẽ được lưu trữ trên node nào của cluster và thực hiện lưu trữ vào node đó. Việc truy xuất các document bằng key cũng có cơ chế tương tự. Ngoài ra thì Couchbase Server Cluster sẽ tự động thực hiện việc tái cân bằng để đảm bảo lượng dữ liệu lưu trên các node là gần tương đương nhau, để giúp đạt hiệu suất cao nhất.

Để các Webservice có thể mở rộng (scale) theo chiều ngang thì nó phải *vô trạng thái* (stateless). Nghĩa là không có bất kỳ trạng thái, ngữ cảnh nào của client được lưu trữ trên server, mỗi lần request client phải gửi tất cả dữ liệu cần thiết để server xử lý cho request đó, và tất cả các request đều được xử lý độc lập, không liên quan tới nhau, việc xử lý request này trước hay request kia trước không ảnh hưởng tới kết quả. Sau khi đảm bảo được web service hoàn toàn vô trạng thái (stateless) thì hệ thống Web service có thể dễ dàng được mở rộng (scale) theo chiều ngang bằng cách đưa thêm server vận hành vào sau Server cân bằng tải (Load Balancer).

Để đạt hiệu suất cao nhất thì đôi khi bản thân server phải tự thực hiện cache những dữ liệu cần thiết. Chi phí của việc sinh news feed của người dùng là khá lớn (nếu so với chi phí để xử lý một request thông thường từ client), nên việc tạo ra news feed mới mỗi lần người dùng request là không cần thiết, vì việc cập nhật news feed có thể được thực hiện mỗi 2 – 5 phút 1 lần. Do đó, giải pháp để tăng hiệu suất của server là sử dụng Redis để cache news feed và đặt thời gian hết hạn là 2-5 phút. Mỗi lần client request lấy news feed, server sẽ check xem trong cache có news feed chưa, nếu có rồi thì trả về, nếu chưa có thì sinh ra news feed mới, cache lại và đặt thời gian hết hạn là 2-5 phút. Sau đó trả về news feed vừa mới sinh bởi server.

Đối với một mạng xã hội, việc cập nhật danh sách người dùng đang online và tính năng realtime chat là không thể thiếu. Nếu tận dụng cài đặt phần cập nhật online và realtime chat vào Webservice thì việc cài đặt sẽ rất đơn giản. Nhưng tính năng cập nhật người dùng online và chat realtime đòi hỏi client và server phải liên tục giao tiếp với nhau. Vì HTTP muốn thực hiện 1 request lên server đều phải mở connection, sau khi thực hiện xong thì đóng connect, nên chi phí của nó sẽ rất lớn so với các tác vụ tương tác realtime liên tục, và server sẽ dễ dàng trở nên quá tải nếu số lượng người dùng online tại cùng một thời điểm tương đối lớn. Do đó để đảm bảo server có thể chịu tải với lượng lớn người dùng, Chat Server phải được cài đặt trên một server riêng, sử dụng giao thức TCP để giao tiếp, sử dụng Redis để lưu trữ dữ liệu chat của người dùng, và sẽ sử dụng mô hình kiến trúc hướng sự kiện bất đồng bộ.

#### 4.4.2. Kiến trúc hệ thống



Hình 4.31 Sơ đồ kiến trúc hệ thống của hệ thống mạng xã hội

Hệ thống sẽ được chia làm nhiều thành phần riêng biệt:

- Social mobile app (Client): Ứng dụng mạng xã hội.
- External Load Balancer: Có trách nhiệm cân bằng tải cho hệ thống bao gồm nhiều back-end server, giúp viet mở rộng theo chiều ngang của hệ thống back-end server trở nên dễ dàng.

