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

**CHUYÊN ĐỀ WEB 2** | 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

[DỰ ÁN CHUYÊN ĐỀ WEB 5](#_Toc89450748)

[2.1. Mô Tả Công Nghệ Được Sử Dụng 6](#_Toc89450749)

[2.1.1. ReactJS là gì? 6](#_Toc89450750)

[2.1.2. Redux là gì? 7](#_Toc89450751)

[2.1.3. NextJS là gì? 7](#_Toc89450752)

[2.2. Đặc Tả Dự Án 8](#_Toc89450753)

[2.2.1. Hiển thị confessions 9](#_Toc89450754)

[2.2.2. Phân trang hiển thị confessions 11](#_Toc89450755)

[2.2.3. Sửa confessions 15](#_Toc89450756)

[2.2.4. Đăng nhập confessions 17](#_Toc89450757)

[2.2.5. Đăng ký confessions 20](#_Toc89450758)

[2.2.6. Đăng xuất confessions 23](#_Toc89450759)

[2.2.7. Thêm confessions 25](#_Toc89450760)

[2.2.8. Xóa confessions 27](#_Toc89450761)

[2.2.9. Loading khi xử lý 29](#_Toc89450762)

[2.2.10. Tìm kiếm bài viết 31](#_Toc89450763)

[2.2.11. Trang chi tiết người dùng 32](#_Toc89450764)

[2.2.12. Trang chi tiết bài viết (bài viết liên quan) 34](#_Toc89450765)

[2.2.13. Trang chi tiết bài viết (nội dung) 37](#_Toc89450766)

[2.2.14. Đa ngôn ngữ 39](#_Toc89450767)

[Hình 1. Giao diện trang chủ của dự án khi hoàn thành. 4](#_Toc89450658)

[Hình 2. Hình ảnh Logo của ReactJS. 5](#_Toc89450659)

[Hình 3. Hình ảnh mô tả vòng đời (life cycle) của Redux. 6](#_Toc89450660)

[Hình 4. Hình ảnh Logo của NextJS. 6](#_Toc89450661)

[Hình 5. Trang chủ hiển thị confessions. 8](#_Toc89450662)

[Hình 6. Gọi useEffect để dispatch actions dữ liệu. 8](#_Toc89450663)

[Hình 7. Gọi API và đưa vào reducers của Redux. 9](#_Toc89450664)

[Hình 8. Sử dụng useSelector của Redux để lấy dữ liệu. 9](#_Toc89450665)

[Hình 9. Hiển thị dữ liệu bằng cách map qua mảng dữ liệu. 9](#_Toc89450666)

[Hình 10. Hiển thị phân trang trên giao diện. 10](#_Toc89450667)

[Hình 11. Lấy số trang hiện tại bằng useRouter, mặc định là 1. 10](#_Toc89450668)

[Hình 12. Sử dụng useEffect để xử lí bài viết hiển thị. 11](#_Toc89450669)

[Hình 13. Hiển thị pagination khi số lượng bài viết lớn hơn 0.. 11](#_Toc89450670)

[Hình 14. Hiển thị Pagination cho bài viết. 12](#_Toc89450671)

[Hình 15. Logic xử lí hiển thị pagination. 13](#_Toc89450672)

[Hình 16. Hiển thị sửa khi người dùng đăng nhập và đúng bài viết. 14](#_Toc89450673)

[Hình 17. Hiển thị modal sửa khi người dùng nhấp vào nút sửa. 14](#_Toc89450674)

[Hình 18. Khi click vào nút sửa sẽ gọi đến function sửa. 15](#_Toc89450675)

[Hình 19. Xử lí sửa bài viết và load lại dữ liệu. 15](#_Toc89450676)

[Hình 20. Nơi hiển thị button Sign In. 16](#_Toc89450677)

[Hình 21. Hiện model để đăng nhập. 16](#_Toc89450678)

