|  |  |
| --- | --- |
|  | **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM**  **KHOA CÔNG NGHỆ THÔNG TIN** |

**BÁO CÁO ĐỒ ÁN LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG**

**Chuyên ngành: Công nghệ phần mềm**

***Đề tài:***

**SOCIAL NETWORK APP**

|  |
| --- |
| *Thành phố Hồ Chí Minh, ngày 4 tháng 1 năm 2025* |

|  |  |  |  |
| --- | --- | --- | --- |
| Giảng viên hướng dẫn | | **Lê Nhật Tùng** |  |
| Sinh viên thực hiện: | | **Huỳnh Gia Khiêm**  **Lê Quang Hiến** | **2180604879**  **2180607487** |
|  | | **Nguyễn Lộc Xuân Sang** | **2180607959** |
| Lớp: | | **21DTHD5** |  |
|  | |  |  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV/Họ và Tên** | **Nội dung công việc thực hiện** | **% góp phần** | **Tự đánh giá**  **Điểm** |
| Huỳnh Gia Khiêm  (2180604879) | - Khảo sát và mô tả các chức năng của hệ thống  - Phân tích thiết kế cơ sở dữ liệu  - Code toàn bộ giao diện (font-end) và toàn bộ logic của hệ thống (back-end) | 40% | 10 điểm |
| Nguyễn Lộc Xuân Sang (2180607959) | - Làm PPT , hỗ trợ code | 15% | 6.5 |
| Lê Quang Hiến  (2180607487) | - Làm báo cáo, hỗ trợ code | 20% | 7.5 |

**LỜI CAM ĐOAN**

Chúng em xin cam đoan đề tài “Social Network App” là thành quả của một quá trình học tập và nghiên cứu nghiêm túc, độc lập dưới sự hướng dẫn của Thầy **Lê Nhật Tùng** khoa công nghệ thông tin Trường Hutech Đại học công nghệ tp. HCM.Tất cả nội dung trong bài tiểu luận không có bất kỳ sự gian lận hay sao chép của người khác, đó là sản phẩm do chính em đã đạt được sau quãng thời gian học tập tại trường. Các số liệu và minh chứng được trình bày trong báo cáo là hoàn toàn đúng sự thật. Nếu có bất kì vướng mắc hay vấn đề nào phát sinh tôi xin được chịu trách nhiệm trước hội đồng kỷ luật của khoa và nhà trường.

**LỜI CẢM ƠN**

Để hoàn thành tốt đồ án này em xin chân thành cảm ơn tới thầy **Lê Nhật Tùng** đã hướng dẫn chúng em em trong suốt thời gian vừa qua. Em xin chân thành cảm ơn thầy **Lê Nhật Tùng** đã hướng dẫn em trong suốt quá trình làm đồ án.

Bên cạnh đó để hoàn thành tốt đồ án này chúng em cũng nhận được nhiều sự giúp đỡ quý báo từ các bạn, từ người anh. Em xin chân thành cảm ơn.

Mặc dù đã nổ lực hết sức mình nhưng đồ án của chúng em khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và sự chỉ bảo tận tình của quý thầy cô và các bạn để thực hiện tốt đồ án này. Một lần nữa chúng em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, Tháng 1 năm 2025

Sinh viên thực hiện:

**2180607487 – Lê Quang Hiến**

**2180604879 – Huỳnh Gia Khiêm**

**2180607959 – Nguyễn Lộc Xuân Sang**

Mục Lục

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 8](#_Toc186841412)

[1.1 Giới thiệu đề tài 8](#_Toc186841413)

[1.2 Ý nghĩa thực tiễn 8](#_Toc186841414)

[1.3 Nhiệm vụ đồ án 9](#_Toc186841415)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 11](#_Toc186841416)

[2.1 Flutter là gì? 11](#_Toc186841417)

[2.1.1 Cấu trúc nổi bật của Flutter 12](#_Toc186841418)

[2.1.2 Widget trong Flutter 16](#_Toc186841419)

[2.1.3 Ưu điểm và nhược điểm Flutter 17](#_Toc186841420)

[2.2 Spring Boot là gì? 18](#_Toc186841421)

[2.2.1 Spring Boot và các đặc điểm chính 19](#_Toc186841422)

[2.2.2 Cấu trúc một ứng dụng Spring Boot 21](#_Toc186841423)

[2.2.3 Dependency Injection trong Spring Boot 22](#_Toc186841424)

[2.3 Tích hợp Flutter với Spring Boot 24](#_Toc186841425)

[2.3.1 Kiến trúc Client-Server 24](#_Toc186841426)

[2.3.2 Giao tiếp giữa Flutter và Spring Boot qua REST API 25](#_Toc186841427)

[2.3.3 Cách thức gửi và nhận dữ liệu 26](#_Toc186841428)

[2.4 MySQL 28](#_Toc186841429)

[Chương 3: KẾT QUẢ THỰC NGHIỆM 30](#_Toc186841430)

[3.1 Phân tích và thiết kế hệ thống 30](#_Toc186841431)

[3.1.1 Sơ đồ usecase 30](#_Toc186841432)

[3.1.2 Sơ đồ Sequence 36](#_Toc186841433)

[3.1.3 ERD 37](#_Toc186841434)

[3.1.4 ClassDiagram 38](#_Toc186841435)

[3.1.5 ActivityDiagram 39](#_Toc186841436)

[3.1.6 Các bảng của cơ sở dữ liệu 39](#_Toc186841437)

[3.2 Giao diện ứng dụng thực nghiệm 42](#_Toc186841438)

[3.2.1 Trang đăng kí, đăng nhập 42](#_Toc186841439)

[3.2.2 Trang chủ 43](#_Toc186841440)

[3.2.3 Trang thông tin cá nhân 44](#_Toc186841441)

[3.2.4 Trang thông báo 45](#_Toc186841442)

[3.2.5 Trang danh sách tin nhắn 46](#_Toc186841443)

[3.2.6 Chi tiết tin nhắn 47](#_Toc186841444)

[3.2.7 Trang thông báo lời mời kết bạn 48](#_Toc186841445)

[3.2.8 Trang cài đặt 49](#_Toc186841446)

[Chương 4: KẾT LUẬN 50](#_Toc186841447)

[4.1 Kết luận về Những Kết quả Đạt được: 50](#_Toc186841448)

[4.2 Định Hướng Tương Lai của Sản Phẩm: 50](#_Toc186841449)

[4.3 Những Đóng Góp và Đề Xuất Mới: 50](#_Toc186841450)

[TÀI LIỆU THAM KHẢO 51](#_Toc186841451)

MỤC LỤC HÌNH ẢNH

[Hình 1. Kiến trúc Widget-Based 13](#_Toc186839862)

[Hình 2. Kiến trúc Render Tree 13](#_Toc186839863)

[Hình 3. Hot Reload & Hot Restart 14](#_Toc186839864)

[Hình 4. Dart Programming Language 15](#_Toc186839865)

[Hình 5. Cross-Platform 15](#_Toc186839866)

[Hình 6. Flutter 18](#_Toc186839867)

[Hình 7. Spring Boot 19](#_Toc186839868)

[Hình 8. Kiến trúc Client-Server 25](#_Toc186839869)

[Hình 9. Giao tiếp giữa Flutter và Spring Boot qua REST API 26](#_Toc186839870)

[Hình 10. JSON 27](#_Toc186839871)

[Hình 11. MySQL 28](#_Toc186839872)

[Hình 12. Usecase tổng quát 30](#_Toc186839873)

[Hình 13. Usecase thông báo 31](#_Toc186839874)

[Hình 14. Usecase thống kê 31](#_Toc186839875)

[Hình 15. Usecase bảo mật 32](#_Toc186839876)

[Hình 16. Usecase quản lý tương tác 32](#_Toc186839877)

[Hình 17. Usecase quản lý thông tin người dùng 33](#_Toc186839878)

[Hình 18. Usecase quản lý tài khoản 33](#_Toc186839879)

[Hình 19. Usecase quản lý nhóm & trang 34](#_Toc186839880)

[Hình 20. Usecase quản lý nội dung chat 34](#_Toc186839881)

[Hình 21. Usecase quản lý mối quan hệ 35](#_Toc186839882)

[Hình 22. Usecase quản lý nội dung tìm kiếm 35](#_Toc186839883)

[Hình 23. Usecase quản lý thông tin bài đăng 36](#_Toc186839884)

[Hình 24. Sequence đăng nhập 36](#_Toc186839885)

[Hình 25. Sequence đăng ký 37](#_Toc186839886)

[Hình 26. ERD 37](#_Toc186839887)

[Hình 27. ClassDiagram 38](#_Toc186839888)

[Hình 28. ActivityDiagram 39](#_Toc186839889)

[Hình 29. Trang đăng kí, đăng nhập 43](#_Toc186839890)

[Hình 30. Trang chủ 44](#_Toc186839891)

[Hình 31. Trang thông tin cá nhân 45](#_Toc186839892)

[Hình 32. Trang thông báo 46](#_Toc186839893)

[Hình 33. Trang danh sách tin nhắn 47](#_Toc186839894)

[Hình 34. Chi tiết tin nhắn 48](#_Toc186839895)

[Hình 35. Trang thông báo lời mời kết bạn 49](#_Toc186839896)

[Hình 36. Trang cài đặt 50](#_Toc186839897)

### CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Trong một thế giới kết nối không ngừng, mạng xã hội đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. Một ứng dụng quản lý mạng xã hội là một nền tảng kỹ thuật số được thiết kế để kết nối và tương tác giữa các người dùng thông qua việc chia sẻ nội dung, giao tiếp và xây dựng cộng đồng trực tuyến.

Đề tài này tập trung vào việc nghiên cứu, phát triển và triển khai một ứng dụng quản lý mạng xã hội hiện đại và linh hoạt. Mục tiêu của dự án là xây dựng một nền tảng đa chức năng, mang lại trải nghiệm người dùng tốt nhất có thể và đồng thời cung cấp cho người quản trị công cụ để quản lý và điều hành mạng xã hội một cách hiệu quả.

Trong quá trình nghiên cứu, chúng tôi sẽ tập trung vào các mục tiêu sau:

* Nghiên cứu về lịch sử và phát triển của mạng xã hội: Đây là một phần quan trọng để hiểu rõ hơn về cách mạng xã hội đã phát triển từ khi bắt đầu và những ảnh hưởng của nó đối với xã hội và văn hóa.
* Phân tích về các tính năng chính của một ứng dụng quản lý mạng xã hội: Bạn có thể xem xét về các tính năng như chia sẻ nội dung, tạo hồ sơ cá nhân, kết nối bạn bè, nhóm và trang, cũng như các tính năng bảo mật và quản lý tài khoản.
* Tầm quan trọng của việc tạo ra trải nghiệm người dùng tốt nhất: Đây là một mục tiêu quan trọng trong việc thiết kế và phát triển một ứng dụng mạng xã hội, vì nó ảnh hưởng trực tiếp đến sự hài lòng và sự tiếp tục sử dụng của người dùng.
* Phân tích về yếu tố thành công của một ứng dụng quản lý mạng xã hội: Bạn có thể nghiên cứu về những yếu tố quan trọng như sự tương tác, tính viral, tính đa dạng của nội dung, và cách mà những yếu tố này ảnh hưởng đến sự phát triển và thành công của mạng xã hội.
* Xem xét về yếu tố kỹ thuật và quản trị: Bạn có thể tập trung vào các yếu tố kỹ thuật như cơ sở hạ tầng, độ tin cậy và bảo mật của hệ thống, cũng như các công cụ quản trị để hỗ trợ việc quản lý và điều hành mạng xã hội một cách hiệu quả.

## Ý nghĩa thực tiễn

Đồ án này mang lại nhiều ý nghĩa thực tiễn đối với các cá nhân và tổ chức có liên quan đến việc phát triển và quản lý mạng xã hội, cũng như cộng đồng người dùng mạng xã hội:

* Đáp ứng nhu cầu của người dùng: Với sự phát triển không ngừng của công nghệ và nhu cầu người dùng ngày càng đa dạng, việc phát triển một ứng dụng quản lý mạng xã hội mang lại giải pháp linh hoạt và đáp ứng nhu cầu thực tế của người dùng.
* Hỗ trợ doanh nghiệp và cá nhân: Cung cấp cho doanh nghiệp và cá nhân một nền tảng mạnh mẽ để tương tác, quảng bá thương hiệu, tiếp cận khách hàng, và phát triển mối quan hệ trong cộng đồng trực tuyến.
* Tạo ra cộng đồng trực tuyến tích cực: Xây dựng một môi trường trực tuyến an toàn và tích cực, tạo điều kiện cho sự kết nối, chia sẻ kiến thức và kinh nghiệm, cũng như giao lưu văn hóa giữa các thành viên của cộng đồng.
* Nâng cao hiệu suất quản lý: Đồ án này cung cấp cho người quản trị công cụ và tài nguyên để quản lý và điều hành mạng xã hội một cách hiệu quả, từ quản lý nội dung đến bảo mật thông tin và theo dõi hoạt động của người dùng.
* Thúc đẩy sự phát triển công nghệ: Việc phát triển một ứng dụng quản lý mạng xã hội yêu cầu sử dụng các công nghệ và phương pháp mới nhất, từ đó thúc đẩy sự tiến bộ trong lĩnh vực công nghệ thông tin và phát triển ứng dụng.
* Nâng cao nhận thức về an ninh mạng: Trong bối cảnh nguy cơ an ninh mạng ngày càng cao, việc áp dụng các biện pháp bảo mật trong quá trình phát triển và quản lý mạng xã hội giúp nâng cao nhận thức và bảo vệ thông tin cá nhân của người dùng.

## 1.3 Nhiệm vụ đồ án

**- Đăng ký, Đăng nhập và Đăng xuất:**

* Cho phép người dùng đăng ký tài khoản mới bằng cách cung cấp thông tin cần thiết và xác minh email.
* Hỗ trợ đăng nhập vào hệ thống bằng tên đăng nhập hoặc email cùng mật khẩu.
* Cung cấp chức năng đăng xuất để người dùng có thể thoát khỏi tài khoản của mình.

**- Quản lý Bài đăng:**

* Cho phép người dùng thêm, chỉnh sửa và xóa bài đăng của mình.
* Hỗ trợ tải ảnh lên và chia sẻ nội dung văn bản trong bài đăng.
* Cung cấp tính năng lưu bài đăng để người dùng có thể xem lại sau này.

**- Quản lý Bình luận:**

* Cho phép người dùng thêm, chỉnh sửa và xóa bình luận trên các bài đăng.
* Hỗ trợ tính năng tương tác như thả cảm xúc (like, heart, haha, etc.) trên các bình luận.

