

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



BÀI TẬP LỚN
MÔN: AN TOÀN VÀ BẢO MẬT THÔNG TIN
ĐỀ TÀI
RSA VÀ SẢN PHẨM DEMO ỨNG DỤNG CỤ THỂ SỬ DỤNG
MÃ HÓA RSA TRONG TRUYỀN TIN

GVHD:	TS. Lê Thị Anh
Lớp:	20231IT6001001
Nhóm:	Nhóm 2
Thành viên nhóm:	1. Nguyễn Duy Sơn - 2021601496 2. Trịnh Tuấn Đạt - 2021601016 3. Đinh Tuấn Đạt - 2021604175 4. Nguyễn Ngọc Tú Tài - 2021604019 5. Hoàng Văn Vương - 2020602011

Hà Nội - Năm 2023

LỜI CẢM ƠN

Đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến Trường Đại học Công nghiệp Hà Nội đã đưa môn học An toàn và bảo mật thông tin vào chương trình giảng dạy. Đặc biệt, nhóm em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – Cô Lê Thị Anh đã dạy dỗ, truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học An toàn và bảo mật thông tin của cô, nhóm em đã trang bị cho mình nhiều kỹ năng, kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để em có thể vững bước sau này.

Bộ môn An toàn và bảo mật thông tin là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù nhóm em đã cố gắng hết sức nhưng chắc chắn bài tiểu luận khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy cô xem xét và góp ý để bài tiểu luận của nhóm em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

Lời Cảm Ơn	2
Mục Lục	3
Danh Mục Hình Ảnh.....	6
Lời Mở Đầu.....	8
Chương I: Tổng Quan Về Hệ Mã RSA.....	9
1.1. Giới thiệu về mật mã học.....	9
1.2. Tổng quan về đề tài	10
1.3. Ý tưởng chọn đề tài	11
1.4. Nội dung nghiên cứu và kiến thức cần thiết	11
1.5. Lĩnh vực nghiên cứu	11
Chương 2: Cơ Sở Lý Thuyết, Kiến Thức.....	12
2.1. Giới thiệu	12
2.2. Kiến thức Cần Thiết.....	12
2.2.1. Hệ mã RSA	12
2.2.1.1. Thuật toán Euclid mở rộng.....	12
2.2.1.2. Hàm số Euler	19
2.2.1.3. Định lý Fermat	21
2.2.1.4. Thuật toán Miller - Rabin	22
2.2.1.5. Mã hóa và giải mã RSA	23
2.2.2. JSON Web Token	26
2.2.2.1. Khái niệm Json Web Token	26
2.2.2.2. Cấu trúc của một JWT	26
2.2.2.3. Hệ thống sử dụng JWT	28
2.2.3. TCP Protocol + WebSocket.....	29
2.2.3.1. TCP Protocol	29
2.2.3.2. WebSocket	31
2.2.4. Restful API	34
2.2.4.1. Restful API là gì?	34
2.2.4.2. Cách thức hoạt động	35
2.2.4.3. Ưu điểm của RESTful API	35
2.2.5. Mô hình client - server	36

2.2.5.1. Giới thiệu	36
2.2.5.2. Khái niệm về mô hình Client – Server	37
2.2.5.3. Ưu điểm và nhược điểm của mô hình Client – Server	38
2.2.6. JavaScript + React TypeScript (Frontend)	40
2.2.6.1. Javascript.....	40
2.2.6.2. JavaScript Framework là gì?.....	41
2.2.6.3. ReactJs TypeScript.....	44
2.2.6.4. Tạo dự án ReactJS + Typescript.....	46
2.2.7. Java Spring Boot (Backend)	47
2.2.7.1. Giới thiệu về Back-end.....	47
2.2.7.2. Java Spring Boot.....	48
Chương 3: Thực Nghiệm Đề Tài.....	52
3.1. Giới thiệu ứng dụng	52
3.1.1. Kiến thức cần thiết	52
3.1.2. Tính Năng Chính của Ứng Dụng	53
3.2. Mô hình hóa dữ liệu	54
3.2.1. Các yêu cầu về dữ liệu	54
3.2.2. Thiết kế bảng.....	55
3.2.3. Biểu đồ thực thể liên kết mức vật lý.....	58
3.3. Mô hình hóa chức năng	58
3.3.1. Usecase chính.....	58
3.3.2. Đặc tả chi tiết các usecase	60
3.3.2.1. Đặc tả usecase Đăng nhập.....	60
3.3.2.2. Đặc tả usecase Đăng ký	63
3.3.2.3. Đặc tả usecase Đăng xuất	66
3.3.2.4. Đặc tả usecase Xem các cuộc trò chuyện tham gia	68
3.3.2.5. Đặc tả usecase Gửi tin nhắn đến cuộc trò chuyện	70
3.3.2.6. Đặc tả usecase Gửi lời mời kết bạn.....	73
3.3.2.7. Đặc tả usecase Chấp nhận lời mời kết bạn	76
3.3.2.8. Đặc tả usecase Tạo cuộc trò chuyện nhóm	78
3.3.2.9. Đặc tả usecase Tìm kiếm bạn bè.....	81
3.3.2.10. Đặc tả usecase Tìm kiếm cuộc trò chuyện đã tham gia	84
3.3.2.11. Đặc tả usecase Cập nhật thông tin cá nhân	86
3.3.2.12. Đặc tả usecase Xem trang cá nhân bạn bè.....	89

3.3.2.13. Đặc tả usecase Cập nhật cuộc trò chuyện	92
3.4. Ứng dụng hệ mã RSA trong ứng dụng Chat App RSA.....	94
3.4.1. Khởi chạy ứng dụng	94
3.4.2. Hoạt động của ứng dụng khi đăng nhập.....	97
3.4.2. Hoạt động của ứng dụng khi chấp nhận lời mời kết bạn	103
3.4.3. Hoạt động của ứng dụng khi gửi và nhận tin nhắn	105
Chương 4: Kết Luận	112
Nội dung đã làm trong bài tập lớn.....	112
Bài học kinh nghiệm	112
Tài Liệu Tham Khảo.....	115

DANH MỤC HÌNH ẢNH

Hình 2.1. Hình ảnh minh họa code mã hóa RSA	24
Hình 2.2. Hình ảnh minh họa code giải mã RSA	25
Hình 2.3. Hình ảnh minh họa ví dụ JWT.....	26
Hình 2.4. Hình ảnh minh họa cấu trúc JWT	27
Hình 2.5. Hình ảnh minh họa cách thức hoạt động của JWT.....	28
Hình 2.6. Hình ảnh minh họa nhiệm vụ của TCP.....	30
Hình 2.7. Hình ảnh minh họa cách thức hoạt động của TCP.....	30
Hình 2.8. Hình ảnh minh họa websocket.....	32
Hình 2.9. Hình ảnh minh họa cách thức hoạt động của RESTful API	35
Hình 2.10. Hình ảnh minh họa mô hình mạng Client – Server.....	37
Hình 2.11. Hình ảnh minh họa nguyên tắc hoạt động của mô hình client-server	37
Hình 2.12. Hình ảnh minh họa logo Framework ReactJS	42
Hình 2.13. Hình ảnh minh họa các ưu điểm của ReactJS	43
Hình 2.14. Hình ảnh minh họa các thành phần của TS	44
Hình 2.15. Hình ảnh minh họa tạo dự án ReactJS	47
Hình 2.16. Hình ảnh minh họa java spring boot.....	49
Hình 2.17. Hình ảnh minh họa đặc tính cơ bản của spring boot	50
Hình 3.1. Hình ảnh minh họa bảng tbl_user	55
Hình 3.2. Hình ảnh minh họa bảng tbl_rsa_key.....	55
Hình 3.3. Hình ảnh minh họa bảng tbl_user_room	55
Hình 3.4. Hình ảnh minh họa bảng tbl_room.....	56
Hình 3.5. Hình ảnh minh họa bảng tbl_room_type	56
Hình 3.6. Hình ảnh minh họa bảng tbl_friend	56
Hình 3.7. Hình ảnh minh họa bảng tbl_message.....	57
Hình 3.8. Hình ảnh minh họa bảng tbl_message_type	57
Hình 3.9. Biểu đồ thực thể liên kết mức vật lý	58
Hình 3.10. Biểu đồ usecase chính.....	60
Hình 3.11. Biểu đồ trình tự usecase đăng nhập	62
Hình 3.12. Biểu đồ lớp usecase đăng nhập.....	63
Hình 3.13. Biểu đồ trình tự usecase đăng ký.....	65
Hình 3.14. Biểu đồ lớp usecase đăng ký	66
Hình 3.15. Biểu đồ trình tự usecase đăng xuất.....	68
Hình 3.16. Biểu đồ lớp usecase đăng xuất.....	68
Hình 3.17. Biểu đồ trình tự usecase xem các cuộc trò chuyện đã tham gia.....	69
Hình 3.18. Biểu đồ lớp usecase xem các cuộc trò chuyện đã tham gia	70
Hình 3.19. Biểu đồ trình tự usecase gửi tin nhắn	72
Hình 3.20. Biểu đồ lớp usecase gửi tin nhắn.....	73
Hình 3.21. Biểu đồ trình tự usecase gửi lời mời kết bạn.....	75
Hình 3.22. Biểu đồ lớp usecase gửi lời mời kết bạn.....	76
Hình 3.23. Biểu đồ trình tự usecase chấp nhận kết bạn	77
Hình 3.24. Biểu đồ lớp usecase chấp nhận kết bạn	78
Hình 3.25. Biểu đồ trình tự usecase tạo cuộc trò chuyện nhóm	80

Hình 3.26. Biểu đồ lớp usecase tạo cuộc trò chuyện nhóm.....	81
Hình 3.27. Biểu đồ trình tự usecase tìm kiếm bạn bè	83
Hình 3.28. Biểu đồ lớp usecase tìm kiếm bạn bè	83
Hình 3.29. Biểu đồ trình tự usecase tìm kiếm cuộc trò chuyện đã tham gia	85
Hình 3.30. Biểu đồ lớp usecase tìm kiếm cuộc trò chuyện đã tham gia.....	86
Hình 3.31. Biểu đồ trình tự usecase cập nhật thông tin cá nhân.....	88
Hình 3.32. Biểu đồ lớp usecase cập nhật thông tin cá nhân.....	89
Hình 3.33. Biểu đồ trình tự usecase xem trang cá nhân bạn bè	91
Hình 3.34. Biểu đồ lớp usecase xem trang cá nhân bạn bè	91
Hình 3.35. Biểu đồ trình tự usecase cập nhật cuộc trò chuyện.....	93
Hình 3.36. Biểu đồ lớp usecase cập nhật cuộc trò chuyện	94
Hình 3.37. Hình ảnh mô tả file application.properties	95
Hình 3.38. Hình ảnh mô tả chạy lệnh yarn trong terminal.....	95
Hình 3.39. Hình ảnh mô tả chạy lệnh yarn start trong terminal	96
Hình 3.40. Giao diện trang đăng nhập.....	96
Hình 3.41. Hình ảnh mô tả file ChatApplication.java.....	97
Hình 3.42. Hình ảnh mô tả khi run file ChatApplication.java	97
Hình 3.43. Hình ảnh mô tả chuỗi JWT mà Backend trả về	98
Hình 3.44. Hình ảnh mô tả bộ 3 số n, e, d được tạo phía client.....	98
Hình 3.45. Hình ảnh mô tả public key khi được gửi sẽ cập nhật vào trường publicKeyId trong tbl_user	99
Hình 3.46. Hình ảnh mô tả đoạn mã tự động sinh khóa khi đăng nhập thành công.....	100
Hình 3.47. Hình ảnh mô tả đoạn mã tính nghịch đảo của a mod m	101
Hình 3.48. Hình ảnh mô tả đoạn mã kiểm tra số nguyên tố Miller-Rabin	102
Hình 3.49. Hình ảnh mô tả đoạn mã tìm số nguyên tố sắp xỉ với num.....	103
Hình 3.50. Hình ảnh mô tả đoạn mã tạo một phòng chat mới	104
Hình 3.51. Hình ảnh mô tả đoạn mã khởi tạo cặp khóa.....	105
Hình 3.52. Giao diện trang chat.....	106
Hình 3.53. Hình ảnh mô tả đoạn mã người dùng gửi tin nhắn đến server sau khi mã hóa tin nhắn bằng rsa	107
Hình 3.54. Hình ảnh mô tả hàm mã hóa phía client.....	108
Hình 3.55. Hình ảnh mô tả dữ liệu được gửi đi khi mã hóa	108
Hình 3.56. Hình ảnh mô tả dữ liệu nhận được phía Backend.....	108
Hình 3.57. Hình ảnh mô tả đoạn mã giải mã tin nhắn nhận được từ client	109
Hình 3.58. Hình ảnh mô tả việc mã hóa tin nhắn theo public key của từng người dùng trong room chat	110
Hình 3.59. Hình ảnh mô tả đoạn mã giải mã tin nhắn	111

LỜI MỞ ĐẦU

Trong bối cảnh hiện nay, internet đóng vai trò quan trọng như một công cụ mạnh mẽ, mang lại sự thuận tiện trong việc trao đổi thông tin, tổ chức dữ liệu, và tìm kiếm thông tin. Tuy nhiên, điều này đồng nghĩa với việc chúng ta phải đối mặt với những thách thức về bảo mật thông tin, nhất là khi các dịch vụ trực tuyến đòi hỏi sự bảo mật cao với các yếu tố như tính bí mật, tính toàn vẹn, và tính xác thực. Trong ngữ cảnh này, sự phát triển của Công nghệ thông tin đồng điều với sự gia tăng của các rủi ro an ninh mạng.

Để đối phó với những thách thức này, đề án này tập trung vào việc xây dựng một chương trình chat thời gian thực, nơi mà việc bảo mật tin nhắn được đặt lên hàng đầu, sử dụng thuật toán RSA. Thuật toán này, với cơ sở toán học vững chắc, đảm bảo tính an toàn và bí mật cho thông tin truyền tải qua mạng.

Chúng tôi đã tích hợp các kiến thức được học và nghiên cứu từ nhiều nguồn tài liệu, cùng với sự hỗ trợ và chỉ dẫn tận tình của cô Lê Thị Anh, để triển khai thành công đề tài này. Tuy nhiên, trong quá trình nghiên cứu và thực hiện, chúng tôi đã phải đối mặt với nhiều thách thức, và nhận ra rằng sự hiểu biết của chúng tôi vẫn còn hạn chế, dẫn đến một số điểm chưa hoàn thiện trong sản phẩm cuối cùng.

Chúng tôi mong rằng sự đóng góp ý kiến và nhận xét của cô sẽ giúp chúng tôi hiểu rõ hơn về những khía cạnh còn thiếu sót, từ đó hoàn thiện và nâng cao chất lượng của đề tài. Đồng thời, chúng tôi hy vọng rằng công trình nghiên cứu này sẽ là một bước đầu tích cực, góp phần vào lĩnh vực bảo mật thông tin trong các ứng dụng truyền thông hiện đại.

CHƯƠNG I: TỔNG QUAN VỀ HỆ MÃ RSA

1.1. Giới thiệu về mật mã học

Mật mã học (Cryptography) là ngành khoa học nghiên cứu về việc đảm bảo an toàn thông tin. Mật mã học gắn liền với quá trình mã hóa nghĩa là chuyển đổi thông tin từ dạng “có thể hiểu được” thành dạng “không thể hiểu được” và ngược lại quá trình giải mã. Mật mã học giúp đảm bảo những tính chất sau cho thông tin:

- Tính bí mật (confidentiality): thông tin chỉ được tiết lộ cho những ai được phép.
- Tính toàn vẹn (integrity): thông tin không thể bị thay đổi mà không bị phát hiện
- Tính xác thực (authentication): người gửi (hoặc người nhận) có thể chứng minh đúng họ.
- Tính chống chối bỏ (non-repudiation): người gửi hoặc nhận sau này không thể chối bỏ việc đã gửi hoặc nhận thông tin.

Để thực hiện điều này chúng ta áp dụng các biện pháp xác thực và mã hóa. Và mật mã học là nghiên cứu về vấn đề mã hóa. Mã hóa dữ liệu là chuyển dữ liệu từ dạng này sang dạng khác hoặc sang dạng code mà chỉ có người có quyền truy cập vào khóa giải mã hoặc có mật khẩu mới có thể đọc được nó. Dữ liệu được mã hóa thường gọi là ciphertext, dữ liệu thông thường, không được mã hóa thì gọi là plaintext. Dữ liệu hoặc plaintext được mã hóa với một thuật toán mã hóa và một key mã hóa, tạo ra một ciphertext. Dữ liệu sau khi mã hóa chỉ có thể xem được dưới dạng ban đầu nếu giải mã với các key chính xác. Có hai loại mã hóa dữ liệu chính tồn tại: mã hóa bất đối xứng còn được gọi là mã hóa công khai, và mã hóa đối xứng. Thuật toán mã hóa bất đối xứng còn được gọi là mã hóa khóa công khai, sử dụng hai khóa khác nhau: khóa công khai và khóa riêng tư. Mục đích của việc mã hóa dữ liệu là bảo vệ dữ liệu số khi nó được lưu trữ trên

các hệ thống máy tính và truyền qua Internet hay các mạng máy tính khác, là ngăn ngừa việc tấn công đánh cắp dữ liệu trái phép hoặc phòng ngừa việc mất mát dữ liệu khi bị tấn công vật lý như trộm đĩa cứng, máy tính xách tay hay thậm chí đột nhập vào hệ thống vẫn không thể xem được dữ liệu riêng tư, bí mật đã được bảo vệ bằng các thuật toán mã hóa mạnh mẽ.

1.2. Tổng quan về đề tài

Mã hóa RSA chính là một thuật toán hay còn gọi là hệ mã hóa bất đối xứng có phạm vi ứng dụng rộng rãi và phổ biến, đặc biệt người ta sử dụng RSA rất nhiều ở công tác mã hóa, thiết lập chữ ký điện tử, với vai trò là một mã hóa khóa công khai. Bất cứ ai cũng có thể dùng khóa công khai để có thể mã hóa được nguồn dữ liệu muốn gửi đi thế nhưng để giải mã được dữ liệu gửi đi đó thì buộc phải có sự hỗ trợ của khóa bí mật.

Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

Không giống với loại mã hóa có khóa đối xứng, loại khóa bí mật của RSA tuyệt đối không truyền được thông tin ra bên ngoài ngay cả khi có thiết bị nghe trộm thì nếu đối tượng xấu không có khóa bí mật cũng sẽ không thể giải mã được thông tin đó.

Như vậy rõ ràng với 2 tính năng mã hóa và giải mã tối ưu đến tuyệt đối ở một phương trình bất đối xứng như vậy cho nên giá trị của RSA vô cùng lớn, nó sẽ được sử dụng ở hầu hết mọi trường hợp cần bảo mật thông tin. Giao thức SSL hay TLS, HTTPS cùng với chứng chỉ điện tử đều sử dụng RSA.

1.3. Ý tưởng chọn đề tài

Trong quá trình học tập bộ môn An toàn bảo mật thông tin, chúng em đã được học những thuật toán khác nhau, các cách để mã hóa và giải mã thông tin. Đề tài này chúng em được cô giáo phân công thực hiện, chúng em dựa trên cơ sở kiến thức đã học, cùng với đó là các nguồn khác nhau trên internet để hoàn thiện đề tài.

1.4. Nội dung nghiên cứu và kiến thức cần thiết

Để hoàn thành đề tài này, chúng em đã vận dụng kiến thức của các thuật toán đã học và tìm hiểu về khóa công khai, khóa riêng tư, sinh khóa, mã hóa và giải mã RSA

Những kiến thức cần thiết để hoàn thành đề tài :

1. Ứng dụng chat client server
2. Restful Api
3. React typescript frontend
4. Java spring boot backend
5. Websocket
6. JSONWebtoken
7. Kiến thức về phương pháp mã hóa, giải mã RSA

1.5. Lĩnh vực nghiên cứu

Chủ đề: Xây dựng sản phẩm demo ứng dụng cụ thể sử dụng mã hóa RSA trong truyền tin thuộc lĩnh vực nghiên cứu và an toàn bảo mật thông tin giữa trên các nền tảng ngôn ngữ: Java, Javascript...

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT, KIẾN THỨC

2.1. Giới thiệu

- Nhiệm vụ, công việc chính cần thực hiện
- Đề tài nghiên cứu: Xây dựng chương trình mã hóa và giải mã RSA
- Kiến thức cần thiết cho việc phát triển ứng dụng trong đề tài:
 1. Hệ mã RSA
 2. JSONWebToken
 3. TCP Protocol + WebSocket
 4. Restful API
 5. Mô hình client - server
 6. JavaScript + React TypeScript (Frontend)
 7. Java Spring Boot (Backend)
- Hình thức sản phẩm: Sản phẩm ứng dụng.
- Kết quả đạt được:
 - + Nghiên cứu, cài đặt thuật toán ứng dụng.
 - + Cài đặt chương trình demo.

2.2. Kiến thức Cần Thiết

2.2.1. Hệ mã RSA

2.2.1.1. Thuật toán Euclid mở rộng

Giải thuật Euclid mở rộng được sử dụng để giải một phương trình vô định nguyên (còn được gọi là phương trình Di-ô-phăng) có dạng: $ax + by = c$

Trong đó a, b, c là các hệ số nguyên, x, y là các ẩn nhận giá trị nguyên. Điều kiện cần và đủ để phương trình này có nghiệm (nguyên) là $\text{UCLN}(a,b)$ là ước của c . Khẳng định này dựa trên một mệnh đề sau:

Nếu $d = \text{UCLN}(a,b)$ thì tồn tại các số nguyên x, y sao cho $ax + by = d$

Cơ sở lý thuyết của giải thuật

Giải thuật Euclid mở rộng kết hợp quá trình tìm ƯCLN (a, b) trong thuật toán Euclid với việc tìm một cặp số x, y thỏa mãn phương trình Di-ô-phăng. Giả sử cho hai số tự nhiên a, b , ngoài ra $a > b > 0$. Đặt $r_0 = a$, $r_1 = b$, chia cho r_0 được số dư r_2 và thương số nguyên q_1 . Nếu $r_2 = 0$ thì dừng, nếu r_2 khác không, chia r_1 cho r_2 được số dư r_3, \dots . Vì dãy các r_i là giảm thực sự nên sau hữu hạn bước ta được số dư $r_{m+2} = 0$.

$$r_0 = q_1 * r_1 + r_2, 0 < r_2 < r_1;$$

$$r_1 = q_2 * r_2 + r_3, 0 < r_3 < r_2;$$

....;

$$r_{m-1} = q_m * r_m + r_{m+1}, 0 < r_{m+1} < r_m$$

$$r_m = q_{m+1} * r_{m+1}$$

trong đó số dư cuối cùng khác 0 là $r_{m+1} = d$. Bài toán đặt ra là tìm x, y sao cho

$$a * x + b * y = r_{m+1} (=d)$$

Để làm điều này, ta tìm x, y theo công thức truy hồi, nghĩa là sẽ tìm x_i và y_i sao cho:

$$a * x_i + b * y_i = r_{m+1} (=d) \text{ với } i = 0, 1, \dots$$

Ta có

$$a * 1 + b * 0 = a = r_0 \text{ và } a * 0 + b * 1 = b = r_1 \text{ nghĩa là } x_0 = 1, x_1 = 0 \text{ và } y_0 = 0, y_1 = 1.$$

(1)

Tổng quát, giả sử có

$$a * x_i + b * y_i = r_i \text{ với } i = 0, 1, \dots$$

$$a * x_{i+1} + b * y_{i+1} = r_{i+1} \text{ với } i = 0, 1, \dots$$

Khi đó từ

$$r_i = q_{i+1} * r_{i+1} + r_{i+2}$$

suy ra

$$r_i - q_{i+1} * r_{i+1} = r_{i+2}$$

$$(a * x_i + b * y_i) - q_{i+1} * (a * x_{i+1} + b * y_{i+1}) = r_{i+2}$$

$$a * (x_i - q_{i+1} * x_{i+1}) + b * (y_i - q_{i+1} * y_{i+1}) = r_{i+2}$$

từ đó, có thể chọn

$$x_{i+2} = x_i - q_{i+1} * x_{i+1} \quad (2)$$

$$y_{i+2} = y_i - q_{i+1} * y_{i+1} \quad (3)$$

Khi $i = m - 1$ ta có được x_{m+1} và y_{m+1} . Các công thức (1), (2), (3) là công thức truy hồi để tính x , y .

Giải thuật

{Thuật toán Euclide: a , b không đồng thời bằng 0, trả về $\text{gcd}(a, b)$ }

function $\text{gcd}(a, b)$;

begin

while $b \neq 0$ **do**

begin

$r := a \bmod b$; $a := b$; $b := r$;

end;

Result := a ;

end;

{Thuật toán Euclide mở rộng: a , b không đồng thời bằng 0, trả về cặp (x, y)

sao cho $a * x + b * y = \text{gcd}(a, b)$

Về tư tưởng là ghép quá trình tính cặp số (x, y) vào trong vòng lặp chính của thuật toán Euclide.}

function $\text{Extended_gcd}(a, b)$;

begin

$(x_a, y_a) := (1, 0)$;

$(x_b, y_b) := (0, 1)$;

while $b \neq 0$ **do**

```

begin
  q:= a div b;
  r:= a mod b; a:= b; b:= r; //Đoạn này giống thuật toán Euclide.
  (xr, yr):= (xa, ya) - q * (xb, yb); //Hiệu là: (xr, yr):= (xa, ya) "mod" (xb,
yb);
  (xa, ya):= (xb, yb);
  (xb, yb):= (xr, yr);
end;
Result:= (xa, ya);
end;

```

Giải thuật sau chỉ thực hiện với các số nguyên $a > b > 0$, biểu diễn bằng giải mã:

Sub Euclid_Extended(a,b)

Dim x0, x, y, y1 **As Single**

x0=1: x1=0: y0=0: y1=1

While b>0

r= a **mod** b

if r=0 then Exit While

q= a / b

x= x0-x1*q

y= y0-y1*q

a=b

b=r

x0=x1

x1=x

y0=y1

y1=y

Wend

Me.Print d:=b, x, y

code

End Sub

Ví dụ: Với $a=29$, $b=8$, giải thuật trải qua các bước như sau:

Bước										
0	29	8	5	3	1	0	1	0	1	-3
1	8	5	3	1	0	1	-1	1	-3	4
2	5	3	2	1	1	-1	2	-3	4	-7
3	3	2	1	1	-1	2	-3	4	-7	11
4	2	1	0	2						

Kết quả thuật toán cho đồng thời $d = \text{UCLN}(29,8) = 1$ và $x = -3$, $y=11$.

Dễ dàng kiểm tra hệ thức $29 * (-3) + 8 * 11 = 1$

Áp dụng giải thuật Euclid mở rộng tìm số nghịch đảo trong vành Z_m

Số nghịch đảo trong vành Z_m

Trong lý thuyết số, vành Z_m được định nghĩa là vành thương của Z với quan hệ đồng dư theo modulo m (là quan hệ tương đương) mà các phần tử của nó là các lớp đồng dư theo modulo m (m là số nguyên dương lớn hơn 1). Ta cũng có thể xét Z_m chỉ với các đại diện của nó. Khi đó

$$Z_m = \{0, 1, \dots, m-1\}$$

Phép cộng và nhân trong Z_m là phép toán thông thường được rút gọn theo modulo m :

$$a + b = (a + b) \bmod m$$

$$a * b = (a * b) \bmod m$$

Phần tử a của Z_m được gọi là khả nghịch trong Z_m hay khả nghịch theo

modulo m nếu tồn tại phần tử a' trong Z_m sao cho $a * a' = 1$ trong Z_m hay $a * a' = 1 \pmod{m}$. Khi đó a' được gọi là nghịch đảo modulo m của a . Trong lý thuyết số đã chứng minh rằng, số a là khả nghịch theo modulo m khi và chỉ khi ƯCLN của a và m bằng 1.

Khi đó tồn tại các số nguyên x, y sao cho:

$$m * x + a * y = 1$$

Đẳng thức này lại chỉ ra y là nghịch đảo của a theo modulo m . Do đó có thể tìm được phần tử nghịch đảo của a theo modulo m nhờ thuật toán Euclid mở rộng khi chia m cho a .

Giải thuật

// $a, m > 0$. Trả về $a^{-1} \pmod{m}$, $\gcd(a, m)$ phải bằng 1, chú ý là ta không cần quan tâm y khi giải pt diophante $a * x + m * y = 1$

function ModuloInverse(a, m);

begin

$xa := 1$; $xm := 0$;

while $m \neq 0$ **do**

 begin

$q := a \text{ div } m$;

$xr := xa - q * xm$;

$xa := xm$;

$xm := xr$;

$r := a \text{ mod } m$;

$a := m$;

$m := r$;

end;

 Result := xa ;

end;

Giải thuật sau chỉ thực hiện với các số nguyên $m > a > 0$, biểu diễn bằng giả mã:

Procedure Euclid_Extended (a,m)

int, $y_0 := 0, y_1 := 1;$

While $a > 0$ **do** {

$r := m \bmod a$

if $r = 0$ **then** Break

$q := m \div a$

$y := y_0 - y_1 * q$

$y_0 := y_1$

$y_1 := y$

$m := a$

$a := r$

}

If $a > 1$ **Then Return** "A không khả nghịch theo môđun m"

else Return " Nghịch đảo modulo m của a là y"

Ví dụ: Tìm số nghịch đảo (nếu có) của 30 theo môđun 101

Bước i	m	A	r	q	y_0	y_1	y
0	101	30	11	3	0	1	-3
1	30	11	8	2	1	-3	7
2	11	8	3	1	-3	7	-10
3	8	3	2	2	7	-10	27
4	3	2	1	1	-10	27	-37
5	2	1	0

Kết quả tính toán trong bảng cho ta -37 . Lấy số đối của 37 theo modun 101 được 64. Vậy $30^{-1} \bmod 101 = 64$.

Ứng dụng

Số nghịch đảo theo môđun được ứng dụng nhiều trong việc giải phương trình đồng dư, trong lý thuyết mật mã.

2.2.1.2. Hàm số Euler

Cho số nguyên dương n , số lượng các số nguyên dương bé hơn n và nguyên tố cùng nhau với n được ký hiệu $f(n)$ và gọi là hàm Euler.

Nhận xét: Nếu p là số nguyên tố, thì $f(p) = p - 1$

Ví dụ: Tập các số nguyên không âm nhỏ hơn 7 là $Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$.

Do 7 là số nguyên tố, nên tập các số nguyên dương nhỏ hơn 7 và nguyên tố cùng nhau với 7 là $Z_7^* = \{1, 2, 3, 4, 5, 6\}$.

Khi đó $|Z| = f(p) = p - 1 = 7 - 1 = 6$

* Định lý về hàm Euler: Nếu n là tích của hai số nguyên tố p, q thì

$$f(n) = f(p) \cdot f(q) = (p-1) \cdot (q-1)$$

* Quan hệ Đồng dư

Khái niệm: Cho hai số nguyên a, b, m ($m > 0$). Ta nói rằng a và b “đồng dư” với nhau theo modulo m , nếu chia a và b cho m , ta nhận được cùng một số dư.

Ký hiệu: $a \equiv b \pmod{m}$.

Ví dụ: $17 \equiv 5 \pmod{3}$ vì chia 17 và 5 cho 3, được cùng số dư là 2.

Tính chất của quan hệ đồng dư:

- Với mọi số nguyên dương m ta có:

$a \equiv a \pmod{m}$ với mọi a thuộc Z ; (tính chất phản xạ)

$a \equiv b \pmod{m}$ thì $b \equiv a \pmod{m}$; (tính chất đối xứng)

$a \equiv b \pmod{m}$ và $b \equiv c \pmod{m}$ thì $a \equiv c \pmod{m}$; (tính chất bắc cầu)

- Tổng hay hiệu các đồng dư:

$$(a+b) \pmod{m} \equiv [(a \pmod{m}) + (b \pmod{m})] \pmod{m}$$

$$(a-b) \pmod{m} \equiv [(a \pmod{m}) - (b \pmod{m})] \pmod{m}$$

- Tích các đồng dư:

$$(a*b) \pmod n \equiv [(a \pmod n) * (b \pmod n)] \pmod n$$

* Tập thặng dư thu gọn theo modulo

Khái niệm: Kí hiệu $Z_n = \{0, 1, 2, \dots, n-1\}$ là tập các số nguyên không âm $< n$.

Với Z_n và phép cộng (+) lập thành nhóm Cyclic có phần tử sinh là 1, phần tử trung lập $e = 0$, $(Z_n, +)$ gọi là nhóm cộng, đó là nhóm hữu hạn có cấp n .

Với phép (+) là phép cộng thông thường của các số nguyên.

Kí hiệu $Z_n^* = \{x \in Z_n, x \text{ là nguyên tố cùng nhau với } n\}$. Tức là x phải $\neq 0$.

Z_n^* được gọi là Tập thặng dư thu gọn theo mod n , có số phần tử là $\phi(n)$. Z_n^* với phép nhân mod n lập thành một nhóm (nhóm nhân), phần tử trung lập $e = 1$. Tổng quát $(Z_n^*, \text{phép nhân mod } n)$ không phải là nhóm Cyclic. Nhóm nhân Z_n^* là Cyclic chỉ khi n có dạng: $2, 4, p^k$ hay $2p^k$ với p là nguyên tố lẻ.

Ví dụ: Cho $n = 21$, $Z_n^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

* Độ phức tạp của thuật toán

Bài toán được diễn đạt bằng hai phần:

Đầu vào: Các dữ liệu vào của bài toán.

Đầu ra: Các dữ liệu ra của bài toán (kết quả).

Khái niệm Thuật toán: “Thuật toán” được hiểu đơn giản là cách thức để giải một bài toán. Cũng có thể được hiểu bằng hai quan niệm: trực giác hay hình thức như sau:

Một cách trực giác, Thuật toán được hiểu là một dãy hữu hạn các quy tắc (chỉ thị, mệnh lệnh) mô tả một quá trình tính toán, để từ dữ liệu đã cho đầu vào ta nhận được kết quả đầu ra của bài toán.

Một cách hình thức, người ta quan niệm thuật toán là một máy Turing. Thuật toán được chia thành hai loại: Đơn định và không đơn định. Chi phí thời gian của một quá trình tính toán là thời gian cần thiết để thực hiện một quá trình

tính toán. Với thuật toán tựa Algol: Chi phí thời gian là số các phép tính cơ bản thực hiện trong quá trình tính toán.

Chi phí bộ nhớ của một quá trình tính toán là số ô nhớ cần thiết để thực hiện một quá trình tính toán. Gọi A là thuật toán, e là dữ liệu vào của bài toán đã được mã hóa bằng cách nào đó. Thuật toán A tính trên dữ liệu vào e phải trả một giá nhất định. Ta ký hiệu: $t_A(e)$ là giá thời gian và $I_A(e)$ là giá bộ nhớ.

2.2.1.3. Định lý Fermat

“Nếu a là một số nguyên dương và p là một số nguyên tố thì $a^p \equiv a \pmod{p}$.”

Chứng minh 1. Sử dụng phương pháp quy nạp theo

Với $a = 1$ thì mệnh đề luôn đúng.

Giả sử mệnh đề đúng đến a tức là $p \mid a^p - a$.

Ta sẽ chứng minh mệnh đề đúng đến $a + 1$. Thật vậy:

$$(a+1)^p - (a+1) = (a^p - a) + \sum_{k=1}^{p-1} C_p^k a^k$$

Sử dụng $p \mid C_p^k$ với $1 \leq k \leq p-1$ và giả thiết quy nạp ta suy ra

$$p \mid (a+1)^p - (a+1). \text{ Khi đó } (a+1)^p \equiv (a+1) \pmod{p}.$$

Vậy ta hoàn tất chứng minh.

Chứng minh 2. Giả sử rằng $\gcd(a, p) = 1$ và cần chứng minh rằng $a^{p-1} \equiv 1 \pmod{p}$.