[Hình 22. Hiện model để đăng ký. 17](#_Toc89450679)

[Hình 23. Hàm Sign Up action ở Redux. 18](#_Toc89450680)

[Hình 24. Model Sign In. 19](#_Toc89450681)

[Hình 25. Sau đăng nhập thành công. 19](#_Toc89450682)

[Hình 26. Hàm Redux xử lý đăng nhập. 20](#_Toc89450683)

[Hình 27. Hàm sẽ thực hiện khi người dùng reload trang. 20](#_Toc89450684)

[Hình 28. Hàm Redux loading đăng nhập. 21](#_Toc89450685)

[Hình 29. Hiện button Sign Out. 22](#_Toc89450686)

[Hình 30. Ở hàm authReducer. 22](#_Toc89450687)

[Hình 31. Hàm Logout ở Redux. 23](#_Toc89450688)

[Hình 32. HIện thông báo. 24](#_Toc89450689)

[Hình 33. Model Thêm Confessions. 24](#_Toc89450690)

[Hình 34. Khi click vào nút thêm sẽ gọi đến function thêm. 25](#_Toc89450691)

[Hình 35. Xử lý thêm bài viết và load lại dữ liệu. 25](#_Toc89450692)

[Hình 36. Hiển thị xóa khi người dùng đăng nhập và đúng bài viết. 26](#_Toc89450693)

[Hình 37. Hiển thị Model xóa khi người dùng click vào nút xóa. 26](#_Toc89450694)

[Hình 38. Khi click vào nút xóa sẽ gọi đến function xóa. 27](#_Toc89450695)

[Hình 39. Xử lý xóa bài viết và load lại dữ liệu. 27](#_Toc89450696)

[Hình 40. Giao diện loading. 28](#_Toc89450697)

[Hình 41. Tạo 2 case action. 28](#_Toc89450698)

[Hình 42. Dispatch action trong một function xử lý khác. 29](#_Toc89450699)

[Hình 43. Gọi giá trị loading từ Redux và sử dụng. 29](#_Toc89450700)

[Hình 44. Giao diện sau khi tìm kiếm. 30](#_Toc89450701)

[Hình 45. Khai báo onSearch. 30](#_Toc89450702)

[Hình 46. useEffect hiển thị mục tìm kiêm. 30](#_Toc89450703)

[Hình 47. Lọc các bài viết của người dùng đã tạo. 31](#_Toc89450704)

[Hình 48. Sử dụng redux mà trước đó đã lưu thông tin người dùng. 31](#_Toc89450705)

[Hình 49. Nhúng thư viện mã hóa md5 vào dự án. 31](#_Toc89450706)

[Hình 50. Hiển thị hình ảnh người dùng sau khi hash ra mã md5. 32](#_Toc89450707)

[Hình 51. Giao diện hiển thị của bài viết liên quan. 33](#_Toc89450708)

[Hình 52. Function trộn thứ tự của các bài viết. 34](#_Toc89450709)

[Hình 53. Khai báo và gọi hàm bài viết liên quan. 34](#_Toc89450710)

[Hình 54. Kế thừa từ thằng cha của bài viết liên quan. 35](#_Toc89450711)

[Hình 55. Click vào tiêu đề bài viết. 36](#_Toc89450712)

[Hình 56. Giao diện trang chi tiết bài viết. 36](#_Toc89450713)

[Hình 57. useEffect và UseState trang chi tiết bài viết. 37](#_Toc89450714)

[Hình 58. HIển thị thông tin trong trang chi tiết bài viết. 37](#_Toc89450715)

[Hình 59. Thư mục chứa nội dung bản dịch và hình ảnh quốc kỳ. 38](#_Toc89450716)

[Hình 60. File nội dung của bản dịch Tiếng Anh. 38](#_Toc89450717)

[Hình 61. Sử dụng thẻ Link của NextJS để set locale của ứng dụng. 39](#_Toc89450718)