**- Chức năng Chia sẻ bài đăng:**

* Cho phép người dùng chia sẻ bài đăng của mình lên trang cá nhân hoặc trong các nhóm mạng xã hội.

**- Quản lý Danh sách bạn bè:**

* Cung cấp tính năng thêm và xóa bạn bè từ danh sách bạn bè của người dùng.
* Thông báo cho người dùng khi họ nhận được lời mời kết bạn mới.

**- Chức năng Nhắn tin:**

* Cho phép người dùng gửi và nhận tin nhắn riêng tư với bạn bè trên nền tảng mạng xã hội.

**- Tạo và Quản lý Nhóm chat:**

* Cho phép người dùng tạo và tham gia các nhóm chat để trò chuyện với nhiều người cùng một lúc.
* Cung cấp chức năng quản lý thành viên và nội dung trong nhóm chat.

**- Chỉnh sửa Thông tin cá nhân:**

* Cho phép người dùng chỉnh sửa thông tin cá nhân như ảnh đại diện, mô tả, v.v.

**- Tìm kiếm và Xem trang cá nhân của người khác:**

* Cung cấp chức năng tìm kiếm trang cá nhân của người dùng khác bằng tên hoặc tài khoản.
* Cho phép người dùng xem thông tin trang cá nhân của người khác, bao gồm bài đăng, ảnh đại diện, v.v.

**- Tạo và Quản lý Nhóm trên Mạng xã hội:**

* Cho phép người dùng tạo và quản lý các nhóm trên nền tảng mạng xã hội để chia sẻ chung sở thích, sự kiện hoặc mục đích.
* Cung cấp chức năng mời người khác tham gia nhóm và quản lý thành viên trong nhóm.

**- Tạo và Quản lý Sự kiện trên Mạng xã hội:**

* Cho phép người dùng tạo và quản lý các sự kiện trên nền tảng mạng xã hội như hội thảo, buổi biểu diễn, hay sinh nhật.
* Cung cấp chức năng mời người khác tham gia sự kiện và quản lý danh sách tham dự.

**- Chức năng Theo dõi và Thông báo:**

* Cho phép người dùng theo dõi hoạt động của bạn bè, nhóm, hoặc trang mà họ quan tâm.
* Cung cấp thông báo cho người dùng khi có hoạt động mới trên trang cá nhân, bài viết hoặc nhóm của họ.

**- Chức năng Xem livestream và Video:**

* Cho phép người dùng xem livestream và video trực tiếp từ bạn bè hoặc các trang mạng xã hội.
* Hỗ trợ tính năng tương tác như bình luận và thả cảm xúc trong quá trình xem video.

**- Chức năng Thanh toán và Mua sắm :**

* Nếu có tính năng thương mại điện tử, hệ thống cần hỗ trợ tích hợp các cổng thanh toán trực tuyến an toàn và tiện lợi.
* Cho phép người dùng mua sản phẩm hoặc dịch vụ trên nền tảng mạng xã hội.

**- Chức năng Bảo mật và Quyền riêng tư Nâng cao:**

* Hỗ trợ cài đặt quyền riêng tư chi tiết cho các bài đăng, hình ảnh và thông tin cá nhân của người dùng.
* Cung cấp các tùy chọn bảo mật như xác minh hai yếu tố và quản lý hoạt động đăng nhập.

**- Chức năng Tương thích và Đa nền tảng:**

* Đảm bảo rằng nền tảng mạng xã hội hoạt động mượt mà trên các thiết bị và trình duyệt phổ biến như máy tính, điện thoại di động và máy tính bảng.
* Cung cấp ứng dụng di động cho cả iOS và Android để người dùng có thể truy cập nền tảng mạng xã hội từ bất kỳ đâu.

**- Chức năng Phân tích và Thống kê Nâng cao:**

* Hỗ trợ các công cụ phân tích và thống kê chi tiết về hoạt động người dùng, tương tác và nội dung trên nền tảng mạng xã hội.
* Cung cấp báo cáo phân tích sâu về hiệu suất và xu hướng để giúp người quản trị đưa ra quyết định chiến lược.

**1.4 Cấu trúc đồ án**

Cấu trúc đồ án gồm 4 chương :

* Chương 1. Tổng quan: Giới thiệu ngắn gọn về đề tài, nêu tóm tắt những lý thuyết, nghiên cứu trước đây có liên quan đến đề tài
* Chương 2. Cơ sở lý thuyết: Các khái niệm và phương pháp giải quyết vấn đề bao gồm mô tả các công nghệ, hệ thống, các ràng buộc hoặc các giải pháp mới, những mô hình toán, lý giải xây dựng mô hình
* Chương 3. Kết quả thực nghiệm: Mô tả công việc thực nghiệm đề tài đã tiến hành, các kết quả nghiên cứu lý thuyết, kết quả thực nghiệm đạt được.
* Chương 4. Kết luận và kiến nghị: Những kết luận chung, khẳng định những kết quả đạt được, những đóng góp, đề xuất mới và kiến nghị.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Flutter là gì?

- Flutter được Google giới thiệu là một người mới trong thế giới ứng dụng di động. Là một SDK mới của Google dành cho các thiết bị di động giúp developers và designers xây dựng nhanh chóng ứng dụng dành cho các thiết bị di động (Android, iOS). Flutter là dự án mã nguồn mở đang trong giai đoạn thử nghiệm. Flutter bao gồm Reactive framework và công nghệ hiển thị 2D (2D rendering engine) và các công cụ phát triển (development tool). Các thành phần này làm việc cùng nhau giúp ta thiết kế, xây dựng, test, debug ứng dụng. Không có gì ngạc nhiên khi Flutter giúp các nhà phát triển tạo ra các ứng dụng native đẹp mắt và giúp họ phát triển các ứng dụng đa nền tảng một cách dễ dàng. [1]

- 5 điều mô tả ngắn về Flutter:

* Ngôn ngữ Dart. Dart là gì? Đó là phản ứng của hầu hết các nhà phát triển. Dart là ngôn ngữ thuần OOP, nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động.
* Các IDE được khuyến nghị với Flutter là Android Studio hoặc Intellij Idea.
* Flutter có cầu nối là Dart, kích thước ứng dụng lớn hơn, nhưng nó hoạt động nhanh hơn nhiều. Không giống như React Native với cầu nối là Javascript.
* React Native và Flutter chia sẻ các mô hình tương tự như trình xử lý sự kiện và mở rộng lớp.
* Animation và trải nghiệm cài đặt của Flutter thực sự tốt và mượt mà tương ứng.

### 2.1.1 Cấu trúc nổi bật của Flutter

**1. Kiến trúc Widget-Based**

* **Mô hình Everything is a Widget**: Mọi thứ trong Flutter, từ nút bấm, giao diện, đến toàn bộ màn hình, đều được định nghĩa là widget.
* **Widgets Cây (Widget Tree)**: Flutter sử dụng một cây cấu trúc widget để tổ chức giao diện. Các widget có thể được lồng vào nhau để xây dựng giao diện phức tạp.
* **Stateless vs Stateful Widgets**:
  + **Stateless Widget**: Không thay đổi trạng thái trong suốt vòng đời của widget.
  + **Stateful Widget**: Có thể thay đổi trạng thái (state) và tự động cập nhật giao diện.



Hình 1. Kiến trúc Widget-Based

**2. Kiến trúc Render Tree**

* **Layered Architecture**: Giao diện được xây dựng dựa trên kiến trúc nhiều lớp (Layers):
  + **Framework Layer**: Gồm các widget dựng sẵn (Material, Cupertino).
  + **Engine Layer**: Gồm thư viện đồ họa Skia để render UI.
  + **Embedder Layer**: Kết nối với nền tảng gốc như iOS hoặc Android.
* **Render Tree**: Flutter tạo một render tree song song với widget tree để tối ưu hóa việc vẽ giao diện.



Hình 2. Kiến trúc Render Tree

**3. Hot Reload & Hot Restart**

* **Hot Reload**: Cho phép nhà phát triển thấy thay đổi tức thì trong giao diện mà không cần khởi động lại toàn bộ ứng dụng.
* **Hot Restart**: Khởi động lại ứng dụng nhanh chóng để áp dụng các thay đổi lớn.



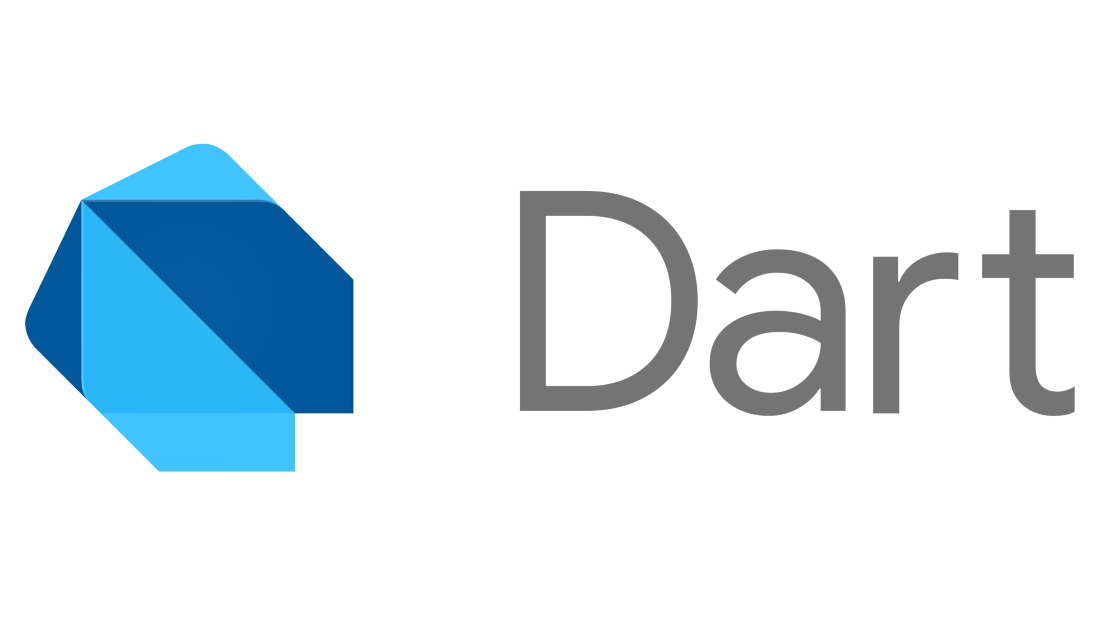
Hình 3. Hot Reload & Hot Restart

**4. Hệ Thống Layout Hiệu Quả**

* Flutter sử dụng hệ thống layout dựa trên **Constraints**:
  + **Parent-Child Constraints**: Các widget con nhận ràng buộc từ widget cha để xác định kích thước và vị trí.
  + **Flexible Layouts**: Hỗ trợ bố cục linh hoạt như Expanded, Flexible, Stack, GridView.

**5. Dart Programming Language**

* Flutter được viết bằng ngôn ngữ **Dart**:
  + **Ahead-of-Time Compilation (AOT)**: Tăng tốc độ chạy ứng dụng.
  + **Just-in-Time Compilation (JIT)**: Hỗ trợ phát triển nhanh với Hot Reload.



Hình 4. Dart Programming Language

**6. Hỗ Trợ Cross-Platform**

* **Native-like Performance**: Giao diện được render trực tiếp thông qua Skia mà không cần qua cầu nối với native (như React Native).
* **Viết Một Lần, Chạy Mọi Nơi**: Dùng chung mã nguồn cho Android, iOS, web, và desktop.



Hình 5. Cross-Platform

**7. Thư Viện Mạnh Mẽ**

* **Material Design và Cupertino**: Cung cấp các widget dựng sẵn theo phong cách Material Design (Google) và Cupertino (iOS).
* **Package Ecosystem**: Hỗ trợ nhiều thư viện và package từ cộng đồng để mở rộng tính năng.

### 2.1.2 Widget trong Flutter

**1. Stateless Widget:** không thay đổi trạng thái trong suốt vòng đời.

**2. Stateful Widget:** có thể thay đổi trạng thái, ví dụ: nút bấm, form nhập liệu.

**3. Layout** **Widgets:** dùng để tổ chức và định vị các widget con.

- Single-child layout(chỉ có một widget con):

* Container: gói các widget và hỗ trợ syling.
* Center: canh giữa widget.
* Padding: thêm khoảng cách xung quanh widget.

- Multi – child Layout (Nhiều widget con):

* Column: bố trí các widget theo chiều dọc.
* Row: bố trí các widget theo chiều ngang.
* Stack: xếp chồng các widget lên nhau.
* Listview: Danh sách cuộn.

**4. Interactive Widgets**

- Các widget hỗ trợ tương tác của người dùng:

* Button: như ElvatedButton, TextButton, IconButton.
* TextField: widget nhập liệu.
* GestureDetector: Nhận diện các thao tác (vuốt, nhấn, kéo).

**5. Styling Widgets**

- Dùng để tùy chỉnh giao diện:

* Text: hiển thị văn bản với nhiều tùy chọn (màu sắc, kiểu chữ).
* Image: Hiển thị hình ảnh từ mạng hoặc tài nguyên cục bộ.
* Icon: Hiển thị biểu tượng.

**6. Material Design Widgets**

- Dựa trên phong cách thiết kế Material Design:

* Scaffold: cấu trúc cơ bản của ứng dụng (AppBar, body, drawer).
* Appbar: thanh công cụ trên cùng.
* BottomNavigationBar: thanh điều hướng phía dưới.

**7. Cupertino Widgets**

- Dành cho giao diện phong cách IOS:

* CupertinoButton.
* CupertinoNavigationBar.
* CupertinoTextField.

### 2.1.3 Ưu điểm và nhược điểm Flutter

**- Ưu điểm:**

* Ngôn ngữ Dart: Dart - OOP, nó khá dài, nhưng những người quen viết bằng Java / C # sẽ hài lòng với Dart.
* Ít phải viết các config khi tạo project mới theo cách thủ công, ngược lại với React Native.
* Chỉ cần tải về Flutter từ git, chạy “flutter doctor”, và nó sẽ cho bạn biết những loại vấn đề bạn có thể có trong hệ thống. Sau đó, tạo một project từ Android Studio hoặc bất kỳ IDE nào sau khi cài đặt Flutter Plug-in sơ bộ và start project. Ngoài ra nó còn có Hot-reload giống như trong React-Native.
* Hỗ trợ tuyệt vời cho IDE (Android Studio, IDE IntelliJ, VSCode)
* Trình điều hướng tích hợp sẵn. Sau khi tạo ra một ứng dụng trên Flutter, bạn có thể thấy nó có Navigator được xây dựng sẵn. Bạn có thể tạo một route mới mà không cần kết nối bất kỳ thứ gì (không giống như trong RN, nơi bạn cần kết nối bất kỳ package điều hướng gốc nào vv);
* Giải pháp quản lý dữ liệu được tạo sẵn, ví dụ trong Flutter này là "Mọi thứ đều có một Widget" cho nó. Có một StatefulWidget đặc biệt, với sự giúp đỡ của nó mà chúng ta có thể quản lý trạng thái widget và thay đổi nó một cách năng động.

