**MỤC LỤC**

[Chương 1: Tổng quan về mô hình MVC 3](#_Toc54034459)

[1. Mô hình MVC 3](#_Toc54034460)

[1.1 Tổng quan về kiến trúc MVC 3](#_Toc54034461)

[1.2 Mô hình MVC và các thành phần bên trong của MVC 3](#_Toc54034462)

[1.3 Luồng đi trong mô hình MVC như thế nào? 6](#_Toc54034463)

[1.4 Ưu/nhược điểm của kiến trúc MVC 6](#_Toc54034464)

[2. Mô hình MVP 8](#_Toc54034465)

[3. So sánh mô hình MVP và MVC 8](#_Toc54034466)

[Chương 2: Tài liệu kiến trúc phần mềm 10](#_Toc54034467)

[1. Giới thiệu 12](#_Toc54034468)

[1.1. Mục đích 12](#_Toc54034469)

[1.2. Phạm vi 12](#_Toc54034470)

[1.3. Định nghĩa, từ viết tắt và chữ viết tắt 12](#_Toc54034471)

[1.4. Tài liệu tham khảo 13](#_Toc54034472)

[1.5. Tổng quát 13](#_Toc54034473)

[2. Các mục tiêu và ràng buộc kiến trúc 13](#_Toc54034474)

[3. Tổng quan về kiến trúc 14](#_Toc54034475)

[3.1. Khung nhìn kiến trúc 14](#_Toc54034476)

[3.2. Kiểu kiến trúc 14](#_Toc54034477)

[3.3. Tiến trình kiến trúc 15](#_Toc54034478)

[4. Các khung nhìn kiến trúc 16](#_Toc54034479)

[4.1. Khung nhìn Use-case 16](#_Toc54034480)

[4.2. Khung nhìn logic 16](#_Toc54034481)

[4.3. Khung nhìn tiến trình 17](#_Toc54034482)

[4.4. Khung nhìn phát triển 25](#_Toc54034483)

[4.5. Khung nhìn vật lý 26](#_Toc54034484)

[5. Kích thước và hiệu suất 26](#_Toc54034485)

[6. Chất lượng 26](#_Toc54034486)

[Chương 3: Demo chương trình 28](#_Toc54034487)

[1. Cơ sở dữ liệu. 28](#_Toc54034488)

[2. Giao diện đăng nhập. 28](#_Toc54034489)

[3. Giao diện đăng ký. 28](#_Toc54034490)

[4. Giao diện trang chủ. 29](#_Toc54034491)

[5. Giao diện sửa thông tin tài khoản. 29](#_Toc54034492)

[6. Giao diện thêm tài khoản mạng xã hội. 30](#_Toc54034493)

[6.1. Mã hóa bằng MD5. 32](#_Toc54034494)

[6.2. Mã hóa bằng MD5 + Salt. 32](#_Toc54034495)

[6.3. Mã hóa bằng Bcrypt. 32](#_Toc54034496)

[7. Giao diện sửa tài khoản mạng xã hội. 32](#_Toc54034497)

[8. Giao diện xóa tài khoản mạng xã hội. 33](#_Toc54034498)

[9. Giao diện chi tiết tài khoản mạng xã hội. 34](#_Toc54034499)

[10. Giao diện chứng thực mật khẩu. 35](#_Toc54034500)

Chương 1: Tổng quan về mô hình MVC

## 1. Mô hình MVC

### 1.1 Tổng quan về kiến trúc MVC

Mô hình kiến ​​trúc MVC đã tồn tại từ lâu trong kỹ thuật phần mềm. Hầu hết các nhà phát triển Web đều quen thuộc với kiến ​​trúc MVC vì nó được áp dụng rộng rãi cho thiết kế ứng dụng tương tác trên trang web máy chủ như mua sắm trực tuyến, khảo sát, đăng ký sinh viên và nhiều hệ thống dịch vụ tương tác khác. Kiến trúc MVC được sử dụng đặc biệt trong các ứng dụng mà giao diện người dùng dễ bị thay đổi dữ liệu.

Kiến trúc này tách phần trình bày và tương tác khỏi dữ liệu hệ thống. Kiến trúc này lần đầu tiên được giới thiệu trong Smalltalk-80. Theo Glenn Krasner và Stephen Pope (1988), hệ thống được cấu trúc thành ba thành phần logic Model, View và Controller tương tác với nhau.

Model: cấu phần mô hình quản trị dữ liệu hệ thống (bao gồm cả các thao tác trên dữ liệu)

View (hiển thị): xác định và quản lý cách dữ liệu được trình bày cho người dùng. Thành phần này đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, button, images….

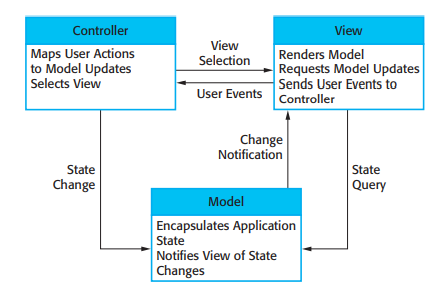
Ví dụ: View trong ASP.NET MVC là HTML, CSS và một số cú pháp đặc biệt (cú pháp Razor) giúp dễ dàng giao tiếp với mô hình và bộ điều khiển.

Controller - Bộ điều khiển: quản lý tương tác của người dùng. Thông thường, Controller nhận yêu cầu sau đó chuyển những tương tác này đến View hoặc Model.

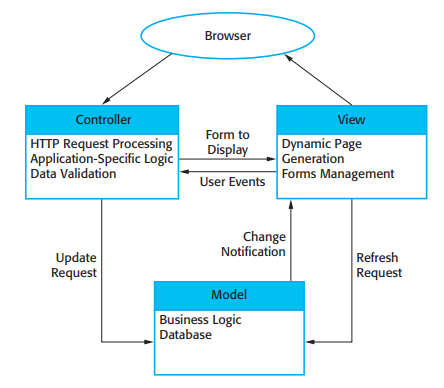
Các mô hình đồ họa của kiến trúc liên kết với mẫu MVC được thể hiện trong Hình 1.1 và 1.2. Chúng trình bày kiến trúc từ các khung nhìn khác nhau — Hình 4.1 là một khung nhìn khái niệm và Hình 1.2 cho thấy một kiến trúc thời gian chạy có thể có khi mô hình này được sử dụng để quản lý tương tác trong một hệ thống dựa trên web.

### 1.2 Mô hình MVC và các thành phần bên trong của MVC

Hình 1.1 mô tả mẫu Model-View-Controller nổi tiếng. Mô hình này là cơ sở của quản lý tương tác trong nhiều hệ thống dựa trên web.

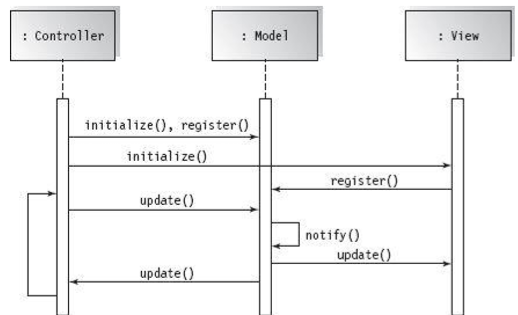


*Hình 1.1. Tổ chức của mô hình MVC*



*Hình 1.2. Kiến trúc ứng dụng Web sử dụng mẫu kiến trúc MVC*

Hình 1.3 mô tả một biểu đồ trình tự cho một kiến trúc MVC chung. Sau khi máy khách khởi động ứng dụng MVC, Controller khởi tạo Model và View, đồng thời gắn chính nó và View vào Model (điều này được gọi là đăng ký với Model). Sau đó, Contrroller chặn một yêu cầu của người dùng trực tiếp từ một dòng lệnh hoặc thông qua giao diện View và chuyển tiếp yêu cầu tới Model để cập nhật dữ liệu trong Model. Các thay đổi trong Model sẽ kích hoạt Model thông báo cho tất cả người nghe được đính kèm hoặc đã đăng ký về tất cả các thay đổi và các giao diện trong View cũng sẽ được cập nhật ngay lập tức

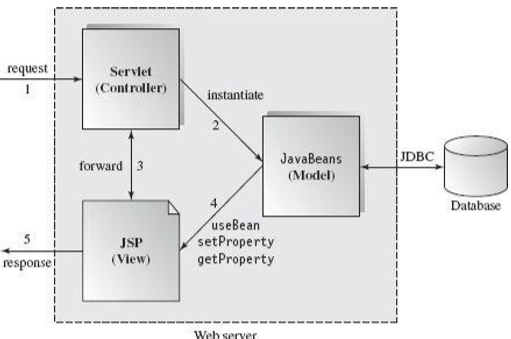


*Hình 1.3. Sơ đồ trình tự cho kiến trúc MVC*

Sơ đồ trong Hình 1.4. cho thấy một sơ đồ kiến trúc khối MVC điển hình trong công nghệ Java. JavaServer Pages (JSP) được sử dụng trong phát triển mô-đun View; Java Servlet được sử dụng trong việc triển khai mô-đun Điều khiển (Controller); và Java Bean, Java Enterprise Bean (EJB) và Java Data Base Connectivity (JDBC) được sử dụng trong phát triển mô-đun Dữ liệu (Model). Tương tự như công nghệ Java, công nghệ Microsoft ASP .NET được sử dụng để phát triển giao diện (View) và ADO .NET để phát triển mô hình. Bộ điều khiển nhận yêu cầu từ người dùng thông qua GUI hoặc giao diện dòng lệnh và khởi tạo các phiên bản tương ứng trong Dữ liệu, chọn các View liên quan để hiển thị dữ liệu, gọi các chức năng nghiệp vụ của Model và chuyển tiếp điều khiển đến View. View lấy dữ liệu từ Mô hình và hiển thị dữ liệu trong giao diện GUI.

