# Chương 1: Giới thiệu về đề tài

## 1.1 Lý do chọn đề tài

Lý do chọn đề tài “Xây dựng website quản lý gia phả họ Ngô” xuất phát từ mong muốn bảo tồn và lưu trữ thông tin quan trọng về dòng họ một cách bền vững. Gia phả là tài sản quý giá của mỗi gia đình, ghi lại những câu chuyện và truyền thống qua nhiều thế hệ. Thay vì lưu trữ bằng các phương pháp truyền thống dễ bị hư hỏng theo thời gian, việc xây dựng website sẽ giúp thông tin được bảo quản lâu dài, giảm thiểu rủi ro mất mát, hư hỏng do yếu tố tự nhiên.

Một website gia phả giúp các thành viên trong gia đình, dù ở bất kỳ đâu, có thể truy cập và tìm hiểu về cội nguồn, ông bà, tổ tiên. Điều này không chỉ giúp kết nối các thế hệ mà còn tạo ra sự gắn kết sâu sắc giữa các thành viên trong dòng họ. Việc truy cập dễ dàng và chia sẻ thông tin trên nền tảng kỹ thuật số cũng khuyến khích mọi người duy trì sự liên lạc, đoàn kết hơn trong gia đình, đặc biệt trong bối cảnh cuộc sống ngày càng bận rộn.

Bên cạnh đó, website gia phả cho phép thông tin được quản lý và cập nhật một cách linh hoạt. Khi có sự thay đổi như thành viên mới, sửa đổi thông tin hay cập nhật các sự kiện, người quản lý có thể nhanh chóng điều chỉnh. Điều này không chỉ giúp thông tin luôn chính xác và kịp thời mà còn đảm bảo tính liên tục trong việc ghi chép dòng họ, đáp ứng nhu cầu theo dõi và bảo tồn truyền thống một cách toàn diện.

Việc xây dựng website quản lý gia phả cũng là một bước tiến trong ứng dụng công nghệ vào quản lý thông tin gia đình. Đây là cách làm hiện đại, sáng tạo, phù hợp với xu hướng số hóa hiện nay. Đặc biệt, thế hệ trẻ dễ dàng tiếp cận với công nghệ và sẽ có cảm giác gần gũi, hào hứng khi tìm hiểu gia phả qua nền tảng trực tuyến, thay vì qua các quyển gia phả truyền thống ít trực quan.

Dự án không chỉ là một bài tập ứng dụng kiến thức mà còn là một sản phẩm thực tiễn, có tiềm năng trong thị trường hiện nay. Với những lý do trên, tôi tin rằng việc xây dựng hệ thống website quản lý gia phả họ Ngô không chỉ đáp ứng nhu cầu cấp thiết của người trong dòng họ còn mang lại giá trị lớn cho bản thân trong quá trình học tập và phát triển kỹ năng công nghệ thông tin.

## 1.2 Mục tiêu đề tài

Mục tiêu chính của dự án là phát triển một hệ thống website gia phả cho gia đình họ Ngô, với khả năng hỗ trợ tổng quát thành viên trong gia phả.

* **Lưu trữ và bảo tồn thông tin gia phả**: Xây dựng một hệ thống trực tuyến giúp lưu trữ thông tin về dòng họ một cách có hệ thống và bền vững. Thông qua website, các thông tin về tổ tiên, các thế hệ, và các mối quan hệ trong gia đình sẽ được ghi chép và bảo tồn một cách an toàn, không lo hư hại như với các tài liệu giấy.
* **Tạo sự kết nối và chia sẻ trong dòng họ**: Website gia phả sẽ giúp các thành viên dễ dàng truy cập và chia sẻ thông tin, bất kể họ đang ở đâu. Qua đó, tạo ra một nền tảng kết nối, giúp tăng cường sự gắn kết và tình cảm gia đình, đặc biệt đối với những thành viên ở xa.
* **Đảm bảo khả năng quản lý và cập nhật thông tin linh hoạt**: Thiết lập hệ thống giúp người quản trị dễ dàng cập nhật thông tin khi có sự thay đổi như thành viên mới, sự kiện gia đình, hoặc các chi tiết khác. Mục tiêu này đảm bảo gia phả luôn chính xác và kịp thời, phù hợp với thực tế.
* **Ứng dụng công nghệ hiện đại vào việc quản lý gia phả**: Mục tiêu của đề tài là phát triển một hệ thống ứng dụng công nghệ, thân thiện với người dùng và đáp ứng nhu cầu tìm hiểu cội nguồn của các thế hệ trẻ. Qua đó, giúp thế hệ sau có thể dễ dàng tiếp cận và hiểu hơn về dòng họ của mình thông qua một giao diện trực tuyến hiện đại, trực quan.

Dự án còn giúp nâng cao kiến thức và kỹ năng lập trình, phát triển ứng dụng thực tế cho các thành viên trong nhóm, tạo tiền đề cho các dự án quy mô lớn hơn trong tương lai.

## 1.3 Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài xây dựng website quản lý gia phả được giới hạn ở các khía cạnh chính sau đây, nhằm đảm bảo tính khả thi và phù hợp với mục tiêu dự án:

* Phạm vi về chức năng: Nghiên cứu tập trung vào các chức năng cơ bản và quan trọng nhất của một hệ thống quản lý. Các chức năng dành cho người dùng bao gồm đăng nhập, tìm kiếm, thêm, sửa, xóa thành viên trong gia phả. Đối với, phạm vi nghiên cứu bao gồm các chức năng quản lý danh mục tài khoản.
* Phạm vi về người dùng: Đối tượng người dùng trong phạm vi nghiên cứu là những người trong dòng họ Ngô
* Phạm vi kỹ thuật: Về công nghệ, phạm vi nghiên cứu giới hạn trong các công cụ và ngôn ngữ lập trình như mongodb, C#(.NET) và JavaScript (Vuejs) để phát triển giao diện và chức năng của hệ thống. Hệ thống cũng sử dụng mongodb làm cơ sở dữ liệu chính để lưu trữ thông tin người dùng, và các dữ liệu liên quan.
* Phạm vi về triển khai và thử nghiệm: Dự án giới hạn trong việc xây dựng hệ thống trên môi trường máy chủ cục bộ và môi trường thử nghiệm, nhằm đảm bảo các tính năng hoạt động ổn định và đáp ứng nhu cầu cơ bản. Phạm vi thử nghiệm sẽ bao gồm các thao tác chính như đăng nhập, tìm kiếm, thêm, sửa, xóa thành viên và cập nhật dữ liệu. Hệ thống chưa bao gồm các tính năng nâng cao như tối ưu hóa cho các nền tảng di động.

## 1.4 Kết quả dự kiến

Website gia phả hoạt động ổn định và thân thiện với người dùng: Hoàn thành một website với giao diện trực quan, thân thiện, dễ dàng sử dụng cho mọi đối tượng, bao gồm cả những người không quen thuộc với công nghệ. Website có cấu trúc rõ ràng, giúp người dùng dễ dàng truy cập, tìm kiếm và thao tác với thông tin gia phả.

Hệ thống lưu trữ và quản lý thông tin gia phả đầy đủ và chi tiết: Đảm bảo website lưu trữ toàn bộ thông tin chi tiết về từng thành viên trong dòng họ như tên, năm sinh, mối quan hệ gia đình, hình ảnh và các sự kiện quan trọng. Thông tin được tổ chức theo cây phả hệ, hiển thị rõ ràng các mối quan hệ giữa các thế hệ.

Chức năng cập nhật và chỉnh sửa thông tin dễ dàng: Website cung cấp chức năng cho phép người quản trị dễ dàng cập nhật thông tin, thêm mới hoặc chỉnh sửa thông tin của từng thành viên. Điều này giúp đảm bảo gia phả luôn được cập nhật, phù hợp với những thay đổi trong dòng họ.

Hỗ trợ quản lý và thông báo sự kiện dòng họ: Website bao gồm tính năng thông báo và lên lịch cho các sự kiện chung của gia đình như họp mặt, giỗ tổ, lễ kỷ niệm. Chức năng này giúp các thành viên dễ dàng theo dõi và tham gia các sự kiện, từ đó tăng cường sự gắn kết trong gia đình.

Nâng cao hiểu biết và gắn kết trong dòng họ: Qua website, các thành viên có thể tìm hiểu sâu sắc hơn về cội nguồn và các giá trị văn hóa, lịch sử của dòng họ. Đây là một kết quả quan trọng, góp phần xây dựng niềm tự hào và tăng cường sự đoàn kết giữa các thế hệ.

## 1.5 Kết luận

Trong chương I, dự án đã trình bày tổng quan về ý tưởng xây dựng website quản lý gia phả họ Ngô, với mục tiêu hỗ trợ người dùng nắm được thông tin dòng họ một cách trực quan. Lý do lựa chọn đề tài xuất phát từ nhu cầu muốn tìm hiểu nguồn gốc của gia đình. Dự án cũng đã nêu rõ các mục tiêu nghiên cứu, nhằm đáp ứng yêu cầu của cả người dùng và quản trị viên, đồng thời giới hạn phạm vi trong các tính năng và kỹ thuật cốt lõi để đảm bảo tính khả thi.

Với những kết quả dự kiến đạt được, dự án hướng tới việc cung cấp một hệ thống hoàn thiện, hiện đại và dễ sử dụng. Chương I đã đặt nền tảng cơ bản, làm tiền đề cho các nội dung chi tiết trong các chương tiếp theo.

# Chương 1: Cơ sở lý thuyết

## 1.1 Cơ sở lý thuyết

Gia phả là hệ thống ghi chép về các thành viên trong dòng họ, thường lưu giữ thông tin về mối quan hệ huyết thống, các sự kiện gia đình, và lịch sử qua nhiều thế hệ. Gia phả không chỉ giúp các thành viên trong dòng họ hiểu rõ về cội nguồn và lịch sử mà còn giúp bảo tồn các giá trị văn hóa và truyền thống gia đình. Hiểu rõ khái niệm này là tiền đề để xây dựng một hệ thống quản lý thông tin gia phả hiệu quả và có ý nghĩa.

Cơ sở dữ liệu trong hệ thống gia phả: Cơ sở dữ liệu (CSDL) là phần quan trọng trong việc xây dựng hệ thống gia phả, giúp lưu trữ thông tin về người dùng. CSDL thường được thiết kế dưới dạng các bảng liên kết với nhau thông qua khóa chính và khóa ngoại.

Gia phả chứa nhiều thông tin cá nhân nhạy cảm, vì vậy lý thuyết về bảo mật dữ liệu đóng vai trò quan trọng trong việc xây dựng hệ thống. Các khái niệm về bảo mật như mã hóa dữ liệu, phân quyền truy cập, và xác thực người dùng giúp bảo vệ thông tin gia đình, đảm bảo chỉ những thành viên được phép mới có thể truy cập và chỉnh sửa thông tin gia phả.

Lý thuyết về quản lý sự kiện và thông báo giúp xây dựng các tính năng cho phép dòng họ dễ dàng lên lịch các hoạt động gia đình, như họp mặt và lễ giỗ, và gửi thông báo đến các thành viên. Điều này không chỉ đảm bảo tính tổ chức cho các sự kiện quan trọng mà còn thúc đẩy sự gắn kết và kết nối trong gia đình.

## 1.2 Công nghệ sử dụng

### 1.2.1 VUEJS

**Vue.js** là một framework JavaScript mã nguồn mở, được sử dụng để xây dựng giao diện người dùng và các ứng dụng web một cách nhanh chóng và hiệu quả. Vue.js nổi bật nhờ tính linh hoạt và khả năng tùy biến cao, giúp các nhà phát triển dễ dàng tạo ra các ứng dụng web có tính tương tác cao.

**Ưu điểm của Vue.js**

* Giao diện tương tác mượt mà: Vue.js hỗ trợ reactivity (tính phản ứng) giúp giao diện cập nhật tự động khi dữ liệu thay đổi, phù hợp cho các ứng dụng cần tính tương tác cao như quản lý gia phả, bảng dữ liệu động, hoặc hiển thị thông tin chi tiết của từng thành viên trong hệ thống.
* Cấu trúc dự án rõ ràng và dễ bảo trì: Vue.js sử dụng các component (thành phần) độc lập, giúp tổ chức code rõ ràng, dễ bảo trì và tái sử dụng.
* Hệ sinh thái phong phú: Vue.js có hệ sinh thái phong phú bao gồm Vuex để quản lý trạng thái, Vue Router để điều hướng, và các thư viện khác hỗ trợ phát triển ứng dụng. Điều này giúp tích hợp các tính năng phức tạp như hệ thống đăng nhập, phân quyền, và hiển thị thông tin gia phả.
* Tài liệu và cộng đồng lớn: Vue.js có tài liệu chi tiết và cộng đồng mạnh mẽ, giúp người phát triển dễ dàng tìm kiếm và áp dụng các giải pháp cho các vấn đề gặp phải.

Nhược điểm của Vue.js

* Không phổ biến bằng một số framework khác: Vue.js có thể ít phổ biến hơn so với các framework như React hoặc Angular, điều này có thể hạn chế khả năng tìm kiếm nhà phát triển có kinh nghiệm trong Vue.js ở một số thị trường.
* Hạn chế khi phát triển ứng dụng lớn: Mặc dù Vue.js phù hợp cho các ứng dụng vừa và nhỏ, nhưng khi xây dựng các ứng dụng lớn và phức tạp, có thể cần tích hợp thêm các công cụ khác để tối ưu hóa hiệu năng và quản lý dự án, đòi hỏi sự hiểu biết sâu hơn về hệ sinh thái Vue.js.
* Quản lý trạng thái phức tạp: Khi ứng dụng có nhiều trạng thái phức tạp, Vuex có thể yêu cầu thiết lập nhiều cấu trúc phức tạp, đòi hỏi thêm thời gian và công sức để tổ chức và quản lý hiệu quả.
* Khả năng bảo mật hạn chế: HTML không hỗ trợ các tính năng bảo mật như mã hóa dữ liệu, bảo vệ tài khoản người dùng, hoặc ngăn chặn các cuộc tấn

### 1.2.2 CSS - Bootstrap

**CSS** (Cascading Style Sheets): CSS là ngôn ngữ dùng để định dạng giao diện và kiểu dáng của trang web. CSS cho phép nhà phát triển web kiểm soát màu sắc, phông chữ, bố cục, khoảng cách, và nhiều yếu tố khác của các phần tử HTML trong trang web. CSS giúp tạo ra những giao diện đẹp mắt và dễ dàng thay đổi phong cách của trang web mà không ảnh hưởng đến cấu trúc HTML. Với CSS, các nhà phát triển có thể dễ dàng tối ưu giao diện cho các thiết bị di động và máy tính để bàn.

**Bootstrap**: Bootstrap là một framework CSS mã nguồn mở được phát triển bởi Twitter, giúp xây dựng các giao diện web nhanh chóng và dễ dàng hơn. Bootstrap cung cấp một bộ sưu tập các thành phần giao diện sẵn có như nút bấm, bảng, form, thanh điều hướng (navbar), modal, v.v. Nó sử dụng các lớp CSS đã được thiết lập sẵn, cho phép người dùng tạo các giao diện responsive (tương thích với nhiều thiết bị) mà không cần phải viết mã CSS từ đầu.

**Ưu điểm của CSS**

* Kiểm soát chi tiết giao diện: CSS cho phép nhà phát triển điều chỉnh chính xác từng yếu tố trên trang web, từ phông chữ, màu sắc, đến kích thước các phần tử, tạo ra giao diện chuyên nghiệp và dễ dàng thay đổi.
* Dễ dàng tái sử dụng: CSS có thể áp dụng cho nhiều trang và phần tử trong cùng một trang web, giúp giảm thiểu việc lặp lại mã, từ đó làm cho mã nguồn gọn gàng và dễ bảo trì.
* Thiết kế giao diện responsive: CSS cho phép thiết kế các giao diện tương thích với nhiều kích thước màn hình, từ desktop đến điện thoại di động, nhờ vào các kỹ thuật như media queries.
* Tiết kiệm thời gian và công sức: Việc tách biệt giao diện với nội dung (HTML) giúp CSS dễ dàng duy trì và cập nhật. Các thay đổi về giao diện có thể thực hiện mà không cần phải sửa đổi mã HTML.
* Dễ dàng phối hợp với các công nghệ khác: CSS có thể kết hợp với các công nghệ khác như JavaScript, HTML, và các framework (ví dụ: Bootstrap, ElementIO) để tạo ra giao diện động và phức tạp.

**Nhược điểm của CSS**

* Khó khăn trong việc quản lý các trang web lớn: Khi dự án web trở nên phức tạp, việc duy trì và cập nhật CSS có thể trở nên khó khăn, đặc biệt khi có nhiều stylesheet và các quy tắc CSS khác nhau. Điều này có thể dẫn đến sự khó hiểu và lộn xộn trong mã nguồn.
* Không hỗ trợ các tính năng động: CSS chỉ định dạng giao diện nhưng không thể xử lý các tính năng động như kiểm tra tính hợp lệ của biểu mẫu, quản lý dữ liệu người dùng, hoặc thực hiện các thao tác động trong trang web. Để thực hiện những chức năng này, cần kết hợp với JavaScript.
* Khó khăn trong việc hỗ trợ trình duyệt cũ: Mặc dù các trình duyệt hiện đại đều hỗ trợ CSS rất tốt, nhưng vẫn có những vấn đề tương thích với các trình duyệt cũ, đặc biệt là trong việc áp dụng các tính năng mới như Flexbox hay Grid.

### 1.2.4 C#

**C#** là một ngôn ngữ lập trình hiện đại, hướng đối tượng, được phát triển bởi Microsoft, thường sử dụng cùng với .NET Framework hoặc .NET Core (phiên bản mã nguồn mở). **.NET** là nền tảng phát triển đa năng của Microsoft, hỗ trợ nhiều ngôn ngữ lập trình và cung cấp một môi trường nhất quán để xây dựng, triển khai và quản lý các ứng dụng trên cả máy chủ và máy khách. C# và .NET thường được sử dụng để phát triển các ứng dụng web, ứng dụng di động, và phần mềm doanh nghiệp với hiệu năng cao và khả năng mở rộng.

**Ưu điểm của C# và .NET**

* Tính đa nền tảng: Với sự ra đời của .NET Core và hiện nay là .NET 5/6/7, các ứng dụng C# có thể chạy trên nhiều nền tảng như Windows, Linux và macOS, giúp giảm chi phí và mở rộng phạm vi sử dụng.
* Hiệu năng cao: .NET được tối ưu hóa để cung cấp hiệu năng cao, đặc biệt trong các ứng dụng web và dịch vụ API. .NET sử dụng công nghệ Just-In-Time (JIT) compilation và hỗ trợ tối ưu hóa bộ nhớ, giúp xử lý tốt các ứng dụng lớn và phức tạp.
* Hỗ trợ mạnh mẽ cho ứng dụng web: Với .NET, các công cụ như ASP.NET Core, MVC, và Web API giúp xây dựng các ứng dụng web hiện đại, nhanh chóng và bảo mật. .NET cũng tích hợp tốt với các công nghệ JavaScript như React và Angular để phát triển các ứng dụng giao diện người dùng phong phú.
* Thư viện phong phú và cộng đồng mạnh mẽ: .NET đi kèm với một thư viện phong phú, cung cấp nhiều công cụ và tiện ích giúp giảm thiểu thời gian phát triển. Ngoài ra, cộng đồng lập trình viên sử dụng .NET rất lớn và sôi động, hỗ trợ giải pháp cho các vấn đề phát sinh.
* Hỗ trợ phát triển ứng dụng doanh nghiệp: C# và .NET rất phổ biến trong các tổ chức lớn nhờ tính bảo mật và tính ổn định cao, cùng với sự hỗ trợ lâu dài từ Microsoft. Các ứng dụng như quản lý quan hệ khách hàng (CRM), quản lý nguồn nhân lực (HRM), và hệ thống tài chính thường được phát triển bằng C# và .NET.
* Khả năng mở rộng và bảo trì tốt: C# là ngôn ngữ hướng đối tượng, dễ hiểu, dễ bảo trì và mở rộng. Điều này rất quan trọng cho các dự án doanh nghiệp dài hạn, khi yêu cầu bảo trì và nâng cấp thường xuyên..

**Nhược điểm của C# và .NET**

* Phụ thuộc vào Microsoft: Dù .NET Core đã trở thành mã nguồn mở, nhưng hệ sinh thái .NET vẫn phụ thuộc khá lớn vào Microsoft, đặc biệt khi có các thay đổi về chiến lược và công nghệ mới.
* Kích thước runtime lớn: Đối với các ứng dụng nhỏ, runtime của .NET có thể tạo ra các ứng dụng có kích thước lớn hơn so với các ngôn ngữ khác, làm cho việc triển khai có thể trở nên phức tạp và tốn kém tài nguyên.
* Độ phức tạp cao đối với người mới: .NET có rất nhiều công nghệ và thư viện đi kèm, như ASP.NET, WPF, WinForms, và các công cụ quản lý như Entity Framework. Đối với người mới, việc làm quen với hệ sinh thái lớn và phức tạp này có thể mất nhiều thời gian.
* Chi phí cao cho các công cụ phát triển: Dù .NET Core và các công cụ phát triển liên quan đã được cung cấp miễn phí, nhưng việc sử dụng Visual Studio (phiên bản đầy đủ) thường yêu cầu chi phí giấy phép khá cao trong môi trường doanh nghiệp..

### 1.2.5 MongoDB

**MongoDB** là một hệ quản trị cơ sở dữ liệu NoSQL, mã nguồn mở, được thiết kế để lưu trữ dữ liệu theo mô hình tài liệu (document-oriented) và tối ưu cho các ứng dụng đòi hỏi khả năng mở rộng cao, dữ liệu phong phú và đa dạng. MongoDB lưu trữ dữ liệu dưới dạng các tài liệu JSON có cấu trúc linh hoạt, điều này giúp việc thao tác và xử lý dữ liệu linh hoạt hơn so với các cơ sở dữ liệu quan hệ truyền thống.

**Ưu điểm của MongoDB**

* Linh hoạt về cấu trúc dữ liệu: MongoDB không đòi hỏi một schema cố định như các cơ sở dữ liệu SQL. Điều này giúp dễ dàng thay đổi cấu trúc của tài liệu khi cần, và phù hợp với dữ liệu không có cấu trúc hoặc có cấu trúc phức tạp, cho phép lưu trữ các đối tượng và mảng lồng nhau.
* Hiệu năng cao và mở rộng theo chiều ngang (horizontal scaling): MongoDB hỗ trợ sharding – kỹ thuật chia nhỏ dữ liệu ra nhiều máy chủ để xử lý song song, giúp tăng khả năng xử lý và lưu trữ dữ liệu lớn mà không ảnh hưởng đến hiệu năng.
* Truy vấn và thao tác nhanh: MongoDB cung cấp các phương thức truy vấn đa dạng và hỗ trợ các chỉ mục (indexing) mạnh mẽ, giúp thực hiện các thao tác đọc/ghi nhanh chóng, ngay cả đối với lượng dữ liệu lớn.
* Phù hợp với dữ liệu lớn và phân tán: MongoDB rất thích hợp cho các hệ thống có dữ liệu lớn và yêu cầu khả năng phân phối dữ liệu trên nhiều máy chủ, như các ứng dụng phân tích dữ liệu, ứng dụng thời gian thực hoặc các ứng dụng phân tán đa vùng.
* Khả năng tích hợp linh hoạt: MongoDB có thể tích hợp với nhiều ngôn ngữ lập trình như Node.js, Python, Java, C#... nhờ có các thư viện và API hỗ trợ đa dạng. Điều này làm cho MongoDB trở thành lựa chọn phổ biến cho các dự án với yêu cầu kết hợp nhiều công nghệ.
* Hỗ trợ tốt cho các loại dữ liệu phi cấu trúc: Do MongoDB lưu trữ dữ liệu dưới dạng JSON hoặc BSON, nó rất phù hợp để lưu các loại dữ liệu phi cấu trúc hoặc bán cấu trúc như thông tin người dùng, nội dung tin nhắn, hoặc dữ liệu từ các thiết bị IoT.

**Nhược điểm của MongoDB**

* Thiếu các tính năng ACID cho giao dịch phức tạp: Dù đã hỗ trợ giao dịch đa tài liệu từ MongoDB 4.0 trở đi, MongoDB vẫn chưa cung cấp khả năng đảm bảo các thuộc tính ACID một cách mạnh mẽ như các cơ sở dữ liệu quan hệ. Điều này có thể gây khó khăn cho các ứng dụng cần xử lý các giao dịch tài chính hoặc dữ liệu nghiêm ngặt về tính nhất quán.
* Không phù hợp cho các ứng dụng có yêu cầu phức tạp về quan hệ: MongoDB lưu trữ dữ liệu dưới dạng tài liệu độc lập, vì vậy các mối quan hệ phức tạp giữa các bảng như trong SQL có thể khó quản lý và tối ưu trong MongoDB. Thông thường, các ứng dụng quan hệ phức tạp sẽ yêu cầu cơ sở dữ liệu quan hệ truyền thống hơn.
* Dung lượng bộ nhớ cao hơn: MongoDB lưu trữ dữ liệu dưới dạng BSON (Binary JSON) và lưu thêm các metadata cho mỗi tài liệu, nên sẽ chiếm dung lượng lưu trữ lớn hơn so với các cơ sở dữ liệu quan hệ truyền thống.
* Đòi hỏi quản lý sharding phức tạp: Đối với các hệ thống lớn, khi sử dụng sharding, việc cấu hình và bảo trì hệ thống có thể trở nên phức tạp, đòi hỏi sự quản lý chi tiết và kinh nghiệm từ người vận hành..

## 1.3 Kết luận

Chương 2 đã giới thiệu các công nghệ và công cụ cơ bản được sử dụng trong quá trình phát triển hệ thống gia phả, bao gồm Vuejs, CSS, C# và MongoDB. Mỗi công nghệ này đóng vai trò quan trọng trong việc xây dựng giao diện người dùng, xử lý các thao tác động, quản lý dữ liệu và tối ưu hóa hiệu suất của hệ thống. Việc áp dụng kết hợp những công nghệ này mang lại một nền tảng vững chắc, giúp website hoạt động hiệu quả, dễ sử dụng và tối ưu hóa cho người dùng cuối.