**- Nhược điểm:**

* Có một rào cản lớn cho những người chỉ học JS, hoặc đến từ RN. Do thực tế là Dart được sử dụng với sự thừa kế, đa hình của nó và tất cả các thứ về OOP.
* Không có JSX mà tất cả chúng ta đã quen khi làm RN. Nhìn vào tệp Dart trong đó không có phân chia thành template, styling và data, nó trở nên khó chịu.
* Styling là một nhược điểm. Thực tế là không có sự tách biệt thành Styles, templates và controller, có một vấn đề khi mô tả một thành phần thì chúng ta cũng cần mô tả các styles cùng một lúc.
* Animation còn khó khăn hơn. Mặc dù nó có một animation tốt, nhưng nó sẽ khó hơn khi tạo hiệu ứng động, không giống như trong RN.
* Tối ưu hóa. Trong Fluter, chỉ có thừa kế widget với các điều kiện có thể thay đổi (trạng thái) và không thể thay đổi (không trạng thái). Trong khi ở React-Native chúng ta có thể quản lý vòng đời. Ngoài ra, một nhược điểm lớn là thiếu các công cụ để lưu trạng thái của ứng dụng, vấn đề này có thể, tuy nhiên, được giải quyết bằng cách tuần tự hóa trạng thái hiện tại.



Hình 6. Flutter

## 2.2 Spring Boot là gì?

- Spring Boot là một framework Java được sử dụng để xây dựng các ứng dụng và dịch vụ web dễ dàng và nhanh chóng. Nền tảng cung cấp các cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based.

[2]



Hình 7. Spring Boot

### 2.2.1 Spring Boot và các đặc điểm chính

- Spring Boot là một framework mạnh mẽ được xây dựng trên Spring Framework, giúp phát triển các ứng dụng Java một cách nhanh chóng và dễ dàng. Dưới đây là các đặc điểm chính của Spring Boot:

**1. Auto Configuration (Cấu hình tự động)**

* Spring Boot tự động cấu hình ứng dụng dựa trên các thư viện được thêm vào project.
* Ví dụ: khi thêm thư viện spring – boot – started – web, Spring Boot tự động cấu hình Tomcat và DispatcherServlet.
* Giảm thiểu công việc cấu hình thủ công trong file XML hoặc Java Config.

**2. Starter Dependencies (Phụ thuộc khởi động)**

- Spring Boot cung cấp các gói phụ thuộc sẵn (starters) để dễ dàng tích hợp các chức năng phổ biến.

- Ví dụ:

* Spring – boot – starter – web: cho ứng dụng web.
* Spring – boot – starter – data – jpa: cho ứng dụng với JPA và Hibernate.
* Spring – boot – starter – security: cho bảo mật.

- Giúp đơn giản hóa việc quản lý các thư viện phụ thuộc.

**3. Embedded Servers (Máy chủ nhúng)**

* Spring Boot hỗ trợ các máy chủ nhúng như Tomcat, Jetty, hoặc Undertow.
* Không cần cài đặt máy chủ riêng biệt, ứng dụng có thể chạy độc lập.
* Thích hợp cho việc phát triển và triển khai nhanh.

**4. Opinionated Defaults (Mặc định có định hướng)**

* Cung cấp các cấu hình mặc định tối ưu, giúp bạn tập trung vào phát triển logic thay vì cấu hình.
* Bạn có thể ghi đè các cấu hình mặc định nếu cần.

**5. Spring Boot CLI (Command – Line Interface)**

* Công cụ dòng lệnh hỗ trợ phát triển ứng dụng Spring Boot bằng cách sử dụng Groovy scripts.
* Ví dụ: tạo ứng dụng Spring Boot nhanh chỉ với vài dòng Groovy.

**6. Actuator (Giám sát và Quản lý)**

- Spring Boot Actuator cung cấp các endpoint để giám sát và quản lý ứng dụng, chẳng hạn:

* /actuator/health: trạng thái sức khỏe ứng dụng.
* /actuator/metrics: thống kê hiệu suất và số liệu ứng dụng.

- Dễ dàng tích hợp với các công cụ giám sát như Prometheus, Grafana.

**7. Spring Initializr**

* Công cụ trực tuyến giúp bạn khởi tạo nhanh một dự án Spring Boot.
* Tùy chỉnh phụ thuộc, phiên bản Java và các thông số khác.

**8. DevTools (Hỗ trợ phát triển)**

* Spring Boot DevTools cung cấp các tính năng hỗ trợ phát triển:
* **Tự động khởi động lại (LiveReload)**: Tự động tải lại ứng dụng khi có thay đổi mã nguồn.
* **Caching Disable**: Vô hiệu hóa bộ nhớ cache trong môi trường phát triển.

**9. Profiles (Hồ sơ cấu hình)**

* Hỗ trợ nhiều cấu hình môi trường thông qua các profile, ví dụ:
  + application-dev.properties cho môi trường phát triển.
  + application-prod.properties cho môi trường sản xuất.
* Cấu hình profile dễ dàng qua tham số.

**10. Hỗ trợ Microservices**

- Spring Boot được thiết kế để dễ dàng phát triển microservices với các tính năng:

* Tích hợp Spring Cloud.
* Cấu hình tập trung.
* Khả năng mở rộng dễ dàng.

**11. Security (Bảo mật)**

- Cung cấp tích hợp với Spring Security để bảo vệ ứng dụng:

* Authentication và Authorization.
* Mã hóa mật khẩu.

**12. Testing (Kiểm thử)**

- Spring Boot hỗ trợ kiểm thử mạnh mẽ với các thư viện tích hợp:

* JUnit, Mockito.
* Spring Boot Test: Giúp kiểm thử các thành phần ứng dụng dễ dàng.

### 2.2.2 Cấu trúc một ứng dụng Spring Boot

- Spring Boot sử dụng cấu trúc thư mục chuẩn Maven/Gradle, giúp tổ chức mã nguồn dễ dàng và tuân thủ các nguyên tắc phát triển ứng dụng Java. Dưới đây là cấu trúc cơ bản của một ứng dụng Spring Boot:

**1. Thư mục gốc của dự án**

- Đây là nơi chứa các tệp cấu hình chính và các công cụ build:

* **pom.xml (Maven)**: File cấu hình các dependencies và plugin.
* **build.gradle (Gradle)**: File cấu hình tương tự nhưng dành cho Gradle.
* **application.properties hoặc application.yml**: File cấu hình ứng dụng.
* **src**: Thư mục chứa mã nguồn chính.

**2. Thư mục chính của ứng dụng (src/main)**

**\*Thư mục java:**

* Chứa mã nguồn chính của ứng dụng.
* Ví dụ cấu trúc thư mục:

src/main/java/com/example/demo/

* Trong đó:
* com.example.demo là package gốc của ứng dụng.
* Các thành phần chính:
  + - **Controller**: Xử lý yêu cầu HTTP từ người dùng.
    - **Service**: Chứa logic nghiệp vụ.
    - **Repository**: Tương tác với cơ sở dữ liệu.
    - **Model/Entity**: Định nghĩa các đối tượng dữ liệu.

**\*File Main Class:**

* Lớp chính có annotation @SpringBootApplication.
* Đây là điểm khởi đầu của ứng dụng Spring Boot.

**\*Thư mục resources:**

- Chứa tài nguyên của ứng dụng, bao gồm:

**Cấu hình ứng dụng**:

* application.properties hoặc application.yml: Cấu hình cơ sở dữ liệu, cổng chạy server, profile, v.v.

**File tĩnh**:

* static/: Chứa file tĩnh (CSS, JS, hình ảnh).
* templates/: Chứa file giao diện động (Thymeleaf, FreeMarker).

**Cơ sở dữ liệu**:

* data.sql: Chứa dữ liệu mẫu.
* schema.sql: Cấu trúc bảng.

**Thông tin khác**:

* banner.txt: File chỉnh sửa banner khi ứng dụng khởi động.

**3. Thư mục kiểm thử (src/test)**

- Chứa các bài kiểm thử cho ứng dụng.

- Cấu trúc thư mục:

src/test/java/com/example/demo/

- Ví dụ:

* Kiểm thử đơn vị (Unit Test) cho các lớp Service.
* Kiểm thử tích hợp (Intergration Test) với MockMVC.

**4. Các điểm quan trọng cần lưu ý**

* Tổ chức thư mục theo đúng mô hình MVC (Model-View-Controller).
* Cấu hình ứng dụng rõ ràng trong file application.properties.
* Tách biệt logic nghiệp vụ (Service) và xử lý dữ liệu (Repository) để dễ bảo trì.
* Cấu trúc kiểm thử rõ ràng để đảm bảo chất lượng mã nguồn.

### 2.2.3 Dependency Injection trong Spring Boot

- Dependency Injection (DI) là một mẫu thiết kế (design pattern) được sử dụng rộng rãi trong Spring Framework, giúp quản lý và cung cấp các phụ thuộc (dependencies) giữa các đối tượng một cách tự động. DI giúp tăng tính module hóa, dễ bảo trì và kiểm thử trong ứng dụng.

**1. Nguyên tắc hoạt động của Dependency Injection**

**- Ý tưởng chính**: Các đối tượng không tự tạo hoặc quản lý phụ thuộc của chính nó mà sẽ được cấp (injected) từ bên ngoài bởi **Spring IoC Container**.

**- IoC Container**: Là thành phần quản lý vòng đời và các phụ thuộc của đối tượng.

**2. Cách thực hiện Dependency Injection**

**\* Annotation chính để triển khai DI**

Spring Boot sử dụng các annotation để quản lý DI dễ dàng:

1. **@Component**: Đánh dấu một class là thành phần (bean) do Spring quản lý.
2. **@Service**: Dùng để chỉ định một class xử lý logic nghiệp vụ.
3. **@Repository**: Đánh dấu một class tương tác với cơ sở dữ liệu.
4. **@Controller**: Đánh dấu một class làm nhiệm vụ xử lý các request từ client.
5. **@Autowired**: Gắn vào các trường (field), constructor, hoặc setter để yêu cầu phụ thuộc.
6. **@Bean**: Định nghĩa một bean thủ công trong file cấu hình.

**\* Các cách Dependency Injection**

Spring Boot hỗ trợ ba cách để inject phụ thuộc:

**-Cách 1.Constructor Injection**:

* + Inject thông qua constructor của class.
  + **Ưu điểm**:
    - Hỗ trợ tính bất biến.
    - Thân thiện với kiểm thử.

**-Cách 2.Setter injection:** inject thông qua các phương thức setter.

-**Cách 3.Field Injection:** Inject trực tiếp vào các trường (field).

**3. Bean LifeCycle và Scopes**

**Lifecycle: IoC Container sẽ:**

1. Tạo bean.
2. Inject phụ thuộc.
3. Khởi tạo (initialize) và hủy (destroy) bean.

**Scopes: Xác định phạm vi tồn tại của bean:**

* @Scope("singleton"): (Mặc định) Một instance duy nhất.
* @Scope("prototype"): Mỗi lần yêu cầu sẽ tạo một instance mới.
* @Scope("request"): Một instance cho mỗi request HTTP.

**4. Ưu điểm của Dependency Injection**

- **Giảm kết nối chặt chẽ (Tight Coupling)**:

* Các class không tự quản lý phụ thuộc, dễ dàng thay đổi hoặc mở rộng.

- **Dễ kiểm thử (Unit Testing)**:

* Inject các mock dependencies để kiểm thử độc lập.

- **Tăng khả năng tái sử dụng (Reusability)**:

* Dễ dàng cấu hình lại các bean và sử dụng lại trong nhiều module.

- **Quản lý lifecycle tốt hơn**:

* Spring IoC Container xử lý tự động việc tạo, khởi tạo và hủy các đối tượng.

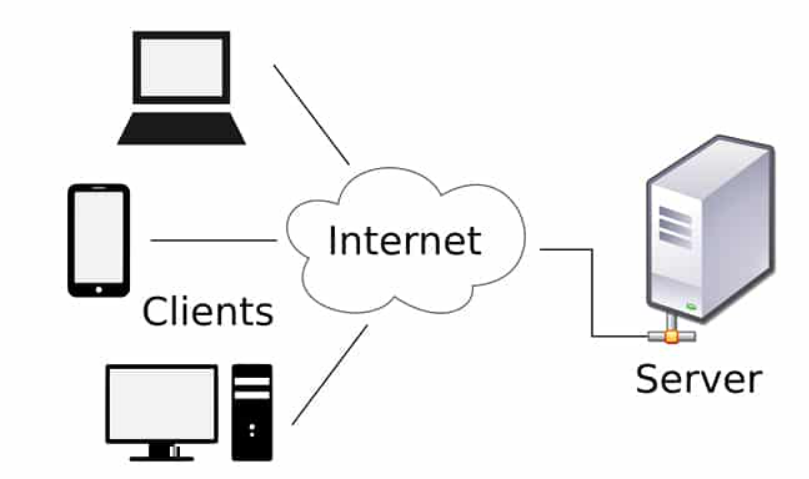
**5. Một số lưu ý**

* Constructor Injection nên được ưu tiên để đảm bảo tính bất biến và dễ kiểm thử.
* Tránh sử dụng Field Injection trực tiếp, vì khó khăn trong kiểm thử và bảo trì.
* Cẩn thận với @Scope("prototype") vì Spring chỉ quản lý lifecycle khi khởi tạo.

## 2.3 Tích hợp Flutter với Spring Boot

### 2.3.1 Kiến trúc Client-Server

* Flutter (Client): Ứng dụng Flutter sẽ đóng vai trò là client. Nó sẽ gửi yêu cầu (requests) đến server và nhận phản hồi (responses). Flutter chịu trách nhiệm xử lý giao diện người dùng (UI) và logic liên quan đến UI.
* Spring Boot (Server): Ứng dụng Spring Boot đóng vai trò là server. Nó cung cấp các REST API endpoints để Flutter client có thể tương tác. Spring Boot xử lý logic nghiệp vụ, truy cập cơ sở dữ liệu, và trả về dữ liệu cho client.