Ví dụ sau minh họa một cách triển khai đơn giản của kiến trúc MVC trong đó chỉ có một lớp Java trong mỗi mô-đun trong ba mô-đun của kiến trúc MVC. Lớp MyBean JavaBean đóng vai trò của Mô hình dữ liệu; MyServlet Lớp Servlet đóng vai trò của Controller; và from-Servlet JSP đóng vai trò của View hiển thị.

Hình 1.4. cho thấy sơ đồ kiến trúc của ứng dụng web này. MyServlet Servlet đặt một giá trị mục và lưu trữ mục này trong một JavaBean có tên là myBean. Sau đó, nó chuyển quyền điều khiển đến một trang JSP có tên fromServlet.jsp, trang này lấy mục từ myBean và hiển thị nó trên một trang web.



*Hình 1.4. Kiến trúc MVC trên nền tảng Java Web*

Các miền ứng dụng của kiến trúc MVC: được sử dụng khi có nhiều cách để view và tương tác với dữ liệu. Cũng được sử dụng khi các yêu cầu tương tác và trình bày dữ liệu trong tương lại không xác định.

### **1.3 Luồng đi trong mô hình MVC như thế nào?**

**Bạn có thể hình dung, khi  một yêu cầu từ máy client gửi tới server, Controller sẽ thực hiện nhiệm vụ của mình đó là tiếp nhận và xử lý yêu cầu. Trong trường hợp cần thiết, nó có thể liên hệ Model – bộ phận làm việc với database để hỗ trợ.**

**Khi xử lý xong yêu cầu, kết quả sẽ được trả về View. Tại View sẽ tạo mã HTML thành giao diện và trả về hiển thị trên trình duyệt.**

### 1.4 Ưu/nhược điểm của kiến trúc MVC

Ưu điểm

* Cho phép dữ liệu thay đổi độc lập với biểu diễn của nó và ngược lại.
* Hỗ trợ trình bày cùng một dữ liệu theo những cách khác nhau với những thay đổi được thực hiện trong một biểu diễn được hiển thị trong tất cả chúng.
* Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế.
* Loose coupling: (gắn kết lỏng lẻo): bản chất của MVC framework là có sự ghép nối thấp giữa các models, views, controllers
* Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp và bảo trì, …
* Nhiều bộ công cụ Framework của nhà cung cấp MVC có sẵn
* Rất hiệu cho sự phát triển nếu các chuyên gia phát triển đồ họa, lập trình và cơ sở dữ liệu đang làm việc trong một nhóm trong một dự án được thiết kế.

Nhược điểm

* Có thể liên quan đến chứa các mã trình bổ xung và mã trình phức tạp ngay cả trong trường hợp mô hình dữ liệu và các tương tác đơn giản.
* Đối với những dự án nhỏ việc áp dụng mô hình MVC gây cồng kềnh, tốn thời gian trong quá trình phát triển
* Tốn thời thời gian trung chuyển dữ liệu của các thành phần.
* Không phù hợp với các ứng dụng hướng về tác nhân như các ứng dụng di động và robot tương tác.
* Nhiều cặp bộ controller và view dựa trên cùng một model làm cho bất kỳ thay đổi mô hình dữ liệu nào trở nên đắt đỏ.
* Code navigability (khả năng điều hướng mã nguồn): các framework điều hướng có thể phức tạp vì nó giới thiệu các gián tiếp mới và yêu cầu người dùng phải thích ứng với các tiêu chí phân rã của MVC.
* Multi-artifact consistency: phân tách một tính năng thành 3 thành phần gây ra sự tán xạ. Do đó, yêu cầu các nhà phát triển phải duy trì một tính nhất quán giữa các thành phần.

2. Mô hình MVP

**Cấu trúc mô hình MVP**

**Tầng trình diễn - Presenter**

**Tầng trình diễn có trách nhiệm như một middie-man giữa View và Model. Nó lấy dữ liệu từ Model, định dạng và trả về cho View. Nhưng không giống như mô hình MVC, nó quyết định những gì sẽ xảy ra khi người dùng tương tác với View, hay nói cách khác nó hàm chứa logic ứng dụng**

**Tầng logic dữ liệu ­ - Model**

**Model là tầng xử lí dữ liệu. Lớp này sẽ chịu trách nhiệm lấy dữ liệu từ database hoặc network một cách bất đồng bộ. Sau đó sẽ trả về dữ liệu cho Presenter. Trong một ứng dụng với thiết kế kiến trúc tốt, mô hình nãy sẽ chỉ là một gateway giữa tầng domain và tầng busines logic. Trong mô hình Clean Architecture của Uncle Bob, Model sẽ là một interactor thực thi một use case. Để đơn giản, ở đây Model đơn thuần được nhìn nhận như một data source – cung cấp dữ liệu cho chúng ta muốn hiển thị trong giao diện ứng dụng**

**Tầng giao diện – View**

**Lớp này chịu trách nhiệm tìm View (bind View), đưa dữ liệu vào view, animation, kiểm soát các input event của user và gửi cho present các event. View thường được implement bởi một Activity (hoặc có thể là một Fragment, một View … tùy thuộc vào cấu trúc ứng dụng), Activity này sẽ chứa một thuộc tính là một lớp Presenter. Lý tưởng nhất Presenter nên được cung cấp bởi một Dependency Injection framework như Dagger, nhưng trong trường hợp ứng dụng không sử dụng một thư viện hay framework như vậy ta hoàn toàn có thể tạo ra các đối tượng Presenter này**

**3. So sánh mô hình MVP và MVC**

**Điểm khác biệt cơ bản của 2 mô hình MVC và MVP là sự khác nhau về vài trò của Presenter (MVP) và Controller (MVC), nó cũng dẫn đến sự khác nhau về vai trò của view. Trong MVP, sự hiện diện của Controller đã bị loại bỏ,thay vào đó,việc xử lý các dữ liệu input được view đảm nhận và chuyển cho Presenter khi có yêu cầu tương tác đến Model.**

**Các mô hình MVP được dựa trên mô hình MVC. Các Presenter (MVP) và Controller (MVC) có vai trò gần tương tự nhau. Chúng có trách nhiệm giao tiếp giữa Model và View. Điều đó nói rằng,Controller (MVC) không quản lý Model va View như Presenter(MVP).**

**Trong mô hình MVC ,tầng View có phần thông minh và có thể lấy dữ liệu trực tiếp từ Model. Trong các mô hình MVP,View là hoàn toàn thụ động và lấy dữ liệu luôn được phân phối đến các View theo Presenter. Controller trong MVC cũng có thể được chia sẻ giữa nhiều view.Ngược lại trong MVP,View và Presenter có mỗi quan hệ một một,do đó một Presenter được gắn vơi 1 View.**

**Những khác biệt về cách hoạt động làm mô hình MVP đảm bảo tính tách biệt giữa tầng dữ liệu và tầng hiển thị, và nó cũng làm tăng đáng kể khả năng kiểm thử của ứng dụng từ việc tách biệt đó**

Chương 2: Tài liệu kiến trúc phần mềm

Trang web lưu trữ mật khẩu tài khoản mạng xã hội

Nguyễn Viết Kỳ, Lê Thanh Tùng, Bùi Văn Tiến

Version 1.0

17/10/2020

Version <1.0>

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| <17/10/20> | <1.0> | Bản chính thức tài liệu kiến trúc phần mềm |
|  |  |  |
|  |  |  |
|  |  |  |

## 1. Giới thiệu

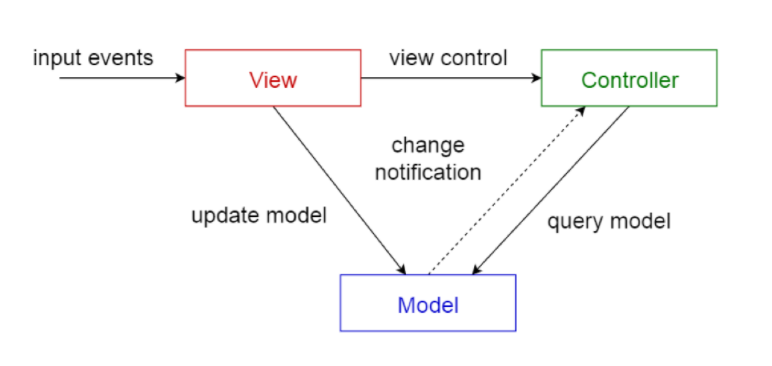
Phần giới thiệu này cung cấp cái nhìn tổng quan về toàn bộ *Tài liệu Kiến trúc Phần mềm* cho Trang web lưu trữ mật khẩu tài khoản mạng xã hội. Tài liệu bao gồm mục đích, phạm vi, định nghĩa, từ viết tắt, chữ viết tắt, tài liệu tham khảo và tổng quan về trang web.