# Chương 2: Phân tích, đặc tả yêu cầu

## 2.1 Biểu đồ use case

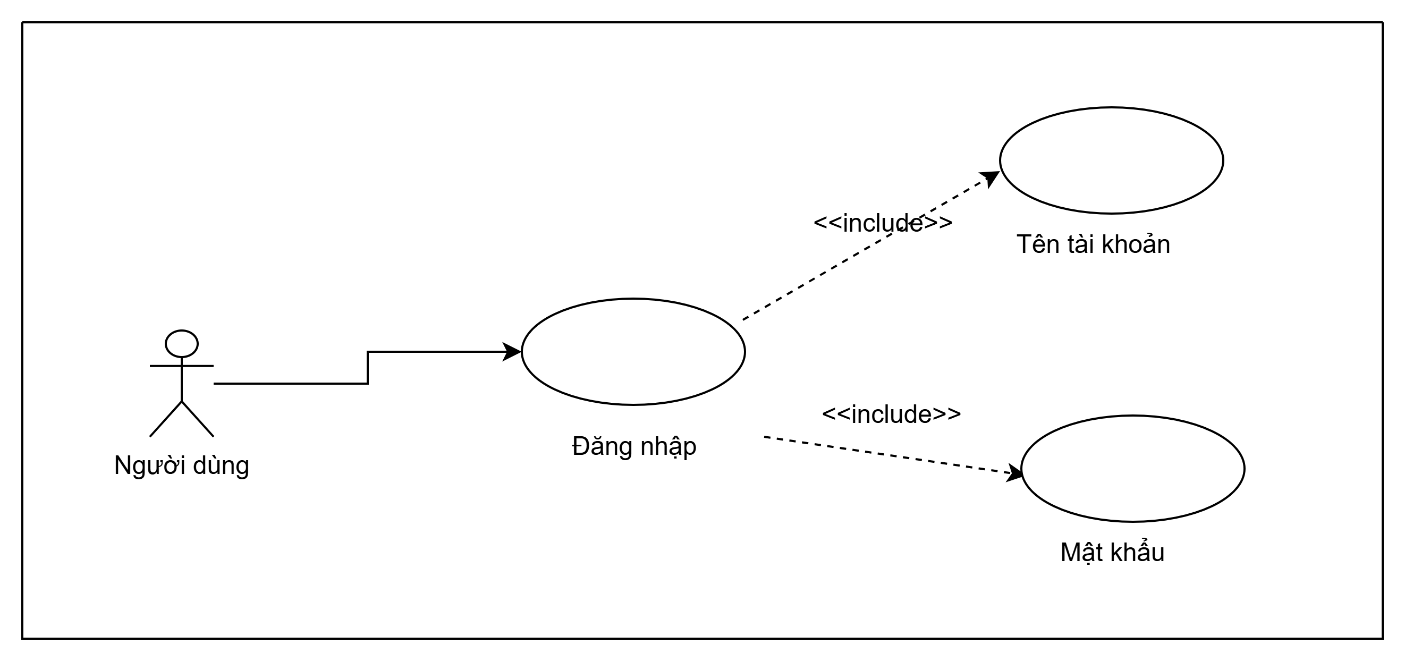


Hình 3.1 Biểu đồ use case tổng quát

## 2.2 Đặc tả use case

### 2.2.1 Đặc tả use case đăng nhập

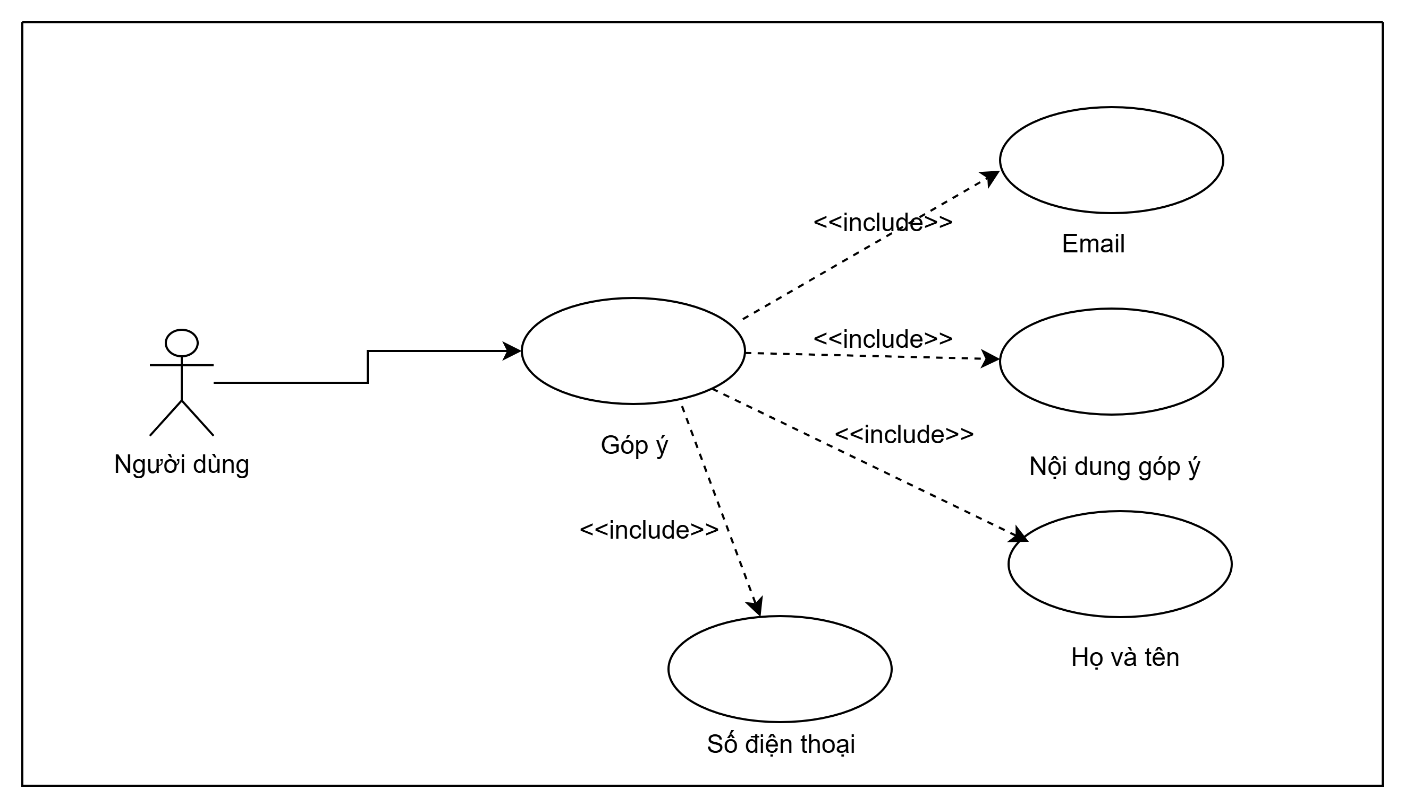
|  |  |
| --- | --- |
| Tên use case | Đăng nhập |
| Mô tả | Use case cho phép người dùng đăng nhập hệ thống |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng truy cập vào trang chủ và di chuyển đến vùng có form đăng nhập hoặc bấm vào nút đăng nhập trên thanh menu 2. Người dùng điền đầy đủ thông tin đăng nhập: tài khoản và mật khẩu và chọn nút “đăng nhập” 3. Hệ thống kiểm tra dữ liệu nhập vào 4. Nếu thông tin không chính xác hoặc thiếu thông tin cần thiết, hệ thống hiển thị thông báo lỗi. 5. Nếu thông tin hợp lệ, hệ thống chuyển hướng đến trang gia phả   Luồng rẽ nhánh:  Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. |
| Tiền điều kiện | * Người dùng truy cập vào trang chủ. |
| Hậu điều kiện |  |



Hình 2.2.0.2 Biểu đồ phân rã usecase đăng nhập

### 2.2.2 Đặc tả use case góp ý

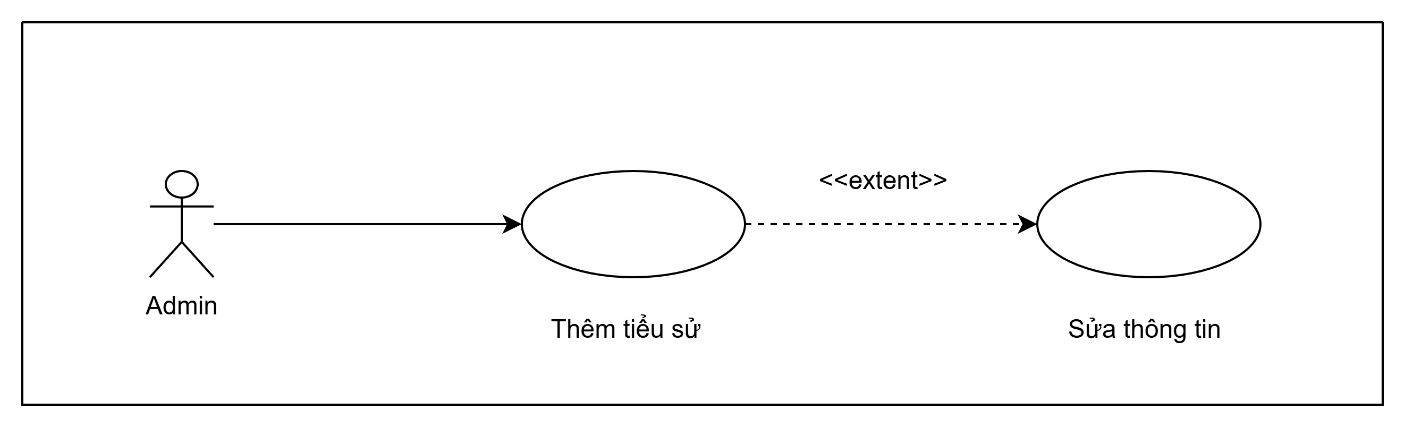
|  |  |
| --- | --- |
| Tên use case | Góp ý |
| Mô tả | Use case cho phép người dùng có thể đóng góp ý kiến để trang web cải tiến hơn |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng truy cập vào trang chủ và kích chuột vào menu “Góp ý” hoặc di chuyển xuống vùng góp ý 2. Người dùng điền đầy đủ thông tin góp ý: email, số điện thoại, họ và tên, nội dung góp ý và chọn nút “Góp ý” 3. Hệ thống kiểm tra dữ liệu nhập vào 4. Nếu thông tin hợp lệ, hệ thống tiến hành lưu dữ liệu và thông báo cho người dùng   Luồng rẽ nhánh:  Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. |
| Tiền điều kiện | * Người dùng đã truy cập vào trang chủ. |
| Hậu điều kiện |  |



Hình 2.2.2 Biểu đồ phân ra usecase góp ý

### 2.2.3 Đặc tả use case thêm tiểu sử

|  |  |
| --- | --- |
| Tên use case | Thêm tiểu sử |
| Mô tả | Use case cho phép người dùng bổ sung hoặc cập nhật tiểu sử của thành viên trong gia phả |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng kích vào biểu tượng thêm tiểu sử 2. Người dùng có thể thêm (nếu chưa có thông tin), sửa tiểu sử của 1 thành viên trong gia phả 3. Người dùng có thể sửa thông tin cá nhân của đối tượng đang thêm tiểu sử. 4. Hệ thống kiểm tra thông tin người dùng nhập và xử lý. 5. Hệ thống hiển thị thông báo thay đổi thành công.   Luồng rẽ nhánh:  Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. |
| Tiền điều kiện | * Người dùng đã đăng nhập thành công. * Hệ thống có cơ sở dữ liệu chứa thông tin cá nhân của người dùng. |
| Hậu điều kiện |  |



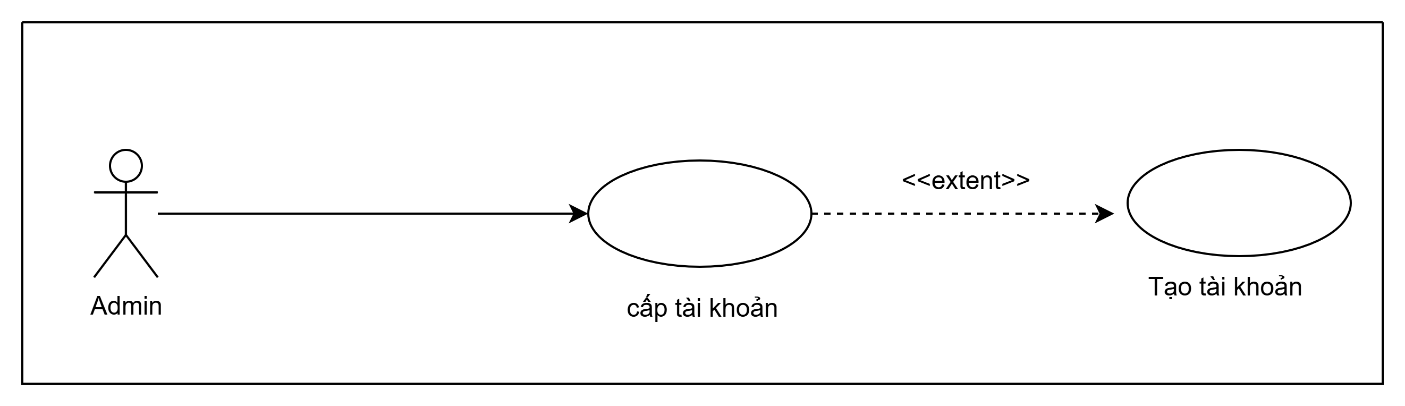
Hình 2.2.3 Biểu đồ phân rã usecase thêm tiểu sử

### 2.2.4 Đặc tả use case xem chi tiết mối quan hệ

|  |  |
| --- | --- |
| Tên use case | Xem chi tiết mối quan hệ |
| Mô tả | Use case cho phép người dùng xem các mối quan hệ xung quanh thành viên đã chọn |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng truy cập vào trang gia phả 2. Người dùng kích chuột vào tên của người bất kỳ có trong cây gia phả 3. Hệ thống kiểm tra thông tin mối quan hệ xung quanh người đó như bố mẹ, anh chị em, vợ con và hiển thị lên màn hình   Luồng rẽ nhánh:  Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. |
| Tiền điều kiện | * Người dùng đã đăng nhập thành công. * Hệ thống có cơ sở dữ liệu chứa thông tin cá nhân của người dùng. |
| Hậu điều kiện |  |

### 2.2.5 Đặc tả use case cấp tài khoản

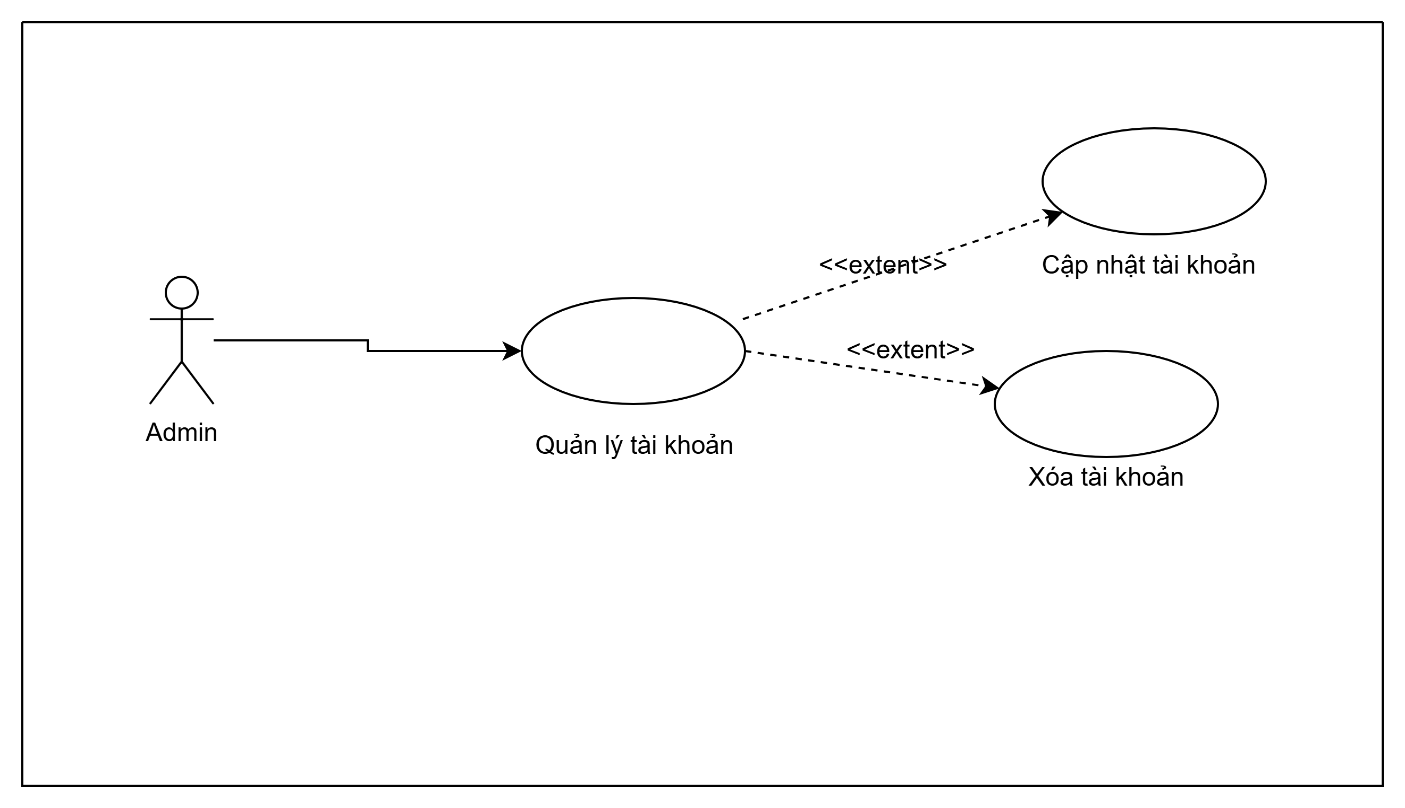
|  |  |
| --- | --- |
| Tên use case | Cấp tài khoản |
| Mô tả | Use case cho phép cấp tài khoản cho một thành viên trong gia phả |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng kích vào biểu tượng cấp tài khoản 2. Hệ thống tự động sinh ra “Tên đăng nhập” và “mật khẩu” cho đối tượng cấp (là tên kết hợp với ngày sinh) và hiển thị form gồm 2 trường đã lêu 3. Người dùng có thể sửa tên đăng nhập hoặc mật khẩu nếu cần thiết 4. Nhấn “Lưu” để cấp tài khoản   Luồng rẽ nhánh:   * Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. * Tại bước 4, nếu người dùng bấm nút “Thoát” thì use case sẽ kết thúc |
| Tiền điều kiện | * Người dùng đã thực hiện đăng nhập. * Tài khoản để cấp phải là tài khoản admin |
| Hậu điều kiện |  |



Hình 2.2.5 Biểu đồ phân rã usecase cấp tài khoản

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

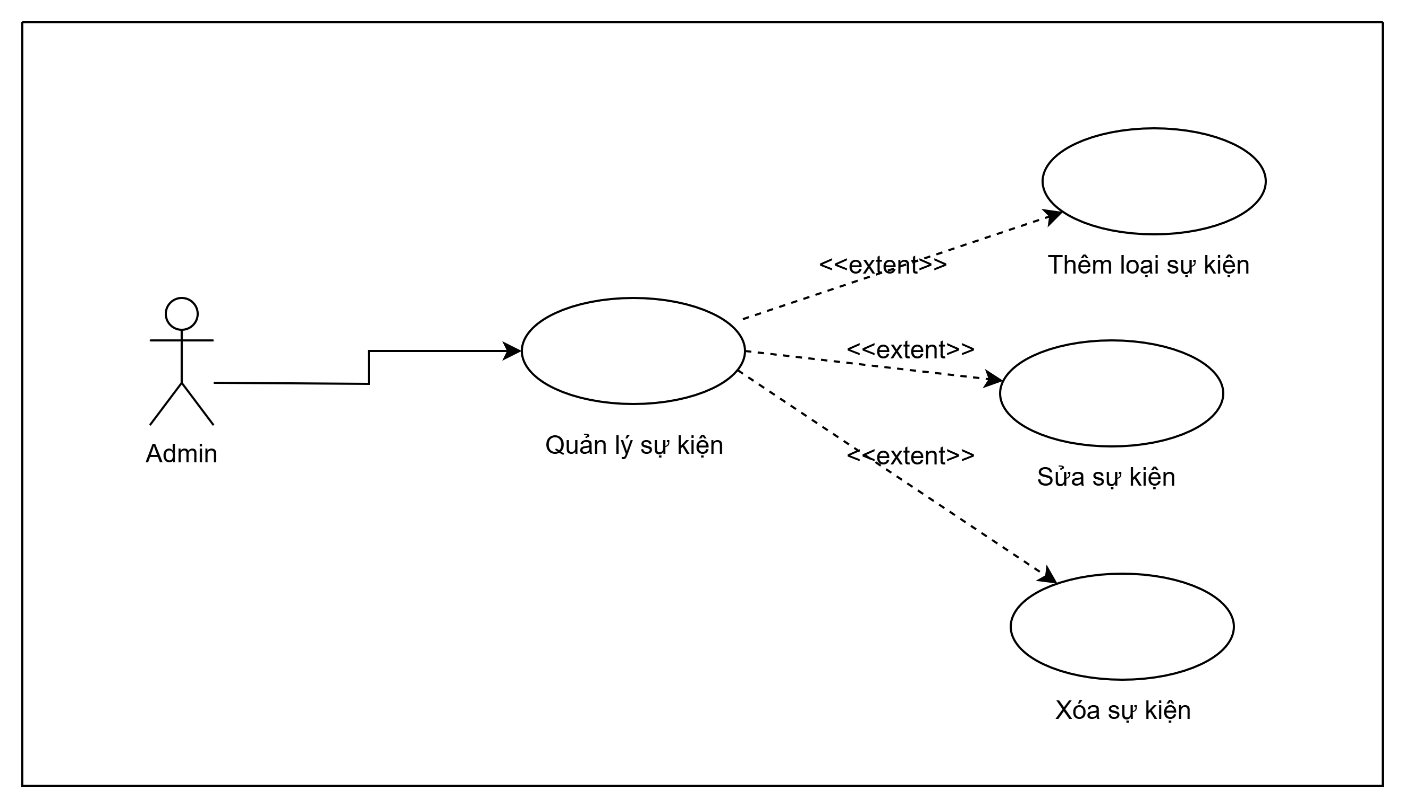
|  |  |
| --- | --- |
| Tên use case | Quản lý tài khoản |
| Mô tả | Use case cho phép người dùng sửa, xóa tài khoản trong hệ thống |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng kích chọn vào biểu tượng sửa trong danh sách 2. Hệ thống kiểm tra thông tin và điền thông tin sẵn có và hiển thị form sửa. 3. Người dùng nhập thông tin muốn sửa và kích nút “Lưu” 4. Hệ thống kiểm tra thông tin và cập nhật lại thông tin vào cơ sở dữ liệu 5. Người dùng kích chọn vào biểu tượng xóa trong danh sách 6. Hệ thống hiển thị thông báo xác nhận xóa. 7. Nếu người dùng chọn “OK” thì hệ thống sẽ xóa tài khoản và thông báo xóa thành công   Luồng rẽ nhánh:   * Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. * Tại bước thứ 7 trong luồng cơ bản, nếu người dùng bấm nút “Cancel” thì thông tin sẽ không được xóa và use case kết thúc |
| Tiền điều kiện | * Người dùng đã thực hiện đăng nhập. * Hệ thống có cơ sở dữ liệu chứa thông tin cá nhân của người dùng. |
| Hậu điều kiện |  |



Hình 2.2.6 Biểu đồ phân rã use case quản lý tài khoản

### 2.2.7 Đặc tả use case quản lý sự kiện

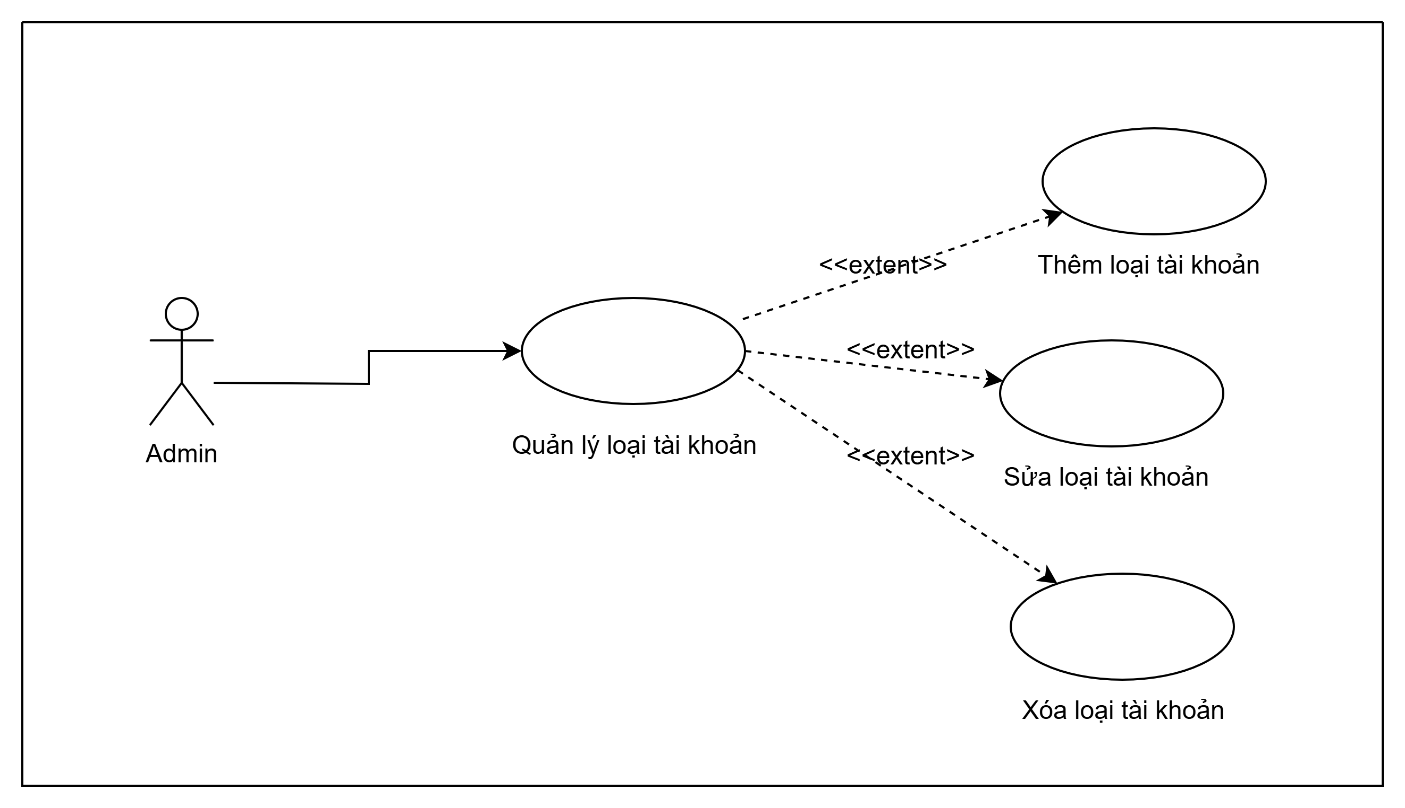
|  |  |
| --- | --- |
| Tên use case | Quản lý sự kiện |
| Mô tả | Use case này cho phép người dùng thêm, sửa, xóa sự kiện trong họ |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùnng truy cập vào quản lý sự kiện. 2. Admin chọn biểu tượng thêm sự kiện. 3. Hệ thống hiện form nhập thêm mới sự kiện. 4. Người dùng điền thông tin mới và nhấn “Lưu” 5. Hệ thống kiểm tra thông tin người dùng nhập và xử lý. 6. Hệ thống hiển thị thông báo lưu thành công. 7. Admin chọn biểu tượng sửa sự kiện trong danh sách sự kiện. 8. Hệ thống hiện form và điền sẵn thông tin đã có. 9. Người dùng cập nhật thông tin và nhấn “Lưu” 10. Hệ thống kiểm tra thông tin người dùng nhập và xử lý. 11. Hệ thống hiển thị thông báo lưu thành công. 12. Admin chọn biểu tượng xóa sự kiện trong danh sách. 13. Hệ thống hiện thị thông báo xác nhận xóa. 14. Người dùng bấm nút “OK” 15. Hệ thống kiểm tra thông tin và xử lý. 16. Hệ thống hiển thị thông báo xóa thành công.   Luồng rẽ nhánh:   * Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. * Tại bước thứ 14 trong luồng cơ bản, nếu người dùng bấm nút “Cancel” thì thông tin sẽ không được xóa và use case kết thúc |
| Tiền điều kiện | * Tài khoản đăng nhập là tài khoản admin |
| Hậu điều kiện |  |