[Hình 62. Kiểm tra thư viện đã cài đặt trong package.json. 39](#_Toc89450719)

[Hình 63. Cấu hình tại file next-i18next.config.js. 40](#_Toc89450720)

[Hình 64. Import file cấu hình vào next.config.js. 40](#_Toc89450721)

[Hình 65. Thêm cấu hình vào file \_app.js. 40](#_Toc89450722)

[Hình 66. Sử dụng serverSideTranslations cho từng trang. 41](#_Toc89450723)

[Hình 67. Sử dụng hook useTranslation để gọi bản dịch. 41](#_Toc89450724)

[Hình 68. Để hiển thị ta chỉ cần gọi ra mã nội dung của nó. 42](#_Toc89450725)

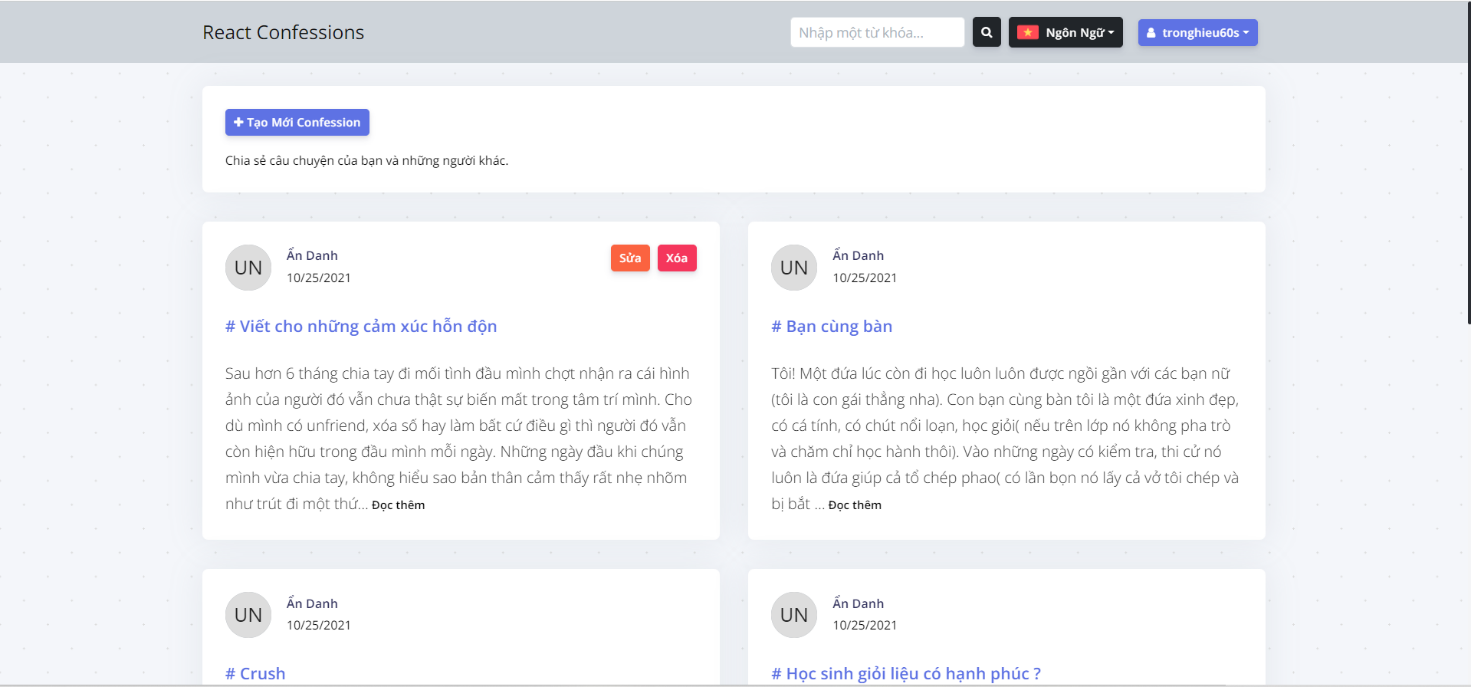
[Hình 69. Nội dung được hiển thị bằng Tiếng Anh. 43](#_Toc89450726)

[Hình 70. Nội dung được hiển thị bằng Tiếng Việt. 43](#_Toc89450727)

# DỰ ÁN CHUYÊN ĐỀ WEB

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.