* 1. Mục đích

Tài liệu này cung cấp kiến trúc tổng quan của Trang web lưu trữ mật khẩu tài khoản mạng xã hội (SNPKW - Social Networks Password Keep Website).

Mục đích chính của SNPKW là để đáp ứng nhu cầu lưu trữ mật khẩu an toàn của người dùng.

Tài liệu này nhằm nắm bắt và truyền đạt các quyết định kiến trúc đã được thực hiện trong việc thiết kế và xây dựng hệ thống. Dưới đây mà là mô hình khái quát của hệ thống:



### Phạm vi

Phạm vi của tài liệu này là giải thích kiến trúc của Trang web lưu trữ mật khẩu tài khoản mạng xã hội.

* 1. Định nghĩa, từ viết tắt và chữ viết tắt
* SNPKW – Trang web lưu trữ tài khoản mạng xã hội
* ASP.NET MVC 5 - Nền tảng Microsoft web theo mô hình MVC version 5
* SQL Server 2017 – Hệ quản trị cơ sở dữ liệu Microsoft 2017
* SAD – Tài liệu kiến trúc phần mềm
* UML – Ngôn ngữ mô hình hóa thống nhất
* User - Bất kỳ người dùng nào đăng ký tài khoản trên SNPKW

### Tài liệu tham khảo

[Adam Freeman]: Pro ASP.NET MVC 5, <https://books.google.com.vn/books/about/Pro_ASP_NET_MVC_5.html?id=ZqZ4AgAAQBAJ&redir_esc=y>

[Kruchten]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf>

* 1. Tổng quát

Tài liệu bao gồm 7 phần, bao gồm:

* Phần 1 giới thiệu về kiến trúc phần mềm SNPKW
* Phần 2 giải quyết các mục tiêu và ràng buộc của kiến trúc hệ thống
* Phần 3 mô tả biểu diễn của hệ thống
* Phần 4 mô tả 5 khung nhìn kiến trúc
* Phần 5 của tài liệu nói về những cân nhắc khác của hệ thống như kích thước và hiệu suất của hệ thống
* Phần 6 mô tả một số vấn đề về chất lượng hệ thống.
* Phần 7 là danh mục tài liệu tham khảo được sử dụng để tạo tài liệu này

1. Các mục tiêu và ràng buộc kiến trúc

Kiến trúc SNPKW đã được thiết kế với các mục tiêu sau:

1. Để tạo điều kiện thuận lợi cho quá trình bảo mật cũng như mã hóa tài khoản của khách hàng trên mạng xã hội
2. Giúp những người thiết kế cũng như lập trình giải thích về quản lý bảo mật, mã hóa tài khoản cá nhân..
3. Cho phép người dùng nâng cao tính cá nhân hóa, bảo vệ thông tin cá nhân bằng việc đưa cho họ một quyết định hợp lí dựa trên quy tắc bảo mật

## 3. Tổng quan về kiến trúc

### 3.1. Khung nhìn kiến trúc

Mô hình hóa, triển khai và lập hồ sơ một hệ thống yêu cầu hệ thống phải được nhìn nhận từ các khía cạnh khác nhau. Do đó, kiến trúc của SNPKW sẽ được biểu diễn theo cách tiếp cận 5 dạng xem: Use Case View, Logical View, Process View, Physical View, và Deployment View. Dưới đây là mô tả ngắn gọn cho từng chế độ xem:

Logical View

Process View

Physical View

Deployment View

Use Case View: mục đích chính của dạng xem ca sử dụng là xác định các trình điều khiển của hệ thống, các yêu cầu hệ thống

Logical View: dạng xem này chứa bất kỳ định nghĩa hệ thống nào cũng như các biểu đồ lớp và đối tượng mô tả các dịch vụ mà hệ thống sẽ cũng cấp cho người dùng cuối của nó

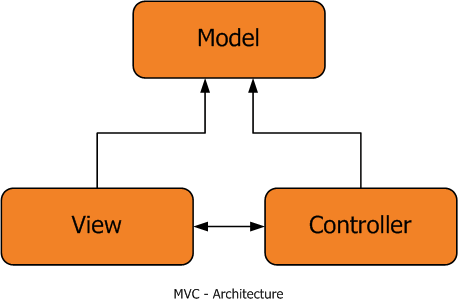
Process View: sẽ hiển thị các quy trình hình thành cơ chế của hệ thống. Chúng sẽ được biểu diễn dưới dạng sơ đồ cộng tác, trình tự và hoạt động

Component View: bao gồm các thông số kỹ thuật của hệ thống và giao diện người dùng, ý nghĩa, các thành phần khác nhau tạo nên hệ thống

Deployment View: miêu tả các nút phần cứng của các hệ thống khác nhau hoạt động cùng nhau như cách mỗi nút phần cứng sẽ được cài đặt và triển khai

### 3.2. Kiểu kiến trúc

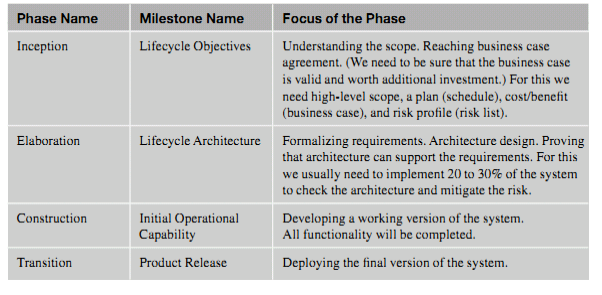
Mô hình có thể được sử dụng để đáp ứng bất kỳ nhu cầu chức năng, phi chức năng hoặc yêu cầu thẩm mỹ của hệ thống. SNPKW sử dụng mô hình kiến trúc MVC bảo gồm 3 thành phần: Model, View, Controller. Dưới đây là mô tả đơn giản về mỗi thành phần:



* Model: cấu phần mô hình quản trị dữ liệu hệ thống (bao gồm cả các thao tác trên dữ liệu)
* View (hiển thị): xác định và quản lý cách dữ liệu được trình bày cho người dùng. Thành phần này đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, button, images….
* Controller - Bộ điều khiển: quản lý tương tác của người dùng. Thông thường, Controller nhận yêu cầu sau đó chuyển những tương tác này đến View hoặc Model

### 3.3. Tiến trình kiến trúc

SNPKW tuân thủ theo quy trình hợp lý (RUP), với mục tiêu là cho phép sản xuất phần mềm chất lượng đáp ứng nhu cầu của người dùng cuối trong một lịch trình và ngân sách có thể dự đoán được. RUP là một quá trình lặp đi lặp lại được chia thành bốn giai đoạn: inception (thiết lập trường hợp kinh doanh của dự án), elaboration( thiết lập kế hoạch dự án và kiến trúc hệ thống), construction ( triển khai hệ thống) và transition ( triển khai hệ thống).



Mỗi lần lặp được lập kế hoạch riêng. Kế hoạch lặp được chuẩn bị khi bắt đầu mỗi lần lặp. Nó cung cấp một mô tả chỉ tiết về các hoạt động sẽ được thực hiện, định nghĩa các công nhân, và xác định các thành phẩm sẽ được tạo. Mỗi lần lặp tạo một số phát hành có thể kiểm thử, mà tiến dần đến sản phẩm cuối cùng. Các thành phẩm có thể theo dạng như sau:

* Phần mềm làm việc
* Các mô hình (Mô hình UC, mô hình Đối tượng, ...), thường được mô tả bởi UML
* Các tài liệu (Tài liệu yêu cầu stakeholder, tài liệu trực quan, các tài liệu UC, ...).

Mỗi đánh giá lần lặp có thể được tạo khi kết thúc mỗi lần lặp để phân tích liệu các mục tiêu có thỏa mãn không. Sau đây là sáu thực hành tốt nhất hình thành nền tảng cho RUP:

* Phát triển phần mềm lặp
* Quản lý các yêu cầu
* Sử dụng các kiến trúc dựa trên thành phần
* Mô hình phần mềm trực quan (với ULM)
* Thẩm định chất lượng liên tục (điều này gồm việc kiểm tra không chỉ sản phẩm cuối cùng mà còn kiểm tra chất lượng các yêu cầu, mã nguồn, các thiết kế, các kế hoạch dự án, và các thành phần khác của hoạt động phát triển hệ thống).
* Quản lý thay đổi (điều này bao gồm quản lý „hoạt động” và quản lý „cấu hình – quản lý tiến trình thay đổi và các thành phần bi thay đổi).