Hình 8. Kiến trúc Client-Server

### 2.3.2 Giao tiếp giữa Flutter và Spring Boot qua REST API

* REST API: Là tiêu chuẩn phổ biến cho việc giao tiếp giữa client và server. REST API sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên tài nguyên.
* Spring Boot (Server-side):
* Sử dụng @RestController để tạo các API endpoints.
* Sử dụng @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping để định nghĩa các phương thức HTTP tương ứng với các hành động cụ thể.
* Sử dụng @RequestBody để nhận dữ liệu từ client gửi lên.
* Sử dụng @PathVariable để lấy tham số từ URL.
* Sử dụng @RequestParam để lấy tham số từ query string.
* Trả về dữ liệu dưới dạng JSON (mặc định) hoặc XML thông qua các đối
* tượng Java (Spring Boot tự động serialize/deserialize sang JSON/XML).
* Flutter (Client-side):
* Sử dụng thư viện http (cần thêm vào pubspec.yaml và chạy flutter pub get) để thực hiện các HTTP request.
* Sử dụng các phương thức http.get(), http.post(), http.put(), http.delete() để tương tác với các API endpoints.
* Gửi dữ liệu lên server (nếu cần) bằng cách thêm vào body của request (thường là JSON).
* Nhận dữ liệu phản hồi từ server và xử lý.
* Sử dụng jsonDecode() để chuyển đổi chuỗi JSON thành đối tượng Dart.



Hình 9. Giao tiếp giữa Flutter và Spring Boot qua REST API

### 2.3.3 Cách thức gửi và nhận dữ liệu

**a) Gửi dữ liệu từ Flutter đến Spring Boot:**

* **JSON:** Là định dạng dữ liệu phổ biến nhất trong REST API.
* **Flutter:**
  + Chuẩn bị dữ liệu cần gửi dưới dạng Map hoặc List.
  + Sử dụng jsonEncode() để chuyển đổi Map hoặc List thành chuỗi JSON.
  + Đưa chuỗi JSON vào body của HTTP request (thường dùng cho POST, PUT).
* **Spring Boot:** **Sử** dụng @RequestBody để nhận dữ liệu JSON từ client và map vào đối tượng Java (DTO - Data Transfer Object).

**b) Nhận dữ liệu từ Spring Boot về Flutter:**

* **JSON:** Spring Boot thường trả về dữ liệu dưới dạng JSON.
* **Spring Boot:** Trả về đối tượng Java (DTO hoặc Entity) từ các API endpoint. Spring Boot sẽ tự động serialize đối tượng này sang JSON.
* **Flutter:**
* Nhận chuỗi JSON từ response.body.
* Sử dụng jsonDecode() để chuyển đổi chuỗi JSON thành Map hoặc List.
* Xử lý dữ liệu nhận được.



Hình 10. JSON

* dàng phục vụ cho nhiều nhu cầu phát triển.
* Sự cô lập: Cô lập dự án ngăn ngừa xung đột và cho phép phát triển sạch sẽ và có tổ chức.
* Tính di động: Khả năng di chuyển môi trường xung quanh mà không gặp vấn đề gì là một lợi thế đáng kể.

## 2.4 MySQL



Hình 11. MySQL

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến, được phát triển bởi MySQL AB vào năm 1995 và hiện thuộc sở hữu của Oracle Corporation. MySQL sử dụng ngôn ngữ truy vấn SQL (Structured Query Language) để quản lý và thao tác dữ liệu. Đây là lựa chọn hàng đầu cho các ứng dụng web, doanh nghiệp, và hệ thống lưu trữ dữ liệu nhờ vào hiệu suất, độ tin cậy và dễ sử dụng.

**Đặc điểm chính:**

* **Mã nguồn mở:** MySQL được phát hành dưới giấy phép GPL (GNU General Public License), nghĩa là nó miễn phí để sử dụng, sửa đổi và phân phối.
* **Quan hệ:** Dữ liệu được tổ chức thành các bảng có cấu trúc, với các mối quan hệ được định nghĩa giữa các bảng.
* **Ngôn ngữ truy vấn có cấu trúc (SQL):** SQL là ngôn ngữ tiêu chuẩn để tương tác với cơ sở dữ liệu quan hệ, cho phép người dùng truy vấn, cập nhật, và quản lý dữ liệu.
* **Hiệu năng cao:** MySQL được tối ưu hóa cho tốc độ và hiệu quả, có thể xử lý khối lượng dữ liệu lớn và lưu lượng truy cập cao.
* **Khả năng mở rộng:** MySQL có thể mở rộng để đáp ứng nhu cầu của các ứng dụng có quy mô lớn.
* **Bảo mật:** MySQL cung cấp nhiều tính năng bảo mật để bảo vệ dữ liệu, bao gồm xác thực người dùng, mã hóa dữ liệu, và kiểm soát truy cập.
* **Đa nền tảng:** MySQL có thể chạy trên nhiều hệ điều hành, bao gồm Windows, Linux, macOS, và Solaris.
* **Cộng đồng lớn:** MySQL có một cộng đồng người dùng và nhà phát triển lớn, cung cấp hỗ trợ, tài liệu, và các công cụ bổ sung.

**Các tính năng nổi bật:**

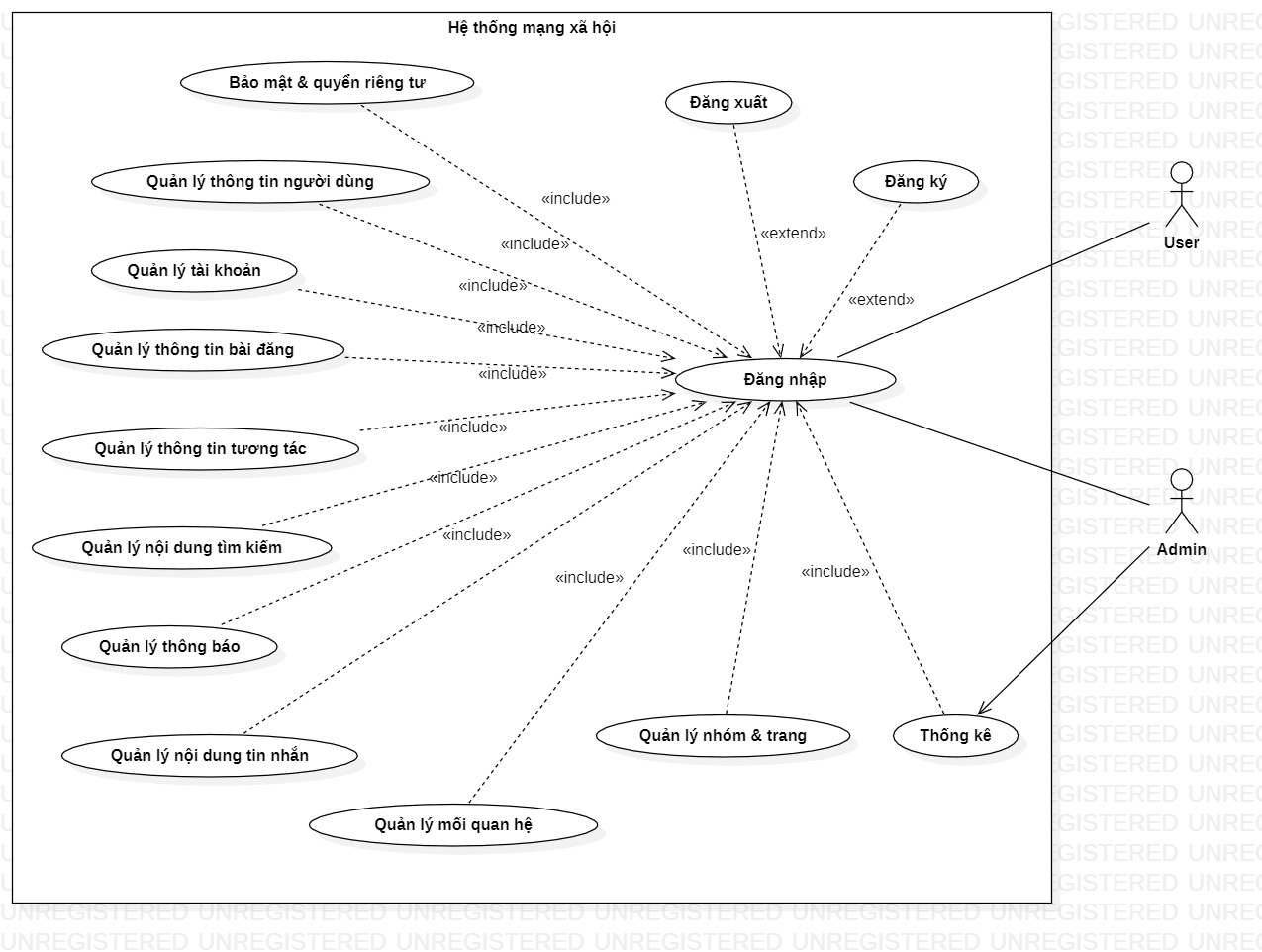
* Lưu trữ dữ liệu: Hỗ trợ nhiều kiểu dữ liệu khác nhau, bao gồm số, chuỗi, ngày tháng, và dữ liệu nhị phân.
* Truy vấn dữ liệu: Cung cấp khả năng truy vấn dữ liệu phức tạp bằng cách sử dụng SQL.
* Chỉ mục (Indexing): Cho phép tạo chỉ mục trên các cột để tăng tốc độ truy vấn.
* Sao lưu và phục hồi: Cung cấp các công cụ để sao lưu và phục hồi cơ sở dữ liệu.
* Replication: Cho phép sao chép dữ liệu sang các máy chủ khác để tăng tính khả dụng và cân bằng tải.
* Stored Procedures, Triggers, and Views: Hỗ trợ các thủ tục lưu trữ, trigger và view để tự động hóa các tác vụ và đơn giản hóa truy vấn.

# Chương 3: KẾT QUẢ THỰC NGHIỆM

## 3.1 Phân tích và thiết kế hệ thống

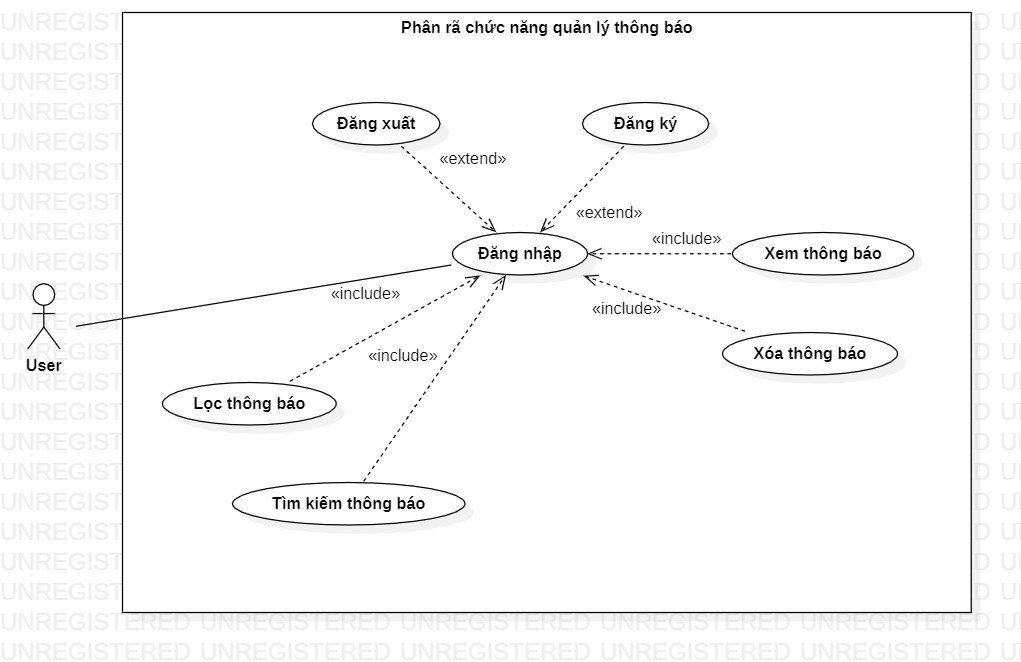
### 3.1.1 Sơ đồ usecase

- Usecase tổng quát:



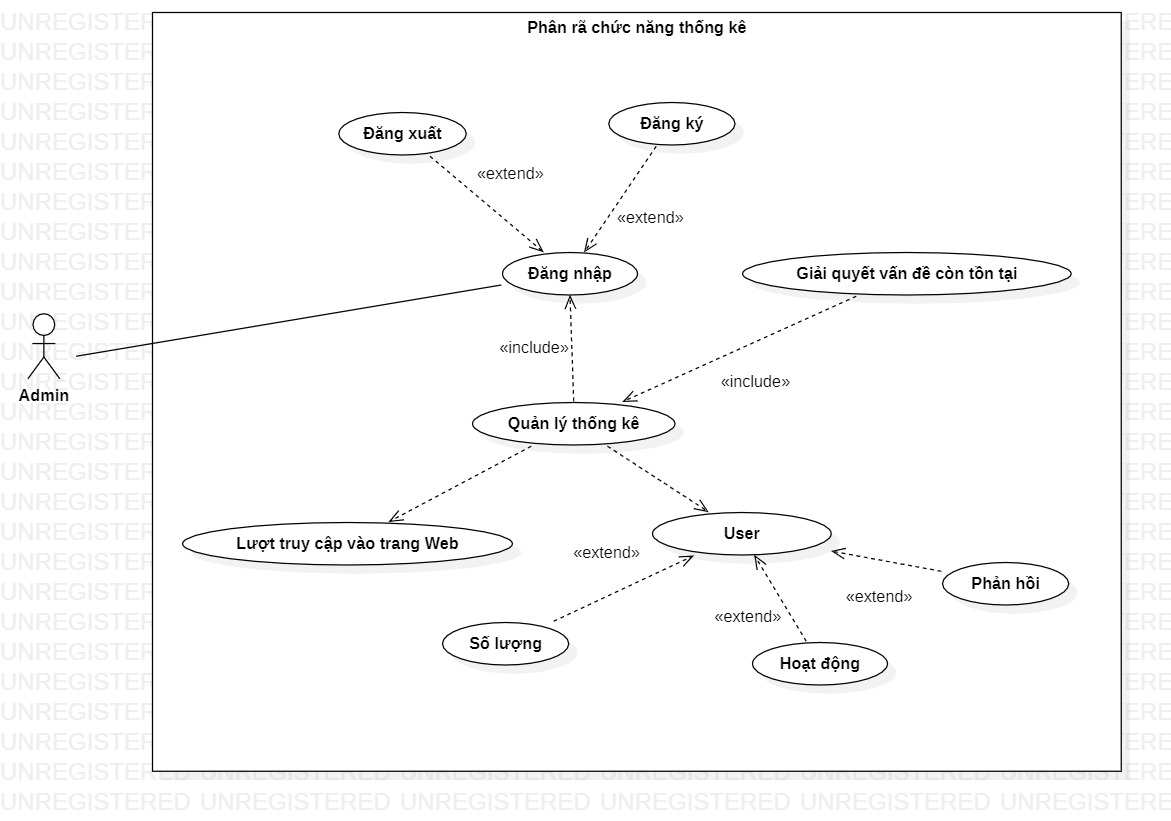
Hình 12. Usecase tổng quát

- Usecase thông báo:



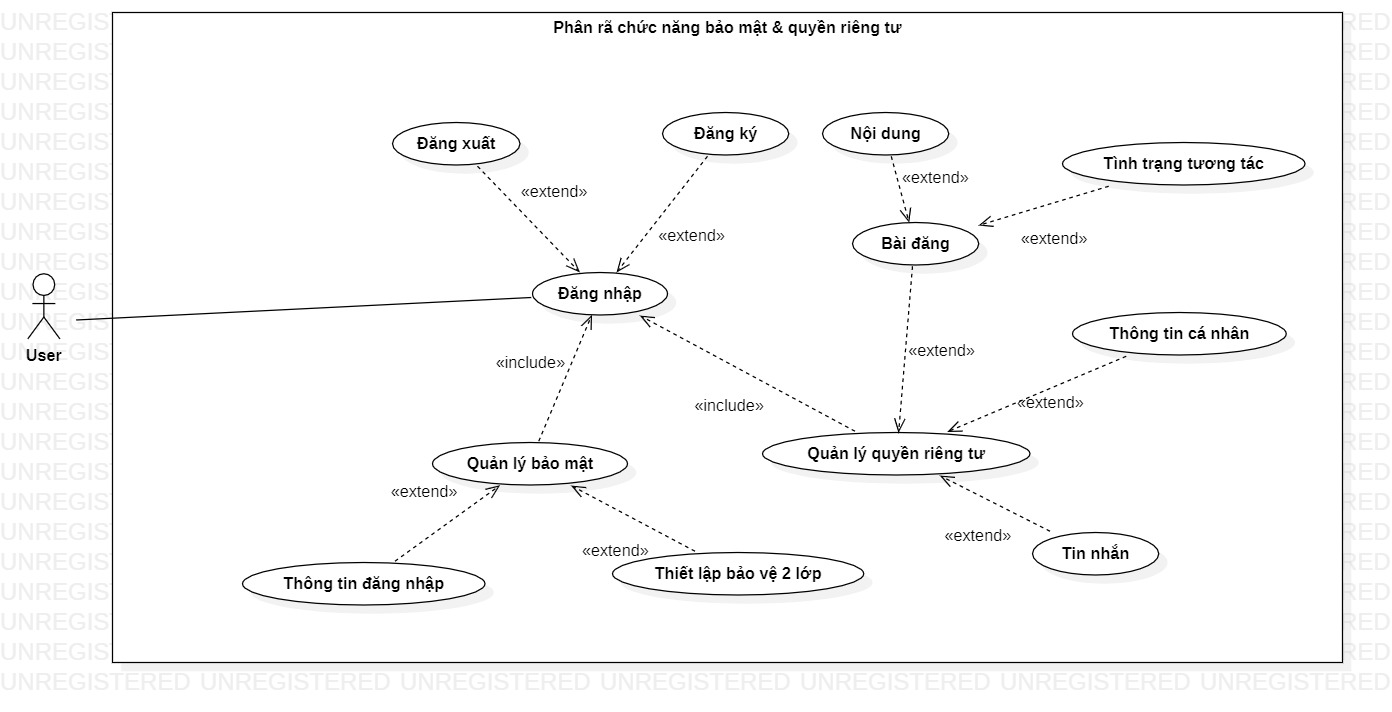
Hình 13. Usecase thông báo

- Usecase thống kê:



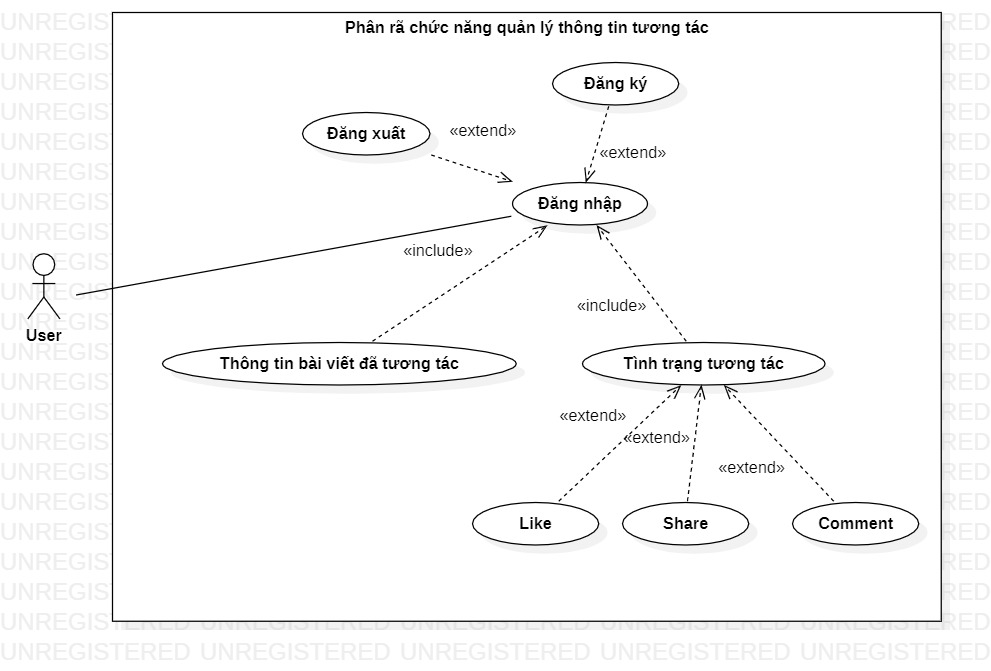
Hình 14. Usecase thống kê

- Usecase bảo mật:



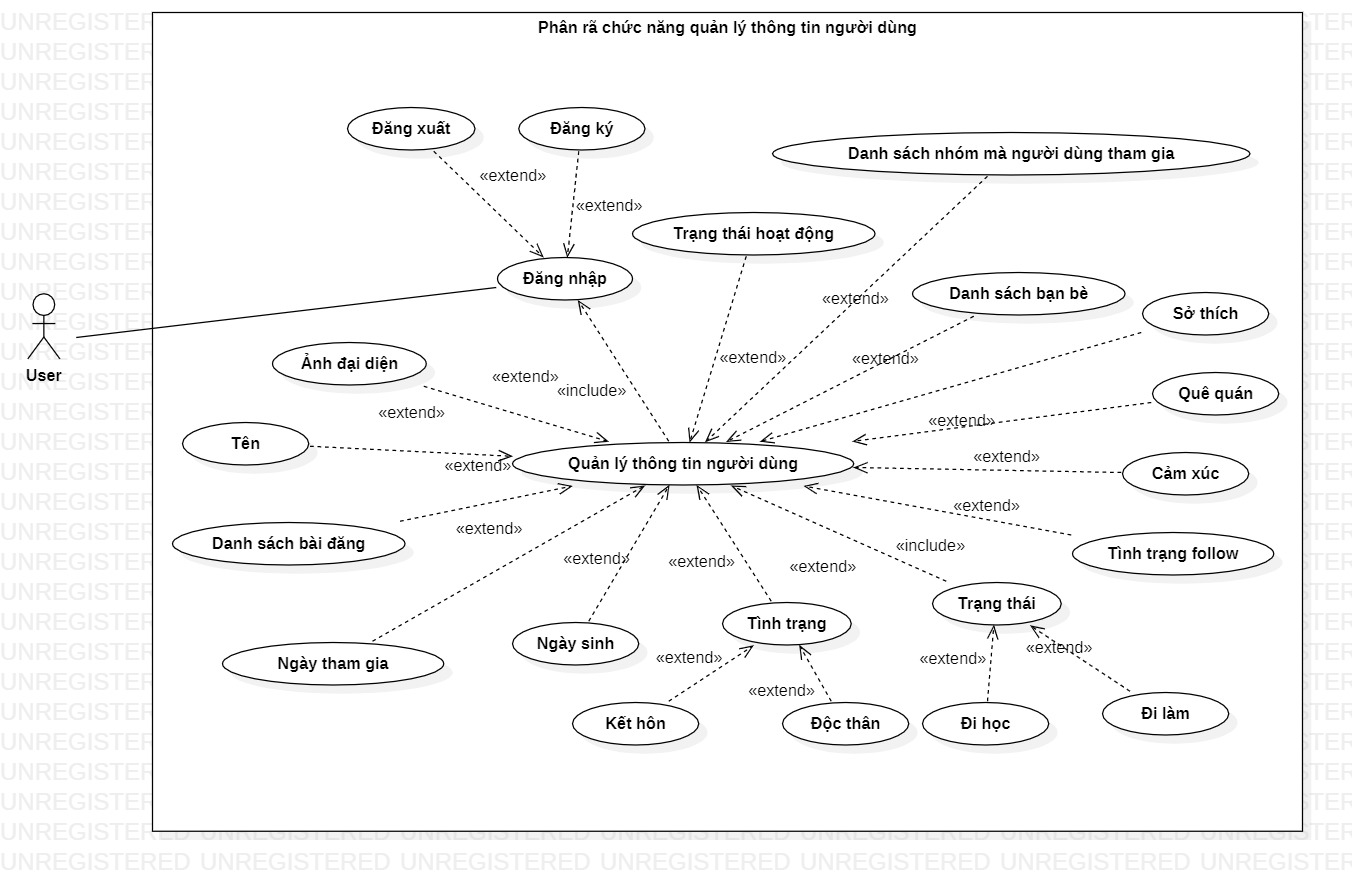
Hình 15. Usecase bảo mật

- Usecase quản lý tương tác:



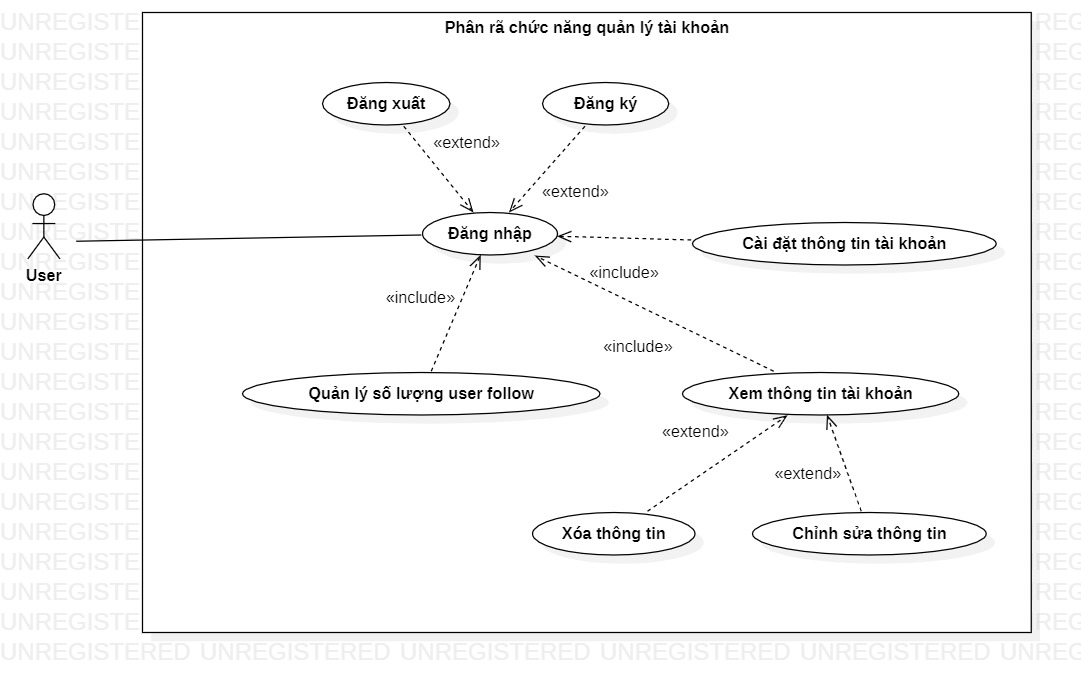
Hình 16. Usecase quản lý tương tác

- Usecase quản lý thông tin người dùng:



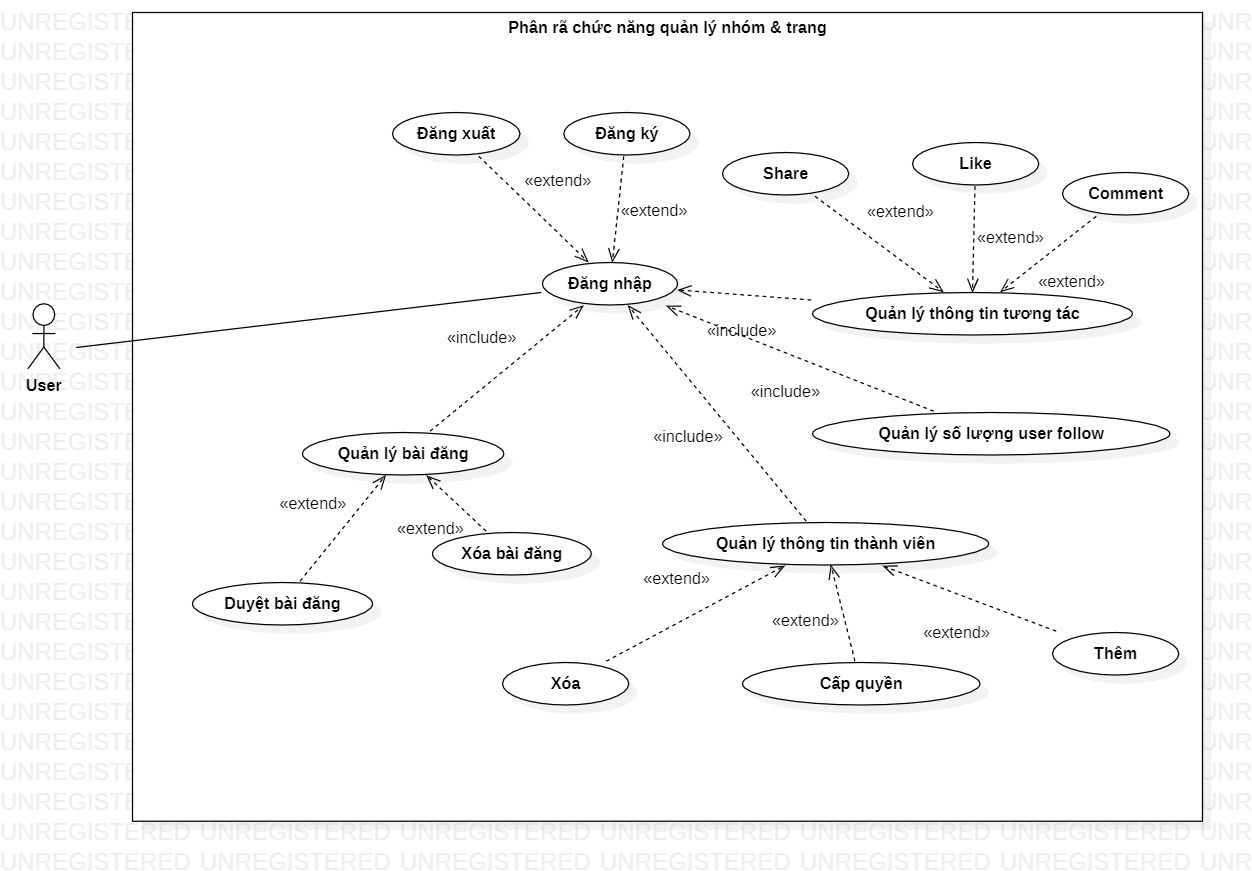
Hình 17. Usecase quản lý thông tin người dùng

- Usecase quản lý tài khoản:



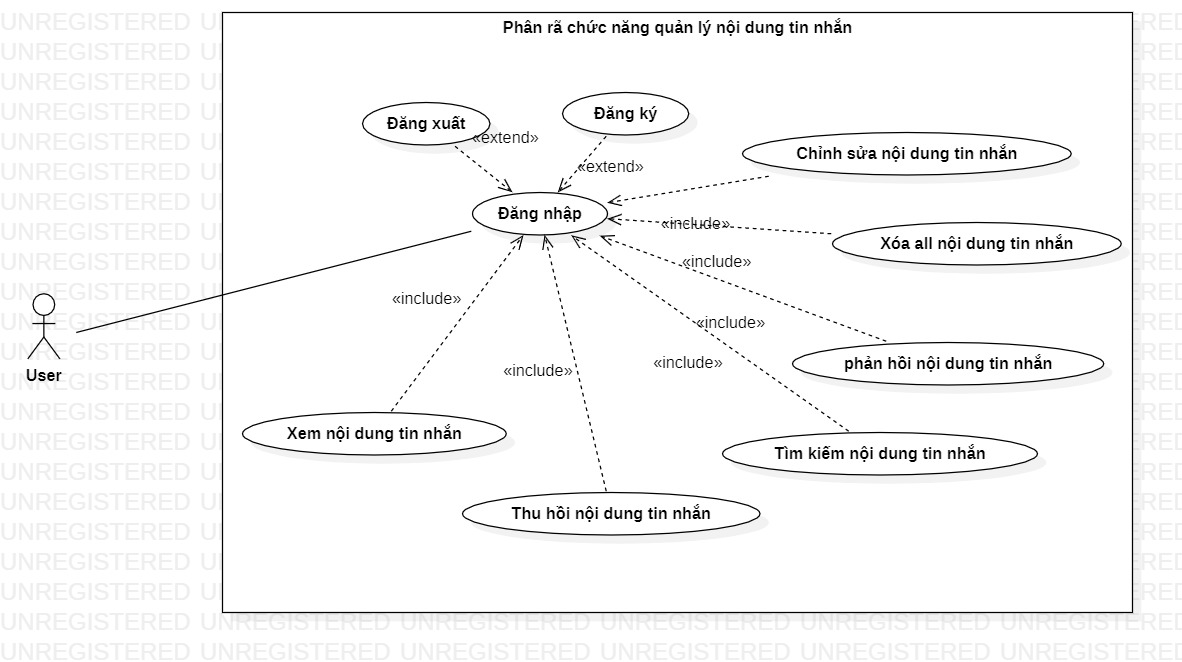
Hình 18. Usecase quản lý tài khoản

- Usecase quản lý nhóm & trang:



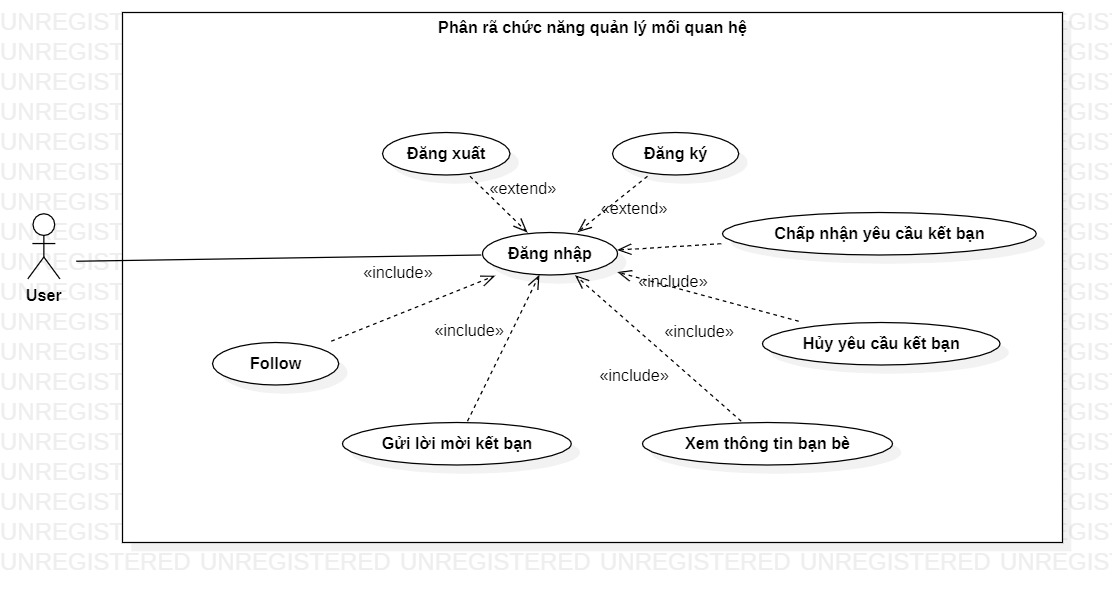
Hình 19. Usecase quản lý nhóm & trang

- Usecase quản lý nội dung chat:



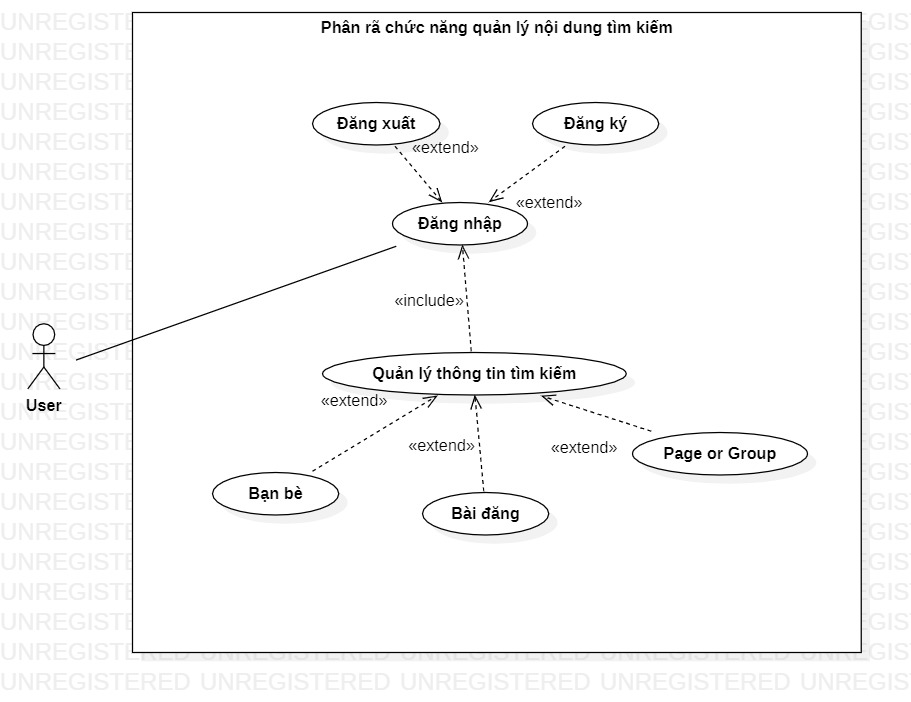
Hình 20. Usecase quản lý nội dung chat

- Usecase quản lý mối quan hệ:



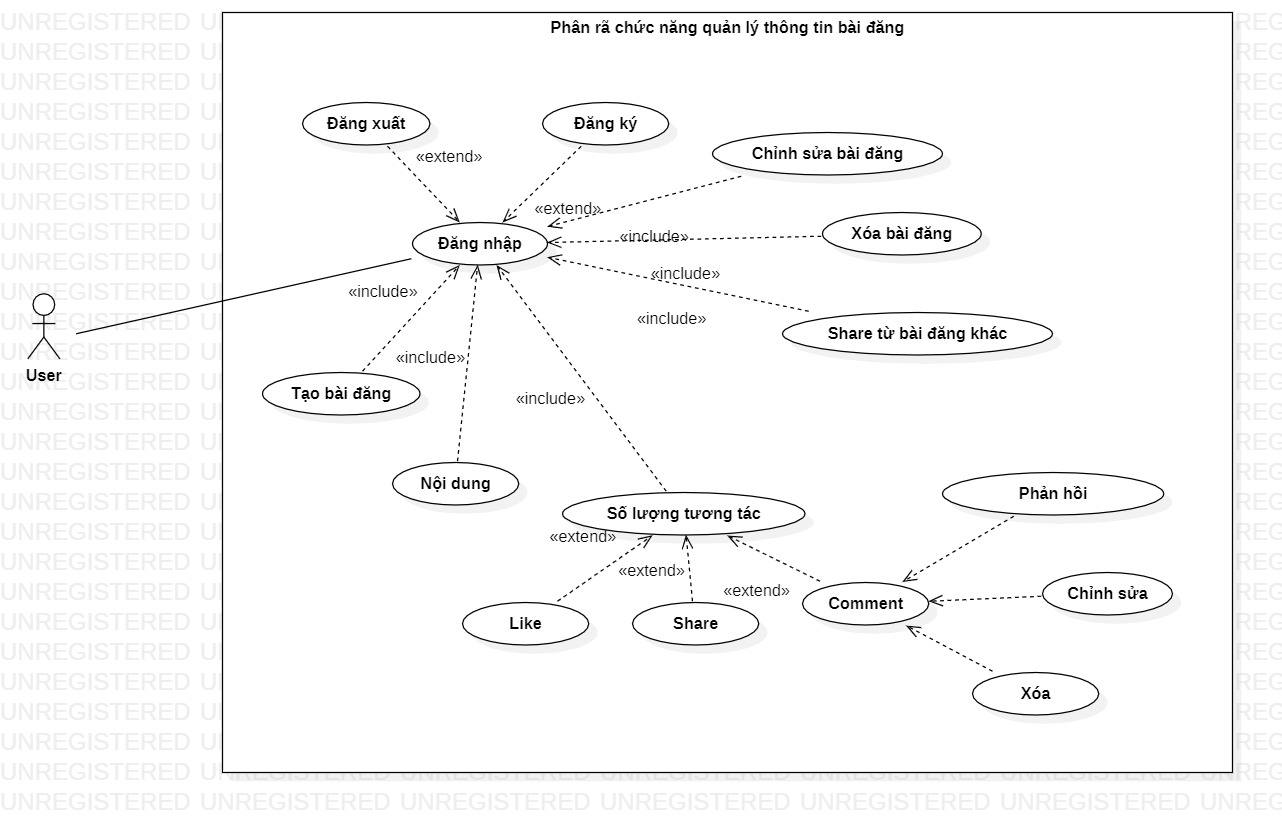
Hình 21. Usecase quản lý mối quan hệ

- Usecase quản lý nội dung tìm kiếm:



Hình 22. Usecase quản lý nội dung tìm kiếm

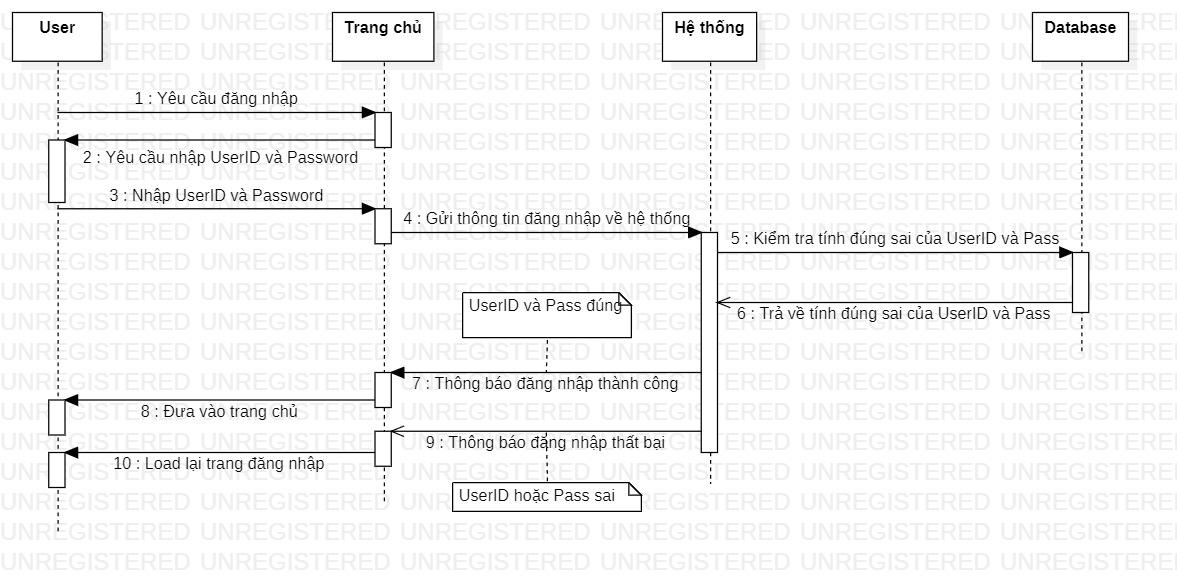
- Usecase quản lý thông tin bài đăng:



Hình 23. Usecase quản lý thông tin bài đăng

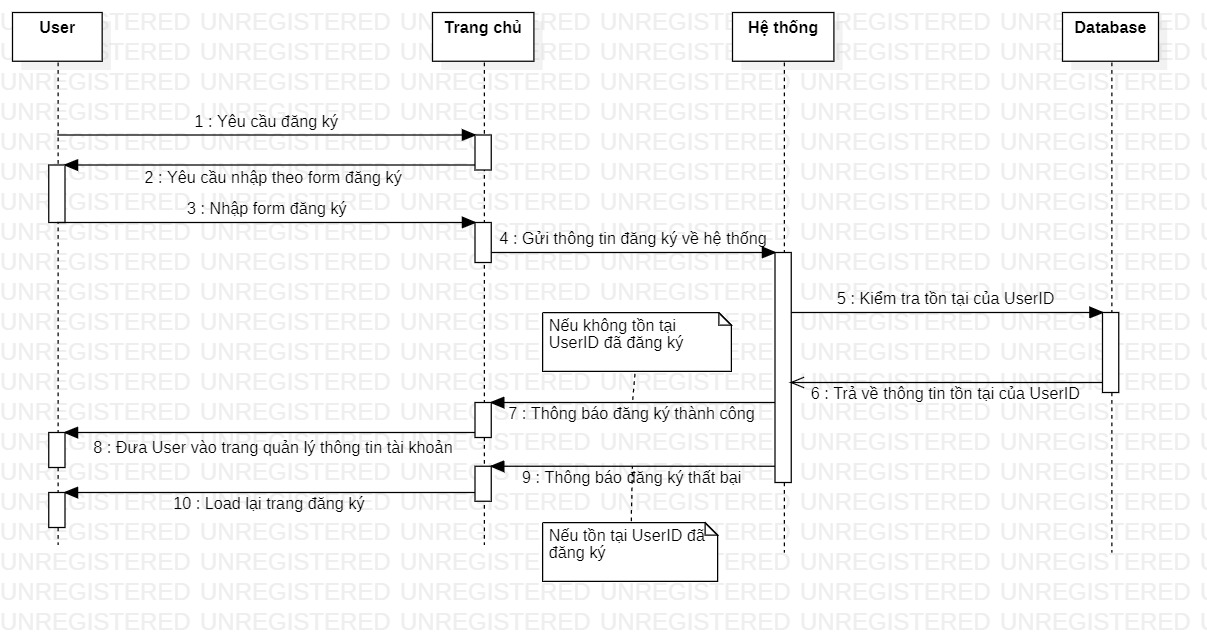
### 3.1.2 Sơ đồ Sequence

- Sequence đăng nhập:



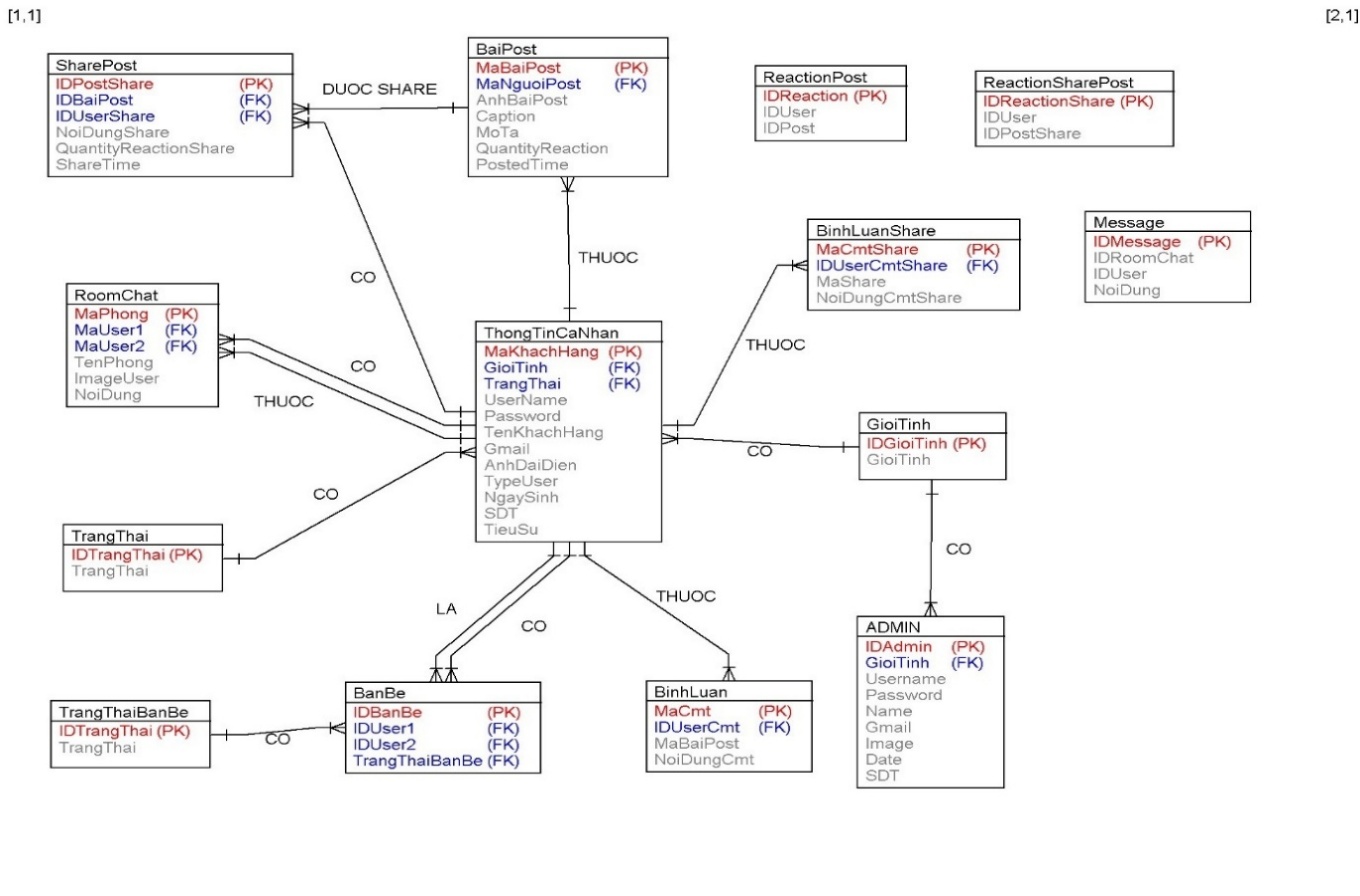
Hình 24. Sequence đăng nhập

- Sequence đăng ký:



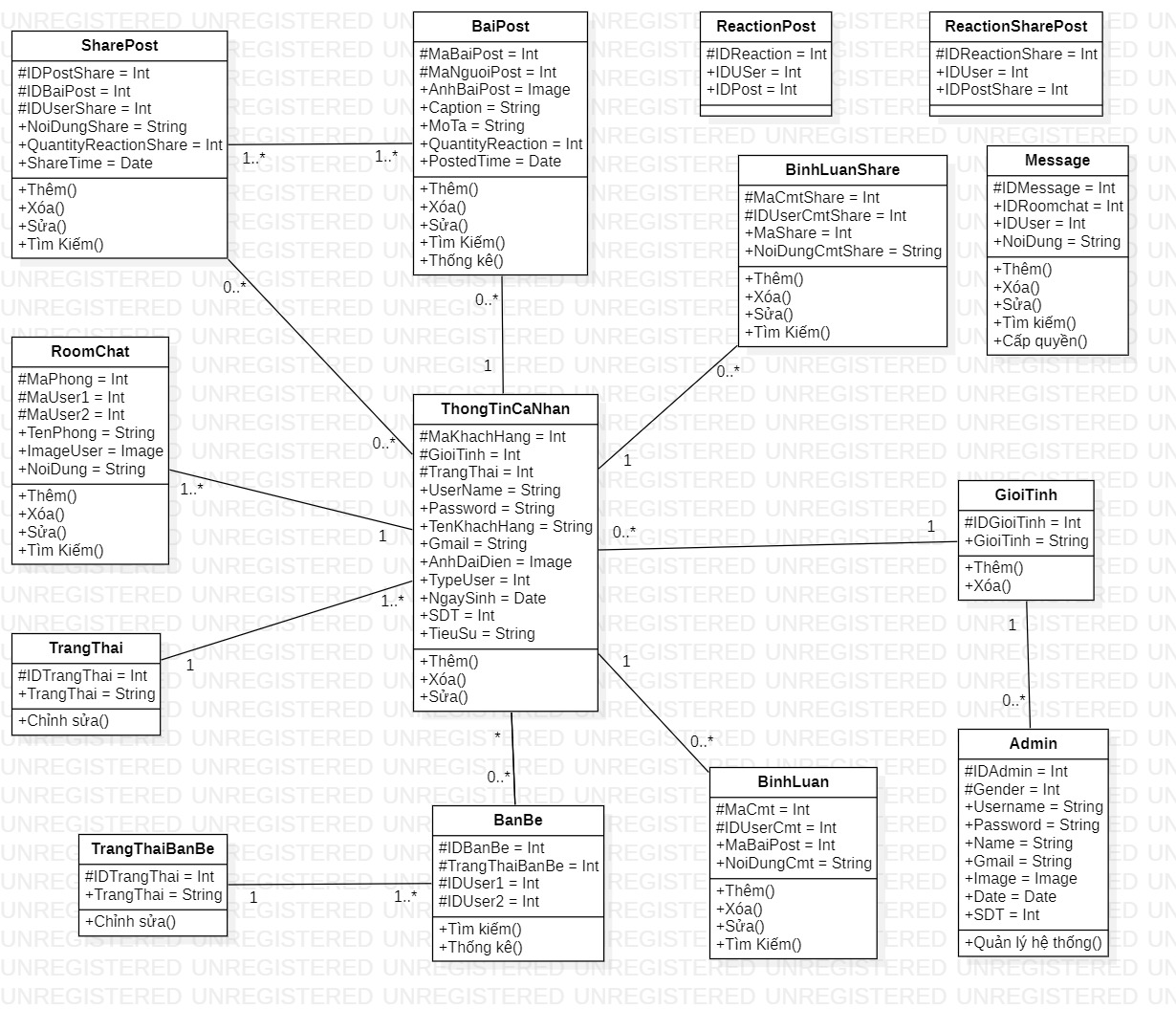
Hình 25. Sequence đăng ký

### 3.1.3 ERD



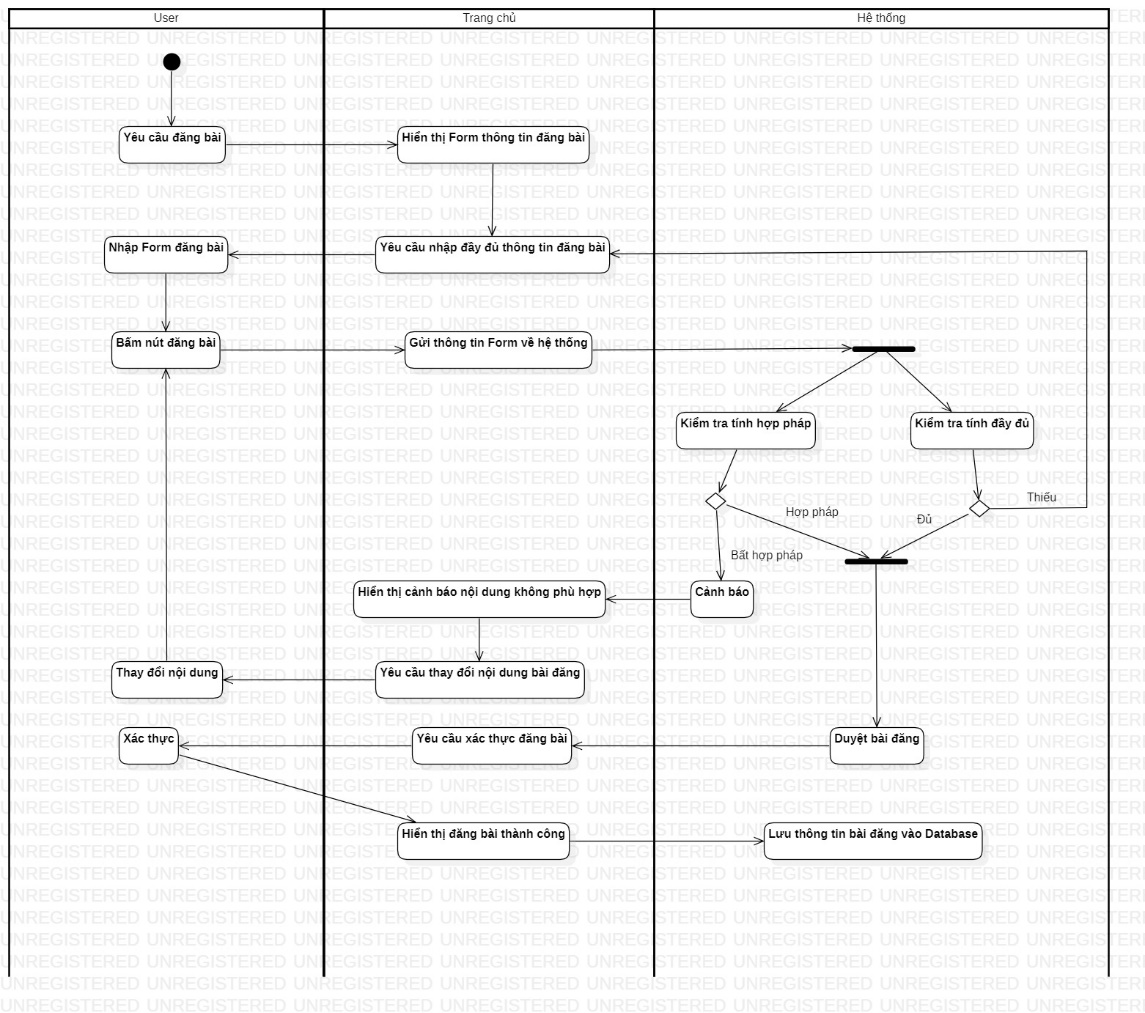
Hình 26. ERD

### 3.1.4 ClassDiagram



Hình 27. ClassDiagram

### 3.1.5 ActivityDiagram



Hình 28. ActivityDiagram

### 3.1.6 Các bảng của cơ sở dữ liệu

* user

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| user\_id | int |  | Khóa chính |
| username | varchar(75) |  |  |
| password | varchar(75) |  |  |
| full\_name | varchar(43) |  |  |
| email | varchar(40) |  |  |
| image | varchar(259) |  |  |
| background\_image | varchar(259) |  |  |
| phone\_number | varchar(10) | Allow |  |
| status | BIT |  |  |
| role\_id | int |  | Khóa ngoại |