Hình 1. Giao diện trang chủ của dự án khi hoàn thành.

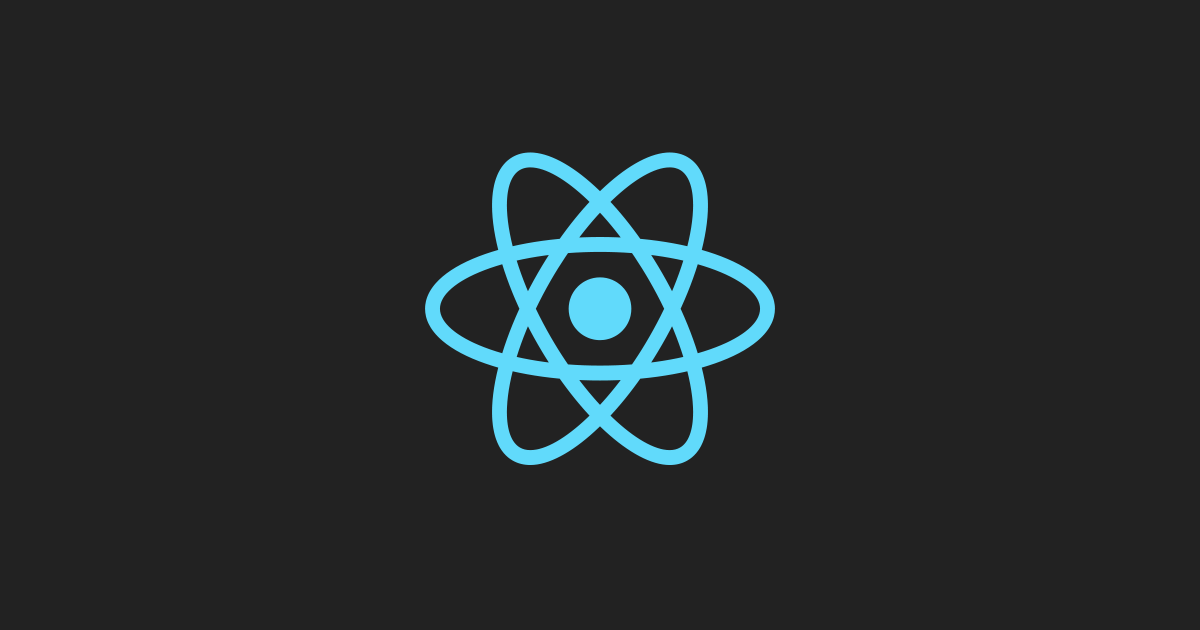
## 2.1. Mô Tả Công Nghệ Được Sử Dụng

Công nghệ được sử dụng: ReactJS (Redux, NextJS)

### 2.1.1. ReactJS là gì?

ReactJs là một thư viện viết bằng **Javascript**, dùng để xây dựng giao diện người dùng (UI). ReactJS được duy trì bởi Facebook và cộng đồng developer hùng mạnh. React cơ bản được sử dụng để phát triển các các **Single Page Application**.