## 4. Các khung nhìn kiến trúc

### 4.1. Khung nhìn Use-case



* 1. Khung nhìn logic



*Biểu đồ lớp*

* 1. Khung nhìn tiến trình

Khung nhìn này mô tả sự phân rã của hệ thống cũng như cách thức giao tiếp giữa các quy trình, thông báo, hoạt động giữa các thành phần và trình tự thông báo.



*Biểu đồ trình tự chức năng đăng nhập*



*Biểu đồ trình tự chức năng đăng ký tài khoản*



*Biểu đồ trình tự chức năng đăng xuất*



*Biểu đồ trình tự thay đổi thông tin tài khoản người dùng*



*Biểu đồ trình tự chức năng tạo tài khoản mạng xã hội*



*Biểu đồ trình tự chức năng sửa tài khoản mạng xã hội*



*Biểu đồ trình tự chức năng xóa tài khoản mạng xã hội*

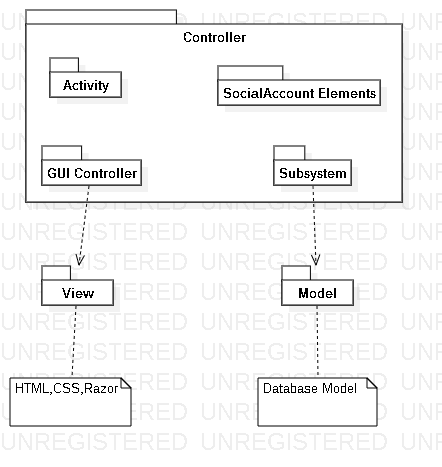


*Biểu đồ trình tự chức năng chi tiết tài khoản mạng xã hội*



*Biểu đồ trình tự chức năng xem mật khẩu tài khoản mạng xã hội*

* 1. Khung nhìn phát triển

*Biểu đồ gói*

### 4.5. Khung nhìn vật lý

*Biểu đồ triển khai hệ thống*

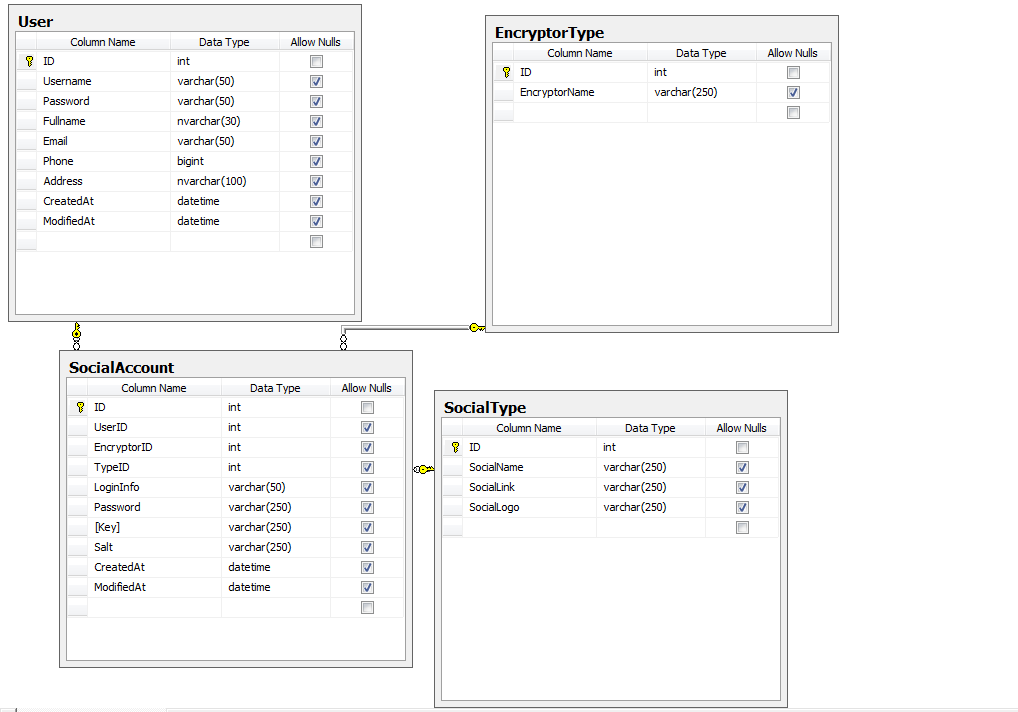
1. Kích thước và hiệu suất

* Số ca sử dụng đã thực hiện: 9
* Số lượng dòng mã C#:
* Thời gian thực hiện:

1. Chất lượng

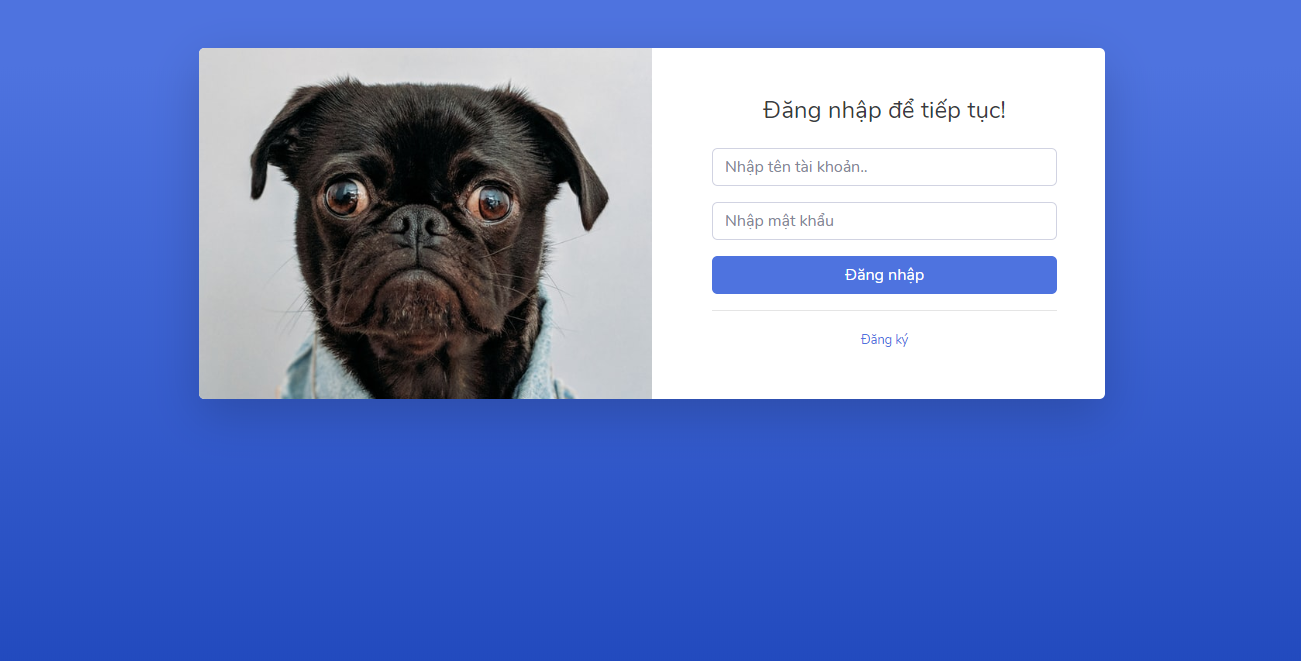
* Thách thức đối với mã hóa dữ liệu:
  + Hầu hết phương thức tấn công cơ bản vào mã hóa hiện nay là Brute Foce (thử và sai liên tục) và thử các khóa ngẫn nhiên cho đến khi khóa đúng được tìm thấy. Có thể giảm thiểu xác suất bằng cách tăng chiều dài, độ phức tạp của khóa. Mã hóa càng mạnh thì tài nguyên cần để thực hiện tính toán sẽ tăng lên, cần nhiều thời gian và vật lực hơn để phá mã
  + Các phương pháp phá vỡ mã khóa khác bao gồm tấn công kênh phụ và phân tích mã. Tấn công bên xảy ra sau khi việc mã hóa hoàn tất thay vì tấn công trực tiếp vào mã hóa. Những cuộc tấn công này có khả năng thành công nếu có lỗi trong thiết kế hệ thống hoặc thực thi
  + Nhìn chung, việc mã hóa dữ liệu là điều cần thiết để chúng ta có thể tăng sự bảo mật hơn cho tài liệu, đặc biệt là nhiễu kiểu tài liệu mật, thông tin tài khoản cá nhân.
* Ứng dụng
  + Có vai trò hết sức quan trọng, đặc biệt là trong bảo mật thông tin cá nhân. Có thể nói mã hóa chính là việc đảm bảo bí mật, toàn vẹn thông tin, khi thông tin được truyền trên mạng internet.

