|  |  |  |
| --- | --- | --- |
|  | TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC  **Khoa Công Nghệ Thông Tin** | Macintosh HD:Users:hoangnguyen:Documents:TDC:logoTDC_blue.png |
|  |  |  |

**CHUYÊN ĐỀ WEB 1** | HKI – 2021

PRJ – REPORT

**BÁO CÁO KẾT THÚC MÔN HỌC**

**NHÓM: D**

**THÀNH VIÊN:**

Trần Trọng Hiếu - 19211TT1201

Ngô Bảy Hên – 19211TT1120

Nguyễn Huỳnh Quốc Trung - 19211TT0744

**Mã lớp học phần:** 21111TNC10711102

Table of Contents

[1. DESIGN PATTERN 3](#_Toc86092985)

[1.1. Factory 3](#_Toc86092986)

[1.1.1. Factory là gì? 3](#_Toc86092987)

[1.1.2. Ví dụ thực tiễn. 3](#_Toc86092988)

[1.1.3. Minh họa lược đồ Factory. 3](#_Toc86092989)

[1.2. Repository 3](#_Toc86092990)

[1.2.1. Repository là gì? 3](#_Toc86092991)

[1.2.2. Ví dụ thực tiễn. 3](#_Toc86092992)

[1.2.3. Minh họa lược đồ Repository. 3](#_Toc86092993)

[1.3. Proxy 3](#_Toc86092994)

[1.3.1. Proxy là gì? 3](#_Toc86092995)

[1.3.2. Ví dụ thực tiễn. 3](#_Toc86092996)

[1.3.3. Minh họa lược đồ Proxy. 3](#_Toc86092997)

[2. DỰ ÁN CHUYÊN ĐỀ WEB 1 4](#_Toc86092998)

[2.1. Phân Công 4](#_Toc86092999)

[2.2. Đặc Tả Đồ Án 4](#_Toc86093000)

[2.2.1. Hiển thị confessions 5](#_Toc86093001)

[2.2.2. Phân trang hiển thị confessions 7](#_Toc86093002)

[2.2.3. Sửa confessions 11](#_Toc86093003)

[Hình 1. Gọi và khởi tạo userModel và bankModel. 4](#_Toc86431535)

[Hình 2. Khởi tạo một lớp FactoryModel có nhiệm vụ khai báo. 4](#_Toc86431536)

[Hình 3. Để sử dụng ta chỉ cần gọi một lớp Factory duy nhất. 5](#_Toc86431537)

[Hình 4. Trang chủ hiển thị confessions. 7](#_Toc86431538)

[Hình 5. Gọi useEffect để dispatch actions dữ liệu. 7](#_Toc86431539)

[Hình 6. Xử lí gọi API và đưa vào reducers của Redux. 8](#_Toc86431540)

[Hình 7. Sử dụng useSelector của Redux để lấy dữ liệu. 8](#_Toc86431541)

[Hình 8. Hiển thị dữ liệu bằng cách map qua mảng dữ liệu. 8](#_Toc86431542)

[Hình 9. Hiển thị phân trang trên giao diện. 9](#_Toc86431543)

[Hình 10. Lấy số trang hiện tại bằng useRouter, mặc định là 1. 9](#_Toc86431544)

[Hình 11. Sử dụng useEffect để xử lí bài viết hiển thị. 10](#_Toc86431545)

[Hình 12. Hiển thị pagination khi số lượng bài viết lớn hơn 0. 10](#_Toc86431546)

[Hình 13. Hiển thị Pagination cho bài viết. 11](#_Toc86431547)

[Hình 14. Logic xử lí hiển thị pagination. 11](#_Toc86431548)

[Hình 15. Hiển thị sửa khi người dùng đăng nhập và đúng bài viết. 12](#_Toc86431549)

[Hình 16. Hiển thị modal sửa khi người dùng nhấp vào nút sửa. 12](#_Toc86431550)

[Hình 17. Khi click vào nút sửa sẽ gọi đến function sửa. 13](#_Toc86431551)

[Hình 18. Xử lí sửa bài viết và load lại dữ liệu. 13](#_Toc86431552)

# 1. DESIGN PATTERN

## 1.1. Factory

### 1.1.1. Factory là gì?

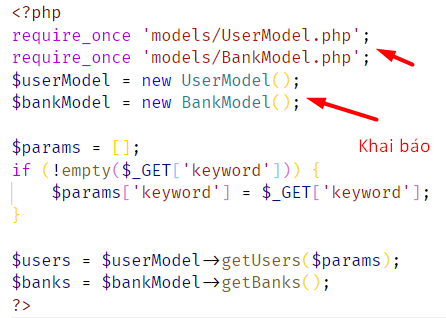
Factory design pattern rất phổ biến và không thể thiếu trong lập trình hướng đối tượng (OOP). Factory là pattern thuộc nhóm gọi là mẫu khởi tạo, giúp quản lý và cung cấp một cách ngắn gọn để khởi tạo đối tượng.