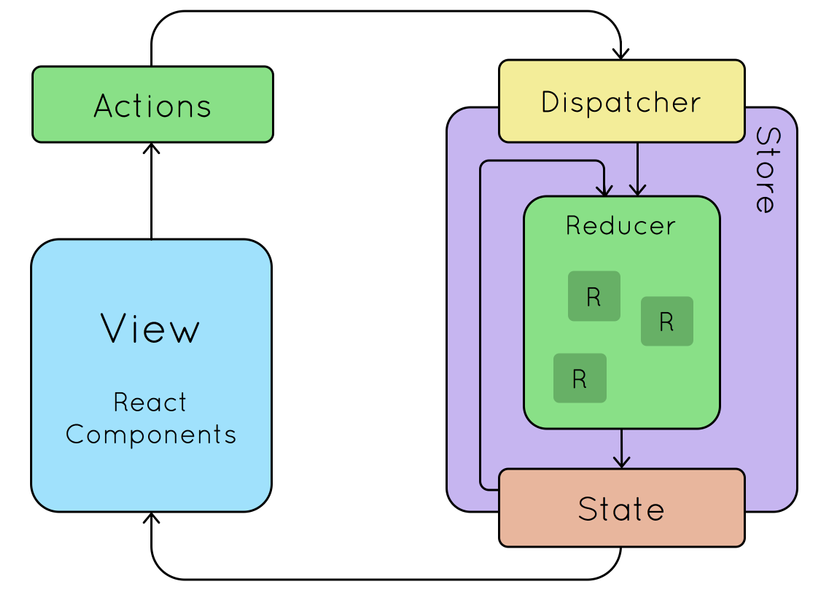
ReactJS sử dụng để tạo giao diện và quản lý cấu trúc cực tốt, và được đứng sau bởi một ông lớn trong ngành IT. Ngoài ra ReactJS được kế thừa một kho thư viện đồ sộ trên NPM. Không thể không kể đến Redux.



Hình 2. Hình ảnh Logo của ReactJS.

### 2.1.2. Redux là gì?

Redux là một thư viện Javascript của **ReactJS**, giúp tạo ra thành một lớp quản lý state của ứng dụng. Mặc định ReactJS cũng có thể quản lý state chung (**React Context**) nhưng cũng khó có thể thay cho Redux vì khả năng quản lý state rõ ràng của nó.



Hình 3. Hình ảnh mô tả vòng đời (life cycle) của Redux.

### 2.1.3. NextJS là gì?

NextJS được tạo ra bởi Vercel là một framework của ReactJS được xây dựng trên nền NodeJS (**Server Side Rendering SSR**) khác với ReactJS là **Client Side Rendering CSR** giúp tạo ra một trang viết bằng ReactJS nhưng được render phía Server, giúp tăng cải thiện SEO đối với một ứng dụng ReactJS.



Hình 4. Hình ảnh Logo của NextJS.

## 2.2. Đặc Tả Dự Án

Các chức năng và phân công công việc CDPTW1:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thành Viên** | **Nội Dung Được Phân Công** | **Hoàn Thành** |
| 1 | Ngô Bảy Hên | Đăng nhập,  Đăng ký,  Đăng xuất (1) | 100% |
| 2 | Trần Trọng Hiếu | Sửa confessions (3),  Hiển thị confessions,  Phân trang confessions (2) | 100% |
| 3 | Nguyễn Huỳnh Quốc Trung | Thêm confessions (3),  Xóa confessions (4) | 100% |

Các chức năng và phân công công việc CDPTW2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thành Viên** | **Nội Dung Được Phân Công** | **Thời Gian** | **Hoàn Thành** |
| 1 | Ngô Bảy Hên | Loading khi xử lý API  Trang chi tiết bài viết (bài viết liên quan) | 10 ngày | 100% |
| 2 | Trần Trọng Hiếu | Đa Ngôn Ngữ  Trang Profile người dùng | 14 ngày | 100% |
| 3 | Nguyễn Huỳnh Quốc Trung | Tìm kiếm bài viết  Trang chi tiết bài viết (nội dung) | 10 ngày | 100% |