Hình 2.2.7 Biểu đồ phân rã usecase quản lý sự kiện

### 2.2.8 Đặc tả use case quản lý loại tài khoản

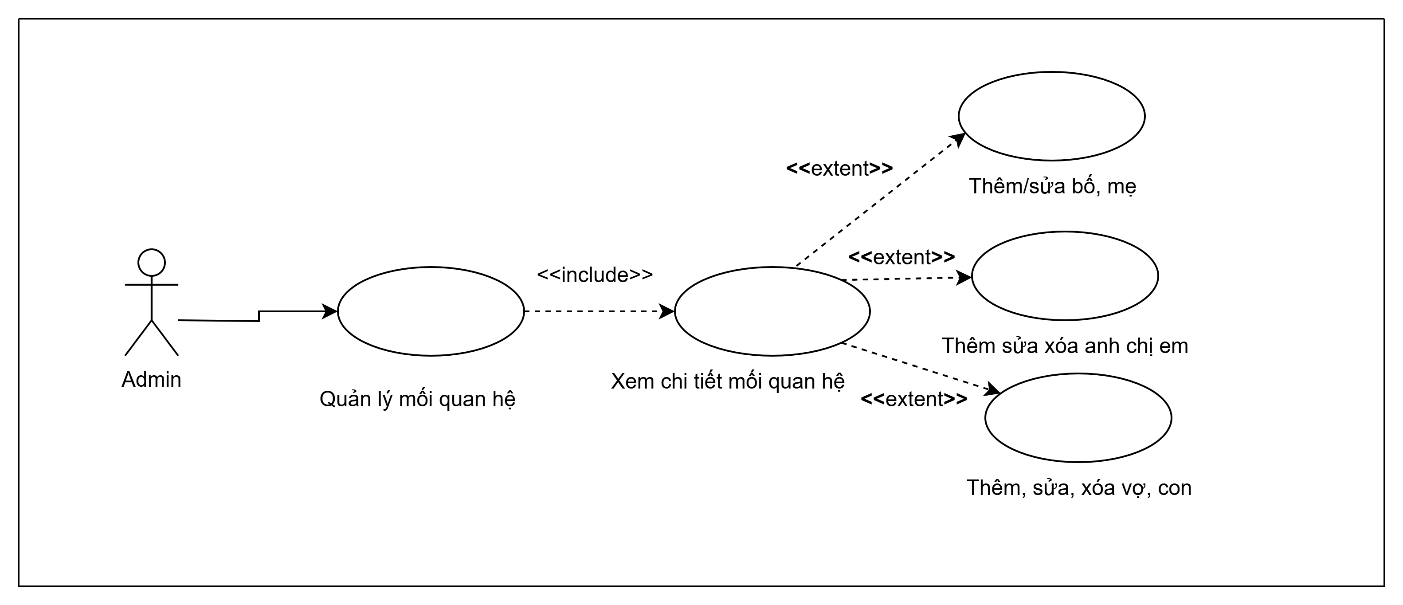
|  |  |
| --- | --- |
| Tên use case | Quản lý loại tài khoản |
| Mô tả | Use case này cho phép người dùng thêm, sửa, xóa loại tài khoản trong hệ thống |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùnng truy cập vào trang loại tài khoản. 2. Admin chọn biểu tượng thêm loại tài khoản. 3. Hệ thống hiện form nhập thêm mới loại tài khoản. 4. Người dùng điền thông tin mới và nhấn “Lưu” 5. Hệ thống kiểm tra thông tin người dùng nhập và xử lý. 6. Hệ thống hiển thị thông báo lưu thành công. 7. Admin chọn biểu tượng sửa loại tài khoản trong danh sách. 8. Hệ thống hiện form và điền sẵn thông tin đã có. 9. Người dùng cập nhật thông tin và nhấn “Lưu” 10. Hệ thống kiểm tra thông tin người dùng nhập và xử lý. 11. Hệ thống hiển thị thông báo lưu thành công. 12. Admin chọn biểu tượng xóa trong danh sách. 13. Hệ thống hiện thị thông báo xác nhận xóa. 14. Người dùng bấm nút “OK” 15. Hệ thống kiểm tra thông tin và xử lý. 16. Hệ thống hiển thị thông báo xóa thành công.   Luồng rẽ nhánh:   * Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. * Tại bước thứ 14 trong luồng cơ bản, nếu người dùng bấm nút “Cancel” thì thông tin sẽ không được xóa và use case kết thúc |
| Tiền điều kiện | * Tài khoản đăng nhập là tài khoản admin |
| Hậu điều kiện |  |



Hình 2.2.8 Biểu đồ phân rã use case quản lý tài khoản

### 2.2.9 Đặc tả use case quản lý mối quan hệ

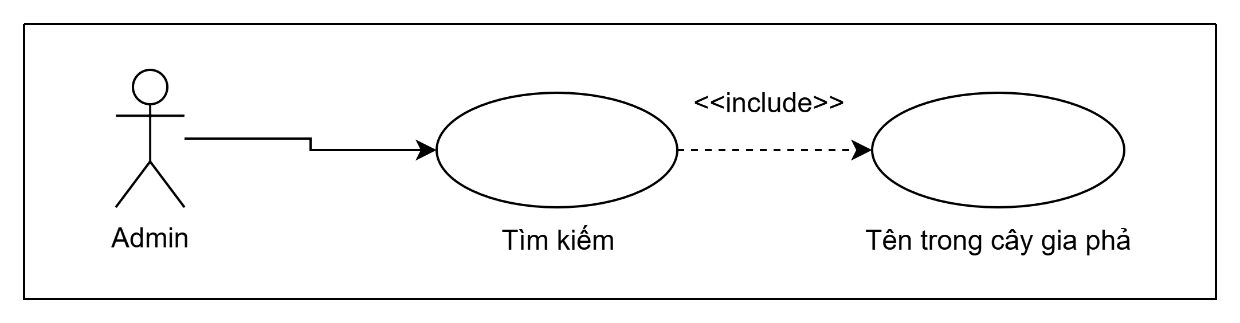
|  |  |
| --- | --- |
| Tên use case | Quản lý mối quan hệ |
| Mô tả | Use case cho phép người dùng thêm, sửa, xóa các mối quan hệ xung quanh như bố me, anh chị em, vợ con |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng truy cập vào trang gia pha của trang web. 2. Người dùng có thể lựa chọn thêm hoặc sửa thông tin của thành viên trong gia đình (bố mẹ, anh chị em, hoặc vợ con). 3. Hệ thống hiển form để người dùng nhập hoặc sửa thông tin 4. Người dùng điền thông tin cần thiết như tên, tuổi, giới tính, năm sinh,… sau đó chọn “lưu” để xác nhận . 5. Hệ thống kiểm tra thông tin Admin nhập và xử lý. 6. Hệ thống hiển thị thông báo thay đổi thành công 7. Người dùng nhấn biểu tượng xóa trong danh sách vợ con hoặc danh sách anh chị em. 8. Hệ thống hiển thị thông báo xác nhận xóa. 9. Người dùng bấm nút “OK” 10. Hệ thống kiểm tra thông tin và thông báo cho người dùng   Luồng rẽ nhánh:   * Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc. * Tại bước 10 trong luồng cơ bản, hệ thống kiểm tra thông tin và phát hiện thông tin không hợp lệ sẽ thông báo cho người dùng |
| Tiền điều kiện | * Người dùng đã thực hiện đăng nhập. * Hệ thống có cơ sở dữ liệu chứa thông tin cá nhân của người dùng. * Người dùng đang thực hiện use case xem chi tiết mối quan hệ * Tài khoản đăng nhập là admin |
| Hậu điều kiện |  |



Hình 2.2.9 Biểu đồ phân rã use case quản lý mối quan hệ

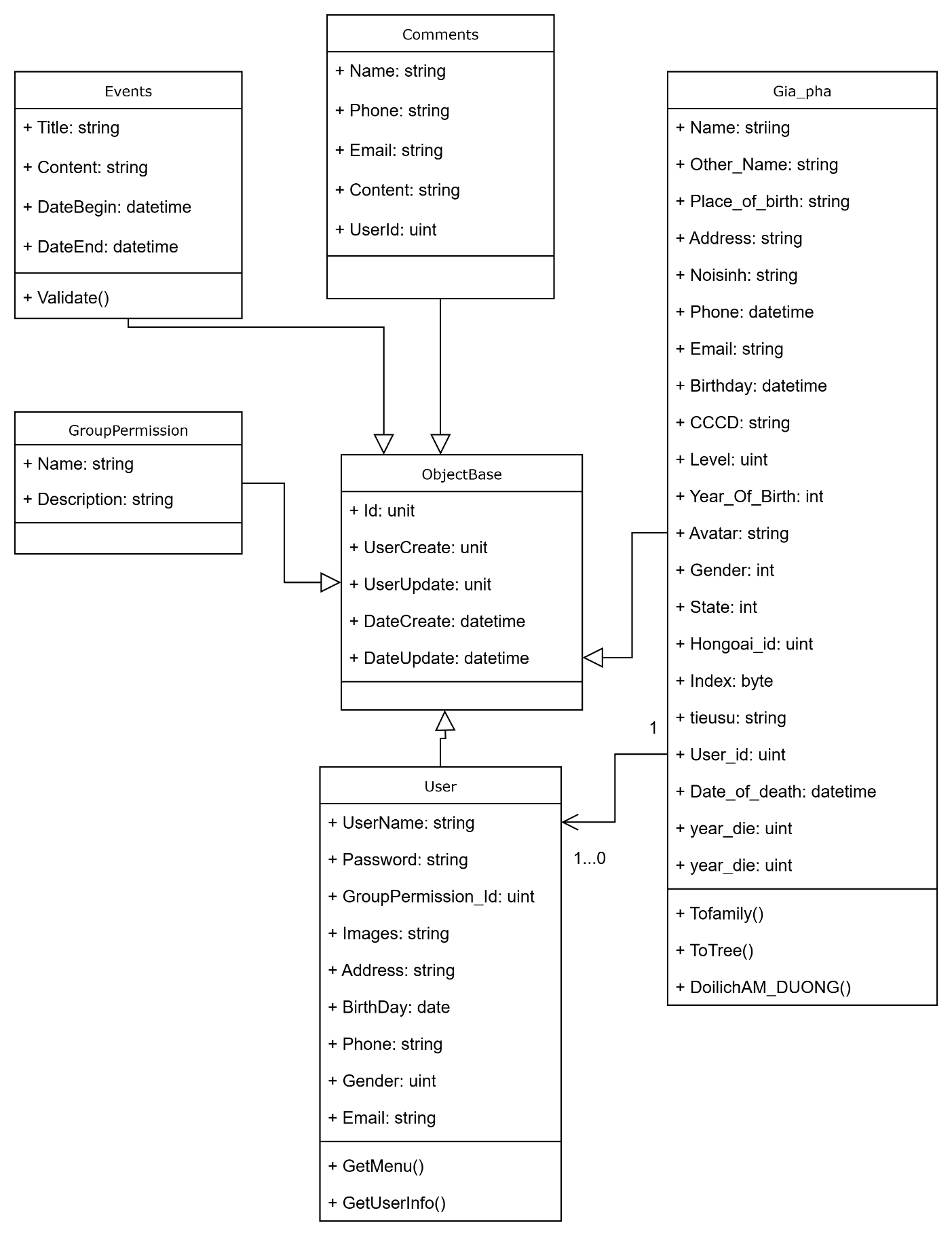
### 2.2.10 Đặc tả use case tìm kiếm

|  |  |
| --- | --- |
| Tên use case | Tìm kiếm |
| Mô tả | Use case cho phép người dùng tìm kiếm thông tin muốn tìm |
| Luồng sự kiện | Luồng cơ bản:   1. Người dùng truy cập vào trang gia pha của trang web. 2. Hệ thống hiển thị ô tìm kiếm ở góc phải màn hình 3. Người dùng chọn trong danh sách đổ ra từ ô tìm kiếm (là 1 danh sách lựa chọn người trong gia phả) 4. Hệ thống tìm kiếm thông tin và di chuyển màn hình đến vị trí người đó trong cây gia phả và làm nổi bật ô thể hiện người đó trong gia phả |
| Tiền điều kiện | * Người dùng đã thực hiện đăng nhập. |
| Hậu điều kiện |  |