### 1.1.2. Ví dụ thực tiễn.

**Vấn đề:**

Để sử dụng một lớp đối tượng, ta cần gọi và khởi tạo để sử dụng nó, nhưng có một bất lợi là ta sử dụng nhiều lớp trong một lần, rất bất tiện khi phải cứ gọi và khởi tạo.

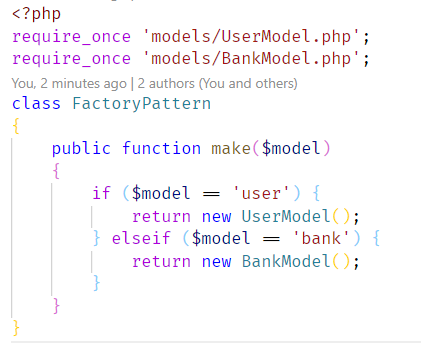
Dưới đây là lớp Users và Banks được gọi vào và khởi tạo để sử dụng, ta có thể thấy nó khá rườm rà và không tối ưu.



Hình . Gọi và khởi tạo userModel và bankModel.

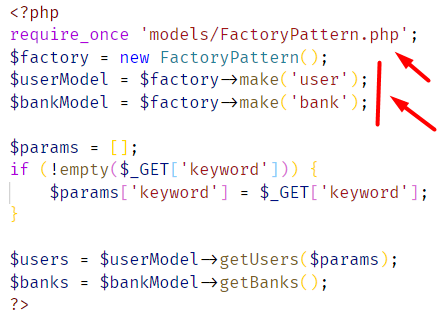
**Giải pháp:**

Để khắc phục điều này, ta sẽ gọi một lớp khác, có nhiệm vụ gọi thay cho lớp gốc và sử dụng rất dễ dàng. Ta cần khai báo thêm một lớp như sau.



Hình . Khởi tạo một lớp FactoryModel có nhiệm vụ khai báo.

Để sử dụng, ta chỉ cần gọi và khai báo duy nhất lớp FactoryModel và mọi thứ sẽ ngắn gọn và tối ưu hơn.



Hình . Để sử dụng ta chỉ cần gọi một lớp Factory duy nhất.

### 1.1.3. Minh họa lược đồ Factory.

Hình minh họa dưới đây mô tả lớp ShapeFactory được khởi tạo, và gọi đến phương thức getShape(), phương thức getShape sẽ được truyền vào 3 giá trị “circle”, “rectangle”, “square” để lấy được lớp tương ứng mà không cần phải tự gọi và khai báo từng phần.

Các lớp tương ứng có cùng một phương thức draw() được kế thừa từ Shape Interface.



Hình . Lược đồ UML Factory Pattern của Shape.

## 1.2. Repository

### 1.2.1. Repository là gì?

### 1.2.2. Ví dụ thực tiễn.

### 1.2.3. Minh họa lược đồ Repository.

## 1.3. Proxy

### 1.3.1. Proxy là gì?

Proxy là một đối tượng sẽ đại diện cho một đối tượng khác.

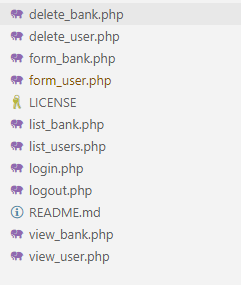
* Kiểm soát quyền truy xuất các phương thức của đối tượng.
* Bổ sung thêm chức năng trước khi thực thi phương thức.
* Tạo ra đối tượng mới có chức năng nâng cao hơn đối tượng ban đầu.
* Giảm chi phí khi có nhiều truy cập vào đối tượng có chi phí khởi tạo ban đầu lớn.