Xét các số nguyên $a, 2a, \dots, (p-1)a$ mà các số dư khi chia cho p phân biệt (nếu không thì, với $ia \equiv ja \pmod{p}$ thì $p \mid (i-j)a$ hay là $p \mid i-j$, dấu “=” xảy ra chỉ nếu $i=j$).

$$\text{Do đó } a \cdot (2a) \dots (p-1)a \equiv 1 \cdot 2 \dots (p-1) \pmod{p}.$$

Vì $\gcd(a, (p-1)!) = 1$ nên ta suy ra điều phải chứng minh.

Lưu ý. Định lý này có thể viết gọn dưới dạng: $a^{p-1} \equiv 1 \pmod{p}$

2.2.1.4. Thuật toán Miller - Rabin

Thuật toán Miller-Rabin là một thuật toán xác định tính nguyên tố của một số nguyên dương. Nó được đặt tên theo tên của các nhà toán học Gary L. Miller và Michael O. Rabin, người đã phát triển thuật toán này vào năm 1980.

Thuật toán Miller-Rabin sử dụng phương pháp tổ hợp để xác định tính nguyên tố của một số nguyên dương n . Nó dựa trên một kết quả trong lĩnh vực lý thuyết số được gọi là Định lý Fermat nhỏ (Fermat's Little Theorem).

Định lý Fermat nhỏ (Fermat's Little Theorem) nói rằng nếu p là một số nguyên tố và a là một số tự nhiên không chia hết cho p , thì $a^{(p-1)} \equiv 1 \pmod{p}$.

Thuật toán Miller-Rabin kiểm tra tính nguyên tố của một số nguyên dương n bằng cách chọn ngẫu nhiên một số nguyên a trong khoảng $(2, n-2)$, và kiểm tra xem $a^{(n-1)} \equiv 1 \pmod{n}$. Nếu điều này không đúng, hoặc nếu tồn tại một số nguyên t tương ứng với mỗi bước lặp sao cho $a^{((2^t)*(n-1))} \equiv 1 \pmod{n}$ và $a^{((2^t)*q)} \not\equiv -1 \pmod{n}$ với mọi q trong khoảng $[0, t-1]$, thì số n được coi là không nguyên tố.

Theo định nghĩa, thuật toán Miller-Rabin không thể xác định chính xác tính nguyên tố của một số, mà nó chỉ đưa ra một kết quả xác suất. Tuy nhiên, với mỗi lần kiểm tra, khả năng sai sót giảm đi và có thể điều chỉnh bằng cách tăng số lần lặp lại (k) trong thuật toán.

Thuật toán Miller-Rabin có hiệu suất tốt và được sử dụng rộng rãi trong các ứng dụng liên quan đến mã hóa và bảo mật, như tạo khóa RSA. Nó cho phép kiểm tra tính nguyên tố của các số lớn với độ phức tạp thời gian $O(k*\log(n)^3)$, trong đó k là số lần lặp lại và n là số cần kiểm tra.

Thuật toán Miller-Rabin:

Bước 1: Nhập số nguyên dương n cần kiểm tra tính nguyên tố và số lần lặp lại k .

Bước 2: Chọn ngẫu nhiên một số nguyên a trong khoảng $(2, n-2)$.

Bước 3: Tính $a^{(n-1)} \bmod n$.

Nếu kết quả là 1, ta tiến hành kiểm tra tiếp theo.

Nếu kết quả khác 1, số n không phải là số nguyên tố và ta kết thúc thuật toán.

Bước 4: Với mỗi bước lặp từ 1 đến $k-1$, ta tính tiếp $a^{((2^t)*(n-1))} \bmod n$ và $a^{((2^t)*q)} \bmod n$ với mọi q trong khoảng $[0, t-1]$.

Nếu tồn tại một số tương ứng với mỗi bước lặp mà $a^{((2^t)*(n-1))} \bmod n$ khác 1 và $a^{((2^t)*q)} \bmod n$ khác -1 với mọi q trong khoảng $[0, t-1]$, số n không phải là số nguyên tố và ta kết thúc thuật toán.

Nếu không tồn tại số như trên, ta tiếp tục kiểm tra tiếp theo.

Bước 5: Nếu qua tất cả các bước kiểm tra mà không có kết quả xác định rằng số n không phải là số nguyên tố, ta kết luận rằng số n có khả năng là số nguyên tố.

Thuật toán Miller-Rabin được sử dụng rộng rãi trong các ứng dụng liên quan đến mã hóa và bảo mật. Ví dụ phổ biến nhất là trong thuật toán RSA, một thuật toán mã hóa đối xứng được sử dụng rộng rãi trong việc bảo vệ thông tin truyền qua mạng. Trong quá trình tạo khóa RSA, thuật toán Miller-Rabin được sử dụng để tìm các số nguyên tố lớn và đảm bảo tính nguyên tố của chúng.

2.2.1.5. Mã hóa và giải mã RSA

Mã Hóa

- **Chọn Cặp Khóa:**

Người gửi và người nhận chọn cặp khóa RSA, bao gồm khóa công khai (n , e) và khóa riêng tư (n , d).

n là tích của hai số nguyên tố lớn.

e là số nguyên tố và là một phần của khóa công khai.

- **Chuẩn Bị Chuỗi Cần Mã Hóa:**

Người gửi chuẩn bị một chuỗi ký tự cần gửi đi, ví dụ: "HELLO".

- Chuyển Đổi Ký Tự Thành Số Nguyên (Mã ASCII): Mỗi ký tự trong chuỗi được chuyển đổi thành mã số ASCII tương ứng.
- Mã Hóa Từng Số Nguyên: Mỗi số nguyên (ASCII) trong chuỗi được mã hóa bằng cách sử dụng phép toán modulo và khóa công khai (n, e). Kết quả là một dãy các số nguyên đã được mã hóa.
- Gửi Chuỗi Đã Mã Hóa: Người gửi gửi đi chuỗi đã được mã hóa đến người nhận.

```
// Mã hoá
encryptRSA = (messageContent: string) => {
  try {
    const { n, e } = this.chosenRoom.publicKey;
    let encryptedChar = [];
    for (let i = 0; i < messageContent.length; i++) {
      let charCode = messageContent.charCodeAt(i);
      encryptedChar[i] = RSAService.mod(charCode, e, n);
    }
    let encryptedString = encryptedChar.join(',');
    return (encryptedString);
  }
  catch (error) {
    console.log("Error" + error.message);
  }
};
```

Hình 2.1. Hình ảnh minh họa code mã hóa RSA

Giải Mã RSA

- Chọn Khóa Riêng Tư: Người nhận sử dụng khóa riêng tư (n, d) để giải mã chuỗi đã được mã hóa.
- Chuyển Đổi Chuỗi Đã Mã Hóa Thành Dãy Số Nguyên: Chuỗi đã mã hóa được chuyển đổi thành một dãy các số nguyên.
- Giải Mã Từng Số Nguyên: Mỗi số nguyên trong dãy được giải mã bằng cách sử dụng phép toán modulo và khóa riêng tư (n, d). Kết quả là một dãy các số nguyên đã được giải mã.

- Chuyển Đổi Số Nguyên Thành Ký Tự (Mã ASCII): Mỗi số nguyên trong dãy được chuyển đổi thành ký tự ASCII tương ứng.
- Tạo Chuỗi Kết Quả: Các ký tự ASCII được kết hợp thành một chuỗi kết quả, ví dụ: "HELLO".

```
rsaDecrypt = (messageContent: string, type: string, privateKey: any) => {
  let plaintext = "";
  const { n, d } = privateKey;
  console.log("n:" + n + "\nd:" + d);
  console.log("Chuoi can gia ma la: " + messageContent);

  if (type == "chat") {
    try {
      // Use `atob` to decode base64-encoded string
      var mang = messageContent.split(",").map(Number);

      for (let i = 0; i < mang.length; i++) {
        // Use mang.length instead of charCode.length
        let decryptedCharCode = RSAService.mod(mang[i], d, n);
        plaintext += String.fromCharCode(decryptedCharCode);
        console.log("running, we are decoding message!");
      }

      console.log("Chuoi giai ma la: " + plaintext);

      return plaintext;
    } catch (error) {
      console.log("loi : " + error.message);
    }
  }

  return messageContent;
};
```

Hình 2.2. Hình ảnh minh họa code giải mã RSA

Kết Luận:

- Quá trình mã hóa và giải mã RSA đảm bảo tính bảo mật thông tin trong quá trình truyền tải.
- Cặp khóa công khai và riêng tư đảm bảo rằng chỉ người nhận có thể giải mã thông điệp.

2.2.2. JSON Web Token

2.2.2.1. Khái niệm Json Web Token

JSON Web Mã (JWT) là một chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và khép kín để truyền một cách an toàn thông tin giữa các bên dưới dạng đối tượng JSON. Thông tin này có thể được xác minh và đáng tin cậy vì nó có chứa chữ ký số. JWTs có thể được ký bằng một thuật toán bí mật (với thuật toán HMAC) hoặc một public / private key sử dụng mã hoá RSA.

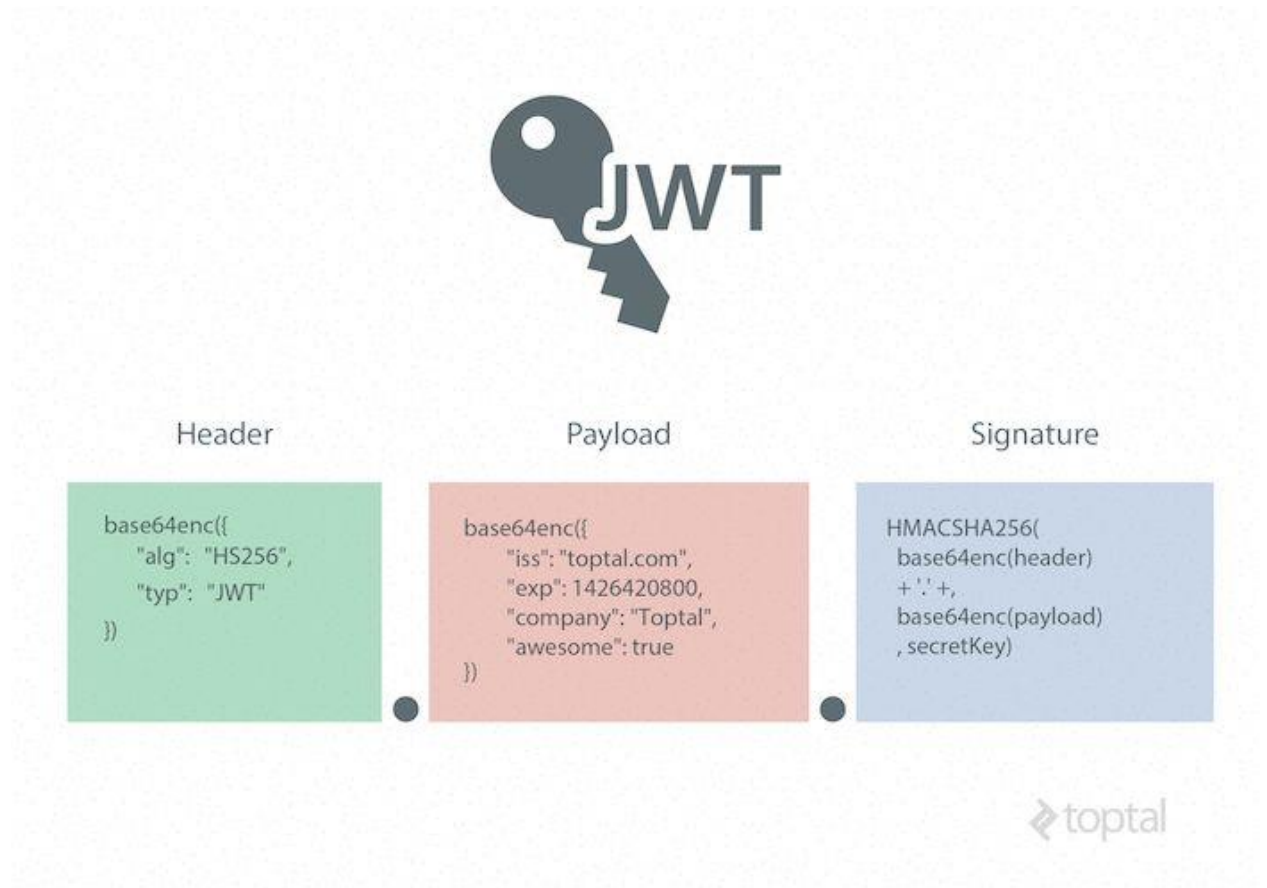
2.2.2.2. Cấu trúc của một JWT

Dưới đây là 1 JSON Web Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaXNTb2NpYWwiOnRydWV9.4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Hình 2.3. Hình ảnh minh họa ví dụ JWT

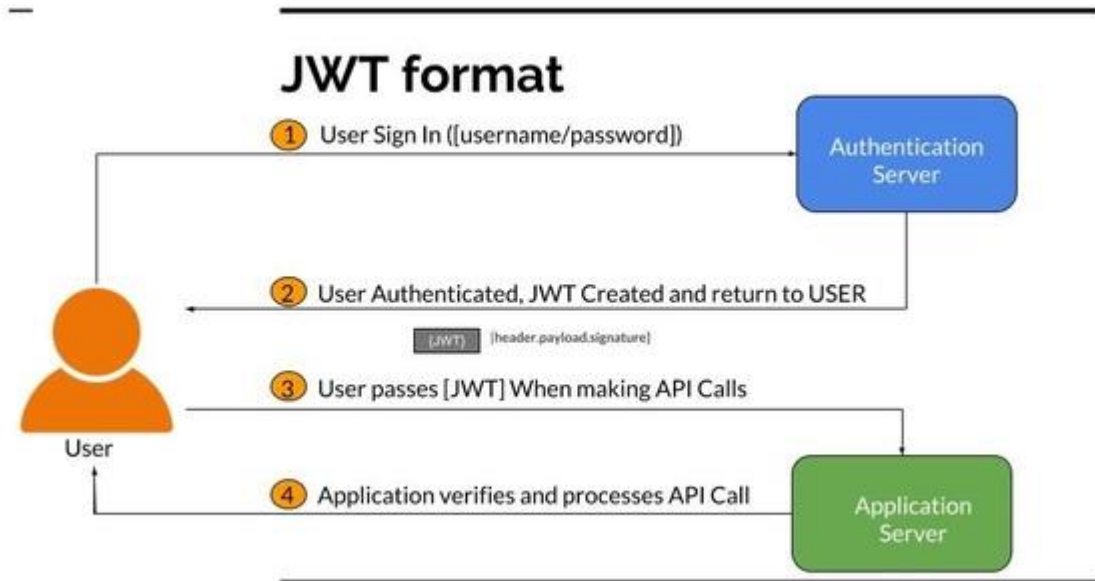
JWT trên bao gồm 3 phần: Header, Payload, Signature.



Hình 2.4. Hình ảnh minh họa cấu trúc JWT

- **Header**
Header bao gồm hai phần chính:
typ – Loại token (mặc định là JWT – cho biết đây là một Token JWT)
alg – Thuật toán đã dùng để mã hóa (HMAC SHA256 – HS256 hoặc RSA).
- **Payload:** Là nơi chứa các nội dung của thông tin (claim). Thông tin truyền đi có thể là mô tả của 1 thực thể (ví dụ như người dùng) hoặc cũng có thể là các thông tin bổ sung thêm cho phần Header. Chúng được chia làm 3 loại: reserved, public và private.
- **Signature:** Phần chữ ký được tạo bằng cách kết hợp 2 phần Header + Payload, rồi mã hóa nó lại bằng 1 giải thuật encode bất kỳ ví dụ như HMAC SHA-256.

2.2.2.3. Hệ thống sử dụng JWT



Hình 2.5. Hình ảnh minh họa cách thức hoạt động của JWT

Nhìn vào hình ta có thể thấy flow đi như sau:

1. User thực hiện login bằng cách gửi id/password hay sử dụng các tài khoản mạng xã hội lên phía Authentication Server (Server xác thực)
2. Authentication Server tiếp nhận các dữ liệu mà User gửi lên để phục vụ cho việc xác thực người dùng. Trong trường hợp thành công, Authentication Server sẽ tạo một JWT và trả về cho người dùng thông qua response.
3. Người dùng nhận được JWT do Authentication Server vừa mới trả về làm “chìa khóa” để thực hiện các “lệnh” tiếp theo đối với Application Server.
4. Application Server trước khi thực hiện yêu cầu được gọi từ phía User, sẽ verify JWT gửi lên. Nếu OK, tiếp tục thực hiện yêu cầu được gọi.

2.2.3. TCP Protocol + WebSocket

2.2.3.1. TCP Protocol

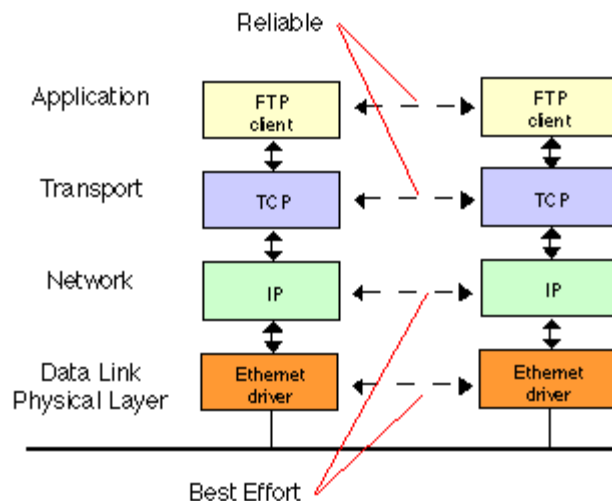
2.2.3.1.1. TCP protocol là gì?

Transmission Control Protocol (TCP) là giao thức tiêu chuẩn trên Internet đảm bảo trao đổi thành công các gói dữ liệu giữa các thiết bị qua mạng. TCP là giao thức truyền tải cơ bản cho nhiều loại ứng dụng, bao gồm máy chủ web và trang web, ứng dụng email, FTP và các ứng dụng ngang hàng.

TCP hoạt động với giao thức Internet (IP) để chỉ định cách dữ liệu được trao đổi trực tuyến. IP chịu trách nhiệm gửi từng gói đến đích của nó, trong khi TCP đảm bảo rằng các byte được truyền theo thứ tự mà chúng được gửi mà không có lỗi hoặc thiếu sót nào. Hai giao thức kết hợp với nhau được gọi là TCP/IP.

2.2.3.1.2. Nhiệm vụ của TCP

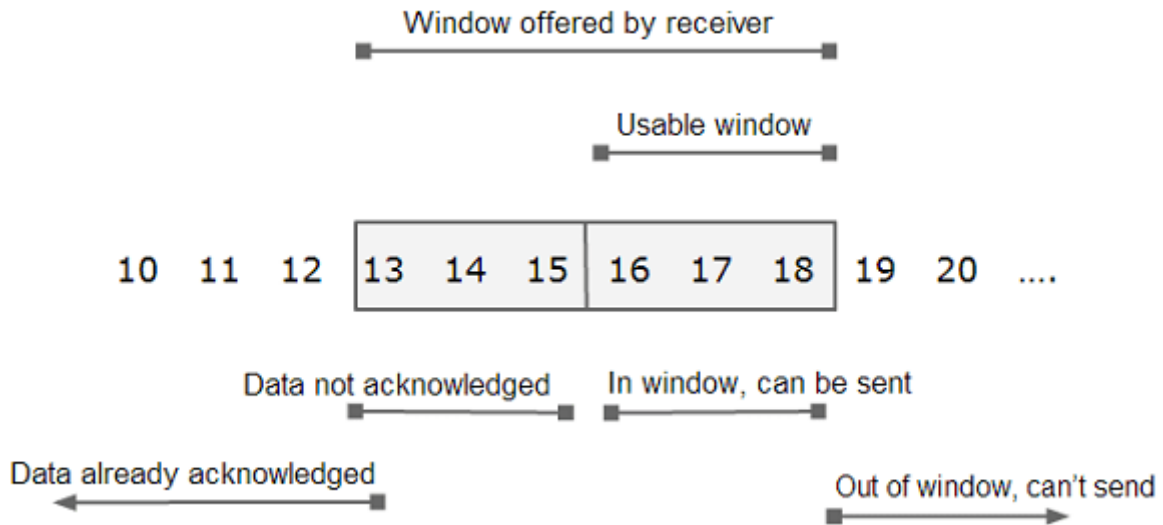
Chức năng của giao thức tcp được xác định là kiểm soát mức độ tin cậy của việc truyền dữ liệu. Trên các mạng như Internet, dữ liệu được truyền theo dạng gói tin, các gói này là các cụm dữ liệu được truyền hoàn toàn độc lập trên mạng, được tập hợp lại với nhau khi chúng đến địa chỉ đích và sau đó trả về dữ liệu gốc.



Hình 2.6. Hình ảnh minh họa nhiệm vụ của TCP

Truyền dữ liệu trên mạng được xử lý theo các lớp, mỗi một giao thức trên một lớp sẽ thực hiện công việc bổ sung cho các lớp khác. Tập hợp các lớp này được gọi là các ngăn giao thức (protocol stack). TCP và IP làm việc liên quan chặt chẽ với nhau, lớp này trên lớp kia.

2.2.3.1.3. Cách thức hoạt động



Hình 2.7. Hình ảnh minh họa cách thức hoạt động của TCP

TCP dán nhãn các gói tin theo dạng đánh số. TCP cũng sẽ đảm bảo rằng dữ liệu tới đích trong một thời hạn xác định (một khoảng thời gian vài trăm mili giây được gọi là thời gian chờ) và tuân theo một số quy định kỹ thuật khác. Với mỗi gói tin nhận được, thiết bị gửi sẽ được thông báo thông qua một gói được gọi là xác nhận. Sau khi hết thời gian chờ, không nhận được xác nhận, nguồn gửi sẽ gửi đi một bản sao của gói tin bị mất hoặc bị hoãn. Các gói tin không theo trình tự cũng sẽ không được xác nhận. Nhờ vậy, tất cả các gói dữ liệu sẽ luôn được tập hợp theo thứ tự, không có sơ hở, trong một khoảng thời gian chờ xác định và chấp nhận được.

2.2.3.1.4. Ứng dụng của giao thức TCP

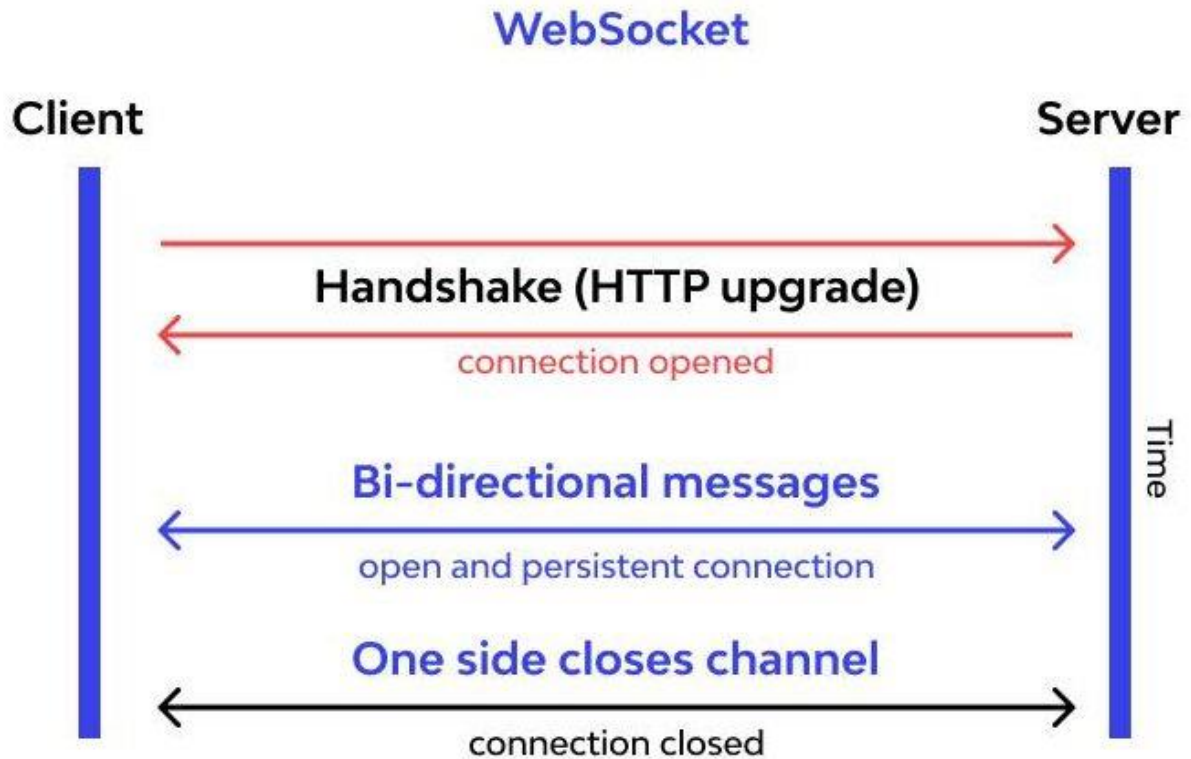
TCP/IP được sử dụng để kết nối thông tin trong Internet. Người dùng có thể thực hiện cung cấp thông tin từ xa, gửi mail, truyền file, ảnh hoặc phân phối web trên mạng Internet. Phương thức này cũng cho phép truy cập máy chủ từ xa, thay đổi trạng thái thông tin truyền trong môi trường Internet. Với giao thức TCP/IP, người dùng có thể dễ dàng thay đổi cách biểu thị thông tin thông qua các giao thức cơ bản hoặc giao thức ở mỗi lớp khi thông tin được truyền qua. Nhờ đó, thao tác truyền thông tin sẽ chính xác và hiệu quả hơn.

2.2.3.2. WebSocket

2.2.3.2.1. WebSocket là gì?

WebSocket là một giao thức truyền tải dữ liệu hai chiều (full-duplex) trên web, cho phép truyền tải dữ liệu trong thời gian thực giữa trình duyệt web và máy chủ. WebSocket cho phép một kết nối duy trì giữa máy khách và máy chủ, vì vậy các thông tin có thể được gửi đi và nhận lại một cách hiệu quả và nhanh

chóng mà không cần phải thiết lập kết nối mới mỗi khi truyền tải thông tin.



Hình 2.8. Hình ảnh minh họa websocket

WebSocket thường được sử dụng để phát triển các ứng dụng web thời gian thực như trò chơi trực tuyến, ứng dụng trò chuyện và các ứng dụng cập nhật dữ liệu trực tiếp. WebSocket được thiết kế để thay thế các giải pháp truyền tải dữ liệu trực tiếp khác như Ajax hoặc Comet, vì nó có hiệu suất cao hơn và tiết kiệm năng lượng hơn.

2.2.3.2.2. Cấu trúc của websocket

Giao thức WebSocket bao gồm hai phần là handshake và data transfer. Máy khách khởi tạo handshake bằng cách gửi một yêu cầu tới WebSocket bằng URI mà máy chủ cung cấp và chờ máy chủ xác nhận. Sau khi quá trình handshake thành công thì lúc này máy khách và máy chủ có thể chủ động trao đổi dữ liệu hai chiều cho nhau.

WebSocket hỗ trợ trao đổi truyền tải dữ liệu dạng văn bản (được mã hóa dưới dạng UTF-8) hoặc nhị phân. Chuẩn giao thức của WebSocket là ws:// (chuẩn thông thường) hoặc wss:// (tương tự https://)

2.2.3.2.3. Cách thức hoạt động của websocket

Về bản chất, WebSocket có tính chất hai chiều, nghĩa là giao tiếp diễn ra qua lại giữa máy khách-máy chủ. Do đó, quá trình hoạt động của giao thức này vẫn cần bắt đầu bằng một "bắt tay" (handshake) đặc biệt, sau đó là duy trì một kết nối liên tục, cho phép truyền dữ liệu một cách nhanh chóng và hiệu quả. Toàn bộ quá trình hoạt động của WebSocket sẽ diễn ra như sau:

Đầu tiên, trình duyệt gửi một yêu cầu HTTP đặc biệt đến máy chủ, yêu cầu mở một kết nối WebSocket. Yêu cầu này bao gồm một tiêu đề (Upgrade header) HTTP/1.1 thông báo cho máy chủ biết rằng trình duyệt muốn "nâng cấp" kết nối từ HTTP sang WebSocket.

Nếu máy chủ có hỗ trợ giao thức WebSocket, nó sẽ phản hồi với một tiêu đề HTTP để xác nhận việc chuyển đổi. Khi tiêu đề này được trình duyệt tiếp nhận, kết nối WebSocket sẽ chính thức được thiết lập.

Sau khi kết nối được thiết lập, máy chủ và trình duyệt đã có thể gửi dữ liệu lẫn nhau mà không cần yêu cầu hoặc phản hồi. Điều này giúp giảm thiểu độ trễ và tăng tốc độ truyền dữ liệu. Dữ liệu được truyền qua WebSocket được đóng gói trong các "khung" (Ws frame). Mỗi khung có thể chứa dữ liệu văn bản hoặc nhị phân, trong đó mỗi bên lại có thể gửi và nhận dữ liệu độc lập với nhau.

Khi có bất kỳ bên nào chủ động đóng kết nối, giao thức WebSocket sẽ được đóng lại và quá trình truyền tin sẽ kết thúc.

2.2.3.2.4. Ứng dụng của websocket

WebSocket được sử dụng rộng rãi trong nhiều lĩnh vực và ứng dụng khác nhau, nhờ vào khả năng truyền dữ liệu hai chiều liên tục và hiệu quả của nó. đây là một số ứng dụng phổ biến của WebSocket:

- ⇒ Ứng dụng trò chuyện trực tuyến: WebSocket là công nghệ lý tưởng cho các ứng dụng trò chuyện trực tuyến. Nó cho phép truyền tin nhắn tức thì giữa các người dùng với nhau mà không cần tải lại trang, tạo ra trải nghiệm trò chuyện mượt mà và tự nhiên.
- ⇒ Ứng dụng tài chính và giao dịch: WebSocket được sử dụng rộng rãi trong các ứng dụng theo dõi thị trường chứng khoán và giao dịch tài chính. Đó là những ứng dụng yêu cầu thông tin được cập nhật trong thời gian thực.
- ⇒ Ứng dụng truyền phát video và âm thanh: WebSocket cũng được ứng dụng trong lĩnh vực truyền phát video và âm thanh để đảm bảo dữ liệu được truyền liên tục và không bị gián đoạn.

2.2.4. Restful API

2.2.4.1. Restful API là gì?

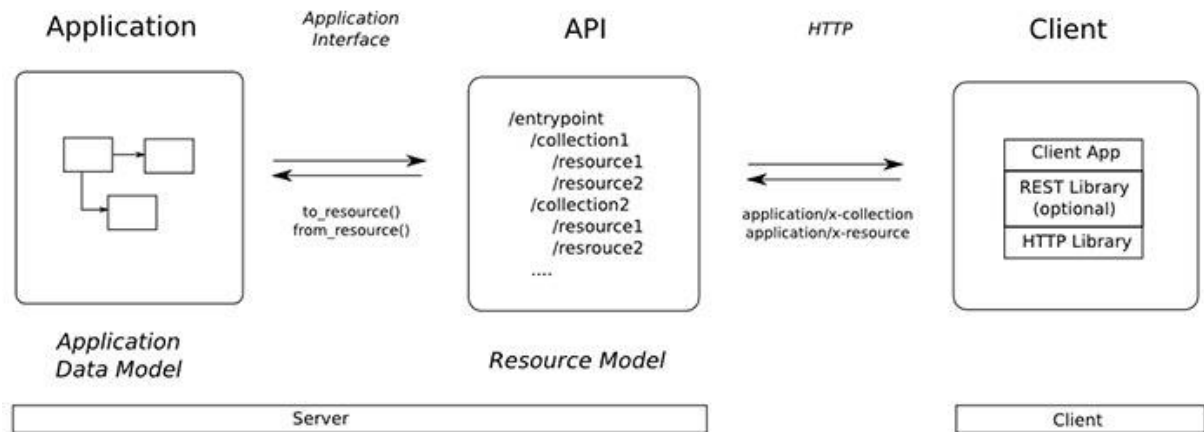
RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

Diễn giải các thành phần:

- ⇒ REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.
- ⇒ API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với

một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

2.2.4.2. Cách thức hoạt động



Hình 2.9. Hình ảnh minh họa cách thức hoạt động của RESTful API

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- ⇒ GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
- ⇒ POST (CREATE): Tạo mới một Resource.
- ⇒ PUT (UPDATE): Cập nhật thông tin cho Resource.
- ⇒ DELETE (DELETE): Xóa một Resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

2.2.4.3. Ưu điểm của RESTful API

RESTful API mang đến rất nhiều lợi ích và những hiệu quả nhất định cho các lập trình viên. Dưới đây là một vài ưu điểm nổi bật của RESTful API:

- ⇒ RESTful API giúp cho ứng dụng rõ ràng và dễ nhìn hơn.
- ⇒ RESTful API giúp cho các dữ liệu được trả về dưới nhiều định dạng khác nhau như XML, HTML, JSON...

- ⇒ RESTful API cho phép sử dụng các lệnh call thủ tục HTTP tiêu chuẩn để truy xuất dữ liệu và request.
- ⇒ Code của REST API đơn giản và ngắn gọn.
- ⇒ RESTful API dựa trên code và có thể dùng nó để đồng bộ hóa dữ liệu với website.
- ⇒ REST chú trọng vào nhiều tài nguyên của hệ thống.

Rất nhiều trang web hiện nay đang sử dụng REST API để cho phép các ứng dụng bên ngoài kết nối dữ liệu với họ dễ dàng hơn.

2.2.5. Mô hình client - server

2.2.5.1. Giới thiệu

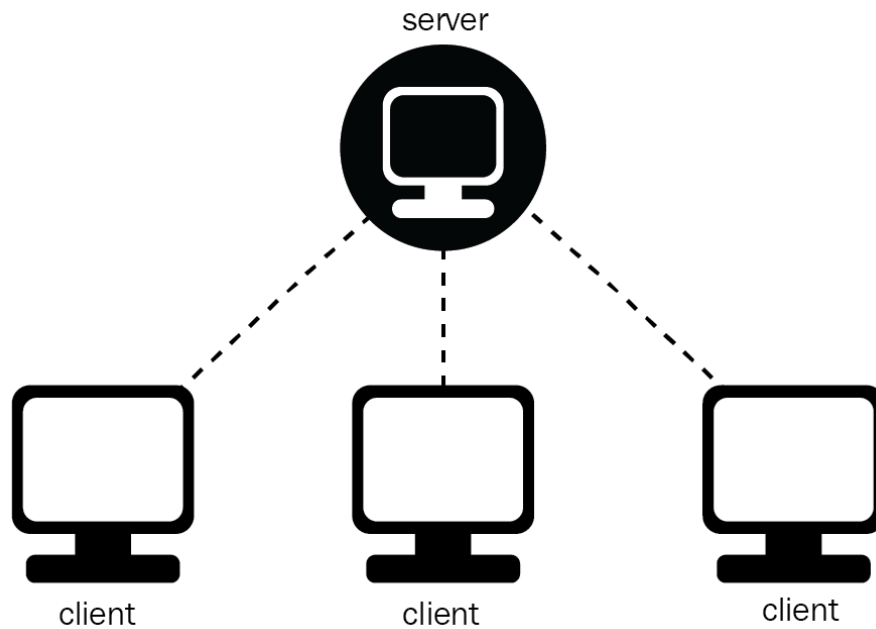
Mô hình client - server là mô hình giúp các máy tính giao tiếp truyền tải dữ liệu cho nhau. Nhắc đến Client với Server thì đây là 2 từ khóa khá phổ biến và ai cũng biết. Và mình cũng nhắc lại tóm tắt về client - server

Client và server về bản chất thì nó là 2 máy tính giao tiếp và truyền tải dữ liệu cho nhau.

Máy tính đóng vai trò là máy khách – **Client**: Với vai trò là máy khách, chúng sẽ không cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ. Một client trong mô hình này có thể là một server cho mô hình khác, tùy thuộc vào nhu cầu sử dụng của người dùng.