Hinh 2.2.10 Biểu đồ phân rã use case tìm kiếm

## 3.3 Biểu đồ lớp



### 3.3.1 Usecase đăng nhập

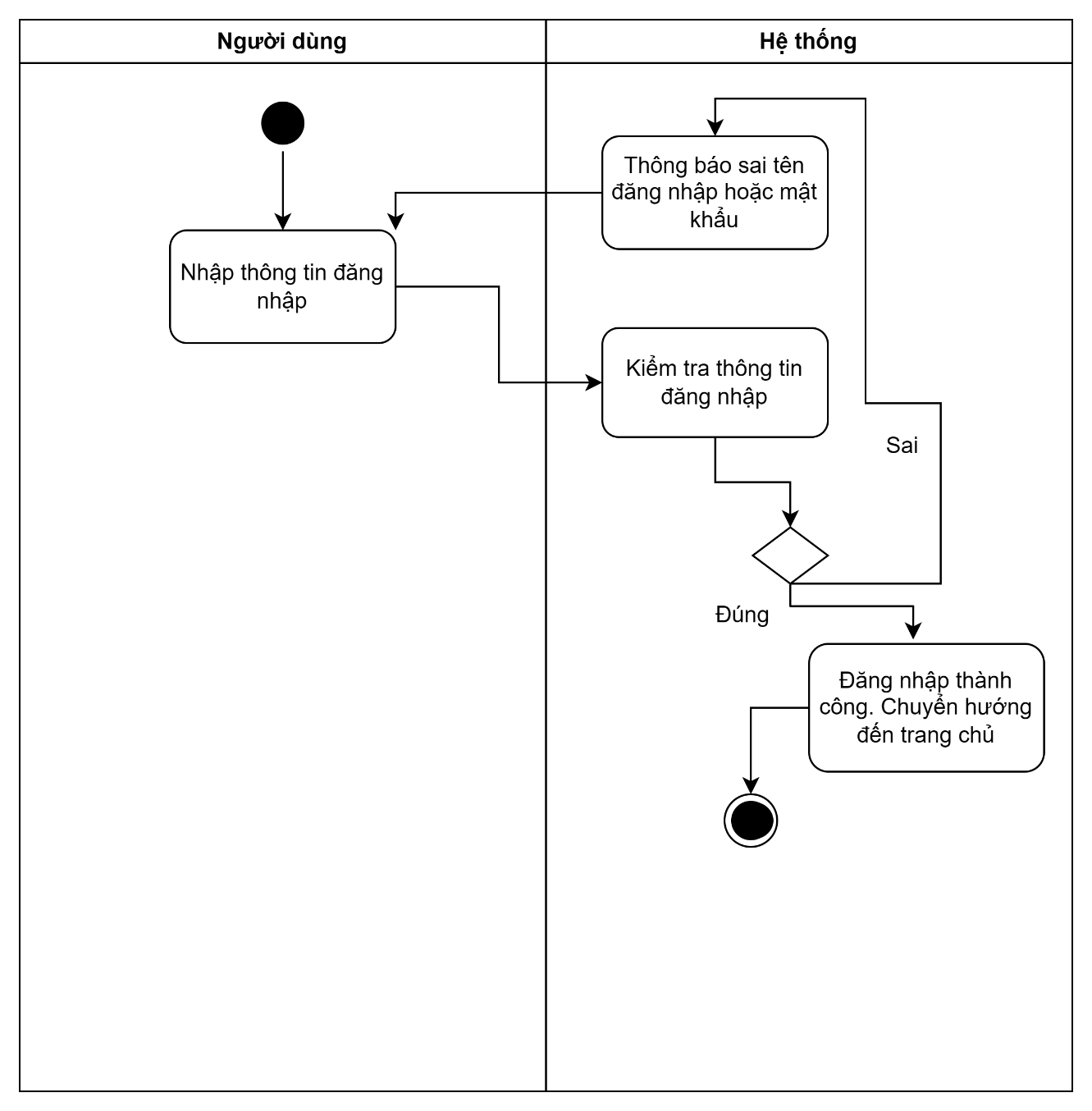


### 3.2.2 Usecase góp ý

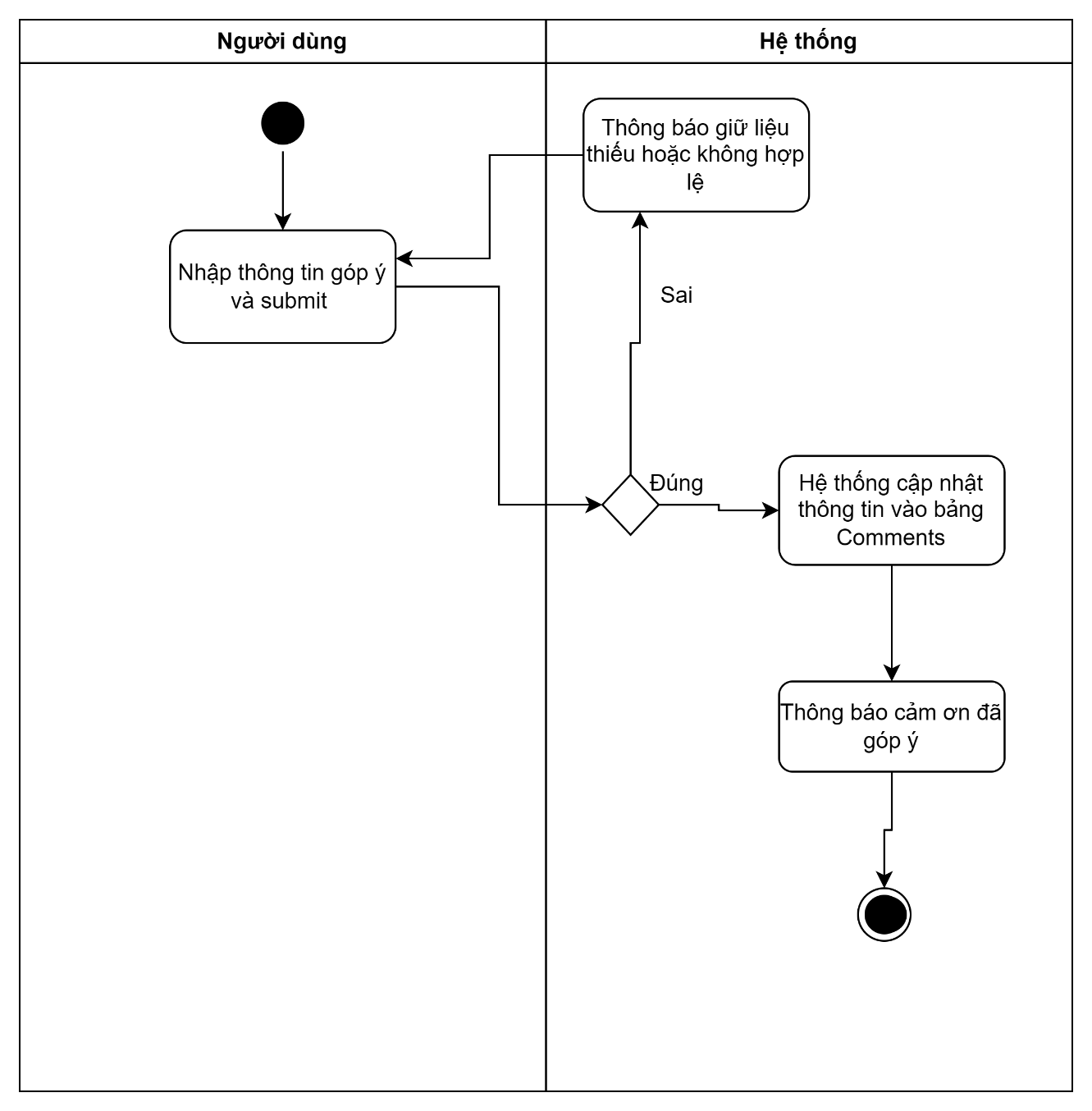


## 3.4 Biểu đồ hoạt động

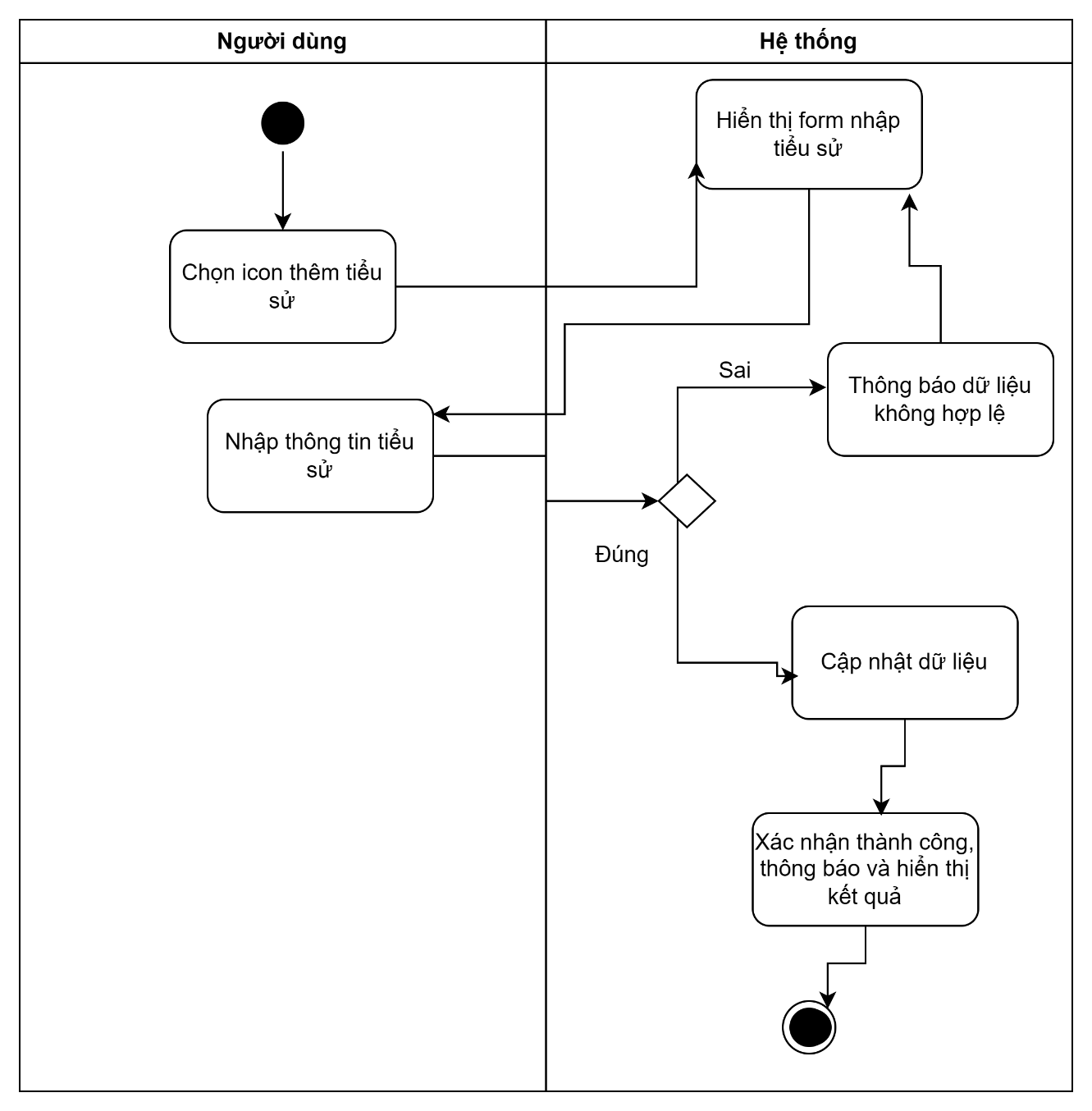
### 3.4.1 Use case đăng nhập



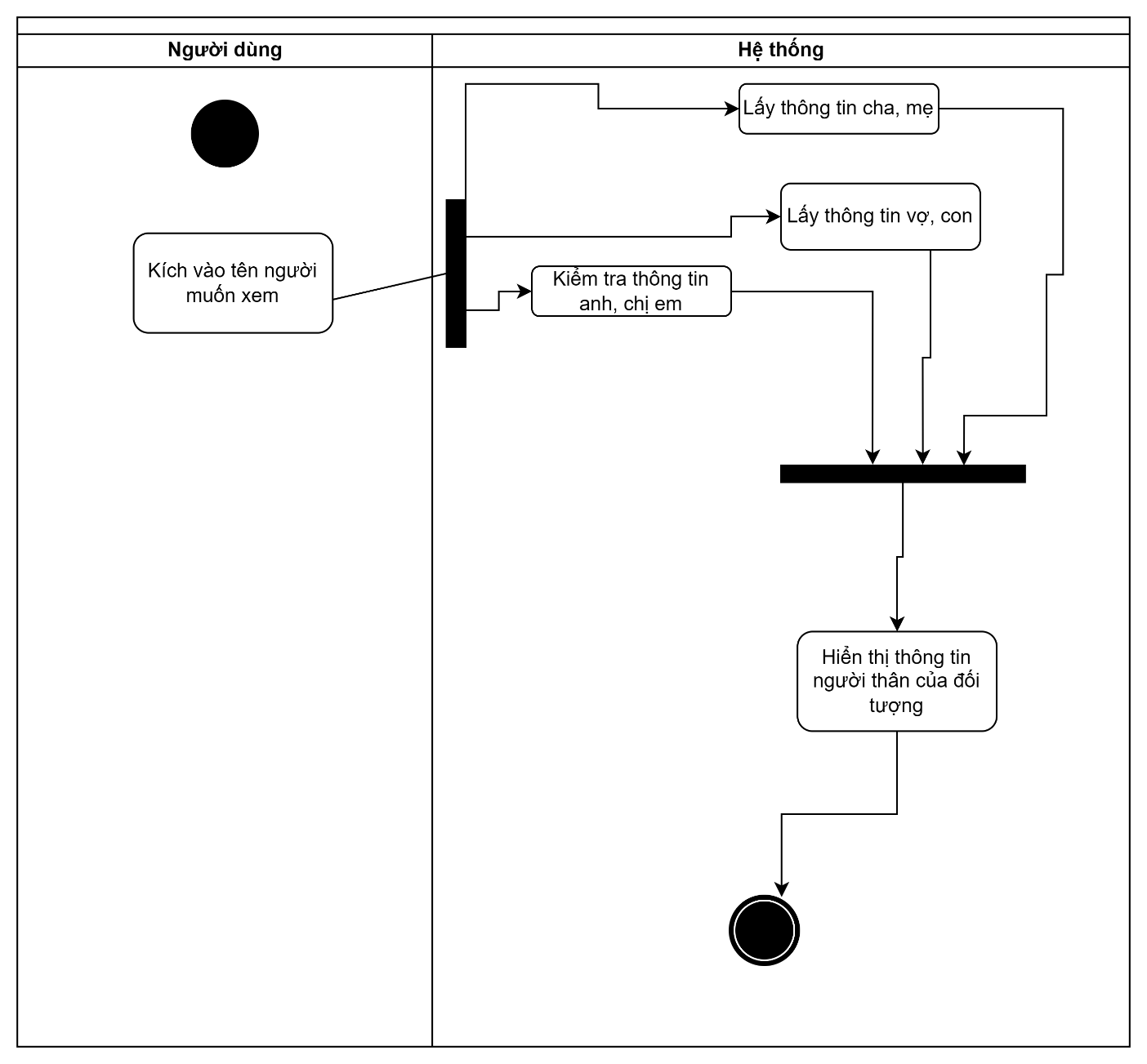
### 3.4.2 Usecase góp ý



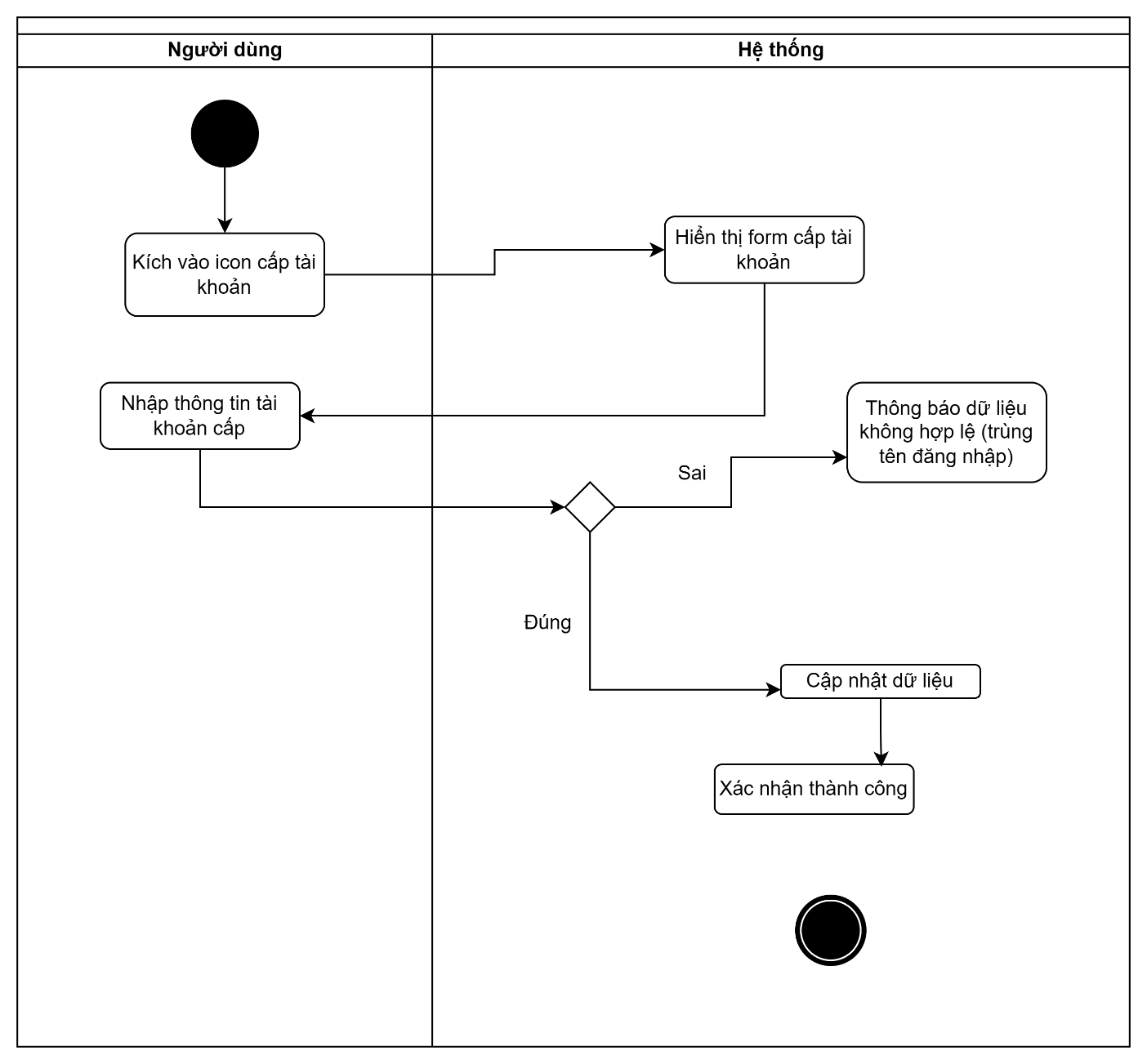
### 3.4.3 Usecase thêm tiểu sử



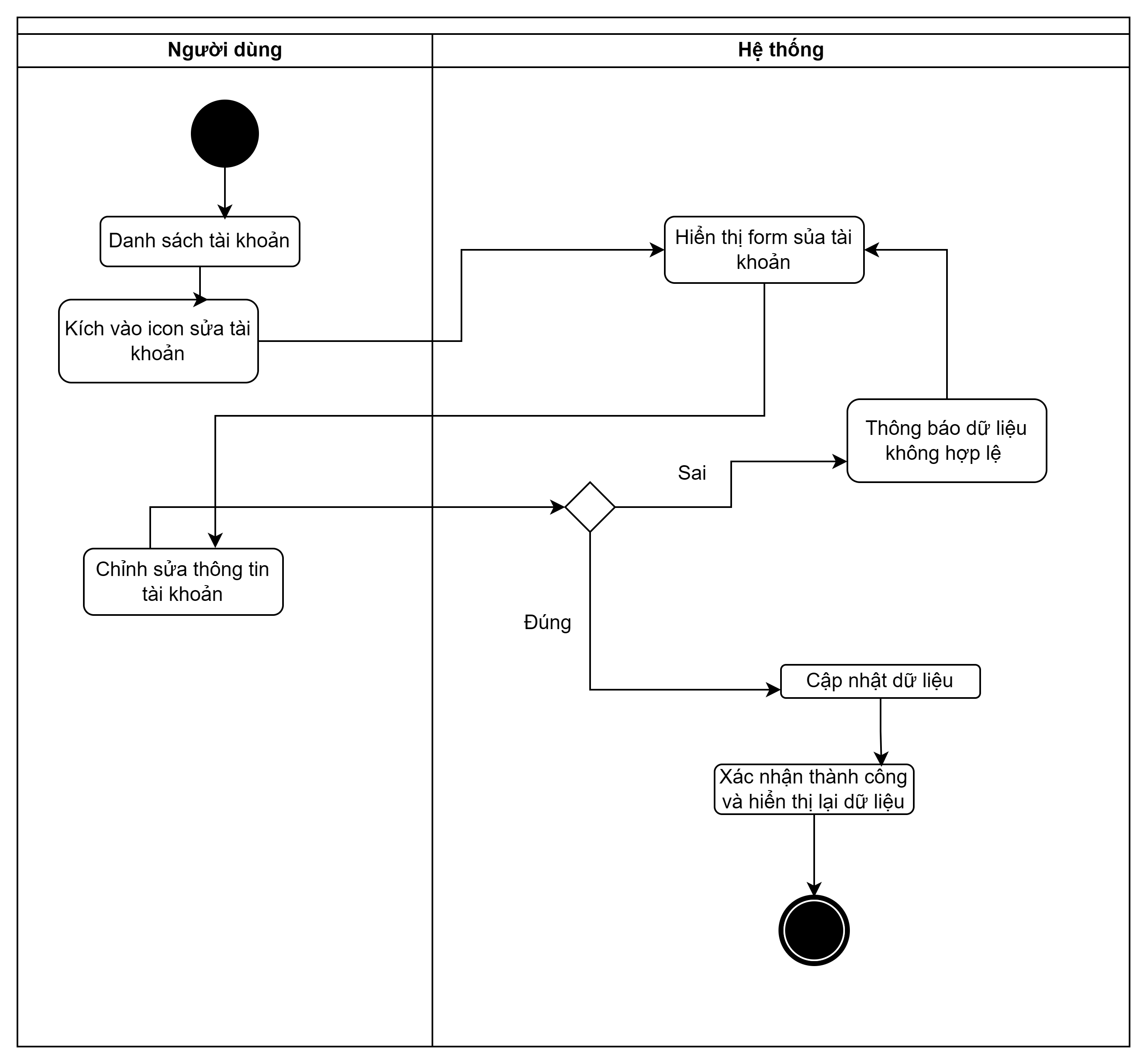
### 3.4.4 Usecase xem chi tiết mối quan hệ