### 1.3.2. Ví dụ thực tiễn.

**Vấn đề:**

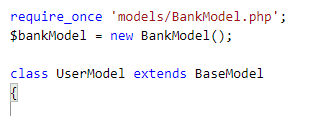
Nếu chúng ta có 1 đồ án lớn mà có những bảng giống nhau, chúng ta cứ code thì sẽ bất tiện và mất thời gian

Dưới đây là Users và Banks, ta có thể thấy chúng ta sẽ code rất nhiều và rườm rà, dù đây là 2 bảng giống nhau

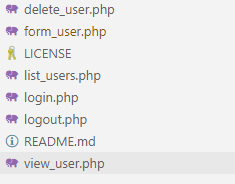


**Giải pháp:**

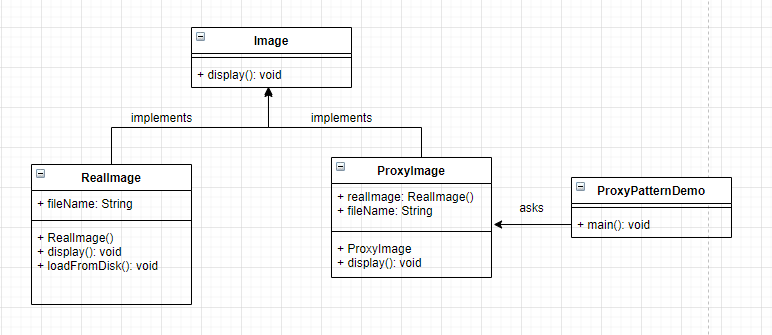
1. Cô lập UsersModel và BankModel
2. UserModel sẽ gọi BankModel và làm những việc của BankModel



1. Để sử dụng, ta chỉ cần gọi vào User thì Bank cũng sẽ chạy, và mọi thứ sẽ ngắn gọn và tối ưu hơn.



### 1.3.3. Minh họa lược đồ Proxy.



# 2. DỰ ÁN CHUYÊN ĐỀ WEB 1

Ứng dụng Confessions Vietnam.

Trong thời buổi công nghệ hiện nay, việc chia sẻ thông tin là rất quan trọng. Những vấn đề trong cuộc sống khó khăn càng khiến người ta mệt mỏi và cần người tâm sự. Ứng dụng được tạo ra giúp các bạn chia sẻ và giải tỏa những điều trong lòng, những niềm vui, nỗi buồn một cách hoàn toàn ẩn danh, hoàn toàn bảo mật. Chia sẻ và kết nối cùng những người khác.

- Hoàn thiện đặc tả yêu cầu đồ án theo hướng tính năng rõ ràng

- Hoàn thiện đặc tả yêu cầu đồ án theo hướng hình minh họa phát thảo rõ ràng

- Hoàn thiện đặc tả yêu cầu đồ án theo hướng kế hoạch (phân công, thời gian rõ ràng, git)