Máy tính đóng vai trò là máy chủ – **Server**: Là máy tính có khả năng cung cấp tài nguyên và các dịch vụ đến các máy khách khác trong hệ thống mạng. Server đóng vai trò hỗ trợ cho các hoạt động trên máy khách client diễn ra hiệu quả hơn.

2.2.5.2. Khái niệm về mô hình Client – Server



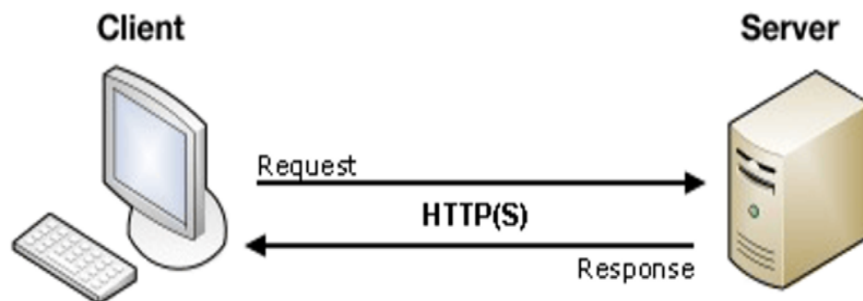
Hình 2.10. Hình ảnh minh họa mô hình mạng Client – Server

Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.

- Nguyên tắc hoạt động

Trong mô hình Client - Server, server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó

Máy tính được coi là máy khách khi chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ và đợi câu trả lời được gửi về.



Hình 2.11. Hình ảnh minh họa nguyên tắc hoạt động của mô hình client-server

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. Một số giao thức được sử dụng phổ biến hiện nay như: HTTPS, TCP/IP, FTP,...

Nếu máy khách muốn lấy được thông tin từ máy chủ, chúng phải tuân theo một giao thức mà máy chủ đó đưa ra. Nếu yêu cầu đó được chấp nhận thì máy chủ sẽ thu thập thông tin và trả về kết quả cho máy khách yêu cầu. Bởi vì Server - máy chủ luôn luôn trong trạng thái sẵn sàng để nhận request từ client nên chỉ cần client gửi yêu cầu tín hiệu và chấp nhận yêu cầu đó thì server sẽ trả kết quả về phía client trong thời gian ngắn nhất.

2.2.5.3. Ưu điểm và nhược điểm của mô hình Client – Server

Các ứng dụng Client-Server được thiết kế để phân tách logic người dùng và logic xử lý dữ liệu, điều này mang lại nhiều lợi ích, bao gồm sự linh hoạt, quản lý tài nguyên hiệu quả, và khả năng mở rộng dễ dàng.

Xét về các ưu điểm của mô hình này, ta sẽ có các ưu điểm như sau:

- Giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông. Giao thức chuẩn này cũng giúp các nhà sản xuất tích hợp lên nhiều sản phẩm khác nhau mà không gặp phải khó khăn gì.
- Có thể có nhiều server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.
- Chỉ mang đặc điểm của phần mềm mà không hề liên quan đến phần cứng, ngoài yêu cầu duy nhất là server phải có cấu hình cao hơn các client.
- Hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa.
- Cung cấp một nền tảng lý tưởng, cho phép cung cấp tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS).

Tuy nhiên, mô hình này vẫn có một số nhược điểm, có thể nói đến như:

- **Tắc nghẽn lưu lượng:** Nói về nhược điểm lớn nhất của mô hình mạng Client-Server đó chính là tắc nghẽn lưu lượng. Trong trường hợp có quá nhiều Client tạo request từ cùng một Server thì nó có thể sẽ làm cho kết nối chậm hơn. Trong trường hợp xấu nhất còn có thể xuất hiện hiện tượng crash. Khi một server bị quá tải thì sẽ tạo ra nhiều vấn đề khi truy cập thông tin.
- **Độ bền:** Client - Server là mạng tập trung chính vì thế, khi Server chính xảy ra sự cố hoặc bị nhiễu thì cũng đồng nghĩa với việc toàn bộ hệ thống mạng sẽ bị gián đoạn. Như vậy, bạn cần chú ý đó là mạng thiếu tính ổn định và độ bền. Bạn cần chú ý khi thực hiện.
- **Chi phí:** Chi phí được sử dụng để thiết lập và bảo trì Server trong Client - Server thường sẽ khá cao. Lý do là vì các hệ thống mạng có sức mạnh rất lớn cũng đồng nghĩa với việc giá để chi cho việc này là rất đắt. Chính vì vậy, không phải ai cũng có khả năng chi trả và sử dụng.
- **Bảo trì:** Khi các Server thực hiện triển khai để làm việc thì nó cũng sẽ hoạt động một cách không ngừng nghỉ. Điều này đồng nghĩa với việc chúng ta cần phải quan tâm đến việc bảo trì hệ thống đúng mức. Khi xảy ra bất cứ vấn đề gì cũng cần phải giải quyết ngay lập tức. Vậy nên, cần phải có một nhà quản lý mạng chuyên biệt để tiến hành duy trì hoạt động của Server khi chúng được đưa vào và sử dụng.
- **Tài nguyên:** Một điều mà chúng ta rất cần phải lưu ý đó chính là không phải tất cả tài nguyên hiện có trên Server đều sử dụng được. Ví dụ một cách đơn giản đó chính là chúng ta không thể in trực tiếp được tài liệu từ trên web cũng như tiến hành chỉnh sửa bất kỳ một thông tin nào trên ổ cứng của Client cả.

2.2.6. JavaScript + React TypeScript (Frontend)

Để xây dựng website cũng như các ứng dụng nói chung cần có 2 thành phần chính là Front-end và Back-end, trong đó Front-end là việc xây dựng các hiển thị giao diện, tương tác phía người dùng. Trong đó không thể không kể đến các ngôn ngữ chính soạn thảo siêu văn bản là HTML, CSS và ngôn ngữ lập trình JavaScript. Với sự phát triển tiến bộ của ngành công nghệ thông tin, các thư viện hỗ trợ, framework ra đời giúp cho quá trình xây dựng website, các ứng dụng trở nên dễ dàng hơn. Trong phạm vi đề tài này, nhóm em sẽ trình bày cơ sở lý thuyết của **JavaScript và React TypeScript**.

2.2.6.1. Javascript

JavaScript là một **ngôn ngữ lập trình thông dịch** với khả năng hướng đến đối tượng. Là một trong 3 ngôn ngữ chính trong lập trình web và có mối liên hệ lẫn nhau để xây dựng một website sống động, chuyên nghiệp:

- **HTML:** Hỗ trợ trong việc xây dựng layout, thêm nội dung dễ dàng trên website.
- **CSS:** Hỗ trợ việc định dạng thiết kế, bố cục, style, màu sắc,...
- **JavaScript:** Tạo nên những nội dung “động” trên website. Cùng tìm hiểu rõ hơn ở phần dưới đây.

JS là viết tắt của **JavaScript**, khi có JS bạn sẽ hiểu đó đang nói đến JavaScript.

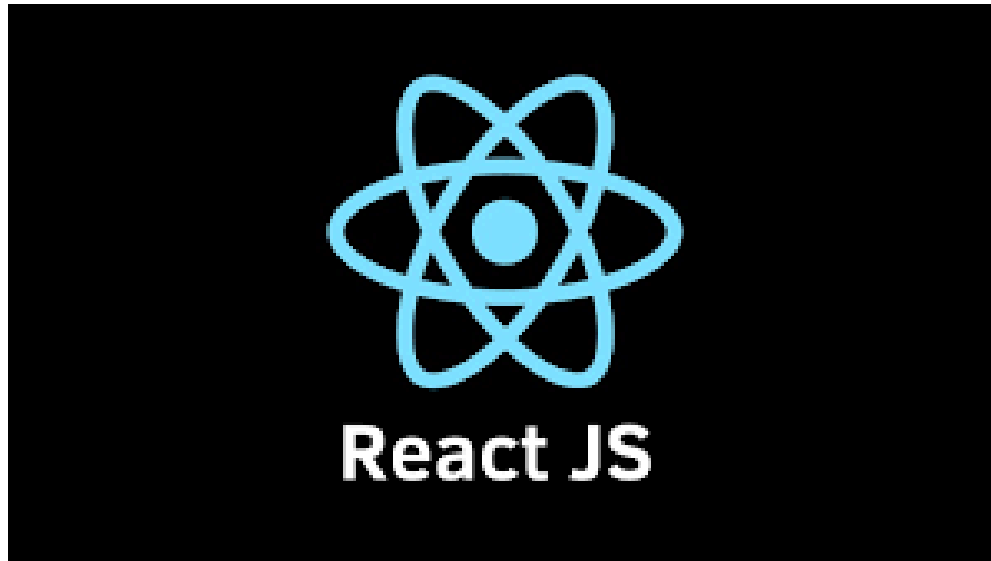
Nhiệm vụ của Javascript là xử lý những đối tượng HTML trên trình duyệt. Nó có thể can thiệp với các hành động như thêm / xóa / sửa các thuộc tính CSS và các thẻ HTML một cách dễ dàng. Hay nói cách khác, Javascript là một ngôn ngữ lập trình trên trình duyệt ở phía client. Tuy nhiên, hiện nay với sự xuất hiện của NodeJS đã giúp cho Javascript có thể làm việc ở backend.

2.2.6.2. JavaScript Framework là gì?

JavaScript Framework là thư viện được xây dựng dựa vào ngôn ngữ lập trình JavaScript. Từ đó, mỗi framework được tạo ra để phục cho từng lĩnh vực khác nhau. Bạn có thể tìm hiểu kỹ hơn về framework là gì, sẽ giúp bạn có thêm nhiều thông tin rõ ràng hơn. Hiện nay, có rất nhiều JavaScript Framework thông dụng như:

- **ReactJs:** Thư viện dành cho mobile.
- **NodeJs:** Dùng để xây dựng và phát triển ứng dụng realtime từ phía máy chủ.
- **Angular:** Dùng để xây dựng ứng dụng Single Page...

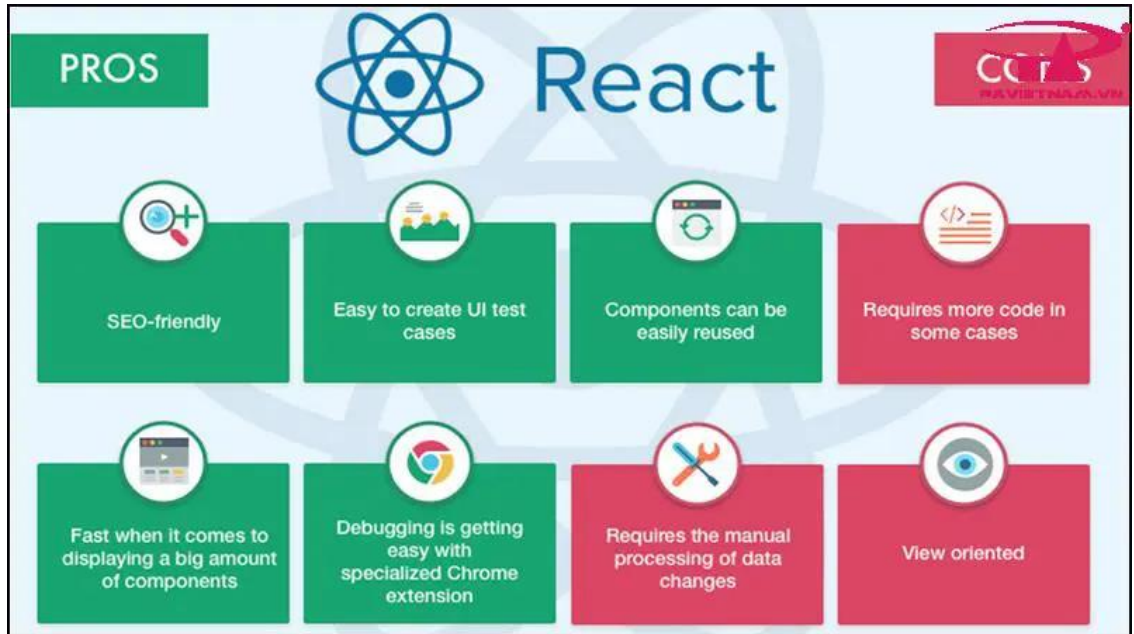
Nhóm chúng em đã sử dụng **ReactJs** để xây dựng chương trình, **ReactJs** là một framework mạnh mẽ được phát triển bởi Facebook, ra mắt vào năm 2013. **React** (Reactjs hay React.js) là một thư viện JavaScript phổ biến nhất để xây dựng giao diện người dùng hoặc các thành phần UI. Được tạo ra bởi sự cộng tác giữa Facebook và Instagram. Nó được duy trì bởi Facebook và một cộng đồng các nhà phát triển và công ty cá nhân. React có thể được sử dụng như một cơ sở để phát triển các ứng dụng trang đơn hoặc di động. Một trong những đặc trưng duy nhất của React là việc render dữ liệu không những có thể thực hiện ở tầng Server mà còn ở tầng Client.



Hình 2.12. Hình ảnh minh họa logo Framework ReactJS

Sơ lược qua các ưu điểm của framework này, ta có các ưu điểm đáng chú ý như sau:

- **Dễ dàng sử dụng, tạo được các component nhẹ:** React Component dễ viết hơn vì nó sử dụng JSX, mở rộng cú pháp tùy chọn cho JavaScript cho phép bạn kết hợp HTML với JavaScript. React cung cấp việc tạo component nhẹ, các thành phần phi trạng thái rất dễ dàng.
- **API thanh lịch:** Khuyến khích bạn nắm lấy bố cục bằng các component.
- **Hỗ trợ cộng đồng lớn:** React rất phổ biến, minh chứng rõ ràng bởi cộng đồng hỗ trợ lớn của nó.
- **Phổ biến trong giới StartUp:** Mức độ phổ biến của React đã giúp thúc đẩy sự phát triển của các công ty khởi nghiệp.
- **Nhiều tiện ích nguồn mở:** Tính khả dụng của một loạt các tiện ích mở rộng do cộng đồng nguồn mở phát triển cho React cung cấp cho bạn rất nhiều tùy chọn để xây dựng các giải pháp hoàn chỉnh.
- **Thân thiện với SEO:** Hầu như các JS Frameworks không thân thiện với các tìm kiếm mặc dù đã được cải thiện nhiều nhưng dưới sự hỗ trợ của các render dữ liệu trả về dưới dạng web page giúp cho SEO chuẩn hơn.

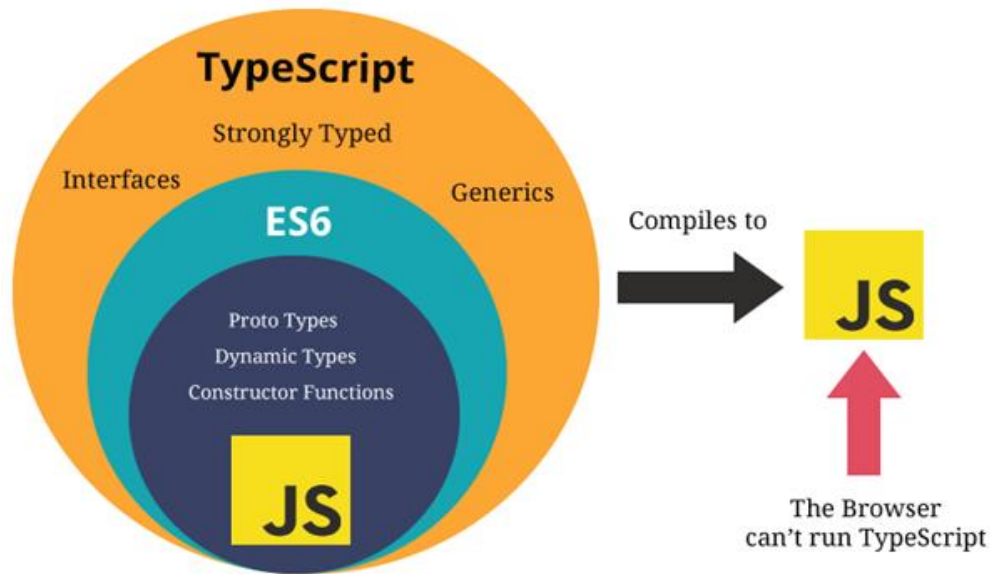


Hình 2.13. Hình ảnh minh họa các ưu điểm của ReactJS

Tuy nhiên, ReactJs cũng có những nhược điểm nhất định:

- **Điều chỉnh cho JSX:** Các Component không phải HTML nguyên bản mà được viết bằng JSX. Vậy nên phải mất thời gian để chờ đợi đội ngũ phát triển React điều chỉnh.
- **Các giải pháp hoàn chỉnh yêu cầu thư viện của bên thứ ba:** Do tập trung vào việc xây dựng giao diện người dùng, bạn có thể cần phải mở rộng React với các thư viện của bên thứ ba.
- **Ví dụ:** Nếu bạn cần hỗ trợ định tuyến phía máy khách trong ứng dụng của mình, bạn có thể sử dụng thư viện React Router của bên thứ ba.
- **Reactjs chỉ phục vụ cho tầng View:** React chỉ là View Library, không phải là một MVC framework như những framework khác. Đây chỉ là thư viện của Facebook giúp render ra phần view. Vì thế React sẽ không có phần Model và Controller, mà phải kết hợp với các thư viện khác. React cũng sẽ không có 2-way binding hay là Ajax.
- **Tính khả dụng của các tùy chọn có thể gây nhầm lẫn:** Có nhiều sự lựa chọn có thể gây nhầm lẫn – đặc biệt đối với người mới bắt đầu.

2.2.6.3. ReactJs TypeScript



Hình 2.14. Hình ảnh minh họa các thành phần của TS

TypeScript là một dự án mã nguồn mở được phát triển bởi Microsoft, nó có thể được coi là một phiên bản nâng cao của Javascript bởi việc bổ sung tùy chọn kiểu tĩnh và lớp hướng đối tượng mà điều này không có ở Javascript.

TypeScript có thể sử dụng để phát triển các ứng dụng chạy ở client-side (Angular2) và server-side (NodeJS).

Như hình ở trên, chúng ta có thể thấy **Typescript** bao trọn Javascript và ES6 ngoài ra còn có Interfaces, Strongly Typed và Generics nữa cấu tạo thành **Typescript**. Qua đó chúng ta thấy **Typescript** sử dụng tất cả các tính năng của ES6 (ECMAScript 2015) như là class, module, destructuring... ngoài ra còn sử dụng các Interfaces, Generics ngoài ra còn có một số tính năng khác mình sẽ nói rõ ở phần dưới. Nhưng bản chất **Typescript** cũng sẽ compile ra Javascript cũng có thể nói là **Typescript** là Javascript và browser thì không thể đọc được **Typescript** nên mới cần biên dịch ra Javascript để thực thi các lệnh và đó là giới thiệu qua về **Typescript**.

Vậy tại sao nên dùng Typescript ?

- **Dễ học và dễ code** : Vì được kế thừa cũng như dựa trên các cú pháp từ javascript và code theo kiểu hướng đối tượng như class, interface, generics... nên sẽ dễ tiếp cận.
- **Dễ scale dự án** : Với việc sử dụng các kỹ thuật mới nhất và lập trình hướng đối tượng nên TypeScript giúp chúng ta phát triển các dự án lớn một cách dễ dàng. Việc code dự án javascript với dự án càng lớn thì sẽ có rất nhiều kiểu dữ liệu đầu vào và đầu ra khác nhau và việc chúng ta phải nhớ hoặc phải mò lại code rất khổ sở cũng như không rõ sẽ làm cho việc code trở nên khó khăn nhưng đối với Typescript thì chúng ta sẽ biết rõ dữ liệu đầu vào (input) cũng như dữ liệu đầu ra (output) một cách rõ ràng giúp chúng ta thuận tiện khi code. Ngoài ra nó còn được phát triển bởi microsoft nên khi sử dụng cùng với VScode sẽ có rất nhiều các extensions giúp hỗ trợ nhắc lệnh hoặc báo lỗi rất tốt.
- **Được nhiều người sử dụng** : Typescript được mọi người sử dụng Javascript rất yêu thích và chuyển qua sử dụng ngoài ra còn có các Framework khuyến khích sử dụng như ReactJS, AngularJS nên sẽ có cộng đồng phát triển rất mạnh hỗ trợ việc chúng ta học và làm việc.

Cài đặt Typescript Global

Chạy lệnh : `npm install -g typescript` ở terminal.

Lưu ý là trong máy đã cài Nodejs để có thể install. Sau khi install thành công thì sử dụng lệnh `tsc -v` để kiểm tra.

```

pwsh Z:\workspace\blog
> npm uninstall -g typescript

removed 1 package, and audited 1 package in 874ms
found 0 vulnerabilities
pwsh Z:\workspace\blog
> npm install -g typescript

added 1 package, and audited 2 packages in 1s
found 0 vulnerabilities
pwsh Z:\workspace\blog
> tsc -v
Version 4.6.4
pwsh Z:\workspace\blog

```

Cách để chạy thử file typescript :




```

console.log("hello world with typescript")

```

Đầu tiên tạo ra một file `blog.ts` là một file Typescript với nội dung như hình. Để compile ra file Javascript thì sử dụng lệnh `tsc blog.ts` để tạo ra file `blog.js` để thực thi các lệnh. Và cuối cùng để thực thi file `blog.js` thì ta dùng lệnh `node blog.js` và đây là kết quả.



```

PowerShell BinH x + v
> npm uninstall -g typescript
removed 1 package, and audited 1 package in 874ms
found 0 vulnerabilities
pwsh Z:\workspace\blog 11:46:25
> npm install -g typescript
added 1 package, and audited 2 packages in 1s
found 0 vulnerabilities
pwsh Z:\workspace\blog 11:46:35
> tsc -v
Version 4.6.4
pwsh Z:\workspace\blog 11:46:46
> vim blog.ts
pwsh Z:\workspace\blog 11:55:51
> tsc blog.ts
pwsh Z:\workspace\blog 11:56:05
> node blog.js
hello world with typescript
pwsh Z:\workspace\blog 11:58:13

```

Vậy là đã thực thi thành công.

2.2.6.4. Tạo dự án ReactJS + Typescript.

Sử dụng `create-react-app` : Nếu các bạn chưa biết về `create-react-app` thì đây là một công cụ giúp chúng ta có thể triển khai một dự án ReactJS nhanh chóng và hiệu quả. Vậy làm sao để có thể sử dụng `create-react-app` để tạo dự án kết hợp ReactJS + Typescript ? Thật may là rất dễ dàng để làm điều đó với một dòng lệnh. Đối với `npm` `npx create-react-app my-app --template typescript` Hoặc với `yarn` `yarn create react-app my-app --template typescript`.

```

PowerShell Binh
cd Z:\workspace\blog
npm start

Happy hacking!
pwsh Z:\workspace\blog > master 16.14.0 12:14:16

Directory: Z:\workspace\blog

Mode                LastWriteTime         Length Name
----                -
d----- 5/15/2022 12:14 PM                node_modules
d----- 5/15/2022 12:13 PM                public
d----- 5/15/2022 12:14 PM                src
-a----- 5/15/2022 12:13 PM             310 .gitignore
-a----- 5/15/2022 12:14 PM        1136710 package-lock.json
-a----- 5/15/2022 12:14 PM             963 package.json
-a----- 5/15/2022 12:13 PM            2117 README.md
-a----- 5/15/2022 12:14 PM             561 tsconfig.json

pwsh Z:\workspace\blog > master 16.14.0 12:15:15
code
pwsh Z:\workspace\blog > master 16.14.0 12:15:23
yarn start
yarn run v1.22.11
$ react-scripts start
(node:16164) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:16164) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view blog in the browser.

```

Hình 2.15. Hình ảnh minh họa tạo dự án ReactJS

Sau khi tạo dự án thành công thì dự án sẽ có các thư mục như một dự án bình thường ngoài ra còn có thêm file tsconfig.json để chúng ta config cho typescript. Ta có thể cd vào dự án và chạy lệnh npm start or yarn start để chạy dự án thử.

2.2.7. Java Spring Boot (Backend)

2.2.7.1. Giới thiệu về Back-end

Backend là những chức năng hỗ trợ hoạt động của một trang web hoặc ứng dụng mà người dùng không nhìn thấy được (hoặc có thể ví như đây là phần chìm của tảng băng). Nó có cơ chế hoạt động gần giống như bộ não của con người, xử lý các yêu cầu, lệnh và chọn thông tin thích hợp để hiển thị trên màn hình.

BackEnd của một trang web bao gồm ba thành phần: máy chủ, ứng dụng và cơ sở dữ liệu. Điều này cho phép trang web hoạt động hiệu quả và cung cấp cho người dùng thông tin chính xác nhanh chóng.

Một trang web sẽ chứa một hoặc nhiều tập lệnh được chạy trên máy chủ mỗi khi truy cập vào website. Mọi hoạt động hiển thị trên trình duyệt web có sự đóng góp một phần Backend. Quy trình của Backend bao gồm:

- Xử lý các yêu cầu của web đến.
- Chạy tập lệnh như (JSP,ASP, PHP,...) để tạo ra HTML.
- Truy cập vào dữ liệu từ cơ sở dữ liệu bằng sử dụng truy vấn SQL.
- Lưu trữ và cập nhật hồ sơ có trong cơ sở dữ liệu.
- Giải mã và mã hóa dữ liệu.
- Xử lý các dữ liệu tệp tải lên và tải xuống.
- Xử lý người dùng bằng JavaScript.

Trong phạm vi đề tài, nhóm chúng em sử dụng Java Spring Boot để xây dựng backend cho chương trình.

2.2.7.2. Java Spring Boot

2.2.7.2.1.Spring Boot là gì?

Spring Boot là một trong số các module của Spring framework chuyên cung cấp các tính năng RAD (Rapid Application Development) cho phép tạo ra và phát triển các ứng dụng độc lập dựa trên Spring một cách nhanh chóng.

Spring Boot ra đời với mục đích loại bỏ những cấu hình phức tạp của Spring, nó không yêu cầu cấu hình XML và nâng cao năng suất cho các nhà phát triển. Với sự góp mặt của Spring Boot, hệ sinh thái Spring đã trở nên mạnh mẽ, phổ biến và hiệu quả hơn bao giờ hết.



Hình 2.16. Hình ảnh minh họa java spring boot

2.2.7.2.2. Ưu điểm của Spring Boot

Với mục đích ra đời rất rõ ràng của mình, Spring Boot đã khắc phục được những hạn chế về cấu hình của Spring. Dưới đây, Stringee sẽ giới thiệu thêm đến bạn một số lợi ích của Spring Boot.

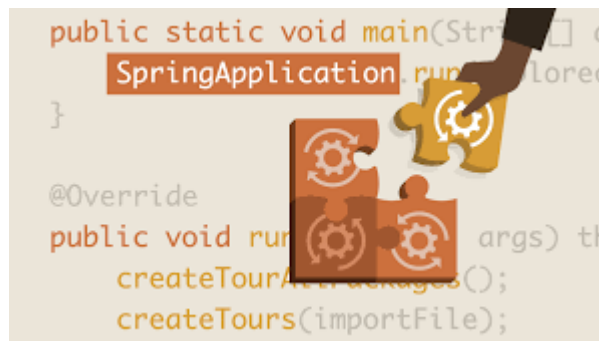
Những ưu điểm đó bao gồm:

- Hội tụ đầy đủ các tính năng của Spring framework.
- Đơn giản hóa cấu hình và xây dựng được các ứng dụng độc lập có khả năng chạy bằng “java -jar” nhờ các dependency starter.
- Dễ dàng deploy vì các ứng dụng server được nhúng trực tiếp vào ứng dụng để tránh những khó khăn khi triển khai lên môi trường production mà không cần thiết phải tải file WAR.
- Cấu hình ít, tự động hỗ trợ bất cứ lúc nào cho chức năng giống với Spring như tăng năng suất, giảm thời gian viết code và không yêu cầu XML config.
- Cung cấp nhiều plugin, số liệu, cấu hình ứng dụng từ bên ngoài.

2.2.7.2.3. Các đặc tính cơ bản của Spring Boot

Vì là một framework RAD nên Spring Boot mang trong nó nhiều đặc tính nổi trội phục vụ cho việc phát triển và cài đặt nhanh một ứng dụng chạy trên Java.

Đặc tính đầu tiên có thể kể đến đó là Spring Boot cung cấp sẵn cho chúng ta một lớp có chứa hàm main và được hoạch định làm điểm mở đầu cho toàn bộ chương trình. Lớp này được đặt tên là Spring Application, nó giúp khởi chạy các ứng dụng từ hàm main, khi chạy nó chúng ta chỉ cần gọi phương thức run.



Hình 2.17. Hình ảnh minh họa đặc tính cơ bản của spring boot

Spring Boot giúp giảm bớt độ phức tạp trong việc cấu hình ứng dụng trong trường hợp chúng ta sử dụng nhiều môi trường. Với Profiles, Spring Boot cung cấp cho người sử dụng một cách phân chia cấu hình cho từng môi trường. Các bên thực hiện việc cấu hình ứng dụng hoàn toàn có thể được đánh dấu profiles để giới hạn thời điểm hay môi trường mà nó sẽ được tải các cấu hình lên ứng dụng.

Externalized Configurations: Externalized Configuration cho phép bạn có khả năng cấu hình được từ bên ngoài. Vì vậy, một ứng dụng được xây dựng có thể được vận hành và hoạt động trên nhiều môi trường khác nhau. Để thực hiện Externalized Configuration bạn có thể sử dụng các file properties, YAML, các tham số command line hay các biến môi trường.

Đặc tính cuối cùng mà bạn nên biết đó là tất cả các tính năng log nội bộ của Spring Boot đều sử dụng common logging. Chúng được quản lý một cách mặc

định, vì vậy bạn không nên sửa các dependency logging nếu không được yêu cầu.

CHƯƠNG 3: THỰC NGHIỆM ĐỀ TÀI

3.1. Giới thiệu ứng dụng

Trong cuộc sống ngày nay, chúng ta đang chứng kiến một cuộc cách mạng kỹ thuật số đang diễn ra với tốc độ nhanh chóng, mang theo những đổi mới không ngừng và tác động sâu rộng đến mọi lĩnh vực. Cách mạng 4.0, với những tiến bộ đáng kinh ngạc trong lĩnh vực công nghệ thông tin, trí tuệ nhân tạo và kết nối mạng, không chỉ là một xu hướng mà còn là nguồn động viên mạnh mẽ cho sự phát triển toàn diện trong xã hội ngày nay. Trong đó vấn đề về an toàn bảo mật thông tin là vô cùng cấp thiết, ảnh hưởng đến kinh tế và chính trị quốc gia nói chung và trải nghiệm riêng tư người dùng nói riêng.

Nhận ra được điều đó, nhóm chúng em đã xây dựng ứng dụng mã hoá tin nhắn đầu cuối theo thời gian thực, làm tiền đề cho việc xây dựng các ứng dụng trong tương lai và để phù hợp với mục đích, tiêu chí đánh giá của học phần “An toàn bảo mật thông tin”.

3.1.1. Kiến thức cần thiết

Hệ Mã RSA: Ứng dụng sử dụng hệ mã hóa RSA để bảo vệ tính toàn vẹn và bí mật của thông tin truyền tải. Cặp khóa công khai và riêng tư được tạo và quản lý an toàn, đảm bảo mức độ an toàn cao trong giao tiếp.

JSON Web Tokens (JWT): JWT được sử dụng để xác thực và giữ thông tin định danh của người dùng. Điều này giúp bảo vệ tính xác thực và ủy quyền trong ứng dụng.

TCP Protocol + WebSocket: Sử dụng giao thức TCP để truyền tin nhắn an toàn và đảm bảo tính toàn vẹn dữ liệu. WebSocket được tích hợp để tạo kết nối liên tục giữa client và server, giúp truyền tin nhanh chóng và hiệu quả.

Restful API: Xây dựng các API theo kiến trúc Restful để tương tác giữa frontend và backend. Điều này tạo ra một giao diện thống nhất, dễ mở rộng và tương tác linh hoạt.

Mô Hình Client - Server: Sử dụng mô hình client - server để quản lý và tương tác giữa phía người dùng (client) và hệ thống (server).

Về Công Nghệ Sử Dụng:

Frontend (JavaScript + React TypeScript) : Sử dụng React TypeScript để xây dựng giao diện người dùng thân thiện và linh hoạt, tận dụng tính năng mạnh mẽ của TypeScript để giảm thiểu lỗi và tăng tính maintainability.

Backend (Java Spring Boot): Java Spring Boot được chọn để xây dựng backend với sự linh hoạt, hiệu suất cao và khả năng mở rộng. Cung cấp các API Restful để giao tiếp với frontend và xử lý logic chính của ứng dụng.

3.1.2. Tính Năng Chính của Ứng Dụng

Ứng dụng mang đến những tính năng chính vô cùng ấn tượng, đặt sự bảo mật và tính liên tục vào trung tâm của trải nghiệm người dùng. Với việc sử dụng hệ mã hóa RSA. Quá trình mã hóa và giải mã thông điệp được thực hiện một cách an toàn và chính xác, tạo nên một tường lửa không thể xâm phạm cho thông tin cá nhân.

“Chat APP RSA” có các chức năng cơ bản như một ứng dụng chat trên thời gian thực như: chat trực tiếp với người khác, chat theo nhóm, gửi lời mời kết bạn, xem trang cá nhân chính,...

Để tăng cường xác thực người dùng và duy trì trạng thái đăng nhập, chúng tôi sử dụng JSON Web Tokens (JWT). Điều này giúp đảm bảo rằng mọi hành động trên ứng dụng chỉ được thực hiện bởi những người dùng được ủy quyền, đồng thời giữ cho trạng thái đăng nhập của họ được duy trì một cách an toàn.

Ở mức độ giao tiếp, chúng em chọn kết hợp cả giao thức TCP và WebSocket. Việc này không chỉ giúp chúng em duy trì kết nối liên tục giữa client và server mà còn tối ưu hóa quá trình gửi và nhận tin nhắn, mang lại trải nghiệm truyền tin nhanh chóng và hiệu quả.

3.2. Mô hình hóa dữ liệu

3.2.1. Các yêu cầu về dữ liệu

Website cần lưu thông tin về:

- Các user: gồm có id, address, avata, birth_date, code, fullname, gender, password, username, public_key_id, một user có thể nằm trong nhiều user_room, một user có thể 0, 1 hoặc nhiều message, một user có 1 rsa_key khi đăng nhập, một user có thể có 0, 1 hoặc nhiều friend.
- Các rsa_key: gồm có id, d, e, n. Một rsa_key sẽ dùng cho 1 user khi đăng nhập, một rsa_key có thể dùng cho 1 user, một room sẽ có 2 rsa_key.
- Các user_room: gồm có id, nick_name, role, room_id, user_id. Một room sẽ có nhiều user_room, một user có thể nằm trong nhiều user_room.
- Các room: gồm có id, avatar, color, create_date, description, name, private_key, public_key, room_type_id. Một room sẽ có nhiều user, một room sẽ có 0, 1 hoặc nhiều message.
- Các room_type: gồm có id, code, description, name. Một room_type có thể nằm trong nhiều room.
- Các friend: gồm cosid, last_modify_date, state, receiver_id, room_id, request_sender_id. Một friend sẽ có 2 user.
- Các message: gồm có id, content, send_date, message_type_id, room_id, user_id. Một user có thể có 0, 1 hoặc nhiều message, một room có thể có 0, 1 hoặc nhiều message, một message sẽ có 1 message_type.
- Các message_type: gồm có id, code, description, name. Một message_type có thể nằm trong 0, 1 hoặc nhiều message.