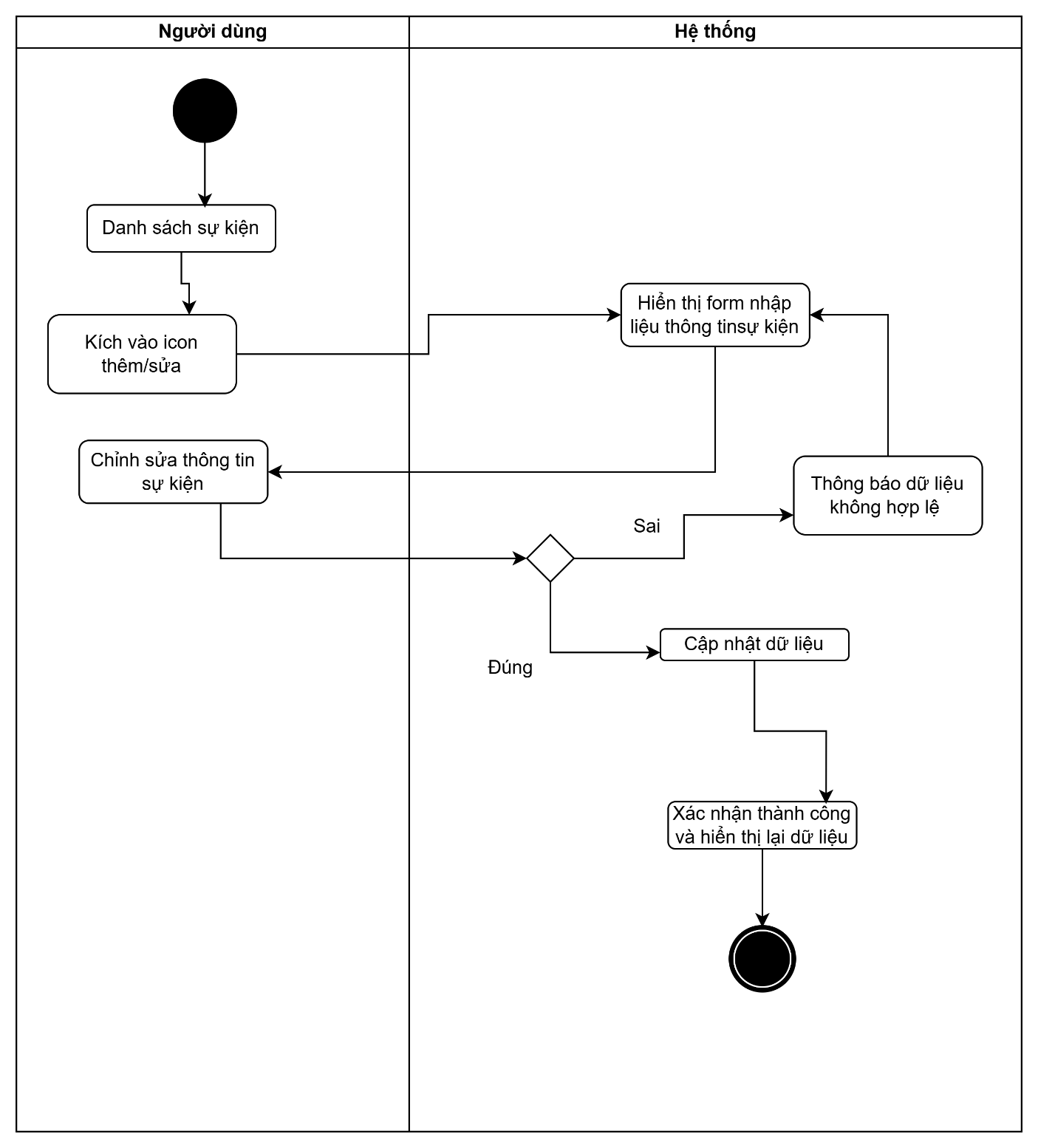
### 3.4.5 Usecase cấp tài khoản

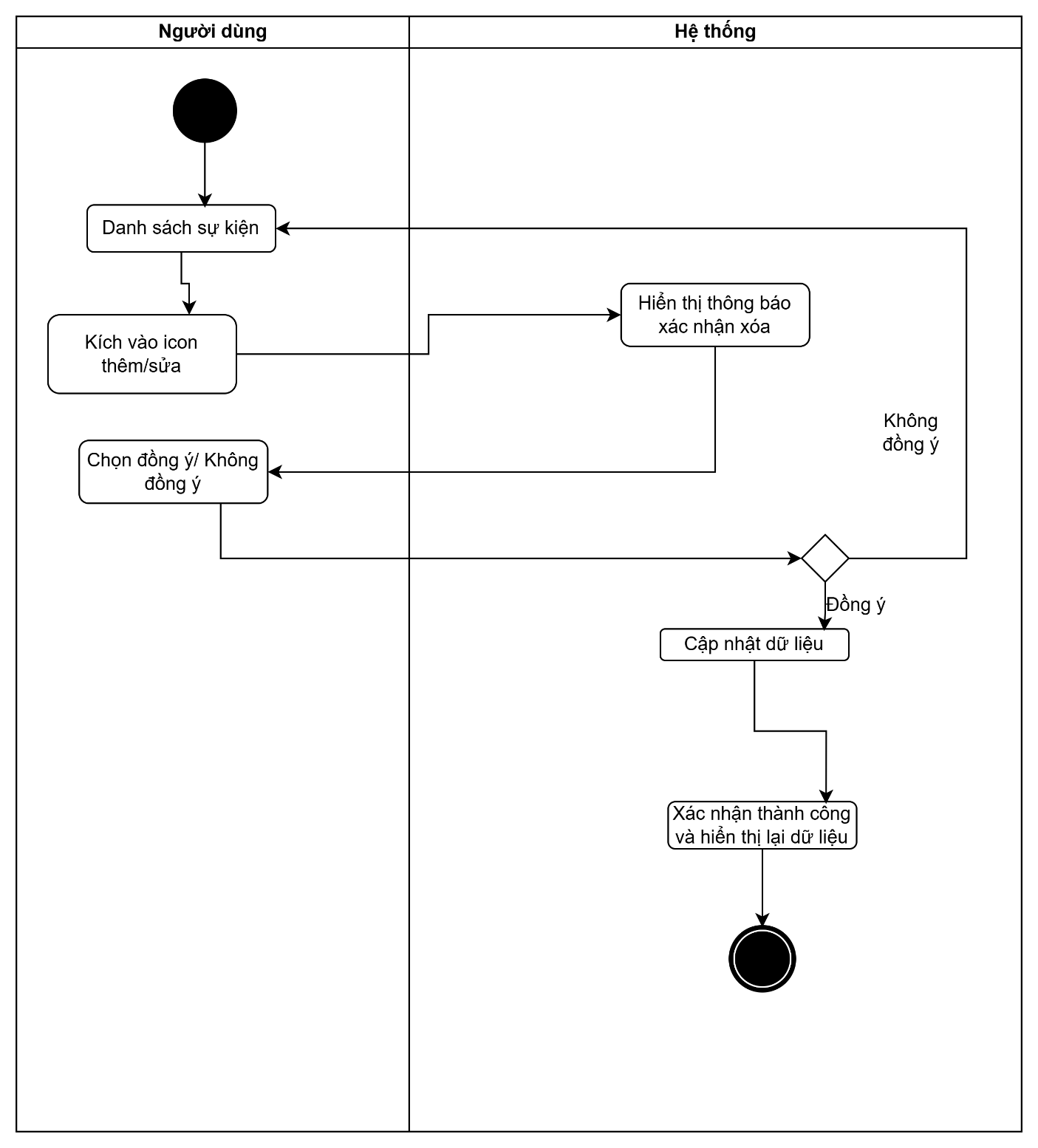


### 3.4.6 Usecase quản lý tài khoản

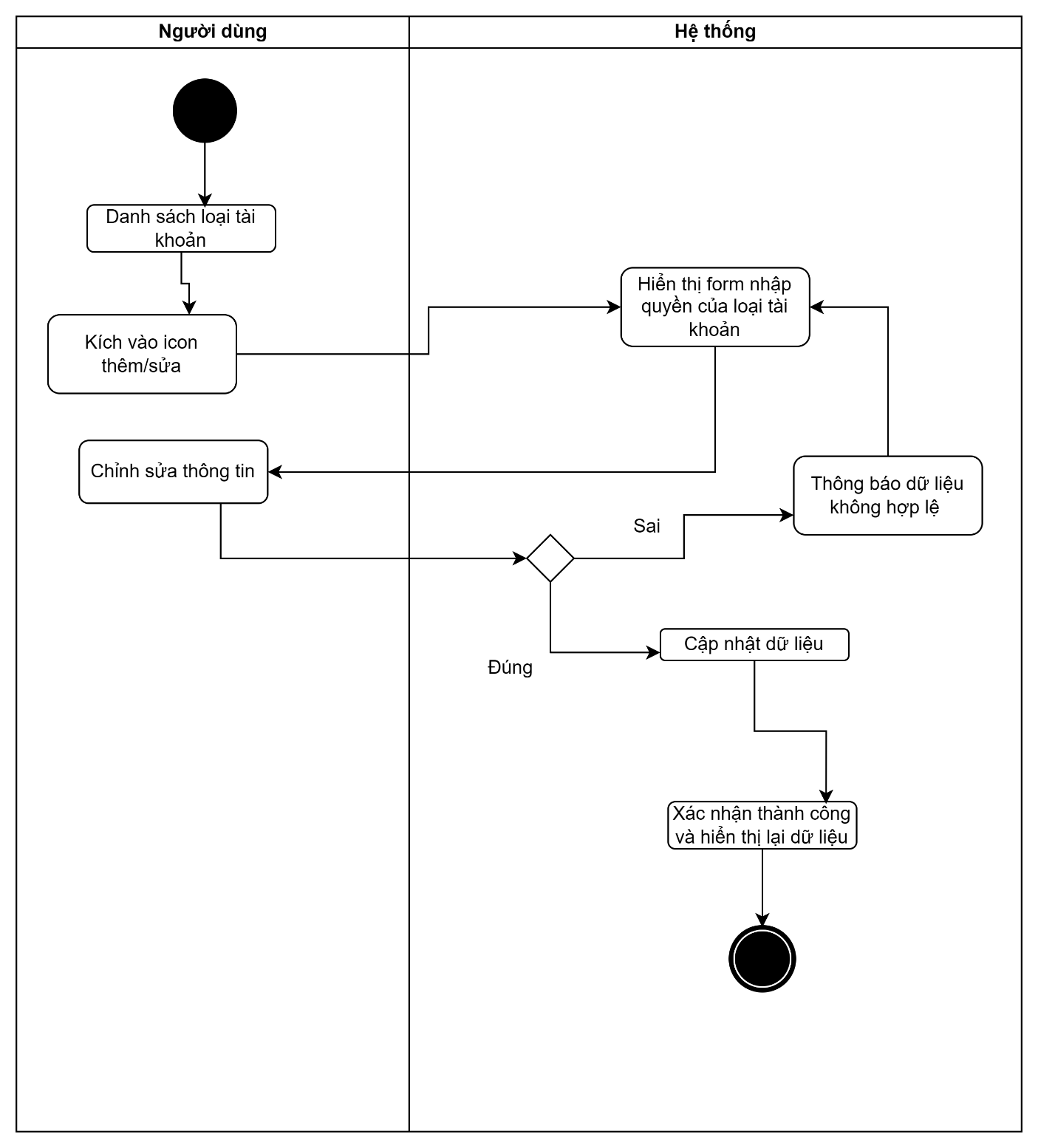


### 3.4.7 Usecase quản lý sự kiện

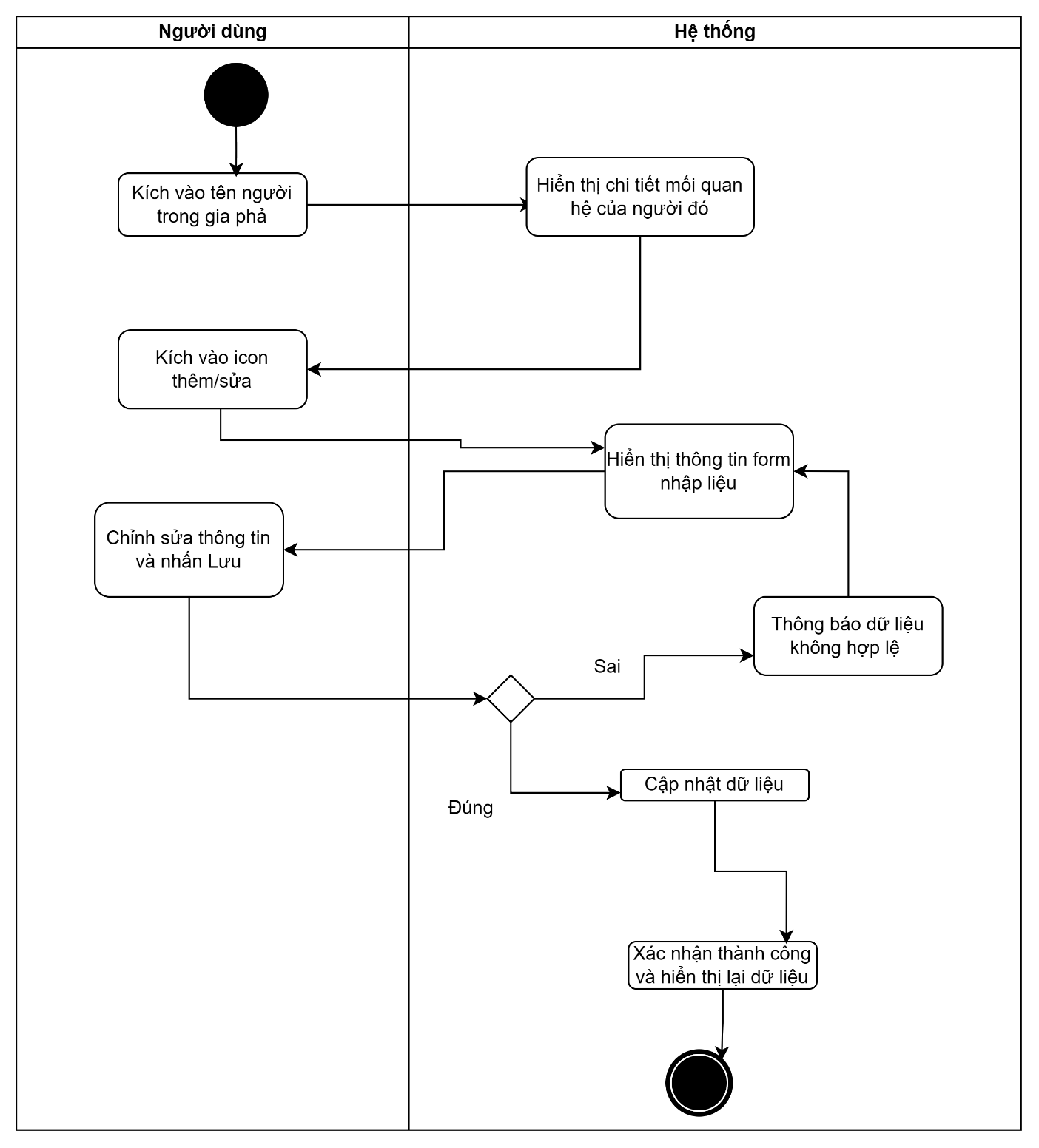


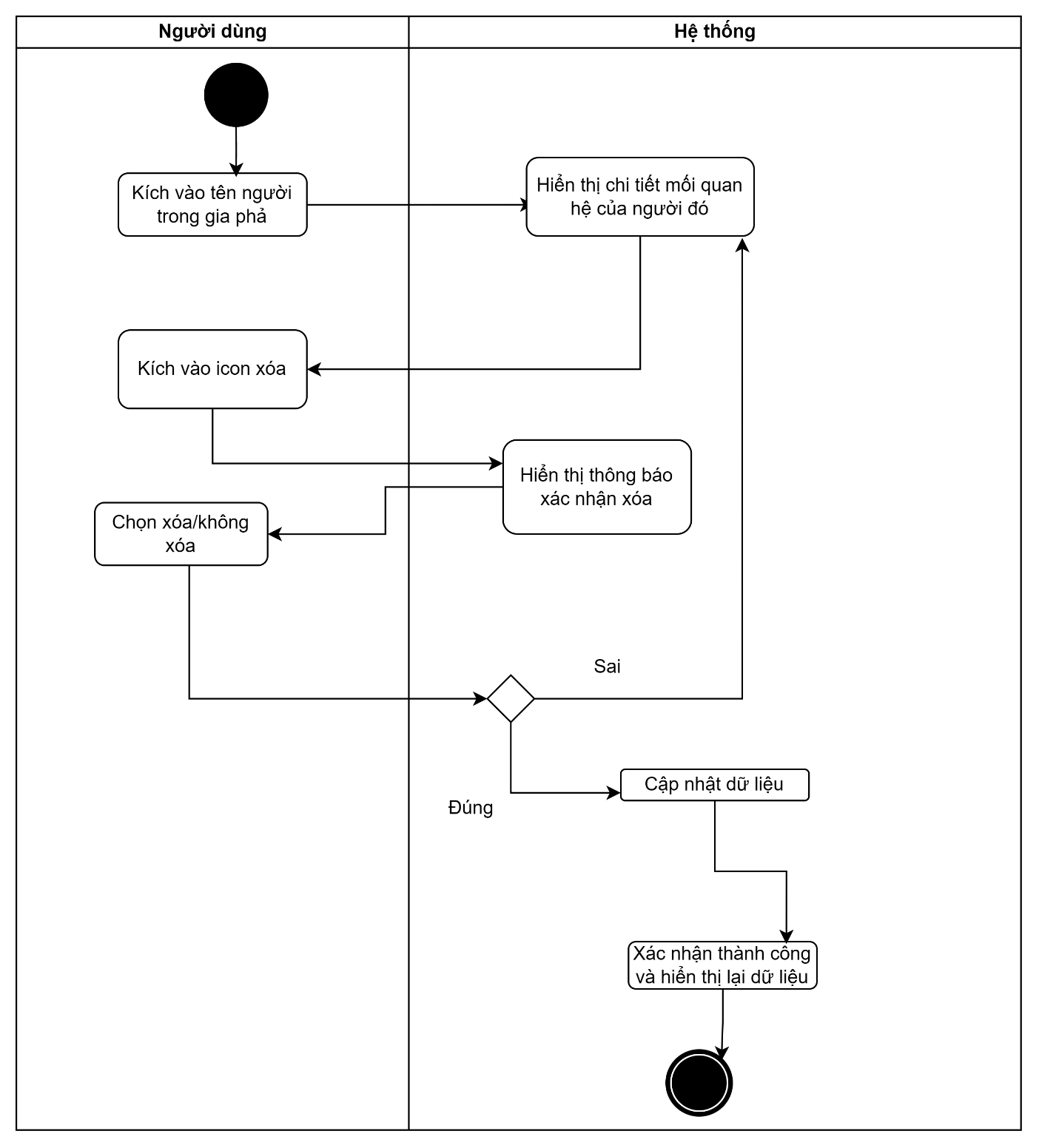


### 3.4.8 Usecase quản lý loại tài khoản



### 3.4.9 Usecase quản lý quan hệ mối





## 3.5 Biểu đồ tuần tự

### 3.5.1 Usecase đăng nhập



### 3.5.2 Usecase góp ý



### 3.5.3 Usecase thêm tiểu sử



### 3.5.4 Use case cấp tài khoản



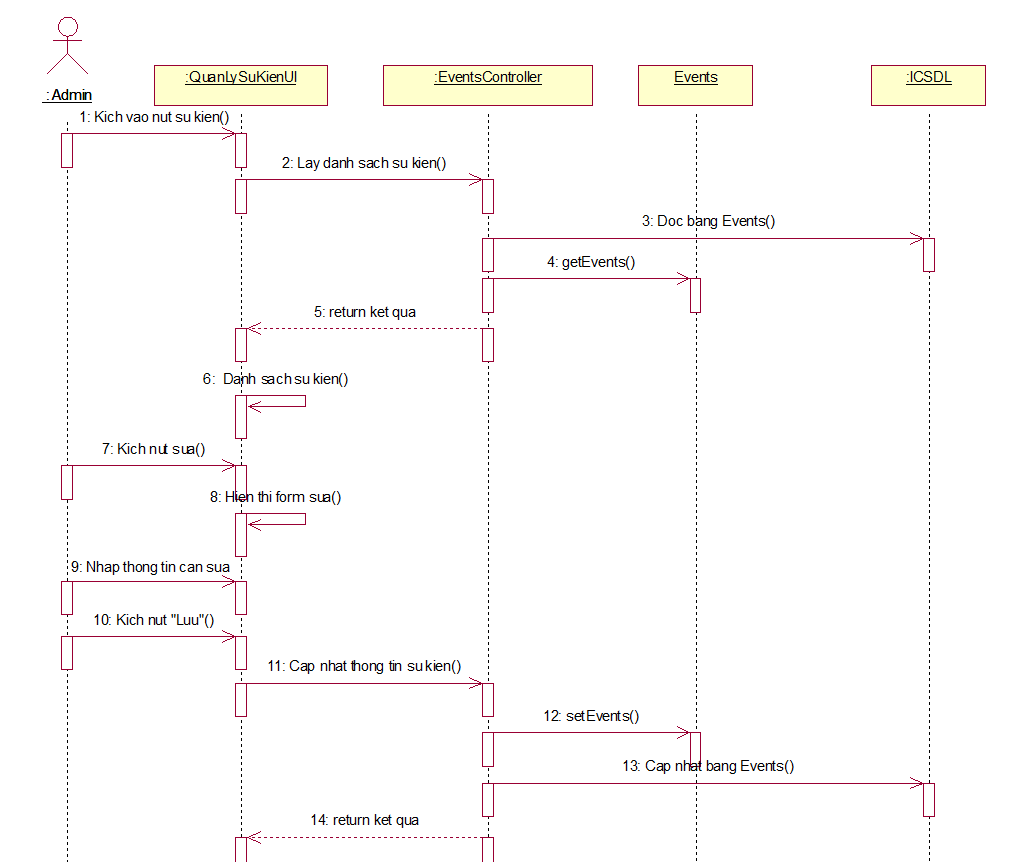
### 3.5.5 Xem chi tiết mối quan hệ

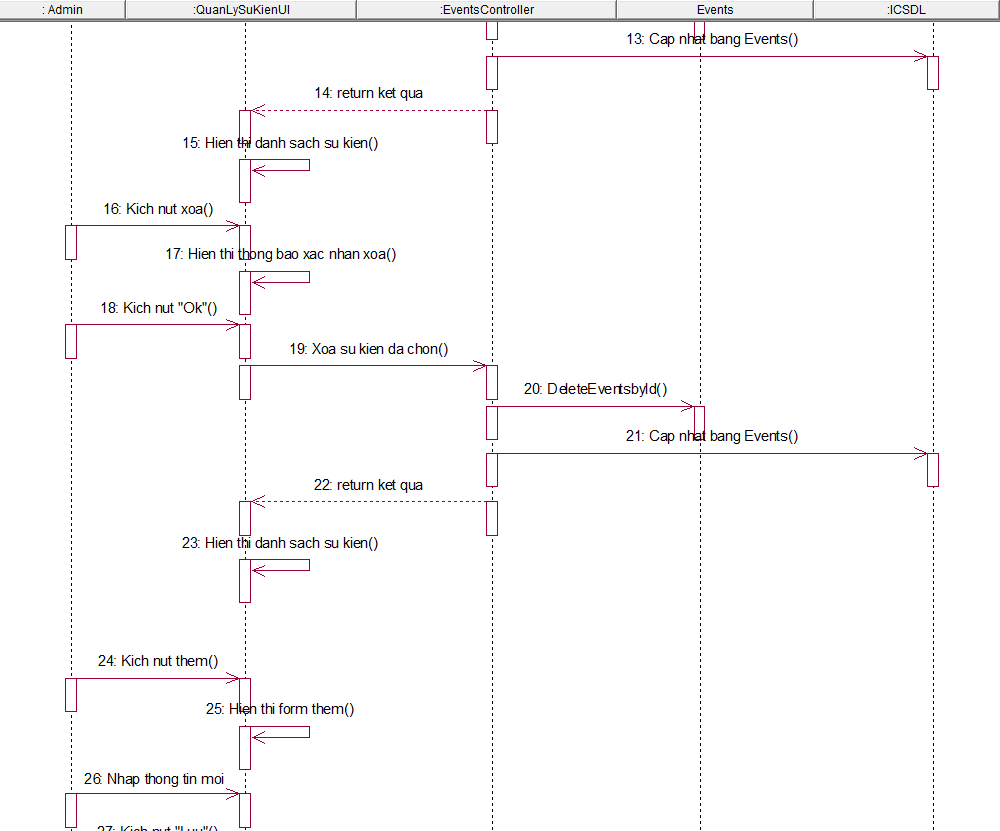


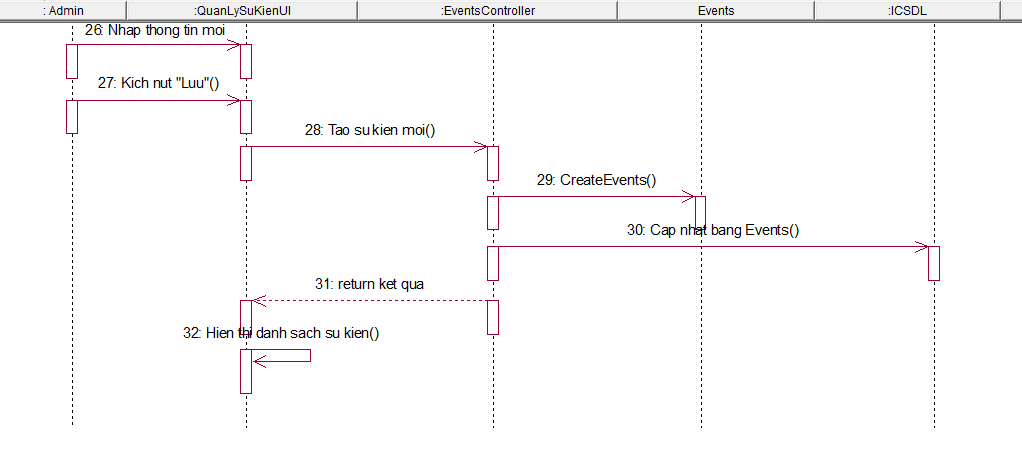
### 3.5.6 Usecase quản lý tài khoản



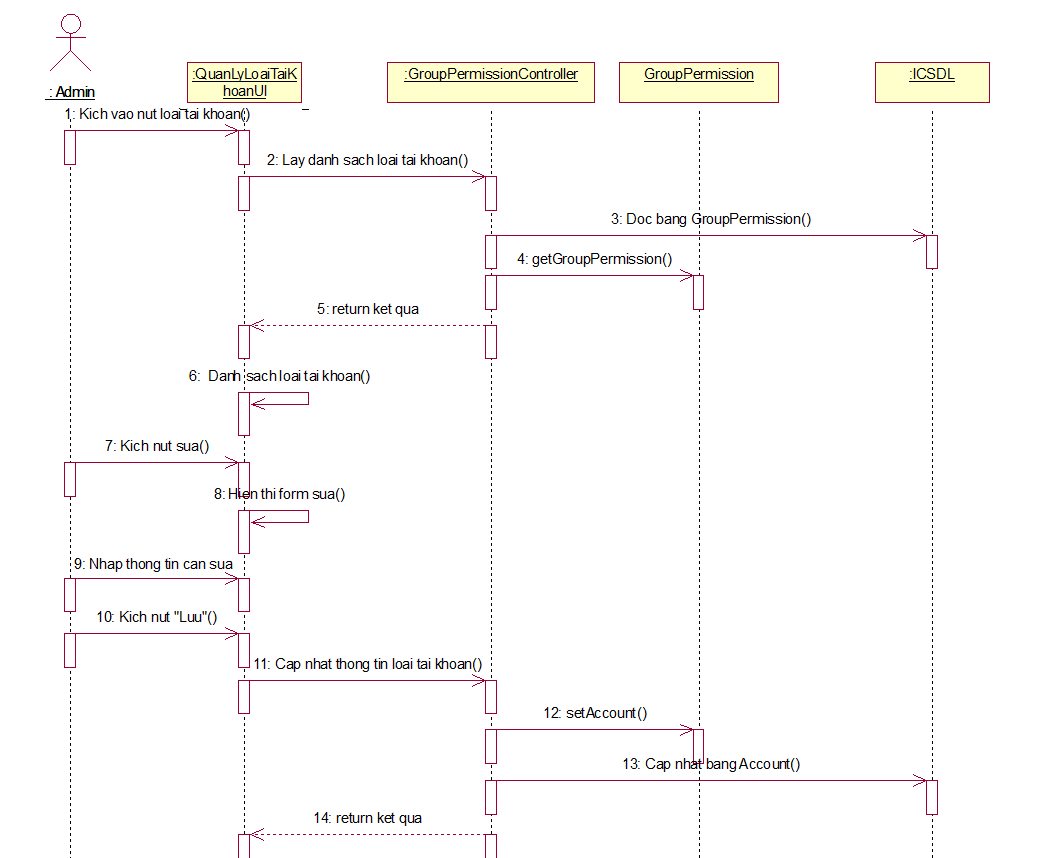
### 3.5.7 Usecase quản lý sự kiện

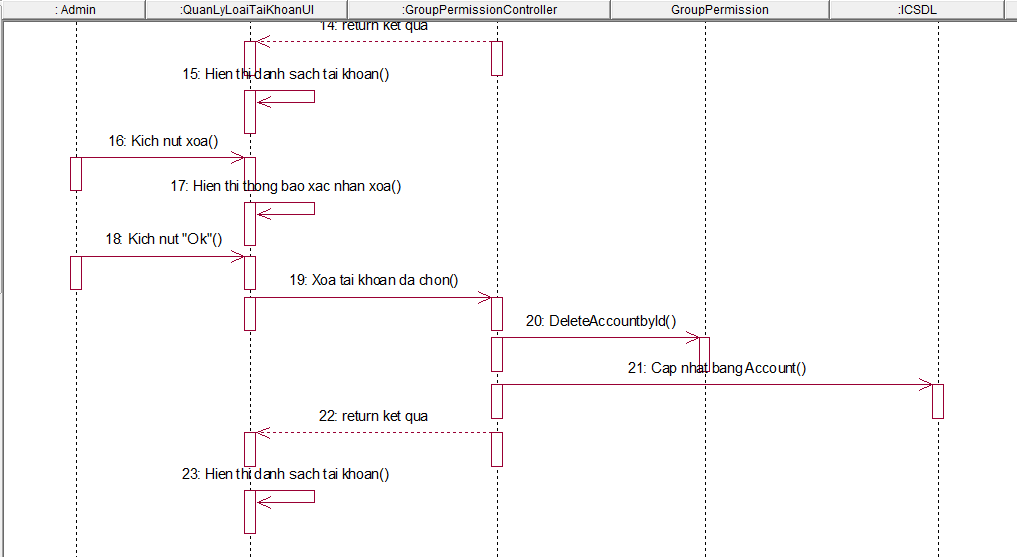


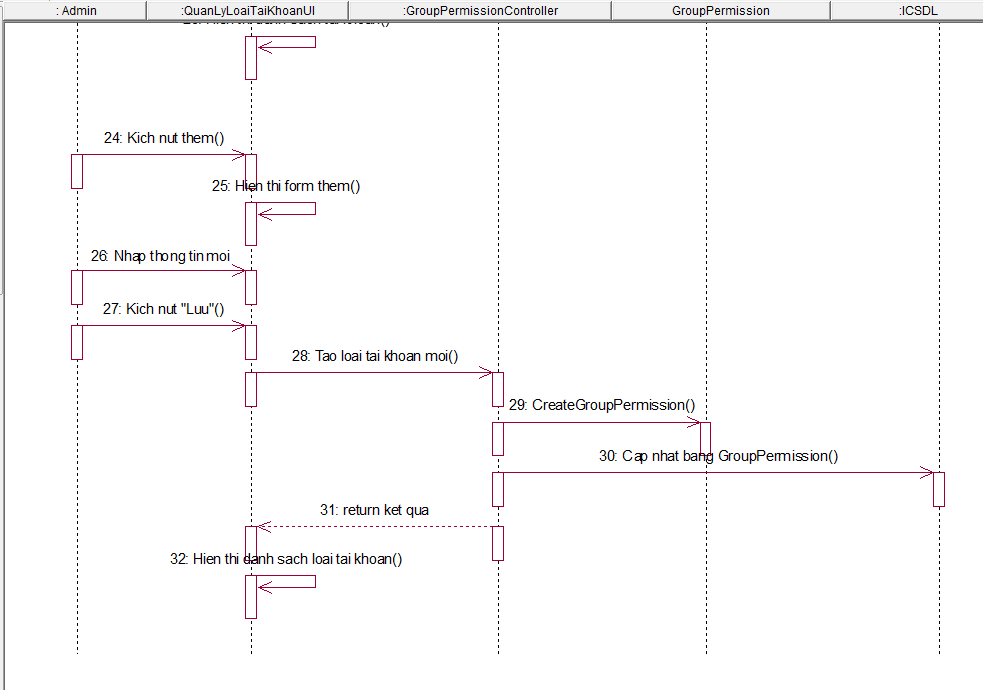




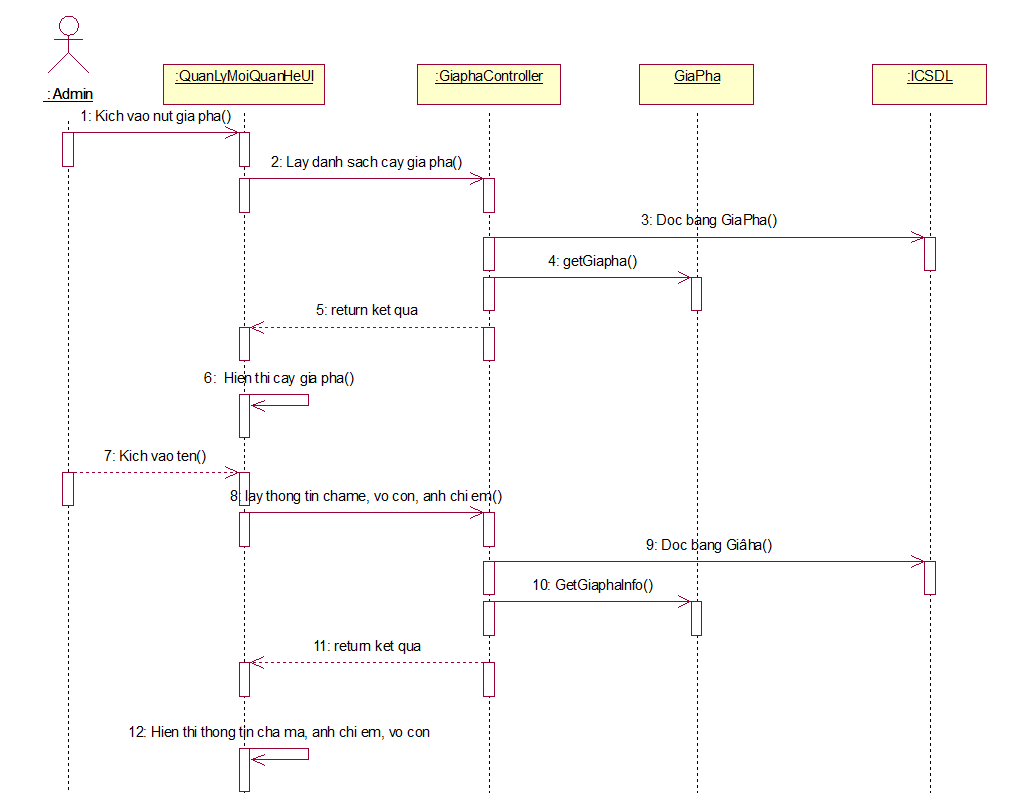
### 3.5.8 Usecase quản lý loại tài khoản

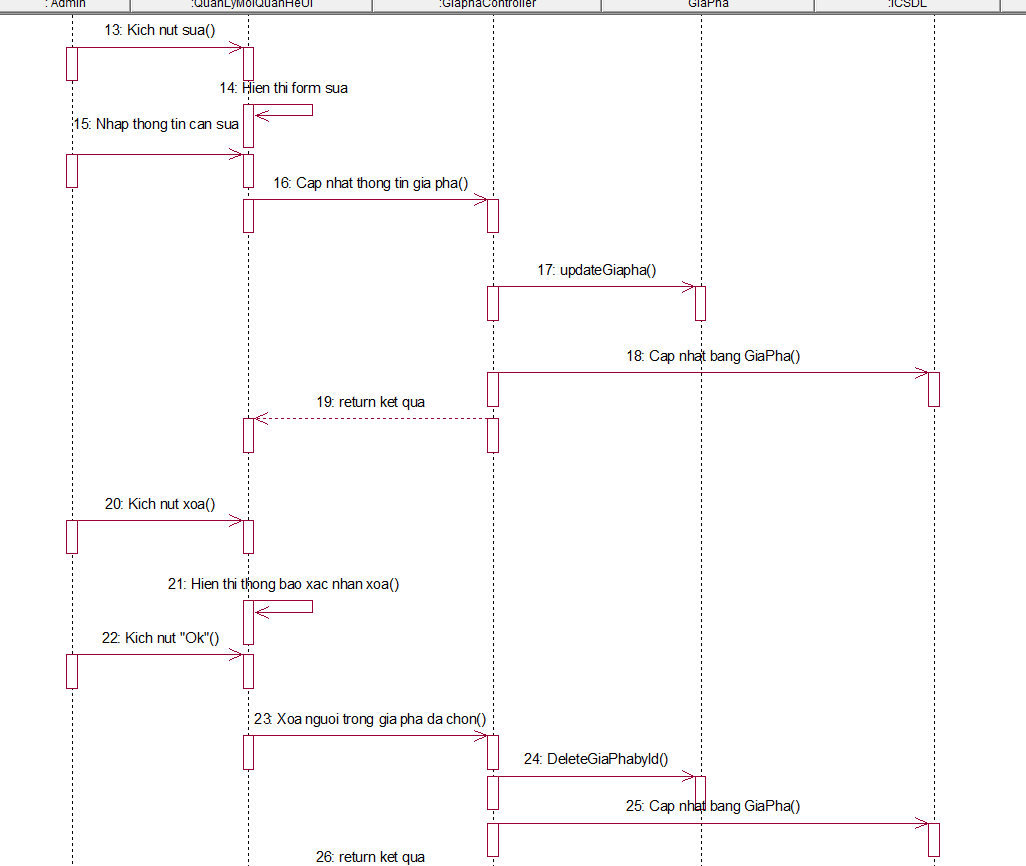
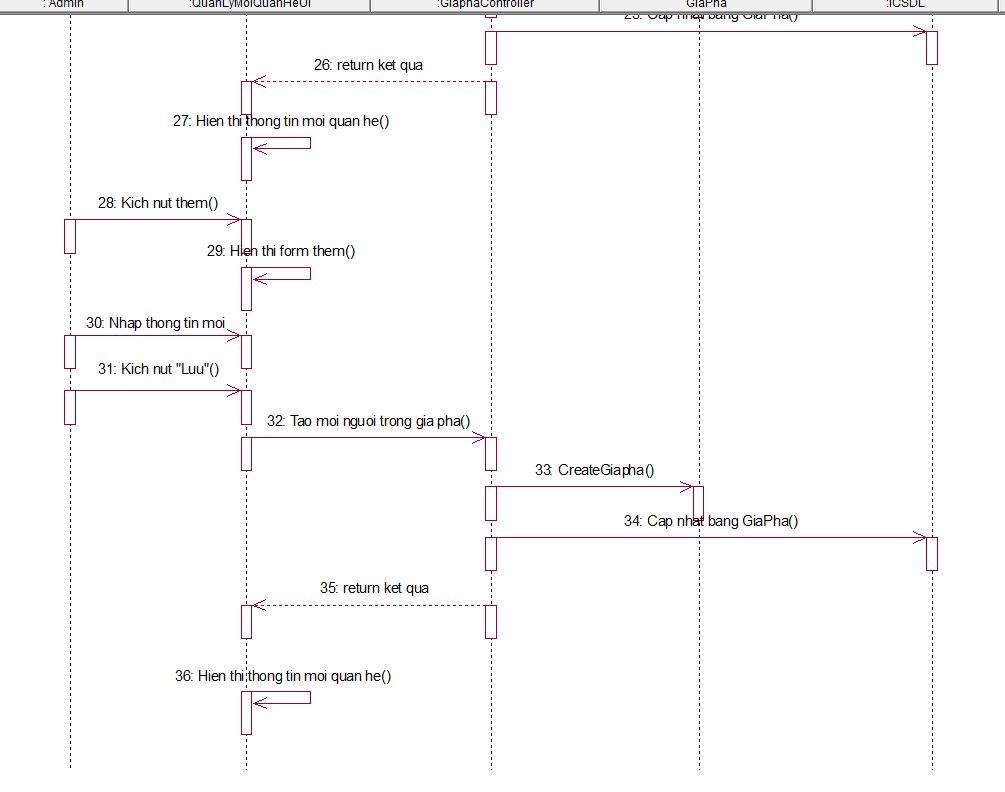