# **Chương 3: Demo chương trình**

1. Cơ sở dữ liệu.

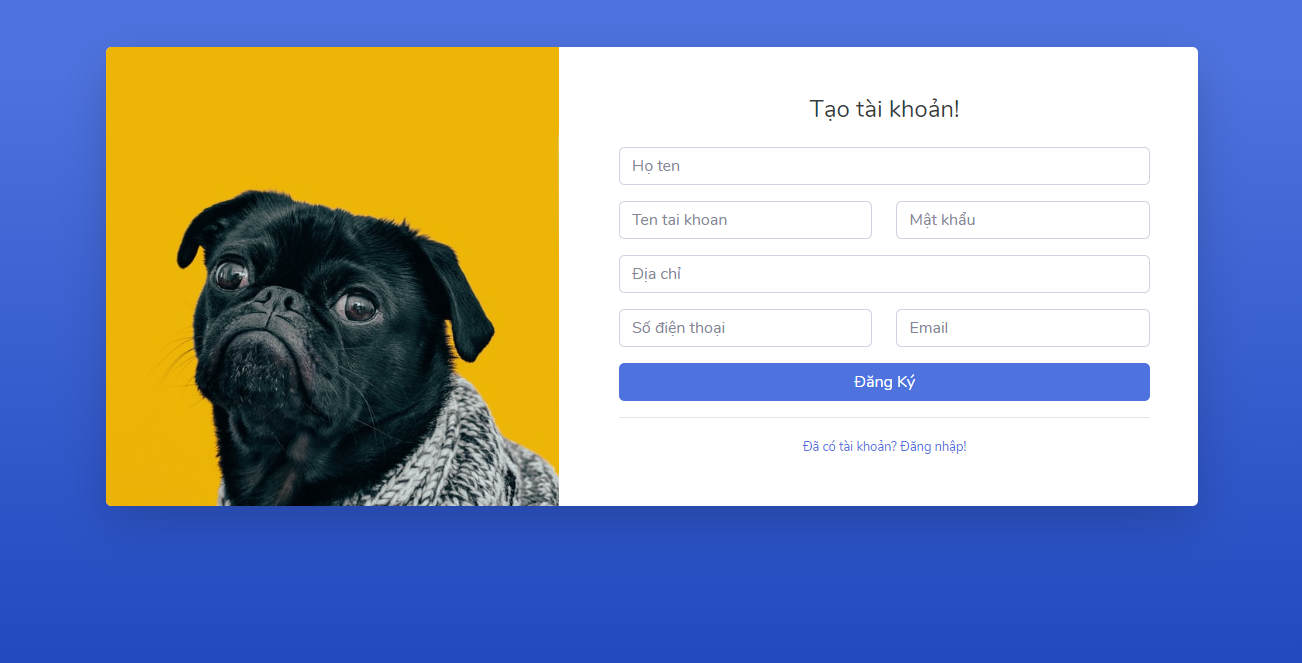
*Hình 1.1. Cơ sở dữ liệu SQL Server.*

1. Giao diện đăng nhập.

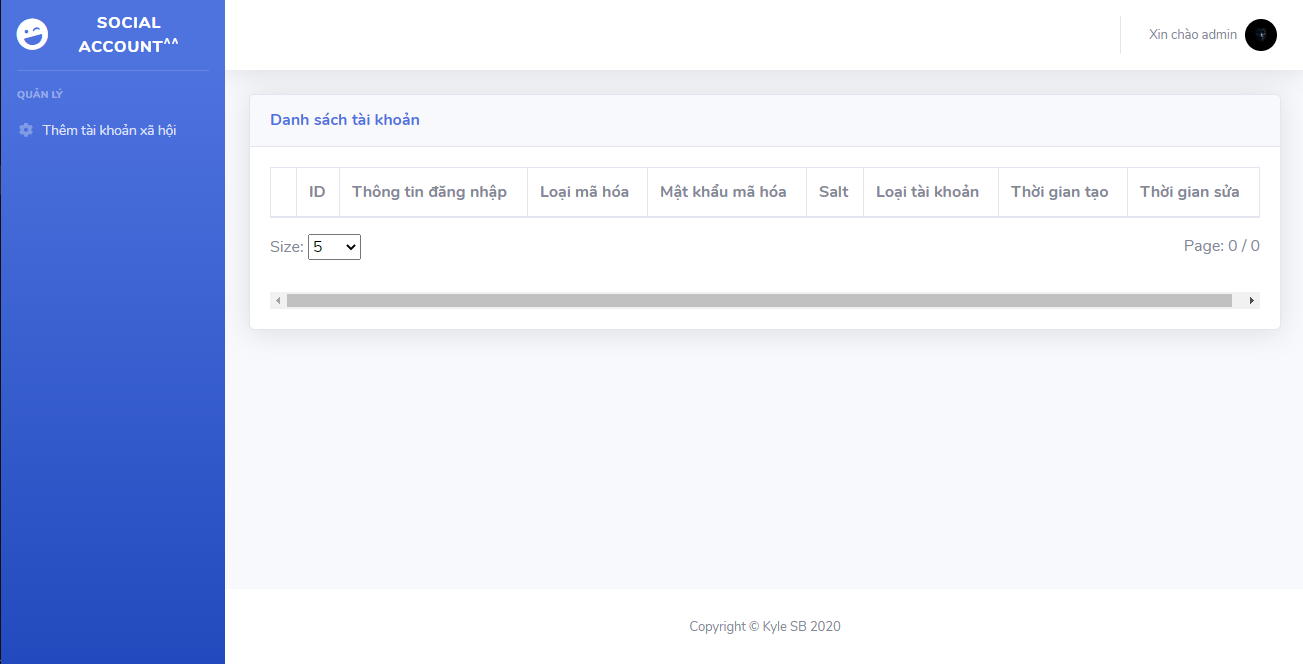


*Hình 2.1. Giao diện đăng nhập.*

1. Giao diện đăng ký.

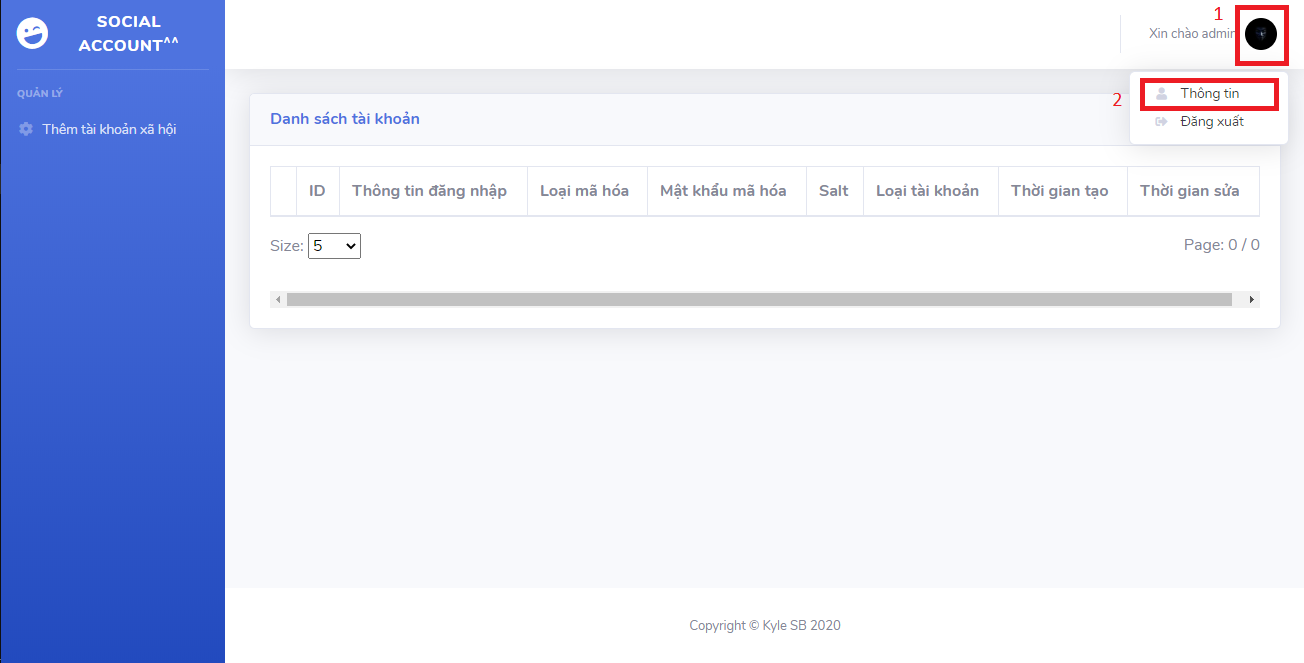


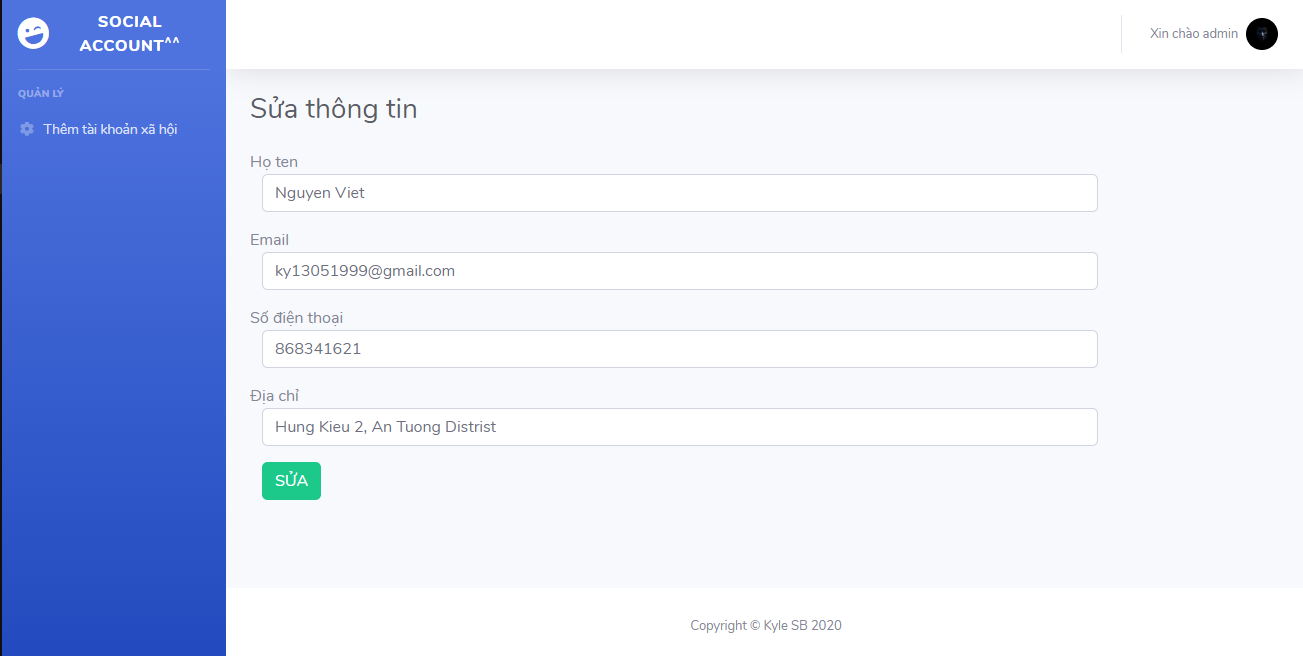
*Hình 3.1. Giao diện đăng ký.*

1. Giao diện trang chủ.

*Hình 4.1. Giao diện trang chủ chưa có tài khoản nào.*

1. Giao diện sửa thông tin tài khoản.

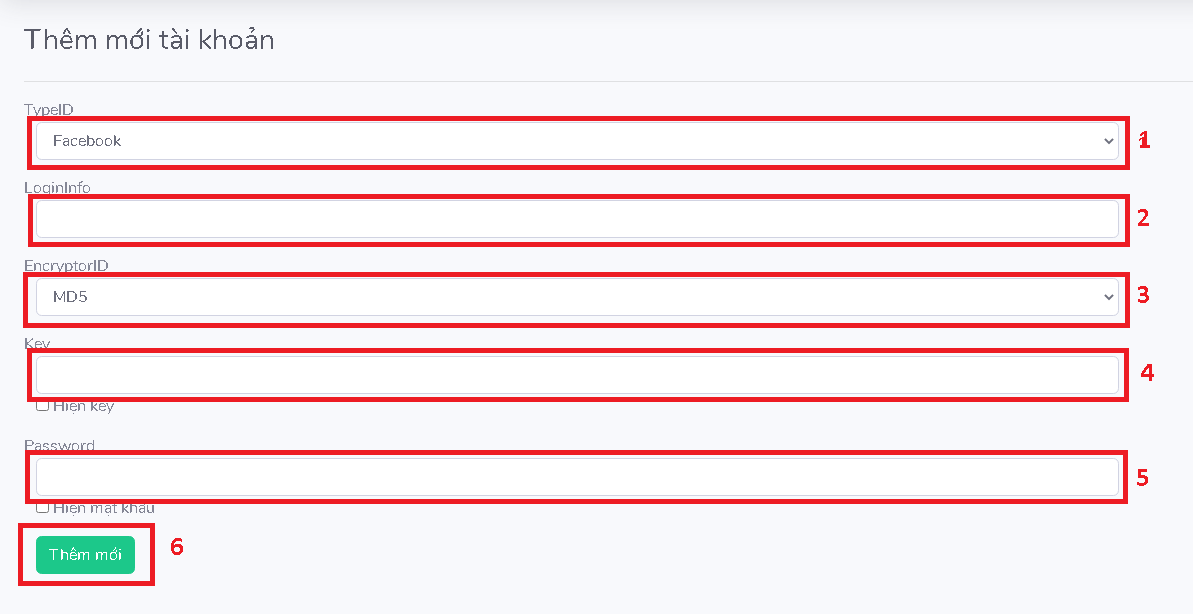
* (1) Click vào avatar
* (2) Chọn mục Thông tin



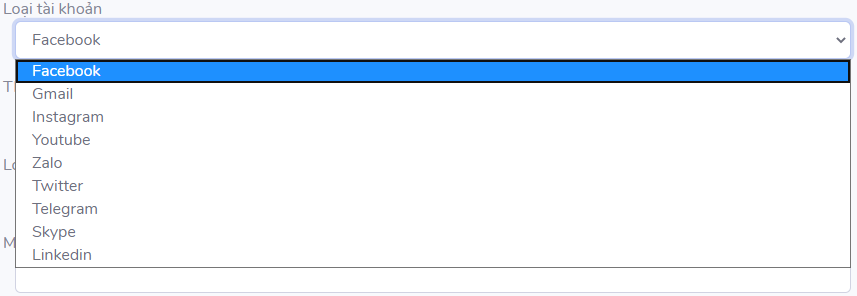
*Hình 5.1. Giao diện sửa thông tin tài khoản*

1. Giao diện thêm tài khoản mạng xã hội.

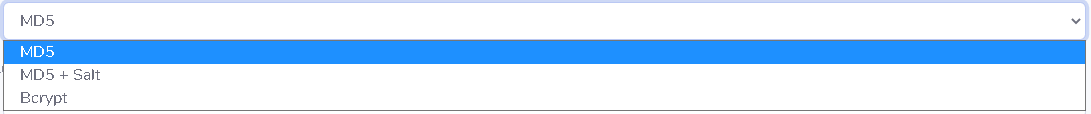
