TRƯỜNG ĐẠI HỌC BÀ RỊA – VŨNG TÀU
KHOA KỸ THUẬT - CÔNG NGHỆ

BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG ỨNG DỤNG MẠNG XÃ HỘI TRÊN NỀN TẢNG DI ĐỘNG

Trình độ đào tạo :Đại học chính quy
Ngành :Công nghệ thông tin
Chuyên ngành :Lập trình Ứng dụng di động & Game

Giảng viên hướng dẫn :TS. Phan Ngọc Hoàng
Sinh viên thực hiện :Nguyễn Tất Tiến
Mã số sinh viên :19033948
Lớp :DH19LT

Vũng Tàu, ngày 08 tháng 5 năm 2023

LỜI NÓI ĐẦU

Trong quá trình gần 4 năm học tập tại Trường Đại học Bà Rịa – Vũng Tàu, trải qua nhiều môn học, bản thân tôi đã được gặp gỡ và nhận được nhiều sự giúp đỡ quý giá từ các thầy cô. Đó không chỉ là những kiến thức chuyên môn, mà còn những kiến thức bên lề, những kỹ năng mà bất kì sinh viên nào cũng cần trong hành trang vào đời của mình.

Tôi xin gửi lời tri ân chân thành nhất tới quý thầy cô Trường Đại học Bà Rịa – Vũng Tàu và đặc biệt là quý thầy cô Khoa Công nghệ thông tin nói riêng vì đã là một phần trong hành trình tìm kiếm tri thức của tôi.

Qua đó, tôi cũng xin gửi lời cảm ơn sâu sắc đến Thầy, Tiến Sĩ Phan Ngọc Hoàng, là người hướng dẫn tôi trong quá trình xây dựng sản phẩm tốt nghiệp này. Cảm ơn thầy vì đã dành nhiều thời gian tận tình chỉ bảo, định hướng cho bản thân tôi có thể hoàn thành trọn vẹn luận văn tốt nghiệp.

Xin kính chúc thầy cô nhiều sức khỏe, giữ mãi ngọn lửa đam mê để dẫn dắt các thế hệ sinh viên tiếp theo nối bước thành công. Chúc Trường Đại học Bà Rịa – Vũng Tàu sẽ là điểm đến hàng đầu của các thế hệ trẻ sau này, là một môi trường để dẫn dắt các bạn trở thành những công dân hàng đầu trong tương lai.

Với hạn chế về thời gian và trong kinh nghiệm thực tế, sản phẩm này vẫn còn đó những thiếu sót khó tránh khỏi. Kính mong nhận được những góp ý từ các quý thầy cô để tôi có cơ hội hoàn thiện tốt hơn cho sản phẩm cũng như được tích lũy thêm kinh nghiệm sau này.

Xin chân thành cảm ơn!

Thành phố Vũng Tàu, ngày 01 tháng 12 năm 2021

Sinh viên thực hiện

Nguyễn Tất Tiến

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Vũng Tàu, ngày … tháng … năm 20…

Người hướng dẫn

TS. Phan Ngọc Hoàng

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Menu

DANH MỤC CÁC TỪ VIẾT TẮT/TỪ TIẾNG ANH

STT	Các kí hiệu/Từ	Đầy đủ	Ý nghĩa
1	CSDL	Cơ sở dữ liệu	Nơi lưu trữ thông tin
2	DB	Database	Nơi lưu trữ thông tin
3	Frontend		Giao diện người dùng
4	Backend		Máy chủ/Phần xử lý Logic
5	Authentication		Xác minh người dùng
6	Authorization		Phân quyền người dùng
7	Token		Chuỗi thông tin đã mã hóa

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

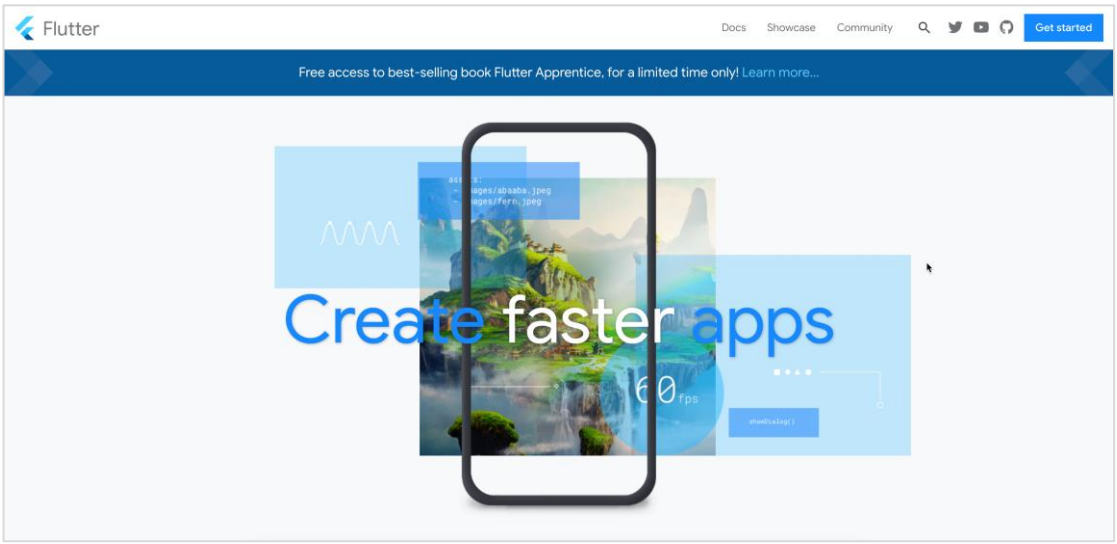
1.1. Đặt vấn đề

Trong bối cảnh phát triển của khoa học và công nghệ, mạng xã hội đã trở nên phổ biến với tất cả mọi người. Với sự hấp dẫn của mình, mạng xã hội đã trở thành một phần không thể thiếu trong cuộc sống thường ngày, đặc biệt là với giới trẻ. Có thể nói mạng xã hội là một kho tàng thông tin và kiến thức khổng lồ. Nó giúp chúng ta dễ dàng tìm kiếm thông tin một cách nhanh chóng và mang đến cho chúng ta những thông tin đa dạng, phong phú.Một số lợi ích có thể kể đến như:

- **Kết nối bạn bè:** Mạng xã hội là phương tiện để những người dùng có thể kết nối với nhau. Thông qua mạng xã hội, có thể **nhắn tin, gọi video call** để gặp mặt bạn bè, người thân bỏ qua những trở ngại về khoảng cách địa lý.
- **Chia sẻ tin tức, kiến thức:** Mạng xã hội là phương tiện để những người dùng có thể kết nối với nhau. Thông qua mạng xã hội, có thể nhắn tin, gọi video call để gặp mặt bạn bè, người thân bỏ qua những trở ngại về khoảng cách địa lý. Đặc biệt, bạn cũng có thể làm quen với những người bạn mới có chung sở thích, lý tưởng, đam mê thông qua các hội nhóm, các trang chung hoặc bắt gặp khi cùng chia sẻ về một vấn đề nào đó. Đó là lý do vì sao nói mạng xã hội là cầu nối gắn kết, củng cố các mối quan hệ hiện hữu và tạo cơ hội xác lập các mối quan hệ mới.
- **Chia sẻ cảm xúc cá nhân:** Thông qua mạng xã hội, có thể bày tỏ quan điểm cá nhân cũng như chia sẻ cảm xúc, ý kiến về một vấn đề nào đó. Đôi khi niềm hạnh phúc sẽ được nhân lên khi có nhiều người bạn cùng chung vui hoặc nỗi buồn vơi đi khi có người cùng chia sẻ, đồng cảm.
- **Bán hàng, kinh doanh:** Mạng xã hội tạo điều kiện cho nhiều bạn có thể bán hàng, kinh doanh các mặt hàng đa chủng loại hoặc quảng cáo cho thương hiệu của mình để tiếp cận nhiều đối tượng khách hàng hơn. Việc kinh doanh, kiếm tiền online trên mạng xã hội ngày càng phổ biến và đem lại nguồn thu nhập không nhỏ cho các chủ shop, doanh nghiệp.
- **Giải trí:** Lợi ích tuyệt vời của mạng xã hội đó là giúp người dùng có những phút giây giải trí, thư giãn với những tiện ích tích hợp như game, âm nhạc, phim ảnh. Thậm chí, việc xem newfeed và đọc những bài đăng mang ý nghĩa tích cực cũng giúp người dùng cảm thấy bớt mệt mỏi và căng thẳng sau một ngày làm việc, học tập vất vả.

1.2. Các công nghệ được lựa chọn

1.2.1. Frontend - Flutter (Cross-platform UI framework)



Hình 1.1. Trang chủ https://flutter.dev

Ra mắt vào Tháng 5 năm 2017, Flutter với sứ mệnh là một bộ thư viện dựa trên ngôn ngữ Dart (đều do Google tạo ra) giúp lập trình viên phát triển các phần mềm đa nền tảng (đa nền tảng đối với Flutter là việc viết phần mềm một lần bằng một ngôn ngữ Dart, tuy nhiên cho khả năng tự biên dịch ra các phần mềm tương tự nhau ở các nền tảng hệ điều hành như Android, iOS, Windows, Linux, MacOS, Web frontend… thay vì như cách phát triển truyền thống phải tự xây dựng từng phần mềm bằng ngôn ngữ lập trình riêng biệt cho mỗi nền tảng[Java/Kotlin cho Android; Swift/Objective-C cho iOS] – dẫn đến tốn thời gian và chi phí gấp nhiều lần).

Các đặc trưng của Flutter:

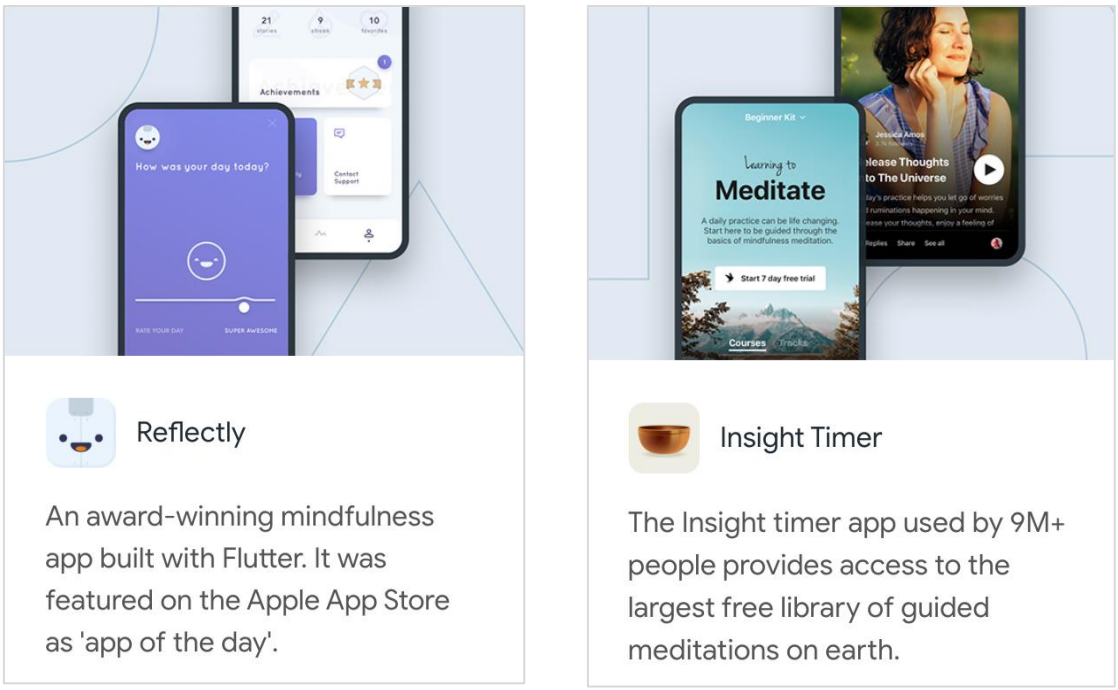
- Sử dụng ngôn ngữ Dart (ngôn ngữ lập trình mềm dẻo, hỗ trợ cả OOP, Reactive Programming và Functional Programming, static và dynamic typing, dễ học với hầu hết mọi người do cú pháp của Dart thuộc họ ngôn ngữ lập trình C [C-based programming language], ngôn ngữ được đội ngũ chuyên gia hàng đầu của Google phát triển - do đó đảm bảo độ tin cậy và mạnh mẽ cho các ứng dụng được viết bằng Dart)
- Hỗ trợ Hot-reload (cơ chế tải lại mã [rebuild] cấp tốc giúp việc phát triển nhanh hơn bao giờ hết, người lập trình chỉ cần tạo thay đổi trong code và nhấn lưu thì giao diện sẽ được ngay lập tức cập nhật)

- Giúp tạo ra các ứng dụng có giao diện nhất quán bởi việc sử dụng trình render giao diện Skia (một trình render giao diện được viết bằng C++, giúp tái tạo lại mọi pixel trong màn hình thay vì sử dụng lại cơ chế sẵn có của hệ điều hành, điều này cho phép người lập trình khả năng tùy chỉnh mọi điểm ảnh trên màn hình của mình, từ đó cho hiệu năng cao hơn so với các công nghệ cùng lĩnh vực như React Native, Xamarin.. hay đặc biệt nhanh hơn khi so với các công nghệ Web-embedded như Ionic/PhoneGap). Ngoài ra, Flutter đảm bảo việc render giao diện giữ ở mức cơ bản 60FPS (60 khung hình/s) hoặc thậm chí 120FPS nếu thiết bị đáp ứng được (xem chi tiết), giúp mang lại trải nghiệm mượt mà cho người dùng
- Bản thân Flutter đã và đang được **đội ngũ phát triển** tạo ra một kho các widget (controls/components/hay gọi là các thành phần giao diện) phong phú, giúp cho Flutter dễ dàng tiếp cận hơn đa số các công nghệ đa nền tảng khác. Họ không ngừng chia sẻ các video giới thiệu về các widgets, các series trên YouTube "vọc vạch" sâu vào cơ chế hoạt động của Flutter để giúp bất kì ai khi làm quen cũng sẽ có một kiến thức vững chắc về cách Flutter hoạt động
- Flutter sử dụng ngôn ngữ Dart cho cả việc xây dựng code logic và giao diện (không giống như Java hay Kotlin cần viết riêng giao diện vào các file .xml), điều này giúp người lập trình không cần phải học thêm những khái niệm khác khi tạo giao diện – giúp quá trình phát triển được nhất quán và nhanh hơn bao giờ hết.

Bản chất Flutter chỉ là một bộ UI framework giúp tạo giao diện nhanh chóng, nó không thể trực tiếp cung cấp cho người lập trình các API của hệ điều hành (các tác vụ thao tác với phần cứng, can thiệp sâu vào hệ thống như đọc ghi file, truy cập Camera, Micro…) như cách mà các công nghệ Native như Android (Java/Kotlin) hay iOS (Swift/Objective-C) làm được.

Để khắc phục nhược điểm đó, Flutter cung cấp cơ chế có tên gọi MethodChannel. Đây là một cơ chế giúp người lập trình có thể đứng ở code Flutter (Dart) và gọi được tới các hàm nằm ở phía ứng dụng Native của hệ điều hành để lấy thông tin, điều này giống như trái tim của Flutter khi nó vốn chỉ là thư viện tạo giao diện, nhờ đó làm tăng tính khả dụng khi lập trình phần mềm cho nhiều nền tảng mà vẫn đảm bảo tính tương thích hoàn toàn và tạo nên được những ứng dụng giúp xử lí hầu hết những yêu cầu khắt khe từ người dùng không kém cạnh những nền tảng Native.

Với những ưu điểm kể trên của Flutter, tôi chọn Flutter làm nền tảng để phát triển phần mềm cho đề tài này, với mong muốn phần mềm được tạo ra nhanh chóng, có một giao diện bắt mắt, hiệu năng tốt và chạy được trên nhiều nền tảng – do trong thực tế đối tượng sử dụng sẽ có thể dùng một trong hai hệ điều hành Android hoặc iOS



Hình 1.2. Các ứng dụng nổi tiếng được xây dựng bởi Flutter

1.2.2. Backend - Firebase

Firebase được Google ra mắt vào Tháng 4 năm 2012, tới nay (2021) đã hơn 9 năm phát triển. Tiếp đến, vào năm 2014, Google mua lại Firebase và phát triển nó thành một dịch vụ đa chức năng được hàng triệu người sử dụng cho đến hiện nay.

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu. Là một bộ công cụ đa năng giúp cho các nhà lập trình thúc đẩy quá trình phát triển sản phẩm với tốc độ cao ở phía giao diện người dùng mà không cần quan tâm nhiều đến Backend.

Firebase cung cấp **các cách tương tác cơ bản** sau để kết hợp và xây dựng nên một hệ thống Phần mềm/Web:

- **Web Console**: Bộ công cụ quản trị các chức năng trên giao diện Web console.firebase.google.com
- **Backend SDK**: Bộ thư viện phía Backend (dành cho máy chủ Web), giúp người lập trình có quyền truy cập sâu (quyền quản trị) vào các chức năng của Firebase
- **Frontend SDK**: Bộ thư viện phía Frontend (dành cho Web frontend, Mobile/Desktop apps), giúp người lập trình kết nối tới/sử dụng các chức năng của Firebase.

Hiện tại, Firebase cung cấp các **chức năng/dịch vụ chính** sau:

- **Authentication**: Xác thực người dùng. Với Firebase Authentication, người lập trình có thể tích hợp nhiều cách thức đăng nhập vào hệ thống như Xác thực bằng Email, số điện thoại, thông qua tài khoản của các bên như Facebook, Apple, Microsoft, Github…
- **Realtime Database:** Cơ sở dữ liệu thời gian thực. Dữ liệu được lưu trữ ở định dạng cây JSON. Firebase Realtime Database cung cấp các bộ Client SDK (thư viện phát triển ở phía giao diện người dùng) cho phép cập nhật (đồng bộ) trong thời gian thực khi dữ liệu có thay đổi
- **Firestore Database:** Cơ sở dữ liệu thời gian thực mở rộng. Đây là một kiểu CSDL mới được Firebase ra mắt vào Tháng 10/2017, cũng cho khả năng cập nhật thời gian thực như Firebase Realtime DB, thêm vào đó cho phép lưu trữ các loại dữ liệu phức tạp hơn, hỗ trợ việc truy vấn với các điều kiện đầy đủ hơn.
- **Storage:** Lưu trữ tập tin. Firebase Storage cho phép lưu trữ, truy xuất, tải lên/xuống các tập tin như: Ảnh, video, âm thanh…
- **Hosting:** Lưu trữ website frontend. Firebase Hosting cho phép lưu trữ các website có nội dung ở phía frontend (JavaScript/HTML/CSS, ReactJS, VueJS, Angular, Svelte…). Hỗ trợ publish ra môi trường Internet với domain tùy chỉnh
- **Functions:** Hàm từ xa. Firebase Functions cho phép người lập trình viết các đoạn mã/hàm và lưu trữ các mã ấy trên Firebase Web Console (có nghĩa những đoạn mã này không nằm trong Phần mềm/Web frontend), những mã này có thể tự động chạy với quyền quản trị để kích hoạt các chức năng tương tự Trigger khi CSDL có thay đổi ở một bảng/một trường nào đó; cho phép các bộ SDK ở phía

Frontend gọi tới các hàm này – làm tăng tính trừu tượng/giúp bao đóng, bảo mật thông tin của các chức năng quan trọng

- **Cloud Messaging:** Gửi tin nhắn/thông báo/token tới Frontend. Firebase Cloud Messaging (FCM) cho phép người lập trình gửi các tin nhắn/thông báo/token (chuỗi thông tin đã má hóa) tới các thiết bị trong hệ thống (dựa vào device-id khi thiết bị đăng nhập hoặc dựa trên topic/nhóm của thiết bị đăng ký). Đây là chức năng thường được dùng cho việc tạo các **Thông báo đẩy** (Push Notification)
- **AdMob:** Tạo các banner quảng cáo. Firebase AdMob cho phép các nhà lập trình hiển thị các quảng cáo của hệ thống Google trong ứng dụng nhằm nhận tiền khi người dùng tương tác
- **Bảo mật:** Firebase cung cấp cơ chế bảo mật cho các CSDL/dịch vụ bằng các Security Rules. Các rules này được lưu trữ ở phía Console (nằm ngoài các ứng dụng frontend), cho phép linh hoạt tạo ra các ràng buộc lên một bảng hoặc một trường dữ liệu nhất định nhằm tạo hàng rào bảo mật cho các CSDL ở mức an toàn nhất có thể.

Ra mắt đã lâu, với cộng đồng lớn mạnh và đội ngũ nhà phát triển chất lượng, Google Firebase là nền tảng vững chắc giúp người lập trình tạo nên các sản phẩm chất lượng trong thời gian ngắn, bảo mật cao, tận dụng được hệ sinh thái của Google để thúc đẩy quá trình phát triển các sản phẩm diễn ra nhanh chóng hơn phương thức tuyền thống nhiều lần.

Với đề tài này, mong muốn của tôi là phần mềm có một nền tảng tốt đáng tin cậy, dễ cấu hình và triển khai, sản phẩm cuối cùng có thể mang ra sử dụng được trong thực tế. Do đó Firebase là một lựa chọn hợp lí.

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Khảo sát sơ bộ

2.1.1. Các đối tượng sử dụng

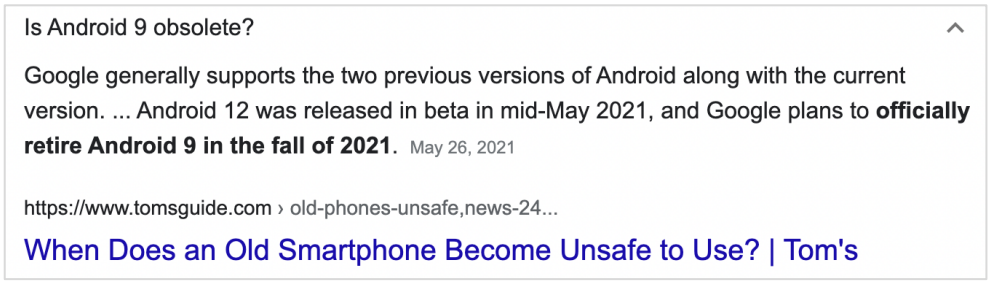
- Người sử dụng: điện thoại, máy tính bảng.

2.1.2. Các dòng thiết bị

- ***Android***: Phiên bản phổ biến hiện tại khoảng *Android 8*.

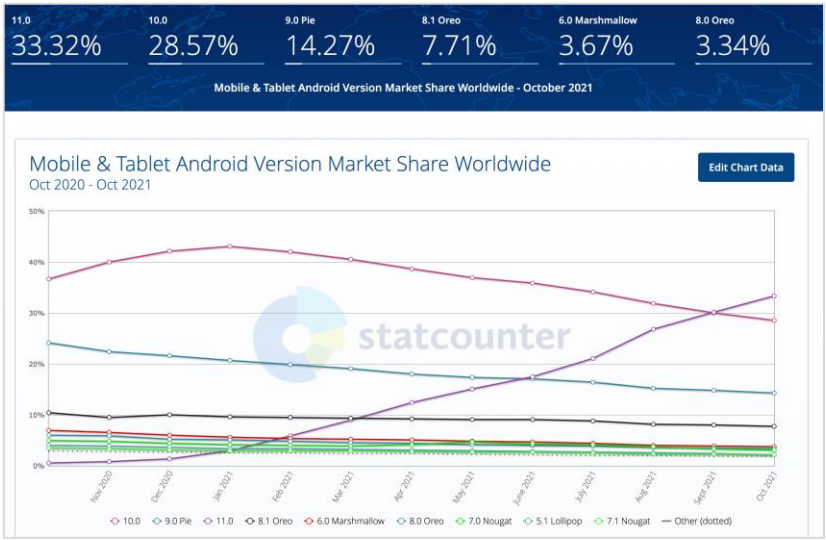
Theo nhận định, những người được trang bị một chiếc điện thoại. Thời điểm này tại Việt Nam, các hãng phân phối điện thoại nổi tiếng như: Thế giới Di động, Điện máy xanh, CellphoneS, Hoàng Hà Mobile… hầu hết đều cung cấp các mẫu điện thoại chạy Hệ điều hành Android phiên bản 8.0 trở lên, kể cả các mẫu điện thoại có giá thấp nhất.

Do các mẫu điện thoại đời mới được sản xuất theo hướng rẻ, đẹp, đa dạng, bắt kịp xu thế… nên nhà sản xuất thường trang bị các cấu hình hiện đại/chưa bị lỗi thời.



Hình 2.1. Các phiên bản Android được hỗ trợ

Thống kê: Mức phổ biến của các phiên bản Android (Tháng 10/2021)



Hình 2.2. Mức phổ biến của các phiên bản Android (Tháng 10/2021)

- ***iOS:*** Phiên bản hệ điều hành phổ biến hiện tại: ***iOS 14.***
  Phiên bản iOS khả dụng: **iOS 12.**

Với đặc trưng là các thiết bị nhất quán (chỉ có các thiết bị thuộc nhà Apple mới có thể chạy hệ điều hành iOS), việc ra các phiên bản cập nhật hệ điều hành của Apple diễn ra thường xuyên hơn do không có nhiều sự khác biệt lớn giữa các dòng điện thoại (đối với Android, do hệ điều hành này được nhiều hãng điện thoại phân phối và tùy chỉnh riêng, do đó tạo nên sự khác biệt lớn giữa các điện thoại Android trong việc cập nhật phiên bản).

Hiện tại **phiên bản iOS thấp nhất còn khả dụng là iOS 12**, sở dĩ vì hiện tại các dòng điện thoại chạy các chip cũ (A7)/bộ nhớ thấp (1GB RAM) như iPhone 5S/6/6 Plus vẫn còn được một bộ phận không nhỏ người dùng sử dụng. Trong khi iOS 13 trở lên không còn hỗ trợ cập nhật cho các mẫu điện thoại kể trên.

Thống kê: Mức phổ biến của các phiên bản iOS (Tháng 10/2021)

Hình 2.3. Mức phổ biến của các phiên bản iOS (Tháng 10/2021)

2.2. Phân tích Hệ thống

Chuyên mục này được triển khai bằng UML – Ngôn ngữ mô hình hoá thống nhất .

Các dạng Biểu đồ được thể hiện cho mỗi chức năng bao gồm:

- UseCase Specification (Đặc tả chức năng)

- UseCase Diagram: Mô tả các mối quan hệ và sự phụ thuộc giữa các chức năng.
- Activity Diagram: Mô tả từng luồng chạy chi tiết của chức năng

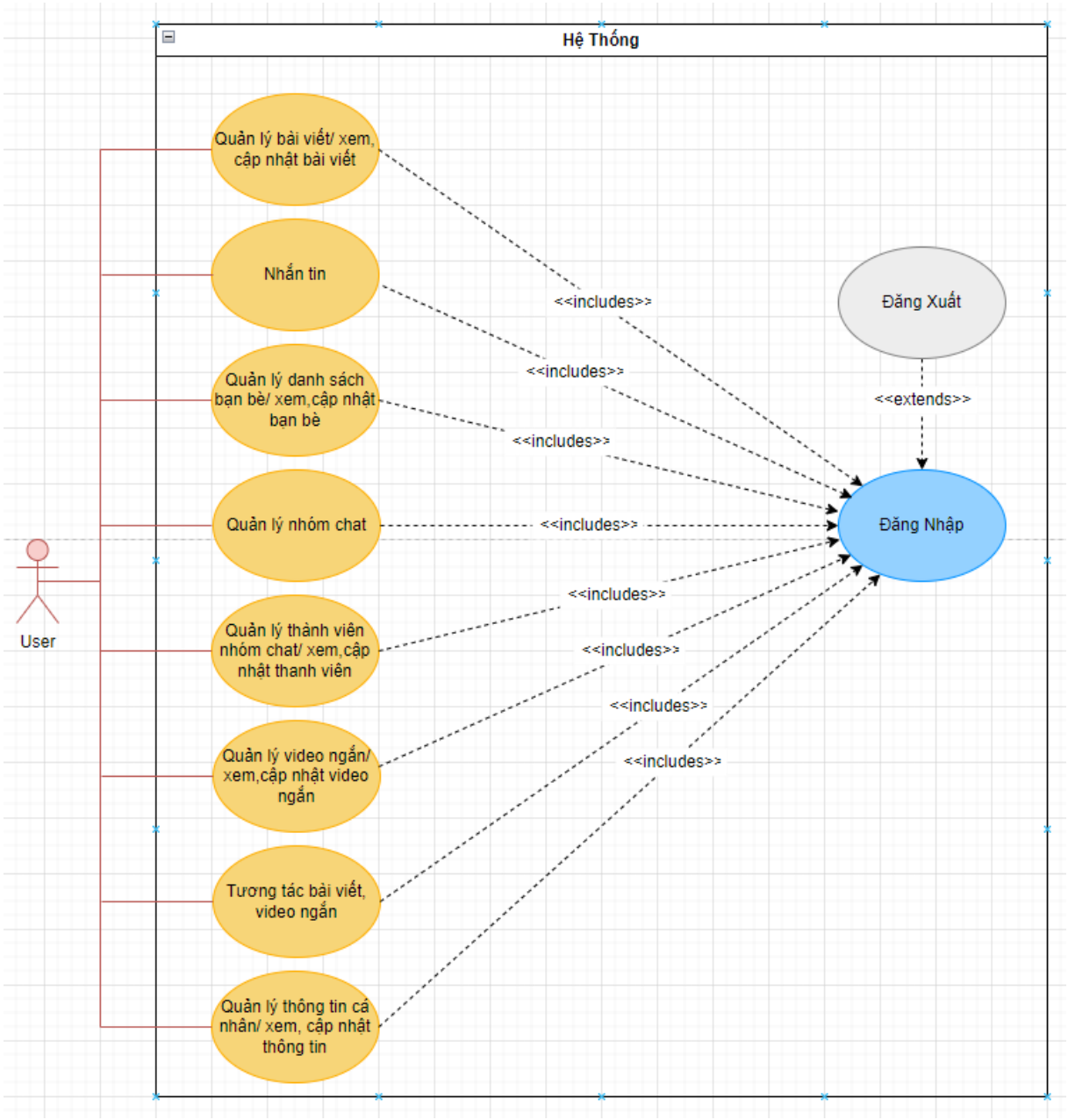
2.2.1. Các tác nhân

Trong hệ thống này, các tác nhân trực tiếp sử dụng ứng dụng bao gồm:

- ***Người dùng:*** *Người trực tiếp sử dụng các chức năng ứng dụng.*

2.2.2. Usecase tổng quát

Biểu đồ khái quát các chức năng có trong ứng dụng.



Hình 2.4. Biểu đồ UseCase tổng quát

2.2.3. Các Usecase chi tiết

2.2.3.1. Chức năng Đăng ký

1. Đặc tả chức năng

Để có thể sử dụng các chức năng của hệ thống, người dùng (User) cần đăng ký thông qua email.