## 2.1. Phân Công

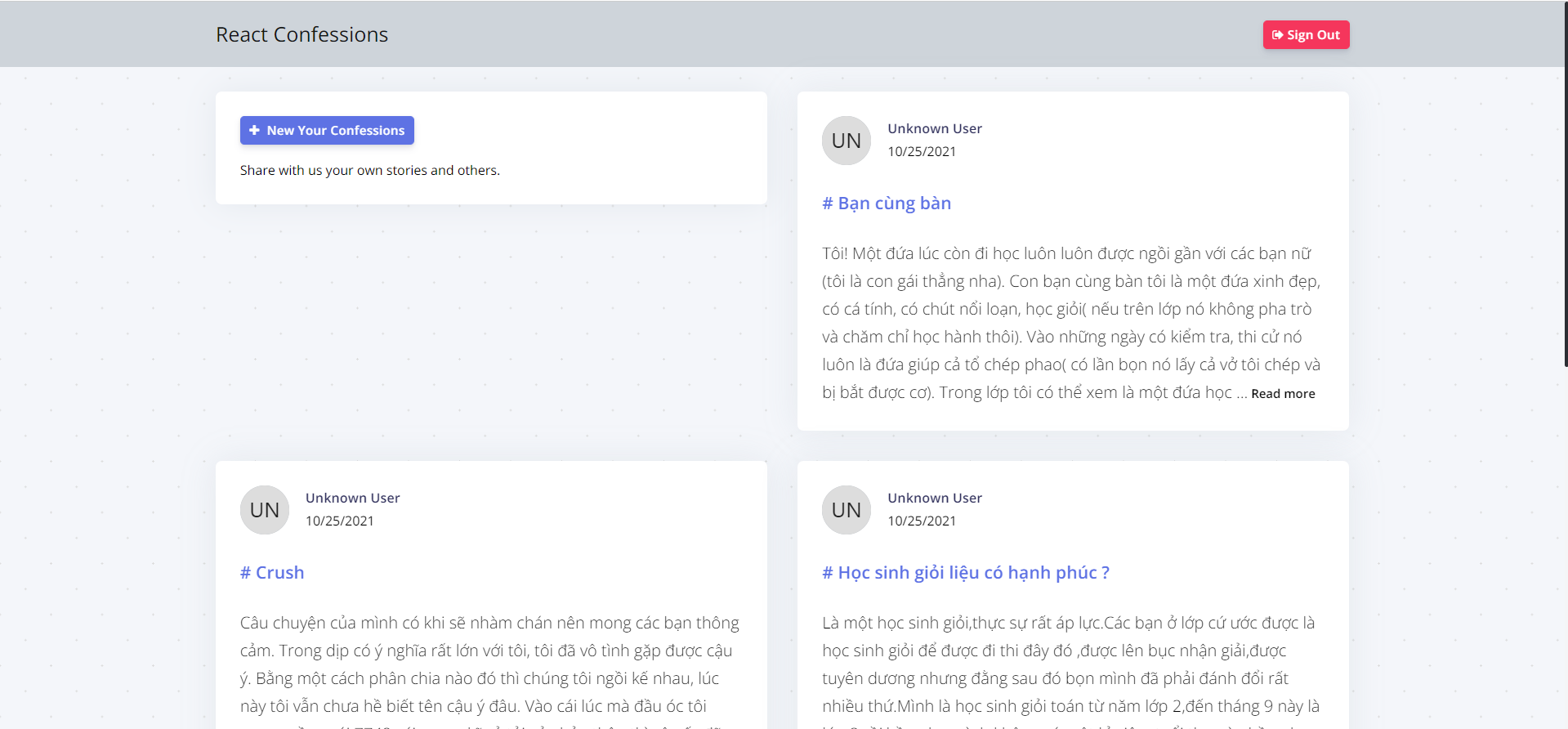
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thành Viên** | **Nội Dung Phân Công** | **Branch (feature/)** | **Thời Gian** |
| 1 | Ngô Bảy Hên | Đăng nhập | Sign-In | 5 ngày |
| Đăng ký | Sign-Up | 5 ngày |
| Đăng xuất người dùng | Sign-Out | 2 ngày |
| 2 | Trần Trọng Hiếu | Hiển thị confessions | Show-Confession | 4 ngày |
| Phân trang hiển thị confessions | Pagination-Confession | 4 ngày |
| Sửa confessions | Edit-Confession | 4 ngày |
| 3 | Nguyễn Huỳnh Quốc Trung | Thêm confessions | Add-Confession | 5 ngày |
| Xóa confessions | Delete-Confession | 3 ngày |

## 2.2. Đặc Tả Đồ Án

### 2.2.1. Hiển thị confessions

Đầu tiên là hiển thị confession. Đây là giao diện hiển thị của confessions bao gồm các phần chính mà dữ liệu được lấy từ API:

* Thời gian của bài viết.
* Tiêu đề của bài viết.
* Nội dung của bài viết.

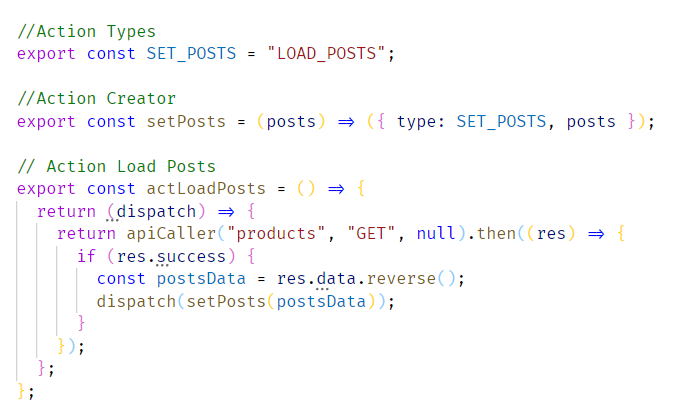


Hình . Trang chủ hiển thị confessions.

Khi vào trang có chứa Posts thực hiện gọi dispatch actions bằng useEffect, và thực hiện gọi API trong Redux, dữ liệu được đưa vào reducers của Redux.

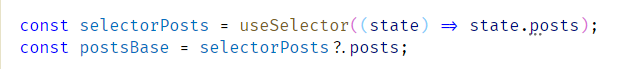


Hình . Gọi useEffect để dispatch actions dữ liệu.

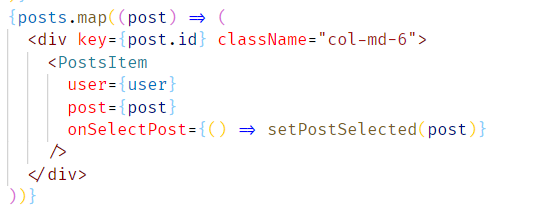


Hình . Xử lí gọi API và đưa vào reducers của Redux.