3.2.2. Thiết kế bảng

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	<i>Không</i>
2	address	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
3	avatar	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
4	birth_date	datetime			Có	<i>NULL</i>
5	code	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
6	fullname 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
7	gender	bit(1)			Có	<i>NULL</i>
8	password	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
9	username 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
10	public_key_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>

Hình 3.1. Hình ảnh minh họa bảng *tbl_user*

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	<i>Không</i>
2	d	longtext	utf8mb4_general_ci		Có	<i>NULL</i>
3	e	longtext	utf8mb4_general_ci		Có	<i>NULL</i>
4	n	longtext	utf8mb4_general_ci		Có	<i>NULL</i>

Hình 3.2. Hình ảnh minh họa bảng *tbl_rsa_key*

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	<i>Không</i>
2	nick_name	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
3	role	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
4	room_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
5	user_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>

Hình 3.3. Hình ảnh minh họa bảng *tbl_user_room*

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	Không
2	avatar	varchar(255)	utf8mb4_general_ci		Có	NULL
3	code	varchar(255)	utf8mb4_general_ci		Có	NULL
4	color	varchar(255)	utf8mb4_general_ci		Có	NULL
5	create_date	datetime			Có	NULL
6	description	varchar(255)	utf8mb4_general_ci		Có	NULL
7	name	varchar(255)	utf8mb4_general_ci		Có	NULL
8	private_key_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL
9	public_key_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL
10	room_type_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL

Hình 3.4. Hình ảnh minh họa bảng *tbl_room*

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	Không
2	code	varchar(255)	utf8mb4_general_ci		Có	NULL
3	description	varchar(255)	utf8mb4_general_ci		Có	NULL
4	name	varchar(255)	utf8mb4_general_ci		Có	NULL

Hình 3.5. Hình ảnh minh họa bảng *tbl_room_type*

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	Không
2	last_modify_date	datetime			Có	NULL
3	state	bit(1)			Có	NULL
4	receiver_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL
5	request_sender_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL
6	room_id 🔑	varchar(255)	utf8mb4_general_ci		Có	NULL

Hình 3.6. Hình ảnh minh họa bảng *tbl_friend*

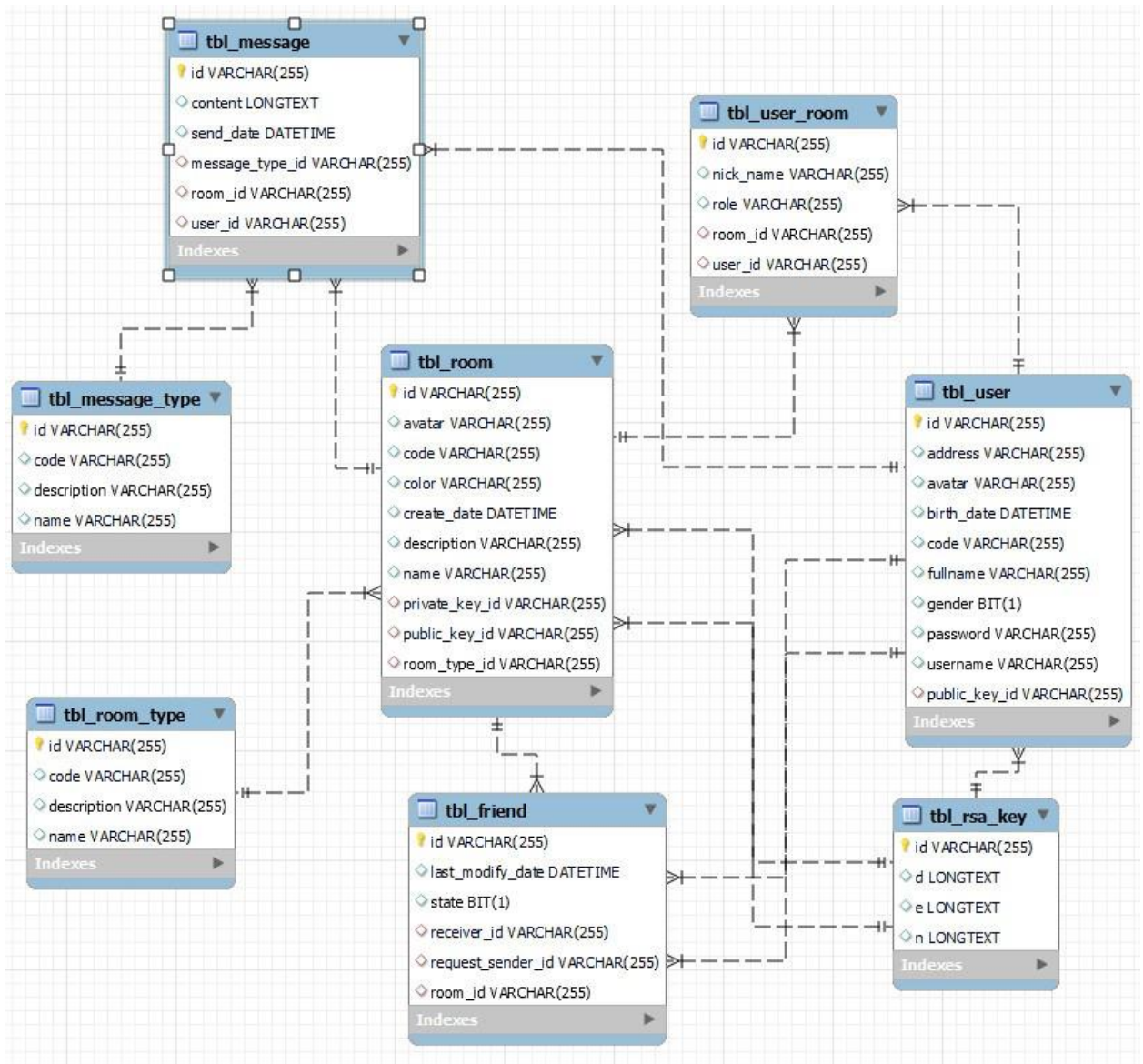
#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	<i>Không</i>
2	content	longtext	utf8mb4_general_ci		Có	<i>NULL</i>
3	send_date	datetime			Có	<i>NULL</i>
4	message_type_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
5	room_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
6	user_id 🔑	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>

Hình 3.7. Hình ảnh minh họa bảng tbl_message

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định
1	id 🔑	varchar(255)	utf8mb4_general_ci		Không	<i>Không</i>
2	code	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
3	description	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>
4	name	varchar(255)	utf8mb4_general_ci		Có	<i>NULL</i>

Hình 3.8. Hình ảnh minh họa bảng tbl_message_type

3.2.3. Biểu đồ thực thể liên kết mức vật lý



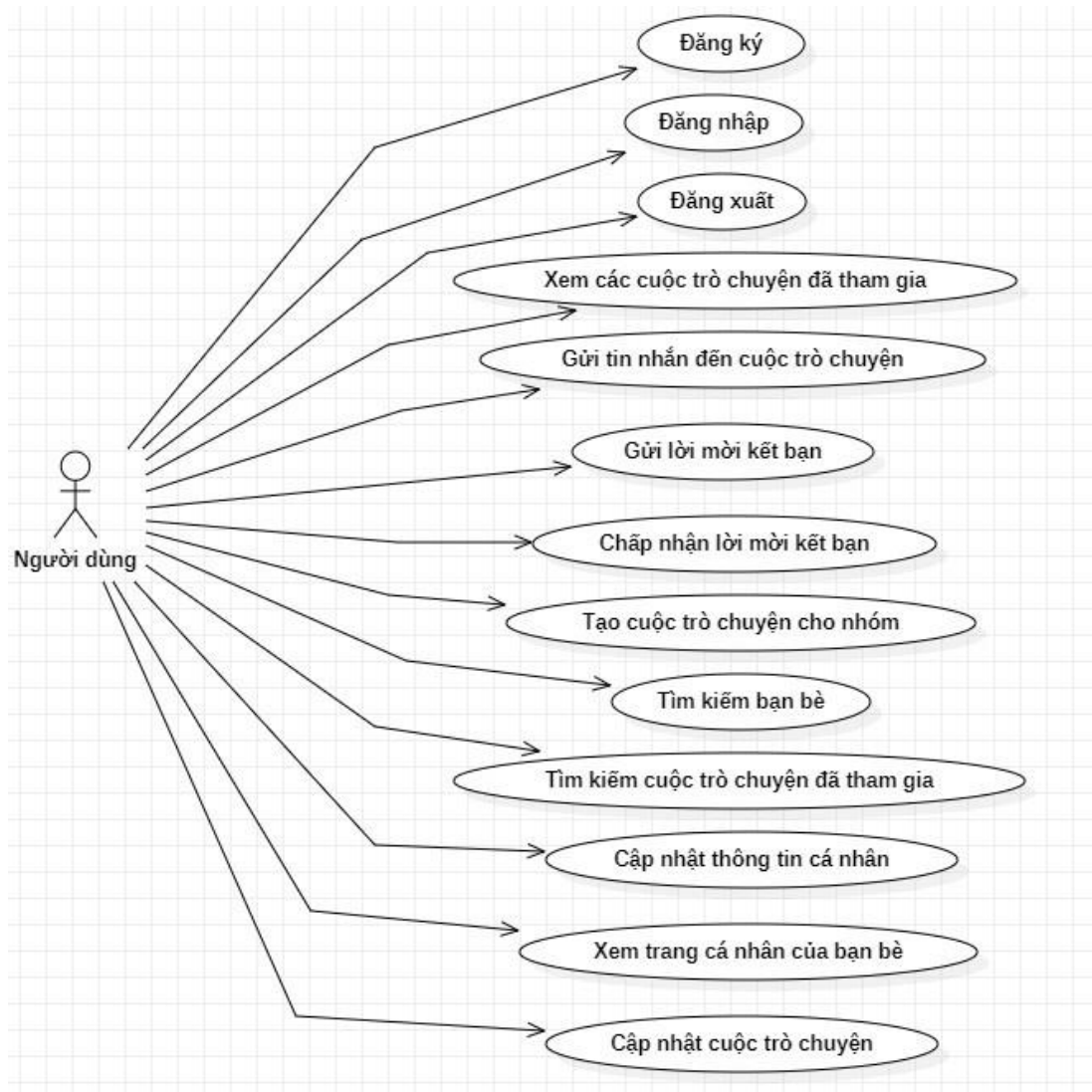
Hình 3.9. Biểu đồ thực thể liên kết mức vật lý

3.3. Mô hình hóa chức năng

3.3.1. Usecase chính

- Đăng nhập: Cho phép người dùng đăng nhập vào hệ thống.
- Đăng ký: Cho phép người dùng đăng ký tài khoản.
- Đăng xuất: Cho phép người dùng đăng xuất ra khỏi hệ thống, có thể đổi tài khoản.

- Xem các cuộc trò chuyện đã tham gia: Cho phép người dùng thấy được các cuộc trò chuyện của mình.
- Gửi tin nhắn đến cuộc trò chuyện: Cho phép người dùng gửi tin nhắn tới người dùng khác trong các cuộc trò chuyện.
- Gửi lời mời kết bạn: Cho phép người dùng kết nối với người dùng khác.
- Tạo cuộc trò chuyện cho nhóm: Cho phép người dùng tạo trò chuyện với nhiều người dùng khác.
- Tìm kiếm bạn bè: Cho phép người dùng tìm kiếm bạn bè theo tên.
- Tìm kiếm cuộc trò chuyện đã tham gia: Cho phép người dùng tìm kiếm cuộc trò chuyện theo tên.
- Cập nhật thông tin cá nhân: Cho phép người dùng cập nhật thông tin cá nhân.
- Xem trang cá nhân của bạn bè: Cho phép người dùng xem thông tin cá nhân của những người mình đã kết nối.
- Cập nhật của trò chuyện: Cho phép người dùng thay đổi thông tin về cuộc trò chuyện.



Hình 3.10. Biểu đồ usecase chính.

3.3.2. Đặc tả chi tiết các usecase

3.3.2.1. Đặc tả usecase Đăng nhập

⇒ Tên use case: Đăng nhập

Mô tả vắn tắt: Use case này cho phép người dùng đăng nhập vào hệ thống để truy cập các chức năng và thông tin cá nhân của mình.

⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn đăng nhập vào hệ thống.

Bước 2: Người dùng nhập tên đăng nhập và mật khẩu vào các trường tương ứng trên màn hình đăng nhập.

Bước 3: Người dùng kích vào nút "Đăng nhập". Hệ thống kiểm tra thông tin đăng nhập.

Bước 4: Nếu thông tin đăng nhập đúng, hệ thống chuyển người dùng đến màn hình chính của ứng dụng và use case kết thúc. Nếu thông tin không chính xác, hệ thống đưa ra thông báo lỗi và yêu cầu người dùng nhập lại.

Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu người dùng kích vào "Quên mật khẩu", hệ thống chuyển người dùng đến màn hình khôi phục mật khẩu và use case kết thúc.

Tại bước 3 ở luồng cơ bản, nếu người dùng kích vào "Đăng ký", hệ thống chuyển người dùng đến màn hình đăng ký mới và use case kết thúc.

Trong tất cả các bước tại luồng cơ bản, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo lỗi. Use case kết thúc.

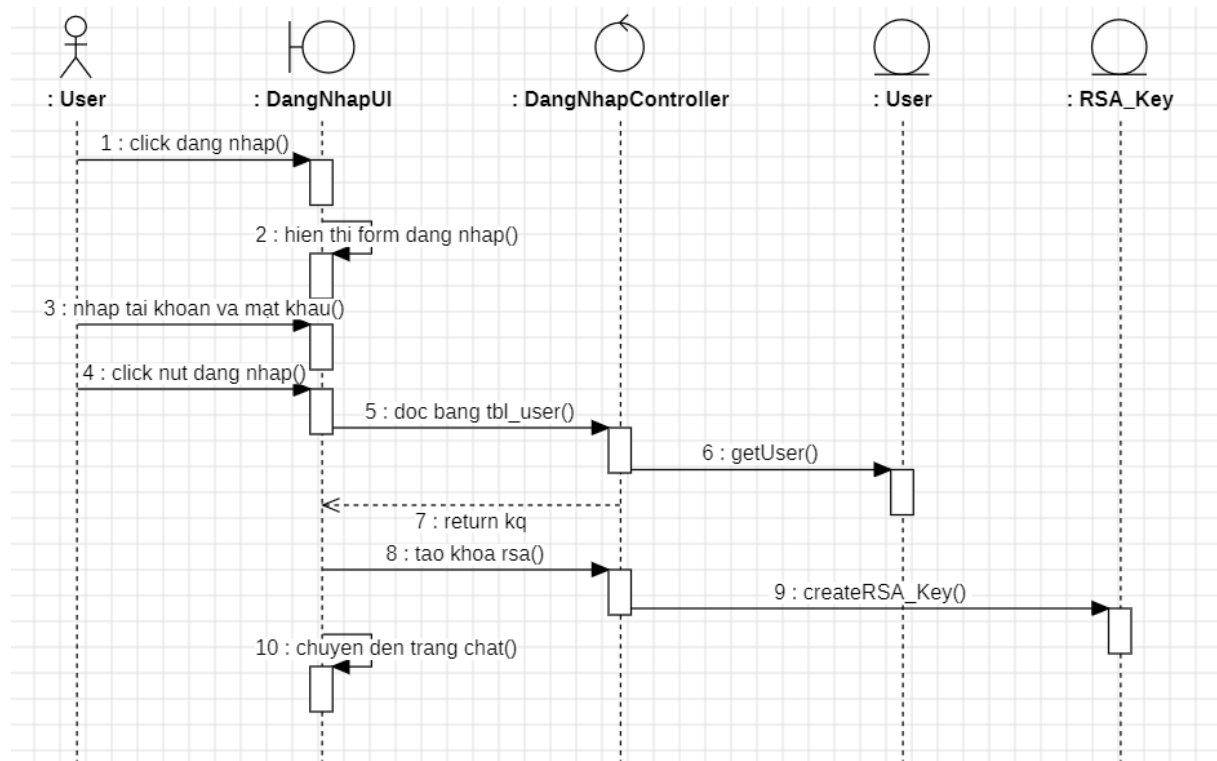
⇒ Các yêu cầu đặc biệt: Không có.

⇒ Tiền điều kiện: Người dùng cần phải có tài khoản đã được đăng ký trước đó để đăng nhập.

⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng sẽ được chuyển đến màn hình chính và thông tin đăng nhập của họ được lưu lại trong phiên làm việc hiện tại.

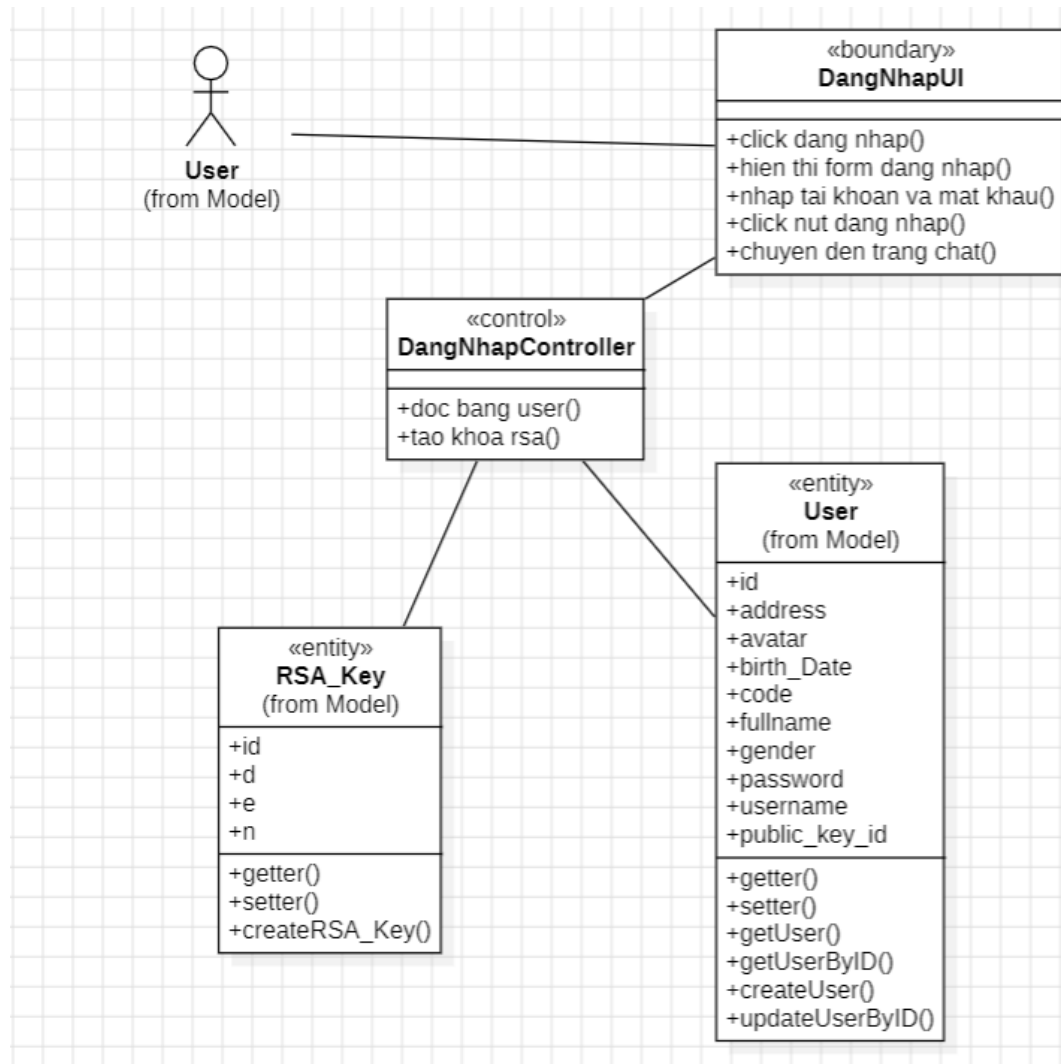
⇒ Điểm mở rộng: Không có.

⇒ Biểu đồ trình tự:



Hình 3.11. Biểu đồ trình tự usecase đăng nhập

⇒ Biểu đồ lớp:



Hình 3.12. Biểu đồ lớp usecase đăng nhập

3.3.2.2. Đặc tả usecase Đăng ký

⇒ Tên use case: Đăng ký

Mô tả vắn tắt: Use case này cho phép người dùng tạo một tài khoản mới trong hệ thống để có thể sử dụng các chức năng và tính năng của ứng dụng.

⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn đăng ký tài khoản mới.

Bước 2: Người dùng nhập thông tin cần thiết như tên, địa chỉ email, mật khẩu, và các thông tin khác liên quan vào các trường trên màn hình đăng ký.

Bước 3: Người dùng kích vào nút "Đăng ký". Hệ thống kiểm tra tính hợp lệ của thông tin và sự sẵn có của địa chỉ email.

Bước 4: Nếu thông tin đăng ký hợp lệ, hệ thống tạo tài khoản mới và chuyển người dùng đến màn hình chính của ứng dụng. Use case kết thúc. Nếu có lỗi hoặc thông tin không hợp lệ, hệ thống đưa ra thông báo lỗi và yêu cầu người dùng nhập lại.

Luồng rẽ nhánh:

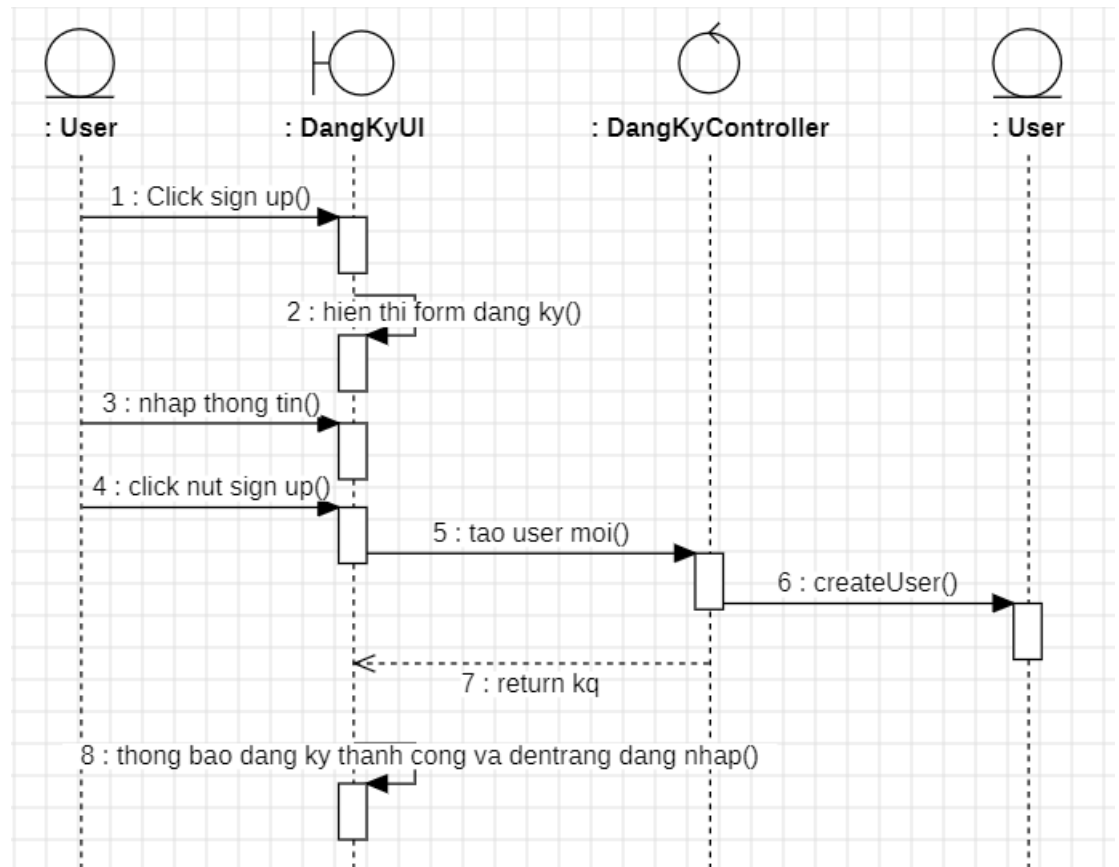
Tại bước 2 ở luồng cơ bản, nếu người dùng kích vào "Quay lại", hệ thống chuyển người dùng đến màn hình đăng nhập và use case kết thúc.

Tại bước 3 ở luồng cơ bản, nếu người dùng nhập email đã tồn tại, hệ thống đưa ra thông báo lỗi và yêu cầu người dùng nhập lại thông tin.

Trong tất cả các bước tại luồng cơ bản, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

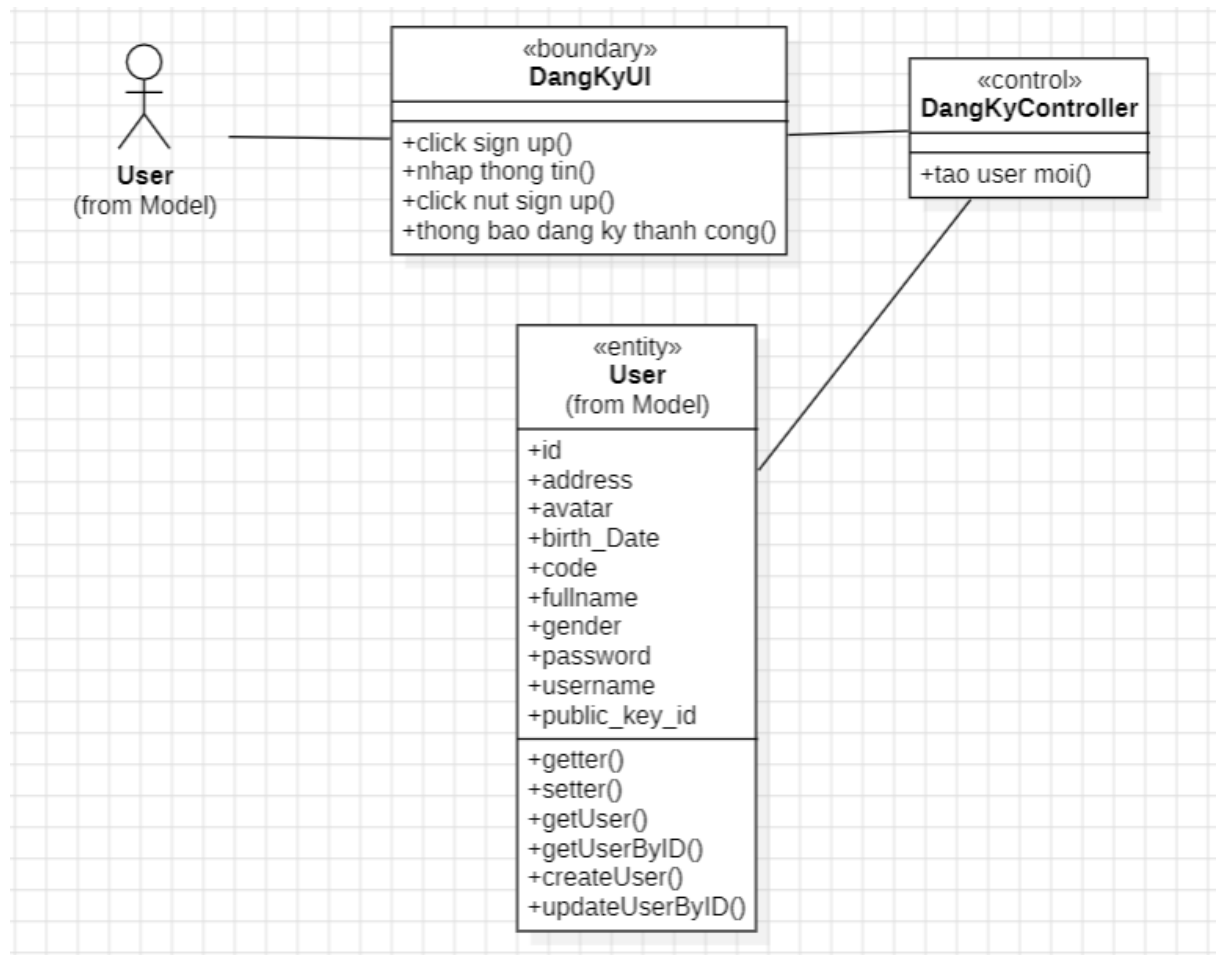
- ⇒ Các yêu cầu đặc biệt: Mật khẩu phải đáp ứng các yêu cầu bảo mật cụ thể, và email phải có định dạng hợp lệ.
- ⇒ Tiền điều kiện: Người dùng không được có tài khoản đã đăng ký với địa chỉ email đã nhập trước đó.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, tài khoản mới sẽ được tạo và thông tin đăng ký sẽ được lưu vào CSDL.
- ⇒ Điểm mở rộng: Không có.

⇒ Biểu đồ trình tự:



Hình 3.13. Biểu đồ trình tự usecase đăng ký

⇒ Biểu đồ lớp:



Hình 3.14. Biểu đồ lớp usecase đăng ký

3.3.2.3. Đặc tả usecase Đăng xuất

- ⇒ Tên use case: Đăng xuất
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng đăng xuất khỏi hệ thống để kết thúc phiên làm việc và bảo vệ thông tin cá nhân.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn đăng xuất khỏi hệ thống.

Bước 2: Người dùng kích vào nút "Đăng xuất" hoặc tương tự trên giao diện người dùng.

Bước 3: Hệ thống xác nhận việc đăng xuất và chuyển người dùng đến màn hình đăng nhập hoặc màn hình chào mừng không đăng nhập. Use case kết thúc.

Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu người dùng chọn "Hủy bỏ" hoặc không thực hiện bất kỳ hành động nào, hệ thống giữ người dùng ở màn hình hiện tại và use case kết thúc.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

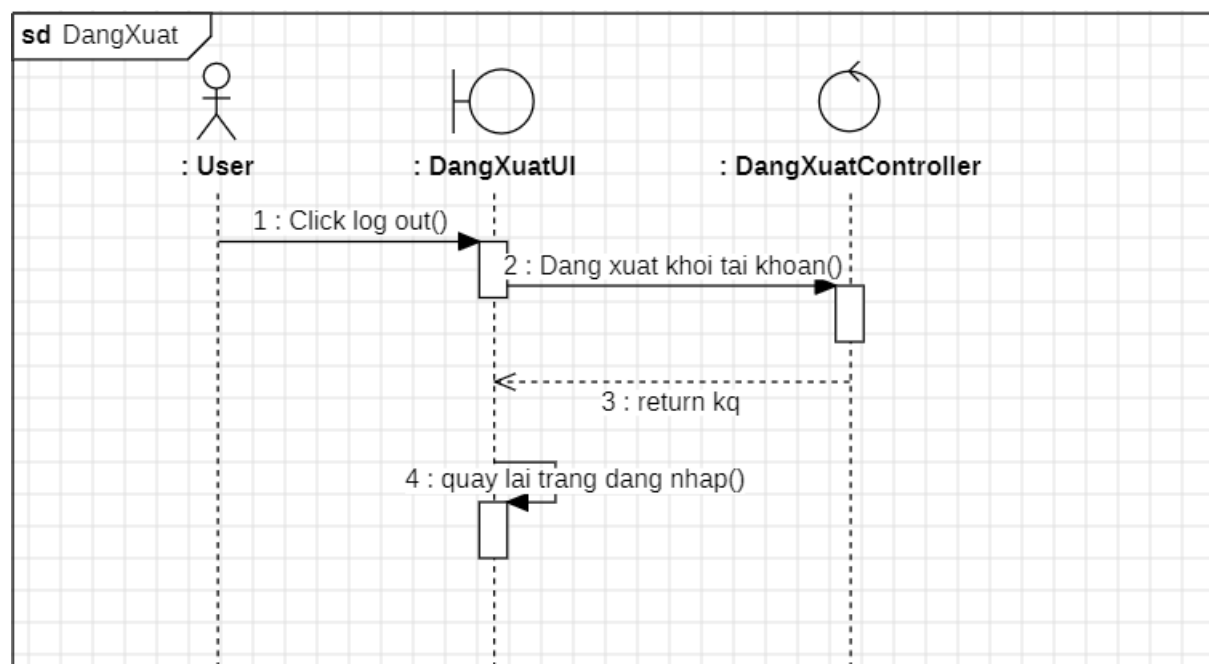
⇒ Các yêu cầu đặc biệt: Không có.

⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống.

⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng sẽ được chuyển đến màn hình đăng nhập hoặc màn hình chào mừng không đăng nhập, và thông tin đăng nhập của họ sẽ không còn hiệu lực trong phiên làm việc hiện tại.

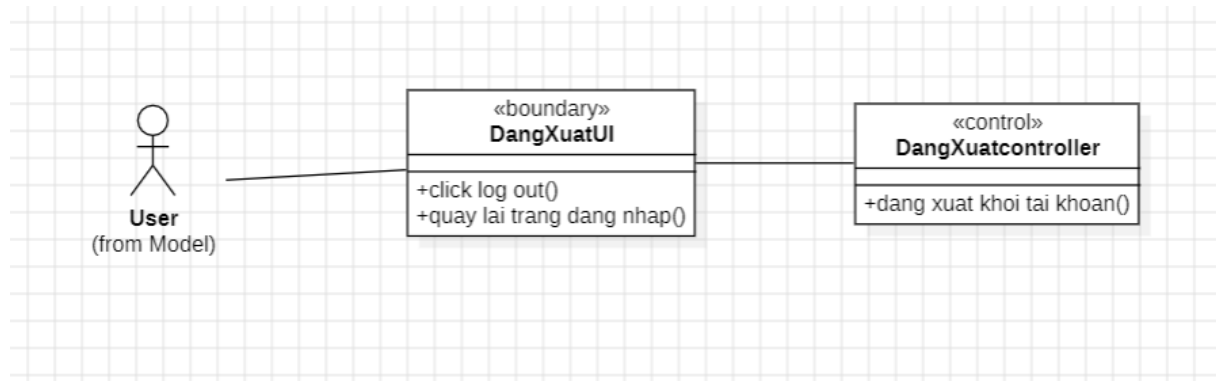
⇒ Điểm mở rộng: Không có.

⇒ Biểu đồ trình tự:



Hình 3.15. Biểu đồ trình tự usecase đăng xuất

⇒ Biểu đồ lớp:



Hình 3.16. Biểu đồ lớp usecase đăng xuất

3.3.2.4. Đặc tả usecase Xem các cuộc trò chuyện tham gia

- ⇒ Tên use case: Xem các cuộc trò chuyện đã tham gia
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng xem danh sách các cuộc trò chuyện mà họ đã tham gia trước đó trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn xem các cuộc trò chuyện đã tham gia.

Bước 2: Người dùng kích vào mục "Cuộc trò chuyện đã tham gia" hoặc tương tự trên giao diện người dùng.

Bước 3: Hệ thống hiển thị danh sách các cuộc trò chuyện đã tham gia của người dùng, bao gồm các thông tin như tên cuộc trò chuyện, người tham gia, và thời gian tham gia.

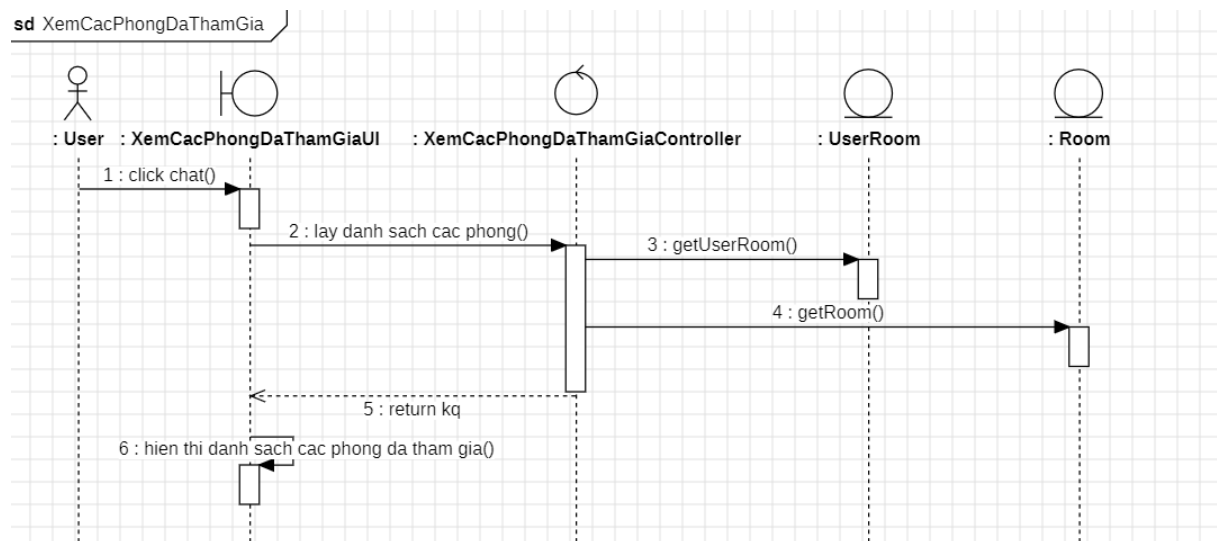
Bước 4: Người dùng có thể chọn một cuộc trò chuyện từ danh sách để xem nội dung chi tiết hoặc tiếp tục thao tác khác. Use case kết thúc.

Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu người dùng không có cuộc trò chuyện nào đã tham gia, hệ thống hiển thị thông báo "Bạn chưa tham gia cuộc trò chuyện nào" và use case kết thúc.

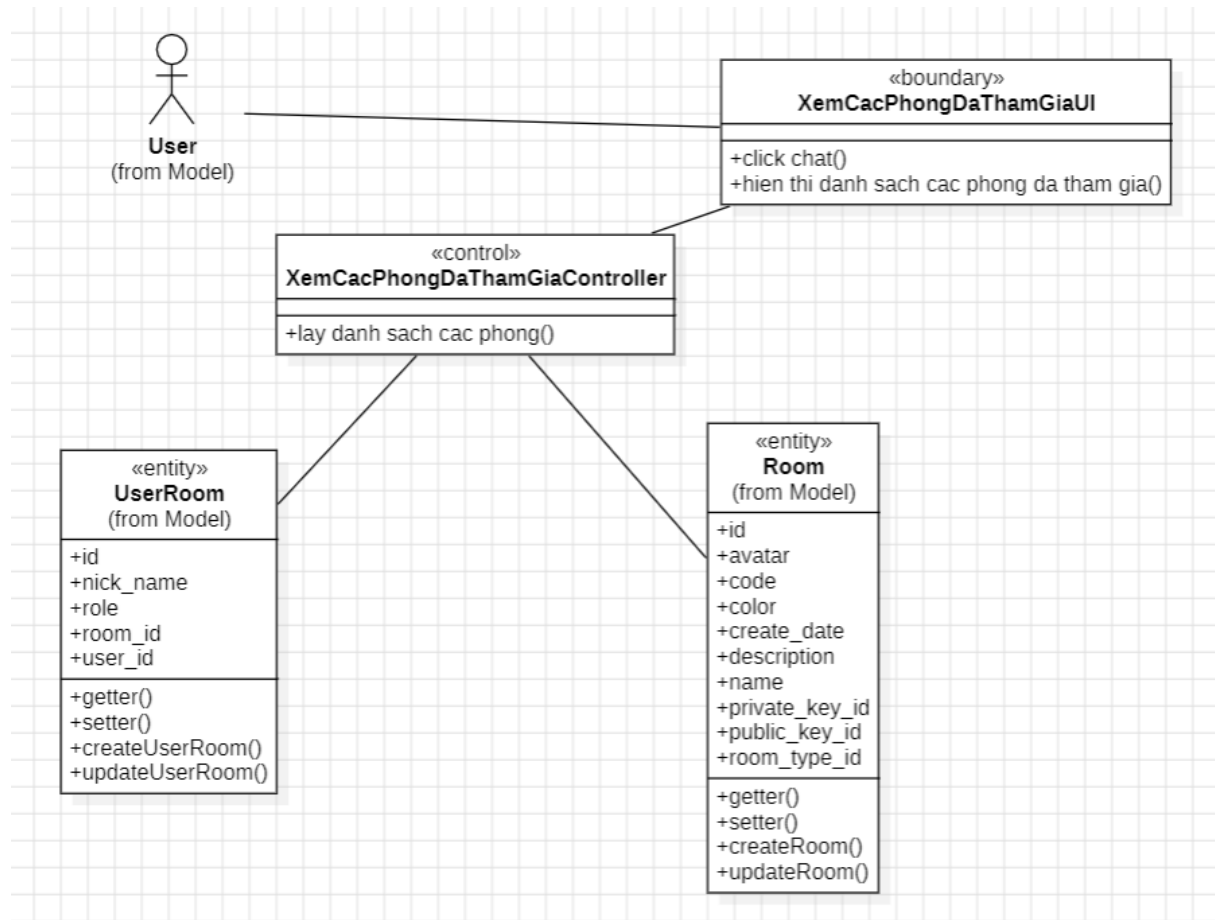
Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

- ⇒ Các yêu cầu đặc biệt: Danh sách cuộc trò chuyện đã tham gia phải được sắp xếp theo thời gian hoặc một tiêu chí khác.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống và đã tham gia ít nhất một cuộc trò chuyện trước đó.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng có thể xem danh sách các cuộc trò chuyện đã tham gia và thực hiện các thao tác khác trên chúng.
- ⇒ Điểm mở rộng: Không có.
- ⇒ Biểu đồ trình tự:



Hình 3.17. Biểu đồ trình tự usecase xem các cuộc trò chuyện đã tham gia

⇒ Biểu đồ lớp:



Hình 3.18. Biểu đồ lớp usecase xem các cuộc trò chuyện đã tham gia

3.3.2.5. Đặc tả usecase Gửi tin nhắn đến cuộc trò chuyện

- ⇒ Tên use case: Xem các cuộc trò chuyện đã tham gia
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng xem danh sách các cuộc trò chuyện mà họ đã tham gia trước đó trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn xem các cuộc trò chuyện đã tham gia.

Bước 2: Người dùng kích vào mục "Cuộc trò chuyện đã tham gia" hoặc tương tự trên giao diện người dùng.

Bước 3: Hệ thống hiển thị danh sách các cuộc trò chuyện đã tham gia của người dùng, bao gồm các thông tin như tên cuộc trò chuyện, người tham gia, và thời gian tham gia.

Bước 4: Người dùng có thể chọn một cuộc trò chuyện từ danh sách để xem nội dung chi tiết hoặc tiếp tục thao tác khác. Use case kết thúc.

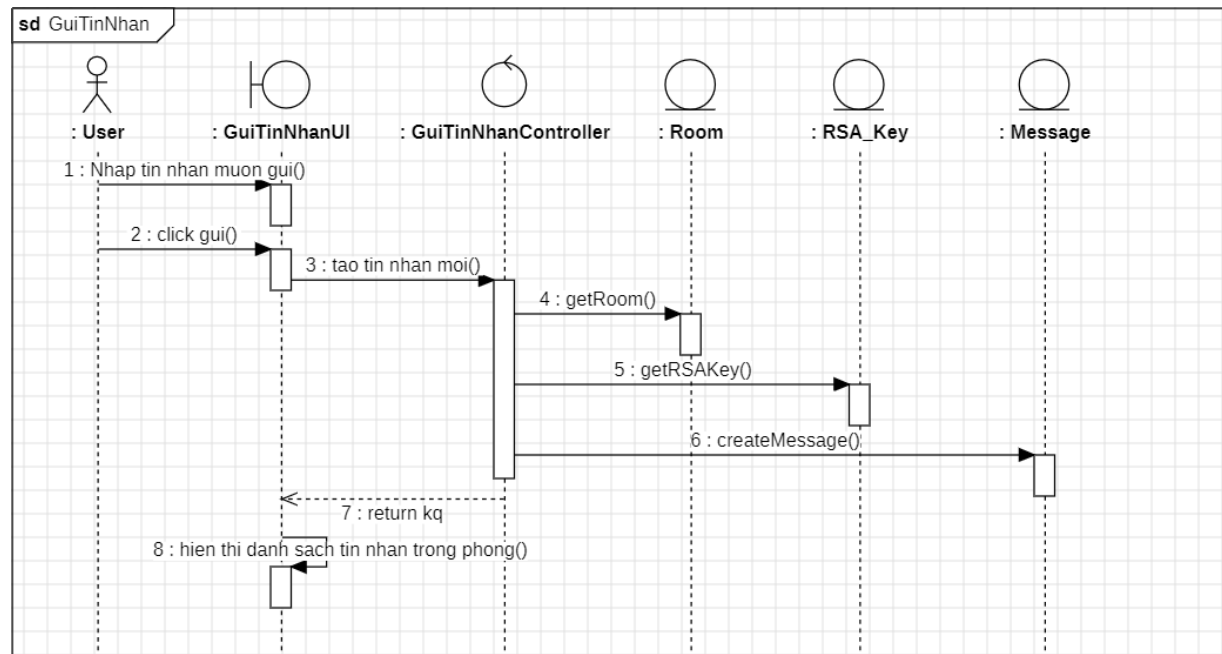
Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu người dùng không có cuộc trò chuyện nào đã tham gia, hệ thống hiển thị thông báo "Bạn chưa tham gia cuộc trò chuyện nào" và use case kết thúc.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

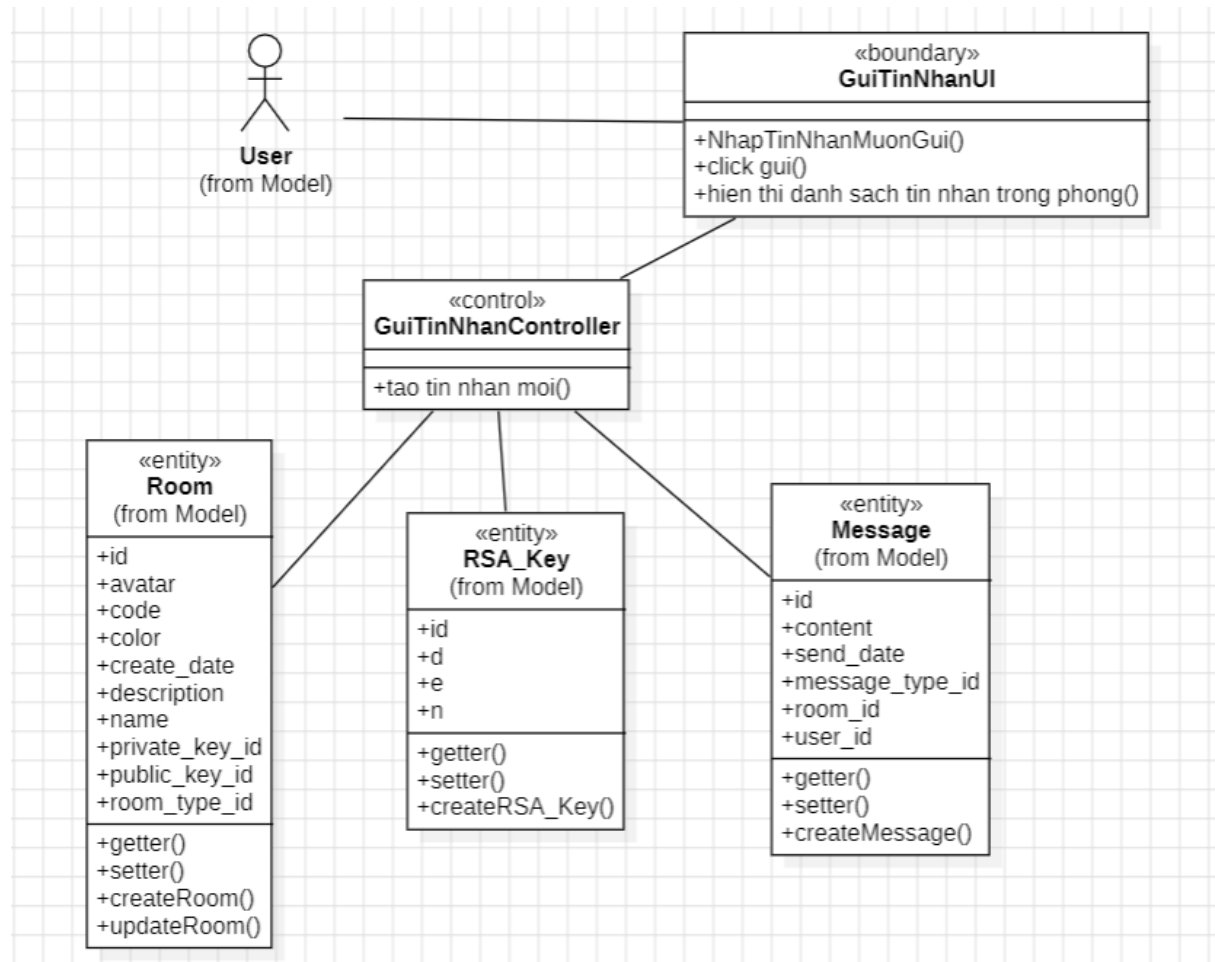
- ⇒ Các yêu cầu đặc biệt: Danh sách cuộc trò chuyện đã tham gia phải được sắp xếp theo thời gian hoặc một tiêu chí khác.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống và đã tham gia ít nhất một cuộc trò chuyện trước đó.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng có thể xem danh sách các cuộc trò chuyện đã tham gia và thực hiện các thao tác khác trên chúng.
- ⇒ Điểm mở rộng: Không có.

⇒ Biểu đồ trình tự:



Hình 3.19. Biểu đồ trình tự usecase gửi tin nhắn

⇒ Biểu đồ lớp:



Hình 3.20. Biểu đồ lớp usecase gửi tin nhắn

3.3.2.6. Đặc tả usecase Gửi lời mời kết bạn

⇒ Tên use case: Gửi lời mời kết bạn

⇒ Mô tả vắn tắt: Use case này cho phép người dùng gửi lời mời kết bạn đến người dùng khác trong hệ thống để tạo mối quan hệ kết nối.

⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn gửi lời mời kết bạn.

Bước 2: Người dùng nhập tên hoặc địa chỉ email của người mà họ muốn kết bạn và kích vào nút "Gửi lời mời".

Bước 3: Hệ thống kiểm tra tính hợp lệ của thông tin nhập vào và xác nhận rằng người dùng được mời chưa nằm trong danh sách bạn bè của người gửi.

Bước 4: Nếu thông tin hợp lệ, hệ thống gửi lời mời kết bạn và thông báo "Lời mời đã được gửi thành công". Use case kết thúc.

Luồng rẽ nhánh:

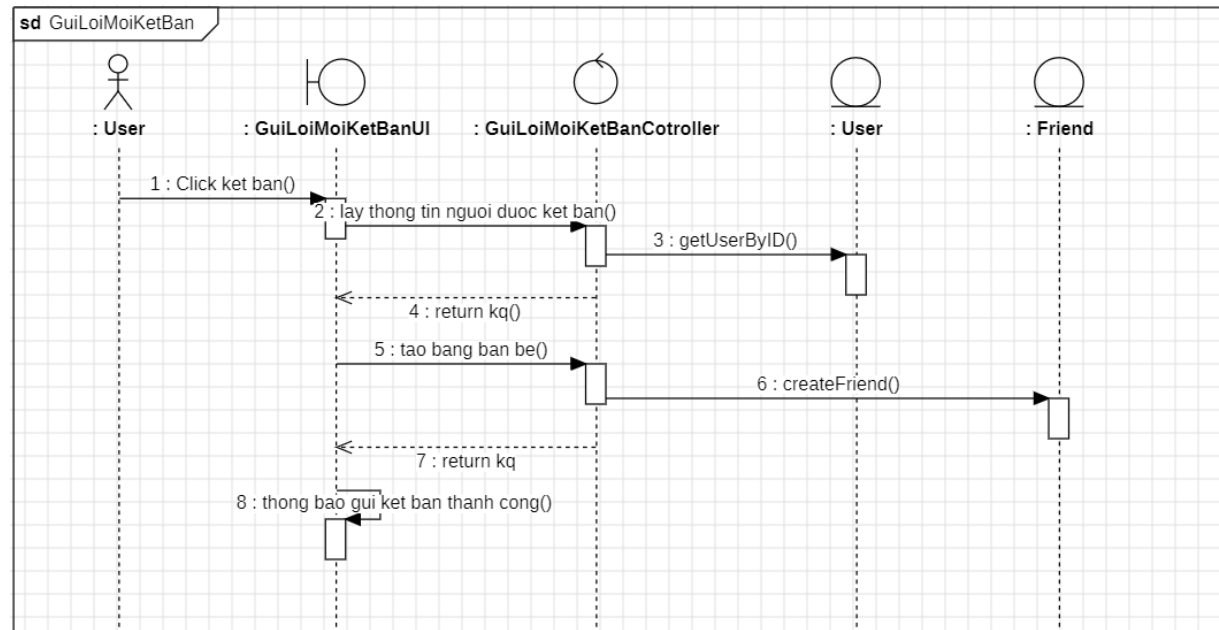
Tại bước 3 ở luồng cơ bản, nếu người dùng được mời đã nằm trong danh sách bạn bè hoặc đã nhận lời mời từ người gửi trước đó, hệ thống hiển thị thông báo "Người dùng này đã là bạn bè của bạn hoặc đã nhận lời mời từ bạn trước đó" và use case kết thúc.

Tại bước 2 ở luồng cơ bản, nếu người dùng nhập thông tin không hợp lệ hoặc không tìm thấy người dùng, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng nhập lại.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

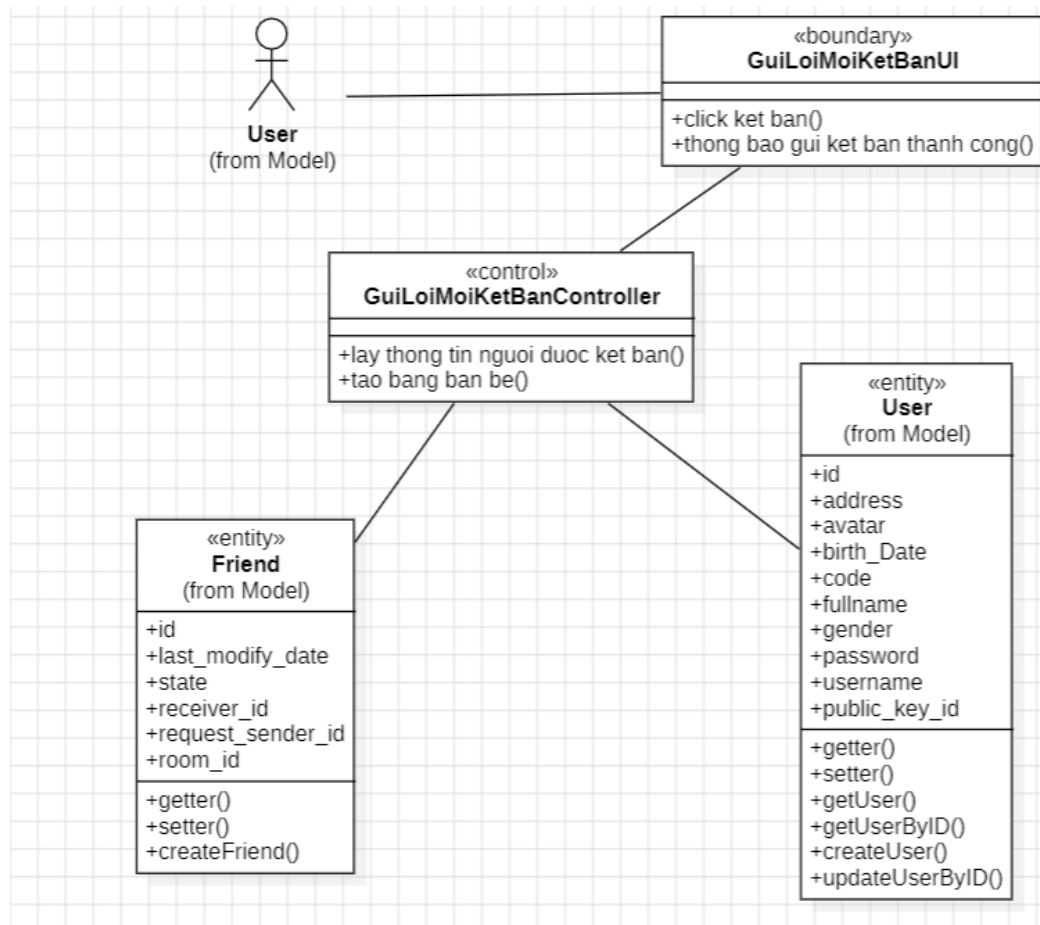
- ⇒ Các yêu cầu đặc biệt: Không có.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, lời mời kết bạn sẽ được gửi và cập nhật trong CSDL.
- ⇒ Điểm mở rộng: Có thể thêm tính năng gửi lời mời kết bạn thông qua các phương tiện truyền thông xã hội khác nhau.

⇒ Biểu đồ trình tự:



Hình 3.21. Biểu đồ trình tự usecase gửi lời mời kết bạn

⇒ Biểu đồ lớp:



Hình 3.22. Biểu đồ lớp usecase gửi lời mời kết bạn

3.3.2.7. Đặc tả usecase Chấp nhận lời mời kết bạn

- ⇒ Tên use case: Chấp nhận lời mời kết bạn
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng chấp nhận lời mời kết bạn từ người dùng khác trong hệ thống để tạo mối quan hệ kết nối.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn chấp nhận lời mời kết bạn.

Bước 2: Người dùng mở thông báo hoặc danh sách lời mời kết bạn và chọn lời mời cụ thể để xem chi tiết.

Bước 3: Người dùng kích vào nút "Chấp nhận" hoặc tương tự để xác nhận việc chấp nhận lời mời kết bạn.

Bước 4: Hệ thống thực hiện quá trình chấp nhận lời mời kết bạn và thông báo "Bạn và [tên người mời] đã trở thành bạn bè". Use case kết thúc.

Luồng rẽ nhánh:

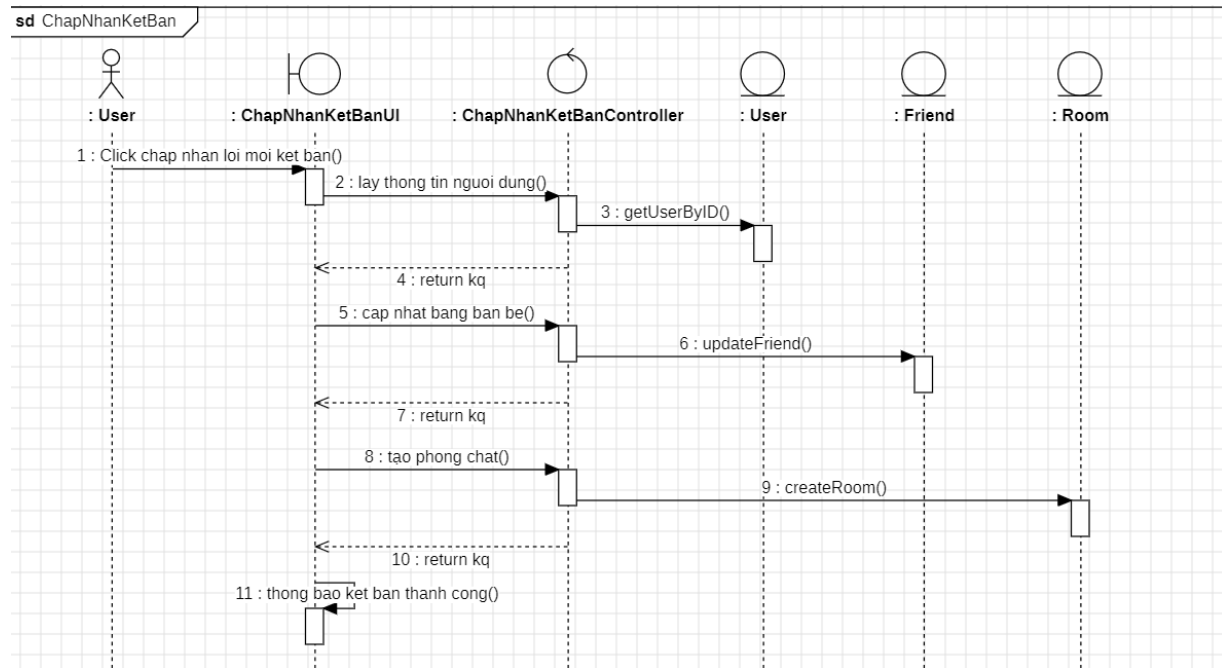
Tại bước 2 ở luồng cơ bản, nếu không có lời mời kết bạn nào hiển thị hoặc người dùng không chọn lời mời cụ thể, hệ thống hiển thị thông báo "Không có lời mời kết bạn hiện tại" và use case kết thúc.

Tại bước 3 ở luồng cơ bản, nếu người gửi lời mời kết bạn đã hủy lời mời trước khi được chấp nhận, hệ thống hiển thị thông báo "Lời mời này đã bị hủy" và use case kết thúc.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

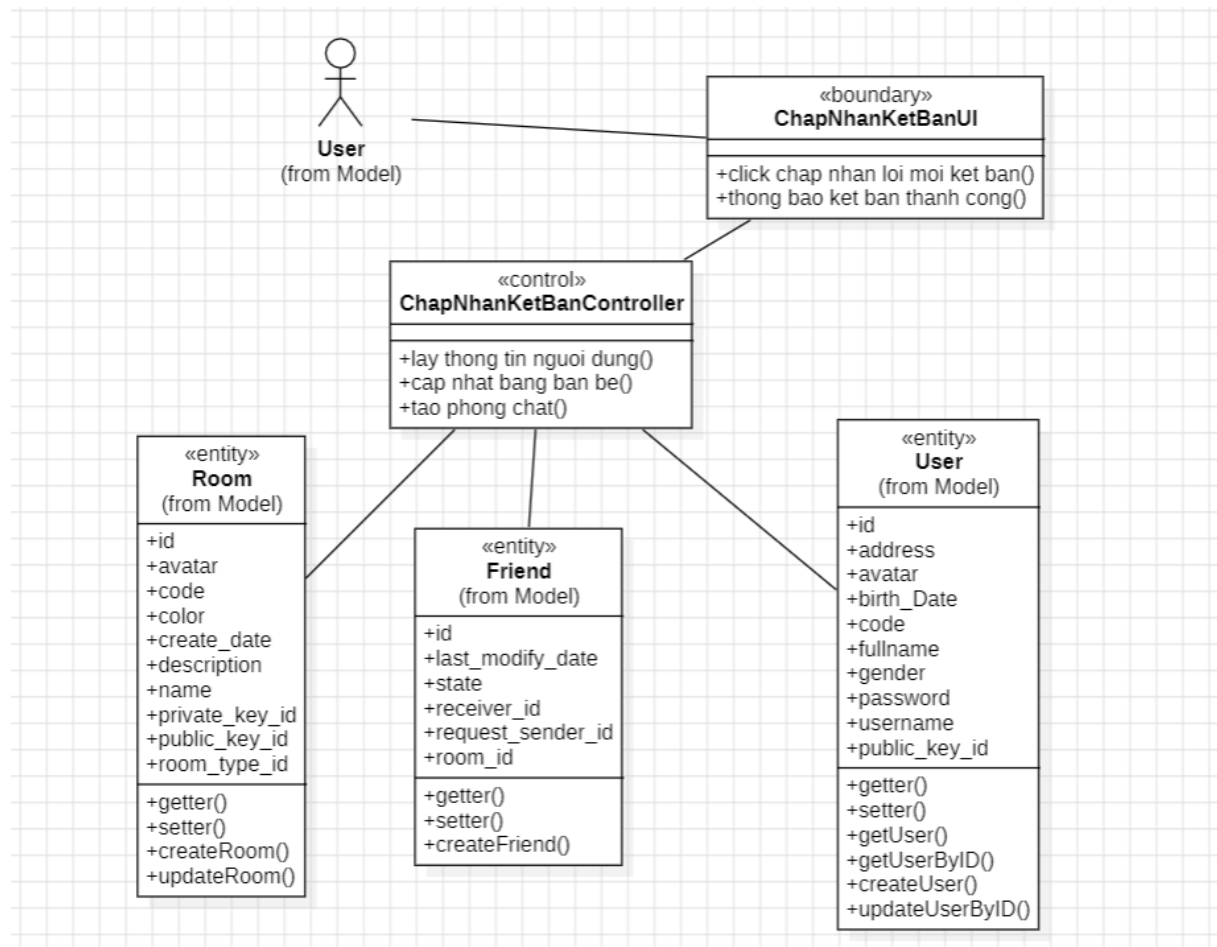
- ⇒ Các yêu cầu đặc biệt: Không có.
- ⇒ Tiên điều kiện: Người dùng đã đăng nhập vào hệ thống và có ít nhất một lời mời kết bạn đang chờ xác nhận.

- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng sẽ trở thành bạn bè với người đã gửi lời mời và thông tin này được cập nhật trong CSDL.
- ⇒ Điểm mở rộng: Có thể thêm tính năng từ chối lời mời kết bạn.
- ⇒ Biểu đồ trình tự:



Hình 3.23. Biểu đồ trình tự usecase chấp nhận kết bạn

⇒ Biểu đồ lớp:



Hình 3.24. Biểu đồ lớp usecase chấp nhận kết bạn

3.3.2.8. Đặc tả usecase Tạo cuộc trò chuyện nhóm

- ⇒ Tên use case: Tạo cuộc trò chuyện nhóm
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng tạo một cuộc trò chuyện nhóm để có thể giao tiếp với nhiều người cùng một lúc trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn tạo một cuộc trò chuyện nhóm mới.

Bước 2: Người dùng chọn tùy chọn "Tạo cuộc trò chuyện nhóm" hoặc tương tự trên giao diện người dùng.

Bước 3: Người dùng nhập tên cho cuộc trò chuyện nhóm và chọn danh sách các thành viên mà họ muốn thêm vào cuộc trò chuyện nhóm.

Bước 4: Người dùng kích vào nút "Tạo" để hoàn tất quá trình tạo cuộc trò chuyện nhóm.

Bước 5: Hệ thống tạo cuộc trò chuyện nhóm mới và thông báo "Cuộc trò chuyện nhóm đã được tạo thành công". Use case kết thúc.

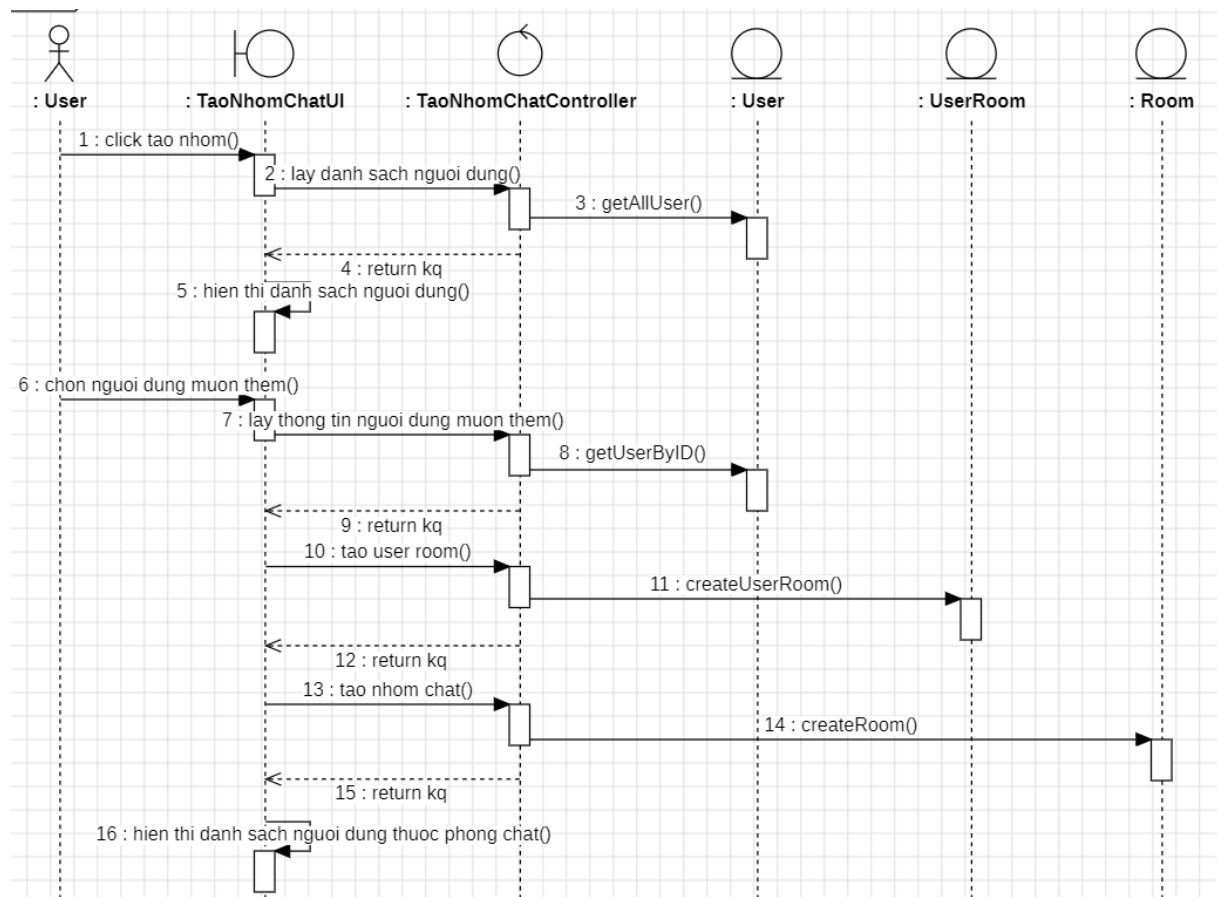
Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu người dùng không nhập tên cho cuộc trò chuyện nhóm hoặc không chọn thành viên nào, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng nhập đầy đủ thông tin.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

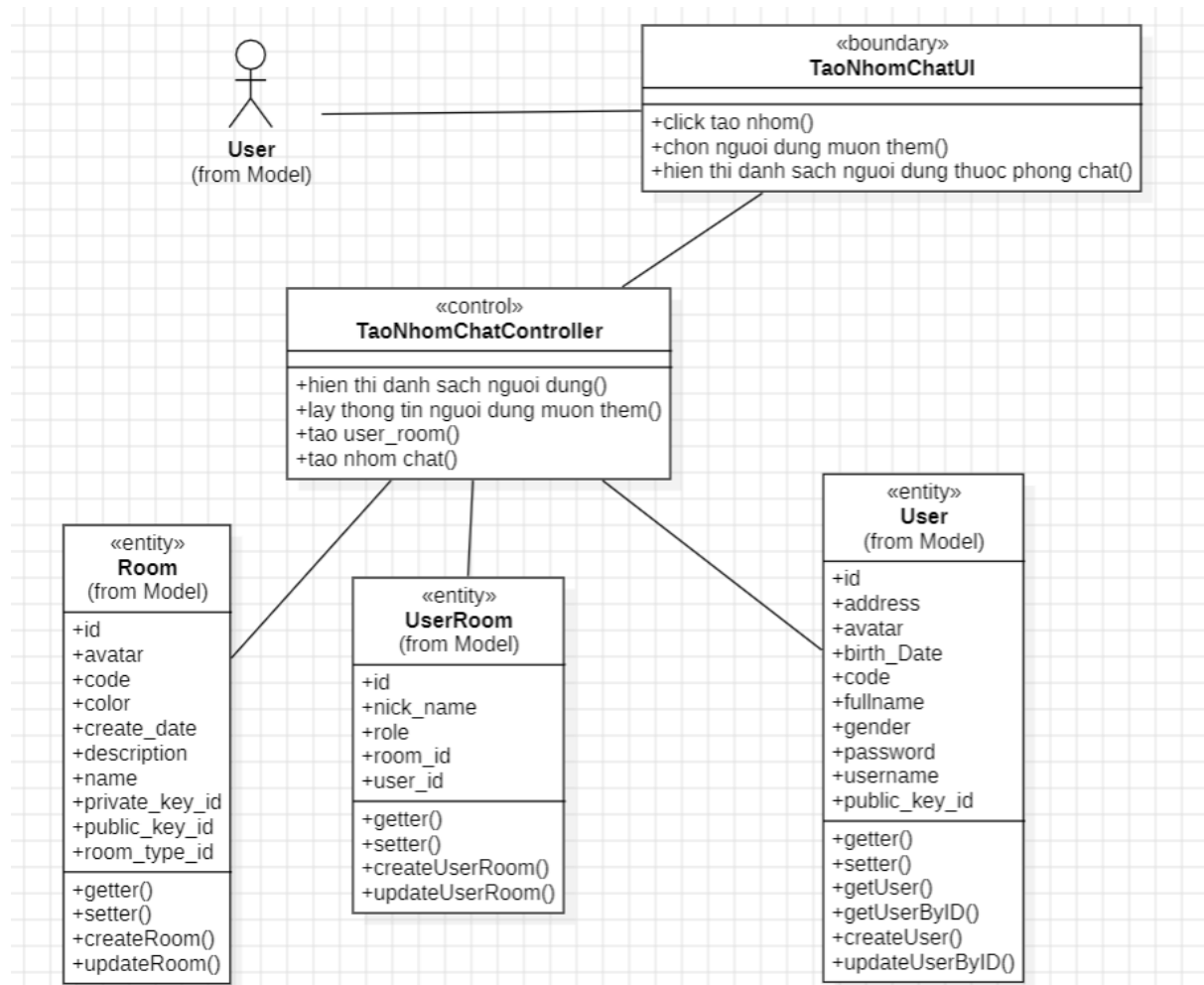
- ⇒ Các yêu cầu đặc biệt: Tên cuộc trò chuyện nhóm không được trùng lặp trong hệ thống và phải đảm bảo tính duy nhất.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, cuộc trò chuyện nhóm mới sẽ được tạo và được liên kết với danh sách thành viên được chọn trong CSDL.
- ⇒ Điểm mở rộng: Có thể thêm tính năng chỉ định quyền truy cập cho các thành viên trong cuộc trò chuyện nhóm.