Ứng dụng này sử dụng Firebase Authentication với phương thức đăng nhập thông qua email. Khi người dùng nhập đầy đủ email và password và các thông tin, hệ thống sẽ gửi link xác thực tới email đó. người dùng vào email ấn vào link đó để xác minh rằng mình là chủ sở hữu.

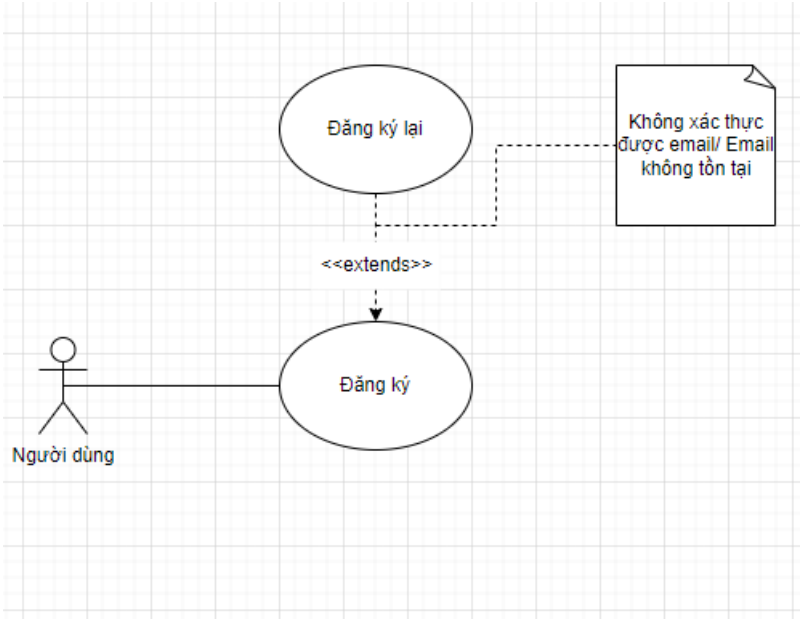
Phương thức đăng ký thông qua email đang trở nên phổ biến những năm gần đây, các ứng dụng về Internet Banking, Sàn thương mại điện tử hay cả những ứng dụng Mạng xã hội đều đã và đang sử dụng phương thức này cho việc xác minh danh tính người dùng, điều đó cho thấy đây là phương thức an toàn và đáng tin cậy.

Bảng 1. Đặc tả chức năng Đăng ký.

Summary	
UseCase Name	Đăng ký
Descriptions	Người dùng muốn đăng ký tài khoản vào ứng dụng để sử dụng các chức năng từ ứng dụng
Actor	Người dùng (Mọi người)
Priority	Phải có / Tiên quyết
Trigger	Nhấn nút “Đăng ký” trong App
Pre-conditions	Sở hữu email. Kết nối Internet ổn định
Post-conditions	Người dùng đăng ký thành công Truy xuất thông báo hiển thị lên màn hình của App.

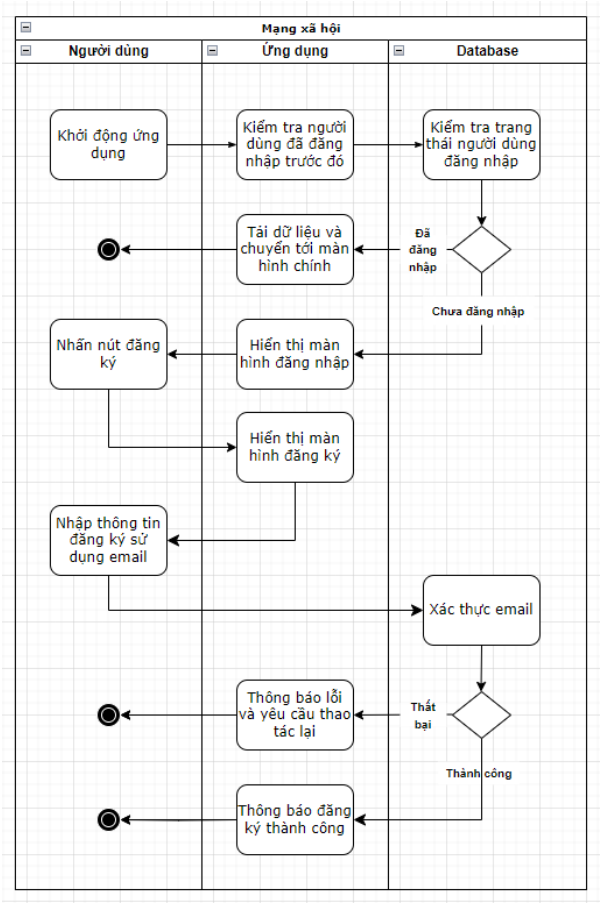
Flows	
Basic Flow	1. Người dùng khởi động ứng dụng 2. Ứng dụng kiểm tra nếu đã đăng nhập thì chuyển đến màn hình chính, ngược lại chuyển đến màn hình đăng nhập. 3. Người dùng không có tài khoản, chọn nút "Đăng ký". 4. Hệ thống chuyển đến màn hình đăng ký, người dùng nhập email, password, họ tên và ấn nút "Đăng Ký". 5. Hệ thống gửi link tới email người dùng đã nhập, đồng thời ứng dụng hiển thị màn hình xác thực gửi email. 6. Người dùng vào email ấn vào link đó để xác thực email đó là của mình. 7. Hệ thống kiểm tra, xác nhận người dùng đăng ký thành công. 8. Ứng dụng thông báo đăng ký thành công.
Exception Flow	7.1. Hệ thống không xác minh được, thông báo lỗi tới người dùng và yêu cầu thao tác lại 7.2. Mã OTP không khớp hoặc đã hết hạn, thông báo lỗi và yêu cầu thao tác lại 7.3. Hệ thống không tìm thấy người dùng tương ứng với số điện thoại vừa nhập, thông báo lỗi và yêu cầu thao tác lại
Requirements	
Non-Functional	Email được gửi tới sau thời gian 5 phút sẽ hết hiệu lực.

2. Biểu đồ Usecase



Hình 2.5. Biểu đồ Usecase chức năng Đăng ký

3. Biểu đồ Activity



Hình 2.6. Biểu đồ Activity chức năng Đăng ký

2.2.3.2. Chức năng Đăng nhập.

1. Đặc tả chức năng

Để có thể sử dụng các chức năng của hệ thống, người dùng (user) cần đăng nhập thông qua email và password.

Ứng dụng này sử dụng Firebase Authentication với phương thức đăng nhập thông qua email. Khi người dùng nhập đúng email và password đã đăng ký.Hệ thống chuyển sang màn hình chính của ứng dụng.

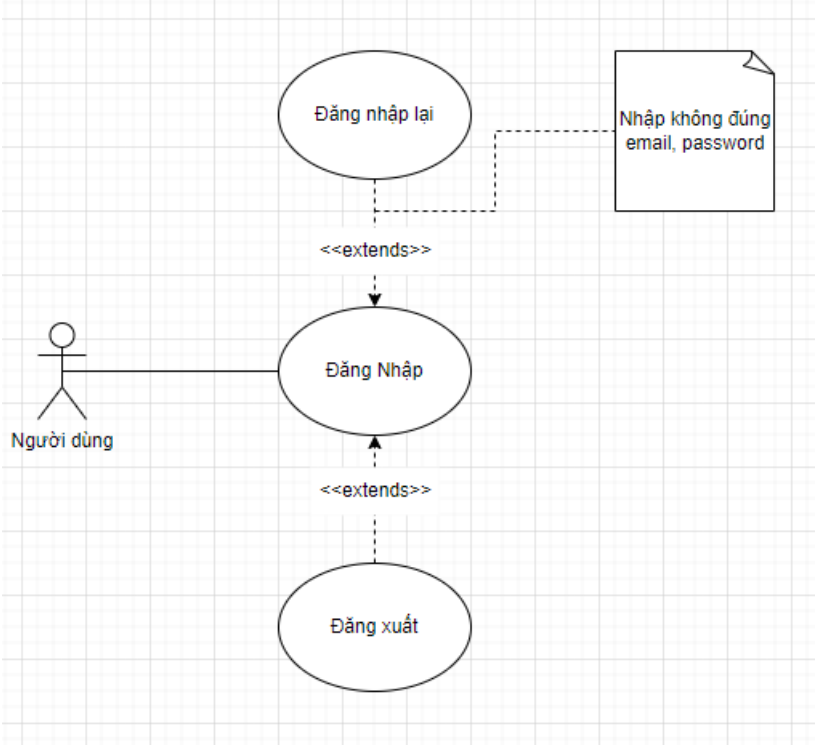
Phương thức đăng nhập thông qua email đang trở nên phổ biến những năm gần đây, các ứng dụng về Internet Banking, Sàn thương mại điện tử hay cả những ứng dụng Mạng xã hội đều đã và đang sử dụng phương thức này cho việc xác minh danh tính người dùng, điều đó cho thấy đây là phương thức an toàn và đáng tin cậy.

Bảng 2. Đặc tả chức năng Đăng nhập.

Summary	
UseCase Name	Đăng nhập
Descriptions	Người dùng muốn đăng nhập vào ứng dụng để sử dụng các chức năng từ ứng dụng
Actor	Người dùng (mọi người)
Priority	Phải có / Tiên quyết
Trigger	Nhấn nút “Đăng nhập” trong App
Pre-conditions	Sở hữu email Kết nối Internet ổn định
Post-conditions	Người dùng đăng nhập thành công Truy xuất các thông tin để hiển thị cho màn hình chính của App.
Flows	
Basic Flow	1. Người dùng khởi động ứng dụng

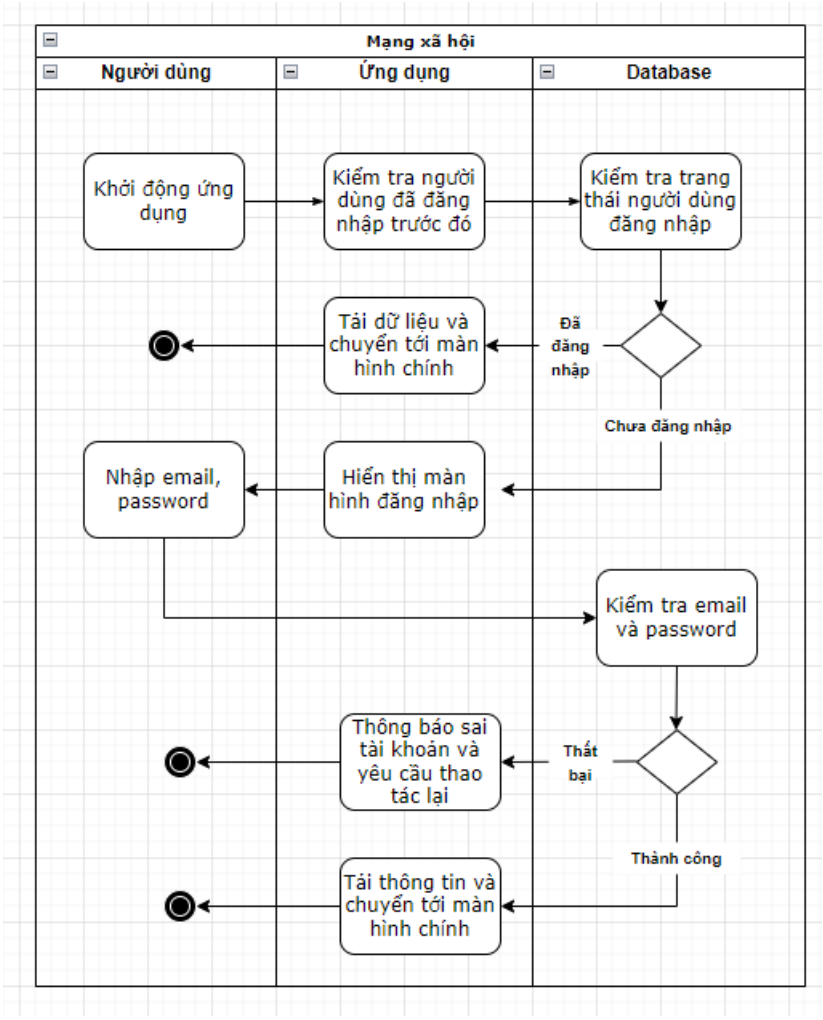
	2. Ứng dụng kiểm tra nếu đã đăng nhập thì chuyển đến Bước 8, ngược lại chuyển đến màn hình đăng nhập 3. Người dùng nhập email và password, chọn nút "Đăng nhập" 4. Hệ thống xác minh thông tin email và password. 5. Hệ thống chuyển dến trang chính của ứng dụng.
Exception Flow	4.1. Hệ thống không xác minh được email, và password. 4.2. Yêu cầu xác minh lại

2. Biểu đồ Usecase



Hình 2.7. Biểu đồ Usecase chức năng Đăng nhập

3. Biểu đồ Activity



Hình 2.8. Biểu đồ Activity chức năng Đăng nhập.

2.2.3.1. Chức năng Nhắn tin

1. Đặc tả chức năng

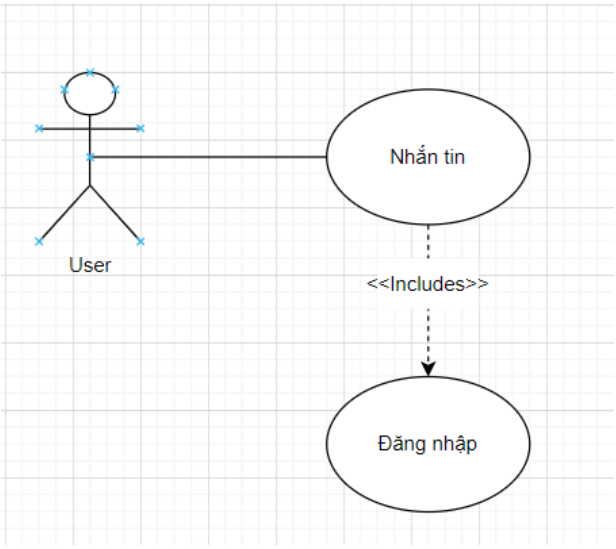
Những người dùng khi đã đăng nhập có thể nhắn tin với nhau. Nhắn tin ngay trong ứng dụng giúp giảm thiểu chi phí so với nhắn thông qua SMS truyền thống.

Bảng 3. Đặc tả chức năng nhắn tin.

Summary	
UseCase Name	Nhắn tin
Descriptions	Người dùng sau khi đăng nhập có thể nhắn tin với nhau

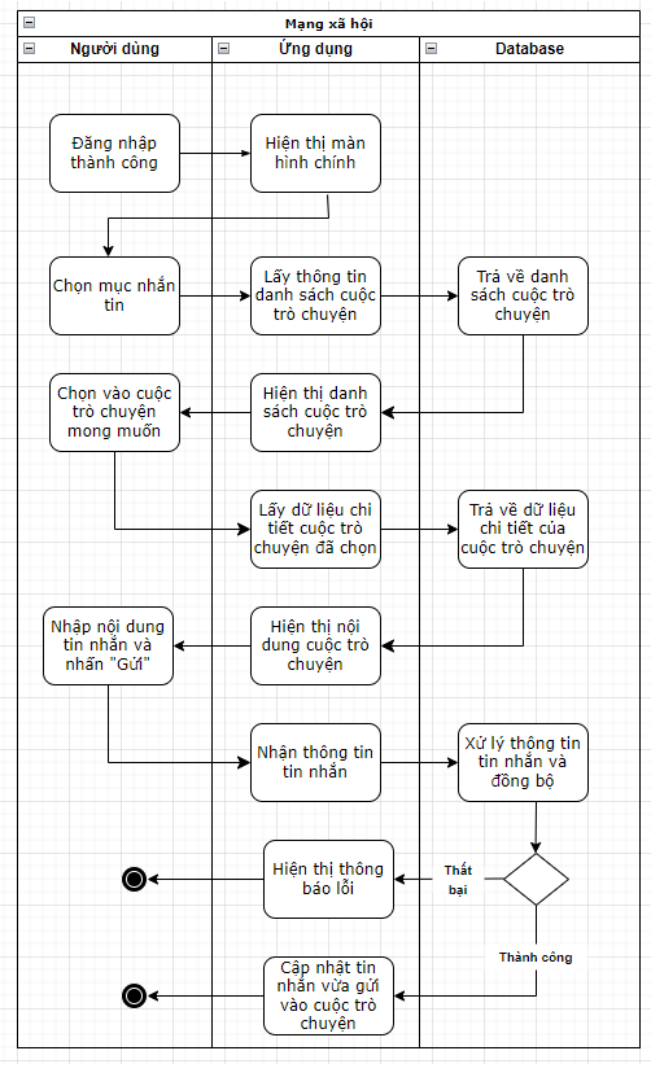
Actor	Người dùng (mọi người)
Priority	Không bắt buộc
Trigger	Mục Tin nhắn trong ứng dụng
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Tin nhắn được gửi tới người nhận, người nhận thấy được tin nhắn mới từ người gửi trong mục Tin nhắn
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn mục Tin nhắn 3. Hệ thống tải các tin nhắn/cuộc trò chuyện 4. Người dùng chọn vào cuộc trò chuyện mong muốn 5. Hệ thống tải chi tiết cuộc trò chuyện 6. Người dùng nhập nội dung muốn nhắn, nhấn "Gửi" 7. Hệ thống xử lý thông tin và chuyển tin nhắn đến Người nhận trong cuộc trò chuyện
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi cho người gửi

2. Biểu đồ Usecase



Hình 2.9. Biểu đồ Usecase chức năng nhắn tin

3. Biểu đồ Activity



Hình 2.10. Biểu đồ Activity chức năng nhắn tin.

2.2.3.2. Chức năng Xem thông báo Bảng tin

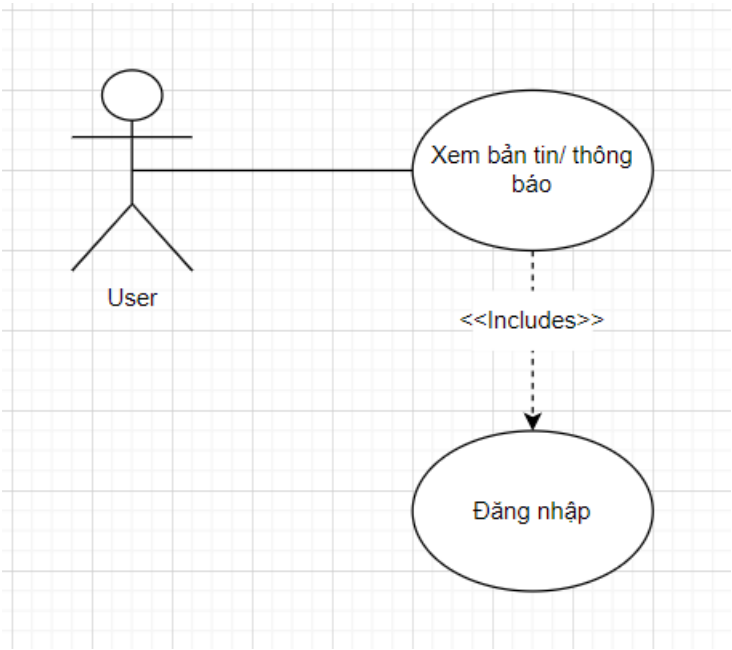
1. Đặc tả chức năng

Những người khi đã đăng nhập có thể xem các thông báo(thông báo như like, comment bài viết của mình).

Bảng 4. Đặc tả chức năng xem thông báo bảng tin.

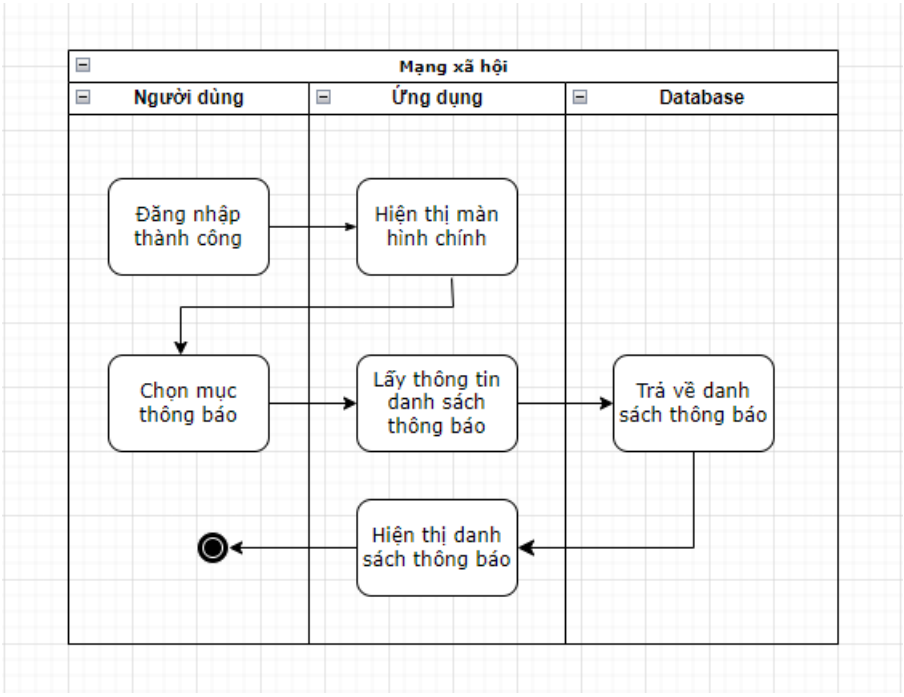
Summary	
UseCase Name	Xem thông báo bảng tin
Descriptions	Người dùng sau khi đăng nhập có thể xem các thông báo ở trang thông báo
Actor	Người dùng (user)
Priority	Không bắt buộc
Trigger	Mục thông báo trong ứng dụng
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng thấy được nội dung của thông báo, ngày giờ.
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn mục thông báo 3. Hệ thống tải các thông báo của bảng tin và hiển thị ra màn hình theo dạng list
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi cho người xem

2. Biểu đồ Usecase



Hình 2.11. Biểu đồ Usecase chức năng xem thông báo

3. Biểu đồ Activity



Hình 2.12. Biểu đồ Activity chức năng xem thông báo

2.2.3.3. Chức năng Quản lý thông tin hồ sơ cá nhân

1. Đặc tả chức năng

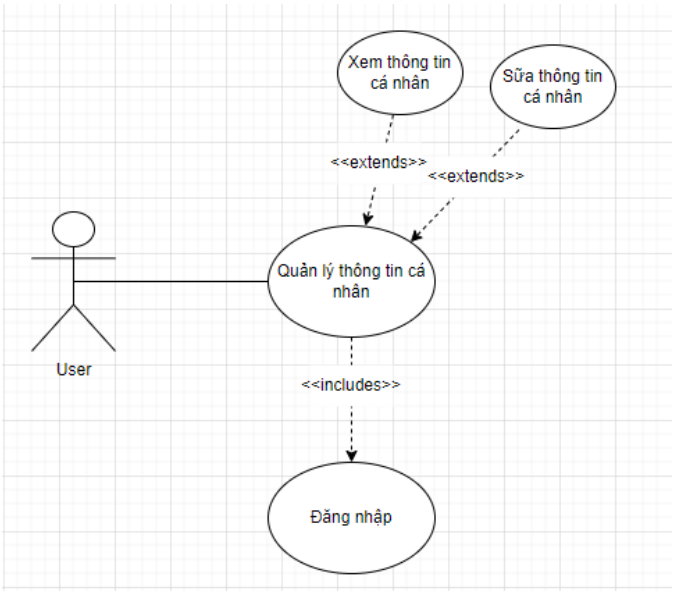
Những người dùng khi đã đăng nhập có thể quản lý thông tin hồ sơ cá nhân của mình.

Bảng 5. Đặc tả chức năng hồ sơ cá nhân.

Summary	
UseCase Name	Quản lý thông tin hồ sơ cá nhân
Descriptions	Người dùng sau khi đăng nhập có thể quản lý hồ sơ cá nhân như xem thông tin cá nhân và chỉnh sửa thông tin cá nhân.
Actor	Người dùng
Priority	Không bắt buộc
Trigger	Mục thông tin cá nhân trong ứng dụng
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng thấy được thông tin cá nhân và có thể sửa thông tin của mình.
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn avatar Cá nhân 3. Hệ thống tải các thông tin cá nhân tương ứng. 4. Người dùng chọn nút “chỉnh sửa thông tin” trên trang thông tin. 5. Hệ thống tại thông tin và người dùng chỉnh sửa và nhấn nút lưu để thay đổi.

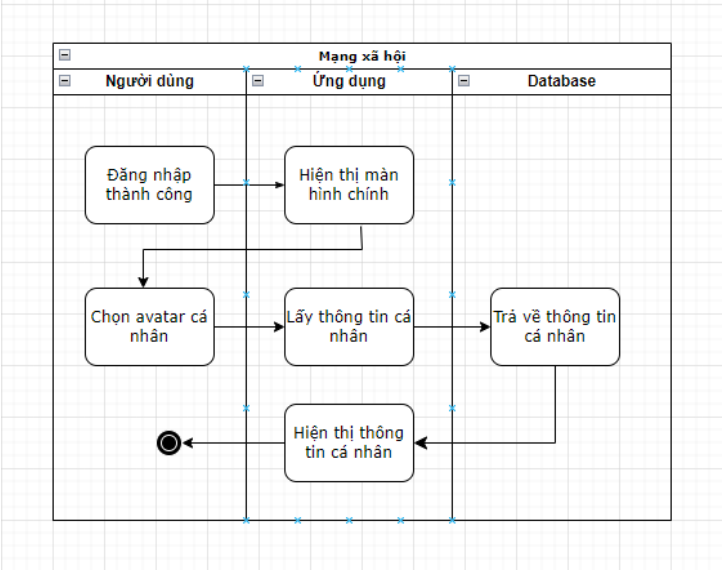
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi cho người xem

2. Biểu đồ Usecase

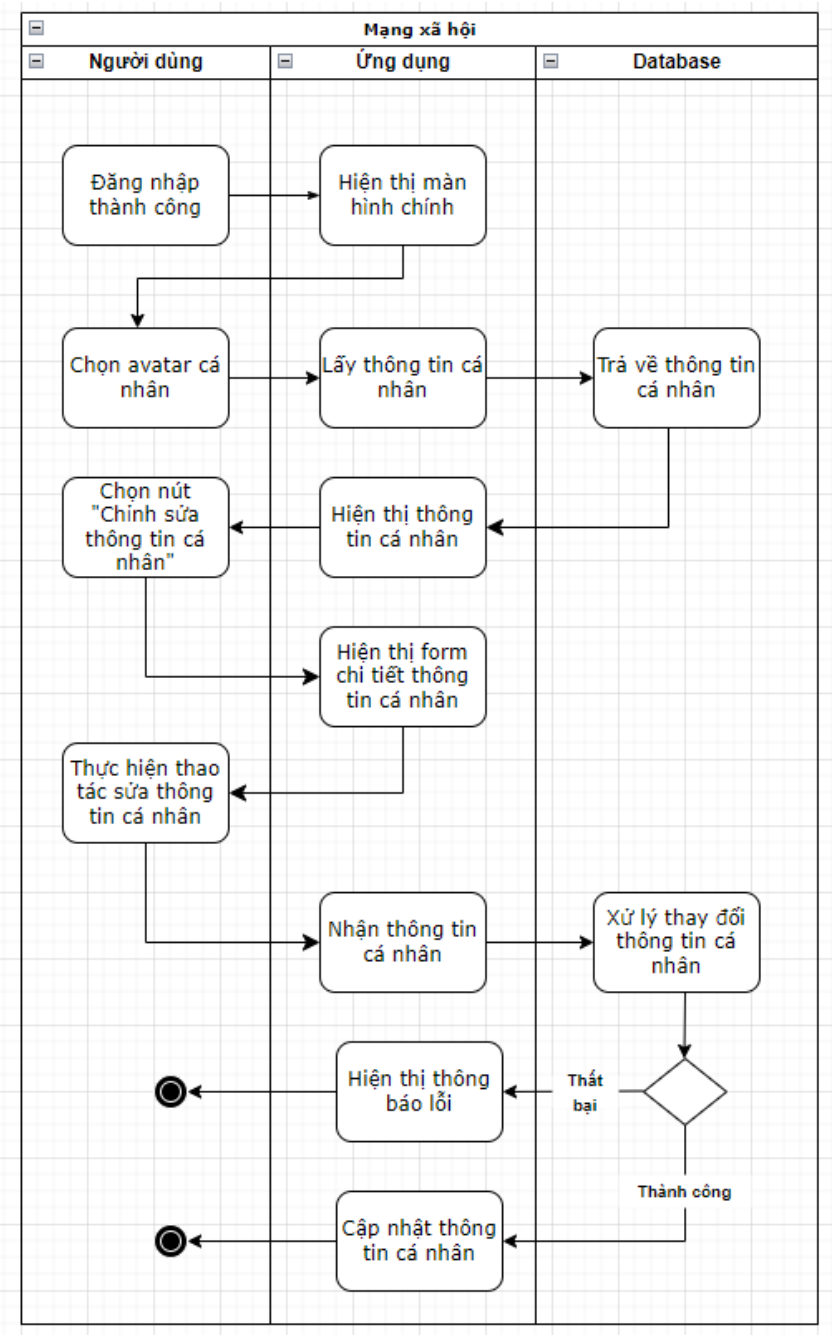


Hình 2.13. Biểu đồ Activity chức năng quản lý thông tin cá nhân.

1. Biểu đồ Activity



Hình 2.14. Biểu đồ Activity chức năng xem thông tin cá nhân



Hình 2.15. Biểu đồ Activity chức năng chỉnh sửa thông tin cá nhân

2.2.3.4. Chức năng Quản lý bài viết

1. Đặc tả chức năng

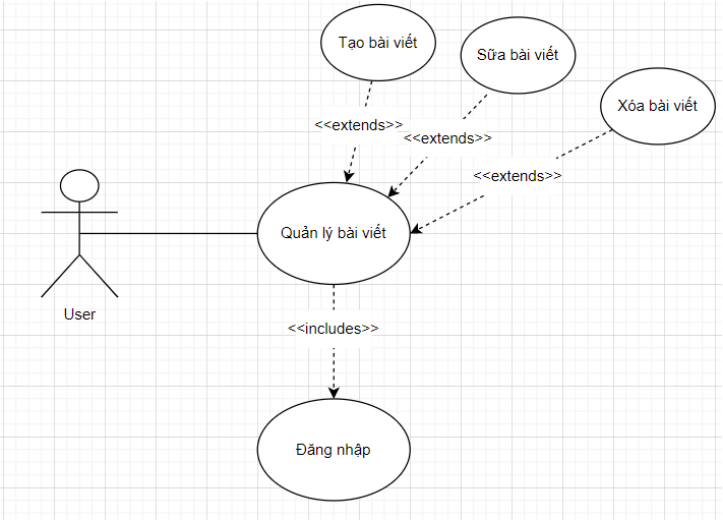
Những người dùng khi đã đăng nhập có thể quản lý thông tin hồ sơ bài viết như thêm bài, xóa bài hoặc chỉnh sửa bài viết của mình

Bảng 6. Đặc tả chức năng Quản lý bài viết.