Sau khi lấy dữ liệu từ API, ta tiến hành lấy dữ liệu ra từ Redux và hiển thị ra màn hình của người dùng (PostsItem là components giao diện hiển thị).



Hình . Sử dụng useSelector của Redux để lấy dữ liệu.

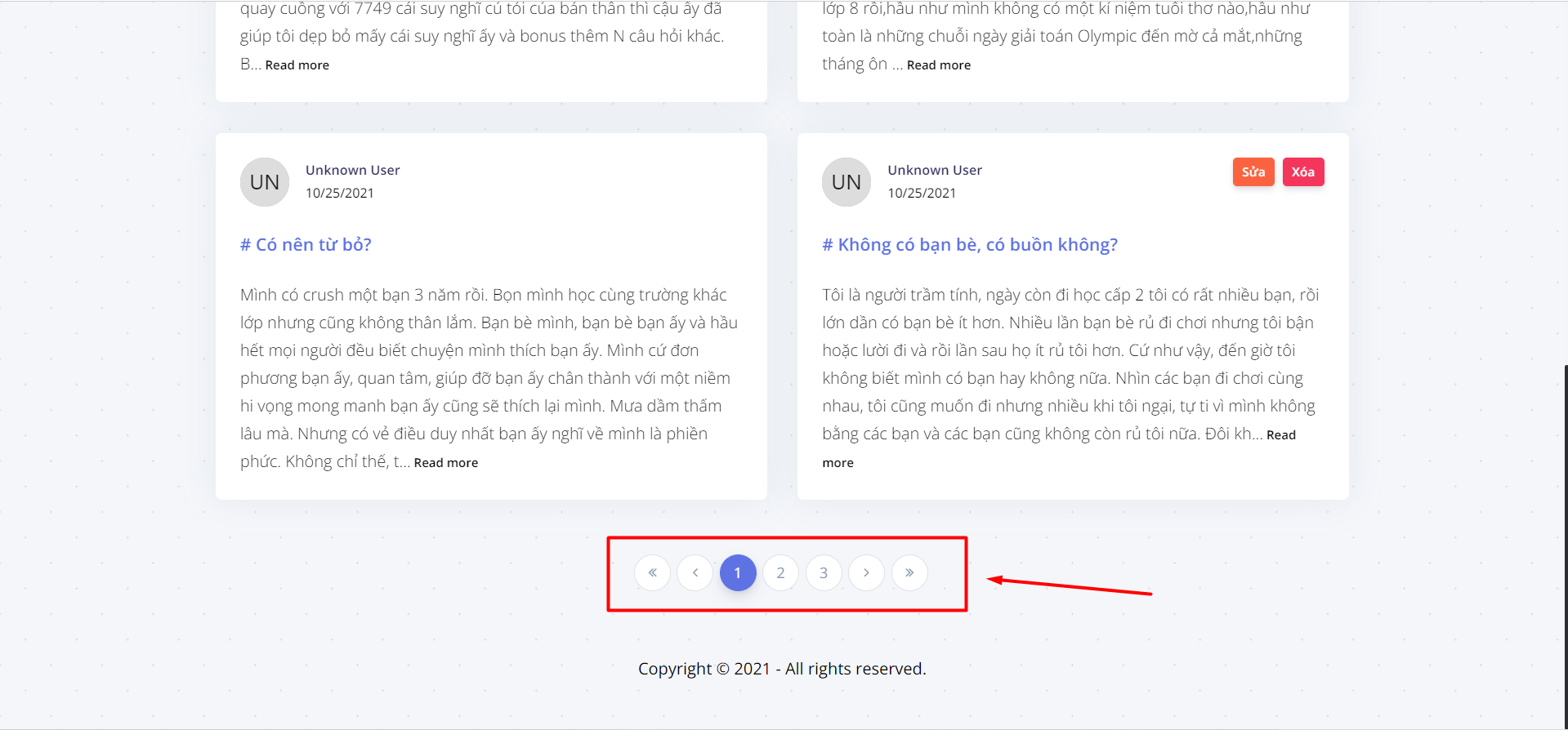


Hình . Hiển thị dữ liệu bằng cách map qua mảng dữ liệu.

### 2.2.2. Phân trang hiển thị confessions

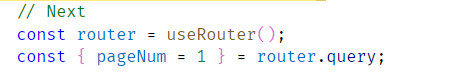
Hiển thị phân trang ở phía dưới của bài viết, giao diện bao gồm:

* Nút back về trang thứ 1.
* Nút back về trang trước đó.
* Hiển thị số trang.
* Nút next đến trang tiếp theo.
* Nút next đến trang cuối cùng.