### 3.5.9 Usecase quản lý quan hệ mối quan hệ



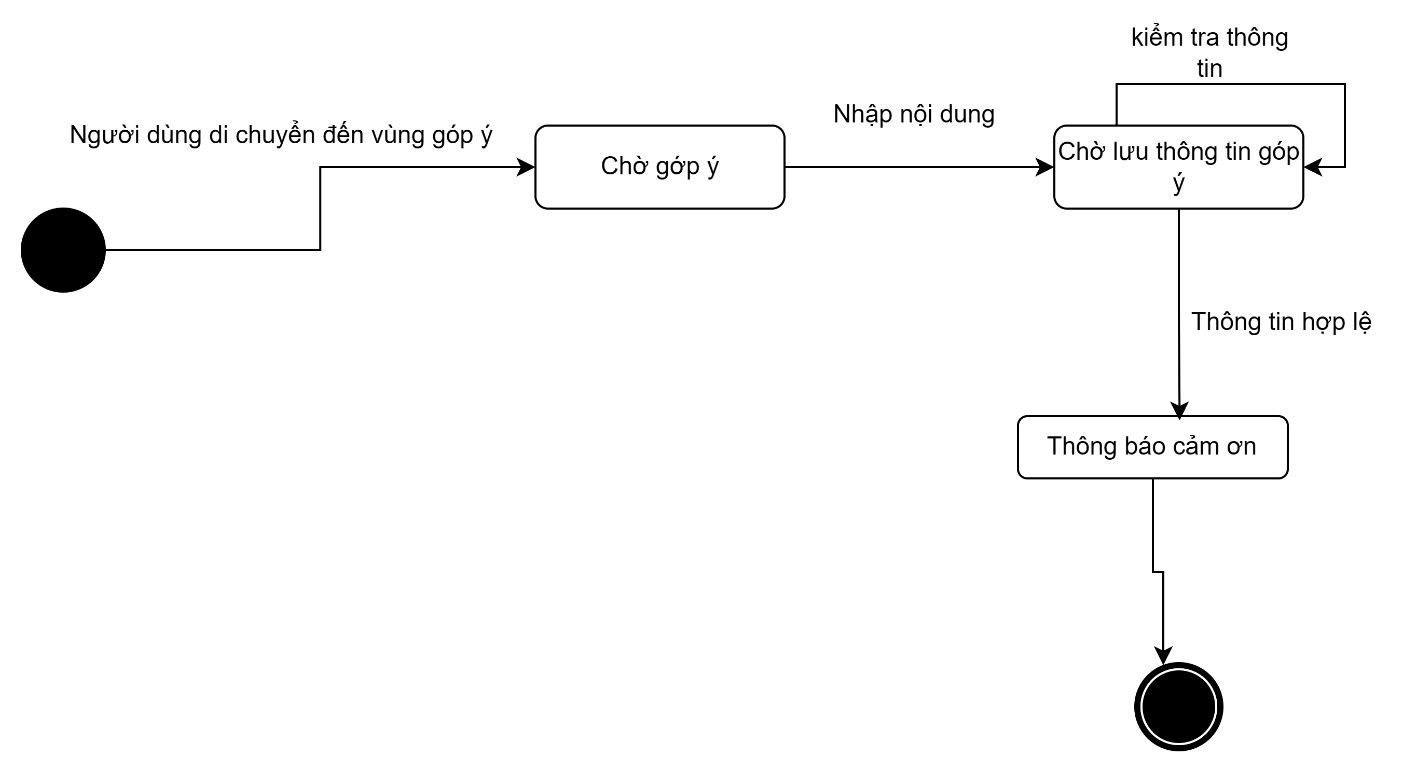
 

## 3.6 Biểu đồ trạng thái

### 3.6.1 Usecase đăng nhập



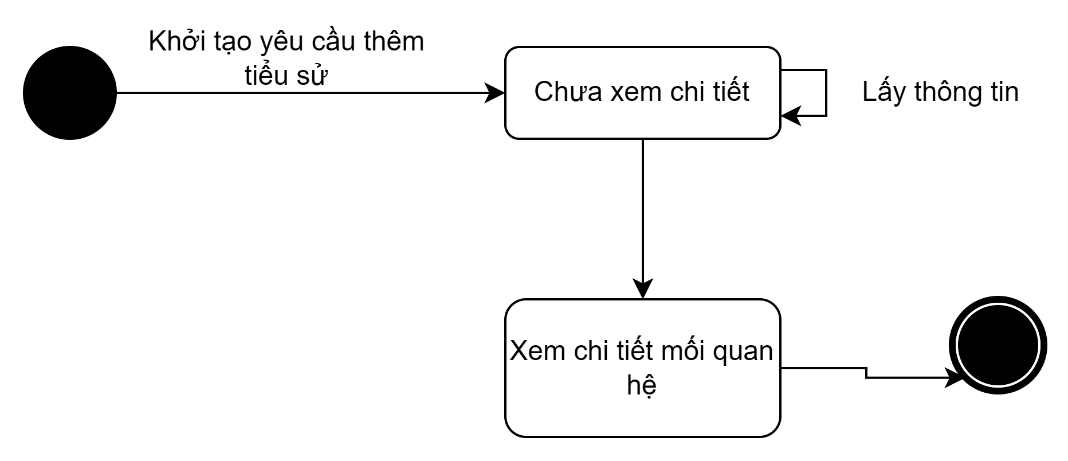
### 3.6.2 Usecase góp ý



### 3.6.3. Thêm tiểu sử

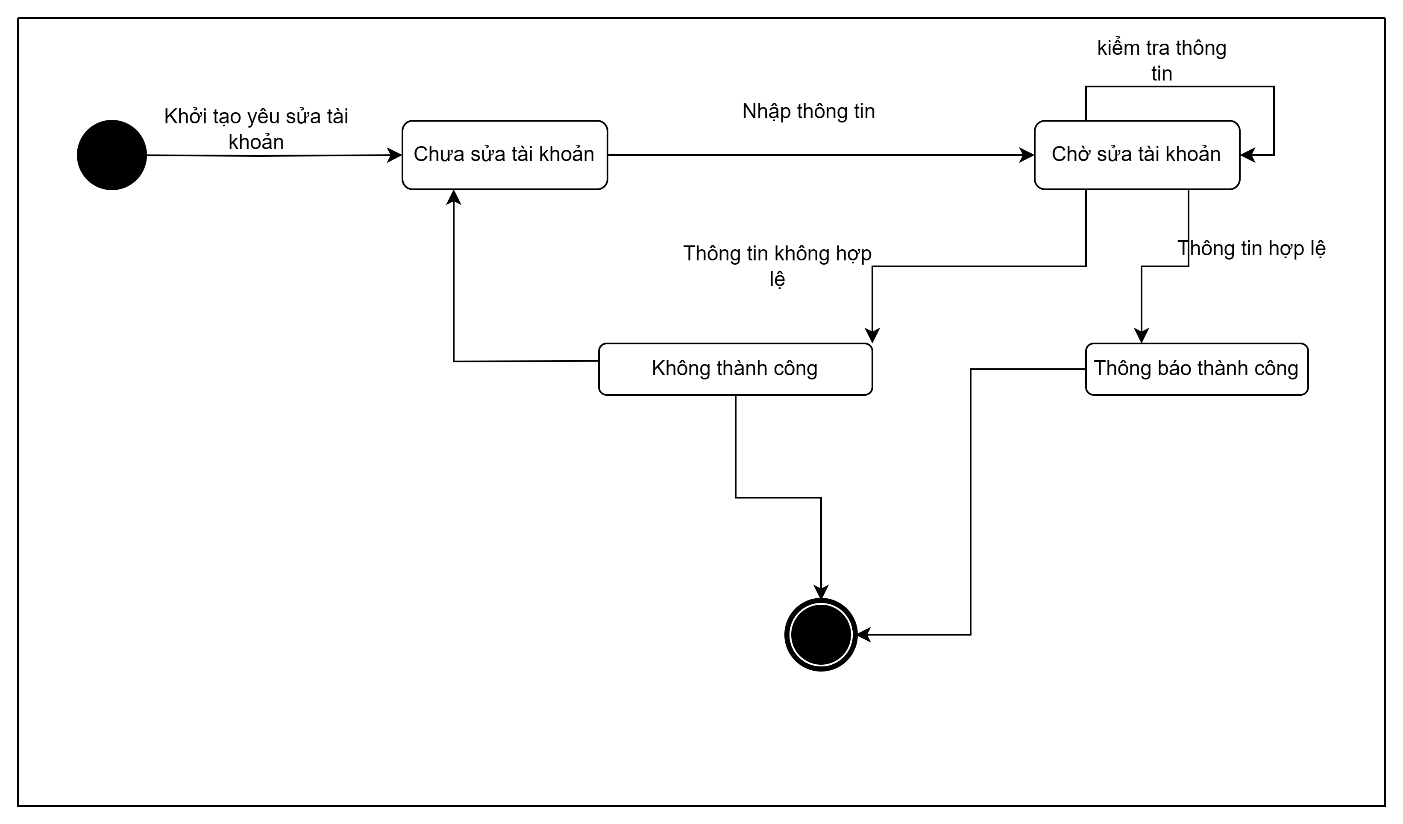


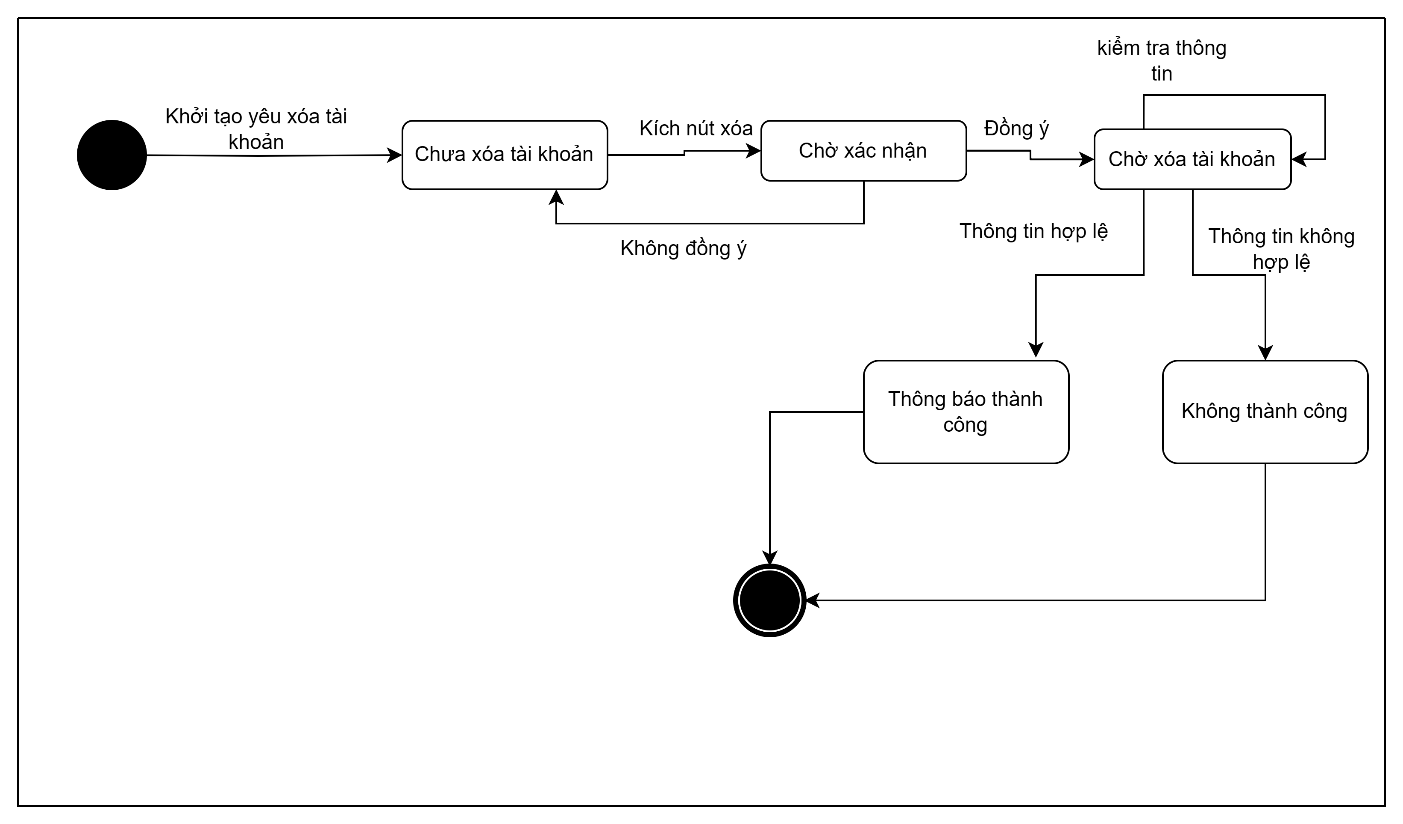
### 3.6.4. Usecase xem chi tiết mối quan hệ