Summary	
UseCase Name	Quản lý bài viết
Descriptions	Người dùng sau khi đăng nhập có thể quản lý bài viết của mình.
Actor	Người dùng
Priority	Không bắt buộc
Trigger	Mục bài viết
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng quản lý bài viết như tạo bài viết hoặc sửa và xóa bài viết.
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Chọn dấu cộng trên avatar cá nhân 3. Ứng dụng hiện form chọn ảnh và nhập nội dung bài viết. 4. Người dùng nhập các nội dung và ấn nút tạo bài viết để tạo bài viết của mình 5. Hệ thống lưu bài viết. 6. Người dùng vào trang cá nhân, trang cá nhân hiển thị danh sách bài viết của mình. 7. Người dùng chọn biểu tượng 3 chấm của bài viết hiện memu chỉnh sửa và xóa bài viết

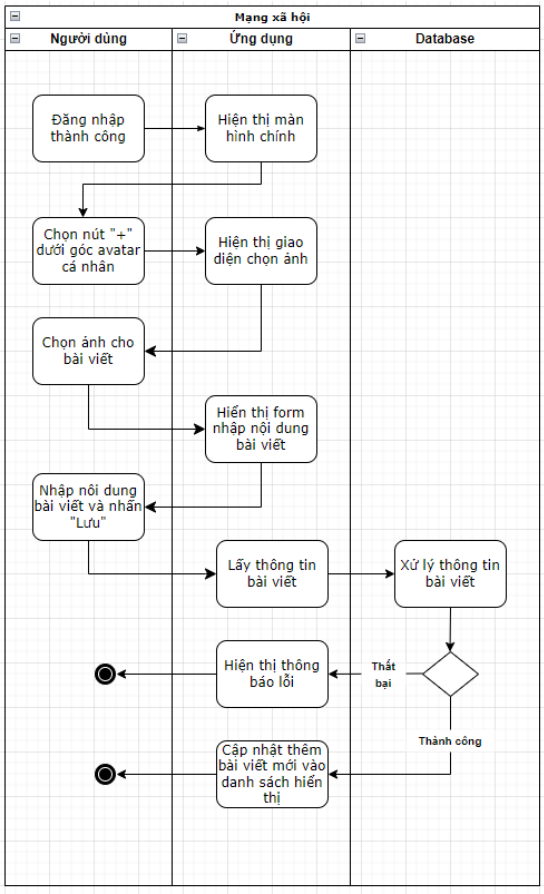
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi

2. Biểu đồ Usecase

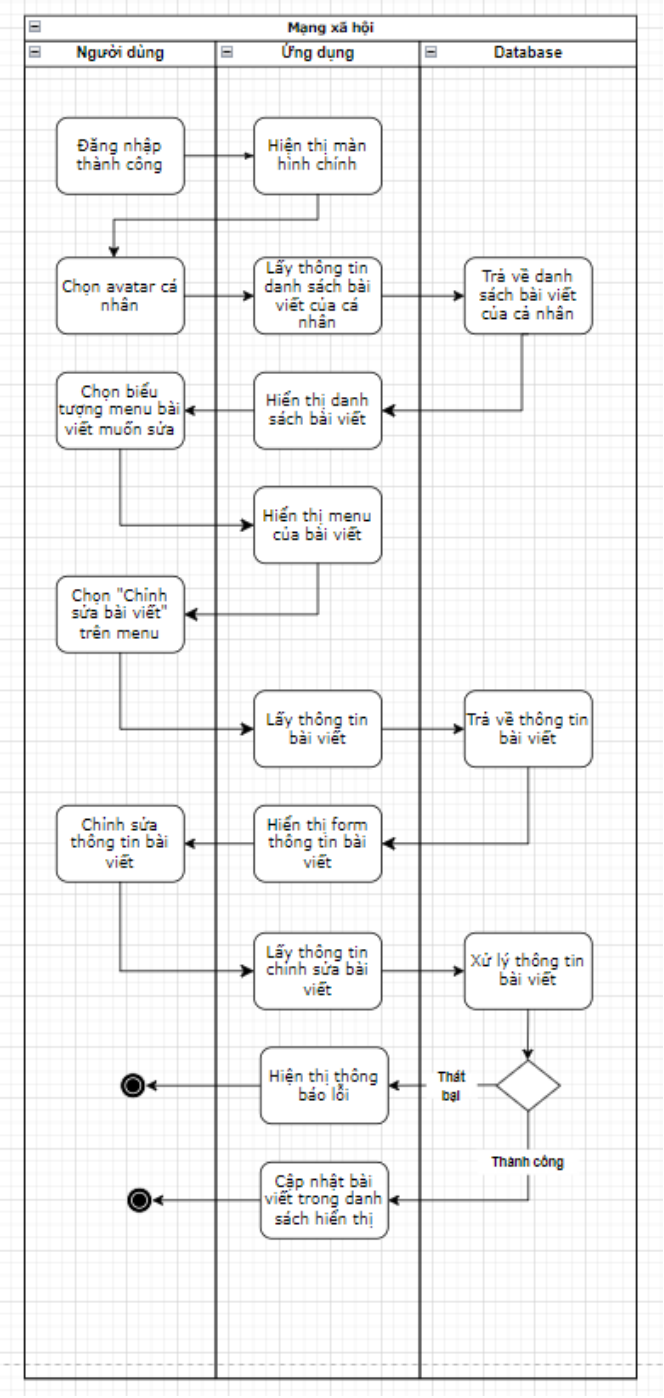


Hình 2.16. Biểu đồ Usecase chức năng Quản lý bài viết

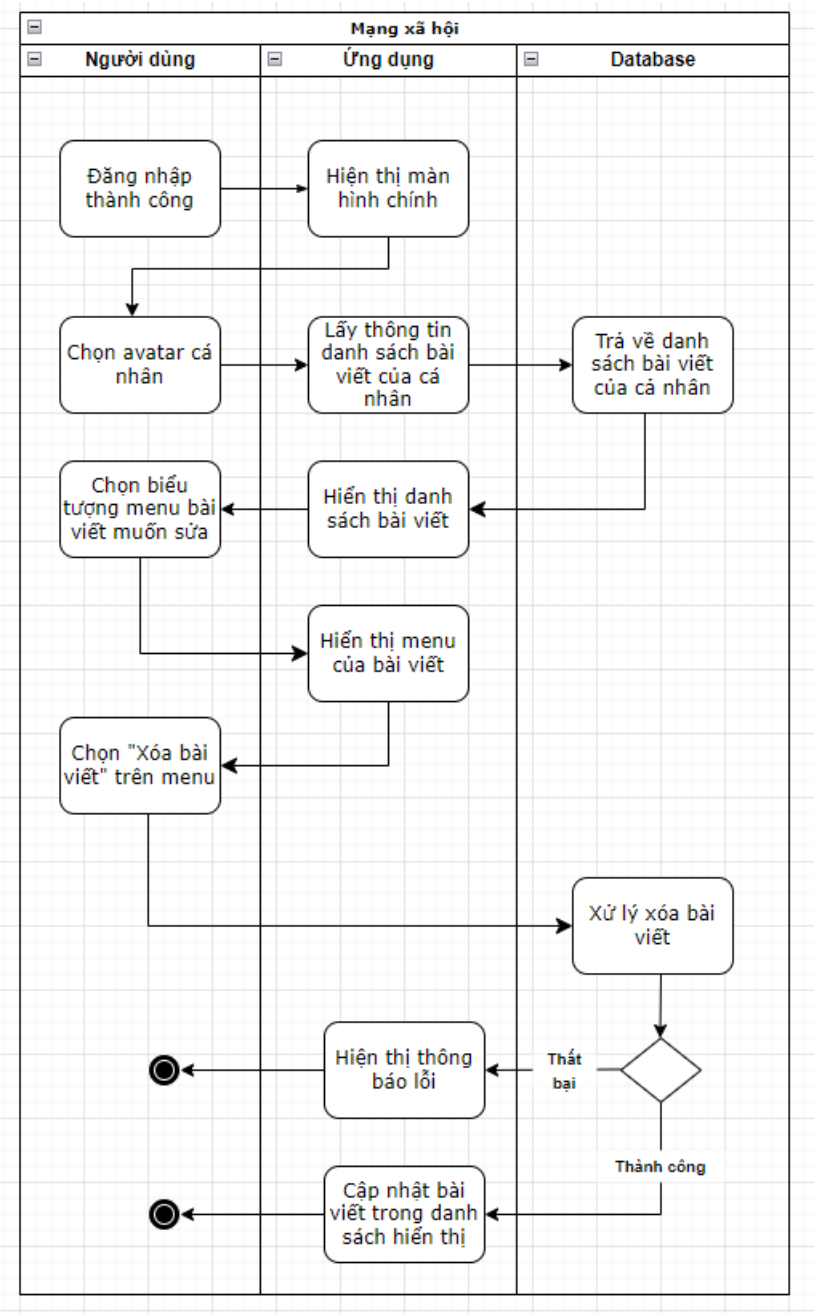
3. Biểu đồ Activity



Hình 2.17. Biểu đồ Activity chức năng thêm bài viết



Hình 2.18. Biểu đồ Activity chức năng chỉnh sửa bài viết



Hình 2.19. Biểu đồ Activity chức năng xóa bài viết

2.2.3.5. Chức năng Quản lý bạn bè

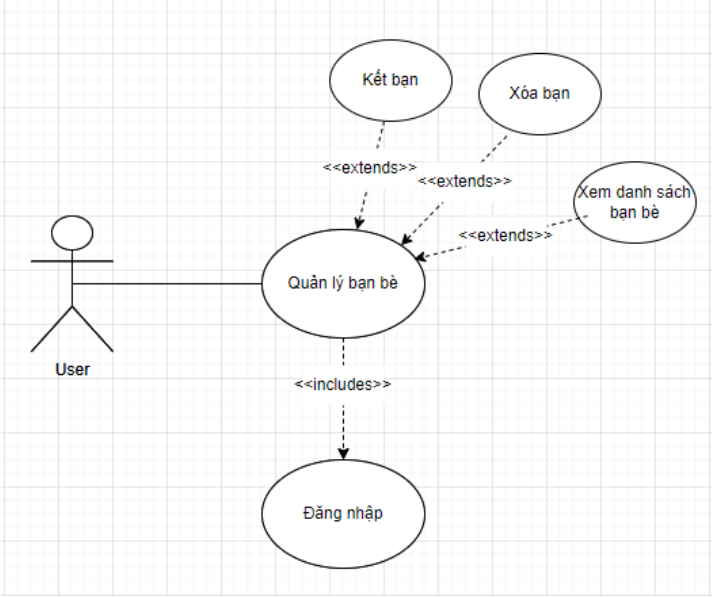
1. Đặc tả chức năng

Những người dùng khi đã đăng nhập có thể quản lý thông tin bạn bè như kết bạn, xóa bạn, xem thông tin bạn bè.

Bảng 7. Đặc tả chức năng Quản lý bạn bè.

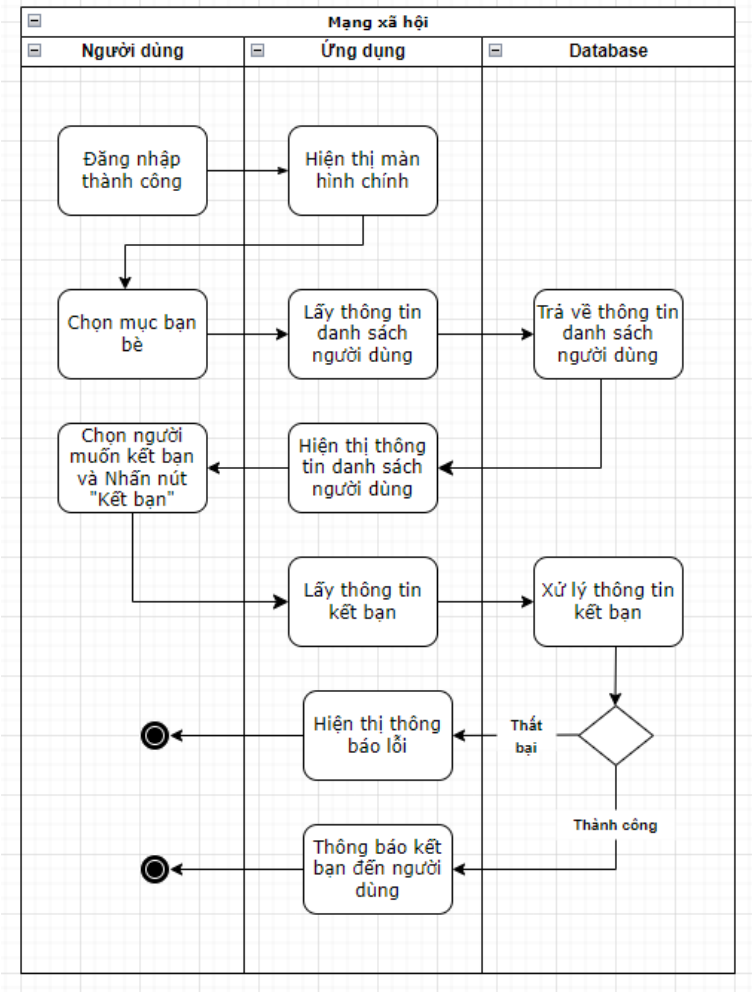
Summary	
UseCase Name	Quản lý bạn bè
Descriptions	Người dùng sau khi đăng nhập có thể kết bạn, xóa bạn
Actor	Người dùng
Priority	Không bắt buộc
Trigger	Kết bạn
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng có thể bạn lẫn nhau Có thông báo kết bạn trên trang thông báo
Flows	
Basic Flow	1. Người quản lí đăng nhập thành công 2. Người mục kết bạn. 3. Hệ thống tải thông tin danh sách các người sử dụng 4. Người dùng tìm kiếm bạn bè. 5. Người dùng ấn vào nút kết bạn 6. Hệ thống xử lí gửi thông báo kết bạn. 7. Người dùng đăng nhập và đồng ý kết bạn
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi

2. Biểu đồ Usecase

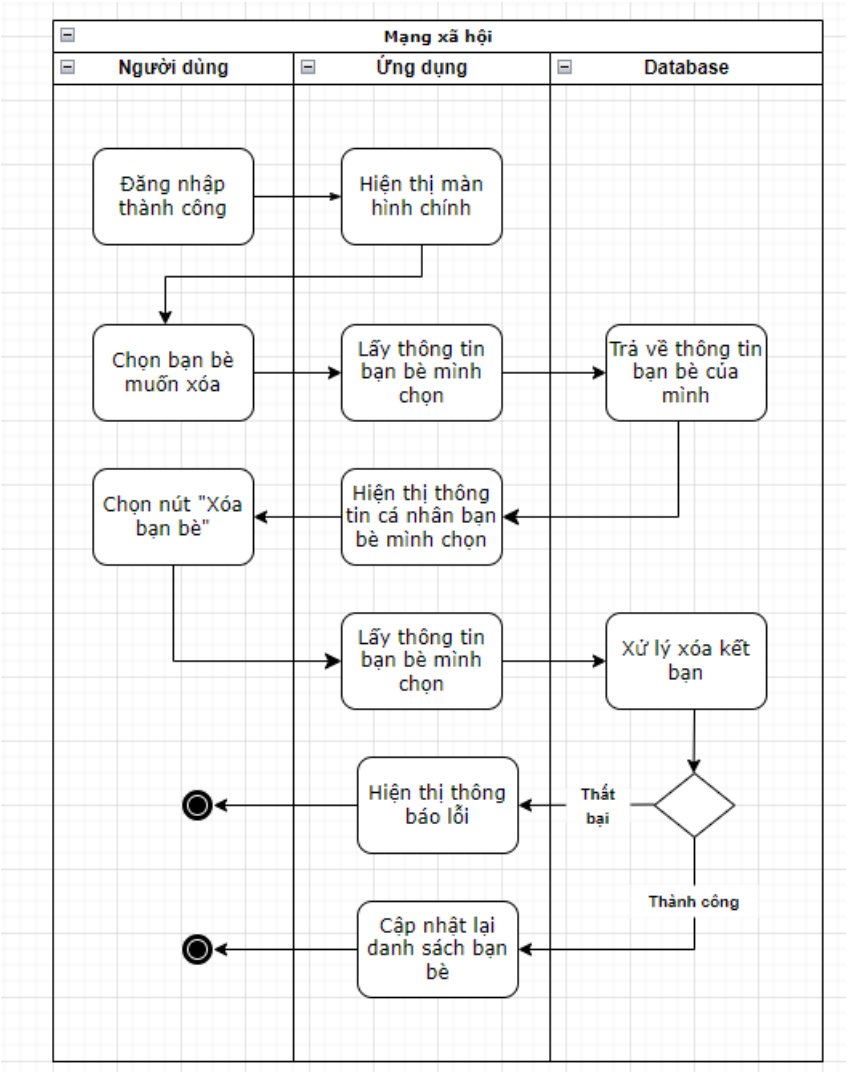


Hình 2.20. Biểu đồ Usecase chức năng quản lý bạn bè

3. Biểu đồ Activity



Hình 2.21. Biểu đồ Activity chức năng kết bạn



Hình 2.22. Biểu đồ Activity chức năng xóa bạn

2.2.3.6. Chức năng quản lý nhóm chat.

1. Đặc tả chức năng

Những người dùng khi đã đăng nhập có thể quản lý thông tin nhóm chat như thêm nhóm, xóa nhóm.

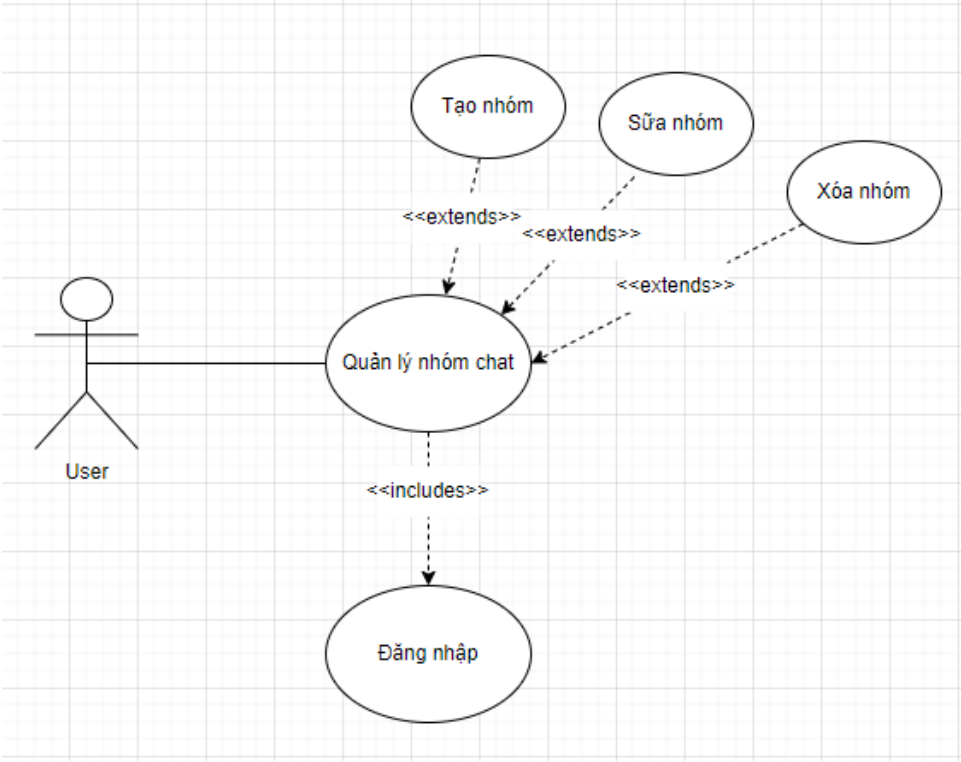
Bảng 8. Đặc tả chức năng quản lý nhóm chat.

Summary	
UseCase Name	Quản lý nhóm chat.
Descriptions	Người dùng sau khi đăng nhập có thể quản lý nhóm như tạo nhóm, xóa nhóm, sửa nhóm

Actor	Người dùng
Priority	Không bắt buộc
Trigger	Mục chat
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Chỉ người trưởng nhóm có thể quản lý nhóm
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn mục chat. 3. Hệ thống tải các danh sách chat. 4. Người dùng chọn tạo nhóm. 5. Người dùng nhập tên nhóm và nhấn nút "Tạo nhóm". 6. Hệ thống xử lý và thêm nhóm tạo vào danh sách chat và qua bước 7 nếu muốn sửa nhóm. 7. Người dùng chọn nhóm sau đó hệ thống hiển thị màn hình chat. 8. Người dùng chọn dấu 3 chấm trên thanh memu và chọn sửa nhóm. 9. Hệ thống hiển thị thông tin nhóm. 10. Người dùng sửa thông tin và nhấn nút "Lưu" để thay đổi, qua bước 11 nếu xóa nhóm. 11. Người dùng chọn dấu 3 chấm trên thanh memu và chọn xóa nhóm.

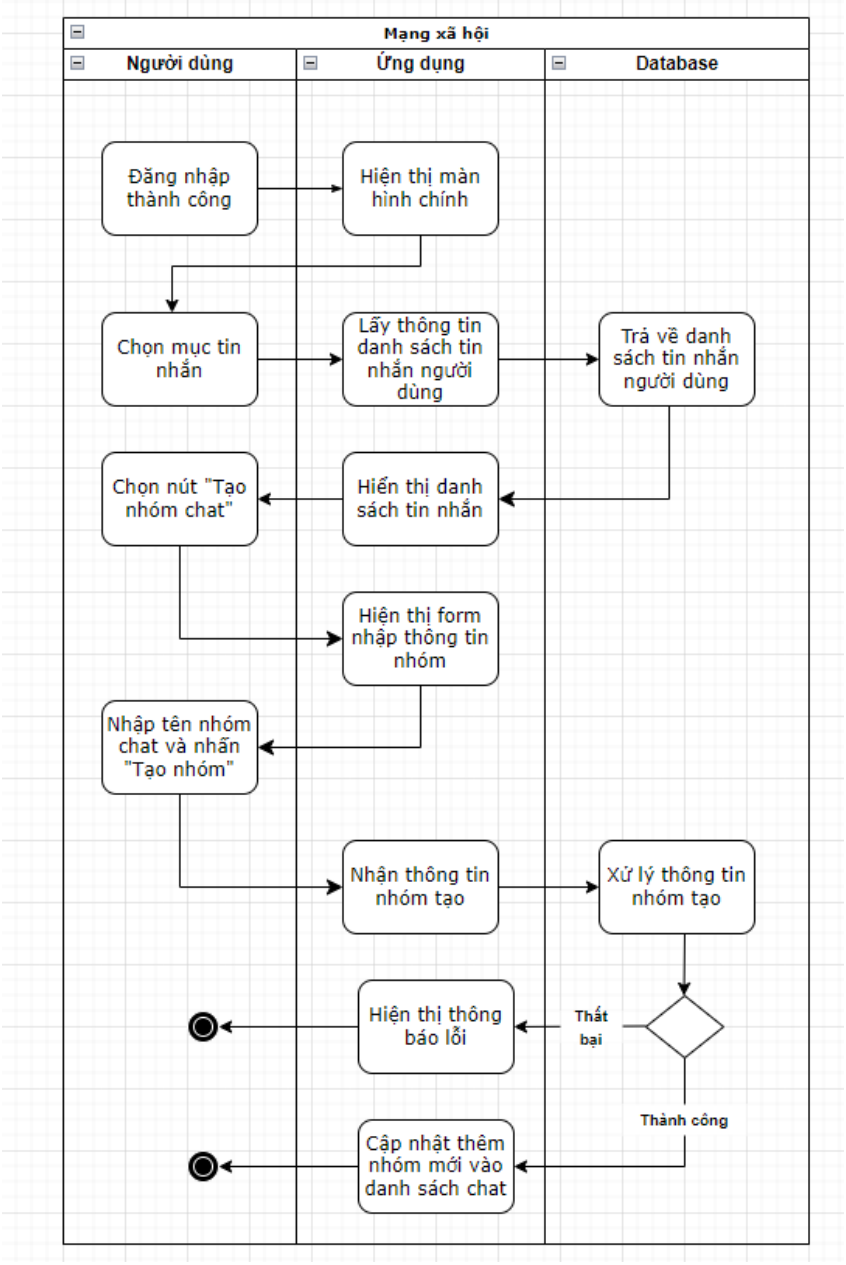
	12. Hệ thống xứ lý và nhóm khỏi danh sách chat.
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi.

2. Biểu đồ Usecase

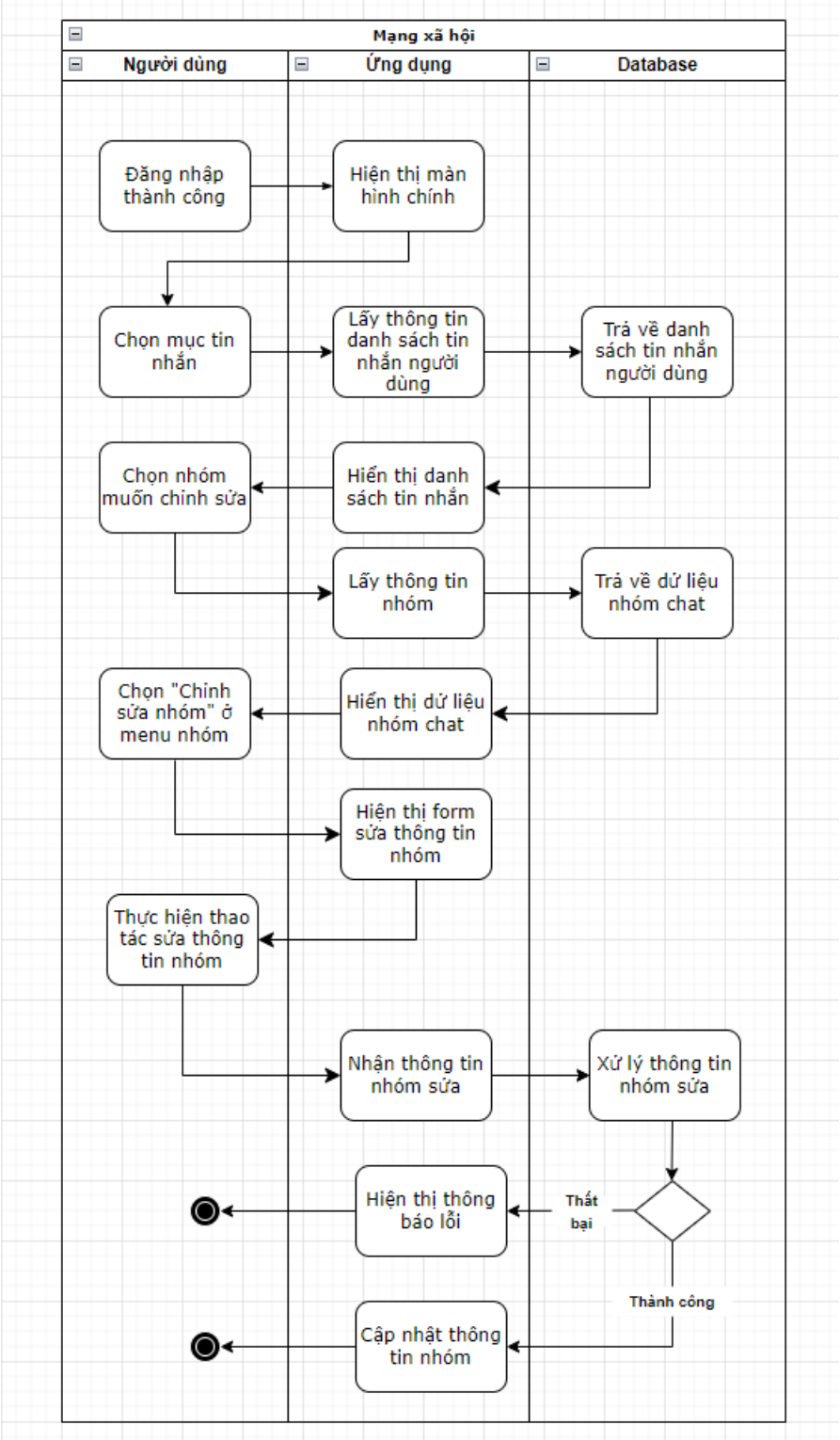


Hình 2.23. Biểu đồ Usecacse chức năng Quản lý nhóm chat

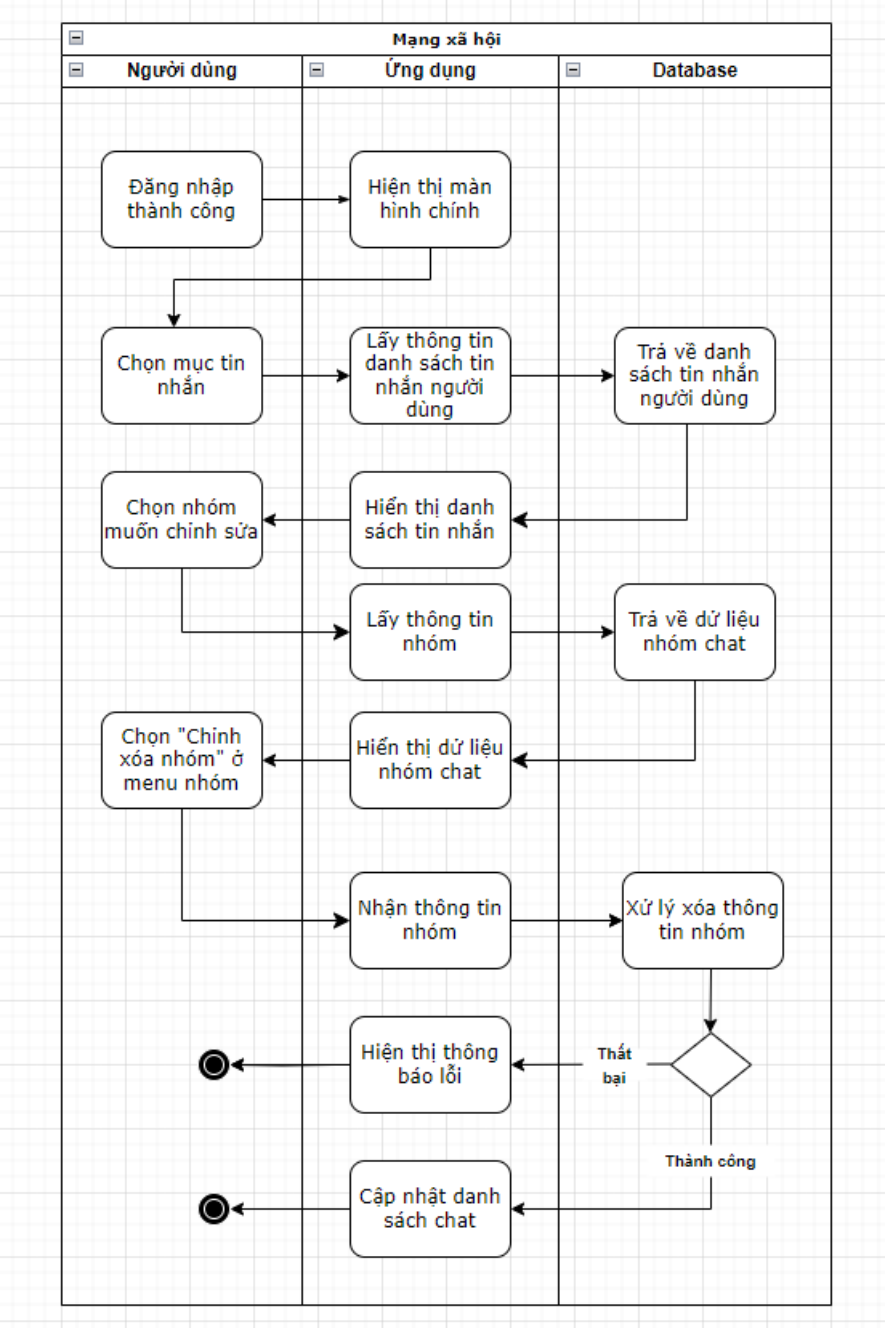
3. Biểu đồ Activity



Hình 2.24. Biểu đồ Activity chức năng tạo nhóm chat.