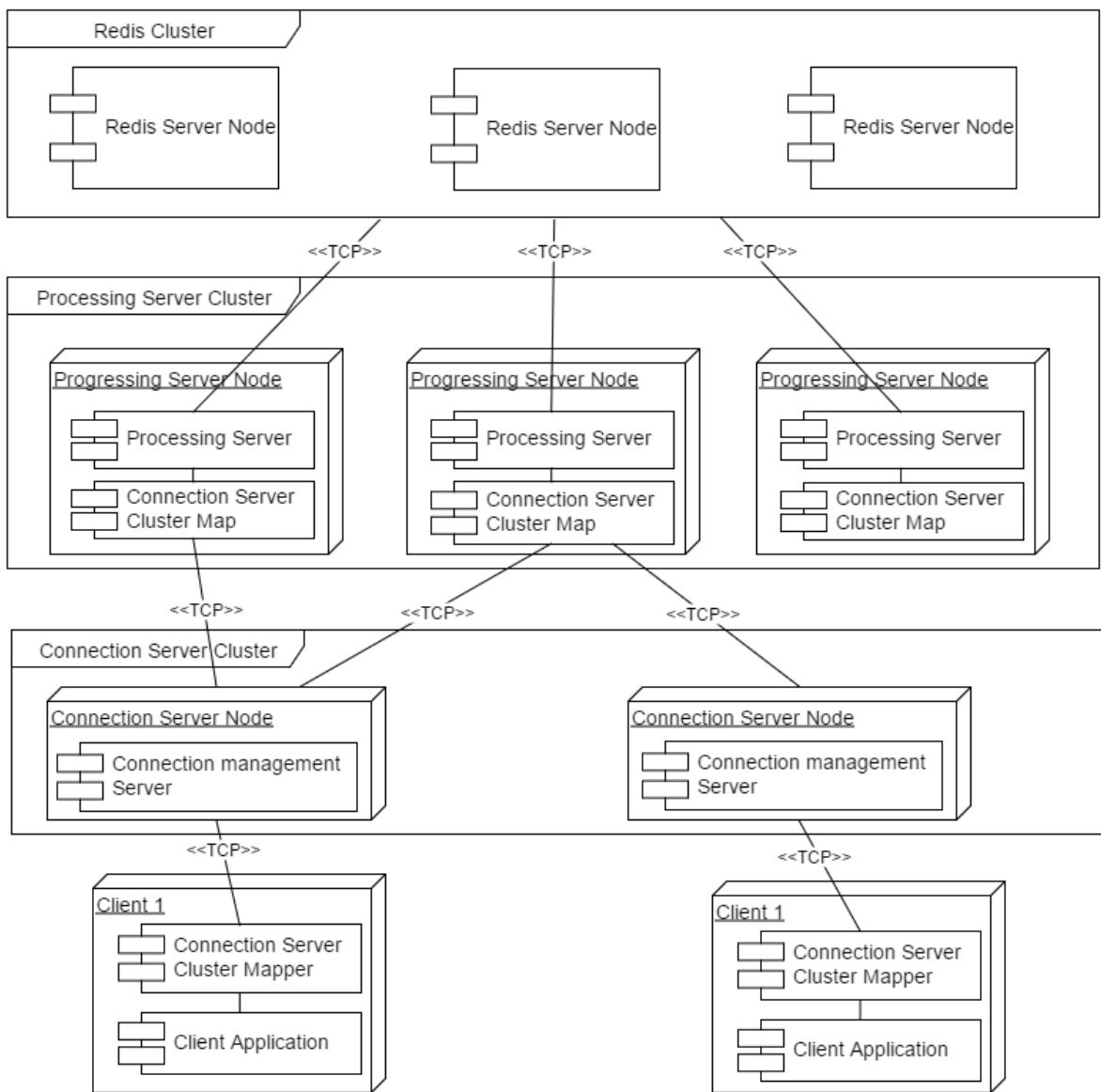
- Amazon S3: Service lưu trữ file được cung cấp bởi Amazon. Vì việc upload và download file lớn gây tốn rất nhiều tài nguyên của server, và thông thường client thường cần phải download rất nhiều hình ảnh (avatar những người dùng khác, hình ảnh trong cập nhật trạng thái...). Do đó, để giảm tải cho server, phần việc chịu trách nhiệm nhận file từ client và gửi file cho client sẽ được chuyển qua Amazon S3.
- Social network service: Chứa hầu hết các xử lý và service của mạng xã hội. Vì thành phần này được cài đặt theo hướng “vô trạng thái” (stateless) việc mở rộng nó theo chiều ngang rất dễ dàng.
- Couchbase Server Cluster: Cụm các Couchbase server node có khả năng tự động phân tán dữ liệu dựa vào key, và tự cân bằng lượng dữ liệu giữa các node để đạt được hiệu suất cao nhất.
- Caching Server: Có nhiệm vụ cache một số dữ liệu cho back-end server. Redis server trên server này sẽ được cấu hình bỏ đi chức năng “persistence” (backup dữ liệu xuống đĩa cứng) để đạt được hiệu suất cao nhất. Bản thân Couchbase Server đã hỗ trợ sẵn khả năng cache tự động, nhưng nó chỉ giúp tăng hiệu suất đối với những dữ liệu thường xuyên được truy xuất. Một số dữ liệu đôi khi không thường được truy xuất nhưng cần phải cache như news feed của người dùng. Do đó, hệ thống sẽ tự thực hiện việc cache riêng cho những loại dữ liệu nhất định.
- Chat Server Cluster: Đảm nhiệm nhiệm vụ xử lý chat real-time.
- Realtime Server Cluster: Chịu trách nhiệm lưu trữ, thực hiện truy vấn, và broadcast dữ liệu realtime của người dùng như: tọa độ, SOS...

#### **4.4.3. Kiến trúc của Chat Server Cluster và Realtime Server Cluster**

Vì Chat Server và Realtime Server sử dụng chung một mô hình xử lý hướng sự kiện bất đồng bộ. Hai hệ thống server này sẽ có chung một mô hình kiến trúc. Vì cùng chung một mô hình xử lý, các tương tác 2 chiều giữa client-server, giao thức tương tác. Cả 2 hệ thống server này sẽ được kết hợp cài đặt với nhau (Về mặt cài đặt, 2 server này chỉ khác nhau về các command xử lý, do đó sẽ được cài đặt chung trên

cùng một project). Nhưng khi deploy ra thực tế, dù một server xử lý có thể đáp ứng cả 2 nhiệm vụ là làm chat server và realtime server, hai hệ thống này vẫn sẽ được deploy riêng biệt để không làm ảnh hưởng tới nhau, cũng như dễ dàng cho việc quản trị và quản lý dữ liệu.

Mô hình kiến trúc phân tán của Chat Server và Realtime Server (trong mô hình kiến trúc này, cả 2 server này đều được gọi cũng là Processing server):



Hình 4.32 Mô hình kiến trúc phân tán của Chat Server và Realtime Server

Trong mô hình kiến trúc này, hệ thống sẽ được chia làm ba tầng:

- Tầng lưu trữ dữ liệu: Tầng này sử dụng Redis để lưu trữ dữ liệu, để đạt được hiệu suất đọc/ghi cao nhất. Trong tầng này, Redis cluster sẽ bao gồm nhiều Redis node, một node này sẽ chứa một phần trong tổng số 4096 hash slot của Redis cluster.
- Tầng xử lý: Ở tầng này sẽ bao gồm nhiều server xử lý tạo thành một cụm server xử lý. Các server xử lý này sẽ hoàn toàn vô trạng thái, và không quản lý bất kỳ kết nối nào với client. Mỗi server xử lý sẽ kết nối trực tiếp tới tất cả các Connection Server trong Connection Server Cluster.
- Tầng quản lý kết nối: Ở tầng này bao gồm nhiều server quản lý kết nối tạo thành một cụm server quản lý kết nối. Nhiệm vụ thứ nhất của server quản lý kết nối là: nhận thông điệp/lệnh từ client và gửi lên một server bất kỳ dựa vào hàm sinh số ngẫu nhiên dựa trên phân phối chuẩn (để đảm bảo lượng thông điệp/lệnh gửi vào từng server xử lý là gần như tương đương nhau) (tương tự như một load balancer). Nhiệm vụ thứ 2 của server quản lý kết nối là: Giúp server xử lý gửi thông điệp/lệnh đến đúng user nhất định dựa trên user-id.