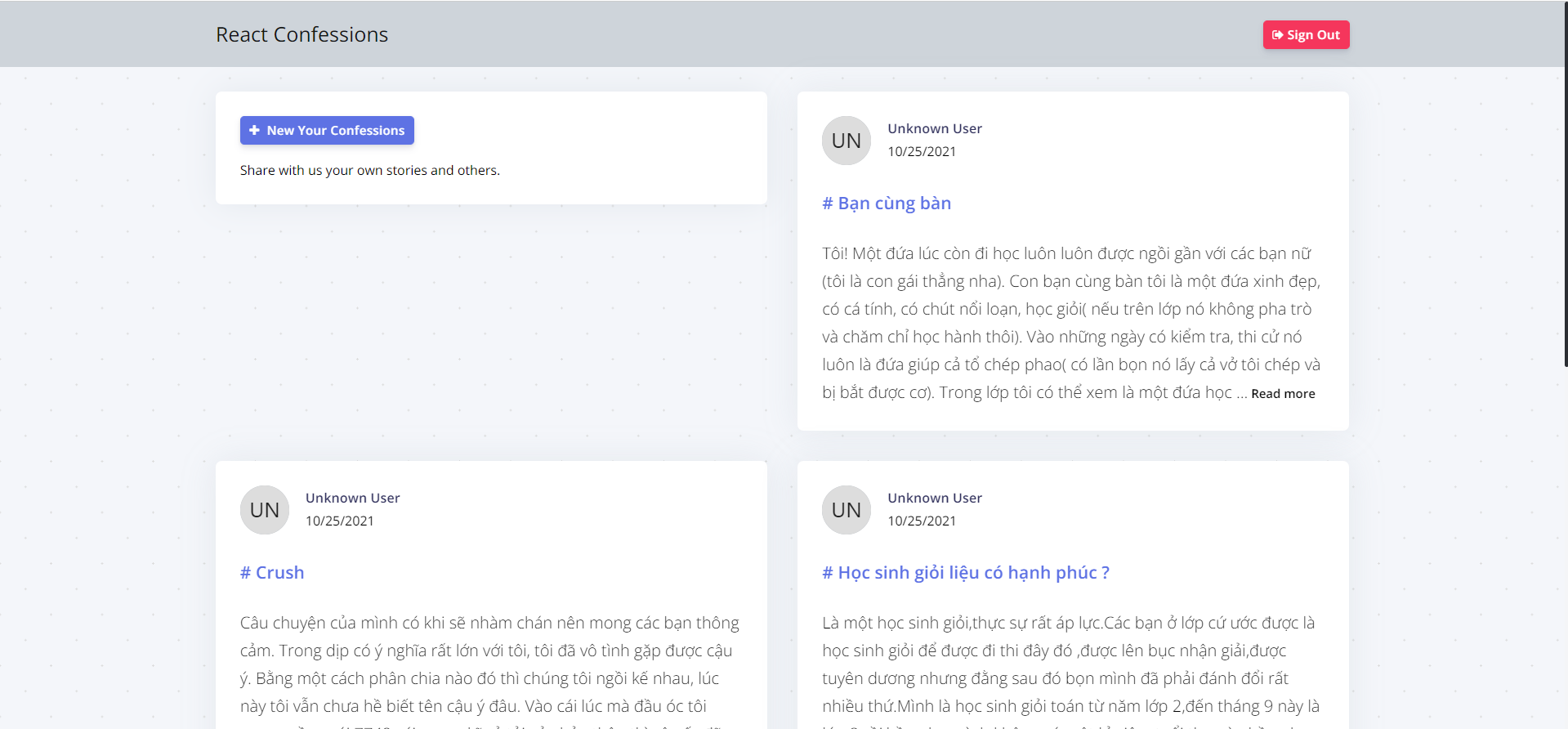
Các chức năng và phân công công việc CDPTW2 (BCKT):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Thành Viên** | **Nội Dung Được Phân Công** | **Bắt Đầu** | **Kết Thúc** | **Hoàn Thành** |
| 1 | Ngô Bảy Hên | Trang chi tiết bài viết (bài viết liên quan) (12) |  |  | 100% |
| 2 | Trần Trọng Hiếu | Đa Ngôn Ngữ (14) |  |  | 100% |
| 3 | Nguyễn Huỳnh Quốc Trung | Trang chi tiết bài viết (nội dung) (13) |  |  | 100% |