Hình 2.25. Biểu đồ Activity chức năng sửa nhóm chat.



Hình 2.26. Biểu đồ Activity chức năng xóa nhóm chat.

2.2.3.7. Chức năng Quản lý thành viên nhóm chat.

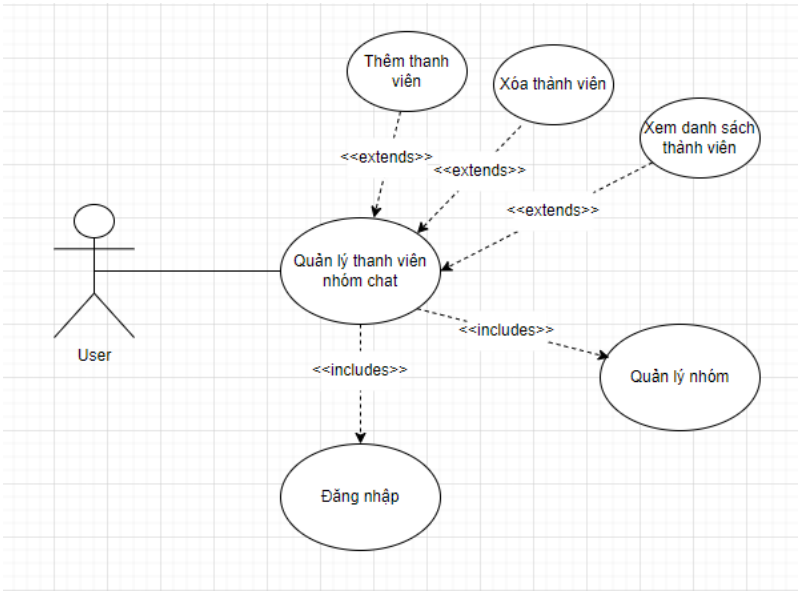
1. Đặc tả chức năng

Những người dùng khi đã đăng nhập có thể quản lý thông tin hồ sơ bài viết như thêm thêm thanh viên hoặc xóa thành viên.

Bảng 9. Đặc tả chức năng Quản lý thành viên nhóm chat.

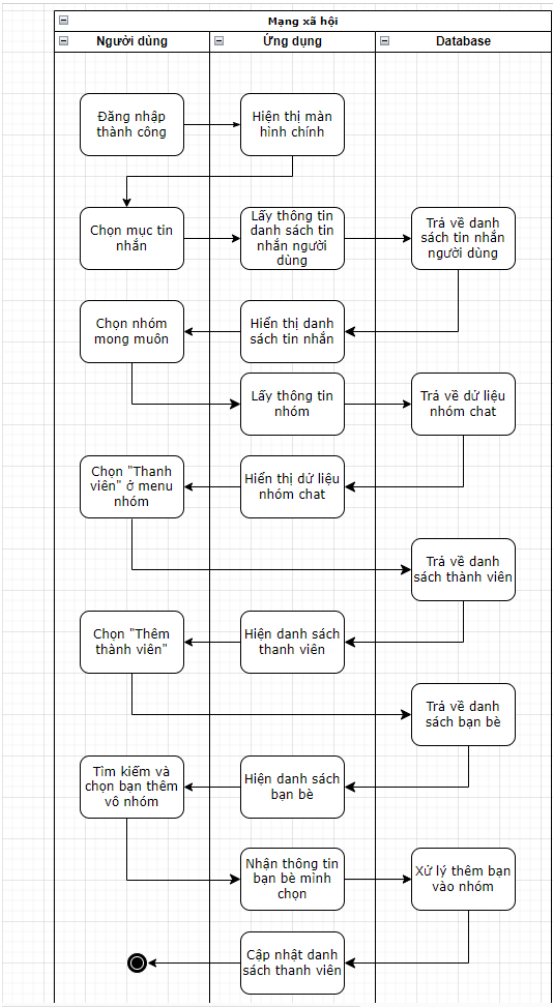
Summary	
UseCase Name	Quản lý thành viên nhóm chat.
Descriptions	Người dùng là thành viện nhóm sau khi đăng nhập có thể xem thanh viên nhóm , thêm thành viên. Người dùng là trưởng nhóm có thể xóa thành viên.
Actor	Người dùng
Priority	Không bắt buộc
Trigger	Mục Chat.
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng có thể quản lý nhóm.
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn mục chat. 3. Hệ thống tải các danh sách chat. 4. Người dùng chọn dấu 3 chấm trên thanh memu và thành viên. 5. Hệ thống tải danh sách thành viên nhóm. 6. Người dùng chọn thêm thành viên. 7. Hệ thống tải danh sách bạn bè. 8. Người dùng chọn bạn thêm vào nhóm.
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi.

2. Biểu đồ Usecase

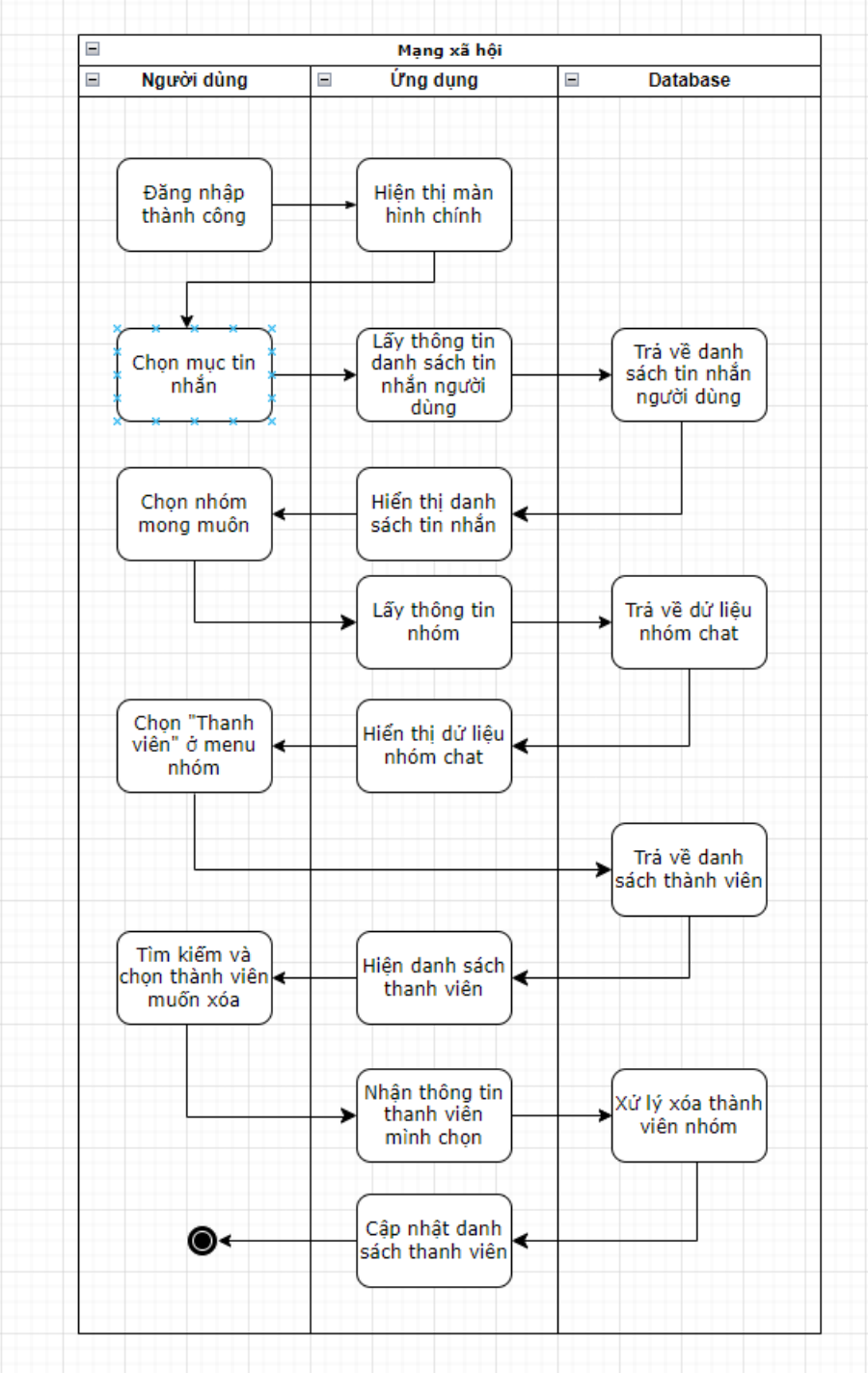


Hình 2.27. Biểu đồ Usecase chức năng Quản lý thành viên nhóm

3. Biểu đồ Activity



Hình 2.28. Biểu đồ Activity chức năng thêm thành viên nhóm



Hình 2.29. Biểu đồ Activity chức năng xóa thành viên nhóm.

2.2.3.8. Chức năng Tương tác bài viết và video ngắn.

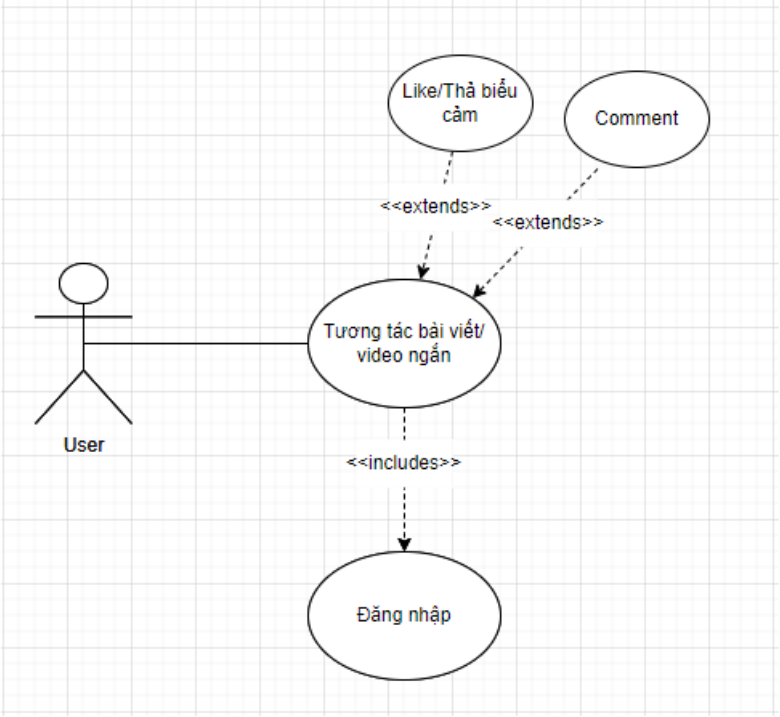
1. Đặc tả chức năng

Những người dùng khi đã đăng nhập có thể quản lý thông tin hồ sơ bài viết như thêm bài, xóa bài hoặc chỉnh sửa bài viết của mình

Bảng 10. Đặc tả chức năng Tương tác bài viết và video ngắn.

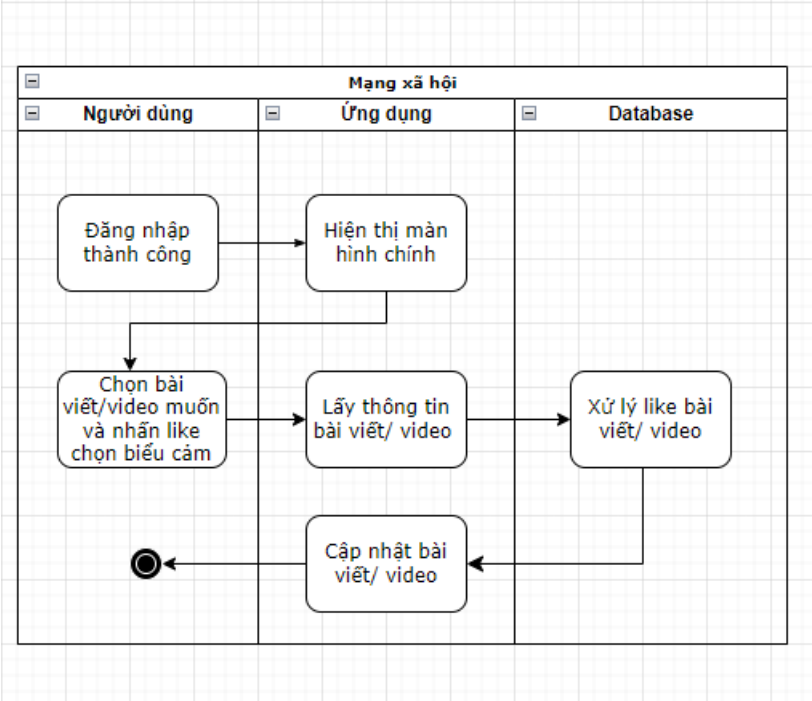
Summary	
UseCase Name	Tương tác bài viết và video ngắn
Descriptions	Người dùng có thể tương tác bài viết và video ngắn như like thả biểu cảm hay comment.
Actor	Người dùng
Priority	Không bắt buộc
Trigger	Mục bài viết
Pre-conditions	Đã đăng nhập Kết nối Internet ổn định
Post-conditions	Người dùng có thể tương tác bài viết.
Flows	
Basic Flow	1. Người dùng đăng nhập thành công 2. Người dùng chọn bài viết hay video ngắn. 3. Người like thả biểu cảm hoặc comment 4. Hệ thống cập nhật. 5. Hệ thống gửi thông báo đến người dùng đăng bài viết hoặc video ngắn đó.
Exception Flow	Có lỗi xảy ra, hệ thống hiển thị thông báo lỗi.

2. Biểu đồ Usecase

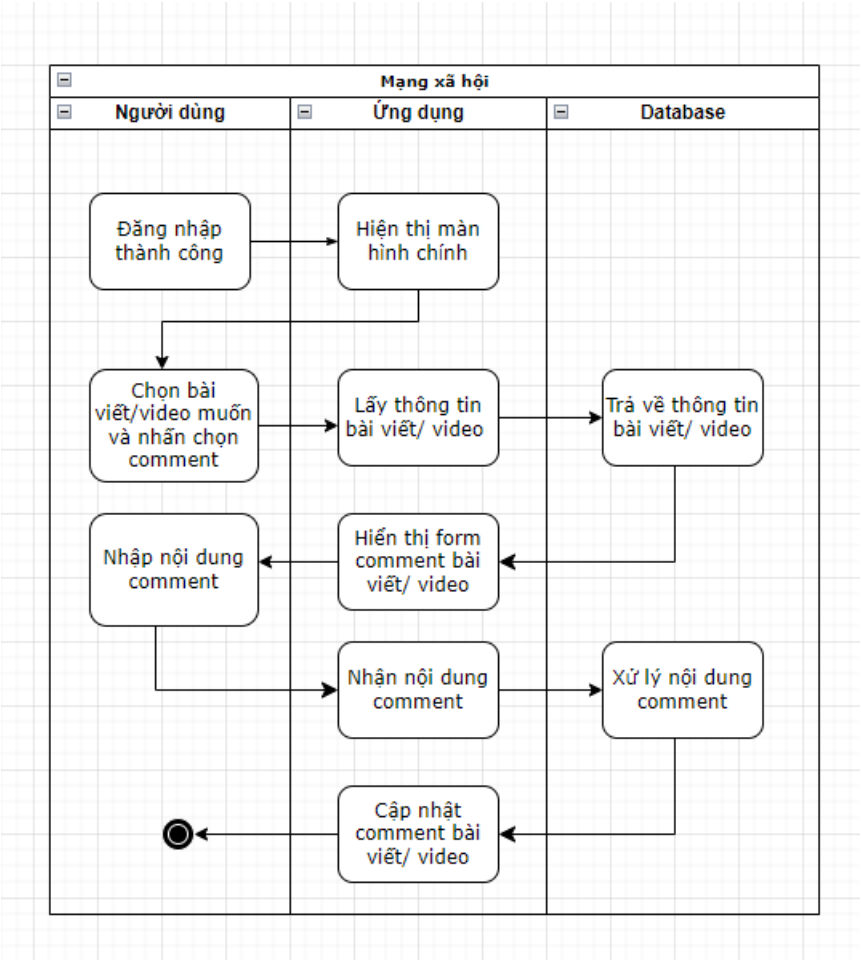


Hình 2.30. Biểu đồ Usecase chức năng Tương tác bài viết

3. Biểu đồ Activity



Hình 2.31. Biểu đồ Activity chức năng like bài viết và video ngắn.



Hình 2.32. Biểu đồ Activity chức năng comment bài viết và video ngắn.

2.3. Thiết kế Hệ thống

2.3.1. Thiết kế Giao diện

2.3.1.1. Ngôn ngữ thiết kế

Mặc định, Flutter hỗ trợ 2 ngôn ngữ thiết kế là Material và Cupertino.

Material là ngôn ngữ thiết kế do Google tạo ra, trong đó phổ biến trên các thiết bị Android và kể cả Web frontend.

Cupertino là ngôn ngữ thiết kế do đội ngũ Flutter phát triển, cho phép người lập trình tạo ra giao diện mang phong cách của những ứng dụng iOS.

Ứng dụng mạng xã hội sử dụng ngôn ngữ thiết kế thiên về bộ Material, vì bản thân tôi thấy rằng đây là một bộ giao diện có tính nhất quán cao, thoáng, và dễ sử dụng.

2.3.2. Thiết kế Cơ sở Dữ liệu

Cơ sở dữ liệu được sử dụng cho ứng dụng mạng xã hội là **Firebase Firestore**. Firestore là một Cơ sở dữ liệu không quan hệ (NoSQL). Nghĩa rằng giữa các bảng dữ liệu sẽ không có ràng buộc khóa chính – khóa ngoại.

Dữ liệu giữa các trường (dòng/bản ghi) có thể khác nhau kể cả khi chúng cùng thuộc một bảng. Điều này làm tăng tốc độ truy vấn lên đáng kể so với các cơ sở dữ liệu SQL như Microsoft SQL, MySQL… Cũng nhờ tốc độ truy vấn cao mà Firestore hỗ trợ cả realtime updating (cập nhật thời gian thực tới các client bất cứ khi nào dữ liệu trong Firestore có thay đổi).

Một số đặc điểm của Firestore:

- **Linh hoạt:** Dữ liệu trong Firestore được cấu trúc theo dạng các Collection (tương ứng với bảng trong SQL), bên trong các Collection chứa các Document (tương ứng với bản ghi/dòng trong SQL). Đặc biệt, Firestore cho phép "lồng Collection bên trong Document", có nghĩa rằng trong một bản ghi lại có thể chứa một "bảng khác"
- **Tốc độ:** Mặc định, các dữ liệu trong Firestore được đánh chỉ mục, điều này làm cho các câu truy vấn trong Firestore luôn giữ ở tốc độ nhanh. Bên cạnh đó Firestore cung cấp phương thức để truy vấn với nhiều điều kiện phức tạp, lồng nhau.
- **Cập nhật thời gian thực:** Cũng tương tự Firebase Realtime DB, Firestore cho phép cập nhật (đồng bộ) dữ liệu trên các thiết bị đã kết nối mạng theo thời gian thực (có nghĩa rằng độ trễ rất thấp). Bên cạnh đó Firestore cũng cho phép lấy dữ liệu theo yêu cầu (on demand) thay vì tự động đồng bộ
- **Hỗ trợ offline:** Firestore cung cấp cơ chế hoạt động offline (ngay cả khi thiết bị không có kết nối mạng). Có nghĩa ngay cả khi không có mạng, Firestore vẫn cho phép các hành động như đọc/ghi/cập nhật dữ liệu diễn ra, khi thiết bị được kết nối tới mạng Internet trở lại, Firestore sẽ tự động tải các thay đổi từ thiết bị lên đám mây
- **Tính scale cao:** Firestore là Cơ sở dữ liệu hiện đại nhất của Google, hỗ trợ việc chuyển vùng dữ liệu để đảm bảo tốc độ; đảm bảo tính nhất quán; hỗ trợ transaction cùng các thao tác hàng loạt (atomic transaction)
- **Bảo mật:** Với cơ chế bảo mật thông qua Security Rule, Firestore thậm chí có thể bảo vệ dữ liệu/phân quyền tới từng trường dữ liệu

(column) của một Document (bản ghi). Ví dụ điển hình là hệ thống cho phép thay đổi các trường thông tin khác nhưng không cho chỉnh sửa username.

Firebase security rules cho phép ai được quyền read, write data trong Cloud Firestore, Realtime Database, và Cloud Storage. Security rules được viết theo cú pháp tương tự javascript dưới dạng cấu trúc JSON.

- read: mô tả với điều kiện nào thì cho phép user đọc data bao gồm: get, list
- write: mô tả với điều kiện nào thì cho phép user ghi data bao gồm: create, update, delete.