Tư tưởng hoạt động của cụm server quản lý kết nối:

- Tất cả các server xử lý và các client đều nắm giữ một cluster mapper. Cluster mapper này là một mảng băm giúp server xử lý và client có thể biết được Connection Server nào trong Connection Server cluster chịu trách nhiệm quản lý kết nối của một user bất kỳ. Và bằng cách dùng giá trị CRC16(user-id) để truy vấn trong cluster mapper, nó sẽ cho ra được địa chỉ của Connection Server quản lý user-id này. Cơ chế hoạt động của Cluster Mapper tương tự cơ chế Distributed Hash Table. Bên dưới Cluster Mapper là một mảng sao cho tại index CRC16(user-id)%số\_lượng\_server sẽ chứa địa chỉ ip của server quản lý kết nối của user đó.
- Client sẽ gửi thông điệp/lệnh lên hệ thống thông qua connection server chịu trách nhiệm quản lý kết nối TCP của mình. Và server xử lý sẽ gửi thông điệp/lệnh tới một user nhất định thông qua connection server chịu trách nhiệm quản lý kết nối TCP của user đó. Việc xác định connection server nào chịu

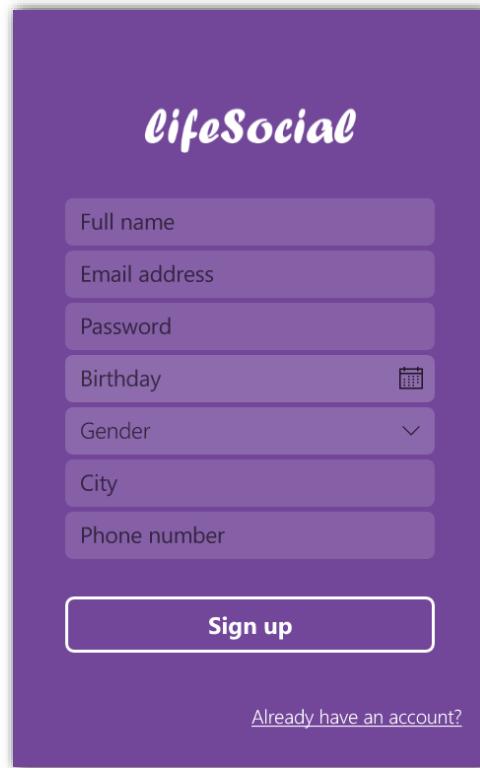
trách nhiệm quản lý cho kết nối của user nào sẽ được thông qua Cluster Mapper.

Mô hình hoạt động:

- Khi một Client muốn gửi một thông điệp/lệnh lên hệ thống: Trước tiên nó sẽ băm user-id của mình bằng CRC16 rồi lấy giá trị đó truy vấn trong Cluster mapper để tìm connection server (địa chỉ ip) chịu trách nhiệm quản lý kết nối của user-id này, và kết nối tới connection server đó. Việc kết nối này sẽ được thực hiện duy nhất lần đầu tiên khi ứng dụng được mở. Sau đó, kết nối sẽ được giữ cho tới khi client bị tắt. Sau khi kết nối được tới connection server, nó sẽ gửi thông điệp/lệnh cần gửi lên, và connection server sẽ chọn ngẫu nhiên một server xử lý (bằng hàm sinh số ngẫu nhiên dựa trên phân phối chuẩn) để thực hiện xử lý thông điệp/lệnh đó.
- Khi một server xử lý muốn gửi một thông điệp/lệnh cho một user: Đầu tiên, nó sẽ dùng băm user-id bằng CRC16 rồi lấy giá trị đó truy vấn trong Cluster Mapper để tìm connection server (địa chỉ ip) chịu trách nhiệm quản lý kết nối của user này, và gửi thông điệp/lệnh đó đến cho connection server này. Connection server sẽ tiếp tục gửi thông điệp/lệnh đó tới tất cả các client đăng nhập vào hệ thống bằng user đó.

## 4.5. Giao diện phần mềm

### 4.5.1. Màn hình “Sign Up”



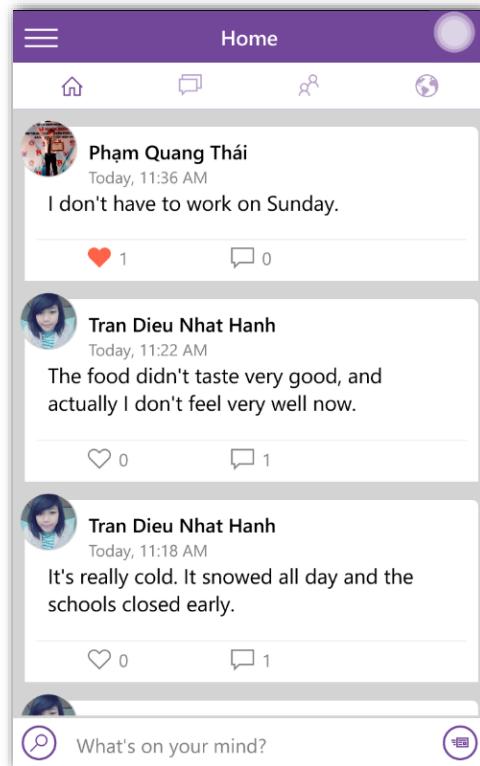
Hình 4.33 Giao diện màn hình “Sign Up”