* (1) Chọn loại tài khoản.
* (2) Nhập thông tin đăng nhập.
* (3) Chọn loại mã hóa.
* (4) Nhập key.
* (5) Nhập mật khẩu
* (6) Click Thêm mới



*Hình 6.1. Giao diện thêm tài khoản mạng xã hội.*

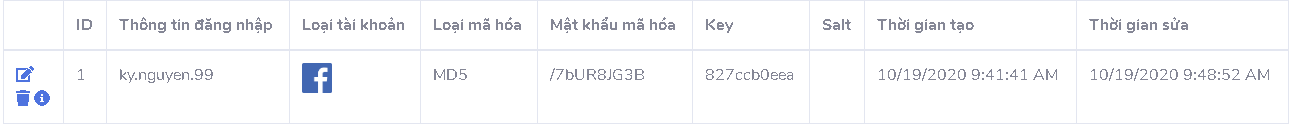


*Hình 6.2. Danh sách loại tài khoản trong database.*



*Hình 6.3. Ba thuật toán mã hóa.*

* 1. Mã hóa bằng MD5.



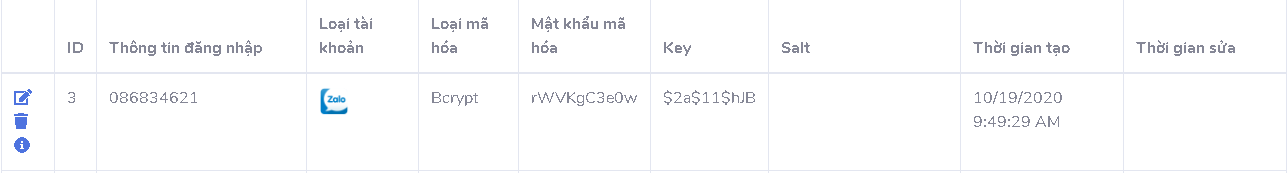
*Hình 6.4. Tài khoản với mật khẩu được mã hóa bằng MD5.*

* 1. Mã hóa bằng MD5 + Salt.



*Hình 6.5. Tài khoản với mật khẩu được mã hóa bằng MD5 + Salt.*

* 1. Mã hóa bằng Bcrypt.

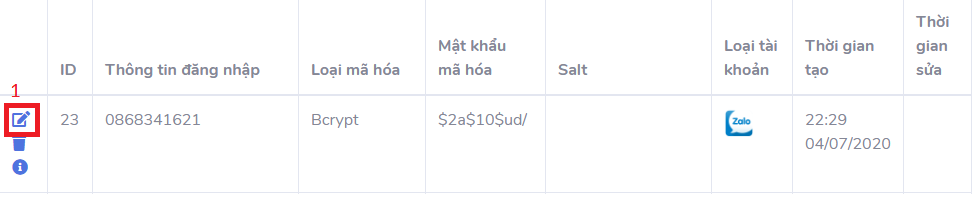


*Hình 6.6. Tài khoản với mật khẩu được mã hóa bằng Bcrypt.*

* Vì trường *Mật khẩu mã hóa* quá nhiều ký tự nên chỉ để hiển thị 10 ký tự đầu tiên.