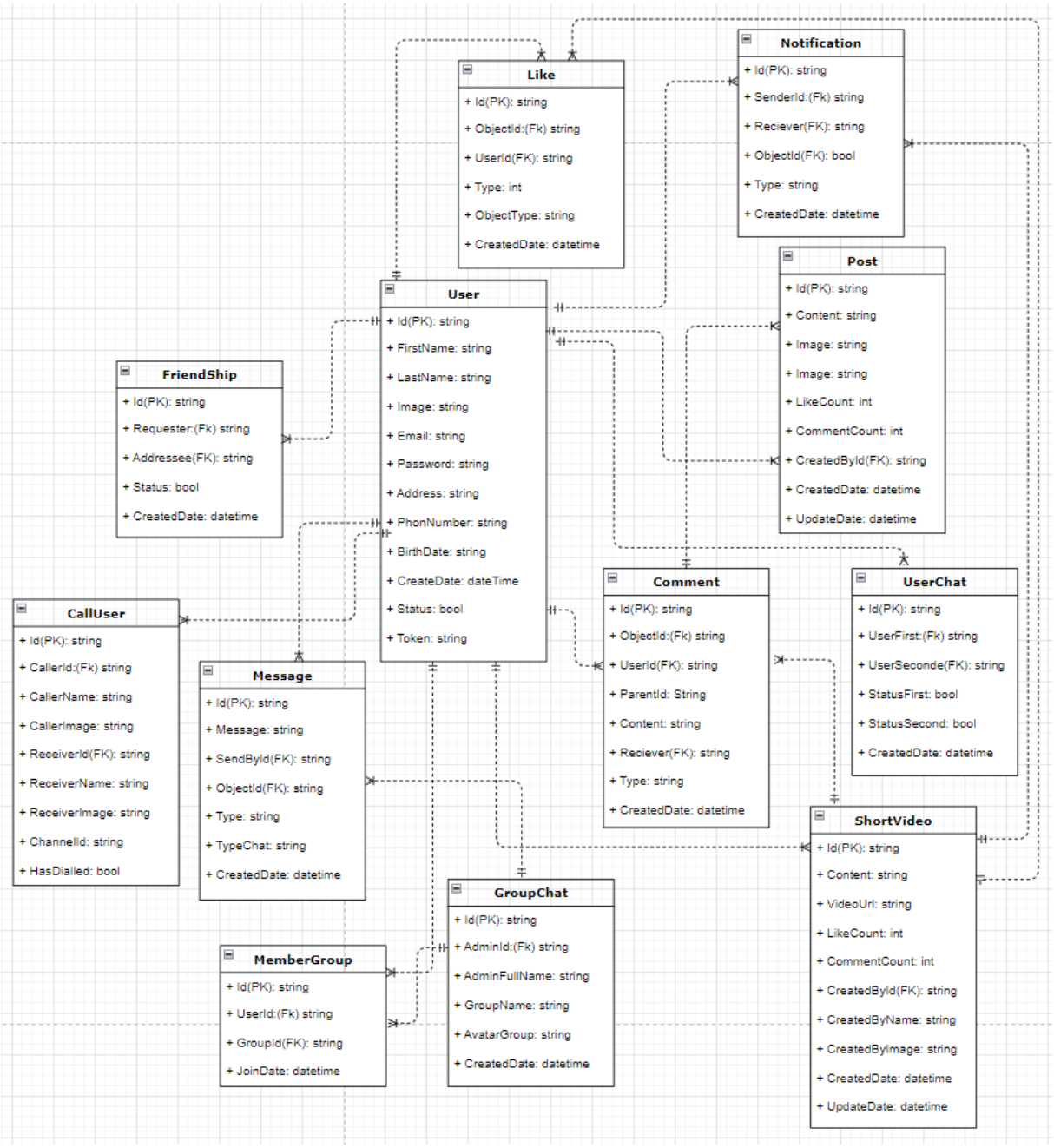
Các Securiry Rule được lưu trữ ở phía Firebase Console (đám mây), do đó mọi client (các ứng dụng phía Frontend) khi dùng tới dữ liệu đều phải tuân theo các quy định do Security Rule đặt ra.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurant/{restId} {
      // Allow the client to update a document only if that document doesn't
      // change the average_score or rating_count fields
      allow update: if (!request.resource.data.diff(resource.data).affectedKeys()
        .hasAny(['average_score', 'rating_count']));
    }
  }
}
```

Hình 2.33. Ví dụ Security Rule – Chặn cập nhật trên các trường nhất định

2.3.2.1. Cấu trúc các bảng dữ liệu

Các bảng dữ liệu trong hệ thống ứng dụng mạng xã hội được cấu hình như sau:



Hình 2.34. Biểu đồ ERD các bảng trong CSDL

2.3.2.2. Chi tiết các bản dữ liệu

- **Quản lý người dùng**

Bảng 11. Bảng cở sở dữ liệu người dùng

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	User	**ID**	String	Mã người dùng
2		Username	String	Tên tài khoản
3		Password	String	Mật khẩu người dùng
4		FirstName	String	Họ người dùng
5		LastName	String	Tên người dùng
6		Email	String	Email người dùng
7		Address	String	Địa chị cư trú người dùng
8		CreatedDate	Datetime	Ngày tạo tài khoản
9		Status	Bool	Trạng thái đăng nhập
10		Token	String	Mã thiết bị đăng nhập
11		PhoneNumber	String	Số điện thoại người dùng
12		Image	String	Hình đại diện
13		BirthDate	DateTime	Ngày sinh người dùng

- **Quản lý bài viết**

Bảng 12. Bảng cở sở dữ liệu bài viết

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Post	**ID**	String	Mã bài viết
2		Content	String	Nội dung bài viết
3		Image	String	Hình ảnh của bài viết
4		CreatedById	String	Người tạo bài viết
5		LikeCount	Int	Số lượng yêu thích bài viết
6		CommentCount	Int	Số lượng bình luận bài viết
7		CreatedDate	DateTime	Ngày tạo bài viết
8		UpdateDate	Datetime	Ngày cập nhật bài viết

❖ **Quản lý video ngắn**

Bảng 13. Bảng cở sở dữ liệu video ngắn

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Video	**ID**	String	Mã video
2		Content	String	Nội dung video
3		VideoỦRL	String	Đường dẫn video
4		CreatedById	String	Người tạo video
5		LikeCount	Int	Số lượng yêu thích video
6		CommentCount	Int	Số lượng bình luận video
7		CreatedDate	DateTime	Ngày tạo video
8		UpdateDate	Datetime	Ngày cập nhật video

❖ **Quản lý bạn bè**

Bảng 14. Bảng cở sở dữ liệu bạn bè

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	FriendShip	**ID**	String	Mã kết bạn
2		SendById	String	Mã người gửi
3		Receiver	String	Mã người nhận
4		Status	bool	Trạng thái bạn bè

❖ **Quản lý nhắn tin nhóm**

Bảng 15. Bảng cở sở dữ liệu nhắn tin nhóm.

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	GroupChat	**ID**	String	Mã nhóm chat
2		GroupName	String	Tên nhóm chat
3		AvatarGroup	String	Hình đại diện nhóm
4		AdminId	String	Mã người trưởng nhóm
5		AdminFullName	String	Tên người trưởng nhóm
6		CreatedDate	DateTime	Ngày tạo nhóm

❖ **Quản lý nhắn tin cá nhân**

Bảng 16. Bảng cở sở dữ liệu nhắn tin cá nhân.

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	UserChat	**ID**	String	Mã thành viên nhóm
2		UserFristById	String	Mã người dùng thứ nhất
3		UserFristByName	String	Tên người dùng thứ nhất
4		UserFristByImage	String	Ảnh người dùng thứ nhất
5		UserSecondById	String	Mã người dùng thứ hai
6		UserSecondByName	String	Tên người dùng thứ hai
7		UserSecondByImage	String	Ảnh người dùng thứ hai
8		StatusUserFirst	Bool	Trạng thái người dùng thứ nhất
9		StatusUserSecond	Bool	Trạng thái người dùng thứ hai
10		CreateDate	DateTime	Ngày tạo

❖ **Quản lý tin nhắn**

Bảng 17. Bảng cở sở dữ liệu tin nhắn.

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Message	**ID**	String	Mã nhắn tin cá nhân
2		SendById	String	Mã người gửi tin
3		SendByName	String	Tên người gửi tin
4		ObjectId	String	Mã đối tượng chat(gửi nhóm hoặc cá nhân)
5		Type	String	Loại tin nhắn gửi(văn bản hoặc hình ảnh)
6		TypeChat	String	Gửi tin nhắn nhóm hay cá nhân
7		CreateDate	DateTime	Ngày gửi tin nhắn

❖ **Quản lý thành viên nhóm chat**

Bảng 18. Bảng cở sở dữ liệu thành viên nhóm chat.

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	MemberGroup	**ID**	String	Mã thành viên nhóm
2		GroupId	String	Mã nhóm chat
3		UserId	String	Mã người dùng
4		JoinDate	DateTime	Ngày tham gia nhóm

❖ **Quản lý gọi video cá nhân**

Bảng 19. Bảng cở sở dữ liệu gọi video cá nhân.

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	CallVideo	**ID**	String	Mã gọi video
2		CallerId	String	Mã người gọi
3		CallName	String	Tên người gọi
4		CallerImage	String	Hình đại diện người gọi
5		ReceiverId	String	Mã người nhận
6		ReceiverName	String	Tên người nhận
7		ReceiverImage	String	Hình đại diện người nhận
7		ChannelId	DateTime	Mã phận biệt cuộc gọi
8		HasDialled	Bool	Trạng thái gọi

❖ **Quản lý yêu thích bài viết và video**

Bảng 20. Bảng cơ sở dữ liệu yêu thích bài viết và video

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Like	**ID**	String	Mã yêu thích
2		ObjectId	String	Mã bài viết hoặc video
3		UserId	String	Mã người dùng
4		Type	int	Loại cảm xúc
5		ObjectType	String	Phân biệt đối tượng(bài viết hay video)
6		CreatedDate	DateTime	Ngày tạo

❖ **Quản lý bình luận viết và video**

Bảng 21. Bảng cơ sở dữ liệu bình luận bài viết và video

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Comment	**ID**	String	Mã yêu thích
2		ObjectId	String	Mã bài viết hoặc video
3		UserId	String	Mã người dùng
4		ReceiverId	String	Mã người nhận bình luận
5		Content	String	Nội dung bình luận
6		ParentId	String	Phản hồi bình luận cha nào
5		ObjectType	String	Phân biệt đối tượng(bài viết hay video)
6		CreatedDate	DateTime	Ngày tạo

❖ **Quản lý thông báo**

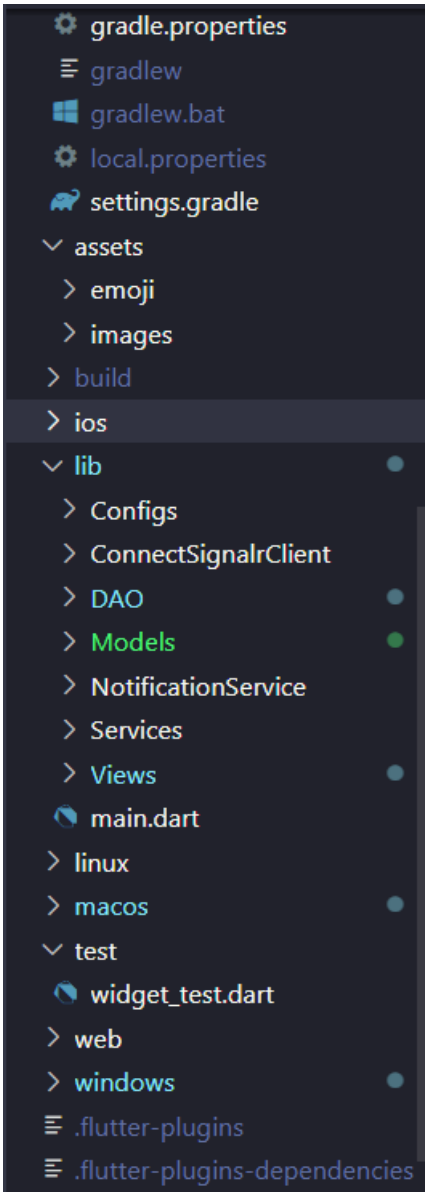
Bảng 22. Bảng cơ sở dữ liệu thông báo

STT	Tên bảng	Tên cột	Kiểu	Mục đích
1	Notification	**ID**	String	Mã thông báo
2		ObjectId	String	Mã đối tượng(bài viết hoặc video)
3		SenderId	String	Mã người gửi
4		ReceiverId	String	Mã người nhận
5		Content	String	Nội dung thông báo
6		CreatedDate	DateTime	Ngày tạo

CHƯƠNG 3. XÂY DỰNG PHẦN MỀM

3.1. Cấu trúc dự án

Cấu trúc các thư mục trong dự án như sau:



Hình 3.1. Cấu trúc thư mục của dự án

Chú thích nội dung các thư mục:

1. **android**: code của ứng dụng Android, thư mục này được Flutter tự động tạo ra khi tạo mới dự án
2. **build**: thư mục chứa các file đầu ra để cài đặt lên các máy thật hoặc xuất bản lên Google Play Store/Apple Store (định dạng .apk/.aab/.ipa)
3. **ios**: code của ứng dụng iOS, thư mục này được Flutter tự động tạo ra khi tạo mới dự án

4. **lib:** chứa toàn bộ code Flutter của dự án

4.1. **assets:** chứa các tài nguyên tĩnh (static) của dự án, ví dụ như hình ảnh, fonts, icon…

4.2. **app:** chứa các code cấu hình chung (global) cho dự án, như cấu hình màu sắc, giao diện (themes), các file ngôn ngữ…