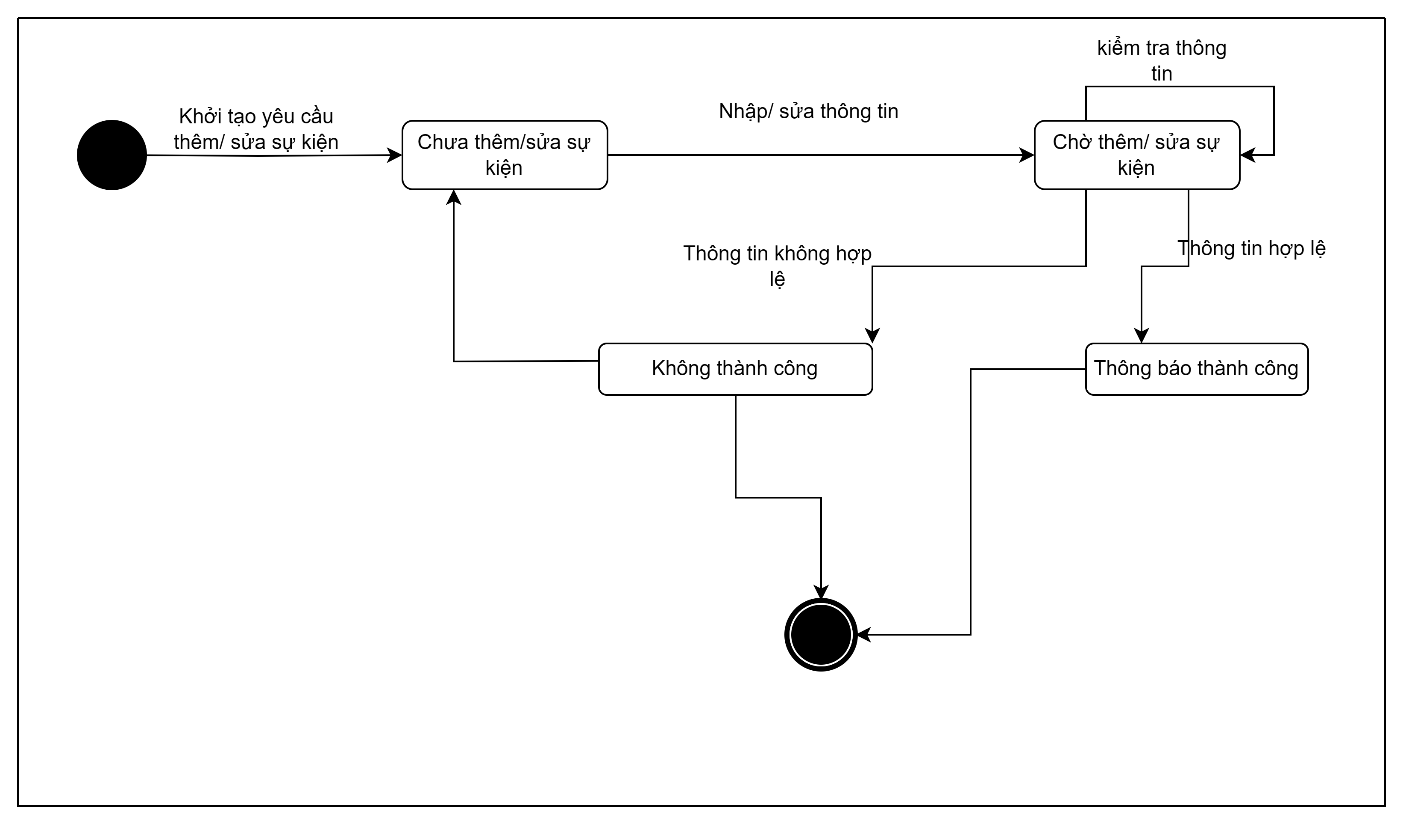
### 3.6.5. Usecase cấp tài khoản

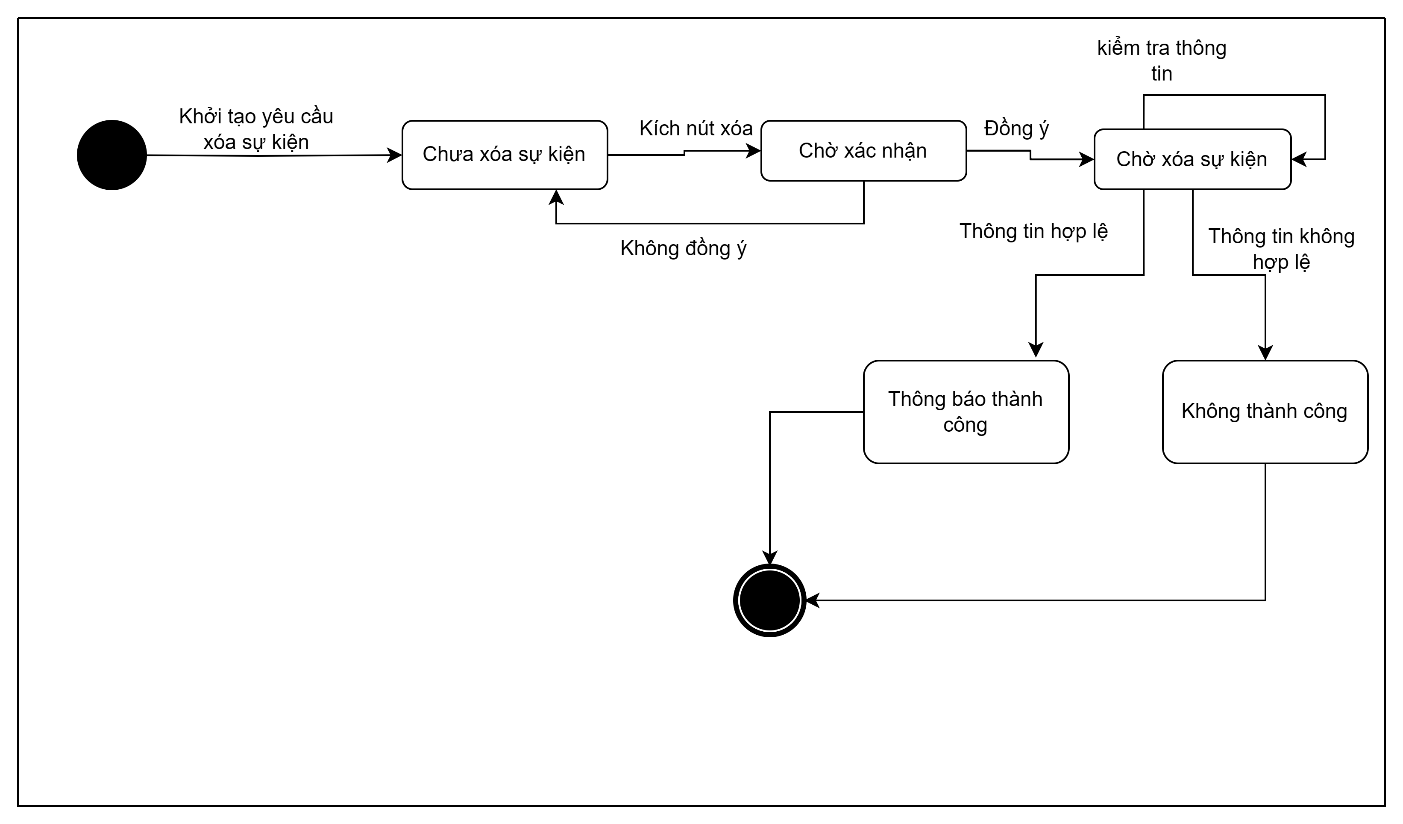
### 3.6.6 Usecase quản lý tài khoản



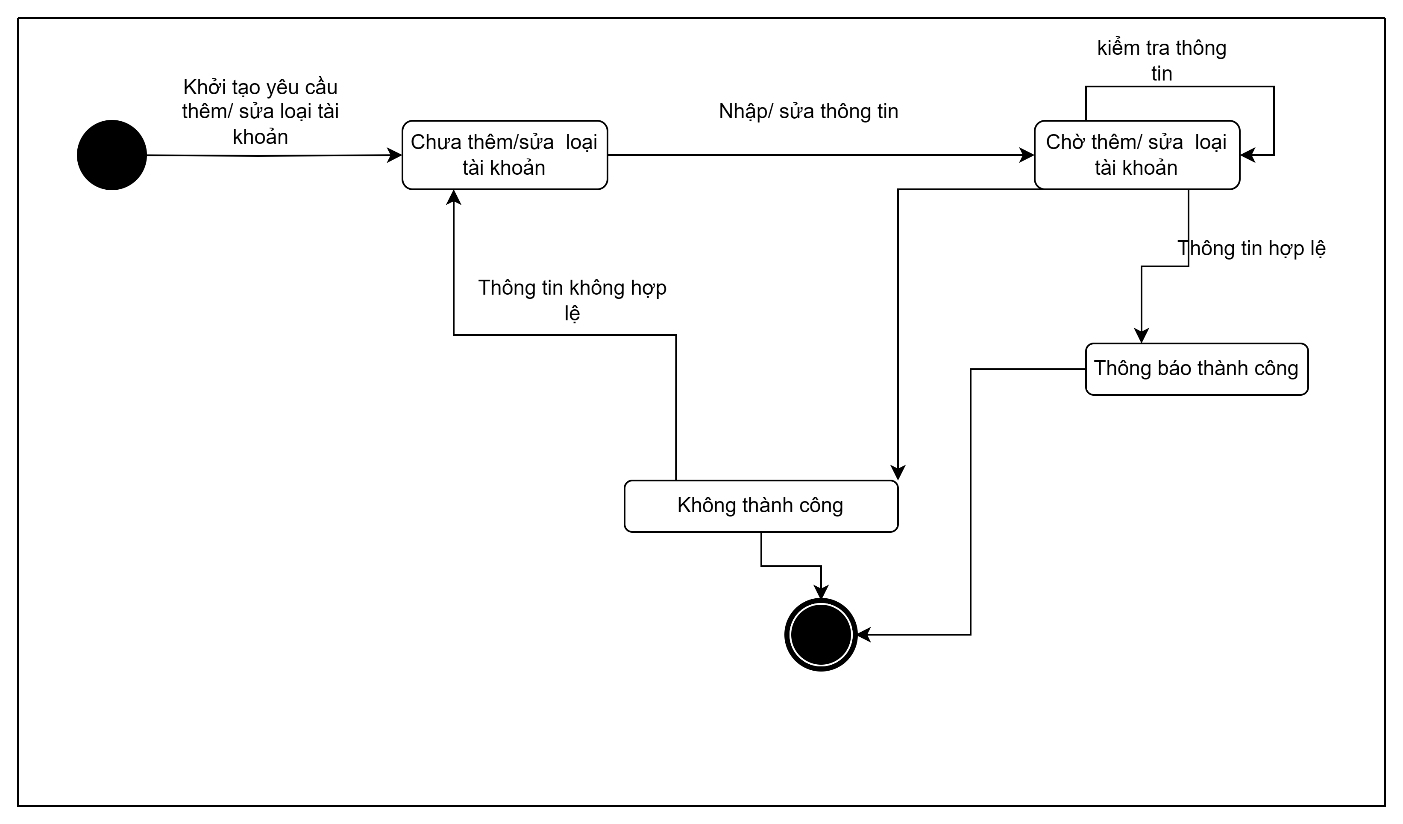


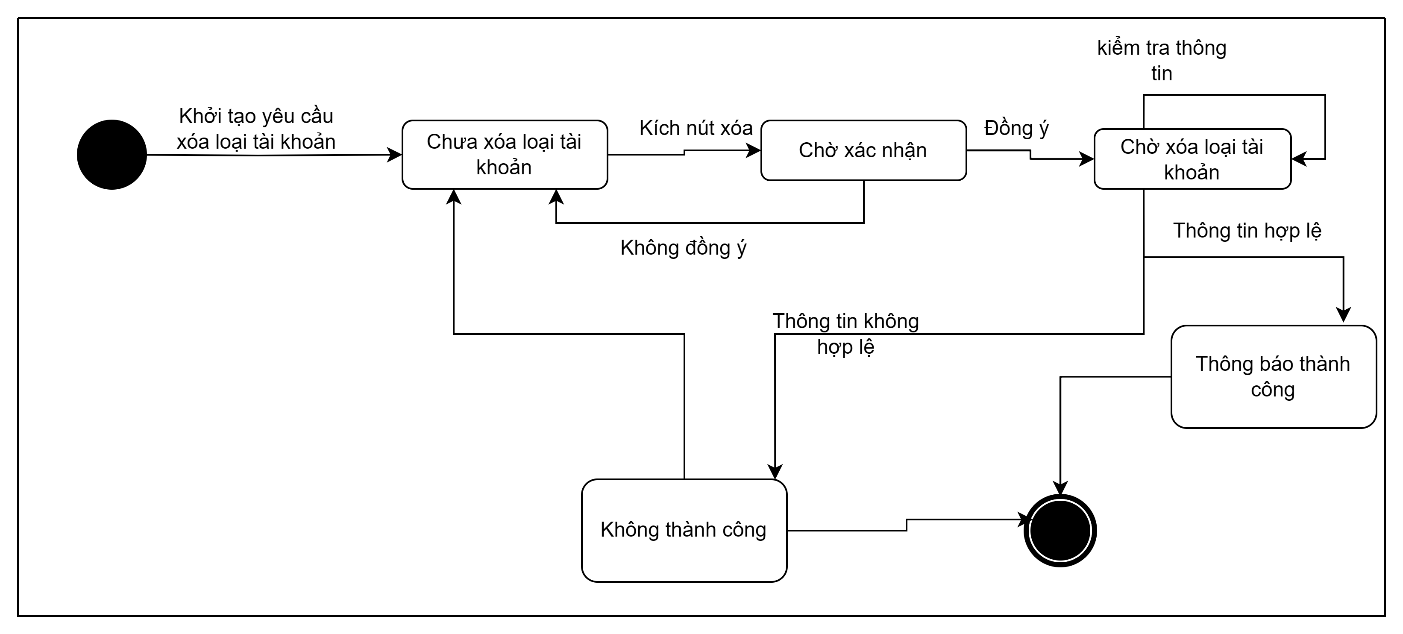
### 3.6.7 Usecase quản lý sự kiện



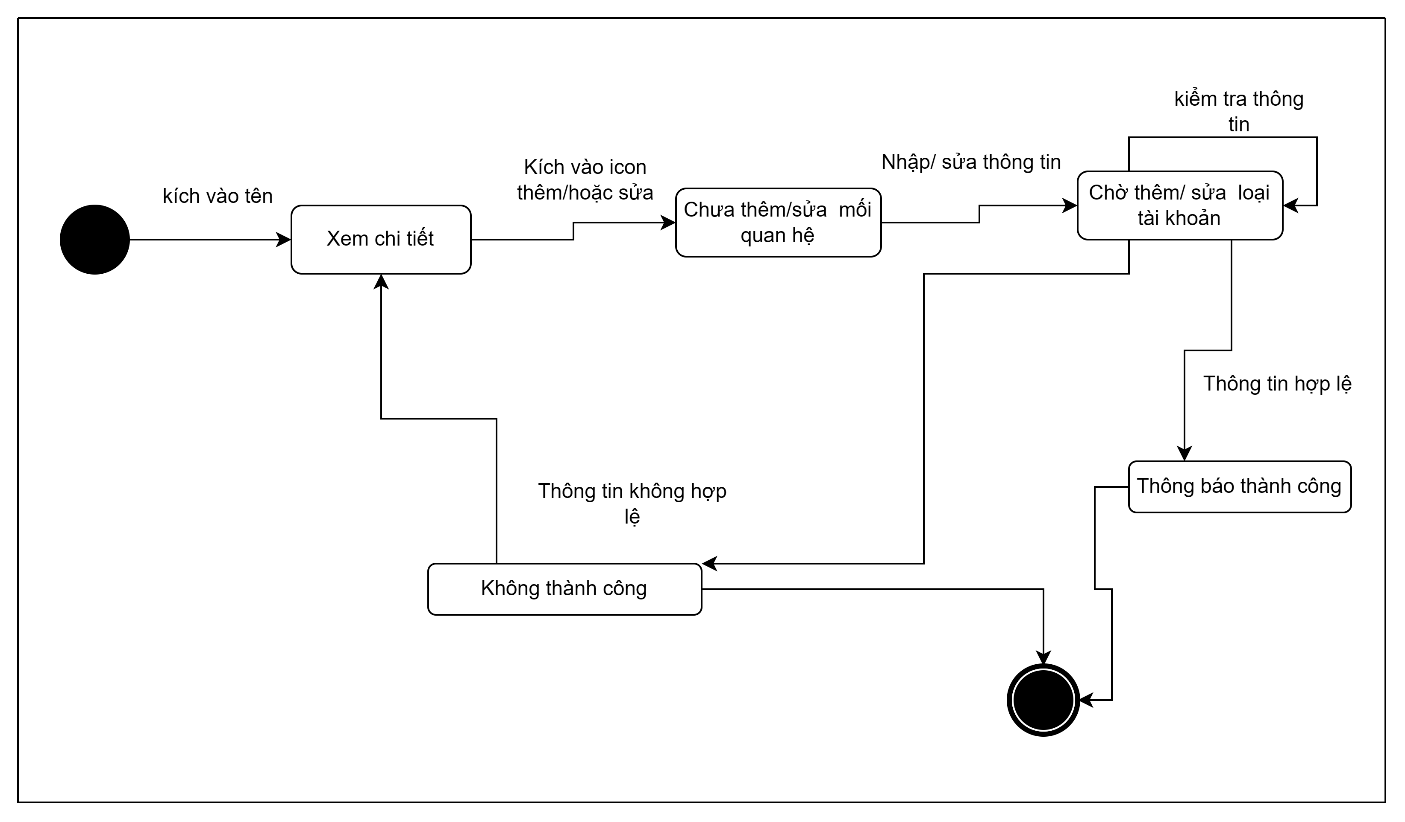


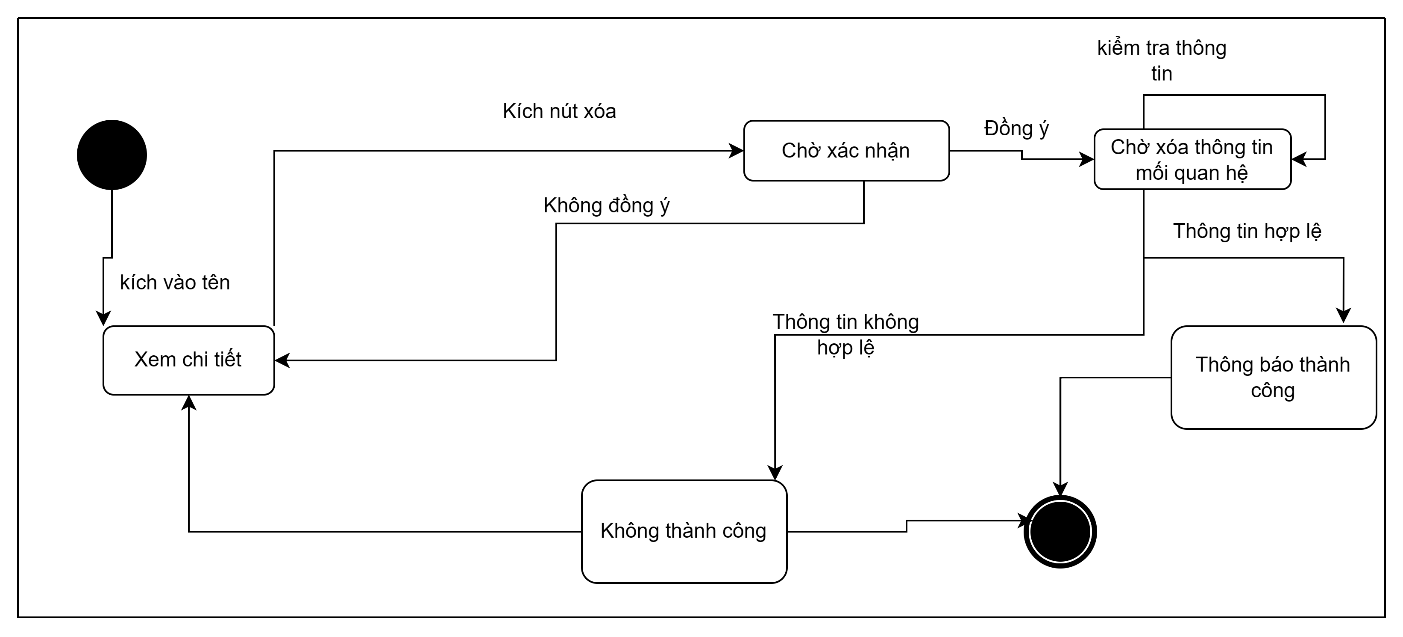
### 3.6.8 Usecase quản lý loại tài khoản





### 3.6.9 Quản lý mối quan hệ



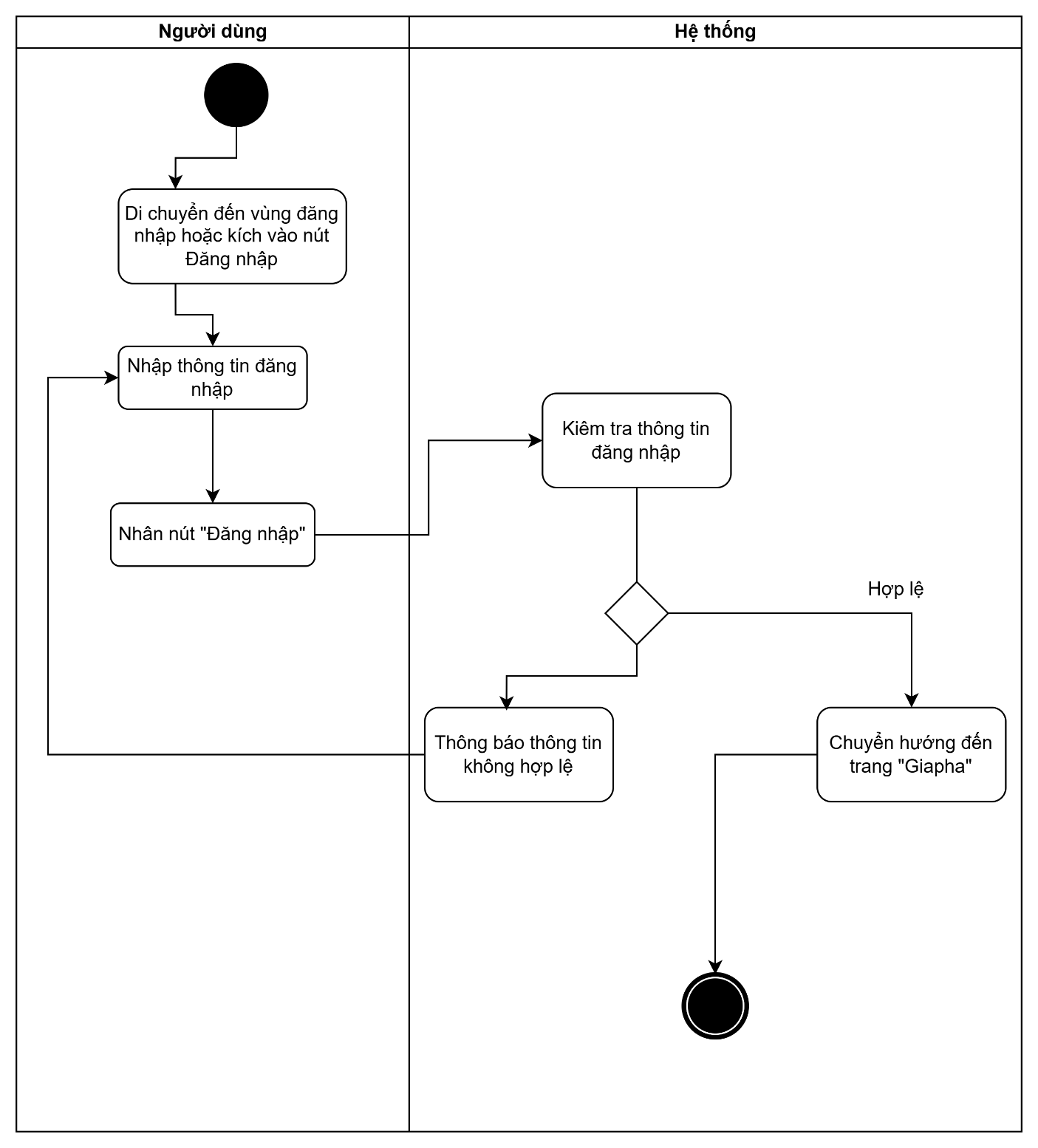


# Chương 3: Xây dựng và kiểm thử website

## Thiết kế các use case

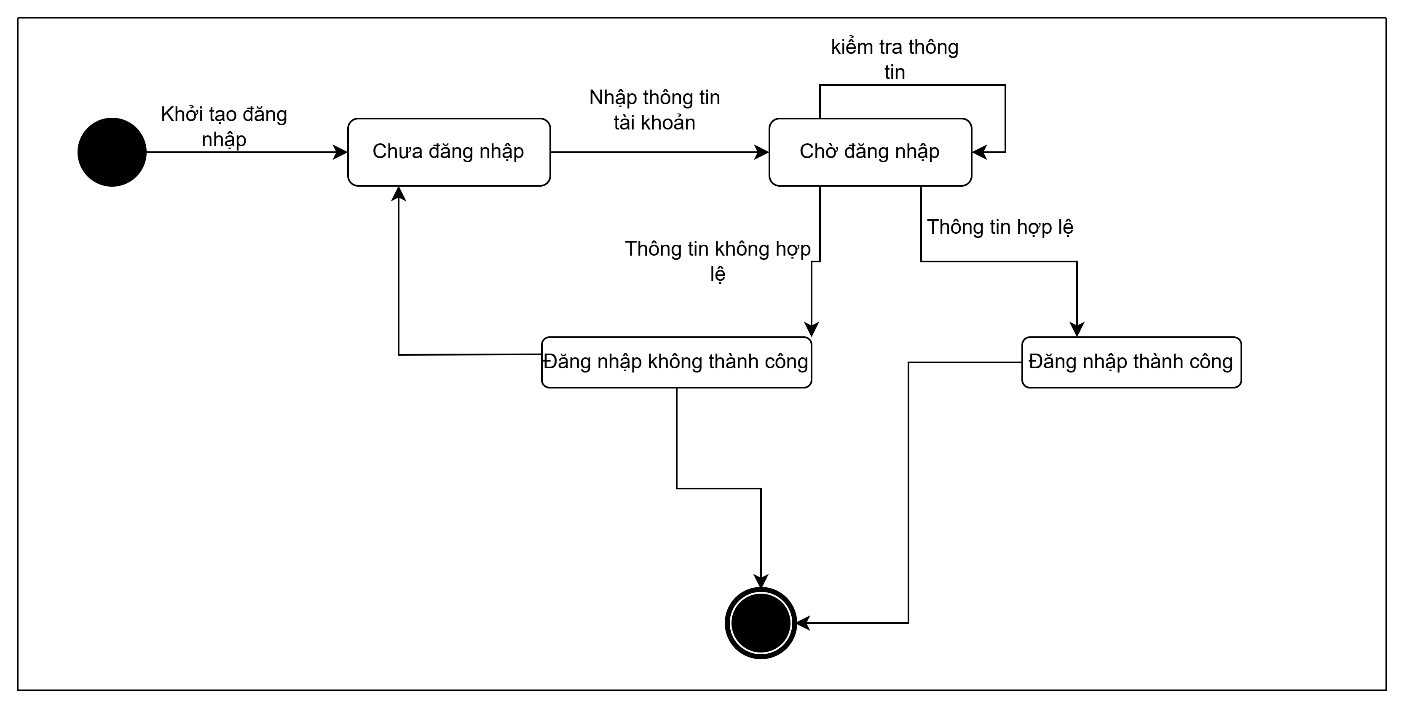
### Use case đăng nh*ập*

#### Biểu đồ hoạt động



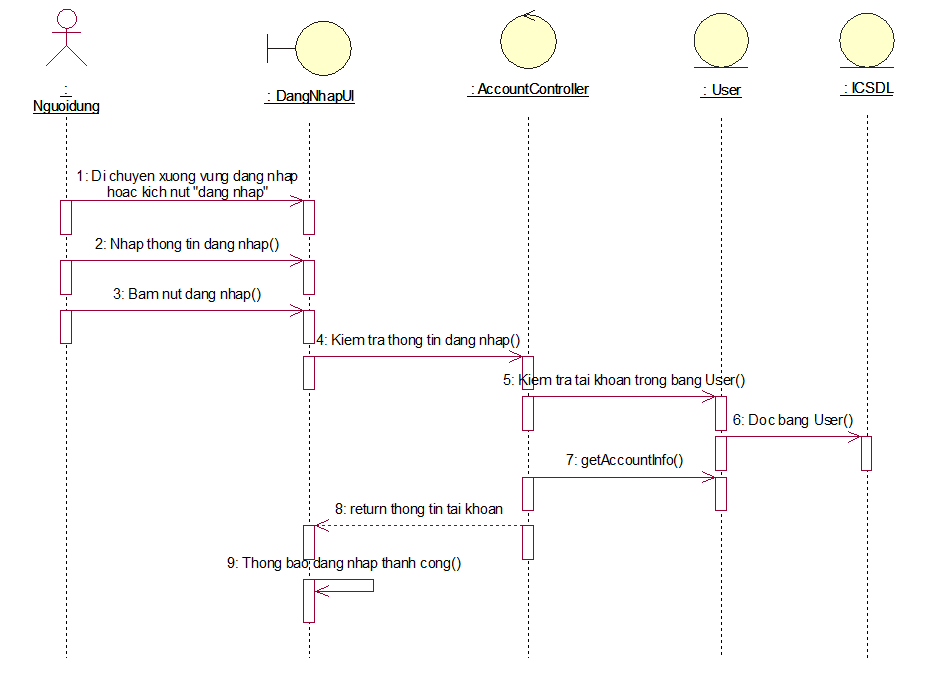
Hinh 3.1.1.1 Biểu đồ hoạt động usecase đăng nhập

#### Biểu đồ trạng thái



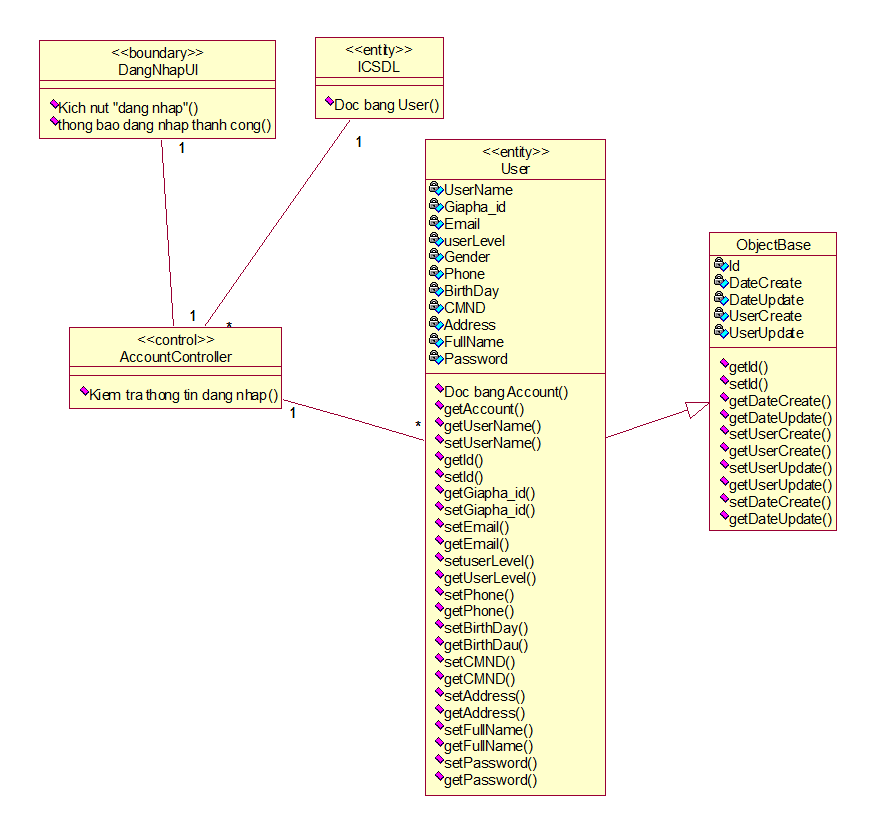
Hinh 3.1.1.2 Biểu đồ trạng thái usecasse đăng nhập

#### Biểu đồ tuần tự



Hinh 3.1.1.3 Biểu đồ tuần tự use case đăng nhập

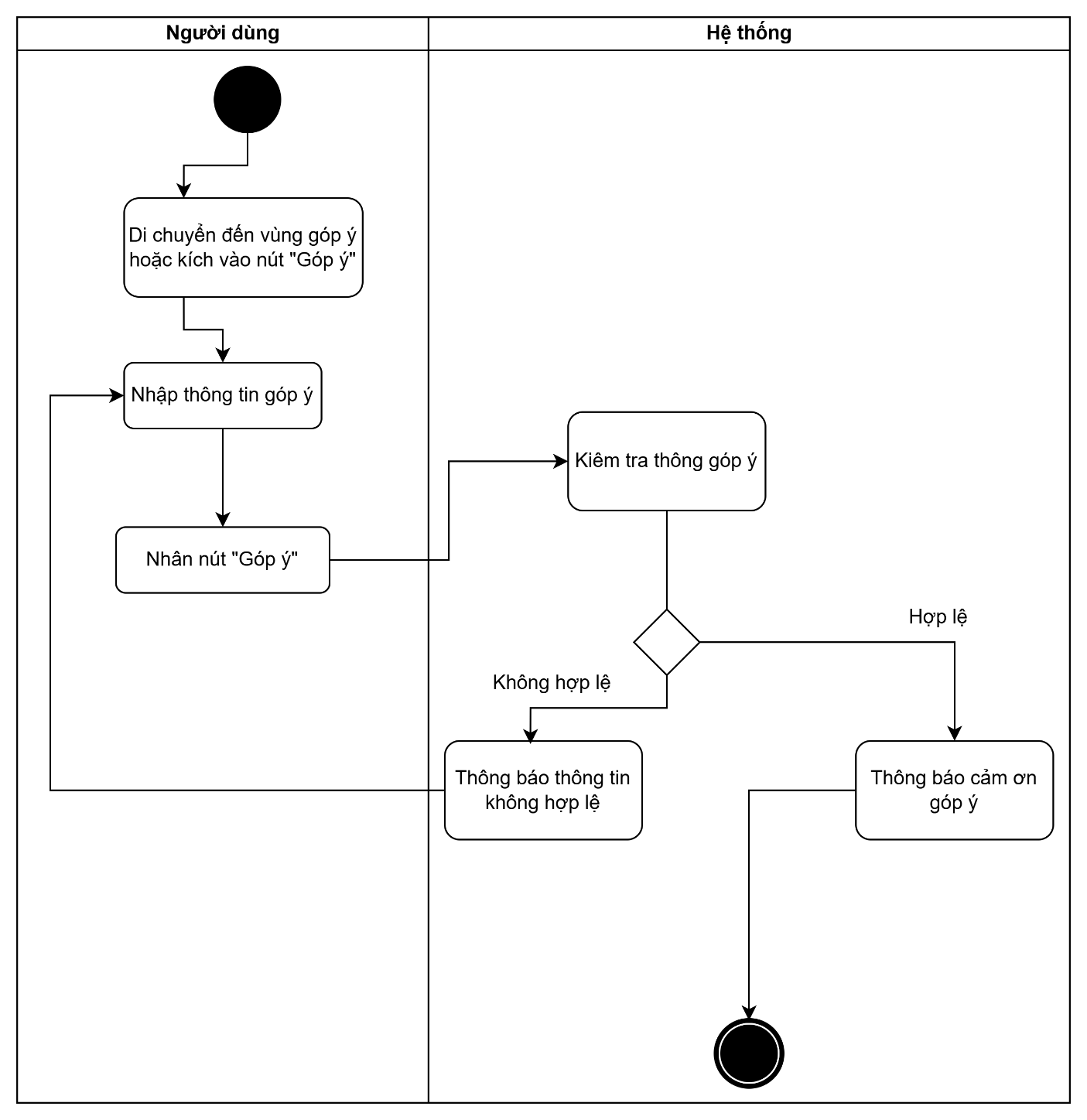
#### 3.1.1.4 Biểu đồ lớp



Hinh 3.1.1.4 Biểu đồ lớp use case đăng nhập

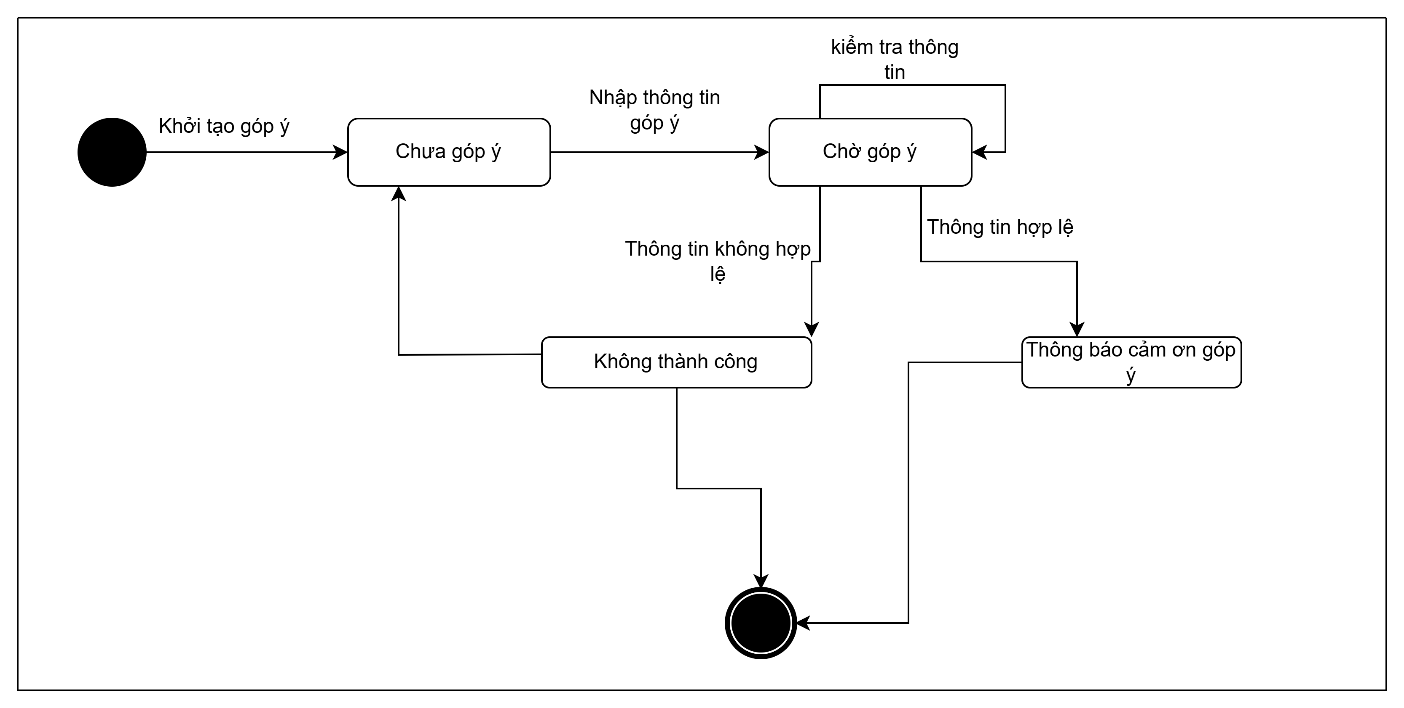
### Use case góp ý

#### Biểu đồ hoạt động



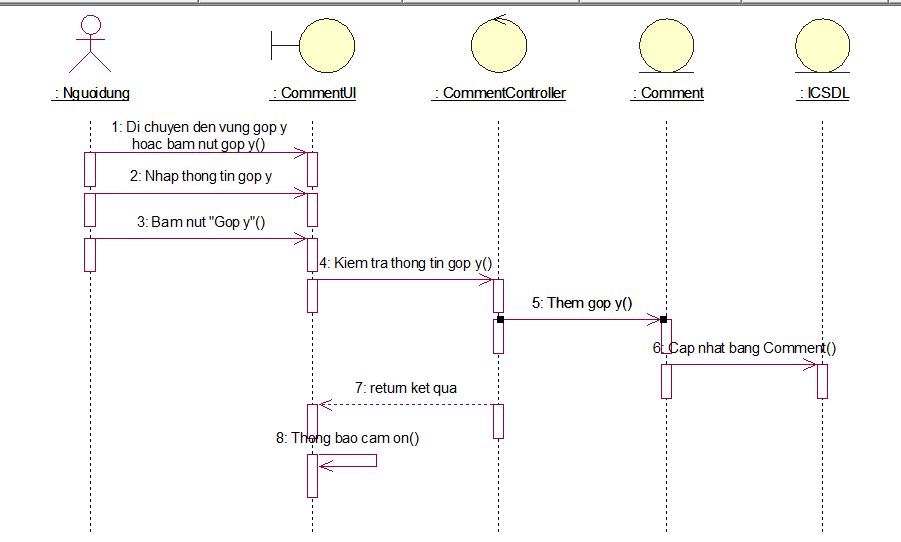
Hinh 3.1.2.1 Biểu đồ hoạt động use case góp ý