#### 4.5.2. Màn hình “Sign In”



Hình 4.34 Giao diện màn hình “Sign In”

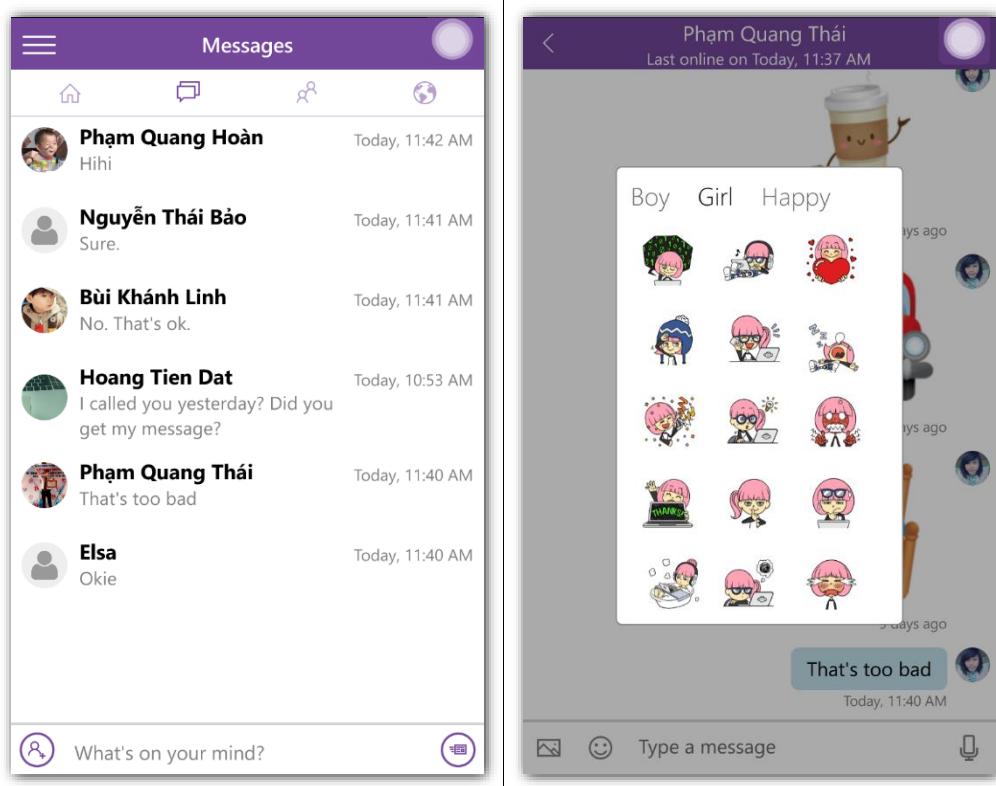
#### 4.5.3. Màn hình “News Feed”



Hình 4.35 Giao diện màn hình “News Feed”

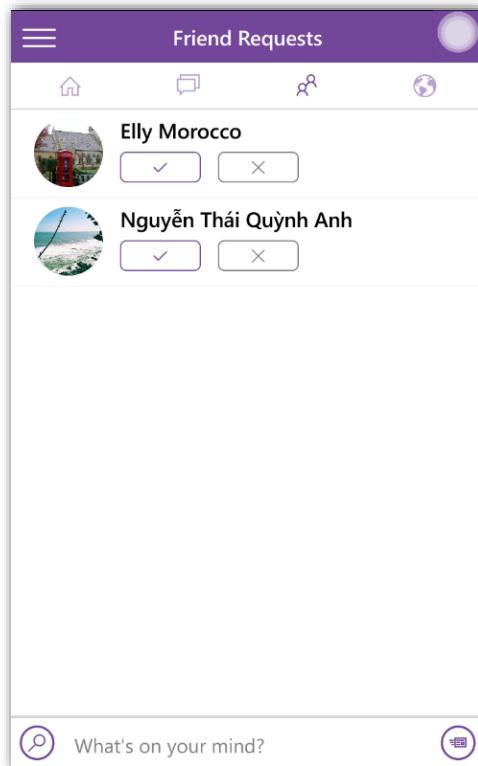
Hiển thị các cập nhật trạng thái của người dùng.

#### 4.5.4. Màn hình “Messages”



Hình 4.36 Giao diện màn hình “Message”

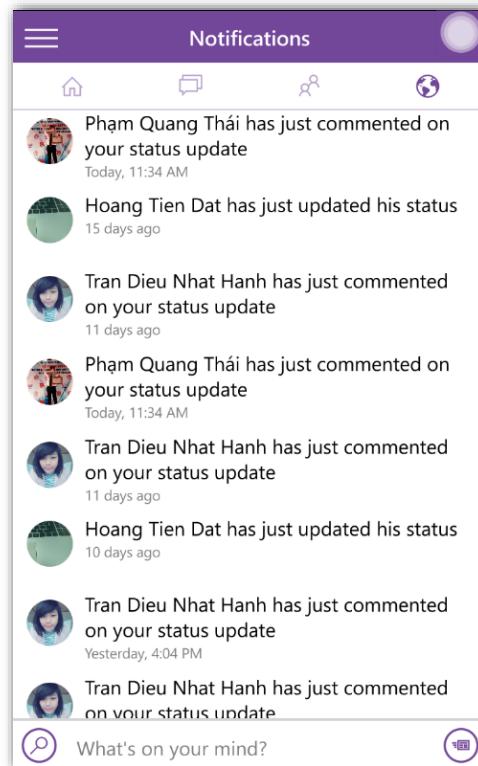
#### 4.5.5. Màn hình “Friend Requests”



Hình 4.37 Giao diện màn hình “Friend Requests”

Hiển thị danh sách các lời mời kết bạn. Người dùng có thể chọn chấp nhận hay từ chối lời mời kết bạn. Hoặc có thể chọn đi đến trang cá nhân của người gửi lời mời.