4.3. **base:** các class khuôn mẫu (blueprints) để các class khác kế thừa, ví dụ: BaseScreen (class mẫu cho các màn hình), BaseController (class xử lý logic của màn hình), BaseFirestoreModel (class cha dành cho việc ánh xạ dữ liệu từ Firestore thành các class nhất định, ví dụ như **Invoice extends BaseFirestoreModel**)

4.4. **dao:** các funcation tiện ích, tương tác với CSDL.

4.5. **models:** các class ánh xạ dữ liệu từ Firestore, ví dụ: User, Message, GroupChat, Post, Comment, Notification…

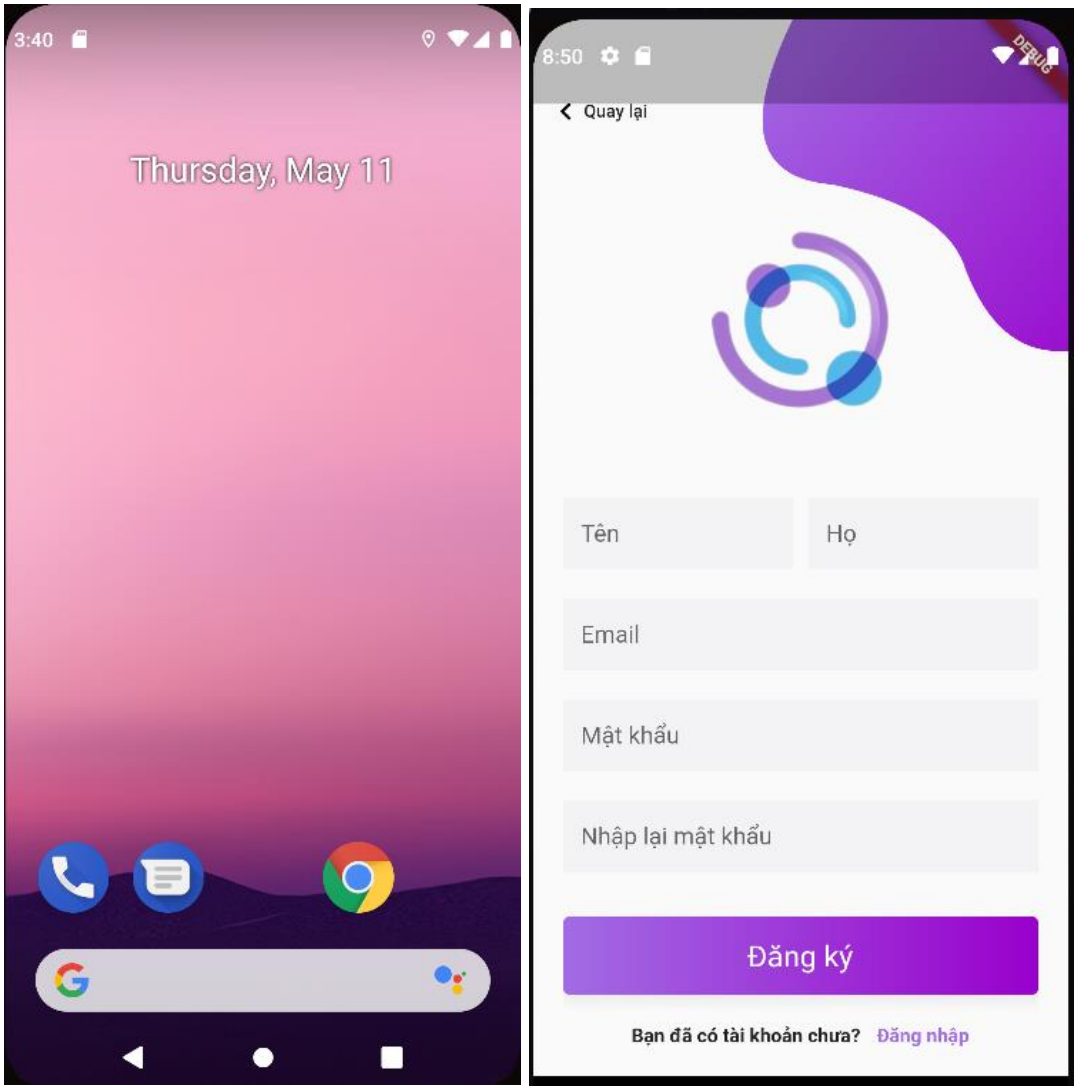
4.6. **views:** các màn hình/chức năng. Mỗi màn hình là một thư mục chứa, trong đó bao gồm view, controller, widget con

3.2. Các màn hình chức năng

3.2.1. Đăng ký

Để có thể đăng nhập vào ứng dụng màn hình chình thì trước tiến cần phải có tài khoản, để tạo tài khoản chọn đăng ký vào màn hình đăng ký. Với việc sử dụng dịch vụ Firebase Authentication, ứng dụng này sử dụng phương thức đăng ký thông qua email để tăng tính bảo mật .

Bước 1: Khởi động ứng dụng nếu chưa có tài khoản tài khoản người dùng chọn đăng ký tài khoản để chuyển sang màn hình đăng ký tài khoản.



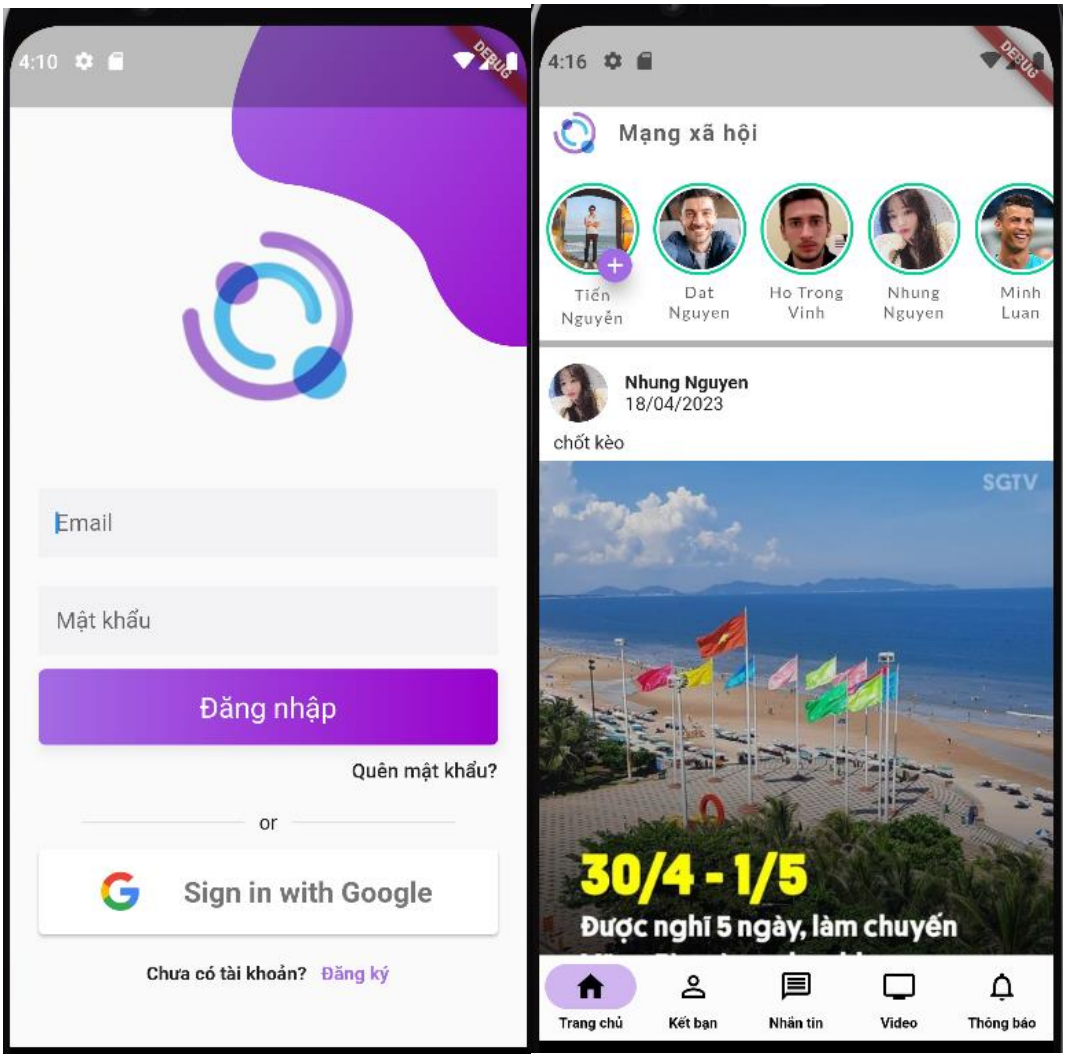
Hình 3.2. Màn hình đăng ký.

Bước 2: Nhập họ tên, email và password để đăng ký tài khoản và chọn nút đăng ký.

3.2.2. Đăng nhập

Để sử dụng được các chức năng khác có trong ứng dụng, trước hết người dùng cần định danh bản thân bằng việc đăng nhập. Ứng dụng này sử dụng phương thức đăng nhập qua địa chỉ emai để tăng tính bảo mật .

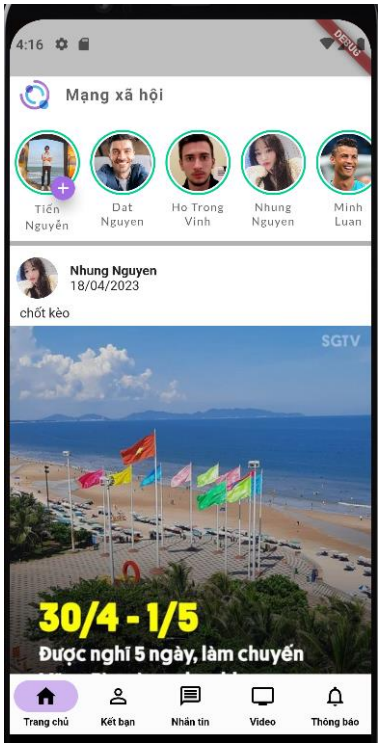
Khởi động ứng dụng khi chưa đăng nhập tài khoản nào ứng dụng chuyển đến màn hình đăng nhập, người dùng nhập email, password tài khoản mình để vào màn hình chính ứng dụng.



Hình 3.3. Màn hình đăng nhập.

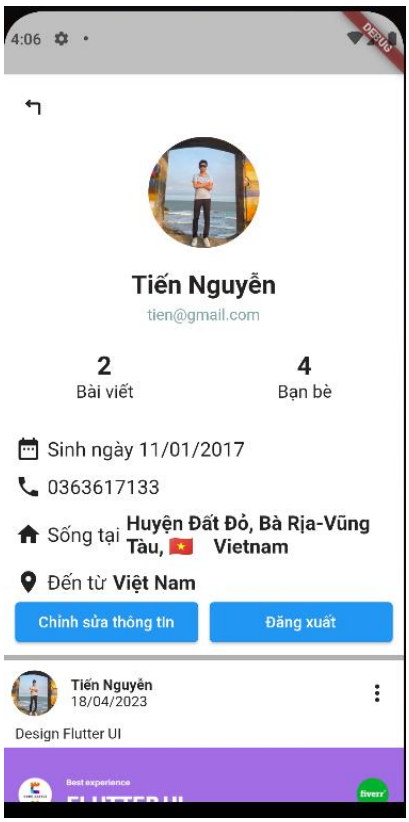
3.2.3. Chức năng Quản lý hồ sơ cá nhân

Người dùng khi nhập vào có thể quản lý thông tin cá nhân của mình như xem hồ sơ của tài khoản của mình và chỉnh sửa thông tin cá nhân.



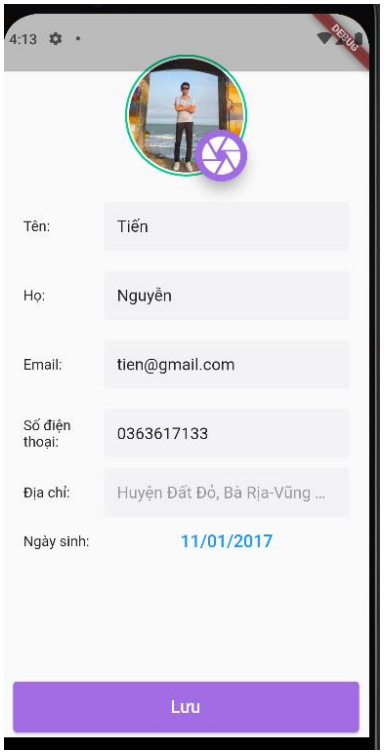
Hình 3.4. Màn hình chính ứng dụng.

Bước 1: Người dùng cần chọn vào avatar cá nhân, sau đó màn hình chuyển sang trang cá nhân hiện thị thông tin cá nhân của người dùng.



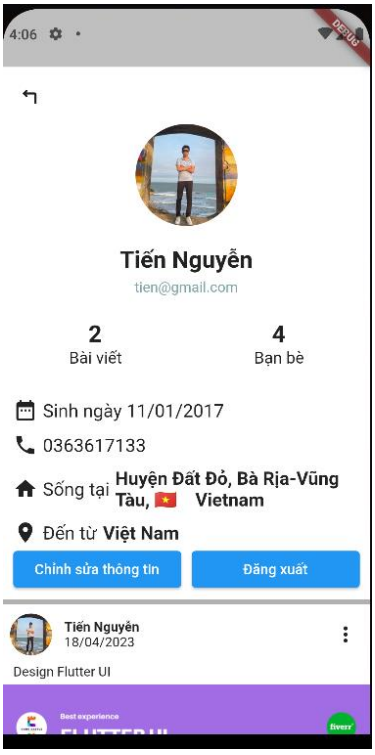
Hình 3.5. Màn hình hồ sơ cá nhân.

Bước 2: Chọn nút chỉnh sửa thông tin để chuyển sang màn hình chỉnh sửa



Hình 3.6. Màn hình chỉnh sửa thông tin cá nhân

Bước 3: Thực hiện thao tác chỉnh sửa thông tin (có thể thay đổi ảnh đại diện) và chọn nút lưu để thay đổi và chuyển về màn hình Hồ sơ cá nhân.

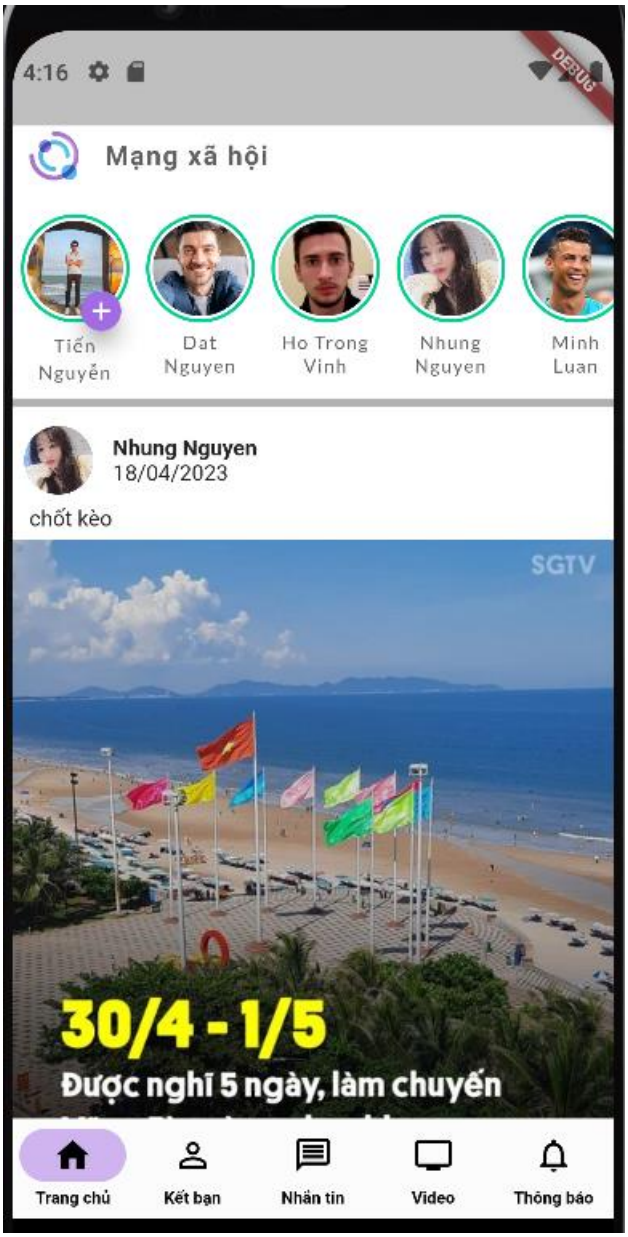


Hình 3.7. Màn hình hồ sơ cá nhân.

3.2.4. Quản lý bài viết

Trong ứng dụng mạng xã hội không thể không có chức năng Quản lý bài viết, đây là chức năng giúp mọi người nắm bắt được nhiều thông tin, tin tức trong cuộc sống , có thể chia sẽ giới thiệu bản thân cuộc sống hằng ngày qua những bài viết.

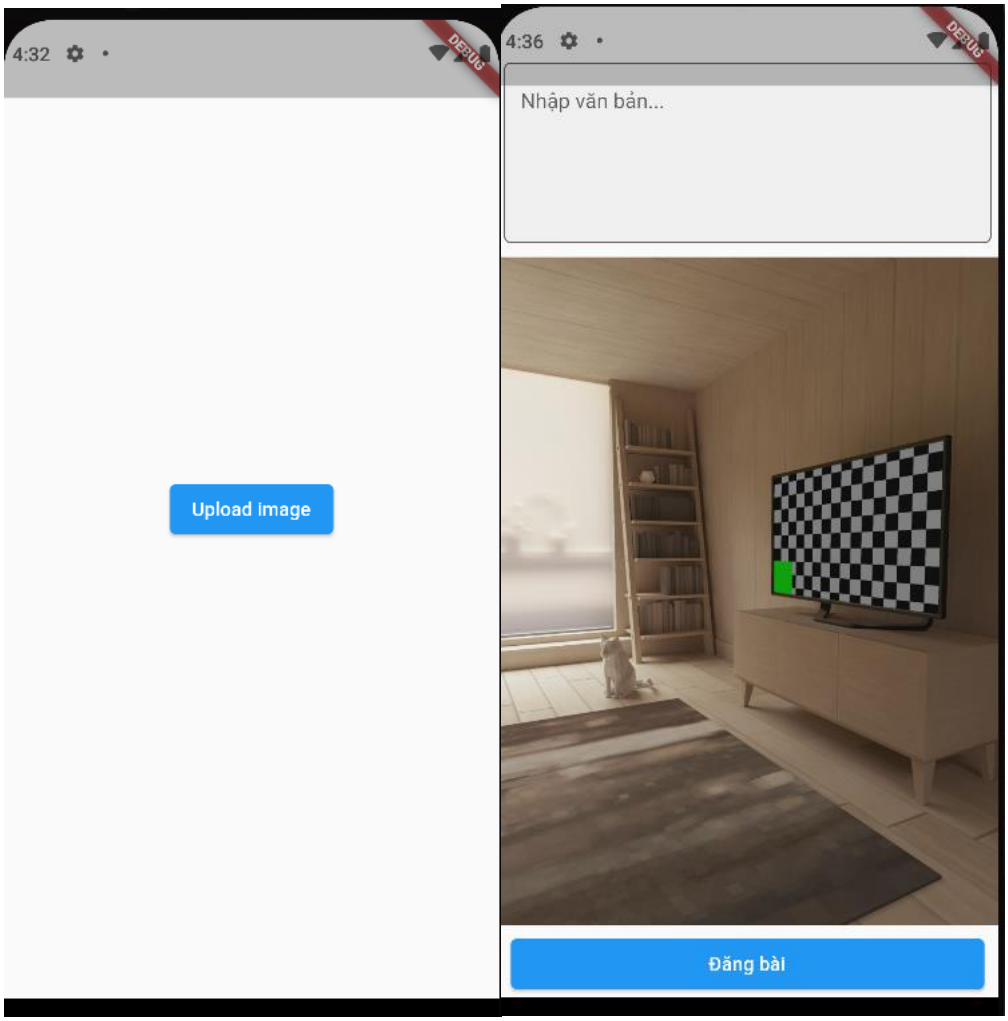
Người dùng xem được bài viết của mình và bạn bè tại màn hình chính của ứng dụng.



Hình 3.8. Màn hình chính ứng dụng

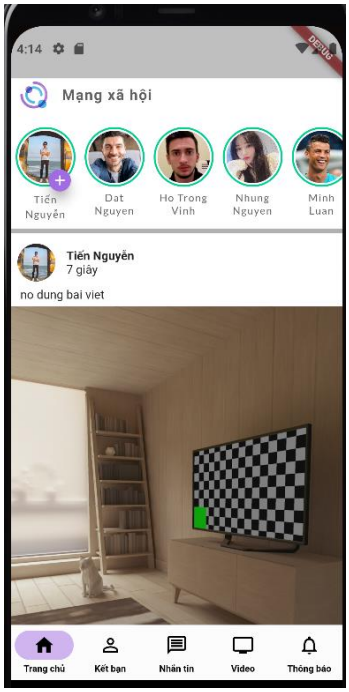
❖ **Thêm bài viết**

Bước 1: Tại màn hình chính người dùng chọn bào biểu tượng dấu + tại avatar cá nhấn để chuyển sang màn hình để thêm bài viết. Người dùng chọn ảnh để đăng bài và chuyển sang màn hình để nhập nội dung bài viết



Hình 3.9. Màn hình tạo nội dung bài viết

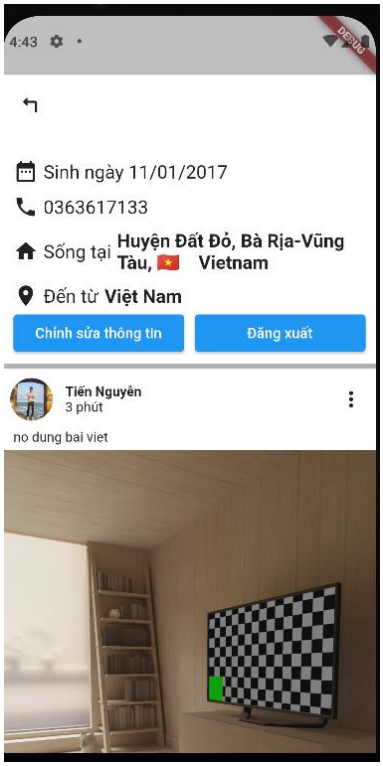
Bước 2: Người dùng nhập nội dung bài viết mình muốn tạo và nhấn lưu để thêm bài viết của mình vào danh sách bài viết và chuyển sang màn hình chính.



Hình 3.10. Màn hình chính ứng dụng

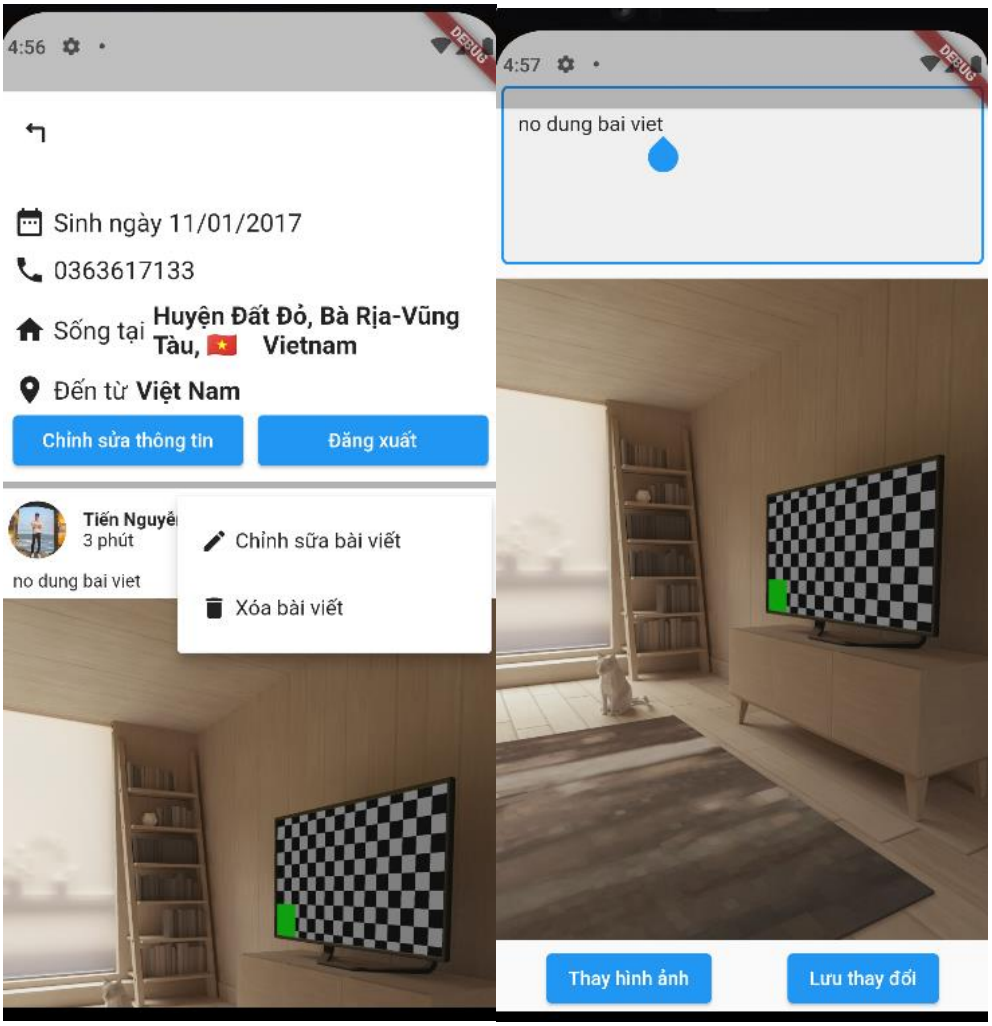
❖ **Chỉnh sửa bài viết**

Bước 1: Tại màn hình chính người dùng chọn avatar cá nhân để vào màn hình hồ sơ cá nhân. Ở đây có danh sách bài viết của cá nhân.



Hình 3.11. Màn hình chính ứng dụng

Bước 2: Người dùng chọn vào memu bên phải bài viết muốn sửa và chọn "chỉnh sửa bài viết" để chuyển sang màn hình chỉnh sửa bài viết.

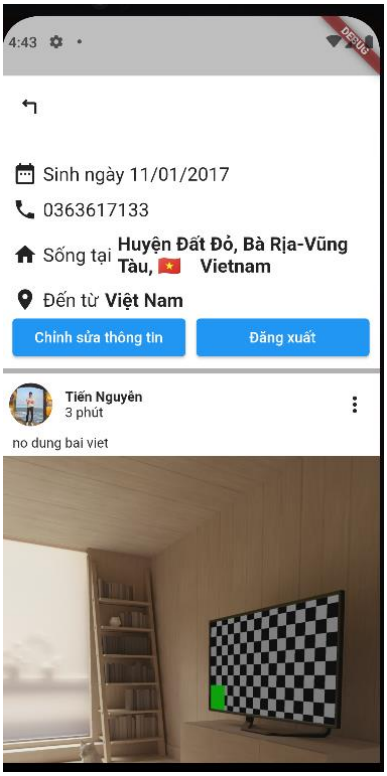


Hình 3.12. Màn hình chỉnh sửa bài viết

Bước 3: Người dùng thực hiện chỉnh sửa nội dung bài viết chỉnh sửa ảnh v à nhấn nút lưu để chỉnh sửa bài viết.

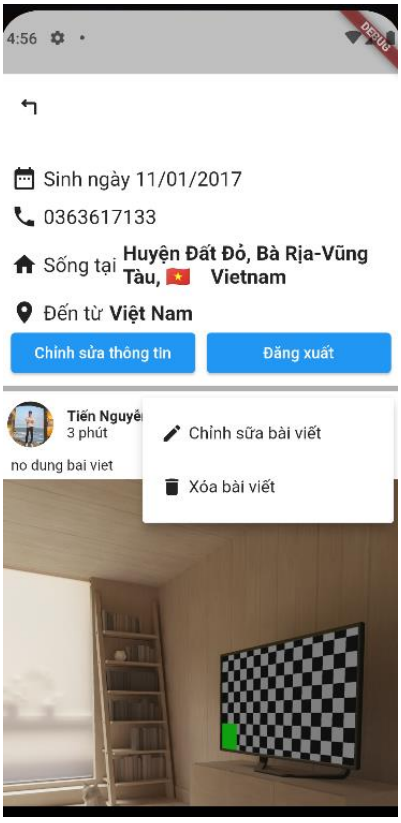
❖ **Xóa bài viết**

Bước 1: Tại màn hình chính người dùng chọn avatar cá nhân để vào màn hình hồ sơ cá nhân. Ở đây có danh sách bài viết của cá nhân.



Hình 3.13. Màn hình hồ sơ cá nhân

Bước 2: Người dùng chọn vào memu bên phải bài viết muốn xóa và chọn "Xóa" để xóa bài viết đó.



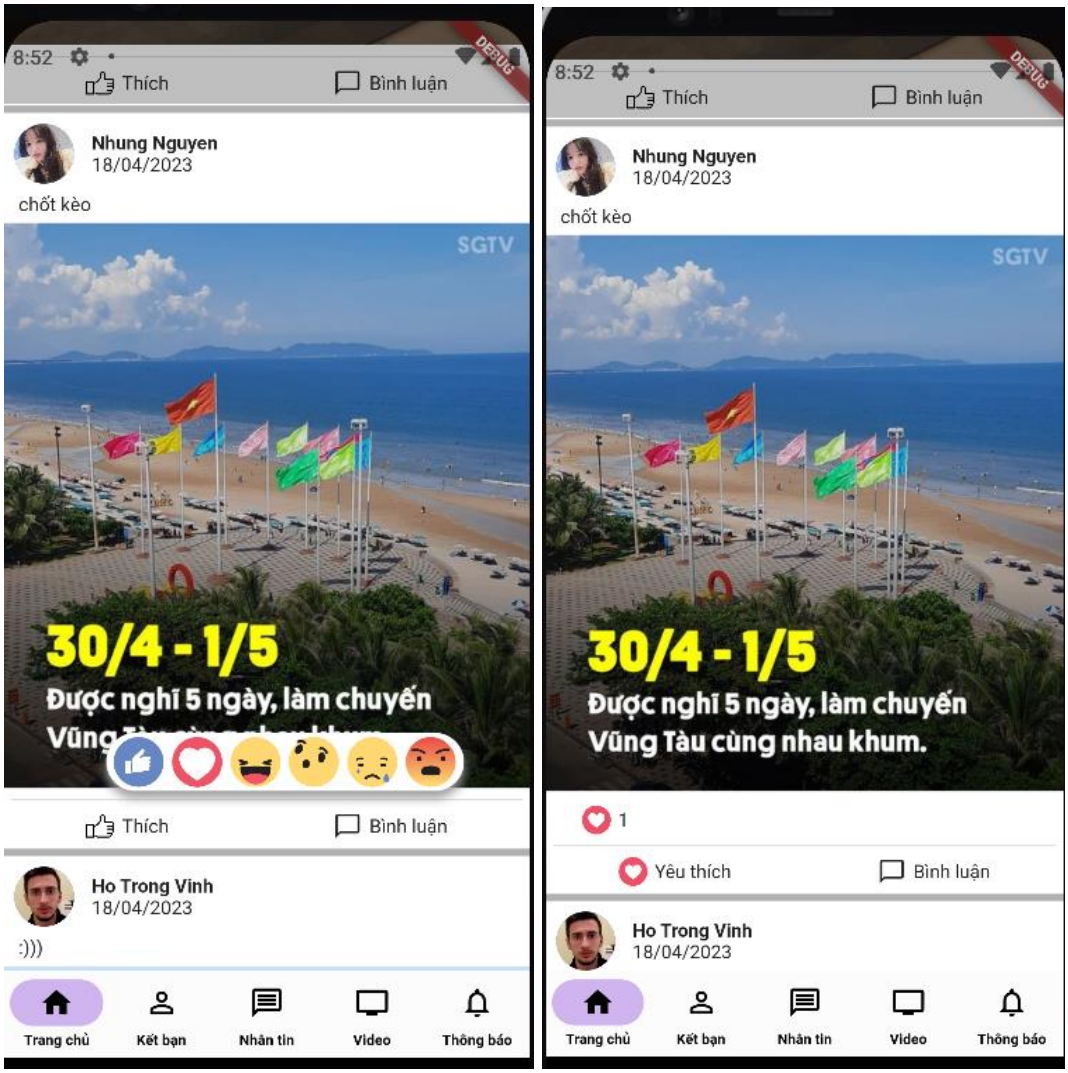
Hình 3.14. Màn hình hồ sơ cá nhân

3.2.5. Tương Tác bài viết/ video ngắn

Người dùng có thể tương tác bài viết hay vieo ngắn bằng cách like thả biểu cảm hoặc comment để nói lên cảm nghĩ của mình.

❖ **Like bài viết**

Bước 1: Trên màn hình chính, người dùng chọn bài viết mình muốn, nhấn giữ nút like và chọn biểu cảm mình muốn.

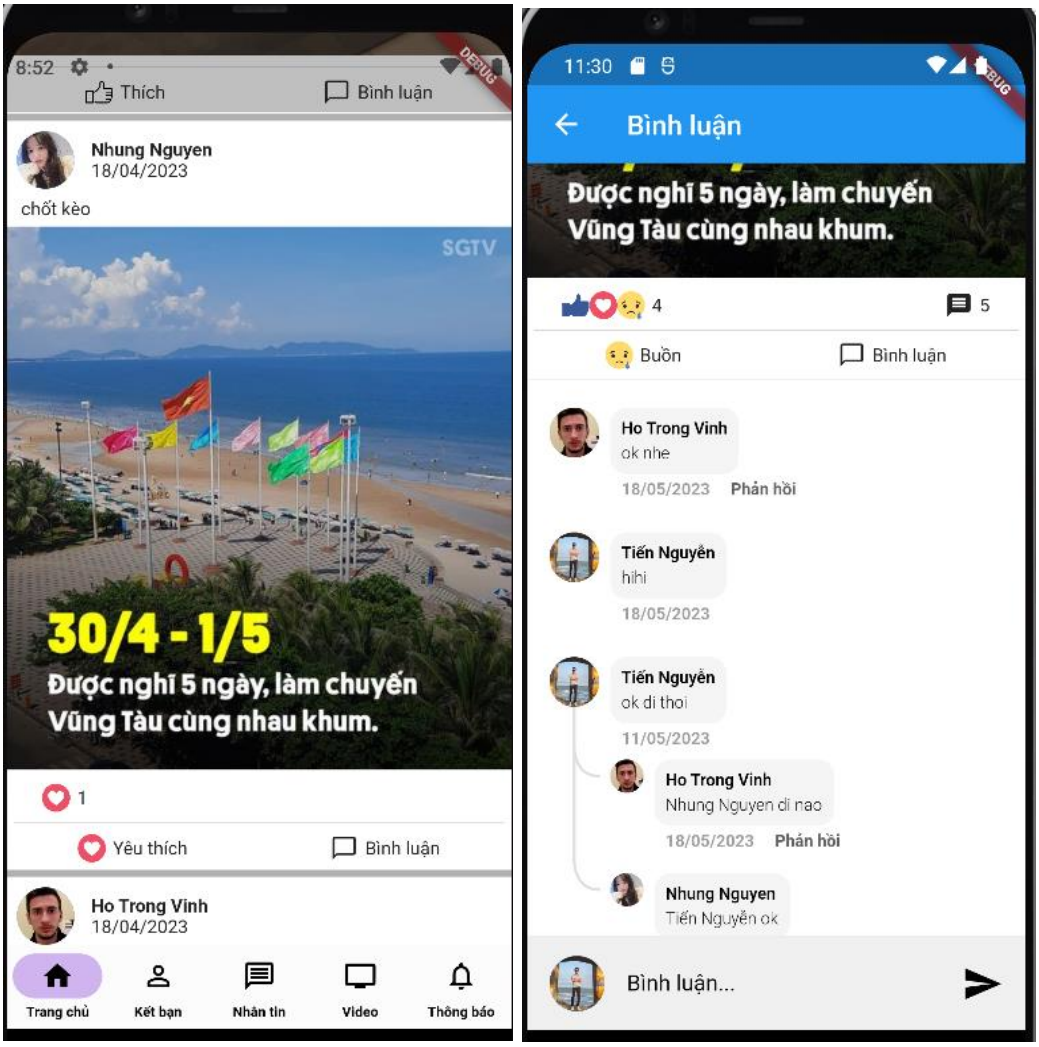


Hình 3.15. Màn hình chính ứng dụng

Bước 2: Bạn có thể hủy like bằng cách thực hiện nhấn lại nút like.

❖ **Comment bài viết**

Bước 1: Trên màn hình chính, người dùng chọn bài viết mình muốn và nhận chọn bình luận để chuyển sang màn hình bình luận bài viết đó.

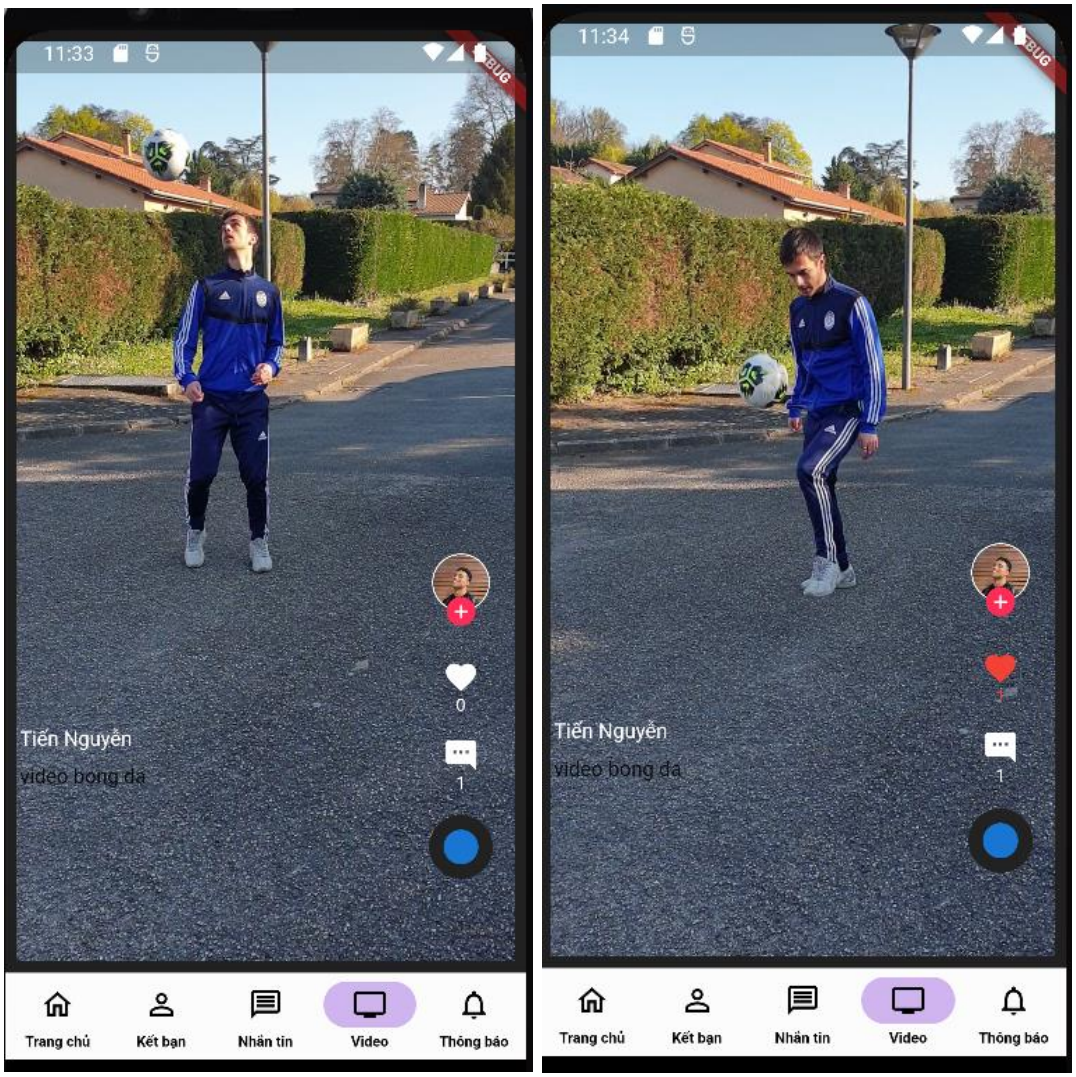


Hình 3.16. Màn hình chính và màn hình comment.

Bước 2: Nhập nội dung bình luận và gửi để bình luận bài viết.

❖ **Like video ngắn**

Bước 1: Trên màn hình chình, người dùng chọn mục video sau đó chọn video mình muốn, nhấn biểu tượng yêu thích để yêu thích video đó .

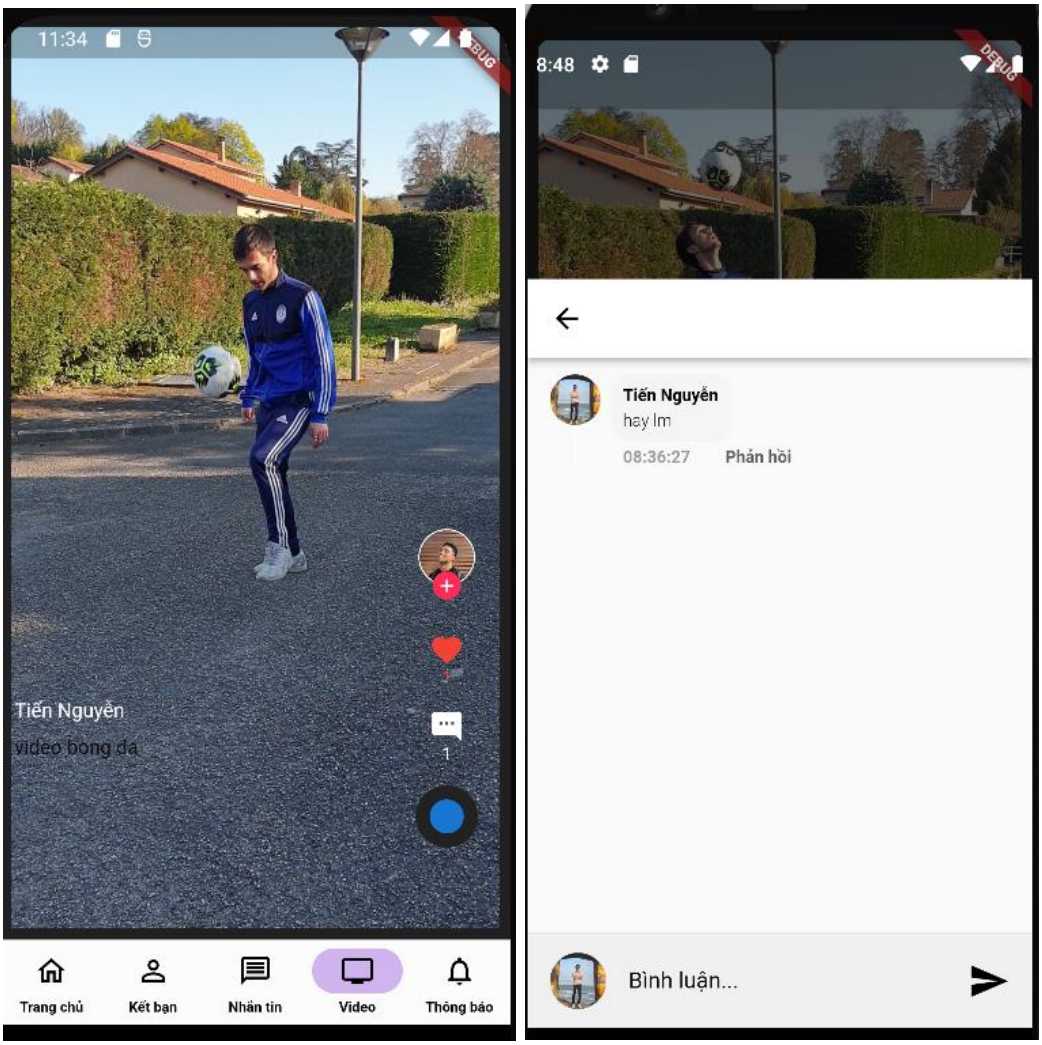


Hình 3.17. Màn hình video

Bước 2: Bạn có thể hủy like bằng cách thực hiện nhấn lại nút like.

❖ **Comment bài viết**

Bước 1: Trên màn hình chính, người dùng chọn mục video, chọn video mình muốn và nhận chọn bình luận để hiển thị màn hình bình luận video đó.



Hình 3.18. Màn hình video và màn hình comment.

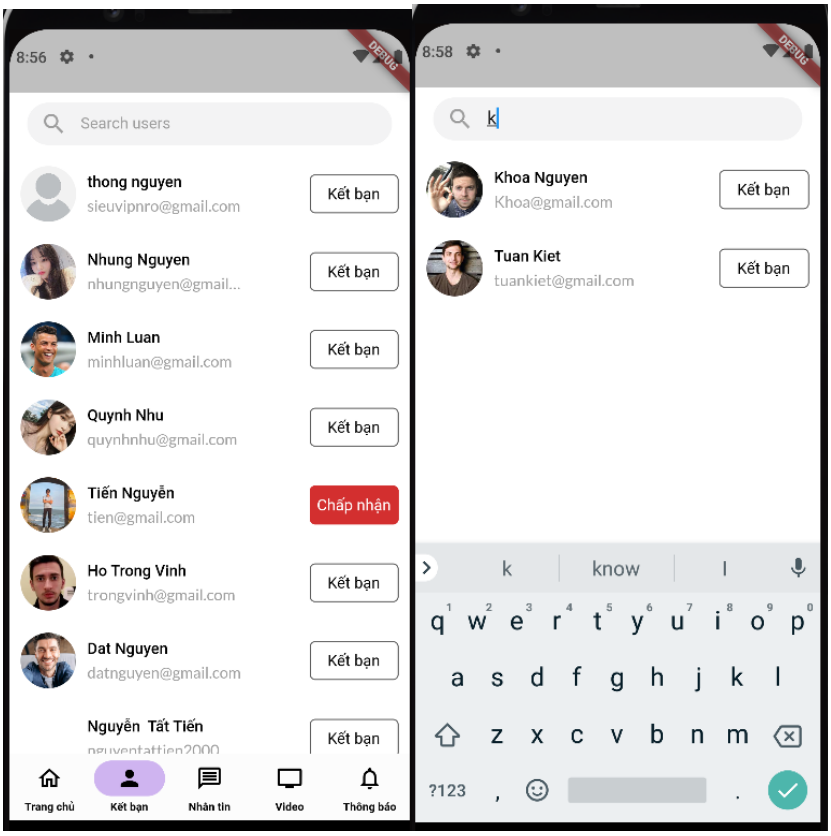
Bước 2: Nhập nội dung bình luận và gửi để bình luận video mình thích.

3.2.6. Quản lý bạn bè

Kết bạn giúp ta có thêm bạn mới gán kết bạn bè với nhau là có thể trao đổi trò chuyện trực tuyến dễ dàng và thuận tiện hơn.

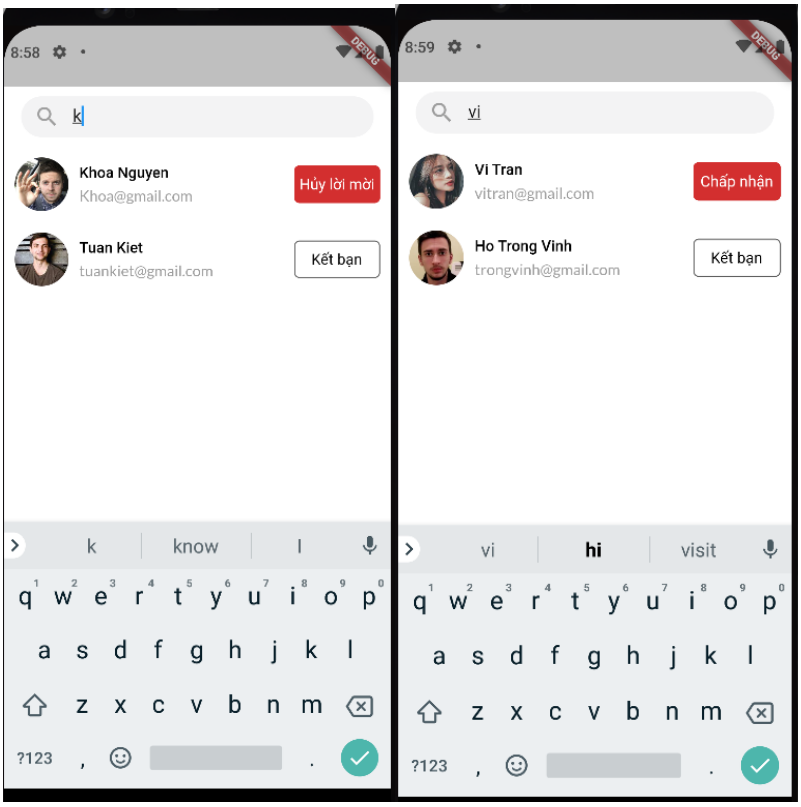
❖ **Kết bạn**

Bước 1: Tại màn hình chính chọn mục kết bạn và tìm kiếm bạn bè bạn muốn kết bạn.



Hình 3.19. Màn kết bạn của ứng dụng.

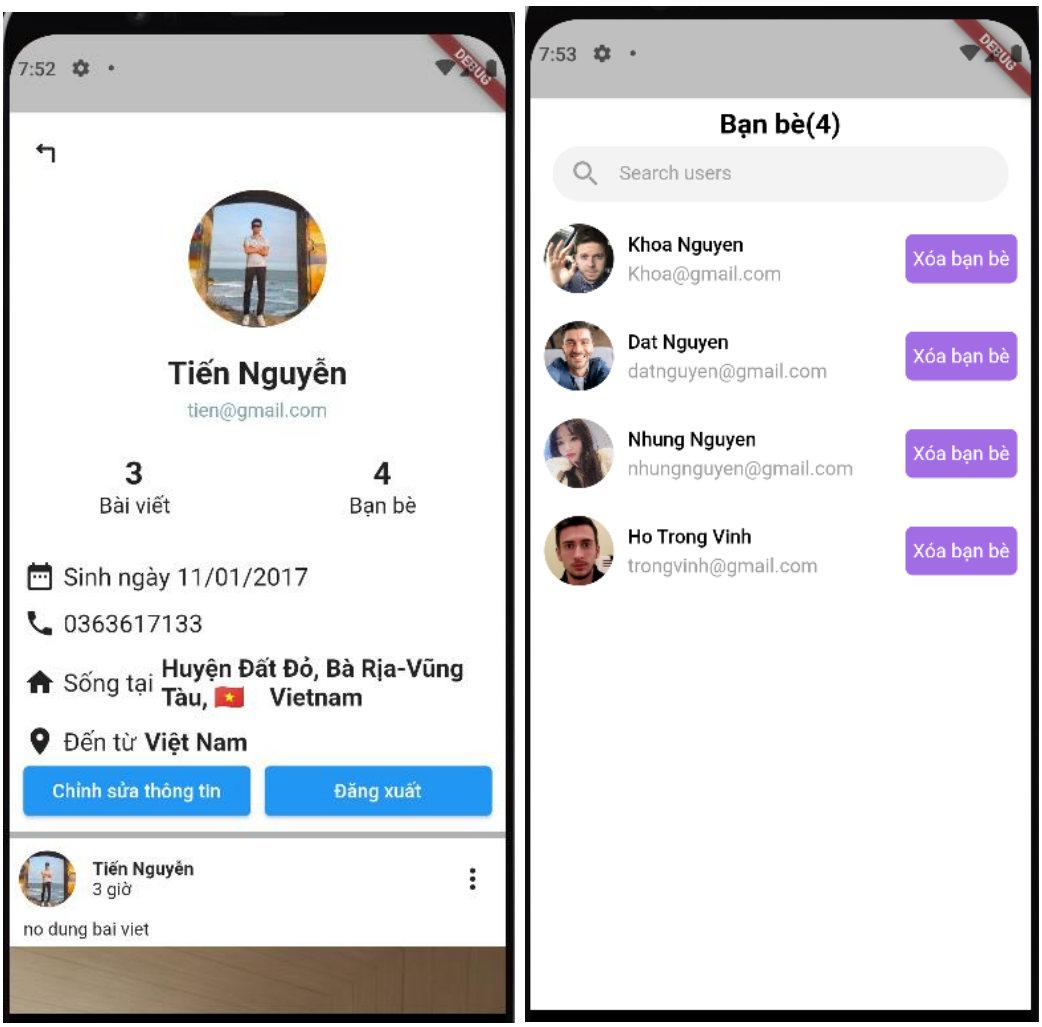
Bước 2: Nhấn người nhấn nút "Kết bạn" để gửi lời mời và đợi bạn bè đồng ý.



Hình 3.18. Màn hình kết bạn của ứng dụng

❖ **Xóa bạn bè**

Bước 1: Tài màn hình chính chọn avatar cá nhân vào hồ sơ cá nhận sau đó nhấn bạn bè để chuyển vào màn hình hiển thị danh sách bạn bè của người dùng.



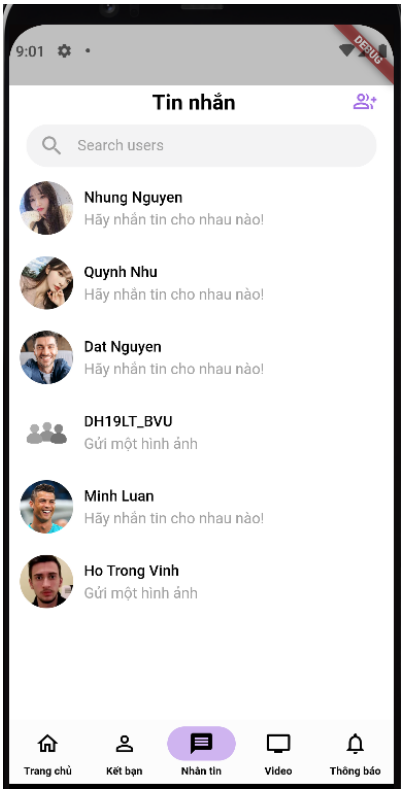
Hình 3.20. Màn hình cá nhân và danh sách bạn bè

Bước 2: Tại đây, người quản dùng tìm chọn bạn bè muốn xóa và nhấn nút xóa bạn bè.