⇒ Biểu đồ trình tự:



Hình 3.25. Biểu đồ trình tự usecase tạo cuộc trò chuyện nhóm

⇒ Biểu đồ lớp:



Hình 3.26. Biểu đồ lớp usecase tạo cuộc trò chuyện nhóm

3.3.2.9. Đặc tả usecase Tìm kiếm bạn bè

⇒ Tên use case: Tìm kiếm bạn bè

⇒ Mô tả vắn tắt: Use case này cho phép người dùng tìm kiếm và kết nối với bạn bè trong hệ thống.

⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn tìm kiếm bạn bè.

Bước 2: Người dùng nhập tên hoặc địa chỉ email của bạn bè cần tìm kiếm vào ô tìm kiếm trên giao diện người dùng.

Bước 3: Người dùng kích vào nút "Tìm kiếm" để thực hiện tìm kiếm.

Bước 4: Hệ thống hiển thị kết quả tìm kiếm, bao gồm danh sách bạn bè phù hợp với thông tin tìm kiếm.

Bước 5: Người dùng có thể chọn một hoặc nhiều bạn bè từ danh sách và kích vào nút "Kết nối" để gửi lời mời kết bạn.

Bước 6: Hệ thống thực hiện quá trình gửi lời mời kết bạn và thông báo "Lời mời đã được gửi thành công". Use case kết thúc.

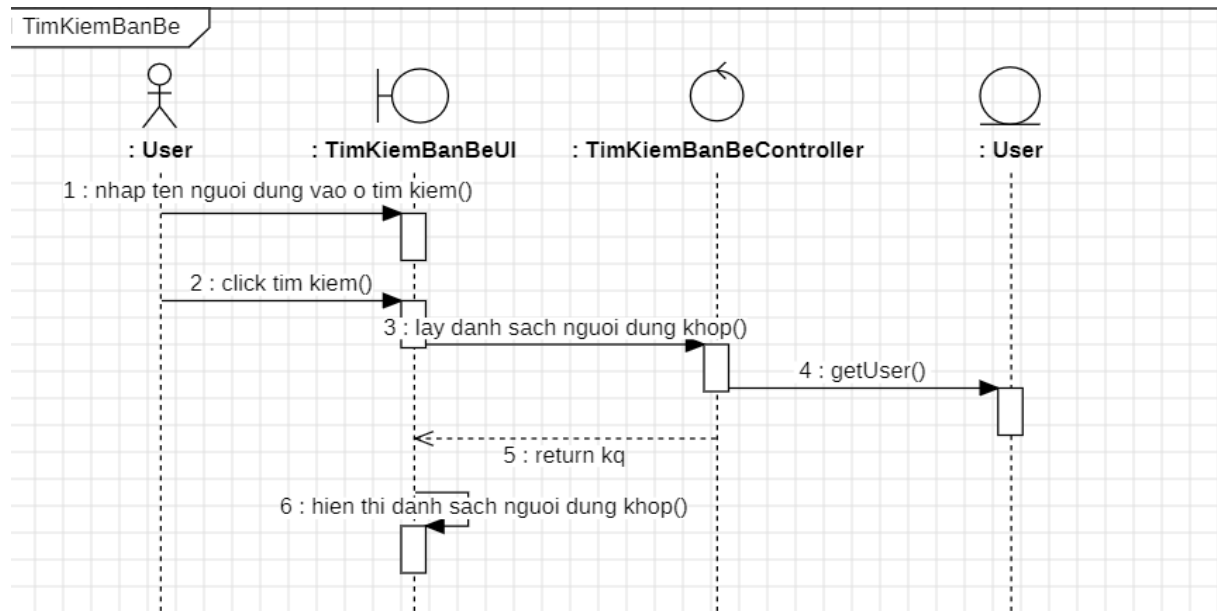
Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu không có kết quả tìm kiếm nào phù hợp, hệ thống hiển thị thông báo "Không tìm thấy kết quả phù hợp".

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

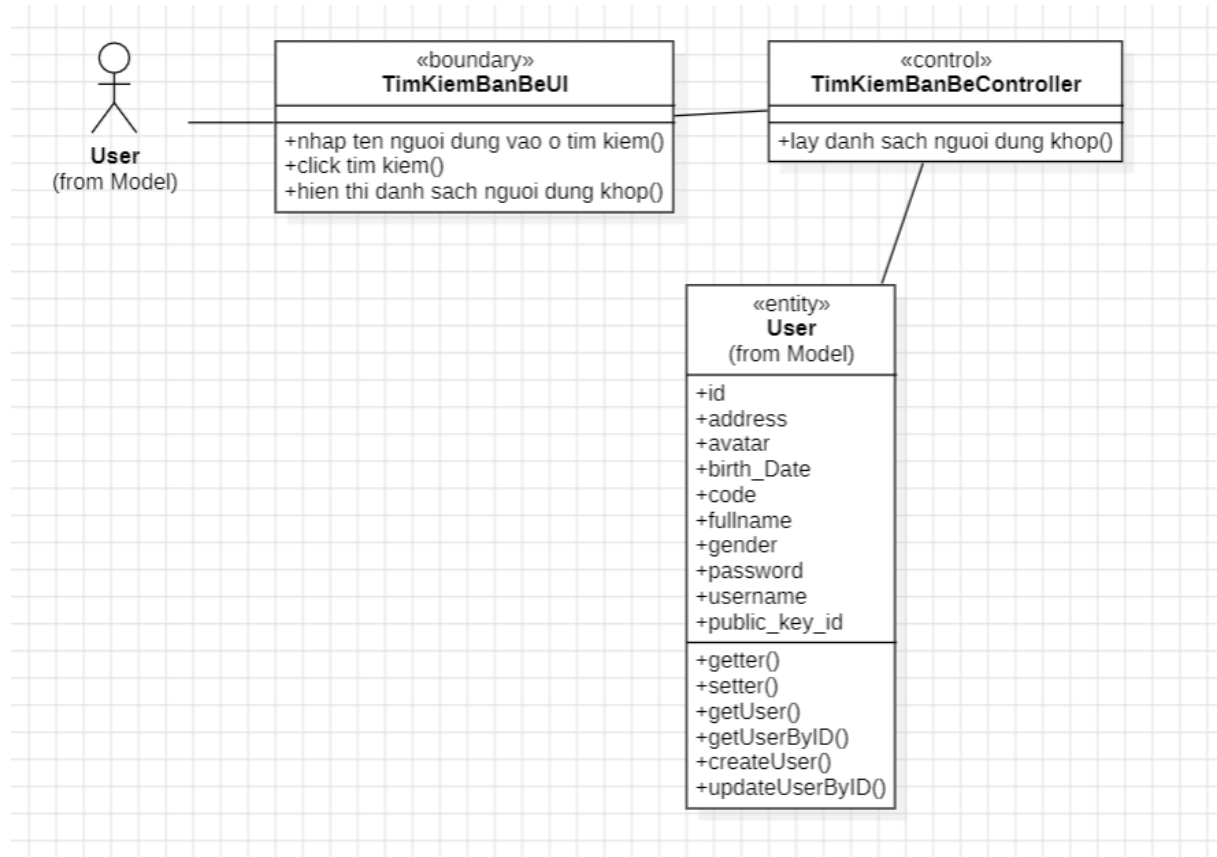
- ⇒ Các yêu cầu đặc biệt: Không có.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, lời mời kết bạn sẽ được gửi và thông tin này được cập nhật trong CSDL.
- ⇒ Điểm mở rộng: Có thể thêm tính năng tìm kiếm bạn bè theo các tiêu chí khác như địa điểm, sở thích, hoặc trạng thái hoạt động.

⇒ Biểu đồ trình tự:



Hình 3.27. Biểu đồ trình tự usecase tìm kiếm bạn bè

⇒ Biểu đồ lớp:



Hình 3.28. Biểu đồ lớp usecase tìm kiếm bạn bè

3.3.2.10. Đặc tả usecase Tìm kiếm cuộc trò chuyện đã tham gia

- ⇒ Tên use case: Tìm kiếm cuộc trò chuyện đã tham gia
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng tìm kiếm và xem các cuộc trò chuyện mà họ đã tham gia trước đó trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn tìm kiếm cuộc trò chuyện đã tham gia.

Bước 2: Người dùng nhập từ khóa tìm kiếm vào ô tìm kiếm trên giao diện người dùng.

Bước 3: Người dùng kích vào nút "Tìm kiếm" để thực hiện tìm kiếm.

Bước 4: Hệ thống hiển thị kết quả tìm kiếm, bao gồm danh sách các cuộc trò chuyện đã tham gia có chứa từ khóa tìm kiếm.

Bước 5: Người dùng có thể chọn một cuộc trò chuyện từ danh sách để xem nội dung chi tiết hoặc thực hiện các thao tác khác trên cuộc trò chuyện. Use case kết thúc.

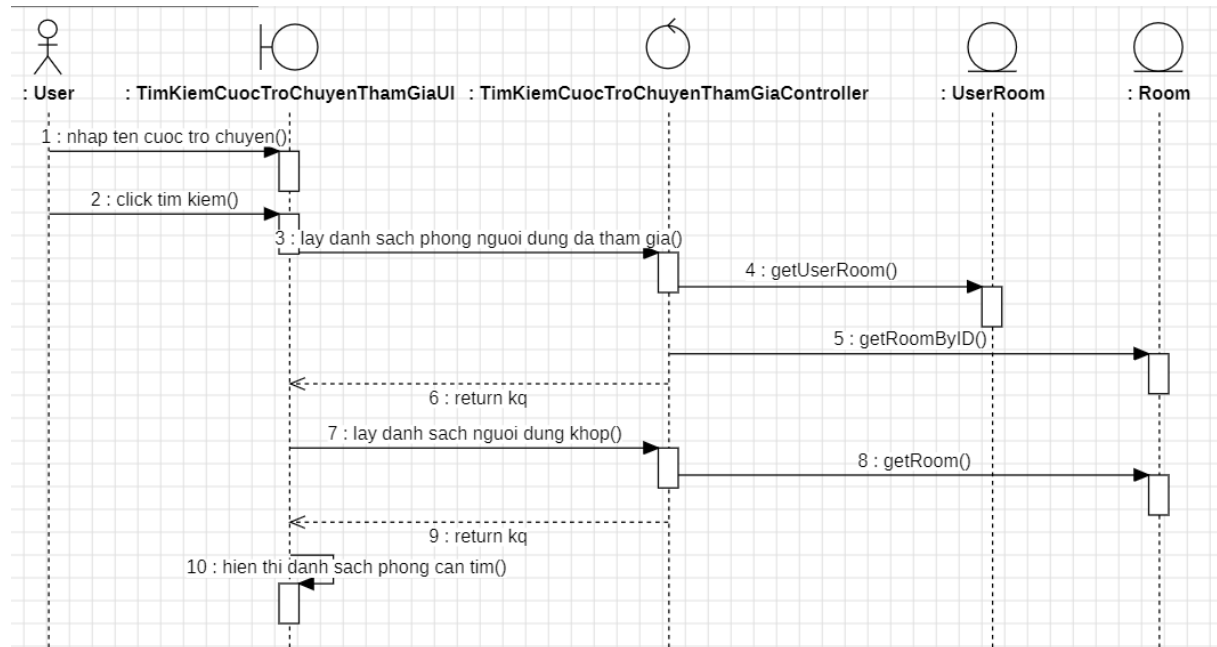
Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu không có kết quả tìm kiếm nào phù hợp, hệ thống hiển thị thông báo "Không tìm thấy cuộc trò chuyện nào phù hợp".

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

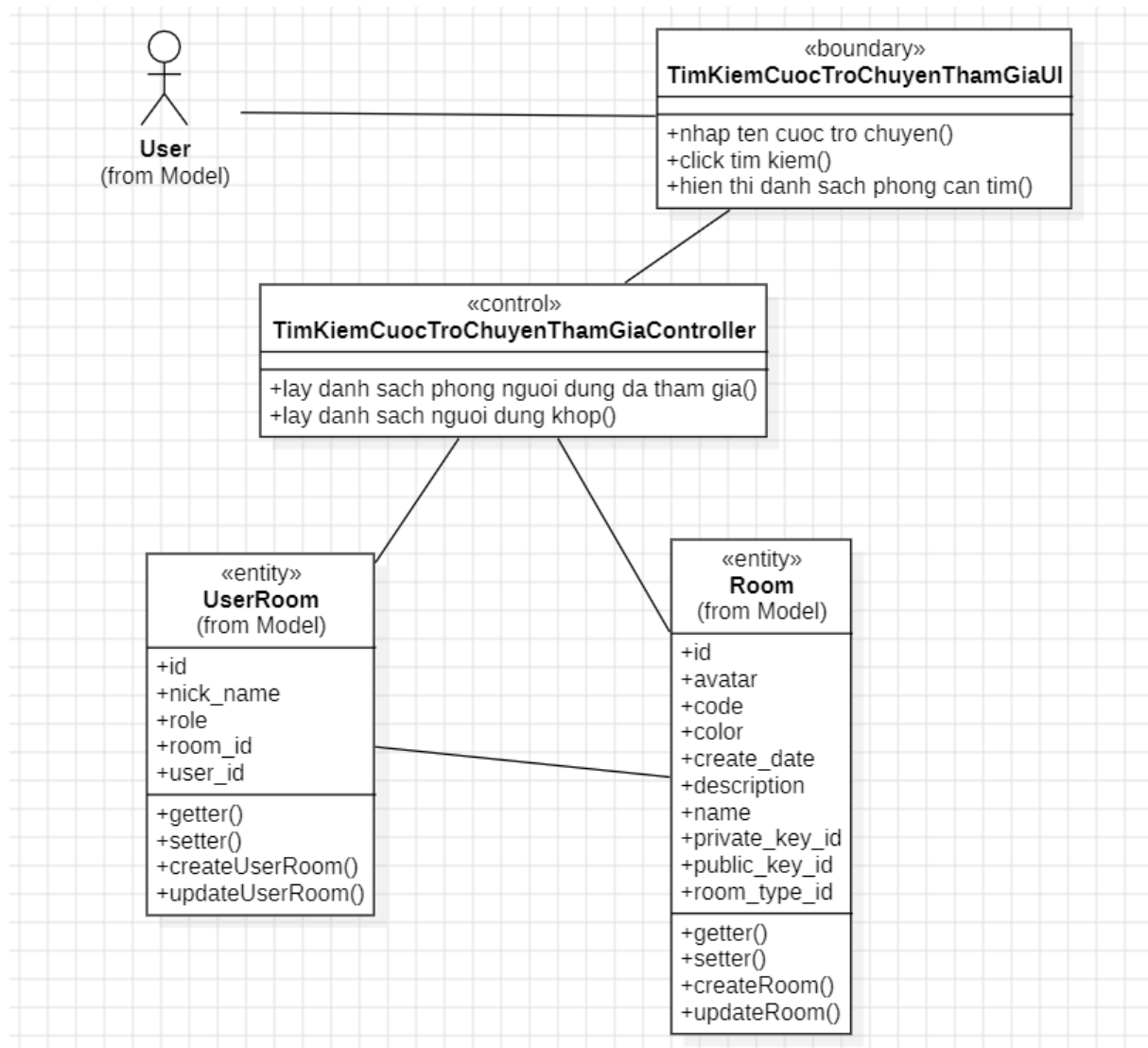
- ⇒ Các yêu cầu đặc biệt: Không có.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống và đã tham gia ít nhất một cuộc trò chuyện trước đó.

- ⇒ **Hậu điều kiện:** Nếu use case kết thúc thành công, người dùng có thể xem danh sách cuộc trò chuyện đã tham gia và thực hiện các thao tác khác trên chúng.
- ⇒ **Điểm mở rộng:** Có thể thêm tính năng tìm kiếm cuộc trò chuyện theo các tiêu chí khác như thời gian, người tham gia, hoặc loại cuộc trò chuyện.
- ⇒ **Biểu đồ trình tự:**



Hình 3.29. Biểu đồ trình tự usecase tìm kiếm cuộc trò chuyện đã tham gia

⇒ Biểu đồ lớp:



Hình 3.30. Biểu đồ lớp usecase tìm kiếm cuộc trò chuyện đã tham gia

3.3.2.11. Đặc tả usecase Cập nhật thông tin cá nhân

⇒ Tên use case: Cập nhật thông tin cá nhân

⇒ Mô tả vắn tắt: Use case này cho phép người dùng cập nhật thông tin cá nhân của mình trong hệ thống.

⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn cập nhật thông tin cá nhân.

Bước 2: Người dùng vào phần "Cài đặt" hoặc tương tự trên giao diện người dùng.

Bước 3: Người dùng chọn tùy chọn "Cập nhật thông tin cá nhân".

Bước 4: Hệ thống hiển thị mẫu biểu mẫu chứa thông tin cá nhân hiện tại của người dùng, cho phép họ chỉnh sửa thông tin cần thiết.

Bước 5: Người dùng nhập thông tin mới và kích vào nút "Lưu" để cập nhật thông tin cá nhân.

Bước 6: Hệ thống thực hiện quá trình cập nhật và thông báo "Thông tin cá nhân đã được cập nhật thành công". Use case kết thúc.

Luồng rẽ nhánh:

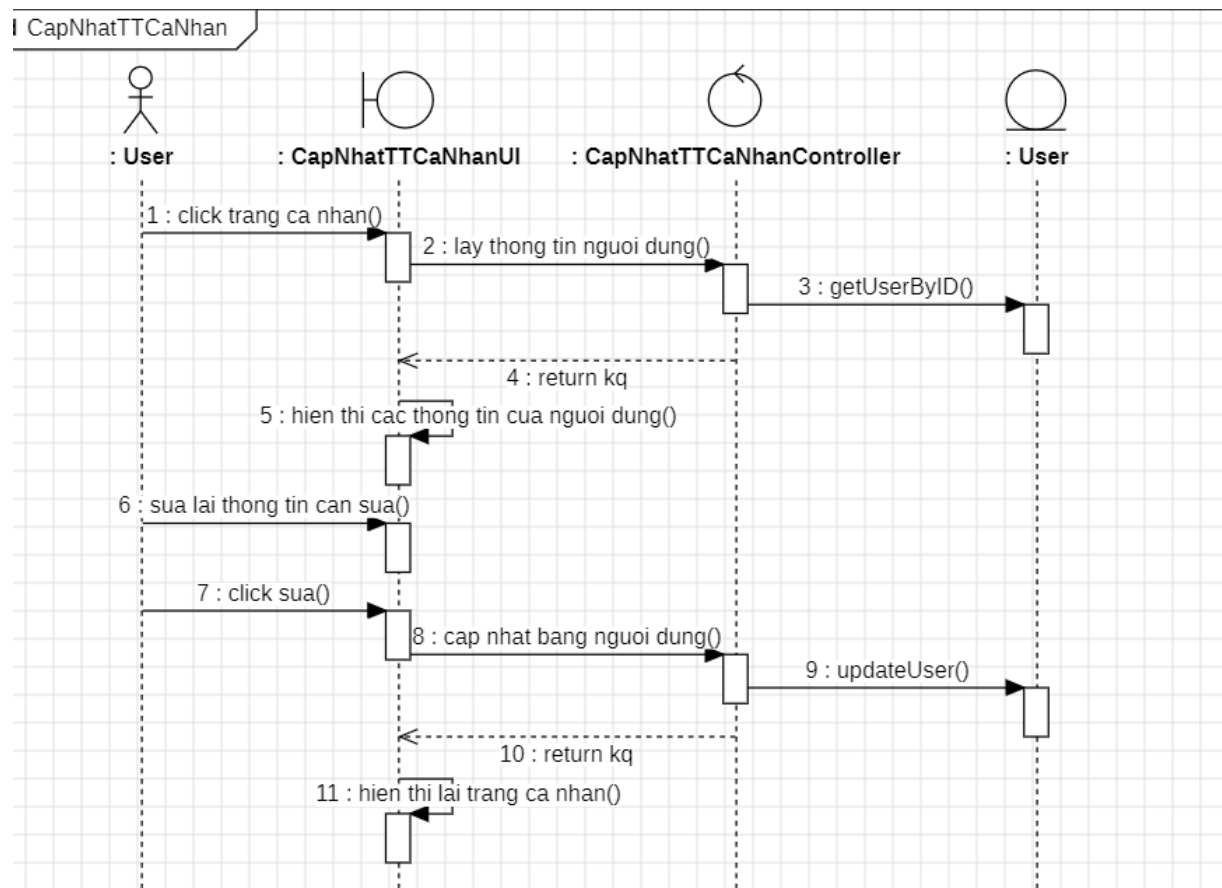
Tại bước 2 ở luồng cơ bản, nếu người dùng không chọn tùy chọn "Cập nhật thông tin cá nhân", hệ thống giữ người dùng ở màn hình cài đặt và use case kết thúc.

Tại bước 5 ở luồng cơ bản, nếu người dùng không nhập thông tin mới hoặc nhập thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng nhập lại.

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

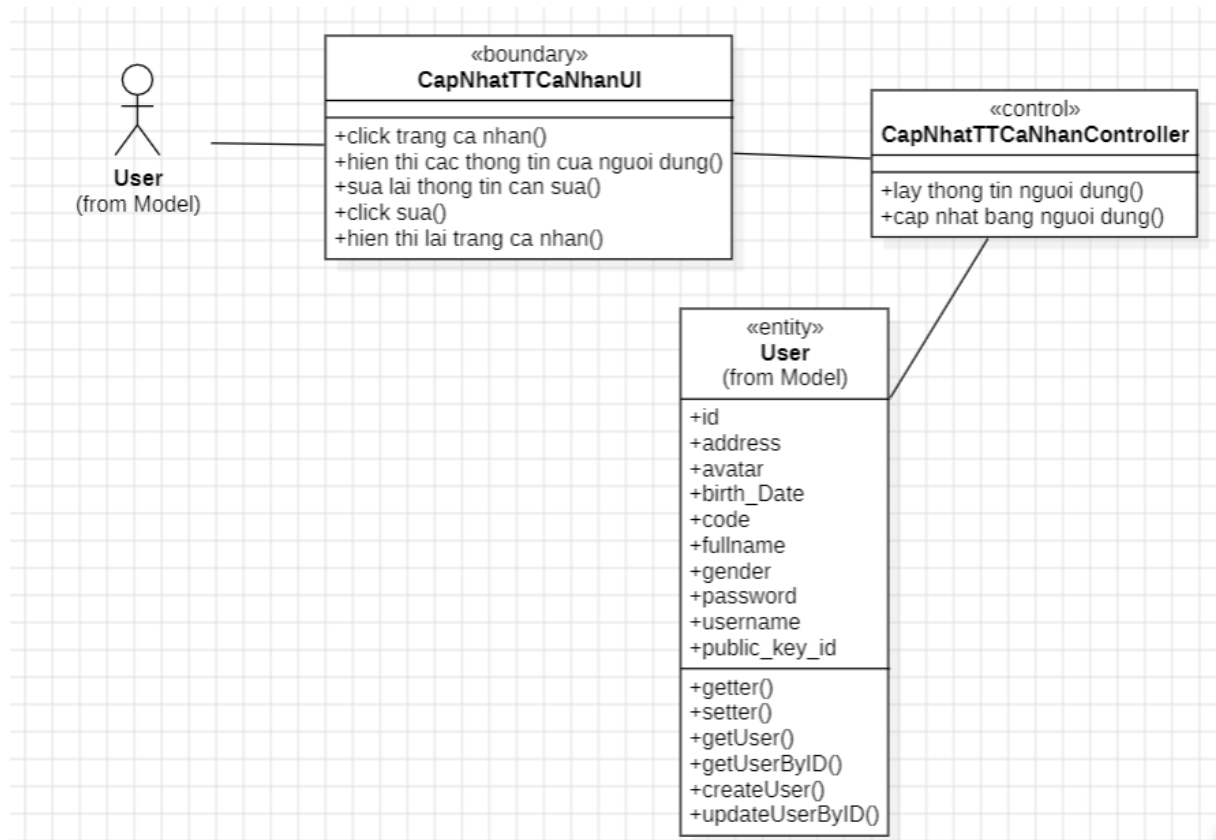
- ⇒ Các yêu cầu đặc biệt: Các trường thông tin cá nhân cần phải đảm bảo tính hợp lệ và tuân thủ theo các quy định an toàn dữ liệu.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, thông tin cá nhân của người dùng sẽ được cập nhật trong CSDL.
- ⇒ Điểm mở rộng: Có thể thêm các tùy chọn cập nhật ảnh đại diện, mật khẩu, hoặc các thông tin khác liên quan đến tài khoản người dùng.

⇒ Biểu đồ trình tự:



Hình 3.31. Biểu đồ trình tự usecase cập nhật thông tin cá nhân

⇒ Biểu đồ lớp:



Hình 3.32. Biểu đồ lớp usecase cập nhật thông tin cá nhân

3.3.2.12. Đặc tả usecase Xem trang cá nhân bạn bè

- ⇒ Tên use case: Xem trang cá nhân bạn bè
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng xem thông tin cá nhân của bạn bè trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn xem trang cá nhân của một người bạn trong hệ thống.

Bước 2: Người dùng vào trang "Danh sách bạn bè" hoặc tương tự trên giao diện người dùng.

Bước 3: Người dùng chọn bạn bè cần xem trang cá nhân và kích vào tên hoặc ảnh đại diện của họ.

Bước 4: Hệ thống hiển thị trang cá nhân của bạn bè, bao gồm các thông tin như tên, ảnh đại diện, trạng thái, và các thông tin cá nhân khác nếu được công khai.

Bước 5: Người dùng có thể xem và thực hiện các thao tác khác như gửi tin nhắn, gửi lời mời kết bạn, hoặc chuyển đến các phần khác của trang cá nhân. Use case kết thúc.

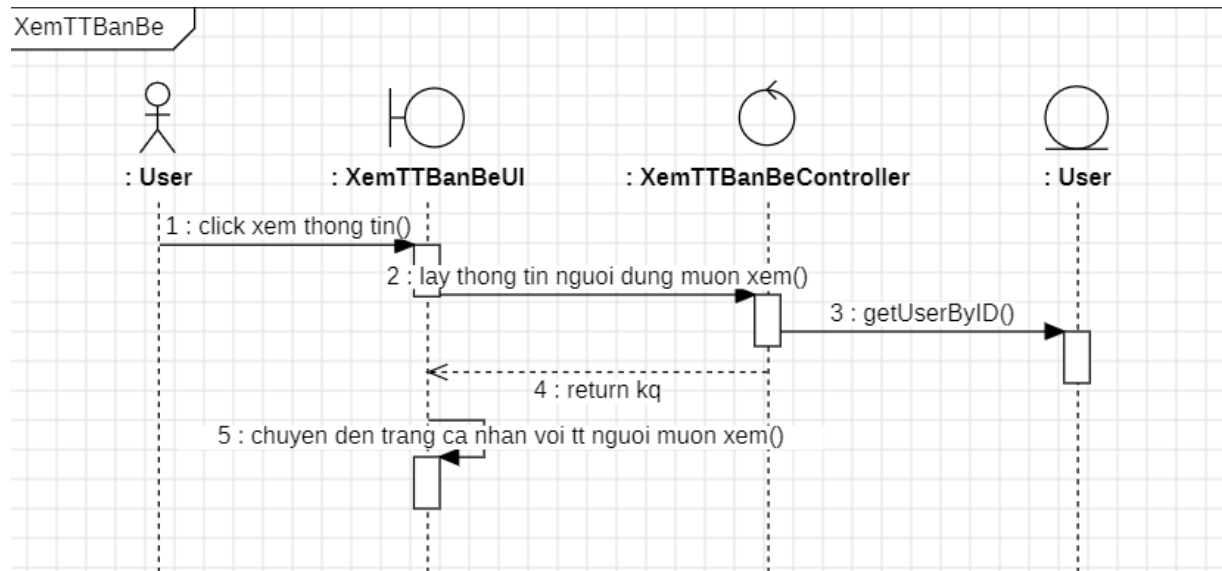
Luồng rẽ nhánh:

Tại bước 3 ở luồng cơ bản, nếu người dùng không chọn bạn bè nào hoặc bạn bè không tồn tại, hệ thống hiển thị thông báo "Không tìm thấy thông tin cá nhân của bạn bè".

Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

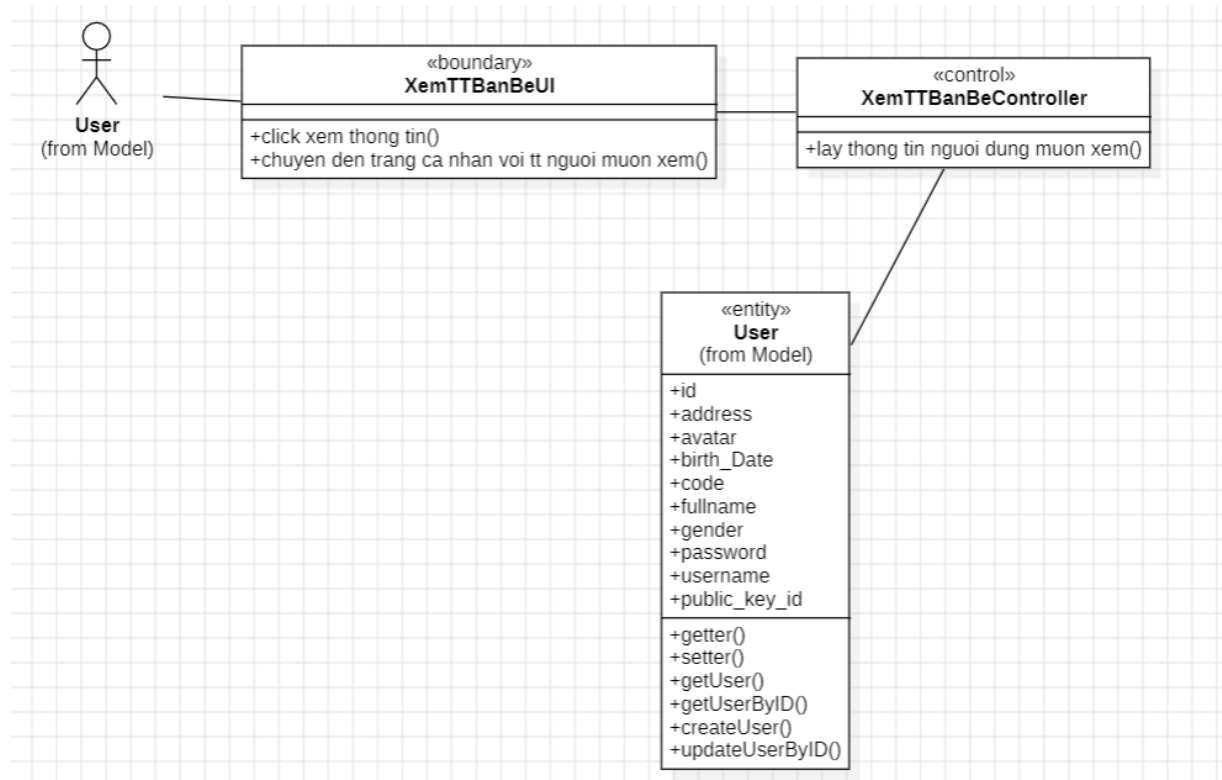
- ⇒ Các yêu cầu đặc biệt: Thông tin cá nhân của bạn bè cần phải tuân thủ theo quy định về quyền riêng tư và an toàn dữ liệu.
- ⇒ Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống và có ít nhất một bạn bè trong danh sách bạn bè.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, người dùng có thể xem thông tin cá nhân của bạn bè và thực hiện các thao tác khác trên trang cá nhân đó.
- ⇒ Điểm mở rộng: Có thể thêm tính năng xem hoạt động gần đây, các bài viết, hoặc thông tin khác của bạn bè.

⇒ Biểu đồ trình tự:



Hình 3.33. Biểu đồ trình tự usecase xem trang cá nhân bạn bè

⇒ Biểu đồ lớp:



Hình 3.34. Biểu đồ lớp usecase xem trang cá nhân bạn bè

3.3.2.13. Đặc tả usecase Cập nhật cuộc trò chuyện

- ⇒ Tên use case: Cập nhật cuộc trò chuyện
- ⇒ Mô tả vắn tắt: Use case này cho phép người dùng cập nhật thông tin hoặc cài đặt của một cuộc trò chuyện trong hệ thống.
- ⇒ Luồng sự kiện:

Luồng cơ bản:

Bước 1: Use case này bắt đầu khi người dùng muốn cập nhật thông tin hoặc cài đặt của một cuộc trò chuyện.

Bước 2: Người dùng vào cuộc trò chuyện cần cập nhật.

Bước 3: Người dùng chọn tùy chọn "Cập nhật" hoặc tương tự trên giao diện người dùng.

Bước 4: Hệ thống hiển thị mẫu biểu mẫu chứa thông tin hoặc cài đặt hiện tại của cuộc trò chuyện, cho phép họ chỉnh sửa thông tin cần thiết.

Bước 5: Người dùng nhập thông tin mới hoặc thực hiện cài đặt mới và kích vào nút "Lưu" để cập nhật cuộc trò chuyện.

Bước 6: Hệ thống thực hiện quá trình cập nhật và thông báo "Thông tin cuộc trò chuyện đã được cập nhật thành công". Use case kết thúc.