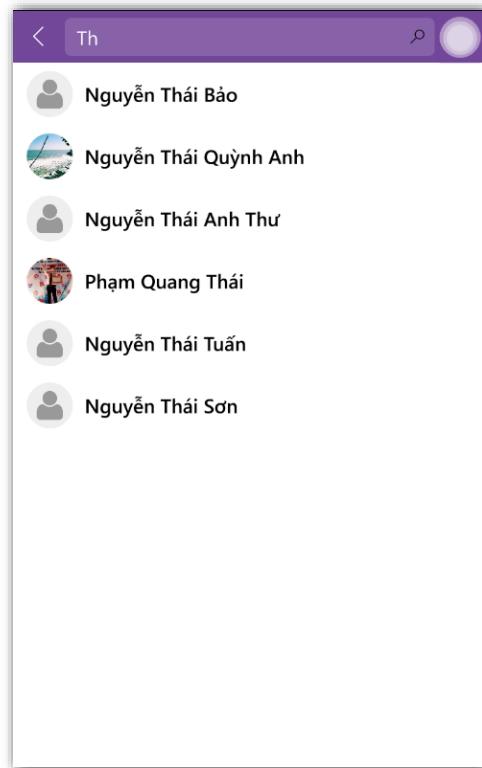
#### 4.5.6. Màn hình “Notifications”



Hình 4.38 Giao diện màn hình “Notifications”

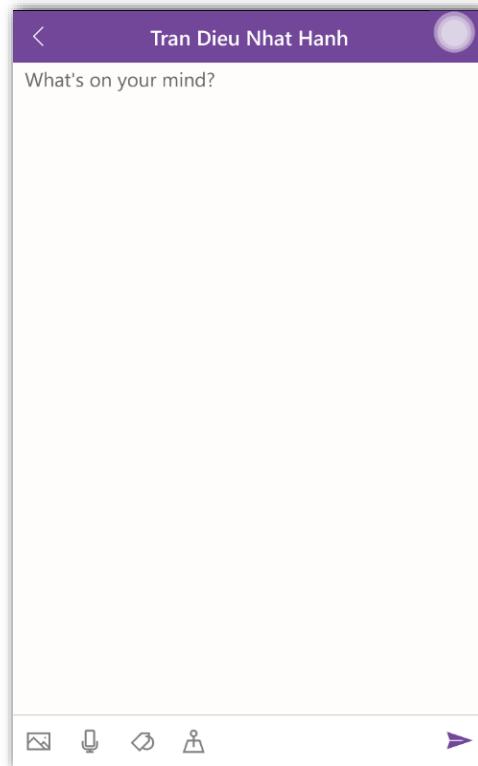
Hiển thị cập nhật các thông báo mới nhất từ bạn bè cho người dùng.

#### 4.5.7. Màn hình “Search”



Hình 4.39 Giao diện màn hình “Search”

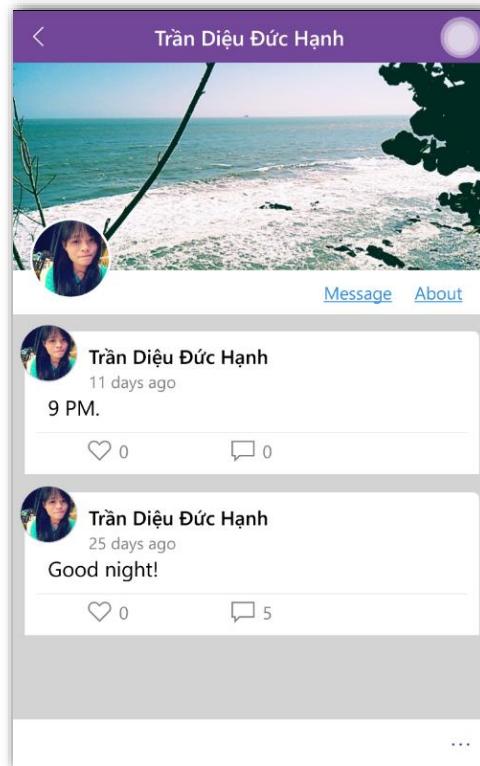
#### 4.5.8. Màn hình “Sharing”



Hình 4.40 Giao diện màn hình “Sharing”

Người dùng có thể tùy chọn chia sẻ cập nhật trạng thái, vị trí, hình ảnh, giọng nói.

#### 4.5.9. Màn hình “Profile”



Hình 4.41 Giao diện màn hình “Profile”

Hiển thị danh sách trạng thái, hoạt động của cá nhân người dùng được chọn để xem.