3.2.7. Nhắn tin

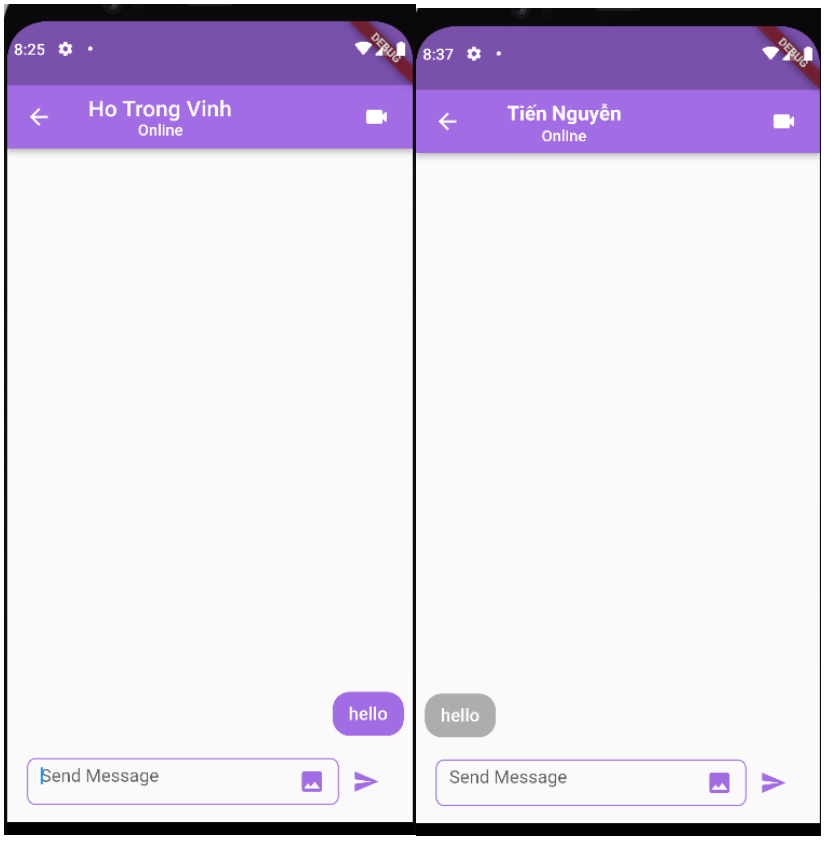
Nhắn tin trực tuyến giúp rút ngắn thời gian trò chuyện, có thể trò chuyện mọi lúc mọi nơi, tiếm kiệm thời gian, thuận tiện, dễ dàng hơn, làm gắn kết bạn bè mối quan hệ tốt đẹp hơn.

Bước 1: Tại màn hình chính, người dùng chọn mục nhắn tin để chuyển sang màn hình hiển thị danh sách cuộc trò chuyện.



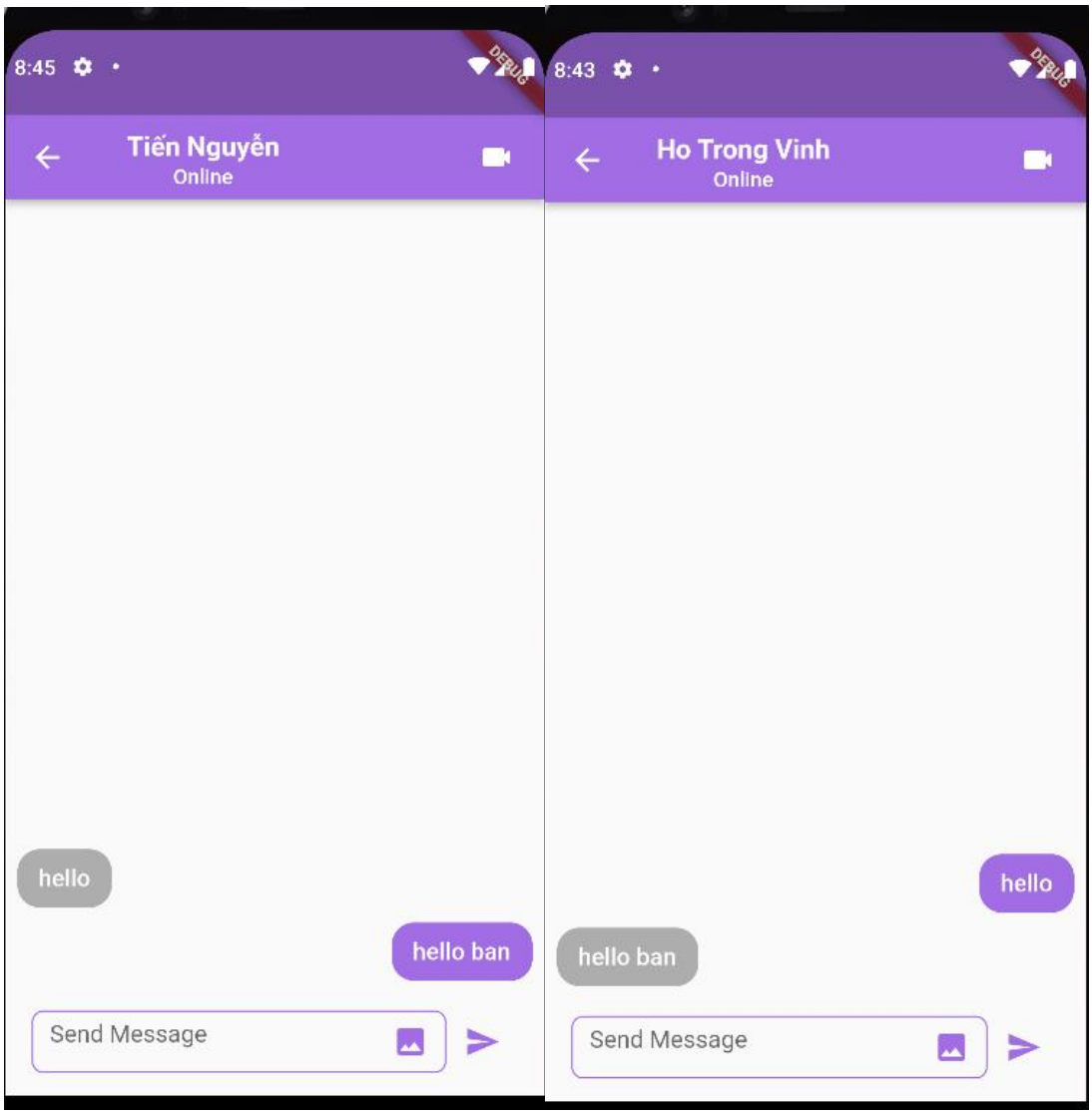
Hình 3.21. Màn hình danh sách cuộc trò chuyện.

Bước 2: Người dùng chọn cuộc trò chuyện mình muốn để hiển thị chi tiết cuộc trò chuyện đó.



Hình 3.22. Màn hình nhắn tin.

Bước 3: Người dùng nhập nội dung tin nhắn muốn giử và nhấn gửi, tin nhắn sẽ được nhận ngay từ người nhận.



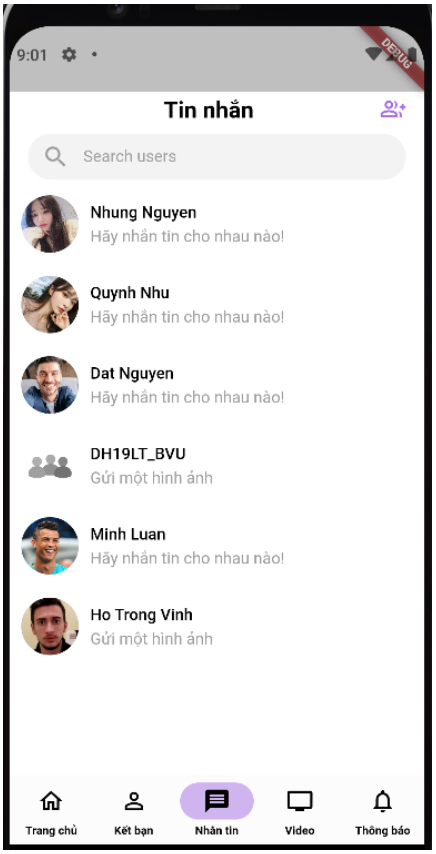
Hình 3.23. Màn hình nhắn tin.

3.2.8. Quản lý nhóm chat.

Ứng dụng mạng xã hội không tương tác trò chuyền giữa hai người với nhau mà còn có thể trò chuyện một nhóm gồm nhiều người giúp thuận tiện trong việc trò chuyện làm gắn kết với nhau hơn.

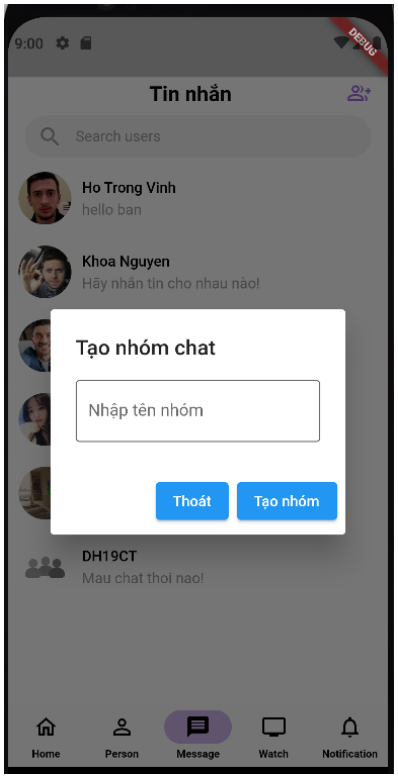
❖ **Tạo nhóm chat**

Bước 1: Tại màn hình chính, người dùng chọn mục nhắn tin để chuyển sang màn hình hiển thị danh sách cuộc trò chuyện.



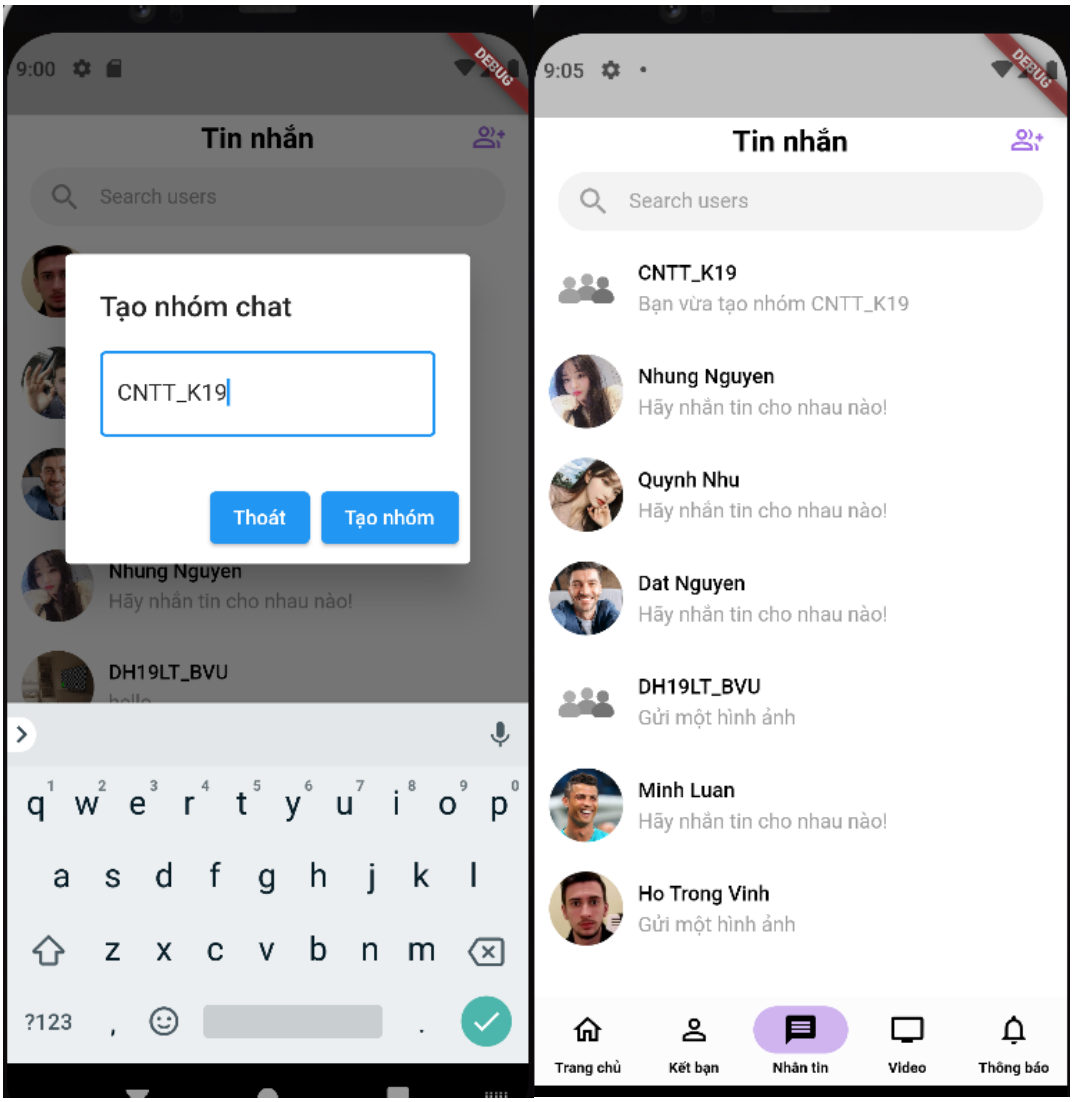
Hình 3.24. Màn hình danh sách cuộc trò chuyện.

Bước 2: Tại màn hình tin nhắn người dùng nhấn vào biểu tượng tạo nhóm để tạo nhóm chat.



Hình 3.25. Màn hình tạo nhóm chat.

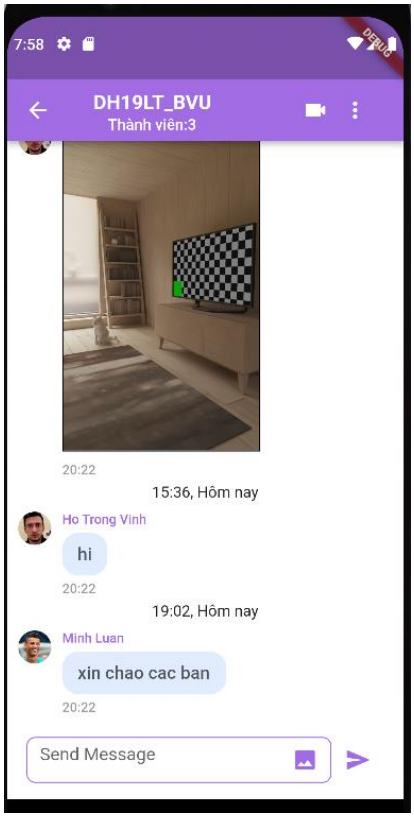
Bước 3: Người dùng nhập tên nhóm và nhấn tạo nhóm để tạo nhóm chat mới.



Hình 3.26. Màn hình tạo nhóm chat.

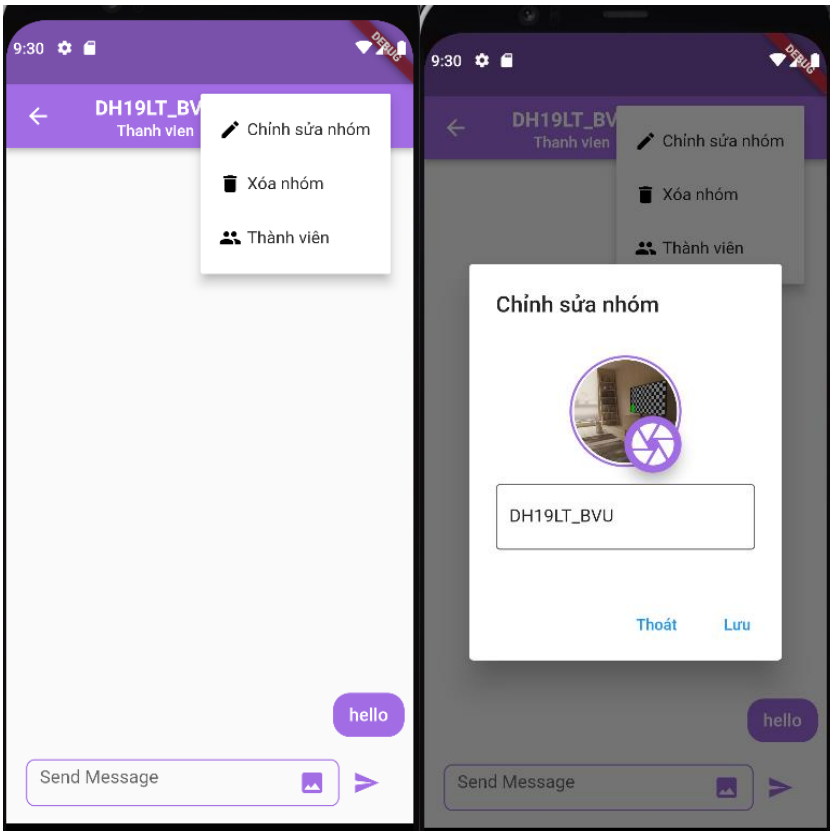
❖ **Chỉnh sửa nhóm**

Bước 1: Tại màn hình tin nhắn. người dùng chọn nhóm cần chỉnh sửa, ứng dụng chuyển sang màn hình chat của nhóm và hiển thị dữ liệu tin nhắn nhóm đó.



Hình 3.27. Màn hình tin nhắn nhóm chat.

Bước 2: Tại màn hình chat của nhóm, người dùng chọn menu góc trên bên phải, chọn chỉnh sửa nhóm.

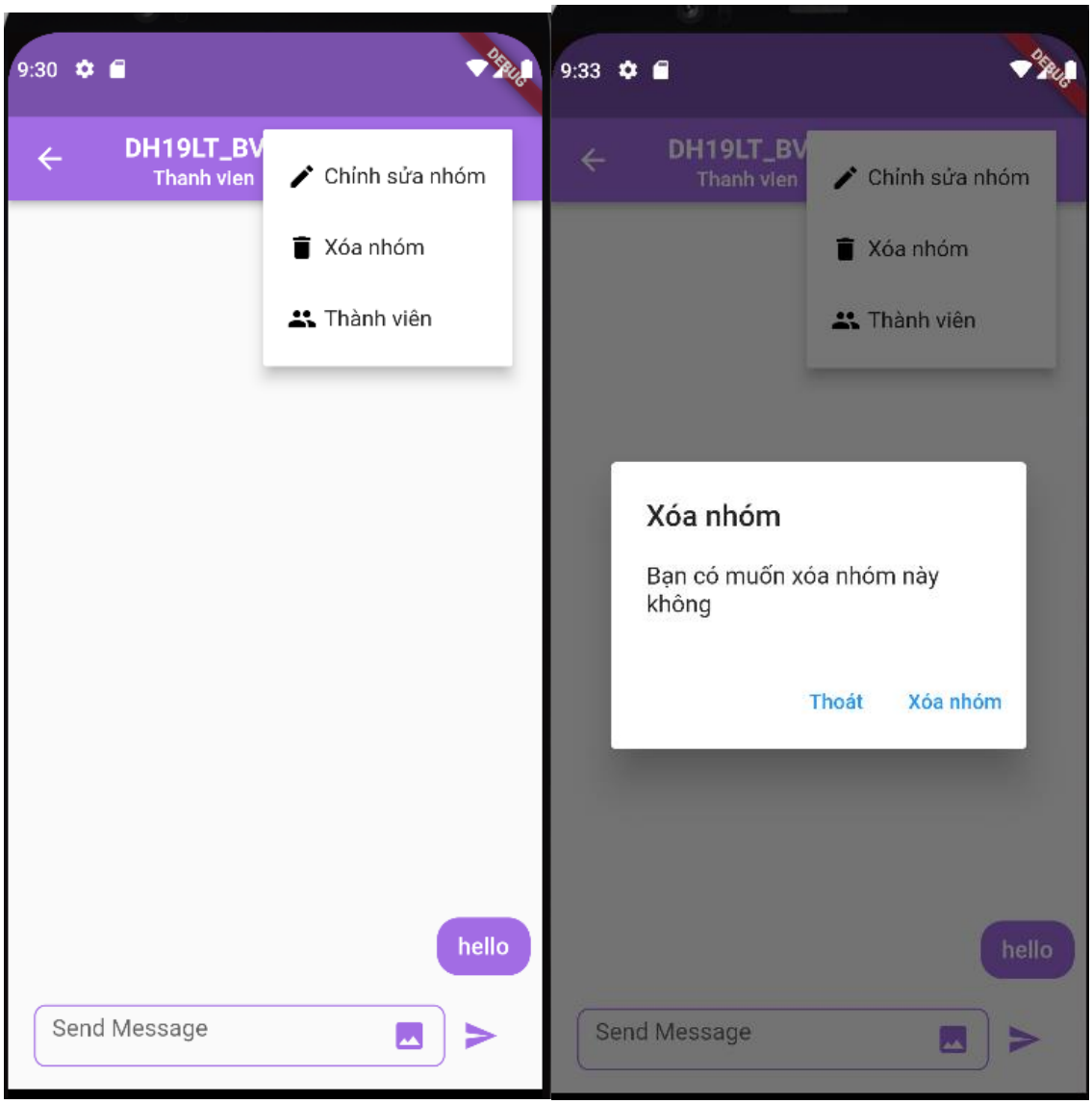


Hình 3.28. Màn hình sửa nhóm chat.

Bước 3: Tại đây, người dùng chỉnh sửa tên nhóm hay avatar nhóm và nhấn lưu để thay đổi.

❖ **Xóa nhóm**

Tại màn hình chat của nhóm, người dùng chọn menu góc trên bên phải, chọn xóa nhóm.



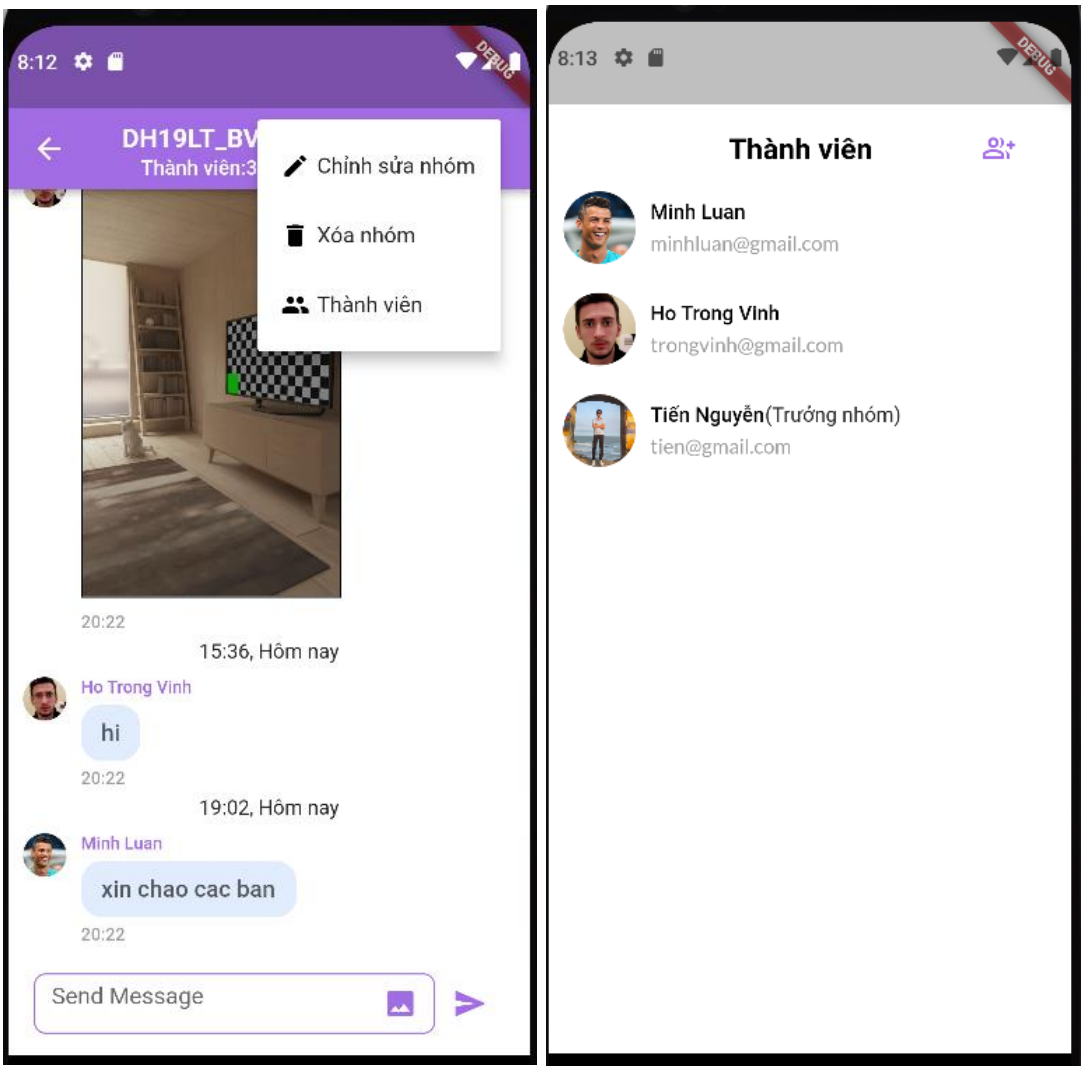
Hình 3.29. Màn hình hiển thị xóa nhóm chat.

3.2.9. Quản lý thành viên nhóm chat.

Một nhóm chat có nhiều thành viên vì vậy ứng dụng mạng xã hội cung cấp các chức năng quản lý thành viên như thêm thành viên hoặc xóa thanh viên ra nhóm.

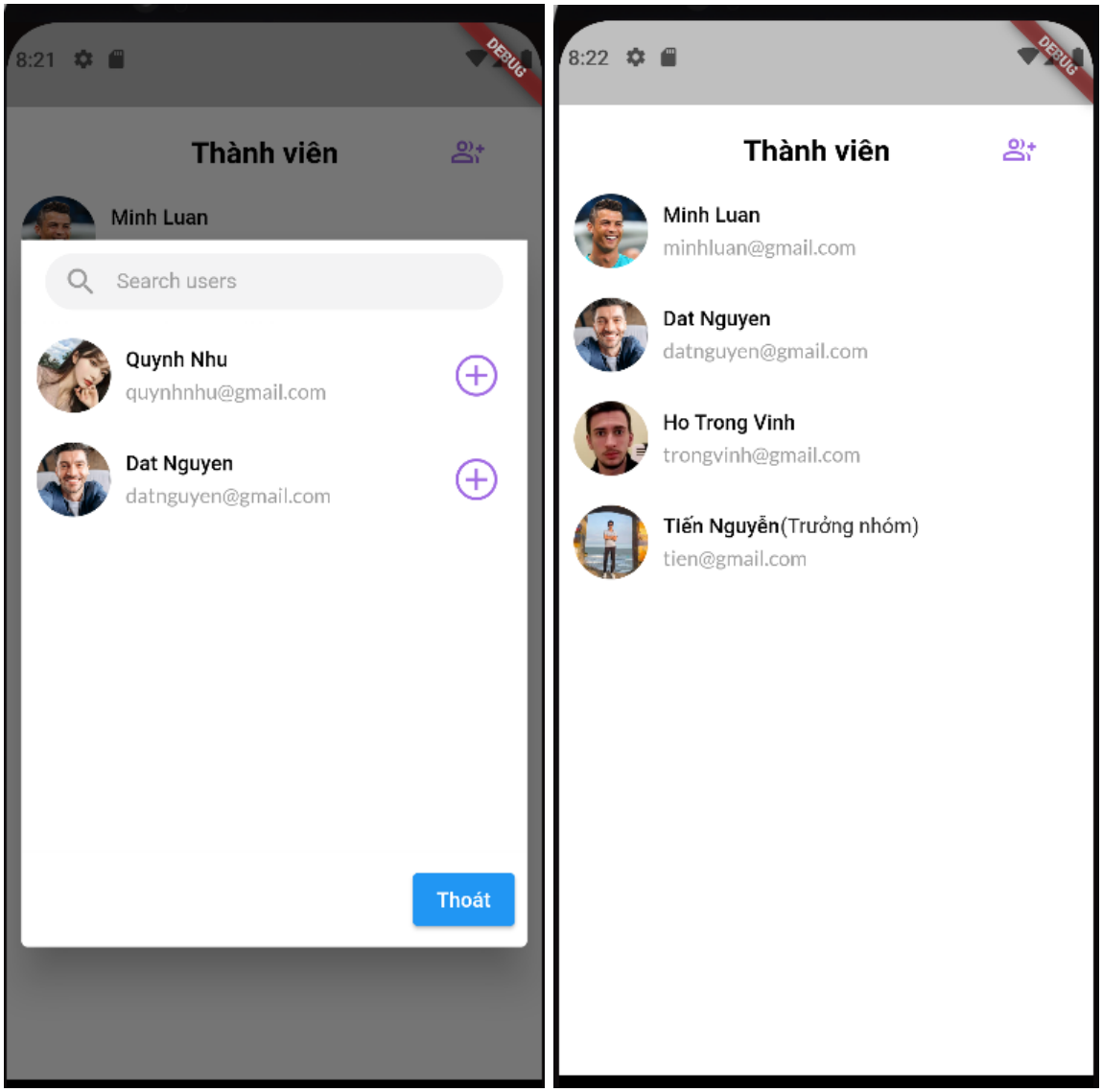
❖ **Thêm thành viên nhóm.**

Bước 1: Tại màn hình tin nhắn, chọn nhóm mong muốn, ứng dụng chuyển sang màn hình chat của nhóm. Người dùng chọn menu góc trên bên phải, chọn thành viên, ứng dụng chuyển đến màn hình danh sách thành viên.



Hình 3.30. Màn hình hiển thị thành viên nhóm chat.

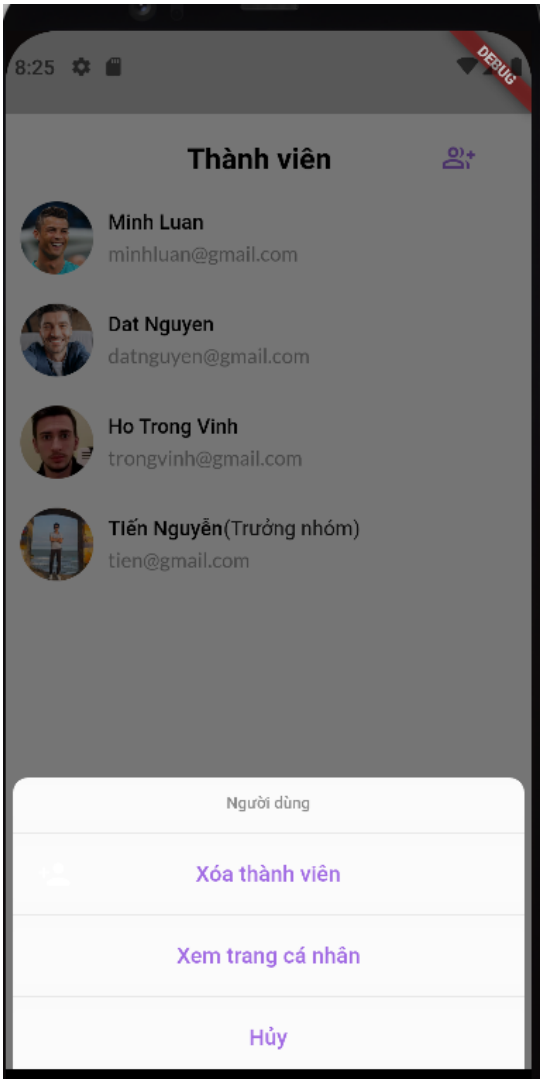
Bước 2: Khi hiện màn hình thêm thanh viên, Người dùng chọn biểu tượng trên góc phải màn hình và tìm bạn muốn thêm vào nhóm và nhấn nút icon (+) để thêm vào nhóm.



Hình 3.31. Màn hình hiển thị thêm thành viên nhóm chat.

❖ **Xóa thành viên nhóm.**

Chức năng này danh cho trưởng nhóm chat, trưởng nhóm vào màn hình danh sách thành viên nhóm, chọn thành viên bất kỳ, sau đó chọn xóa thành viên ở menu bên dưới.

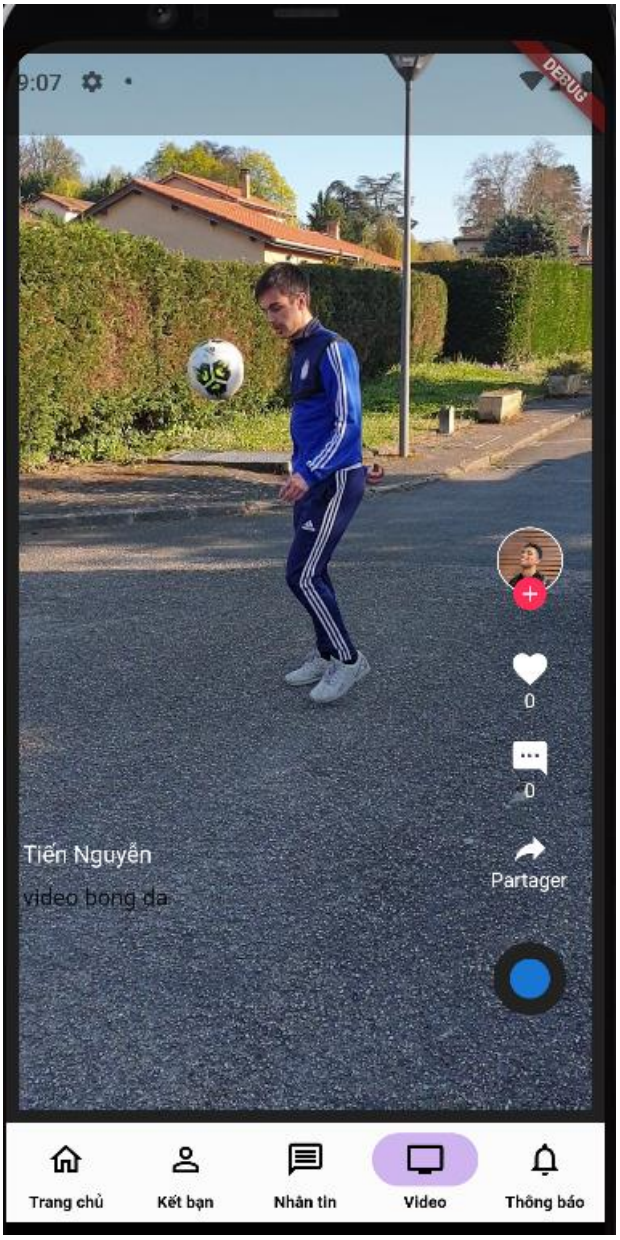


Hình 3.32. Màn hình hiển thị thêm thành viên nhóm chat.

3.2.10. Giao diện xem video ngắn.

Không chỉ có bài viết, ứng dụng mạng xã hội hiện này đều có video ngắn giúp người xem giải trí. Video ngắn được đăng đúng khung giờ và mang lại nhiều giá trị có khả năng tiếp cận được nhiều người. Xu hướng đang được quan tâm: Hiện nay, xem các video ngắn đang là xu hướng hành vi của người dùng.

Người dùng vào màn hình chính ứng dụng và chọn mục video để xem video ngắn của ứng dụng.

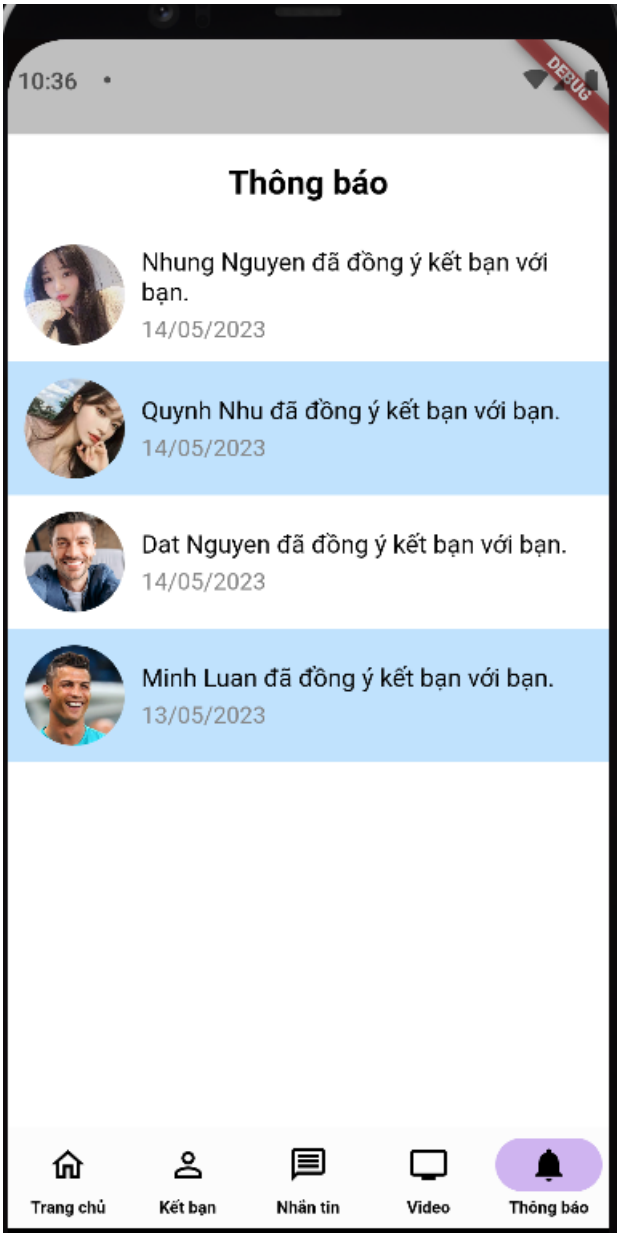


Hình 3.33. Màn hình hiển thị video ngắn.

3.2.11. Giao diện thông báo

Đây là giao diện giúp người dùng nắm bắt được thông tin tốt nhất đến với người dùng một cách nhanh và chính xác nhất.

Người dùng vào màn hình chính chọn mục thông báo để xem thông báo của người dùng trong ứng dụng.



Hình 3.34. Màn hình hiển thị thông báo.

CHƯƠNG 4. KẾT LUẬN

4.1. Kết quả đạt được

Xây dựng ứng dụng chạy trên 2 hệ điều hành di động phổ biến nhất hiện nay là Android và iOS, trong đó các chức năng đã hoàn thiện:

- Đăng nhập, đăng ký tài khoản người dùng.
- Quản lý bài viết, video ngắn.
- Quản lý hồ sơ cá nhân.
- Quản lý bạn bè.
- Quản lý nhóm chat, thành viên nhóm.
- Nhắn tin
- Tương tác bài viết comment, like.

4.2. Hạn chế còn tồn tại

Mặc dù có cơ hội tiếp cận Flutter từ trước đó, tuy nhiên với các chức năng trong ứng dụng có yêu cầu độ phức tạp tương đối, việc xây dựng ứng dụng này trải qua không ít khó khăn. Trong đó, hạn chế về kinh nghiệm thực tế là rào cản lớn khiến tôi chưa thể thực sự hoàn thành sản phẩm một cách trọn vẹn.

Có thể kể đến một số hạn chế của ứng dụng:

- **Giao diện:** Do bản thân không có nhiều năng khiếu về thiết kế, dẫn tới thiết kế của ứng dụng có giao diện chưa thật sự "thoáng", vẫn còn những nơi giao diện bị cứng nhắc, khô khan và thiếu tính sáng tạo, mang nhiều thiên hướng của một ứng dụng hệ thống hơn là dành cho người dùng phổ thông
- **Bảo mật:** Mặc dù đã sử dụng Security Rules, tuy nhiên vẫn còn một số trường hợp cần lưu ý như kiểm tra, ràng buộc dữ liệu khi được thêm vào, xóa đi. Vì FireStore không phải một CSDL có cấu trúc quan hệ như SQL, việc ràng buộc các trường dữ liệu hoàn toàn do người lập trình định nghĩa trong quá trình coding (xây dựng ứng dụng) và kết hợp với Security Rules
- **Thiếu chức năng:** Do thời gian có hạn cùng chưa tích lũy được nhiều kinh nghiệm, ứng dụng chỉ dừng lại ở việc người dùng có thể quản lý bài viết, tương tác, nhắn tin….

4.3. Hướng phát triển trong tương lai

Ứng dụng mạng xã hội này có thể được tiếp tục phát triển trong tương lai, với mục đích tăng tính ổn định trên môi trường thực tế cho sản phẩm, thêm các tính năng đầy đủ hơn và hướng tới những thay đổi nhất định trong quy trình vận hành nhằm thương mại hóa.

TÀI LIỆU THAM KHẢO

1. Flutter Firestore

https://firebase.flutter.dev/docs/firestore/usage/

2. Firestore Atomic Transaction

https://firebase.google.com/docs/firestore/manage-data/transactions

3. Firebase Email Authentication

h https://firebase.google.com/docs/auth/flutter/password-auth?hl=vi

4. Questions

https://stackoverflow.com/questions/

5. Smart Phone

(https://www.thegioididong.com/dtdd?g=android#c=42&p=39237&o=2&pi=0

https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide.

https://gs.statcounter.com/os-version-market-share/ios/mobile-tablet/worldwide

6. Diagrams

https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm

https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

PHỤ LỤC

1. Model người dùng.