#### Biểu đồ trạng thái



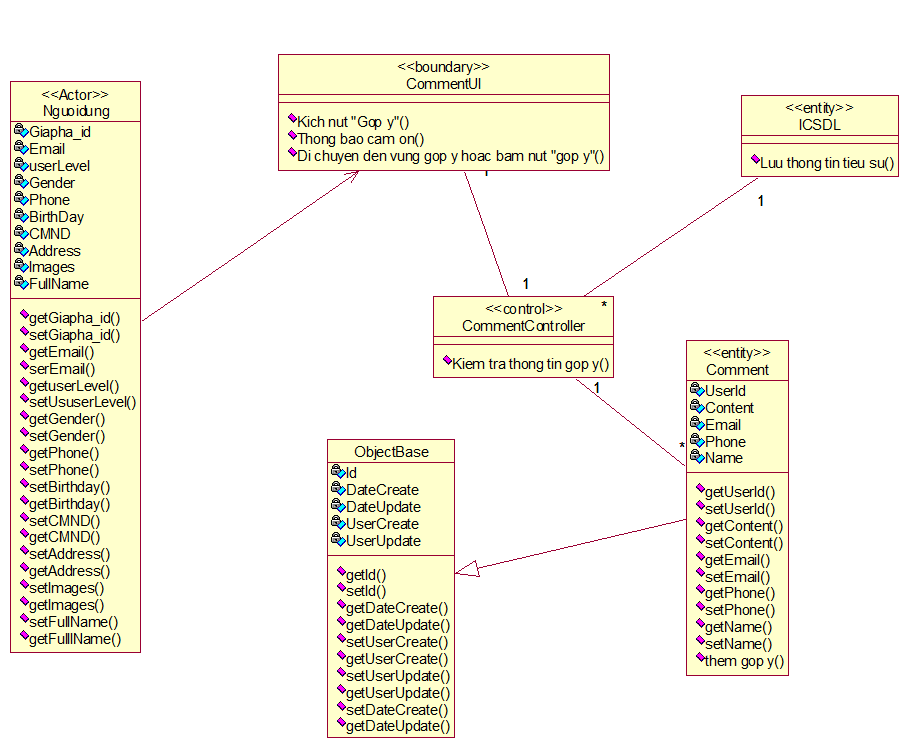
Hinh 3.1.2.2 Biểu đồ trạng thái use case góp ý

#### Biểu đồ tuần tự



Hinh 3.1.2.3 Biều đồ tuần tự usecase góp y

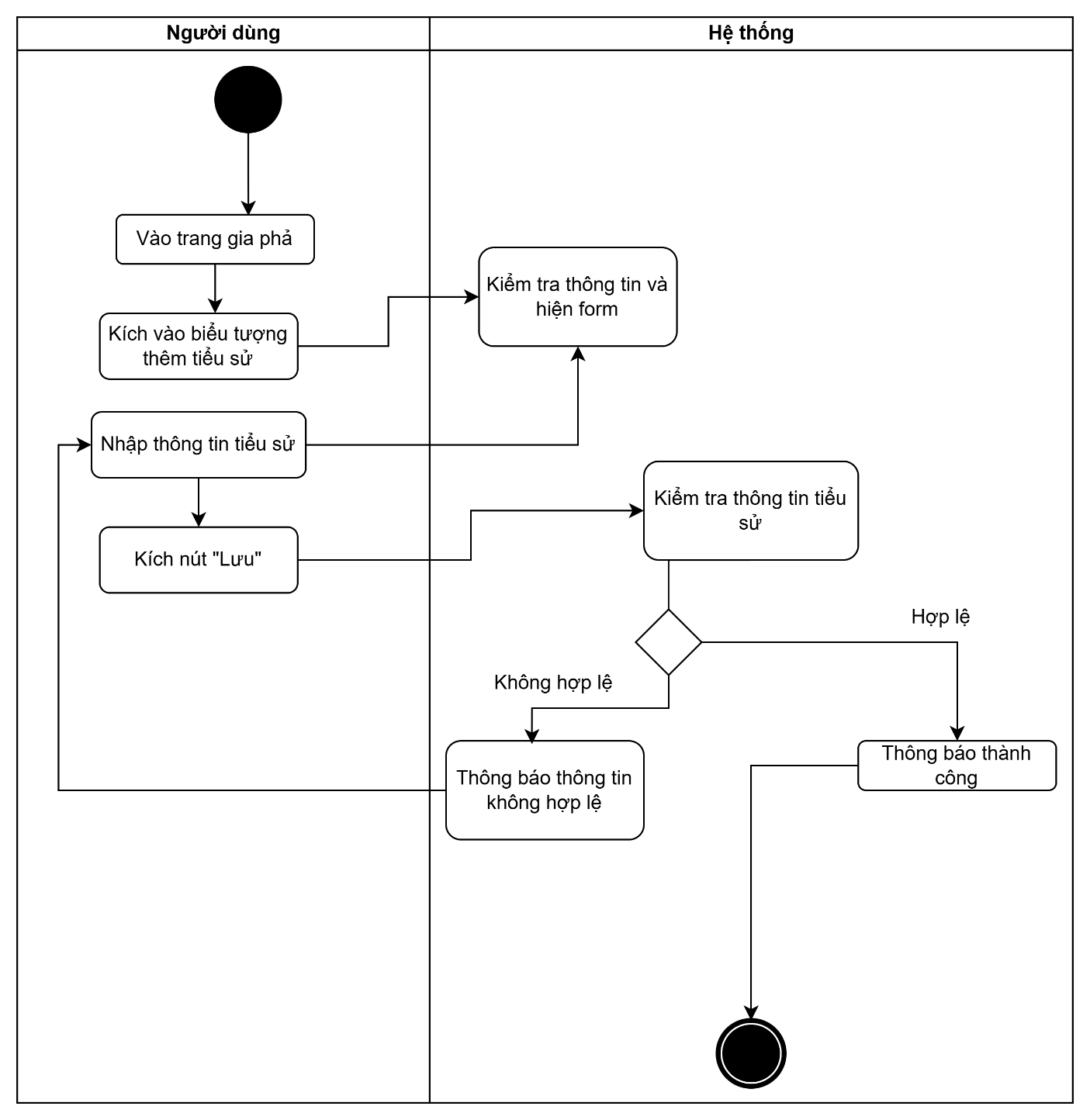
#### Biểu đồ lớp



Hinh 3.1.2.4 Biểu đồ lớp use case góp ý

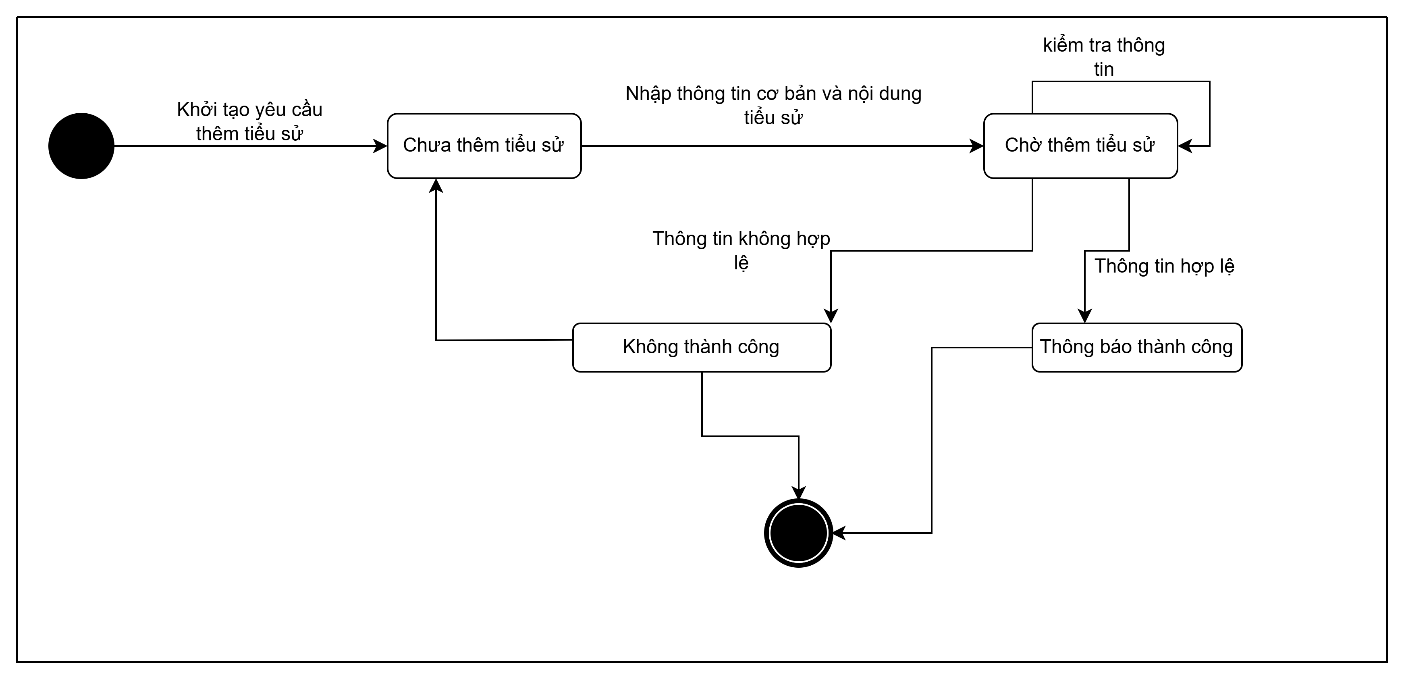
### Use case thêm tiểu sử

#### Biểu đồ hoạt động



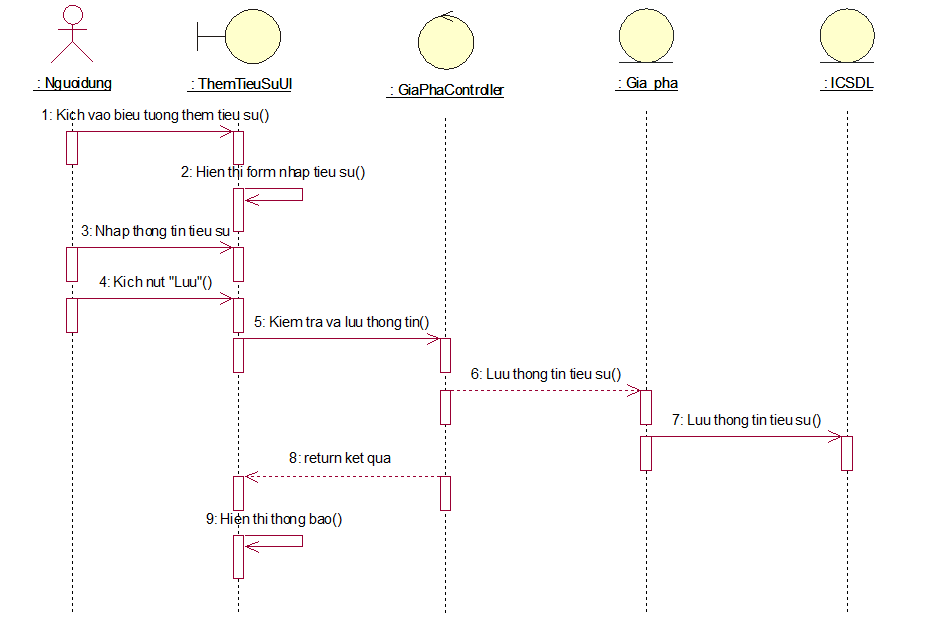
Hinh 3.1.3.1 Biểu đồ hoạt động usecase thêm tiểu sử

#### Biểu đồ trạng thái



Hinh 3.1.3.2 Biểu đồ trạng thái use case thêm tiểu sử

#### Biểu đồ tuần tự



Hinh 3.1.3.3 Biểu đồ tuần tự use case thêm tiểu sử

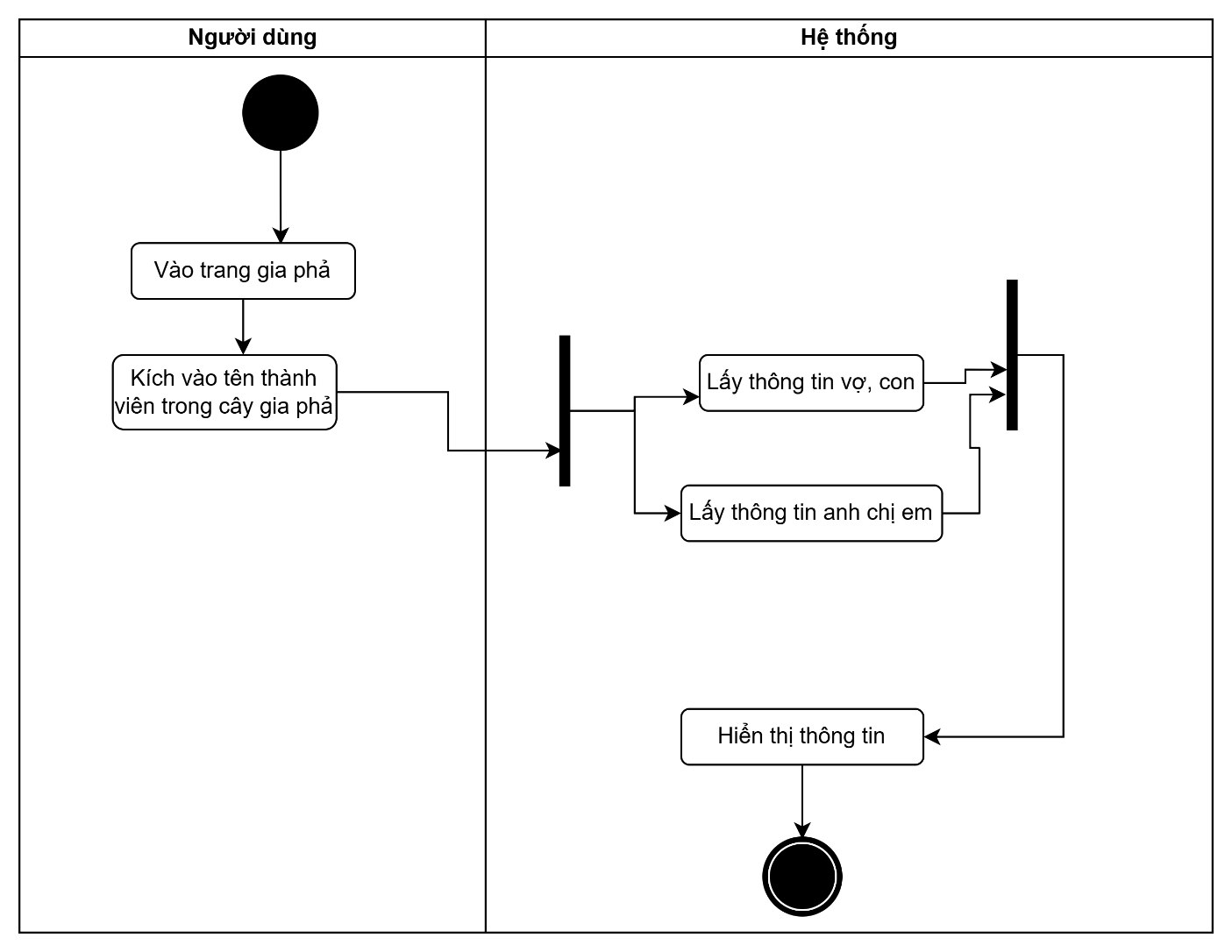
#### 3.1.3.4 Biểu đồ lớp



Hinh 3.1.3.4 Biểu đồ lớp use case thêm tiểu sử

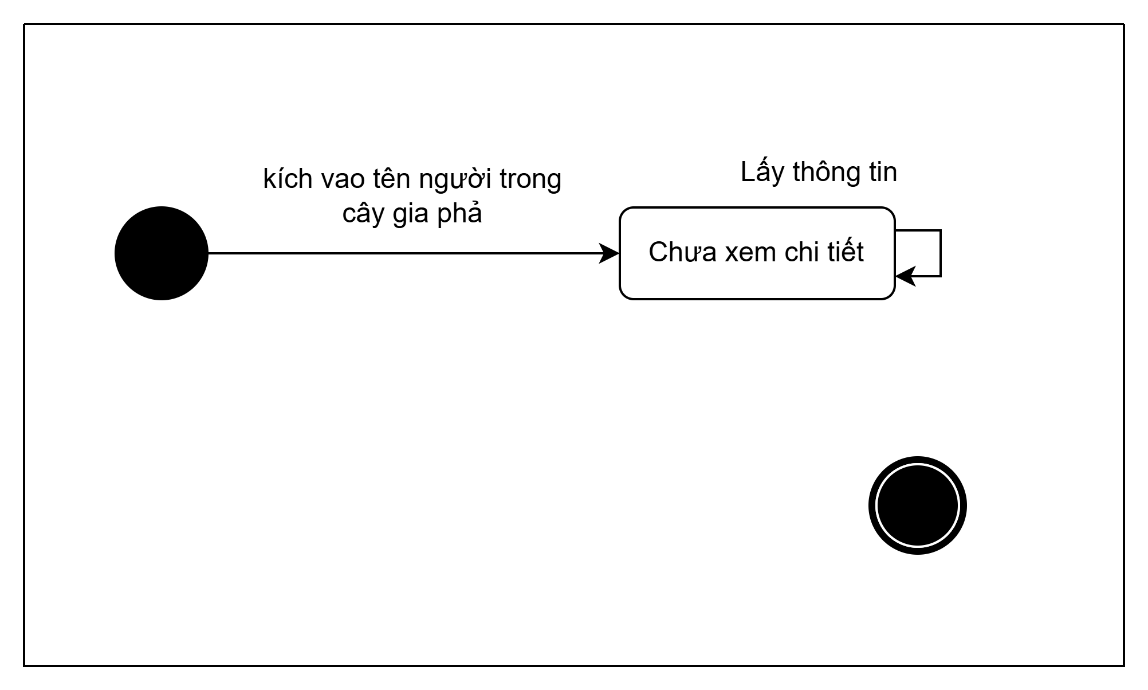
### Use case xem chi tiết mối quan hệ

#### Biểu đồ hoạt động



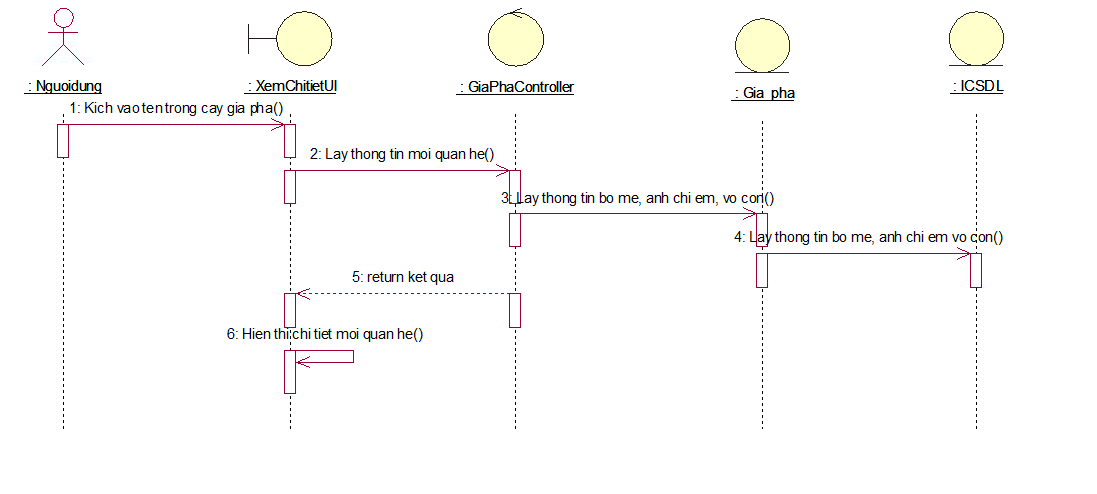
Hinh 3.1.4.1 Biểu đồ hoạt động use case xem chi tiết mối quan hệ

#### Biểu đồ trạng thái



Hinh 3.1.4.2 Biểu đồ trạng thái use case xem chi tiết mối quan hệ

#### Biểu đồ tuần tự



Hinh 3.1.4.3 Biểu đồ tuần tự use case xem chi tiết mối quan hệ

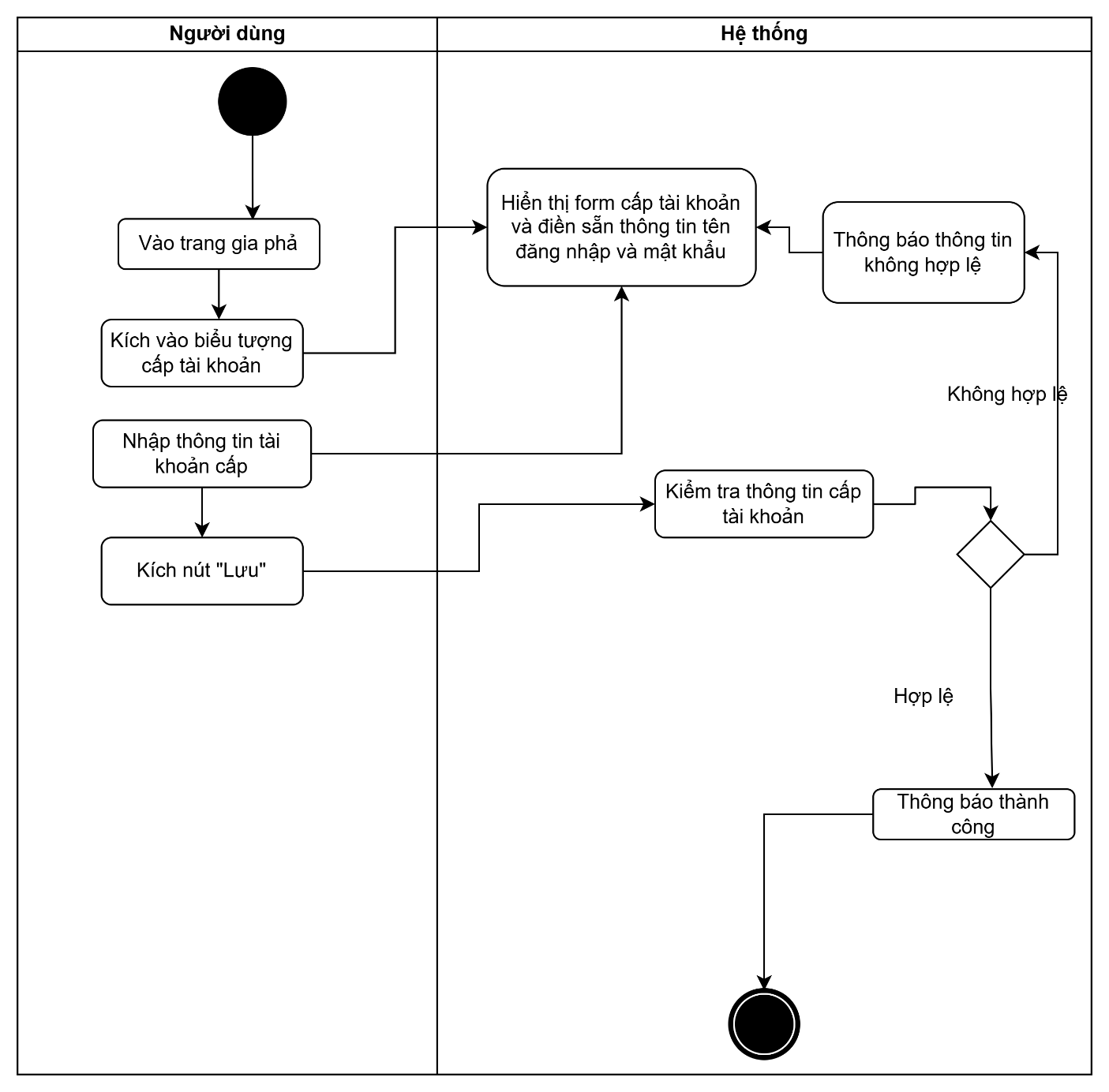
#### Biểu đồ lớp



Hinh 3.1.4.4 Biểu đồ lớp usecase xem chi tiết mối quan hệ

### *3.1.5* Use case cấp tài khoản

#### 3.1.5.1 Biểu đồ hoạt động



Hinh 3.1.5.1 Biểu đồ hoạt động use case cấp tài khoản

#### 3.1.5.2 Biểu đồ trạng thái

#### 3.1.5.3 Biểu đồ tuần tự

#### 3.1.5.4 Biểu đồ lớp

### 3.1.6 Use case quản lý tài khoản

#### 3.1.6.1 Biểu đồ hoạt động

#### 3.1.6.2 Biểu đồ trạng thái

#### 3.1.6.3 Biểu đồ tuần tự

#### 3.1.6.4 Biểu đồ lớp

### *3.1.7* Use case quản lý sự kiện

#### 3.1.7.1 Biểu đồ hoạt động

#### 3.1.7.2 Biểu đồ trạng thái

#### 3.1.7.3 Biểu đồ tuần tự

#### 3.1.7.4 Biểu đồ lớp

### *3.1.8* Use case quản lý loại tài khoản

#### 3.1.8.1 Biểu đồ hoạt động

#### 3.1.8.2 Biểu đồ trạng thái

#### 3.1.8.3 Biểu đồ tuần tự

#### 3.1.8.4 Biểu đồ lớp

### *3.1.9* Use case quản lý mối quan hệ

#### 3.1.9.1 Biểu đồ hoạt động

#### 3.1.9.2 Biểu đồ trạng thái

#### 3.1.9.3 Biểu đồ tuần tự

#### 3.1.9.4 Biểu đồ lớp

### *3.1.10* Use case tìm kiếm

#### 3.1.10.1 Biểu đồ hoạt động

#### 3.1.10.2 Biểu đồ trạng thái

#### 3.1.10.3 Biểu đồ tuần tự

#### 3.1.10.4 Biểu đồ lớp

## 3.2 Xây dựng giao diện

## 3.2 Kiểm thử website

# Chương 4: Kết luận

## 4.1 Kết quả đạt được

## 4.2 Hạn chế

## 4.3 Hướng phát triển