* role

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| role\_id | int |  | Khóa chính |
| role\_name | varchar(20) |  |  |

* post

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| post\_id | int |  | Khóa chính |
| caption | varchar(40) | Allow |  |
| post\_image | varchar(259) | Allow |  |
| posted\_time | varchar(20) |  |  |
| reaction\_quantity | int |  |  |
| user\_up\_load | int |  | Khóa ngoại |

* users\_likes\_posts

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| post\_id | int |  | Khóa chính |
| user\_id | int |  | Khóa chính |

* comments

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| id | int |  | Khóa chính |
| content | varchar(300) |  |  |
| post\_id | int |  | Khóa ngoại |
| user\_id | int |  | Khóa ngoại |

* message

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| id | int |  | Khóa chính |
| user\_send\_message | int |  | Khóa ngoại |
| user\_receive\_message | int |  | Khóa ngoại |
| content | varchar(160) |  |  |

* friends

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| id | int |  | Khóa chính |
| user\_id\_send | int |  | Khóa ngoại |
| user\_id\_receive | int |  | Khóa ngoại |
| status\_relationship | int |  |  |

* notification

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| id | int |  | Khóa chính |
| notification\_category | int |  | Khóa ngoại |
| user\_id\_receive | int |  | Khóa ngoại |
| user\_id\_send | int |  | Khóa ngoại |

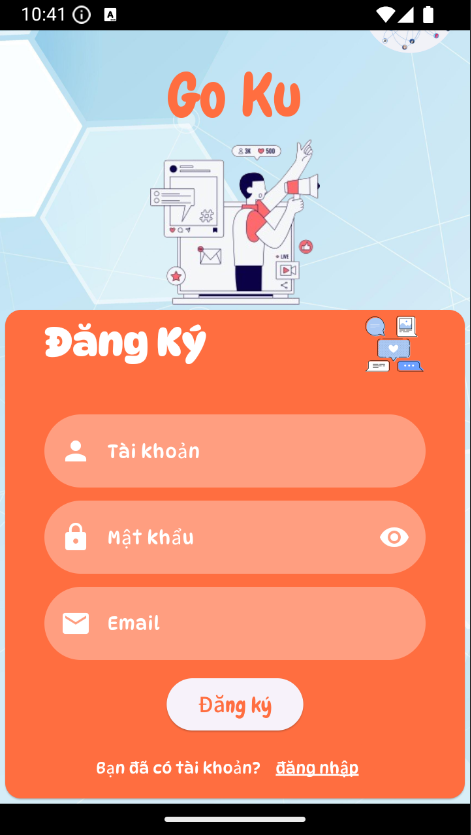
* notification\_category

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Allow Nulls |  |
| id | int |  | Khóa chính |
| name | varchar(20) |  |  |

## 3.2 Giao diện ứng dụng thực nghiệm

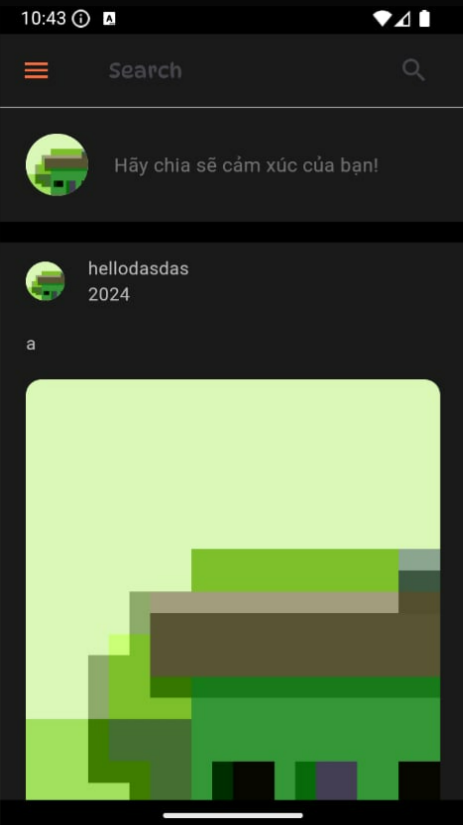
### 3.2.1 Trang đăng kí, đăng nhập





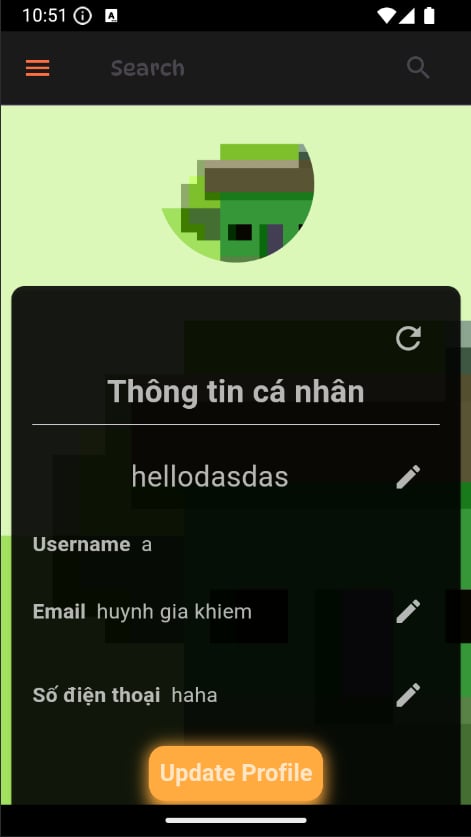
Hình 29. Trang đăng kí, đăng nhập

### 3.2.2 Trang chủ



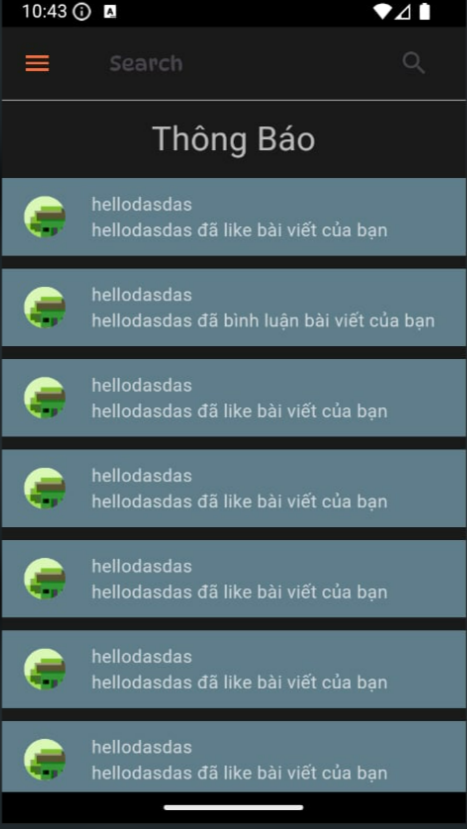
Hình 30. Trang chủ

### 3.2.3 Trang thông tin cá nhân



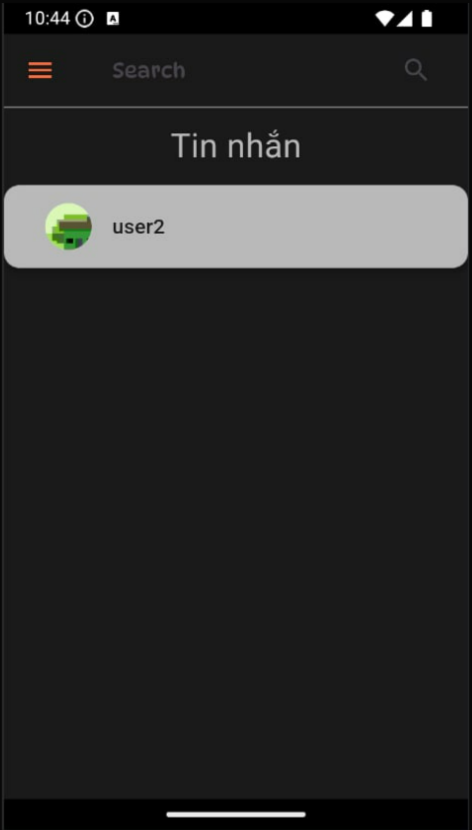
Hình 31. Trang thông tin cá nhân

### 3.2.4 Trang thông báo



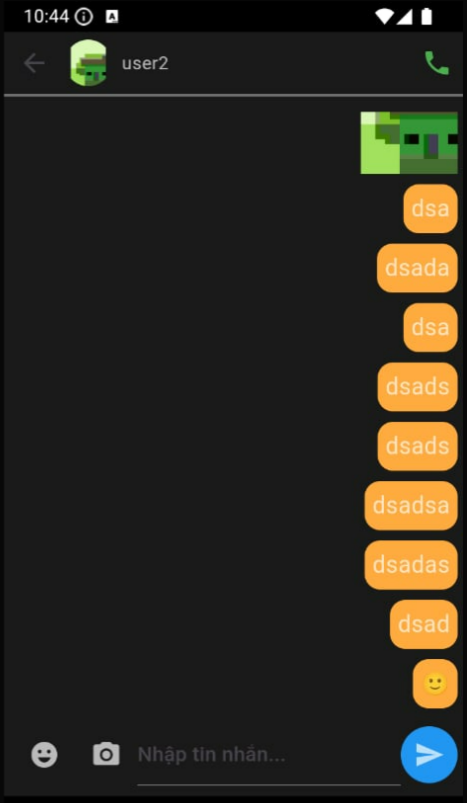
Hình 32. Trang thông báo

### 3.2.5 Trang danh sách tin nhắn



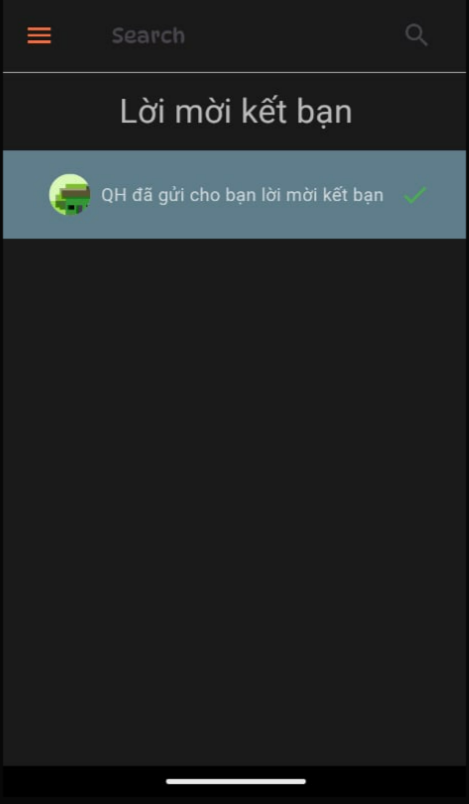
Hình 33. Trang danh sách tin nhắn

### 3.2.6 Chi tiết tin nhắn



Hình 34. Chi tiết tin nhắn

### 3.2.7 Trang thông báo lời mời kết bạn



Hình 35. Trang thông báo lời mời kết bạn

### 3.2.8 Trang cài đặt



Hình 36. Trang cài đặt

# Chương 4: KẾT LUẬN

## 4.1 Kết luận về Những Kết quả Đạt được:

Dự án đã đạt được nhiều kết quả quan trọng trong việc phát triển một nền tảng mạng xã hội đa chức năng và linh hoạt. Các chức năng cơ bản như đăng ký, đăng nhập, đăng xuất, quản lý bài đăng, bình luận, kết bạn và nhắn tin đã được triển khai thành công. Nền tảng cũng đã xây dựng được giao diện người dùng thân thiện và hiệu suất cao, đảm bảo trải nghiệm người dùng tốt nhất có thể.

## 4.2 Định Hướng Tương Lai của Sản Phẩm:

Để phát triển sản phẩm ngày càng tốt hơn và đáp ứng nhu cầu ngày càng cao của người dùng, chúng ta cần tập trung vào các mục tiêu sau:

* Mở rộng và Tối ưu hóa Chức năng Hiện có: Tiếp tục phát triển và tối ưu hóa các chức năng hiện có như tạo nhóm, quản lý sự kiện, cải thiện giao diện người dùng và tăng cường bảo mật.
* Phát triển tính năng Di động: Xây dựng ứng dụng di động cho cả iOS và Android để cung cấp trải nghiệm người dùng tốt nhất trên mọi thiết bị di động.
* Tăng cường Phân tích và Thống kê: Phát triển công cụ phân tích và thống kê mạnh mẽ để cung cấp thông tin chi tiết về hoạt động người dùng và hiệu suất của nền tảng.

## 4.3 Những Đóng Góp và Đề Xuất Mới:

Để nâng cao chất lượng và tính cạnh tranh của sản phẩm, dưới đây là một số đóng góp và đề xuất mới:

* Hoàn thành các chức năng chưa hoàn thiện.
* Tích hợp Thanh toán Trực tuyến: Hỗ trợ tích hợp các cổng thanh toán trực tuyến để người dùng có thể mua sản phẩm hoặc dịch vụ trên nền tảng mạng xã hội.
* Phát triển tính năng Livestream và Video: Cho phép người dùng xem livestream và video trực tiếp, cung cấp tính năng tương tác như bình luận và thả cảm xúc.
* Mở rộng tính năng Bảo mật và Quyền riêng tư: Hỗ trợ cài đặt quyền riêng tư chi tiết và các tùy chọn bảo mật cao hơn như xác minh hai yếu tố.

=> dự án đã đạt được những kết quả đáng kể và cung cấp cơ sở vững chắc cho việc phát triển sản phẩm trong tương lai. Bằng cách tập trung vào mở rộng tính năng, tối ưu hóa hiệu suất và cung cấp trải nghiệm người dùng tốt nhất, chúng ta có thể tiếp tục phát triển sản phẩm thành một nền tảng mạng xã hội tuyệt vời và phổ biến.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | C. L. V. Tiến, “vietnix,” 12 04 2024. [Trực tuyến]. Available: https://vietnix.vn/flutter-la-gi/. |
| [2] | S. Boot, “Spring Boot,” 2022. [Trực tuyến]. Available: https://docs.spring.io/spring-boot/index.html. |