Hình . Hiển thị phân trang trên giao diện.

Để phân trang ta cần lấy số trang hiện tại, vì sử dụng NextJS nên ta có useRouter.



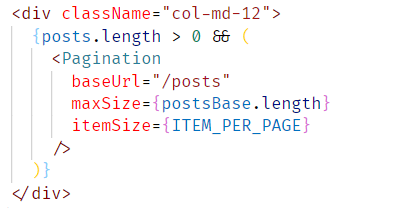
Hình . Lấy số trang hiện tại bằng useRouter, mặc định là 1.

Dữ liệu Posts được lấy từ API sẽ được để trong biến postsBase, và dữ liệu hiển thị theo trang hiện tại (posts) sẽ được cắt mảng ra theo chiều dài được xác định trước (ITEM\_PER\_PAGE).



Hình . Sử dụng useEffect để xử lí bài viết hiển thị.

Hiển thị Pagination ở trang chủ bằng components Pagination.

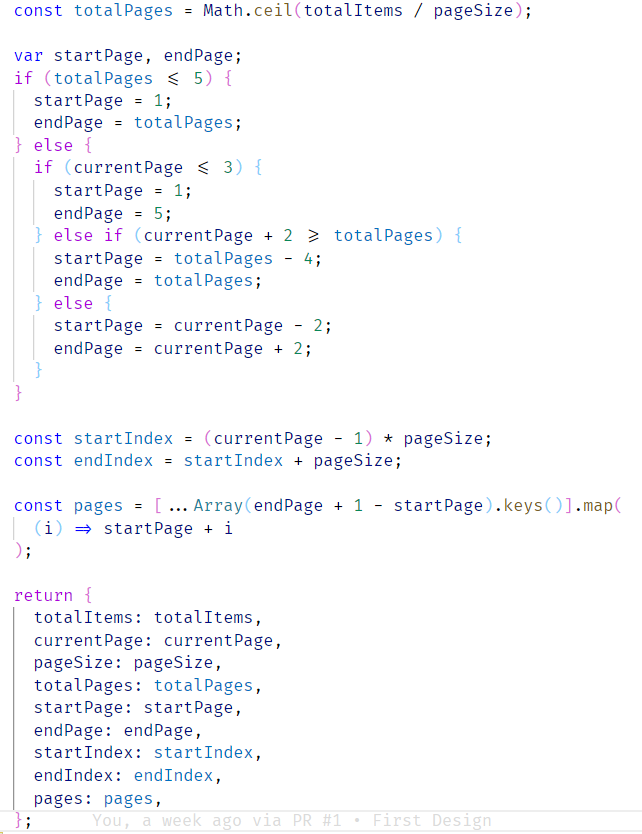


Hình . Hiển thị pagination khi số lượng bài viết lớn hơn 0.

Xử lí logic của Pagination được gọi trong hàm getPaginate và hiển thị số của paginate bằng cách map qua pages.