#### 4.5.10. Màn hình “Info”

The image displays two side-by-side screenshots of a mobile application's 'Info' screen. Both screens have a purple header bar with a back arrow, the word 'Info', and a pencil icon for editing. The left screenshot shows a list of personal information fields with their corresponding values:

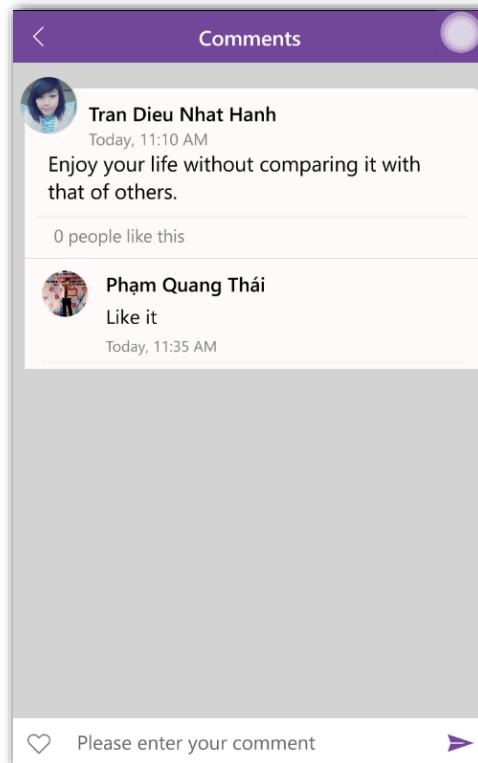
- Name: Tran Dieu Nhat Hanh
- Gender: female
- Birthday: 12/1/1993
- Hometown: Dalat
- Phone: 01689195103
- Education: University of Information Technology
- About me: (empty text area)

The right screenshot shows the same fields but includes a small calendar icon next to the Birthday field. It also features a large purple 'Save' button at the bottom.

Hình 4.42 Giao diện màn hình “Info”

Hiển thị thông tin cá nhân của người dùng. Có thể thay đổi, cập thông tin của bản thân. Và chỉ có thể xem thông tin của người dùng khác.

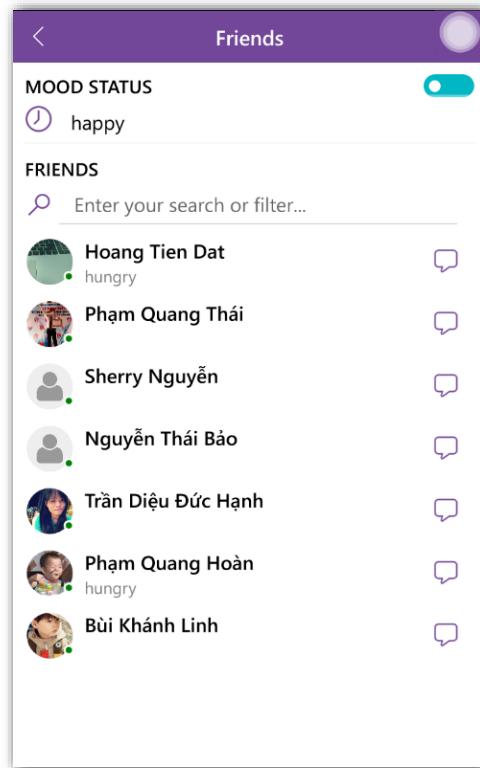
#### 4.5.11. Màn hình “Comments”



Hình 4.43 Giao diện màn hình “Comments”

Hiển thị chi tiết một cập nhật trạng được chọn xem. Người dùng có thể like hoặc comment cập nhật trạng thái đó.

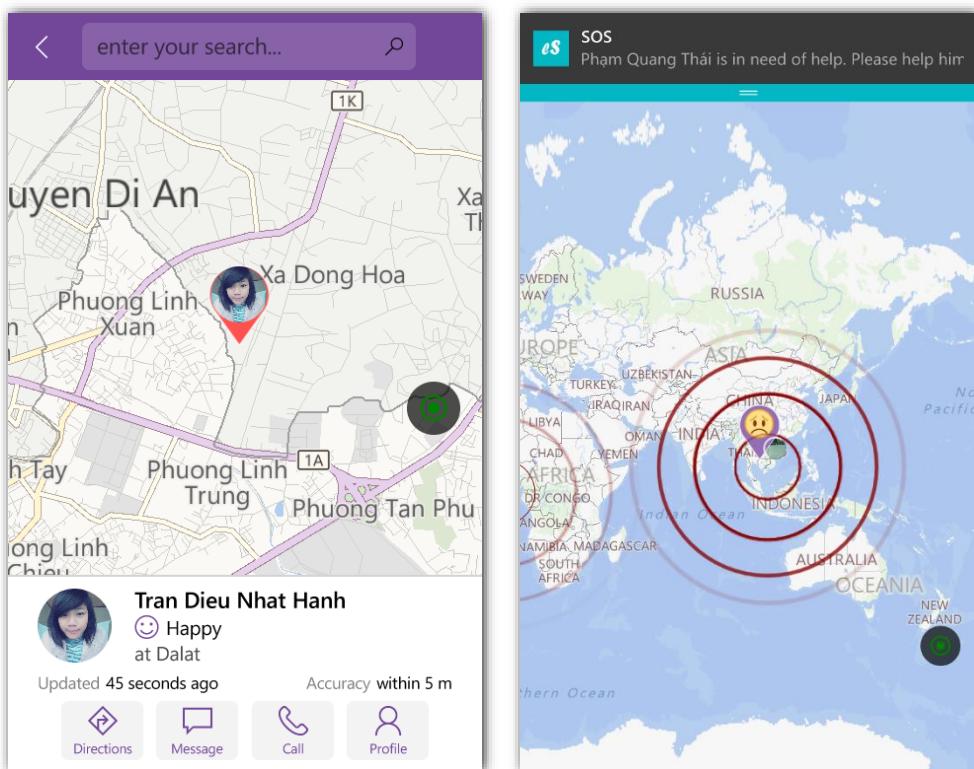
#### 4.5.12. Màn hình “Friends”



Hình 4.44 Giao diện màn hình “Friends”

Hiển thị danh sách tất cả người dùng đã có kết nối với bạn. Người dùng có thể tìm hoặc lọc bạn bè theo tên hoặc trạng thái tạm thời.