### 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 5. 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.

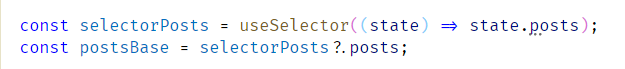


Hình 6. Gọi useEffect để dispatch actions dữ liệu.

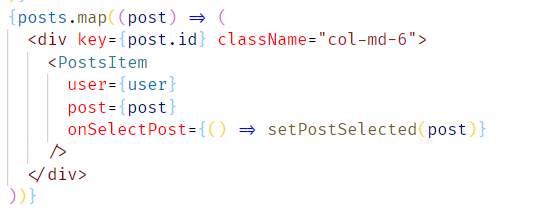


Hình 7. Gọi API và đưa vào reducers của Redux.

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

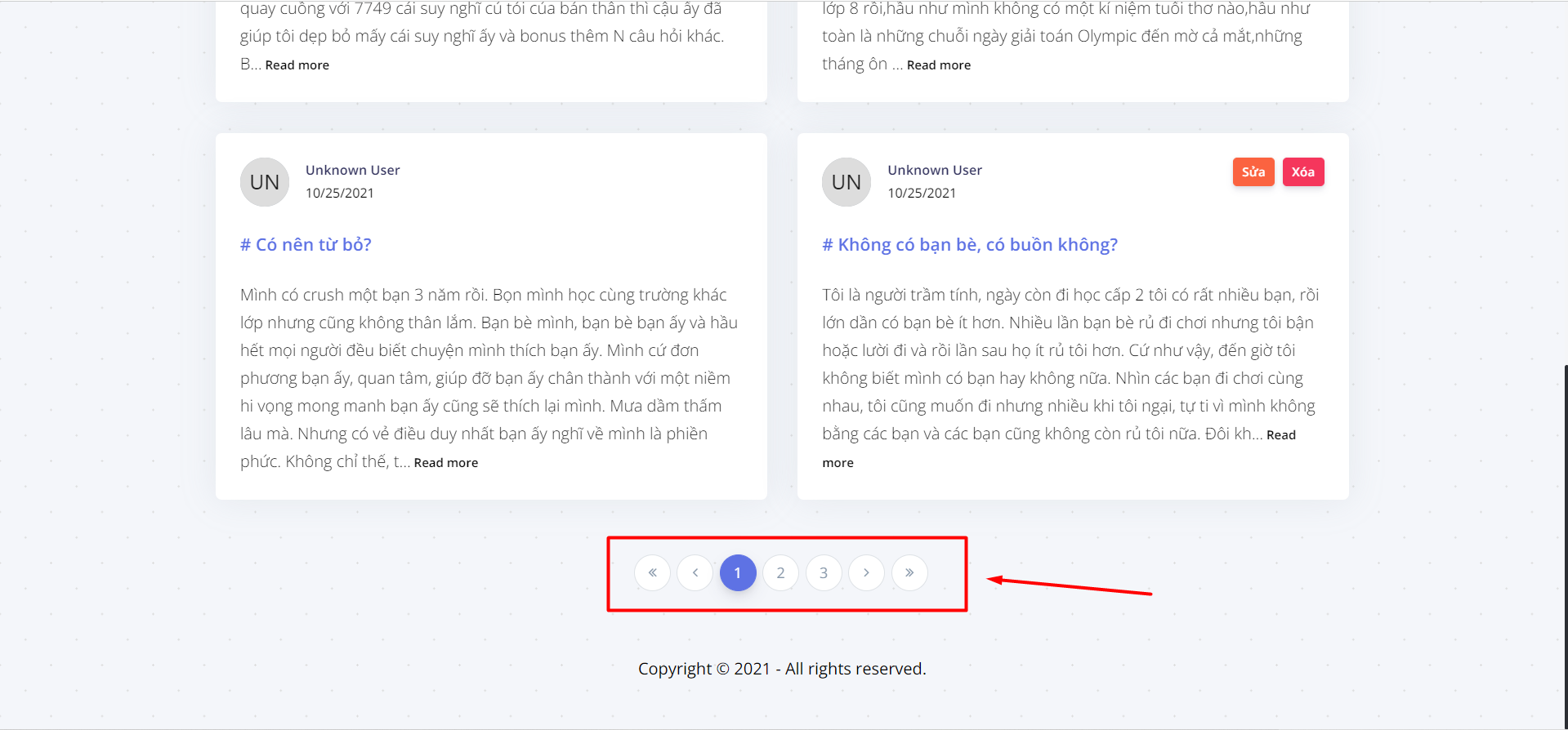


Hình 9. Hiển thị dữ liệu bằng cách map qua mảng dữ liệu.

### 2.2.2. Phân trang hiển thị confessions

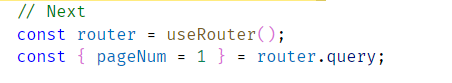