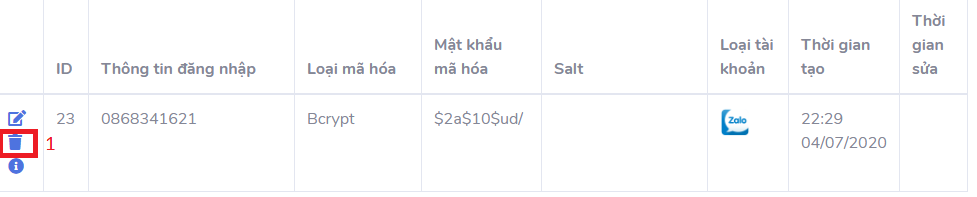
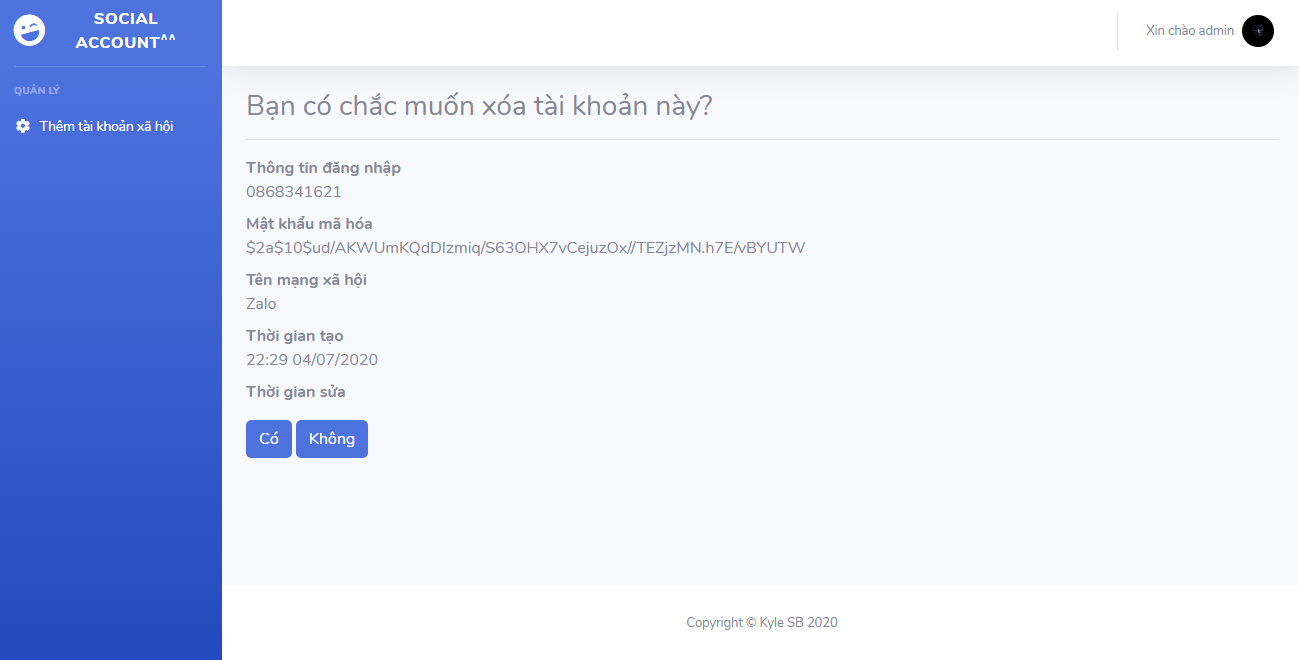
1. Giao diện sửa tài khoản mạng xã hội.

* (1) Click biểu tượng Edit để sửa tài khoản

*Hình 7.1. Giao diện sửa tài khoản mạng xã hội.*

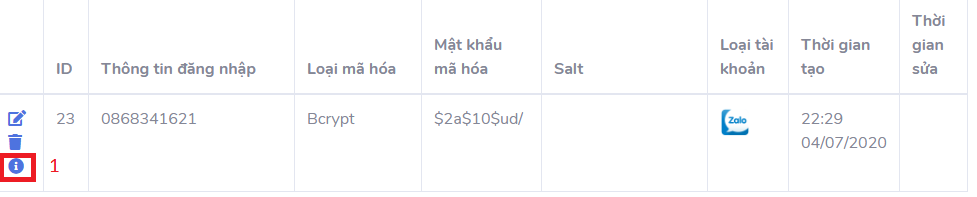
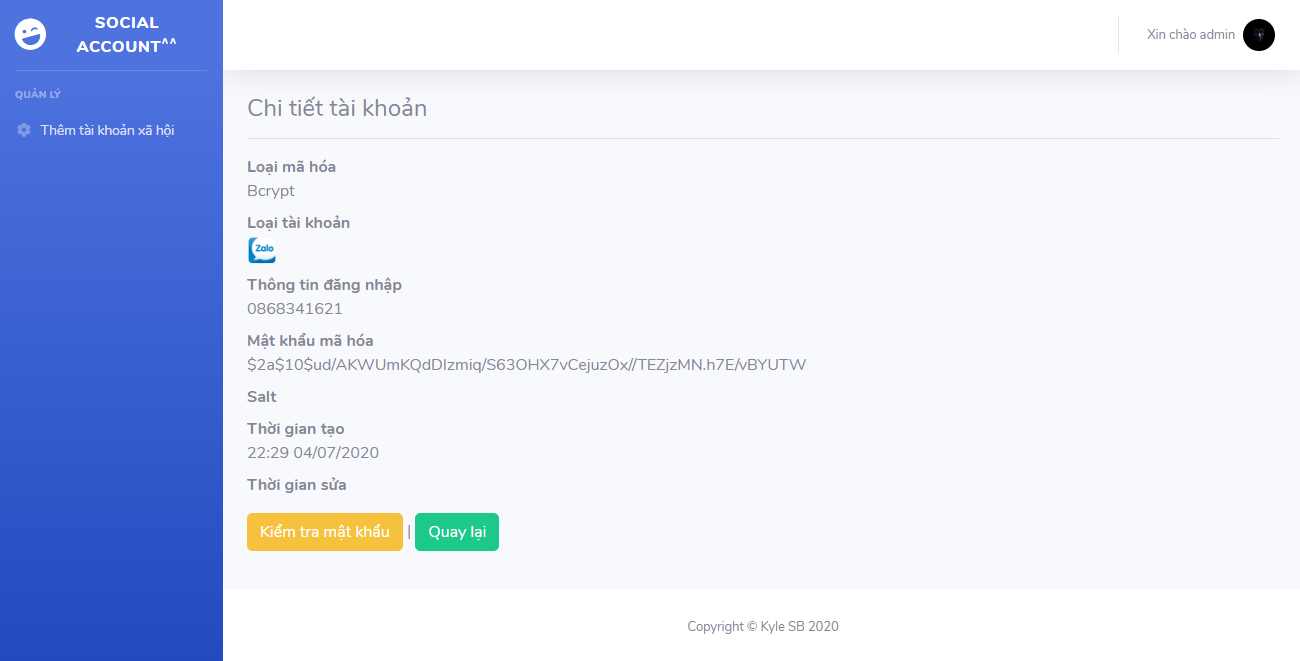
1. Giao diện xóa tài khoản mạng xã hội.

* (1) Click biểu tượng Thùng rác để xóa tài khoản.

*Hình 8.1. Giao diện xóa tài khoản mạng xã hội.*

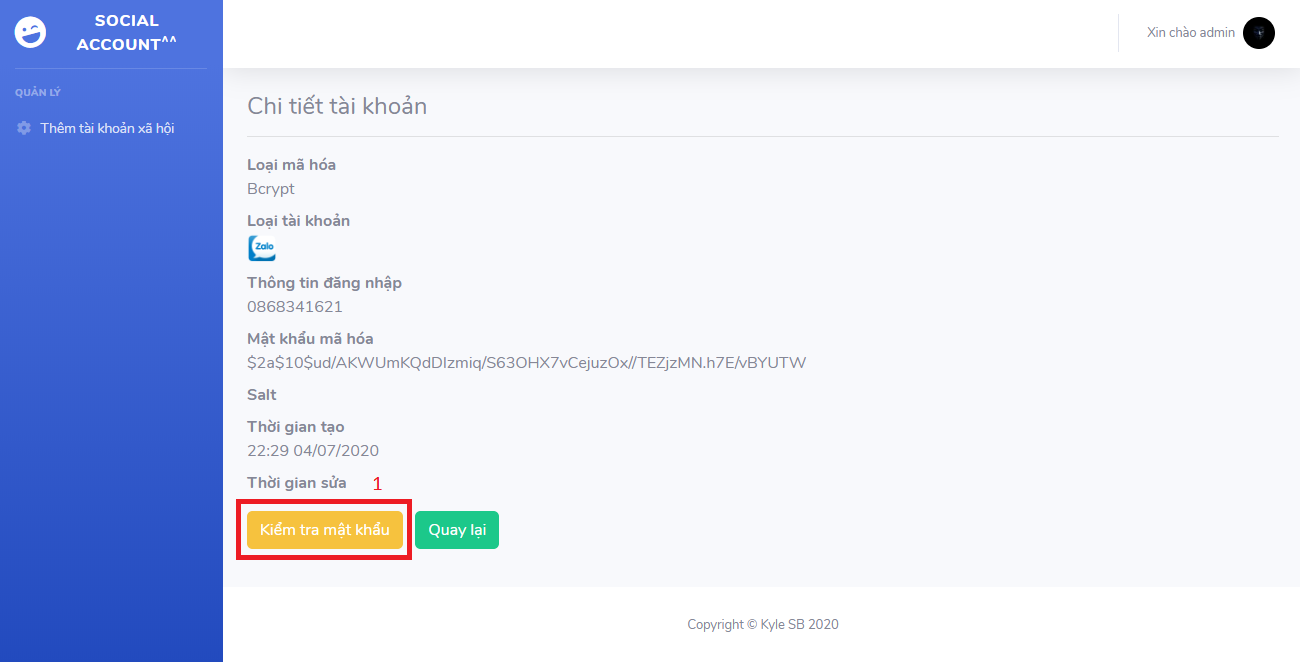
1. Giao diện chi tiết tài khoản mạng xã hội.