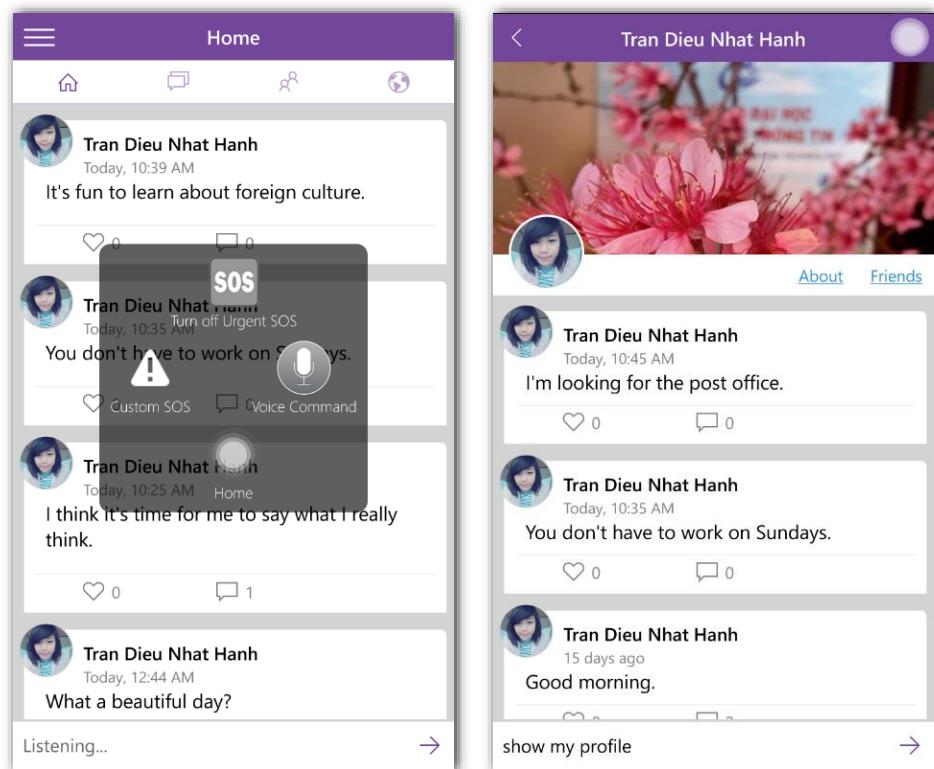
#### 4.5.13. Màn hình “World Map”



Hình 4.45 Giao diện màn hình “World Map”

Hiển thị cập nhật trạng thái và vị trí của bạn bè trên bản đồ. Người dùng có thể cập nhật trạng thái, sử dụng chức năng chỉ đường tới vị trí của bạn, gọi, nhắn tin hoặc xem trang cá nhân của người dùng khác. Ngoài ra, người dùng có thể thực hiện tương tác với các người dùng khác như đánh dấu một điểm bất kỳ trên world map kèm theo icon và nội dung, thông báo đang ở trong tình trạng nguy hiểm và nhờ sự trợ giúp, tìm đường đến chỗ một người bạn nào đó, tạo cảnh báo tại một vùng trên world map (khi những người bạn của người dùng đến gần vùng đó, họ sẽ được cảnh báo. Ví dụ: Tại điểm đó cảnh sát đang bắn tốc độ, khi một người bạn đang chạy với tốc độ cao đến gần thì điện thoại sẽ rung kèm theo thông báo).

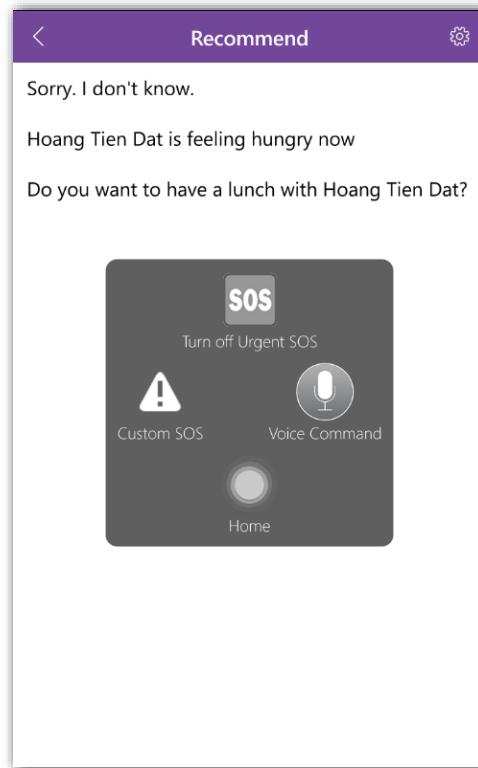
#### 4.5.14. Màn hình tương tác giọng nói



Hình 4.46 Giao diện chức năng tương tác giọng nói

Người dùng có thể kích hoạt chức năng tương tác giọng nói bằng lệnh hoặc ấn nút Voice Command trong menu để kích hoạt. Sau đó ra lệnh theo ý người dùng để ứng dụng thực hiện và hiển thị kết quả trả về nhanh nhất.

#### 4.5.15. Màn hình “Recommend”



Hình 4.47 Giao diện chức năng đề xuất gợi ý

Người dùng có thể kiểm tra trạng thái tạm thời của bạn bè, ứng dụng sẽ xem xét dựa trên vị trí và trạng thái và từ đó đưa ra các đề xuất cụ thể cho người dùng. Ví dụ, nếu bạn của người dùng có trạng thái đang đói giống họ, thì ứng dụng có thể đưa ra gợi ý mời người bạn đó đi ăn.