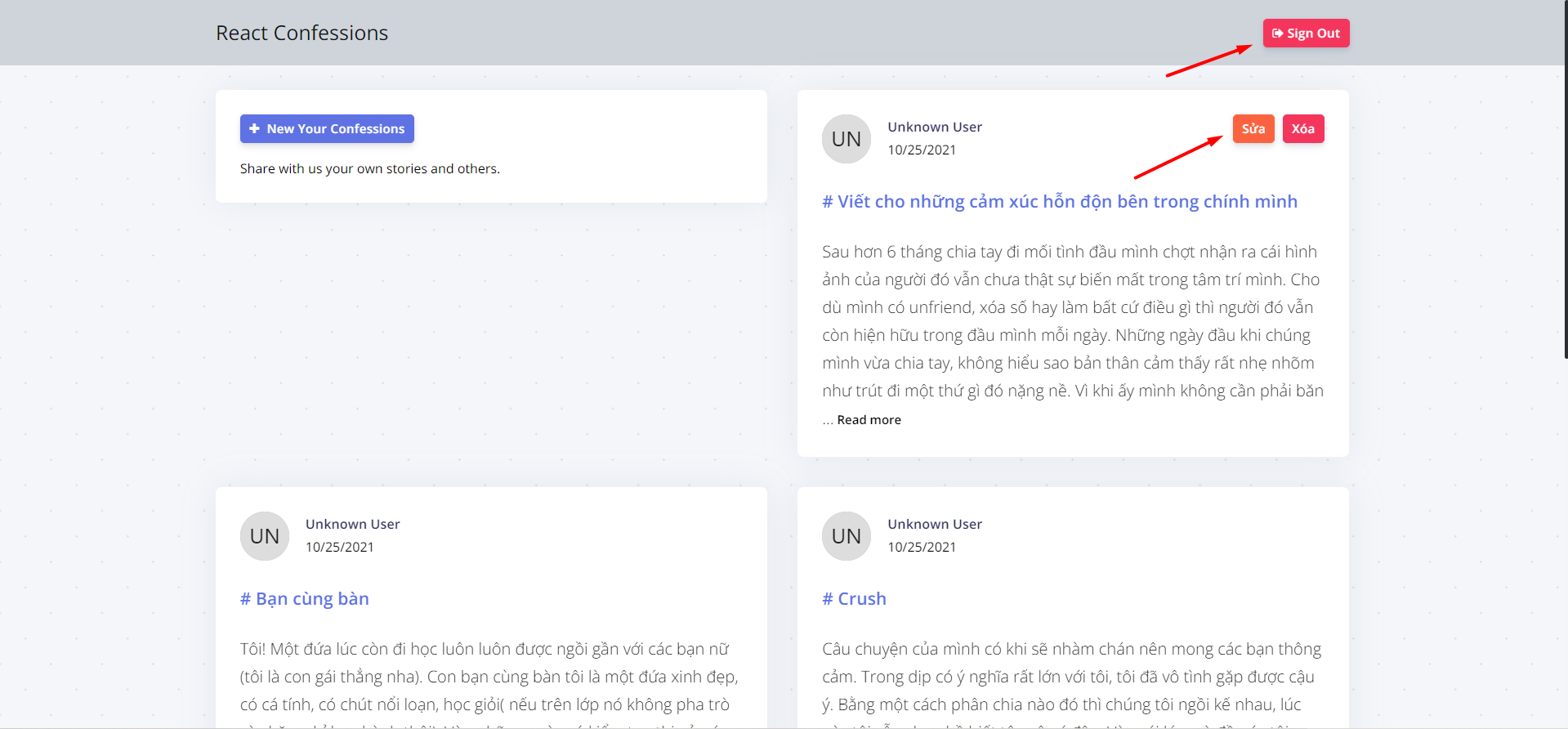
Hình . Hiển thị Pagination cho bài viết.



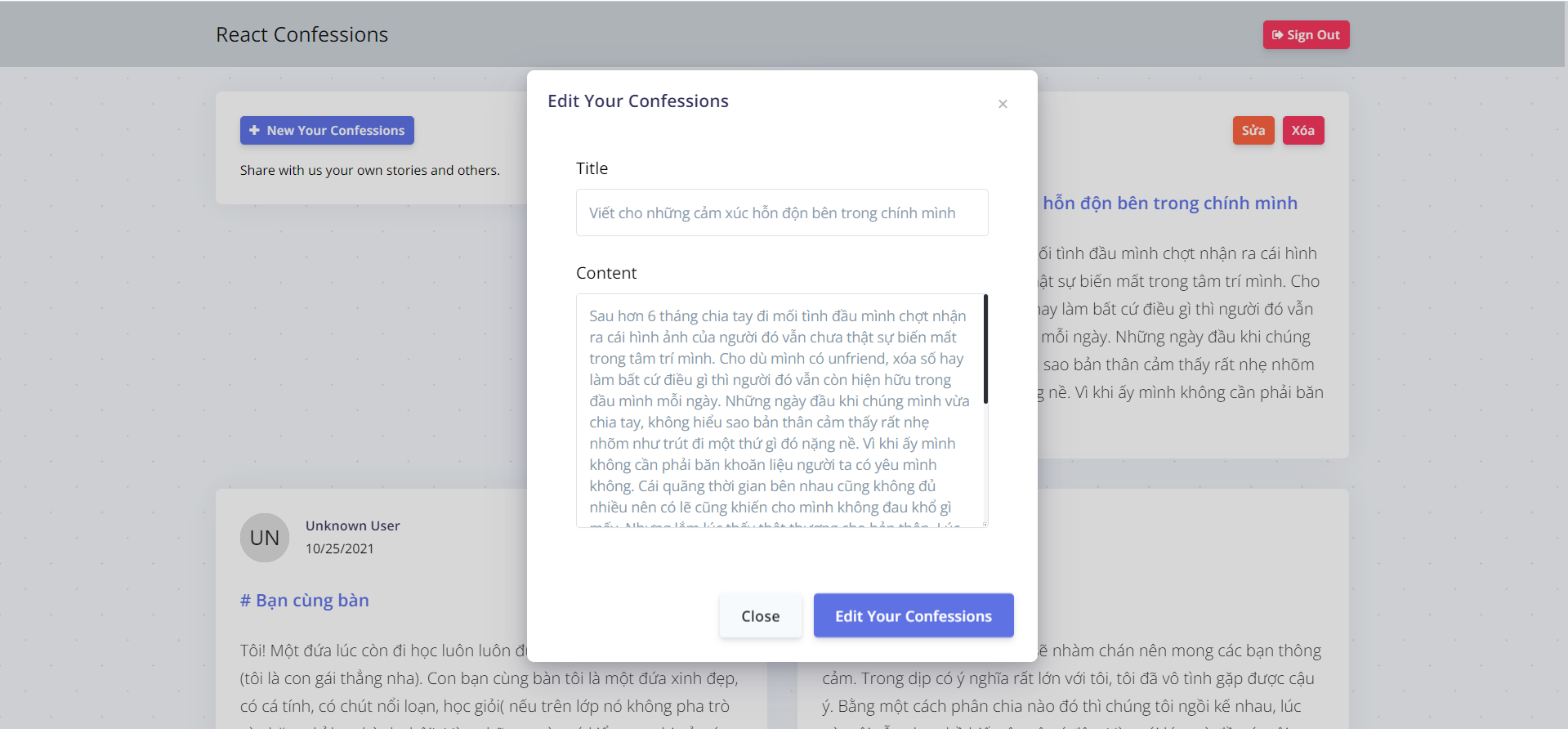
Hình . Logic xử lí hiển thị pagination.