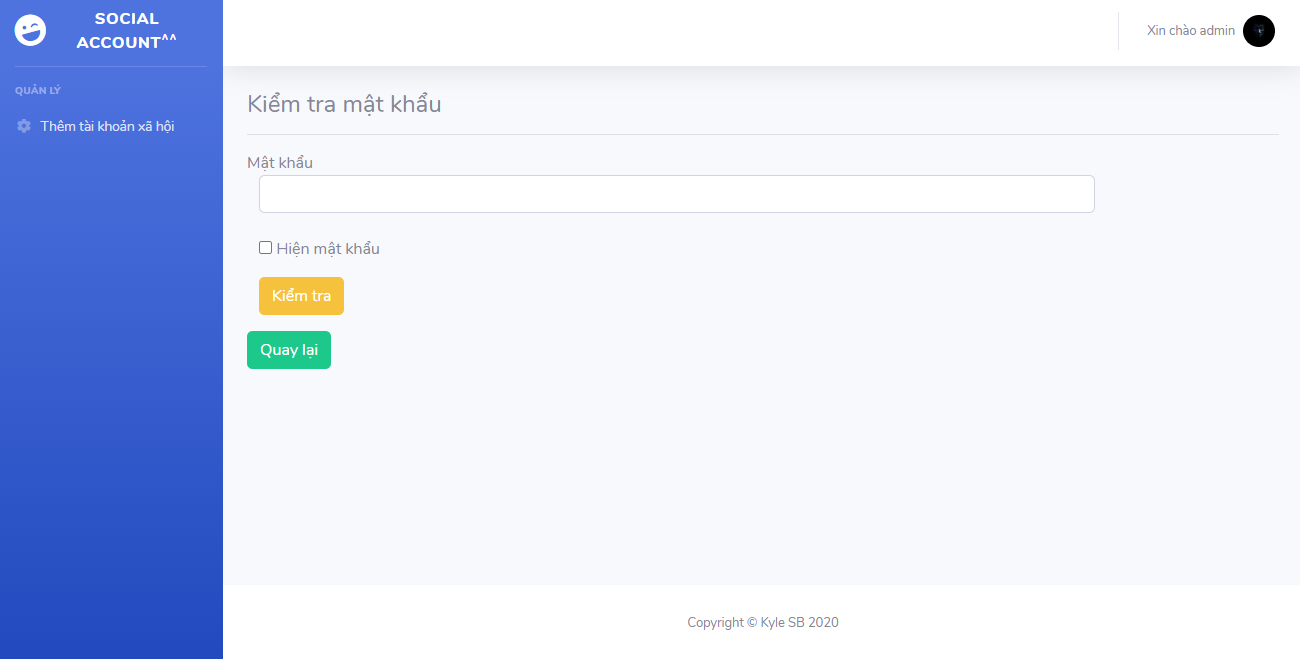
* (1) Click biểu tượng chữ i để xem chi tiết tài khoản.



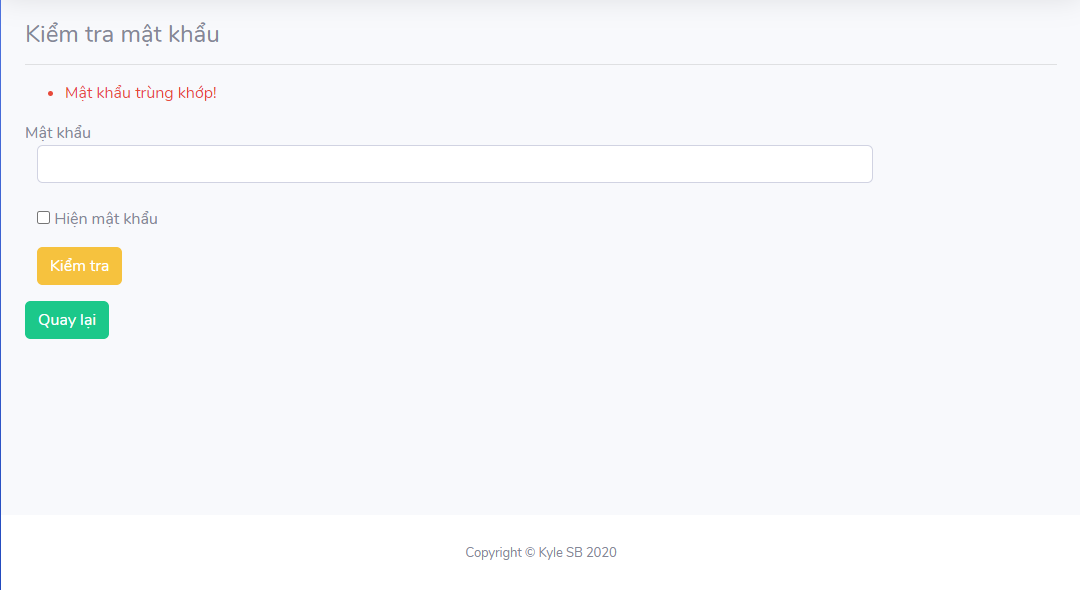
*Hình 9.1. Giao diện chi tiết tài khoản mạng xã hội.*

1. Giao diện chứng thực mật khẩu.

* (1) Cũng trong giao diện Chi tiết tài khoản, click vào nút Kiểm tra mật khẩu.

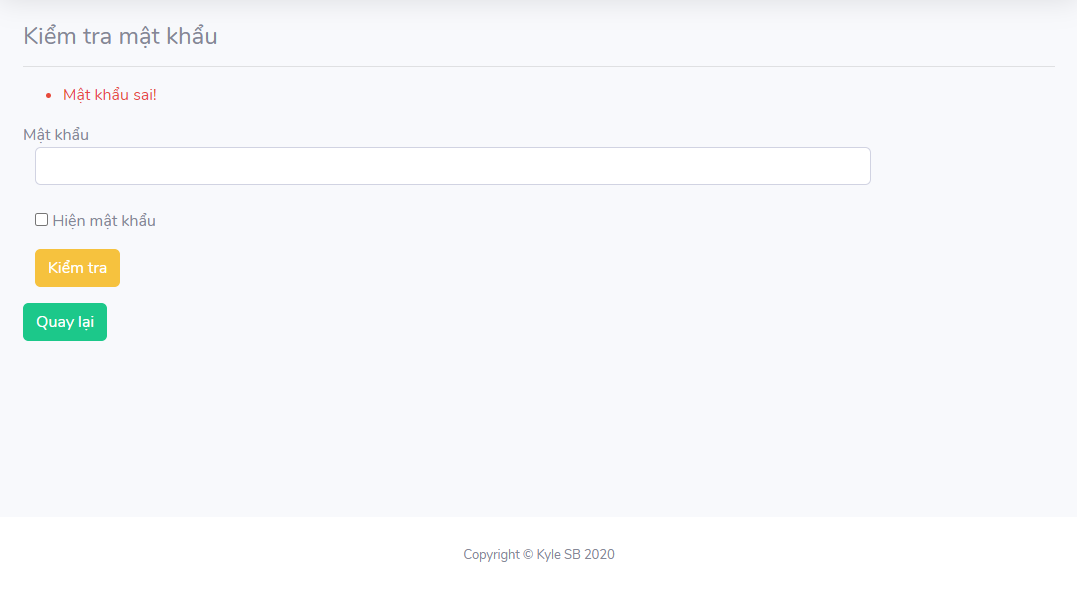


*Hình 10.1. Giao diện chứng thực mật khẩu.*

* Nếu nhập mật khẩu trùng khớp với mật khẩu lúc tạo thì sẽ thông báo Mật khẩu trùng khớp.

*Hình 10.2. Hiển thị thông báo Mật khẩu trùng khớp*

* Ngược lại thì thông báo Mật khẩu sai.



*Hình 10.3. Hiển thị thông báo Mật khẩu sai.*