## **Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **5.1. Kết luận**

Trong khoảng thời gian 3 tháng, nhóm đã thực hiện được:

- Xây dựng một hệ thống mạng xã hội có hiệu suất cao, độ trễ thấp, và có thể đáp ứng được lượng lớn người dùng, bằng cách: Sử dụng các công nghệ và hệ quản trị CSDL có hiệu suất cao và khả năng đáp ứng trên vài triệu operation một giây như Couchbase Server và Redis, xây dựng hệ thống có thể dễ dàng mở rộng (scale) theo chiều ngang để tăng khả năng xử lý cho hệ thống khi cần thiết, tất cả các thành phần có thể dễ dàng phân tán ra nhiều server riêng biệt, mỗi server đáp ứng một nhiệm vụ riêng, giúp tăng tối đa khả năng đáp ứng lượng lớn người dùng.
- Xây dựng hệ thống chat real-time theo mô hình kiến trúc hướng sự kiện bất đồng bộ (Asynchronous event-driven architecture) có đầy đủ tính năng của một hệ thống server chat như: Lấy danh sách bạn bè đang online, thời gian online lần cuối, lấy tin nhắn cuối cùng của tất cả các cuộc hội thoại, gửi tin nhắn tức thì bằng text, hình ảnh, file, hoặc giọng nói của người dùng, hiển thị trạng thái “đang nhập...” khi đối phương đang nhập tin nhắn... Hệ thống đảm bảo được hiệu suất cao nhờ việc giao tiếp trực tiếp với client bằng giao thức TCP, áp dụng kiến trúc hướng sự kiện bất đồng bộ giúp giảm thiểu đi nhiều chi phí so với mô hình “Một thread xử lý một connection (blocking socket)” truyền thống, và sử dụng Redis để cache lịch sử tin nhắn của người dùng (thay vì sử dụng các CSDL quan hệ để lưu lịch sử tin nhắn) để giảm thiểu thời gian thực hiện các tác vụ CRUD với CSDL.
- Xây dựng một mạng xã hội hoàn chỉnh có đầy đủ các tính năng cơ bản của một mạng xã hội thường thấy.
- Tích hợp các tính năng tương tác bằng giọng nói vào mạng xã hội.
- Tích hợp các tính năng giúp tăng tính tương tác thực tế và khuyến khích người dùng tương tác thực tế.

## **5.2. Hướng phát triển**

Trong tương lai, ngoài việc khắc phục các hạn chế, cải tiến về công nghệ, xây dựng thêm các tính năng còn thiếu của hệ thống, khóa luận có thể phát triển thêm một số tính năng khác mà thời gian thực hiện khóa luận chưa cho phép như:

- Mạng xã hội có thể kết hợp thêm với các thiết bị mà người dùng sử dụng như Smart glass.
- Nhận dạng người nào đó thông qua gương mặt. Nhận dạng gương mặt, tự động phân tích tìm kiếm và vào profile của một người. Xem người đó có liên hệ với mình không. Hoặc thông qua nhận dạng gương mặt để biết được tâm trạng, trạng thái của người dùng gần đây.
- Nhận dạng tâm trạng tự động thông qua giọng nói.
- Phân tích các trạng thái để biết được trạng thái của người dùng qua các ký tự trong trạng thái, từ đó xây dựng biểu đồ trạng thái của người dùng trong khoảng thời gian người dùng chọn muốn xem; hoặc từ trạng thái của người dùng đó mà thay đổi giọng nói tương tác cho phù hợp với tâm trạng, độ tuổi,...
- Phát triển mở rộng thêm phần tương tác giọng nói với các ngôn ngữ khác ngoài tiếng Anh.

## **TÀI LIỆU THAM KHẢO**

- [1] Disadvantages of Social Networking: Surprising Insights from Teens  
<http://www.rootsofaction.com/disadvantages-of-social-networking/>
- [2] Facebook is for grandparents: What we need in a next-gen social network  
<http://thenextweb.com/socialmedia/2013/11/24/facebook-grandparents-need-next-gen-social-network/>
- [3] The Next Generation of Social Networks  
<http://www.hbs.edu/news/articles/Pages/next-generation-of-social-networks.aspx>
- [4] What will be the Next generation of Social Networking?  
<https://www.quora.com/What-will-be-the-Next-generation-of-Social-Networking>
- [5] Lợi ích và tác hại của việc sử dụng Facebook  
<http://thuocgiaidocgan.net/loi-ich-va-tac-hai-cua-viec-su-dung-facebook.html>
- [6] Những thống kê đáng chú ý về mạng xã hội năm 2014  
<http://ictnews.vn/internet/nhung-thong-ke-dang-chu-y-ve-mang-xa-hoi-nam-2014-116458.ict>
- [7] Tổng quan thị trường Internet Việt Nam  
<http://www.medialink.com.vn/online-marketing/tong-quan-thi-truong-internet-viet-nam-2015.html>
- [8] Thông kê về mạng xã hội tại Việt Nam 2014  
<https://andytruongblog.wordpress.com/2014/10/21/thong-ke-ve-social-media-mang-xa-hoi-tai-viet-nam-2014/>
- [9] Vì sao Facebook lại được ưa thích như thế?  
<http://www.techz.vn/vi-sao-facebook-lai-duoc-ua-thich-nhu-the-ylt40348.html>
- [10] Xu hướng ứng dụng công nghệ giọng nói tại Việt Nam  
<http://chungta.vn/tin-tuc/cong-nghe/xu-huong-ung-dung-cong-nghe-giong-noi-tai-viet-nam-42639.html>