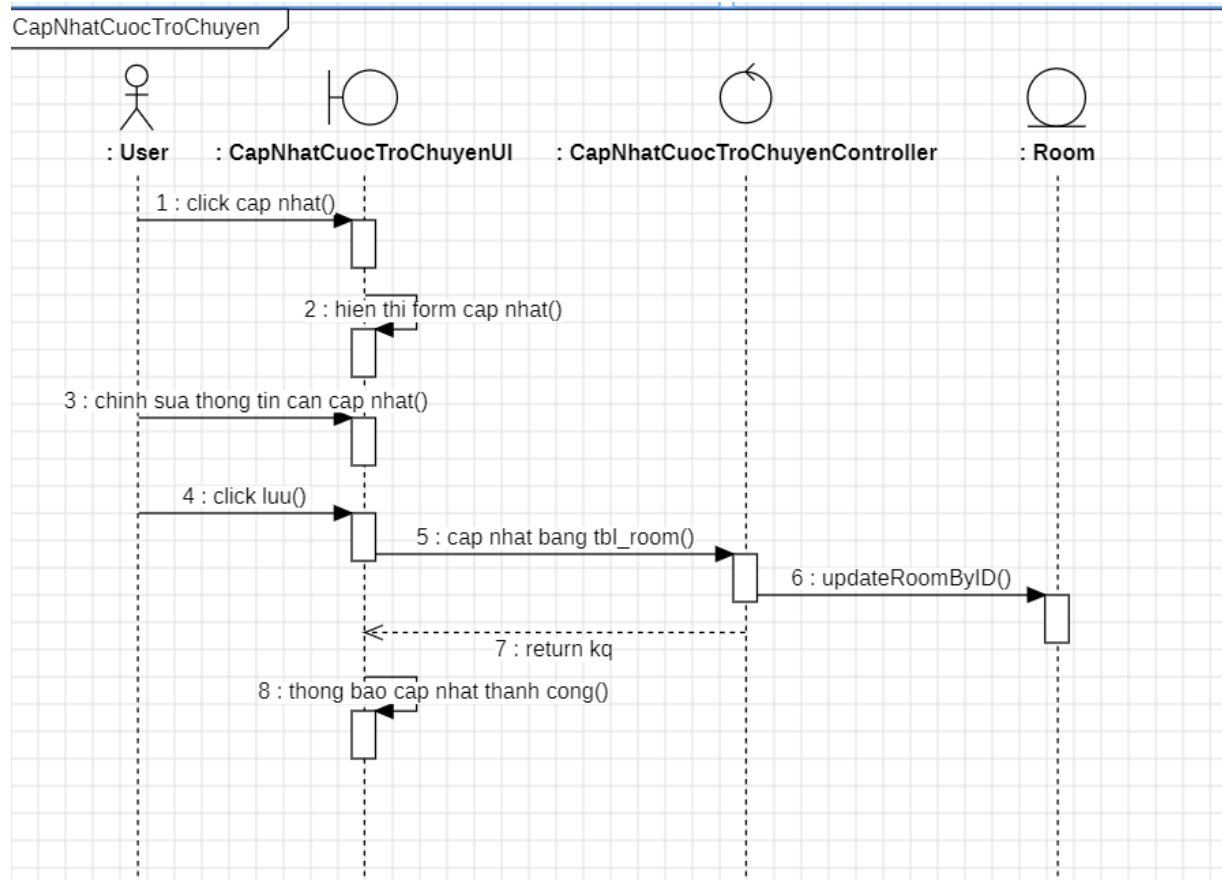
Luồng rẽ nhánh:

Tại bước 2 ở luồng cơ bản, nếu cuộc trò chuyện không tồn tại hoặc người dùng không có quyền truy cập, hệ thống hiển thị thông báo lỗi và use case kết thúc.

Tại bước 5 ở luồng cơ bản, nếu người dùng không nhập thông tin mới hoặc thực hiện cài đặt không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng nhập lại.

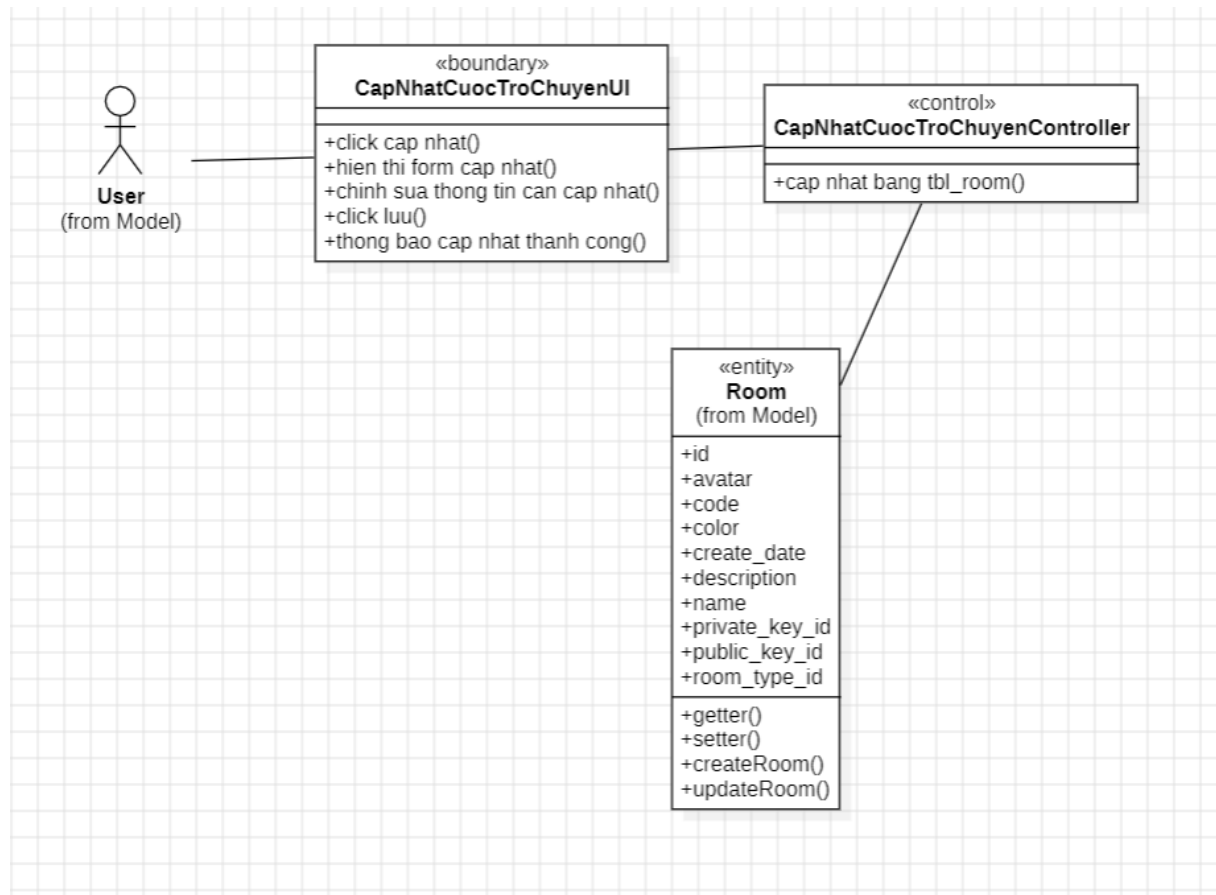
Trong tất cả các bước, nếu hệ thống không thể kết nối được với CSDL, hiển thị ra màn hình thông báo "Hệ thống không thể kết nối được với dữ liệu". Use case kết thúc.

- ⇒ Các yêu cầu đặc biệt: Có thể cập nhật các thông tin như tên cuộc trò chuyện, thành viên, quyền truy cập, hoặc cài đặt khác của cuộc trò chuyện.
- ⇒ Tiên điều kiện: Người dùng đã đăng nhập vào hệ thống và cuộc trò chuyện đã tồn tại trong danh sách cuộc trò chuyện của họ.
- ⇒ Hậu điều kiện: Nếu use case kết thúc thành công, thông tin hoặc cài đặt của cuộc trò chuyện sẽ được cập nhật trong CSDL.
- ⇒ Mở rộng: Có thể thêm các tùy chọn cập nhật khác như thêm/xóa thành viên, thay đổi quyền truy cập, hoặc thay đổi cài đặt bảo mật.
- ⇒ Biểu đồ trình tự:



Hình 3.35. Biểu đồ trình tự usecase cập nhật cuộc trò chuyện

⇒ Biểu đồ lớp:



Hình 3.36. Biểu đồ lớp usecase cập nhật cuộc trò chuyện

3.4. Ứng dụng hệ mã RSA trong ứng dụng Chat App RSA

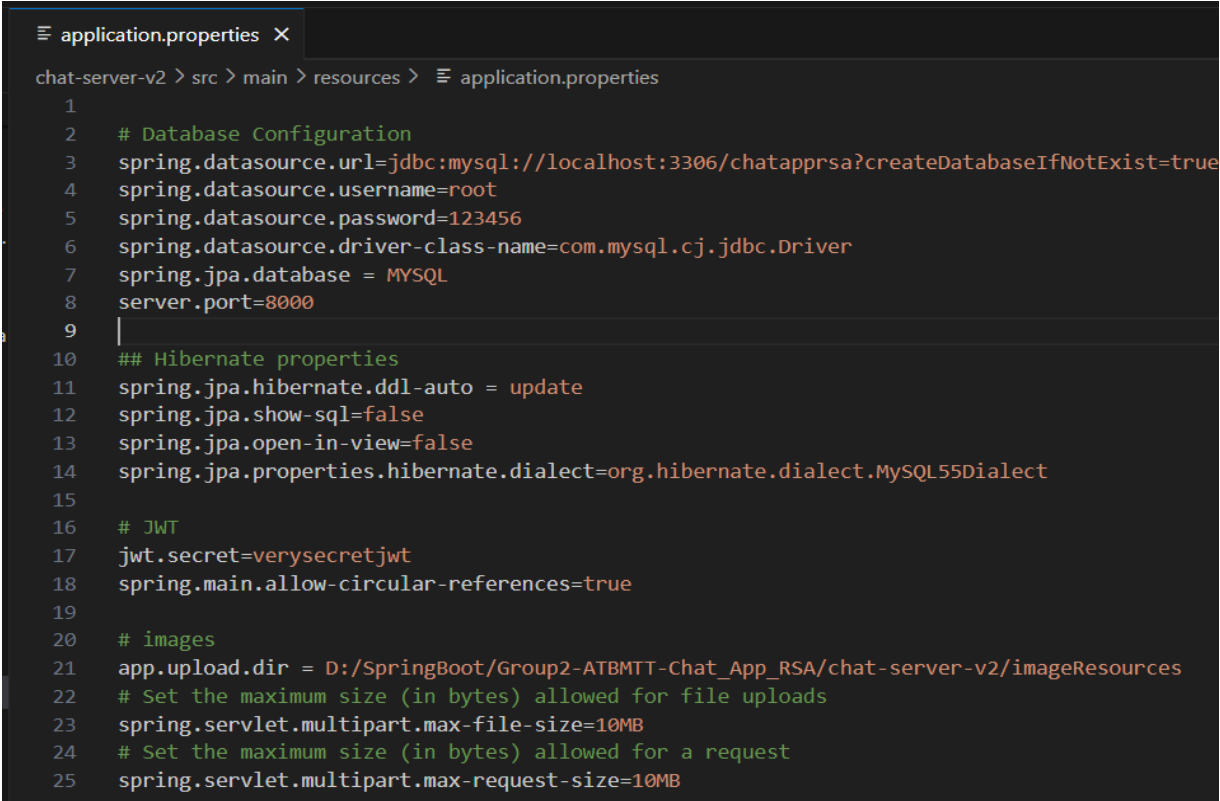
3.4.1. Khởi chạy ứng dụng

Ngoài việc kết hợp những kiến thức cơ sở mà chúng em đã nêu ở trên, chúng em sử dụng GitHub để lưu trữ mã nguồn chương trình, toàn bộ mã nguồn của chương trình được lưu tại liên kết: https://github.com/DiAyTi2004/Group2-ATBMTT-Chat_App_RSA.git

Tải ứng dụng và giải nén, mở ứng dụng bằng phần mềm Visual Studio Code, tại file application.properties:

⇒ Sửa lại tên tài khoản và mật khẩu đã được cài đặt tại máy để kết nối với MySQL.

⇒ Sửa lại đường dẫn để lưu các ảnh của ứng dụng tại folder đó (như ảnh người dùng, ảnh cuộc trò chuyện).



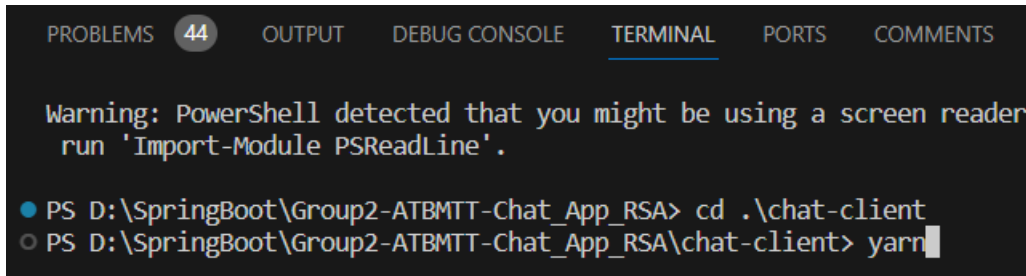
```

1  # Database Configuration
2  spring.datasource.url=jdbc:mysql://localhost:3306/chatapprsa?createDatabaseIfNotExist=true
3  spring.datasource.username=root
4  spring.datasource.password=123456
5  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6  spring.jpa.database = MYSQL
7  server.port=8000
8
9
10 ## Hibernate properties
11 spring.jpa.hibernate.ddl-auto = update
12 spring.jpa.show-sql=false
13 spring.jpa.open-in-view=false
14 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL55Dialect
15
16 # JWT
17 jwt.secret=verysecretjwt
18 spring.main.allow-circular-references=true
19
20 # images
21 app.upload.dir = D:/SpringBoot/Group2-ATBMTT-Chat_App_RSA/chat-server-v2/imageResources
22 # Set the maximum size (in bytes) allowed for file uploads
23 spring.servlet.multipart.max-file-size=10MB
24 # Set the maximum size (in bytes) allowed for a request
25 spring.servlet.multipart.max-request-size=10MB

```

Hình 3.37. Hình ảnh mô tả file *application.properties*

Tạo 1 terminal mới trong Visual Studio Code, trở vào thư mục chat-client, Sau khi đã cài Yarn vào trong máy, gõ lệnh “yarn” để cài các thư viện cần sử dụng cho phần ứng dụng Frontend. Ví dụ:



```

PROBLEMS 44 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Warning: PowerShell detected that you might be using a screen reader
run 'Import-Module PSReadLine'.

PS D:\SpringBoot\Group2-ATBMTT-Chat_App_RSA> cd .\chat-client
PS D:\SpringBoot\Group2-ATBMTT-Chat_App_RSA\chat-client> yarn

```

Hình 3.38. Hình ảnh mô tả chạy lệnh *yarn* trong terminal

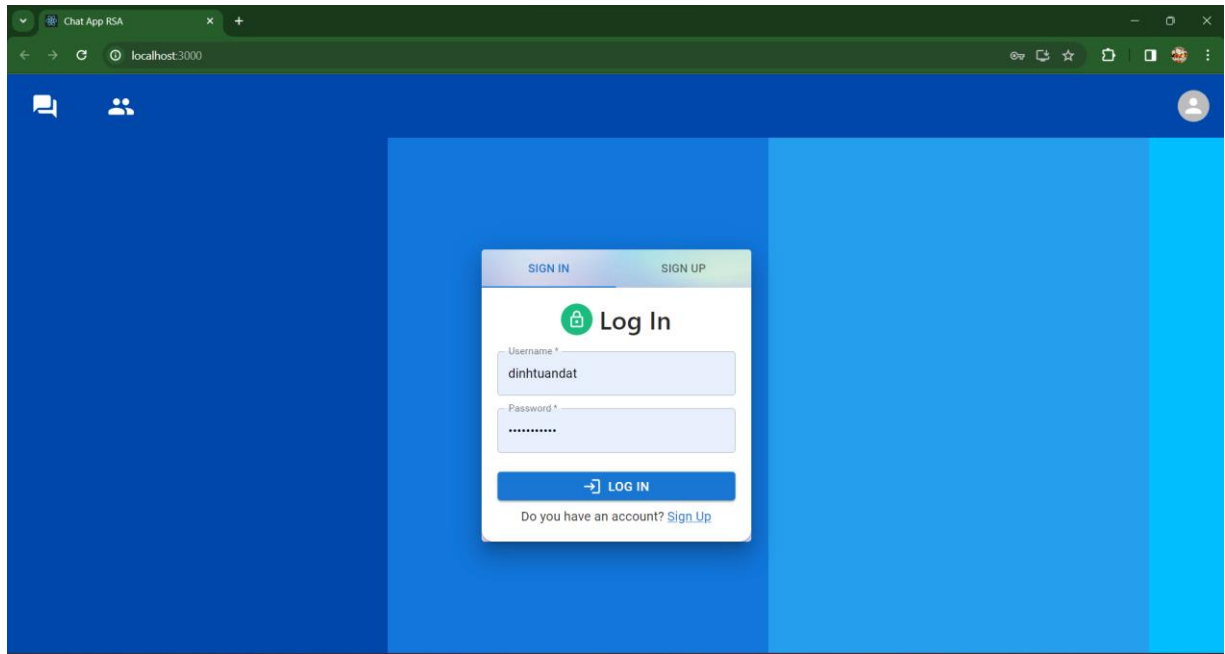
Sau khi cài các thư viện thành công, khởi chạy ứng dụng Frontend React TypeScript bằng lệnh “yarn start” vẫn tại thư mục chat client trong terminal. Ví dụ:

```
Warning: PowerShell detected that you might be using a screen reader and
run 'Import-Module PSReadLine'.

● PS D:\SpringBoot\Group2-ATBMTT-Chat_App_RSA> cd .\chat-client
○ PS D:\SpringBoot\Group2-ATBMTT-Chat_App_RSA\chat-client> yarn start
```

Hình 3.39. Hình ảnh mô tả chạy lệnh yarn start trong terminal

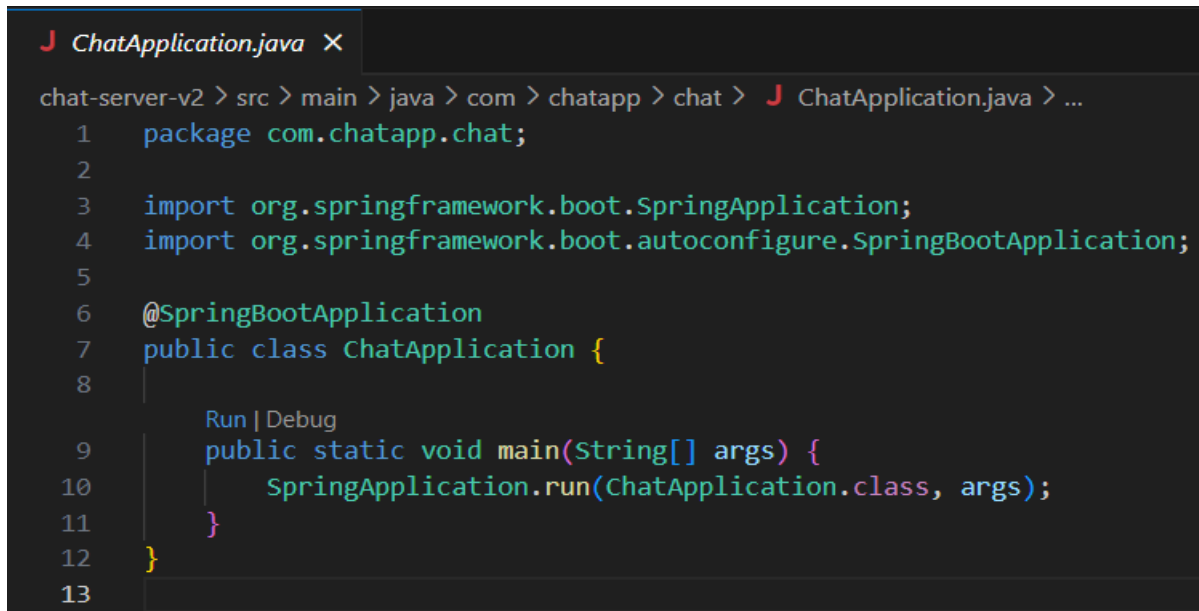
- Chương trình phần Frontend đã được chạy trên localhost:3000



Hình 3.40. Giao diện trang đăng nhập

Để chạy phía Backend của ứng dụng, cài bộ JDK (Java Development Kit) phiên bản 18, sau khi cài thành công, tại file ChatApplication.java, ấn nút “Run” trên hàm main để bắt đầu chạy. Khi chạy phần Backend, các thông số được hiện tại Terminal mới tự động được tạo, khi xuất hiện dòng chữ “Started Application”

- Khi đó toàn bộ phần Frontend và Backend của ứng dụng đã được chạy, người dùng bắt đầu tạo tài khoản tại máy local để trải nghiệm.

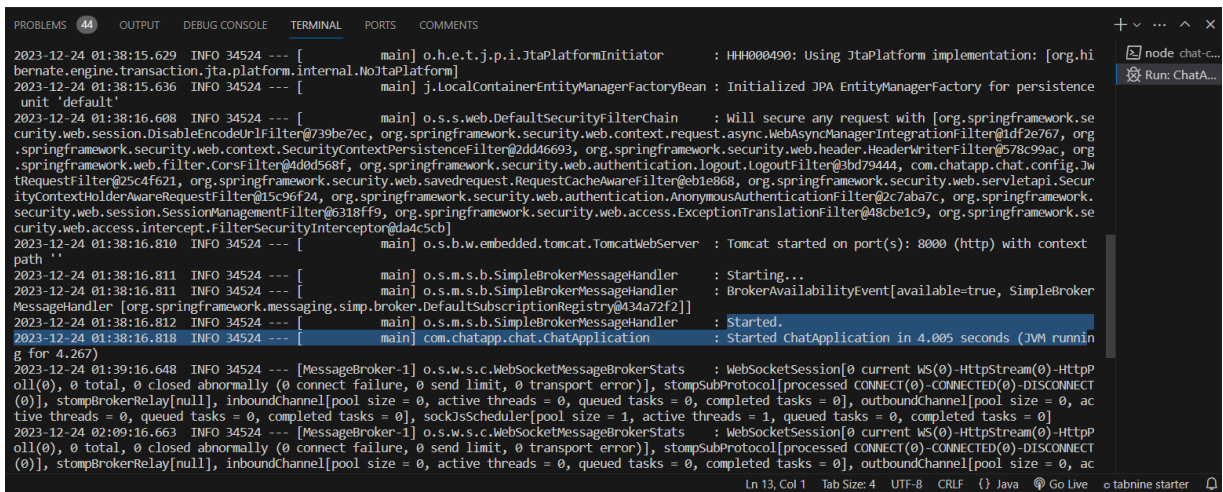


```

1 package com.chatapp.chat;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class ChatApplication {
8
9     Run | Debug
10    public static void main(String[] args) {
11        SpringApplication.run(ChatApplication.class, args);
12    }
13

```

Hình 3.41. Hình ảnh mô tả file ChatApplication.java



```

2023-12-24 01:38:15.629 INFO 34524 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hi
bernat.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-12-24 01:38:15.636 INFO 34524 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
unit 'default'
2023-12-24 01:38:16.608 INFO 34524 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.se
curity.web.session.DisableEncodeUrlFilter@739be7ec, org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@1df2e767, org
.springframework.security.web.context.SecurityContextPersistenceFilter@2dd46693, org.springframework.security.web.header.HeaderWriterFilter@578c99ac, org
.springframework.web.filter.CorsFilter@4d0d568f, org.springframework.security.web.authentication.logout.LogoutFilter@3bd79444, com.chatapp.chat.config.Jw
tRequestFilter@25c4f621, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@eb1e868, org.springframework.security.web.servletapi.Secur
ityContextHolderAwareRequestFilter@15c96f24, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@2c7aba7c, org.springframework.se
curity.web.session.SessionManagementFilter@6318ff9, org.springframework.security.web.access.ExceptionTranslationFilter@48cbe1c9, org.springframework.se
curity.web.access.intercept.FilterSecurityInterceptor@da4c5cb]
2023-12-24 01:38:16.810 INFO 34524 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context
path ''
2023-12-24 01:38:16.811 INFO 34524 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Starting...
2023-12-24 01:38:16.811 INFO 34524 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : BrokerAvailabilityEvent[available=true, SimpleBroker
MessageHandler [org.springframework.messaging.simp.broker.DefaultSubscriptionRegistry@434a72f2]]
2023-12-24 01:38:16.812 INFO 34524 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Started.
2023-12-24 01:38:16.818 INFO 34524 --- [main] com.chatapp.chat.ChatApplication : Started ChatApplication in 4.005 seconds (JVM runnin
g for 4.267)
2023-12-24 01:39:16.648 INFO 34524 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[0 current WS(0)-HttpStream(0)-HttpP
oll(0), 0 total, 0 closed abnormally (0 connect failure, 0 send limit, 0 transport error)], stompSubProtocol[processed CONNECT(0)-CONNECTED(0)-DISCONNECT
(0)], stompBrokerRelay[null], inboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], outboundChannel[pool size = 0, ac
tive threads = 0, queued tasks = 0, completed tasks = 0], sockJsScheduler[pool size = 1, active threads = 1, queued tasks = 0, completed tasks = 0]
2023-12-24 02:09:16.663 INFO 34524 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[0 current WS(0)-HttpStream(0)-HttpP
oll(0), 0 total, 0 closed abnormally (0 connect failure, 0 send limit, 0 transport error)], stompSubProtocol[processed CONNECT(0)-CONNECTED(0)-DISCONNECT
(0)], stompBrokerRelay[null], inboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], outboundChannel[pool size = 0, ac

```

Hình 3.42. Hình ảnh mô tả khi run file ChatApplication.java

3.4.2. Hoạt động của ứng dụng khi đăng nhập

Sau khi đã đăng ký tài khoản thành công, người dùng đăng nhập với tài khoản vừa tạo để trải nghiệm ứng dụng Chat App RSA

Khi đăng nhập thành công, bên Backend trả về một chuỗi JWT (Json Web Token) => Mọi Request (yêu cầu) tiếp theo được gửi từ máy của các người dùng (Client) phải có đính kèm theo chuỗi JWT này trong phần Header của Request để xác thực tài khoản người dùng nào đã thực hiện thao tác này. Nếu token được



Hình 3.45. Hình ảnh mô tả public key khi được gửi sẽ cập nhật vào trường publicKeyId trong tbl_user

Đoạn mã JavaScript được dùng để tạo ra bộ key mới mỗi khi người dùng đăng nhập và các hàm liên quan cần sử dụng được viết trong file RSAService.ts:

```

TS RSAService.ts M X
chat-client > src > components > Auth > TS RSAService.ts > RSAService
110
111 //generate public key and private key once successfully login
112 generate = async () => {
113     try {
114         let p = this.getRandomPrime();
115         let q = this.getRandomPrime();
116
117         let n = p * q;
118         let phi = (p - 1) * (q - 1);
119
120         let e = this.getCoPrime(phi);
121
122         let d = this.modInverse(e, phi);
123
124         // Hiện ra màn hình p,q,n,e,d
125         console.log("p:", p, "\nq:", q, "\nn:", n, "\ne:", e, "\nd:", d);
126         this.publicKey = {
127             n: n,
128             e: e
129         };
130         this.privateKey = {
131             n: n,
132             d: d
133         };
134         return { n, e, d };
135     } catch (error) {
136         console.error("Lỗi: " + error.message);
137     }
138 };
139
140 publicKey = null;
141 privateKey = null;
142 }

```

Hình 3.46. Hình ảnh mô tả đoạn mã tự động sinh khóa khi đăng nhập thành công

```

6  ✓ static mod = (a, b, m) => {
7      let result = 1;
8      a = a % m;
9
10     while (b > 0) {
11         if (b % 2 === 1) {
12             result = (result * a) % m;
13         }
14         a = (a * a) % m;
15         b = Math.floor(b / 2);
16     }
17
18     return result;
19 }
20
21 // tính nghịch đảo a mod m
22 ✓ modInverse = (a, m) => {
23     let m0 = m;
24     let x0 = 0;
25     let x1 = 1;
26
27     if (m === 1) return 0;
28
29     while (a > 1) {
30         let q = Math.floor(a / m);
31         let t = m;
32         m = a % m;
33         a = t;
34         t = x0;
35         x0 = x1 - q * x0;
36         x1 = t;
37     }
38     if (x1 < 0) x1 += m0;
39     return x1;
40 };

```

Hình 3.47. Hình ảnh mô tả đoạn mã tính nghịch đảo của $a \bmod m$

```

42 // kiểm tra số nguyên tố Miller-Rabin
43 isPrime = (n, k) => {
44     if (n <= 1 || n === 4) {
45         return false;
46     }
47     if (n <= 3) {
48         return true;
49     }
50     let d = n - 1;
51     while (d % 2 === 0) {
52         d = Math.floor(d / 2);
53     }
54     for (let i = 0; i < k; i++) {
55         let a = 2 + Math.floor(Math.random() * (n - 4));
56         let x = RSAService.mod(a, d, n);
57         if (x === 1 || x === n - 1) {
58             continue;
59         }
60         let isProbablePrime = false;
61         while (d !== n - 1) {
62             x = (x * x) % n;
63             d *= 2;
64             if (x === 1) {
65                 return false;
66             }
67             if (x === n - 1) {
68                 isProbablePrime = true;
69                 break;
70             }
71         }
72         if (!isProbablePrime) {
73             return false;
74         }
75     }
76     return true;
77 }

```

Hình 3.48. Hình ảnh mô tả đoạn mã kiểm tra số nguyên tố Miller-Rabin

```

79 //random số nguyên tố bất kì
80 getRandomPrime = () => {
81   let prime, k = 10000007;
82   do {
83     prime = Math.floor(Math.random() * 10000) + 2;
84   } while (!this.isPrime(prime, k));
85   return prime;
86 };
87
88 //tìm ước chung lớn nhất
89 gcd = (a, b) => {
90   return b === 0 ? a : this.gcd(b, a % b);
91 };
92
93 // tìm số nguyên tố sắp xỉ với num
94 getCoPrime = (num) => {
95   let coPrime, k = 10000007;
96   do {
97     coPrime = Math.floor(Math.random() * (num - 2)) + 2;
98   } while (
99     (this.gcd(num, coPrime) !== 1 || this.isPrime(coPrime, k) === false) &&
100     num !== coPrime
101   );
102   return coPrime;
103 };

```

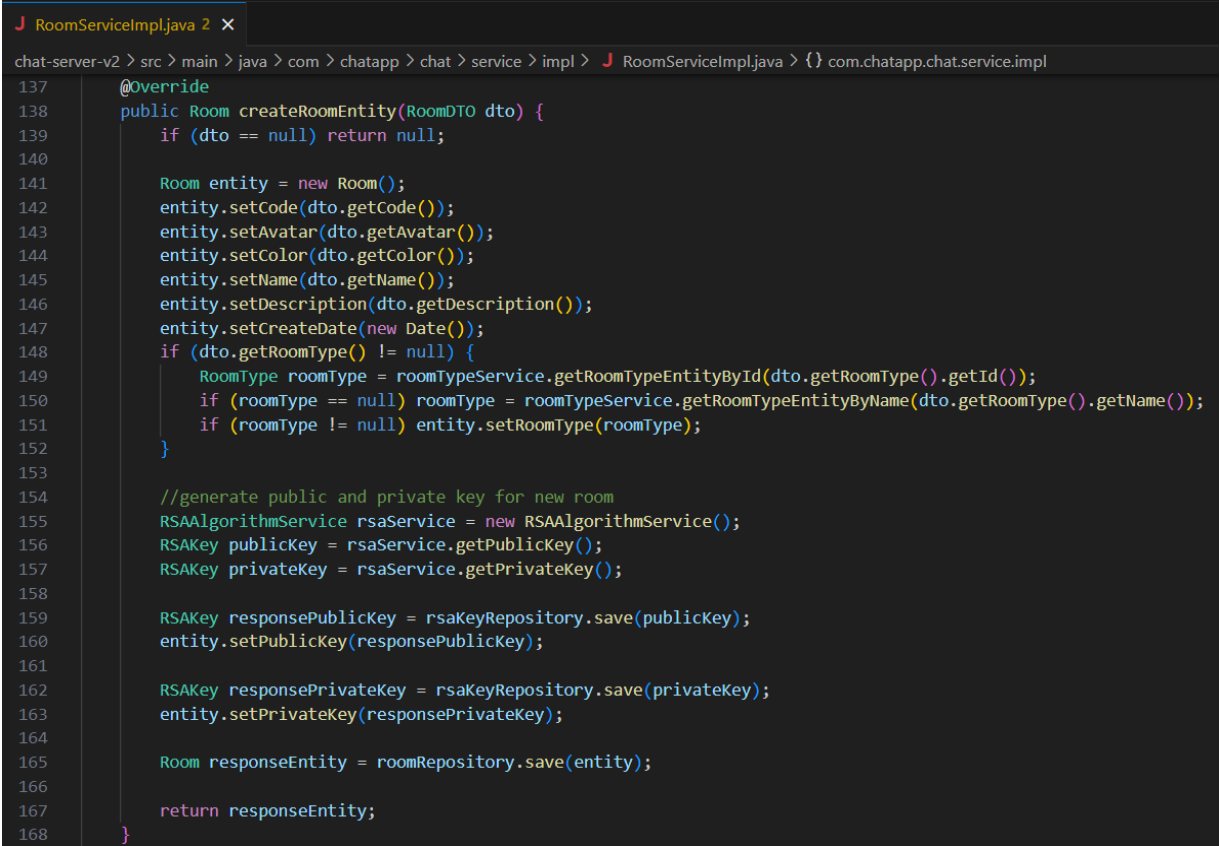
Hình 3.49. Hình ảnh mô tả đoạn mã tìm số nguyên tố sắp xỉ với num

3.4.2. Hoạt động của ứng dụng khi chấp nhận lời mời kết bạn

Tại cùng 1 máy, tạo một tài khoản khác và đăng nhập, gửi lời mời kết bạn cho tài khoản ban đầu, khi đó tài khoản ban đầu có thể chấp nhận lời mời kết bạn. Khi người dùng nút “Accept friend request” (chấp nhận lời mời kết bạn), phía Backend xử lí như sau:

- + Cập nhật status trong bảng tbl_friend từ false thành true (với false: chưa là bạn bè; true: đã là bạn bè).
- + Tạo ra 1 room chat (lưu trong bảng tbl_room), đồng thời tạo ra cặp public key và private key cho room chat này.