### 2.2.3. Sửa confessions

Giao diện sửa confession chỉ cho người đã đăng nhập và tạo bài viết, khi người dùng nhấn vào nút sửa, một modal sẽ hiện ra và cho phép sửa bài viết.

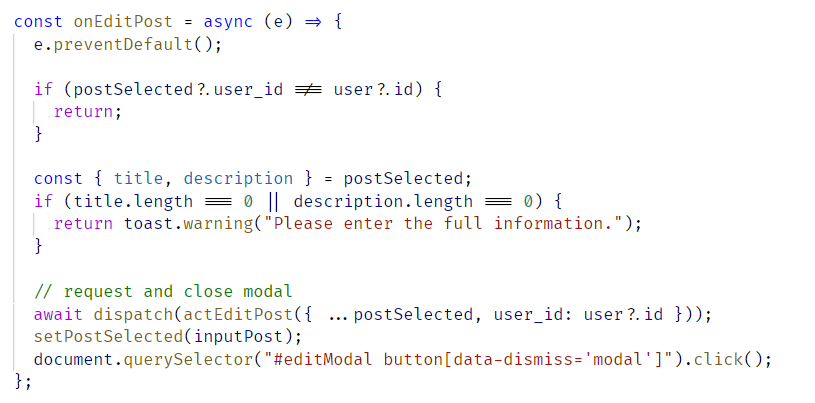


Hình . Hiển thị sửa khi người dùng đăng nhập và đúng bài viết.



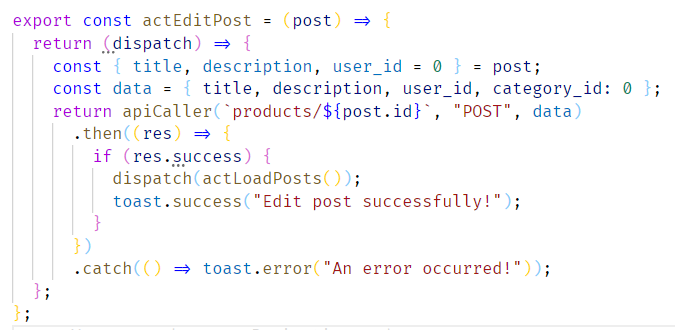
Hình . Hiển thị modal sửa khi người dùng nhấp vào nút sửa.

Khi click vào nút sửa sẻ gọi đến function sửa, đầu tiên sẽ check đúng user id hiện tại. Sau đó check đang input hiện tại có rỗng hay không. Tiếp tục gọi dispatch actions sửa bài viết của Redux, sau đó đóng modal lại.



Hình . Khi click vào nút sửa sẽ gọi đến function sửa.

Ở trong Redux sẽ xứ lí chức năng sửa bài viết, và load lại tất cả bài viết hiện tại.



Hình . Xử lí sửa bài viết và load lại dữ liệu.