```
class User {
  String? id;
  String? image;
  String? firstName;
  String? lastName;
  String? password;
  String? email;
  String? phoneNumber;
  String? address;
  String? birthDay;
  DateTime? createDate;
  bool? status;
  String? token;


  // ignore: invalid_required_positional_param
  User({this.id,this.firstName,
this.lastName,this.email,this.image,this.phoneNumber,this.birthDay,this.address,
this.password,this.createDate,this.status});
    User.fromJson(Map<String, dynamic> json)
      : id = json['Id'],
        email = json['Email'],
        firstName = json['FirstName'],
        lastName = json['LastName'],
        image = json['Image'],
        password = json['Password'],
        phoneNumber = json['PhoneNumber'],
```

```
    address = json['Address'],
    birthDay = json['BirthDay'],
    status = json['Status'],
    token = json['Token'],
    createDate = json['CreateDate'].toDate();

Map<String, dynamic> toJson() => {
    'Id': id,
    'Image': image,
    'FirstName':firstName,
    'LastName':lastName,
    'Password':password,
    'Email':email,
    "PhoneNumber":phoneNumber,
    "Address":address,
    "BirthDay":birthDay,
    'CreateDate':createDate,
    'Status':status,
    'Token':token
  };
   User.fromJson2(Map<String, dynamic> json)
  : id = json['Id'],
    email = json['Email'],
    firstName = json['FirstName'],
    lastName = json['LastName'],
    image = json['Image'],
    password = json['Password'],
    phoneNumber = json['PhoneNumber'],
    address = json['Address'],
```

```
      birthDay = json['BirthDay'],
      status = json['Status'],
      token = json['Token'],
      createDate = json['CreateDate'];
}
```

2. Code model bài viết

```
class Post {
  String? id;
  String? postContent;
  String? postImage;
  int? likeCount;
  int? commentCount;
  String? createBy;
  DateTime? createDate;
  DateTime? updatedDate;

Post({this.id,this.postContent,this.postImage,this.commentCount,this.likeCount,
this.createBy,this.createDate,this.updatedDate});
  Post.fromJson(Map<String, dynamic> json)
      : id = json['Id'],
      postContent = json['Content'],
      postImage = json['Image'],
      likeCount = json['LikeCount'],
      commentCount = json['CommentCount'],
      createBy = json['CreateBy'],
      createDate = json['CreateDate'].toDate(),
      updatedDate = json['UpdateDate'].toDate();

  Map<String, dynamic> toJson() => {
```

```
      'Id': id,
      'Content': postContent,
      'Image':postImage,
      'LikeCount':likeCount,
      'CommentCount':commentCount,
      'CreateBy':createBy,
      "CreateDate":createDate,
      'UpdateDate':updatedDate
    };

}
```

3. Code model nhóm chat

```
class GroupChat{
  String? id;
  String? groupName;
  String? avatarGroup;
  String? adminId;
  String? adminFullName;
  DateTime? createDate;
GroupChat({this.id,this.groupName,this.avatarGroup,this.adminId,this.adminFullName,this.createDate});
   GroupChat.fromJson(Map<String, dynamic> json)
     : id = json['Id'],
       groupName = json['GroupName'],
       avatarGroup = json['AvatarGroup'],
       adminId = json['AdminId'],
       adminFullName = json['AdminFullName'],
       createDate = json['CreateDate'].toDate();
```

```
 Map<String, dynamic> toJson() => {
       'Id': id,
       'GroupName':groupName,
       'AvatarGroup':avatarGroup,
       'AdminId':adminId,
       'AdminFullName':adminFullName,
       'CreateDate':createDate,
      };
      GroupChat.formJson2(Map<String, dynamic> json)
     : id = json['Id'],
       groupName = json['GroupName'],
       avatarGroup = json['AvatarGroup'],
       adminId = json['AdminId'],
       adminFullName = json['AdminFullName'],
       createDate = json['CreateDate'];


}
```

4. Code model thành viên nhóm

```
class MemberGroupChat{
 String? id;
 String? groupId;
 String? userId;
 DateTime? joinDate;
 MemberGroupChat({this.id,this.groupId,this.userId,this.joinDate});
 MemberGroupChat.fromJson(Map<String, dynamic> json)
    : id = json['Id'],
      groupId = json['GroupId'],
      userId = json['UserId'],
      joinDate = json['JoinDate'].toDate();
```

```
  Map<String, dynamic> toJson() => {
        'Id': id,
        'GroupId': groupId,
        'UserId':userId,
        'JoinDate':joinDate
      };
}
```

5. Code model cá nhân chat.

```
class ChatRoom{
  String? id;
  String? userFirstById;
  String? userSecondById;
  String? userFirstByFullName;
  String? userFirstByImage;
  String? userSecondByFullName;
  String? userSecondByImage;
  bool? statusUserFirst;
  bool? statusUserSecond;
  DateTime? createDate;

ChatRoom({this.id,this.userFirstById,this.userFirstByFullName,this.userFirstBy
Image,this.userSecondById,this.userSecondByFullName,this.userSecondByIma
ge,this.statusUserFirst,this.statusUserSecond,this.createDate});
   ChatRoom.fromJson(Map<String, dynamic> json)
      : id = json['Id'],
        userFirstById = json['UserFirstById'],
        userFirstByImage = json['UserFirstByImage'],
        userFirstByFullName = json["UserFirstByFullName"],
```

```
      userSecondById = json['UserSecondById'],
      userSecondByFullName = json["UserSecondByFullName"],
      userSecondByImage = json["UserSecondByImage"],
      statusUserFirst = json['StatusUserFirst'],
      statusUserSecond = json['StatusUserSecond'],
      createDate = json['CreateDate'].toDate();

  Map<String, dynamic> toJson() => {
      'Id': id,
      'UserFirstById': userFirstById,
      'UserFirstByFullName': userFirstByFullName,
      'UserFirstByImage': userFirstByImage,
      'UserSecondById':userSecondById,
      'UserSecondByFullName':userSecondByFullName,
      'UserSecondByImage':userSecondByImage,
      'StatusUserFirst':statusUserFirst,
      'StatusUserSecond':statusUserFirst,
      'CreateDate':createDate,
    };
}
```

6. Code model tin nhắn

```
class Message{
  final String? message;
  final String? sendById;
  final String? sendByFullName;
  final String? sendByImage;
  final String? objectId;
  final String? type;
  final String? typeChat;
```

```
 final DateTime? createDate;

Message({this.message,this.objectId,this.type,this.typeChat,this.sendById,this.sendByFullName,this.sendByImage,this.createDate});
   Message.fromJson(Map<String, dynamic> json)
     : message = json['Message'],
       type = json['"Type"'],
       sendById = json['SendById'],
       sendByFullName = json['SendByFullName'],
       sendByImage = json['SendByImage'],
       typeChat = json['TypeChat'],
       objectId = json['ObjectId'],
       createDate = json['CreateDate'].toDate();

 Map<String, dynamic> toJson() => {
       'Message': message,
       'Type':type,
       'SendById':sendById,
       'SendByFullName':sendByFullName,
       'SendByImage':sendByImage,
       'ObjectId':objectId,
       'TypeChat':typeChat,
       'CreateDate':createDate
     };
}
```

7. Codel model like

```
class Like{
 String? id;
 String? objectId;
```

```
  String? userId;
  int? type;
  String?objectType;
  DateTime? createDate;

Like({this.id,this.objectId,this.userId,this.type,this.objectType,this.createDate});
  Like.fromJson(Map<String, dynamic> json)
      : id = json['Id'],
        objectId = json['ObjectId'],
        userId = json['UserId'],
        type = json['Type'],
        objectType = json["ObjectType"],
        createDate = json['CreateDate'].toDate();

  Map<String, dynamic> toJson() => {
        'Id': id,
        'ObjectId': objectId,
        'UserId':userId,
        "Type":type,
        "ObjectType":objectType,
        'CreateDate':createDate
      };
}
```

8. Code model comment

```
class CommentObject{
   String? id;
   String? postId;
   String? userId;
   String? parentId;
```

```
  String? content;
  String? receiver;
  String? type;
  DateTime? createDate;

CommentObject({this.id,this.postId,this.userId,this.parentId,this.content,this.receiver,this.type,this.createDate});
 CommentObject.fromJson(Map<String, dynamic> json)
    : id = json['Id'],
      postId = json['ObjectId'],
      userId = json['UserId'],
      parentId = json['ParentId'],
      receiver = json['Receiver'],
      content = json['Content'],
      type = json['Type'],
      createDate = json['CreateDate'].toDate();
 Map<String, dynamic> toJson() => {
      'Id': id,
      'ObjectId': postId,
      'UserId':userId,
      'ParentId':parentId,
      "Receiver":receiver,
      'Content':content,
      "Type":type,
      "CreateDate":createDate,
    };
}
```

9. Code model kết bạn

```
class FriendShip{
```

```
  String? id;
  String? requester;
  String? addressee;
  bool? status;
  FriendShip({this.id,this.addressee,this.requester,this.status});
    FriendShip.fromJson(Map<String, dynamic> json)
      : id = json['Id'],
        requester = json['Requester'],
        addressee = json['Addressee'],
        status = json['Status'];

  Map<String, dynamic> toJson() => {
        'Id': id,
        'Requester': requester,
        'Addressee':addressee,
        'Status':status,
      };
}
class OtherShip{
  User? user;
  int? status;
  OtherShip({this.user,this.status});
}
```

10. Code model video ngắn

```
class ShortVideo{
  String? id;
  String? content;
  String? videoURL;
  String? createById;
```

```
String? createByName;
String? createByImage;
int? commentCount;
int? likeCount;
DateTime? createDate;
DateTime? updatedDate;

ShortVideo({this.id,this.content,this.videoURL,this.commentCount,this.likeCount,this.createById,this.createByName,this.createByImage,this.createDate,this.updatedDate});
ShortVideo.fromJson(Map<String, dynamic> json)
    : id = json['Id'],
      content = json['Content'],
      videoURL = json['VideoURL'],
      likeCount = json['LikeCount'],
      commentCount = json['CommentCount'],
      createById = json['CreateById'],
      createByName = json['CreateByName'],
      createByImage = json['CreateByImage'],
      createDate = json['CreateDate'].toDate(),
      updatedDate = json['UpdateDate'].toDate();

Map<String, dynamic> toJson() => {
      'Id': id,
      'Content': content,
      'VideoURL':videoURL,
      'LikeCount':likeCount,
      'CommentCount':commentCount,
      'CreateBy':createById,
```

```
      'CreateByName':createByName,
      'CreateByImage':createByImage,
      "CreateDate":createDate,
      'UpdateDate':updatedDate
    };
}
```

11. Code model thông báo

```
class NotificationObject{
  String? id;
  String? content;
  String? receiver;
  String? idObject;
  String? sender;
  DateTime? createDate;

NotificationObject({this.id,this.content,this.receiver,this.idObject,this.sender,this.createDate});
  NotificationObject.fromJson(Map<String, dynamic> json)
     : id = json['Id'],
      content = json['Content'],
      receiver = json['Receiver'],
      idObject = json['ObjectId'],
      sender = json['Sender'],
      createDate = json['CreateDate'].toDate();
  Map<String, dynamic> toJson() => {
      'Id': id,
      'Content': content,
      'Receiver':receiver,
      'Sender':sender,
```

```
      'ObjectId':idObject,
      "CreateDate":createDate,
    };

}
```

12. Code xử lý việc đăng ký tài khoản

```
Future SignUp() async{
if(controllerEmail.text!=""&&controllerFirstName.text!=""&&controllerLastName.text!=""&&controllerConfirmPassword.text!=""&&controllerPassword.text!=""){
    if(EmailValidator.validate(controllerEmail.text)){
      if(controllerPassword.text==controllerConfirmPassword.text){
        if(await checkUserByEmail(controllerEmail.text)){
          Fluttertoast.showToast(msg: "Email này đã được sử dụng!");
        }
        else{
          try{
            User user = User(
id: usercreate.user!.uid,firstName:controllerFirstName.text,lastName:controllerLastName.text,email:controllerEmail.text,password:controllerPassword.text,createDate:DateTime.now(),phoneNumber: "",image:"https://firebasestorage.googleapis.com/v0/b/project-cb943.appspot.com/o/image%2FlogoPreson%2FUnknown_person.jpg?alt=media&token=061d880a-9464-41e4-af7e-c259aedcaef7",status: false);
            CreateData("User",user);
          }
          catch(e){
            Fluttertoast.showToast(msg: "Email không tồn tại!");
            this.dispose();
          }
```

```
        }


    }
     else{
       Fluttertoast.showToast(msg: "Vui lòng nhập lại mật khẩu chính xác!");
     }
    }
    else{
      Fluttertoast.showToast(msg: "Vui lòng nhập đúng định dạng email!");
    }
  }
  else{
    Fluttertoast.showToast(msg: "Vui lòng nhập đầy đủ thông tin!");
  }


}
```

13. Code xử lý việc like bài viết

```
likePost(Post post,int index) async{
Like getlike = await
checkLike(auth.FirebaseAuth.instance.currentUser!.uid,post.id!);
   // ignore: unnecessary_null_comparison
   if(getlike.id!=null){
     getlike.type = index;
     updateLikePost(getlike);
     setState(() {
       ((posts.firstWhere((element) => (element["post"] as
Post).id==post.id)["listUserLikePost"]) as List<Like>).firstWhere((element) =>
element.userId==auth.FirebaseAuth.instance.currentUser!.uid).type=index;
```

```
    });
  }
  else{

    Like like = Like(id: "",objectId: post.id,userId:
auth.FirebaseAuth.instance.currentUser!.uid,type: index,objectType:
"post",createDate: DateTime.now());
    post.likeCount=post.likeCount! + 1;
    updatePost(post);
    CreateNewData("Like", like);
    if(post.createBy!=auth.FirebaseAuth.instance.currentUser!.uid){
      User user= await
getUserById(auth.FirebaseAuth.instance.currentUser!.uid);
      NotificationObject notification = NotificationObject(id: "",content:
"${user.firstName} ${user.lastName} đã thích một bài viết của bạn",receiver:
post.createBy,createDate: DateTime.now(),idObject: post.id,sender: user.id);
      CreateNewData("Notification", notification);
      User userPost = await getUserById(post.createBy!);
      PushNotification.sendPushNotification(User(),"${user.firstName}
${user.lastName} đã thích một bài viết của bạn",userPost.token!);

    }
    setState(() {
    ((posts.firstWhere((element) => (element["post"] as
Post).id==post.id)["post"]) as Post).likeCount! + 1;

    ((posts.firstWhere((element) => (element["post"] as
Post).id==post.id)["listUserLikePost"]) as List<Like>).add(like);
  });

  }
```

```
}
```

14. Code xử lý việc comment bài viết

```
void commentPost(String parentId,User userReceiver) async{
   if(commentController.text!=""){
      // ignore: unnecessary_null_comparison
      String userid = userReceiver != null ? userReceiver.id! :  "";
      CommentObject comment = CommentObject(id: "",userId:
auth.FirebaseAuth.instance.currentUser!.uid,postId:(jsonPost!["post"] as
Post).id,parentId: parentId,receiver: userid,content:
commentController.text,type: "post", createDate: DateTime.now());
   CreateNewData("Comment", comment);
   if(parentId!=""){
      ((jsonListComment!.firstWhere((element) => (element["parentComment"]
as CommentObject).id==parentId))["jsonSubComment"] as
List<Map<String,Object>>).add({"subComment":comment,"userSubComment"
:user});
      NotificationObject notification = NotificationObject(id: "",content:
"${user.firstName} ${user.lastName} đã nhắc bạn trong bài viết bạn quan
tâm.",receiver: (jsonPost!["post"] as Post).createBy,createDate:
DateTime.now(),idObject: (jsonPost!["post"] as Post).id,sender: user.id);
      CreateNewData("Notification", notification);
      PushNotification.sendPushNotification(User(),"${user.firstName}
${user.lastName} đã bình luận một bài viết của bạn.",userReceiver.token!);
   }
   else{

jsonListComment!.insert(0,{"parentComment":comment,"userComment":user,"j
sonSubComment":<Map<String,Object>>[]});
     if((jsonPost!["post"] as Post).createBy !=
auth.FirebaseAuth.instance.currentUser!.uid){
      NotificationObject notification = NotificationObject(id: "",content:
"${user.firstName} ${user.lastName} đã bình luận một bài viết của
```

```
bạn.",receiver: (jsonPost!["post"] as Post).createBy,createDate:
DateTime.now(),idObject: (jsonPost!["post"] as Post).id,sender: user.id);
      CreateNewData("Notification", notification);
      PushNotification.sendPushNotification(User(),"${user.firstName}
${user.lastName} đã bình luận một bài viết của bạn.",userReceiver.token!);
    }
   }
  (jsonPost!["post"] as Post).commentCount = (jsonPost!["post"] as
Post).commentCount! + 1;
 updatePost((jsonPost!["post"] as Post));
   setState(() {
    jsonListComment;
    jsonPost;
   });
   }
 }
```

15. Code xử lý lấy danh sách cuộc trò chuyện RealTime

```
title: StreamBuilder<DocumentSnapshot>(
        stream:
            _firestore.collection("User").doc(userMap.id).snapshots(),
        builder: (context, snapshot) {
          if (snapshot.data != null) {
            return Container(
              child: Column(
                children: [
                  // ignore: prefer_interpolation_to_compose_strings
                Container(
                width: 120,
```

```
            child: RichText(
              textAlign: TextAlign.center,
            overflow: TextOverflow.ellipsis,
            strutStyle: const StrutStyle(fontSize: 22.0),
            text: TextSpan(
            text: "${userMap.firstName!} ${userMap.lastName!}",
            style: TextStyle(fontSize: 18,fontWeight: FontWeight.bold)
          ),)),
              Text(
                (snapshot.data!['Status'] as bool) ==true?"Online":"Offline",
                style: TextStyle(fontSize: 14),
              ),
            ],
          ),
        );
      } else {
        return Container();
      }
    },
  ),
),
body: SingleChildScrollView(
  child: Column(
    children: [
      Container(
        height: size.height / 1.35,
        width: size.width,
        child: StreamBuilder<QuerySnapshot>(
```

```
            stream: _firestore
                .collection('UserChat')
                .doc(chatRoomId)
                .collection('Message')
                .orderBy("CreateDate", descending: true).limit(30)
                .snapshots(),
            builder: (BuildContext context,
                AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.data != null){
                var listTime = [];
                for(int i = 0 ;i<snapshot.data!.docs.length;i++){
                  if(i<snapshot.data!.docs.length-1){
                    if(((snapshot.data!.docs[(i)].data() as Map<String,
dynamic>)["CreateDate"].toDate() as
DateTime).difference((snapshot.data!.docs[i+1].data() as Map<String,
dynamic>)["CreateDate"].toDate() as DateTime).inMinutes>30){
                      listTime.add((snapshot.data!.docs[i].data() as Map<String,
dynamic>)["CreateDate"].toDate().toString());
                    }
                    else{
                     listTime.add("");

                    }
                  }
                  else{
                    listTime.add("");
                  }
                }
                return ListView.builder(
```

```
                controller: scrollController,
                reverse: true,
                shrinkWrap: true,
                itemCount: snapshot.data!.docs.length,
                itemBuilder: (context, index) {
                  Map<String, dynamic> map = snapshot.data!.docs[index]
                      .data() as Map<String, dynamic>;
                  return Column(children: [
                    listTime[index]!=""?Center(child:
Text(DateTime.now().difference(map['CreateDate'].toDate() as
DateTime).inDays==0?DateFormat("HH:mm").format(map['CreateDate'].toDat
e())+", Hôm nay": DateFormat("HH:mm,
dd/MM/yyyy").format(map['CreateDate'].toDate())),):Container(),
                      messages(size, map, context)]);
                },
              );
            } else {
              return Container();
            }
          },
        ),
```

16. Code xử lý gửi yêu cầu kết bạn

```
sendRequestShip(Map<String,Object> user) async{
    await CreateNewData("FriendShip", FriendShip(id:"",requester: auth.FirebaseAuth.instance.currentUser!.uid,addressee: (user["user"] as User).id!,status: false));
    setState(() {
     user["status"] = 1;
    });
    NotificationObject notification = NotificationObject(id: "",content: "${myUser.firstName} ${myUser.lastName} đã gửi một yêu cầu kết bạn.",receiver:(user["user"] as User).id ,createDate: DateTime.now(),idObject: myUser.id,sender: myUser.id);
    CreateNewData("Notification", notification);

    PushNotification.sendPushNotification(User(),"${myUser.firstName} ${myUser.lastName} đã gửi một yêu cầu kết bạn.",(user["user"] as User).token!);
  }
```

17. Code xử lý đồng ý kết bạn

```
agreeShip(Map<String,Object> user) async{
  await makeFriend((user["user"] as User).id!, auth.FirebaseAuth.instance.currentUser!.uid);
  var chatRoom = ChatRoom(id: "",userFirstById:(user["user"] as User).id!,userFirstByFullName:"${(user["user"] as User).firstName!} ${(user["user"] as User).lastName!}",userFirstByImage: (user["user"] as User).image!,statusUserFirst: false,userSecondById: myUser.id,userSecondByFullName: "${myUser.firstName!} ${myUser.lastName!}",userSecondByImage: myUser.image,statusUserSecond: false,createDate: DateTime.now());
  CreateNewData("UserChat", chatRoom);
   var chatfinal = ChatFinal(object: chatRoom.toJson(),chatContentFinal: "Hãy nhắn tin cho nhau nào!",chatFinalDate: DateTime.now(),typeChat: "user");
  CreateNewData("Chat", chatfinal);
```

```
  setState(() {
    user["status"] = 3;
  });
  NotificationObject notification = NotificationObject(id: "",content: "${myUser.firstName} ${myUser.lastName} đã đồng ý kết bạn với bạn.",receiver:(user["user"] as User).id!,createDate: DateTime.now(),idObject: myUser.id,sender: myUser.id);
   CreateNewData("Notification", notification);

  PushNotification.sendPushNotification(User(),"${myUser.firstName} ${myUser.lastName} đã đồng ý kết bạn với bạn.",(user["user"] as User).token!);
 }
```