- + Tạo ra 2 bản ghi trong bảng tbl_user_room với roomId là id của room vừa được tạo, 2 userId tương ứng là của người gửi lời mời kết bạn và người chấp nhận lời mời kết bạn.
- Mỗi khi người dùng gửi tin nhắn, tin nhắn đó phải được gửi tới 1 room chat cụ thể và tin nhắn phải được mã hóa theo public của room đó, mỗi khi room chat đó nhận được tin nhắn từ người dùng nào đó, bên Backend sử dụng private key của room chat để giải mã tin nhắn.



```

J RoomServiceImpl.java 2 x
chat-server-v2 > src > main > java > com > chatapp > chat > service > impl > J RoomServiceImpl.java > {} com.chatapp.chat.service.impl
137 @Override
138 public Room createRoomEntity(RoomDTO dto) {
139     if (dto == null) return null;
140
141     Room entity = new Room();
142     entity.setCode(dto.getCode());
143     entity.setAvatar(dto.getAvatar());
144     entity.setColor(dto.getColor());
145     entity.setName(dto.getName());
146     entity.setDescription(dto.getDescription());
147     entity.setCreateDate(new Date());
148     if (dto.getRoomType() != null) {
149         RoomType roomType = roomTypeService.getRoomTypeEntityById(dto.getRoomType().getId());
150         if (roomType == null) roomType = roomTypeService.getRoomTypeEntityByName(dto.getRoomType().getName());
151         if (roomType != null) entity.setRoomType(roomType);
152     }
153
154     //generate public and private key for new room
155     RSAAlgorithmService rsaService = new RSAAlgorithmService();
156     RSAKey publicKey = rsaService.getPublicKey();
157     RSAKey privateKey = rsaService.getPrivateKey();
158
159     RSAKey responsePublicKey = rsaKeyRepository.save(publicKey);
160     entity.setPublicKey(responsePublicKey);
161
162     RSAKey responsePrivateKey = rsaKeyRepository.save(privateKey);
163     entity.setPrivateKey(responsePrivateKey);
164
165     Room responseEntity = roomRepository.save(entity);
166
167     return responseEntity;
168 }

```

Hình 3.50. Hình ảnh mô tả đoạn mã tạo một phòng chat mới


```

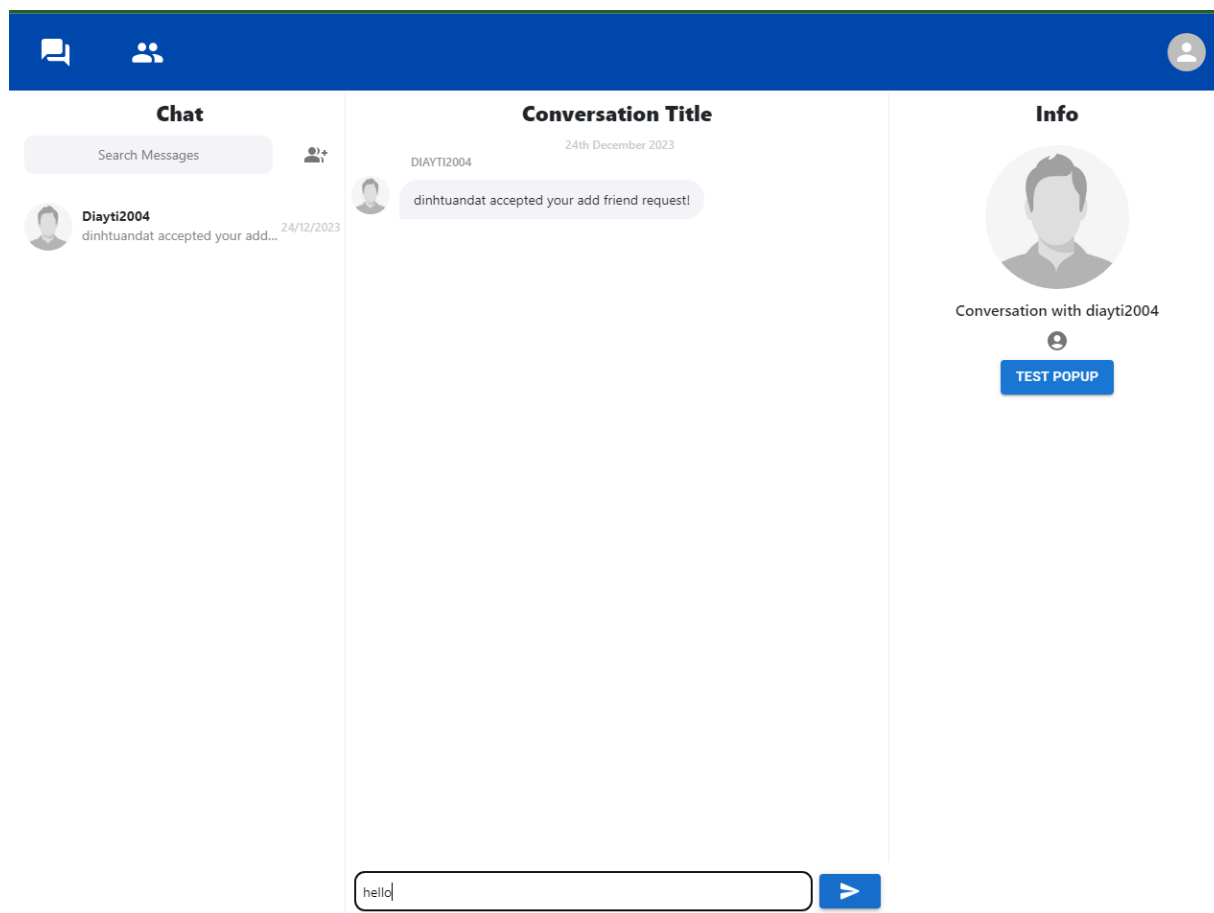
J RSAAlgorithmService.java X
chat-server-v2 > src > main > java > com > chatapp > chat > service > J RSAAlgorithmService.java > R
8 public class RSAAlgorithmService {
9     private BigInteger n;
10    private BigInteger e;
11    private BigInteger d;
12
13    public RSAAlgorithmService(){
14        // initialize key pair n, e, d
15        int bits = 8;
16        SecureRandom r = new SecureRandom();//create BigInteger r random
17        BigInteger p = new BigInteger(bits, certainty:100, r);
18        BigInteger q = new BigInteger(bits, certainty:100, r);
19        n = p.multiply(q);
20        BigInteger m = (p.subtract(BigInteger.ONE)).multiply(q
21            .subtract(BigInteger.ONE));
22        boolean found = false;
23        do {
24            e = new BigInteger(bits, certainty:50, r);
25            if (m.gcd(e).equals(BigInteger.ONE) && e.compareTo(m) < 0) {
26                found = true;
27            }
28        } while (!found);
29        d = e.modInverse(m);
30    }
31
32    public RSAKey getPublicKey(){
33        RSAKey publicKey = new RSAKey();
34        publicKey.setN(n);
35        publicKey.setE(e);
36        return publicKey;
37    }
38
39    public RSAKey getPrivateKey(){
40        RSAKey privateKey = new RSAKey();
41        privateKey.setN(n);
42        privateKey.setD(d);
43        return privateKey;
44    }
45 }

```

Hình 3.51. Hình ảnh mô tả đoạn mã khởi tạo cặp khóa

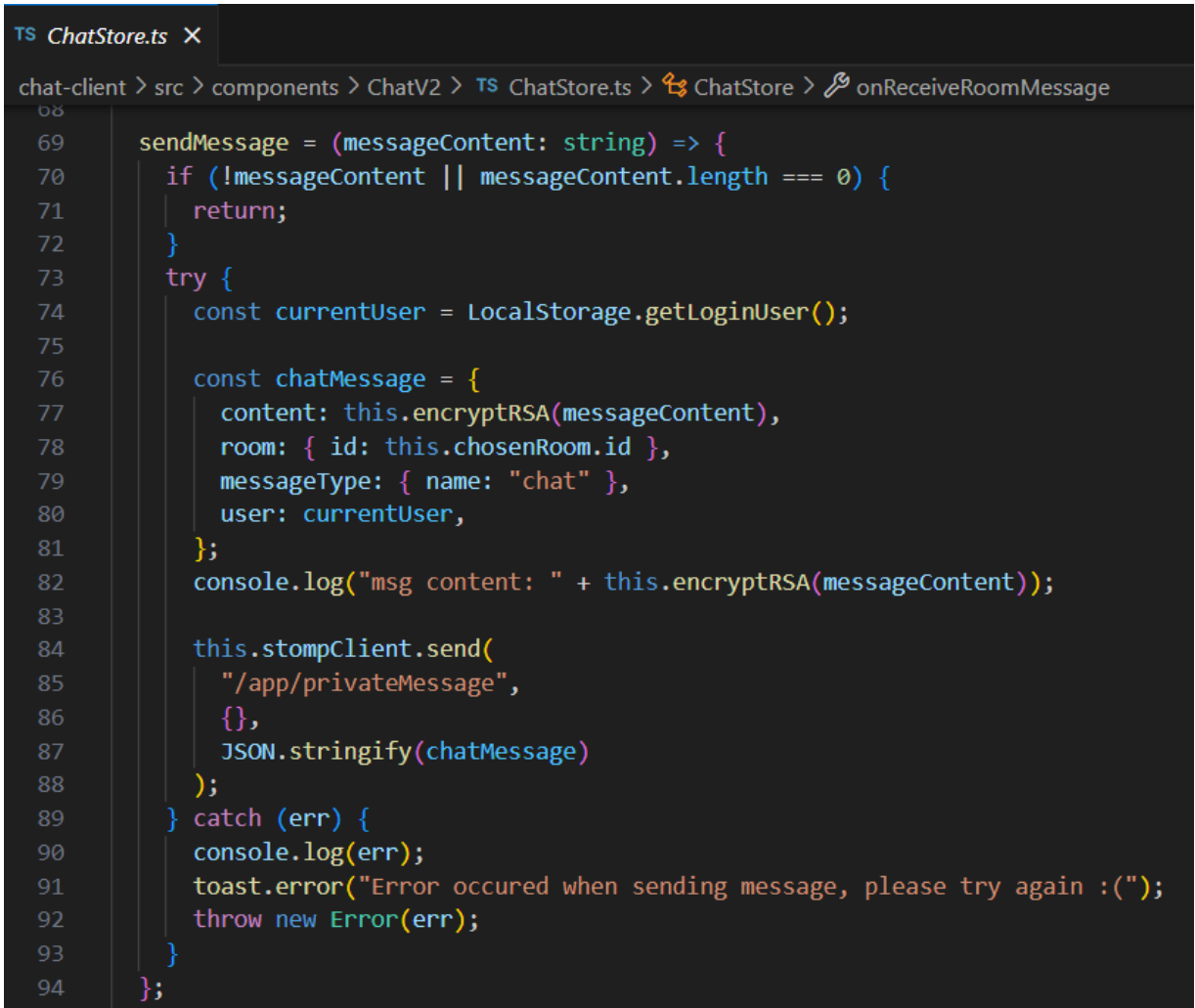
3.4.3. Hoạt động của ứng dụng khi gửi và nhận tin nhắn

Sau khi chọn 1 phòng trò chuyện (room chat/conversation), người dùng có thể soạn tin nhắn (bằng tiếng Anh) và gửi đến phòng chat đó.



Hình 3.52. Giao diện trang chat

Khi người dùng ấn nút Enter trên bàn phím hoặc nút có biểu tượng Send trên màn hình, tin nhắn trước khi gửi đi sẽ được mã hóa theo public key của room gửi đến.



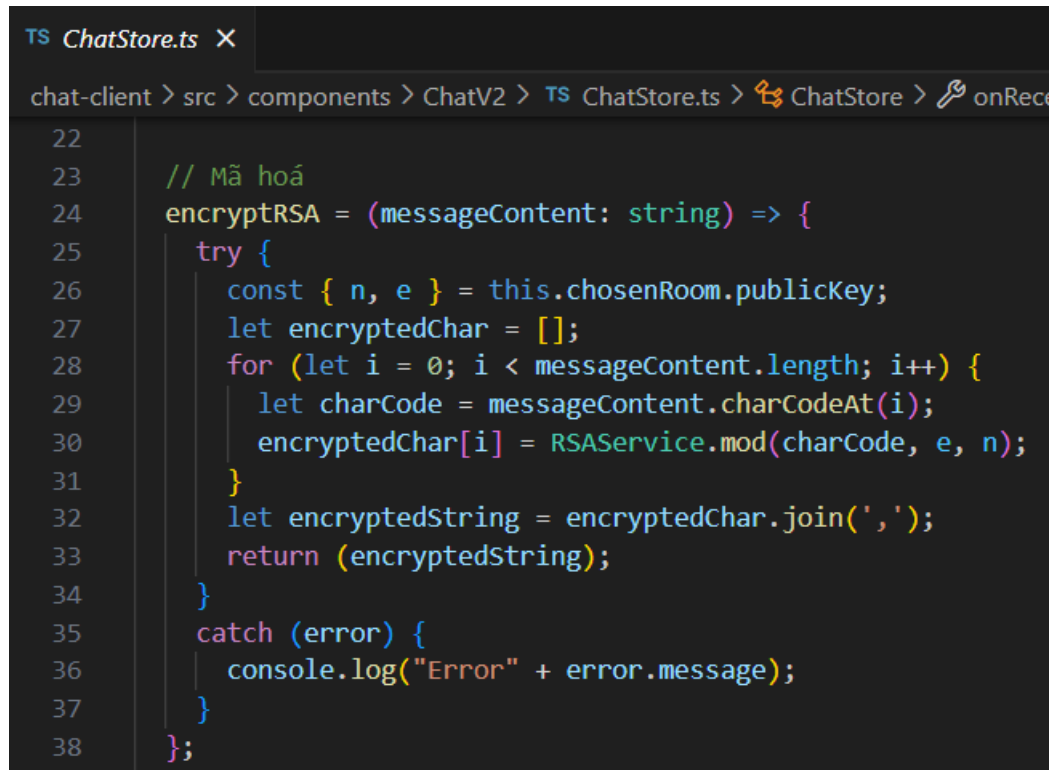
```

68
69  sendMessage = (messageContent: string) => {
70    if (!messageContent || messageContent.length === 0) {
71      return;
72    }
73    try {
74      const currentUser = LocalStorage.getLoginUser();
75
76      const chatMessage = {
77        content: this.encryptRSA(messageContent),
78        room: { id: this.chosenRoom.id },
79        messageType: { name: "chat" },
80        user: currentUser,
81      };
82      console.log("msg content: " + this.encryptRSA(messageContent));
83
84      this.stompClient.send(
85        "/app/privateMessage",
86        {},
87        JSON.stringify(chatMessage)
88      );
89    } catch (err) {
90      console.log(err);
91      toast.error("Error occured when sending message, please try again :(");
92      throw new Error(err);
93    }
94  };

```

Hình 3.53. Hình ảnh mô tả đoạn mã người dùng gửi tin nhắn đến server sau khi mã hóa tin nhắn bằng rsa

Dữ liệu được gửi đi sẽ là 1 object chatMessage, có content là nội dung của tin nhắn được mã hóa theo public key của room chat được nhận, id của room chat, messageType và user gửi tin nhắn là không bắt buộc, tuy nhiên ở đây để làm rõ dữ liệu gửi đi, 2 trường này vẫn được hiển thị



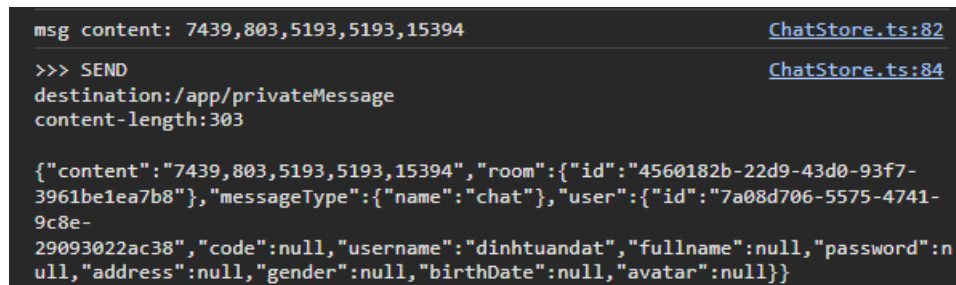
```

22
23 // Mã hoá
24 encryptRSA = (messageContent: string) => {
25     try {
26         const { n, e } = this.chosenRoom.publicKey;
27         let encryptedChar = [];
28         for (let i = 0; i < messageContent.length; i++) {
29             let charCode = messageContent.charCodeAt(i);
30             encryptedChar[i] = RSAService.mod(charCode, e, n);
31         }
32         let encryptedString = encryptedChar.join(',');
33         return (encryptedString);
34     }
35     catch (error) {
36         console.log("Error" + error.message);
37     }
38 };

```

Hình 3.54. Hình ảnh mô tả hàm mã hóa phía client

Khi đó dữ liệu được gửi đi có dạng:



```

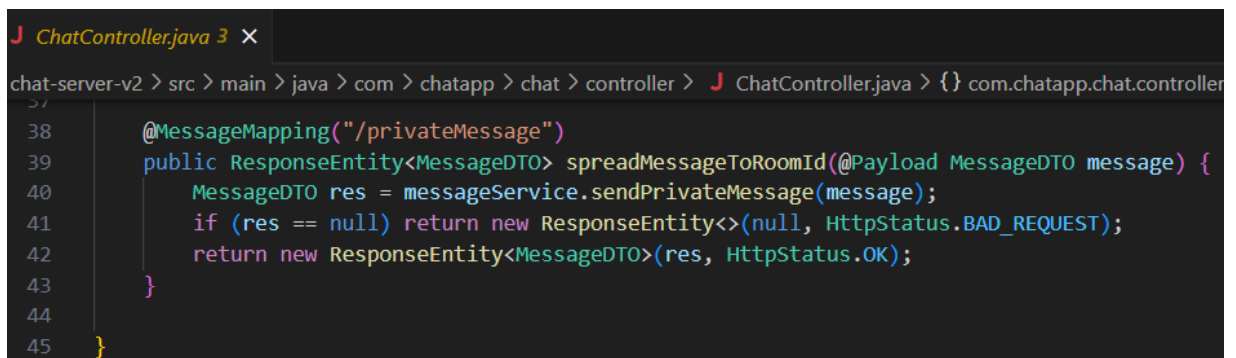
msg content: 7439,803,5193,5193,15394 ChatStore.ts:82
>>> SEND ChatStore.ts:84
destination:/app/privateMessage
content-length:303

{"content":"7439,803,5193,5193,15394","room":{"id":"4560182b-22d9-43d0-93f7-3961be1ea7b8"},"messageType":{"name":"chat"},"user":{"id":"7a08d706-5575-4741-9c8e-29093022ac38","code":null,"username":"dinhtuandat","fullname":null,"password":null,"address":null,"gender":null,"birthDate":null,"avatar":null}}

```

Hình 3.55. Hình ảnh mô tả dữ liệu được gửi đi khi mã hóa

Tại Backend, dữ liệu nhận được tại Controller:



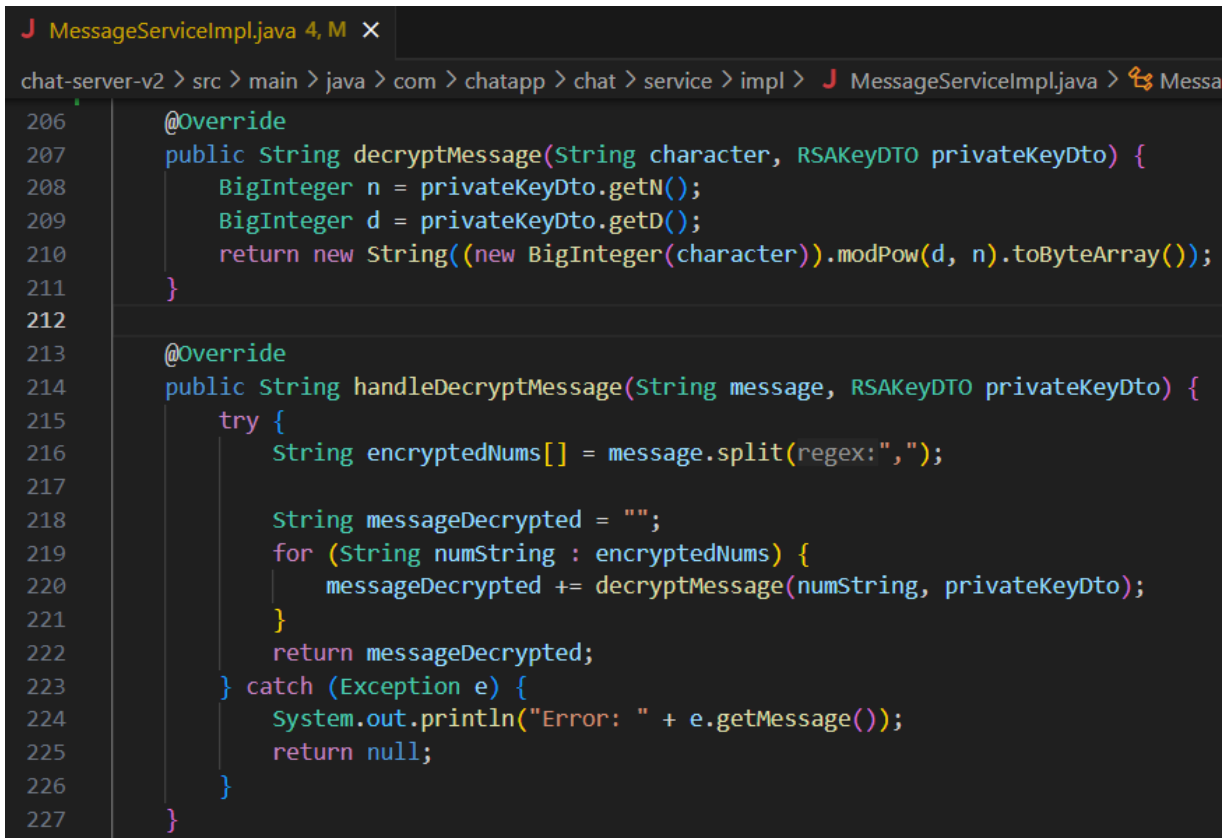
```

J ChatController.java 3 X
chat-server-v2 > src > main > java > com > chatapp > chat > controller > J ChatController.java > {} com.chatapp.chat.controller
38 @PostMapping("/privateMessage")
39 public ResponseEntity<MessageDTO> spreadMessageToRoomId(@Payload MessageDTO message) {
40     MessageDTO res = messageService.sendPrivateMessage(message);
41     if (res == null) return new ResponseEntity<>(null, HttpStatus.BAD_REQUEST);
42     return new ResponseEntity<MessageDTO>(res, HttpStatus.OK);
43 }
44
45 }

```

Hình 3.56. Hình ảnh mô tả dữ liệu nhận được phía Backend

Hệ thống truy xuất private key của room chat đó trong cơ sở dữ liệu, thực hiện giải mã tin nhắn đã nhận được từ người dùng vừa gửi:



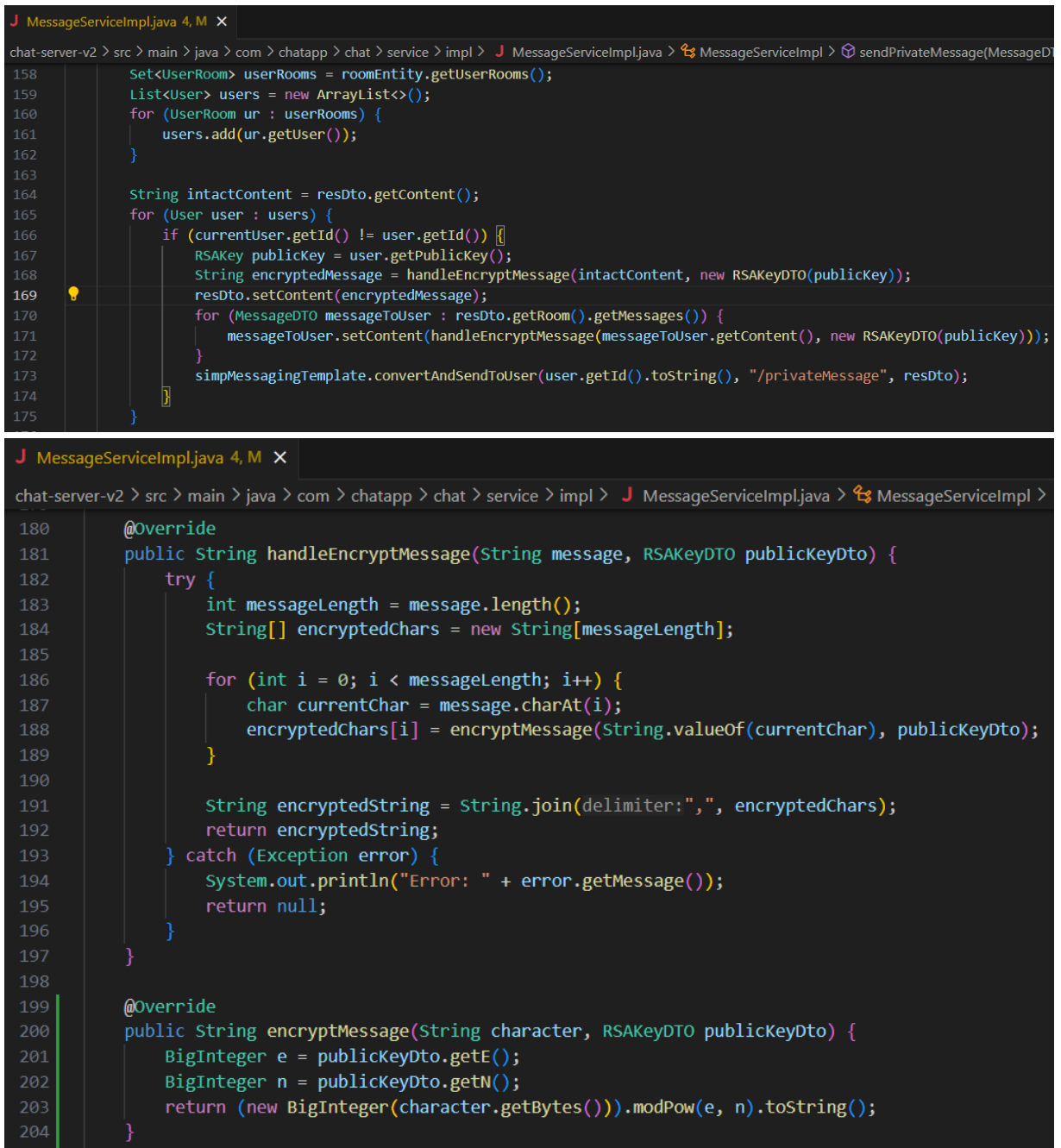
```

206     @Override
207     public String decryptMessage(String character, RSAKeyDto privateKeyDto) {
208         BigInteger n = privateKeyDto.getN();
209         BigInteger d = privateKeyDto.getD();
210         return new String((new BigInteger(character)).modPow(d, n).toByteArray());
211     }
212
213     @Override
214     public String handleDecryptMessage(String message, RSAKeyDto privateKeyDto) {
215         try {
216             String encryptedNums[] = message.split(regex=",");
217
218             String messageDecrypted = "";
219             for (String numString : encryptedNums) {
220                 messageDecrypted += decryptMessage(numString, privateKeyDto);
221             }
222             return messageDecrypted;
223         } catch (Exception e) {
224             System.out.println("Error: " + e.getMessage());
225             return null;
226         }
227     }

```

Hình 3.57. Hình ảnh mô tả đoạn mã giải mã tin nhắn nhận được từ client

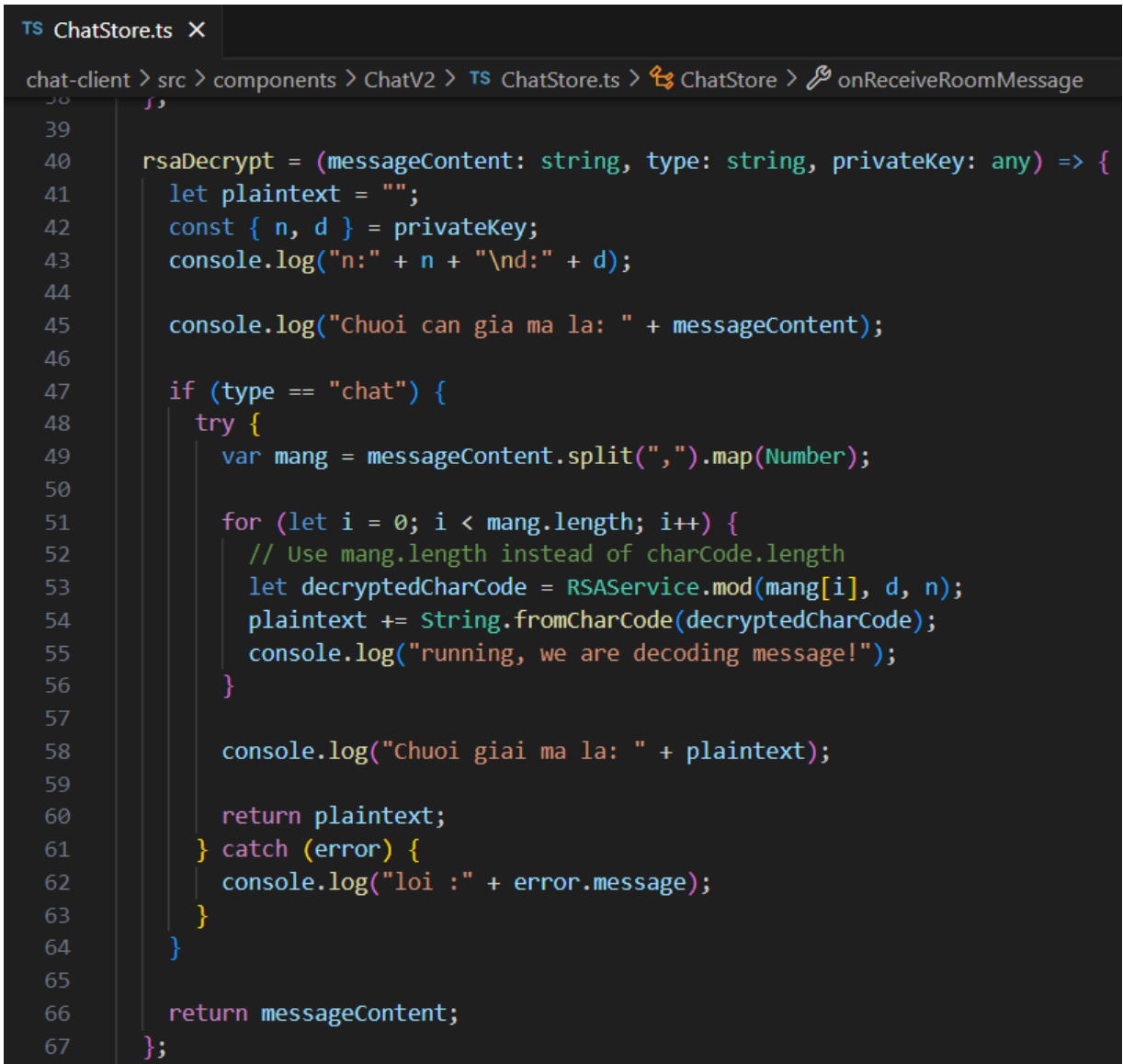
Sau khi giải mã xong, tin nhắn đó được lưu lại cơ sở dữ liệu, tiếp đến, hệ thống lấy tất cả người dùng cũng nằm trong cuộc trò chuyện/room chat đó, mã hóa tin nhắn theo từng public key của từng người dùng và gửi lại tin nhắn đã mã hóa đó đến từng người dùng tương ứng, dưới đây là đoạn code minh họa điều này để Backend xử lí:



Hình 3.58. Hình ảnh mô tả việc mã hóa tin nhắn theo public key của từng người dùng trong room chat

Khi tin nhắn đã được gửi lại từ phía Server (Backend), các Client (người dùng) tương ứng thuộc cuộc trò chuyện/room chat đó sẽ nhận được tin nhắn, việc cần làm hiện tại ở máy người dùng là lấy private key đã được tạo ra mỗi khi

đăng nhập để giải mã tin nhắn được nhận đó và hiển thị trên giao diện.



```

38  },
39
40  rsaDecrypt = (messageContent: string, type: string, privateKey: any) => {
41    let plaintext = "";
42    const { n, d } = privateKey;
43    console.log("n:" + n + "\nd:" + d);
44
45    console.log("Chuoi can gia ma la: " + messageContent);
46
47    if (type == "chat") {
48      try {
49        var mang = messageContent.split("").map(Number);
50
51        for (let i = 0; i < mang.length; i++) {
52          // Use mang.length instead of charCode.length
53          let decryptedCharCode = RSAService.mod(mang[i], d, n);
54          plaintext += String.fromCharCode(decryptedCharCode);
55          console.log("running, we are decoding message!");
56        }
57
58        console.log("Chuoi giai ma la: " + plaintext);
59
60        return plaintext;
61      } catch (error) {
62        console.log("loi :" + error.message);
63      }
64    }
65
66    return messageContent;
67  };

```

Hình 3.59. Hình ảnh mô tả đoạn mã giải mã tin nhắn

CHƯƠNG 4: KẾT LUẬN

Qua thời gian thực hiện, dưới sự hướng dẫn tận tình của cô Lê Thị Anh, nhóm em đã cố gắng hoàn thành đề tài theo đúng yêu cầu và thời gian quy định. Trong suốt quá trình nghiên cứu và thực hiện đề tài, nhóm chúng em đã lĩnh hội được những kiến thức, nội dung và kỹ năng sau đây:

Nội dung đã làm trong bài tập lớn

1. Ứng dụng chat client server
2. Restful Api
3. React typescript frontend
4. Java spring boot backend
5. Websocket
6. JSONWebtoken
7. Kiến thức về phương pháp mã hóa, giải mã RSA

Bài học kinh nghiệm

- Các kỹ năng cần phải có như:
 - + Kỹ năng làm việc nhóm
 - + Kỹ năng tóm tắt, phân tích và giải quyết vấn đề
 - + Kỹ năng nghiên cứu, tìm tòi và học hỏi
- Kiến thức bắt buộc:
 - + Tiếng anh cơ bản
 - + Các môn toán học và lập trình
- Kiến thức chuyên sâu:
 - + Có kiến thức nền tảng về máy tính (phần cứng, phần mềm) và hệ thống mạng
 - + Hiểu và nắm bắt về luật an toàn thông tin
 - + Vận dụng tốt ngôn ngữ lập trình (như Java, Javascript...)
 - + Hiểu và vận hành quy trình phát triển phần mềm
 - + Phân tích lỗ hổng, virus, mã độc, phân tích đánh giá hệ thống

- + Có chuyên môn về mã hóa thông tin, an toàn cơ sở dữ liệu
- + Hiểu về cơ bản của mã hóa RSA, bao gồm cặp khóa công khai và riêng tư.
- + Áp dụng RSA cho bảo mật thông tin quan trọng và tạo, mã hóa, giải mã thông tin.
- + Xây dựng ứng dụng trò chuyện client-server sử dụng RESTful API, React TypeScript, Java Spring Boot, Websocket, JWT, và mã hóa RSA để đảm bảo an toàn và tính toàn vẹn thông tin.
- Đối với mỗi cá nhân:
 - + Có ý thức nâng cao hiểu biết, nhận thức bản thân về an toàn thông tin. Tự trau dồi kinh nghiệm ứng phó sự cố bảo mật cũng như vận hành các quy trình bảo mật mới.
 - + Thường xuyên cập nhật phần mềm, hệ điều hành máy tính cá nhân lên phiên bản mới nhất. Không sử dụng phần mềm crack.
 - + Đề cao cảnh giác khi duyệt email, kiểm tra kỹ tên người gửi để phòng tránh lừa đảo. Tuyệt đối không tải các file đính kèm hoặc nhấp vào đường link không rõ nguồn gốc.

Tóm lại

Mã hóa tin nhắn bằng thuật toán RSA đóng vai trò quan trọng trong việc đảm bảo tính bảo mật và an toàn của thông tin truyền tải. Bằng cách sử dụng cặp khóa công khai và riêng tư, người gửi có thể mã hóa tin nhắn trước khi gửi, chỉ cho phép người nhận với khóa riêng tư tương ứng mới có thể giải mã. Điều này ngăn chặn người không được phép truy cập thông tin và đảm bảo tính toàn vẹn của dữ liệu. Tuy nhiên, thuật toán RSA cũng đối mặt với thách thức về khả năng quản lý khóa an toàn và tính phức tạp tính toán, đặc biệt là với khóa có kích thước lớn. Quản lý cặp khóa là quan trọng, đặc biệt là để bảo vệ khóa riêng tư và tránh các rủi ro bảo mật.

Trên đây nhóm 2 đã trình bày toàn bộ nội dung về đề tài này. Do điều kiện thời gian có hạn cùng với khả năng còn hạn chế nên không tránh khỏi thiếu sót. Vậy mong được thầy cô chỉ bảo để quyền báo cáo được hoàn thiện hơn.

Em xin chân thành cảm ơn cô Lê Thị Anh, cô giáo hướng dẫn trực tiếp, và các thầy cô khác trong bộ môn tận tình giúp đỡ và tạo điều kiện để nhóm em hoàn thành bài báo cáo này.

TÀI LIỆU THAM KHẢO

- [1] Phan Đình Diệu (2002). Lý thuyết mật mã và an toàn thông tin. NXB Đại học quốc gia Hà Nội.
- [2] Secure Communicating Systems: Design, Analysis, and Implementation, Tác giả Michael Huth, Michael R. A. Huth
- [3] [https://vi.wikipedia.org/wiki/RSA_\(m%C3%A3_h%C3%B3a\)](https://vi.wikipedia.org/wiki/RSA_(m%C3%A3_h%C3%B3a))
- [4] <https://www.educba.com/rsa-algorithm/>
- [5] <https://www.geeksforgeeks.org/rsa-and-digital-signatures/>
- [6] <https://jwt.io>
- [7] <https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/>
- [8] <https://vi.wikipedia.org/wiki/WebSocket>
- [9] <https://restfulapi.net/>