Hiển thị phân trang ở phía dưới của bài viết, giao diện bao gồm:

* 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 10. 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.



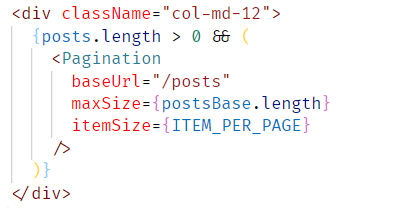
Hình 11. Lấy số trang hiện tại bằng useRouter, mặc định là 1.

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

Hiển thị Pagination ở trang chủ bằng components Pagination.

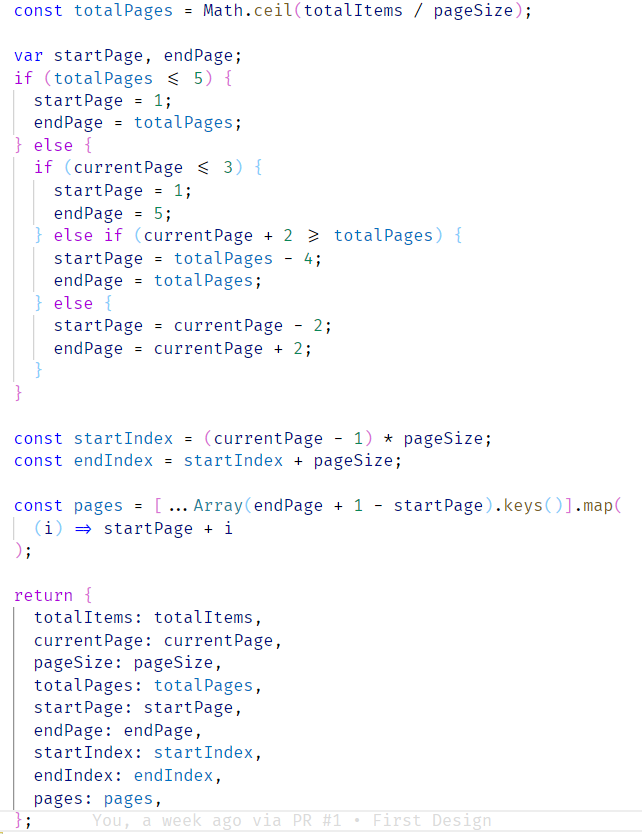


Hình 13. Hiển thị pagination khi số lượng bài viết lớn hơn 0..

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



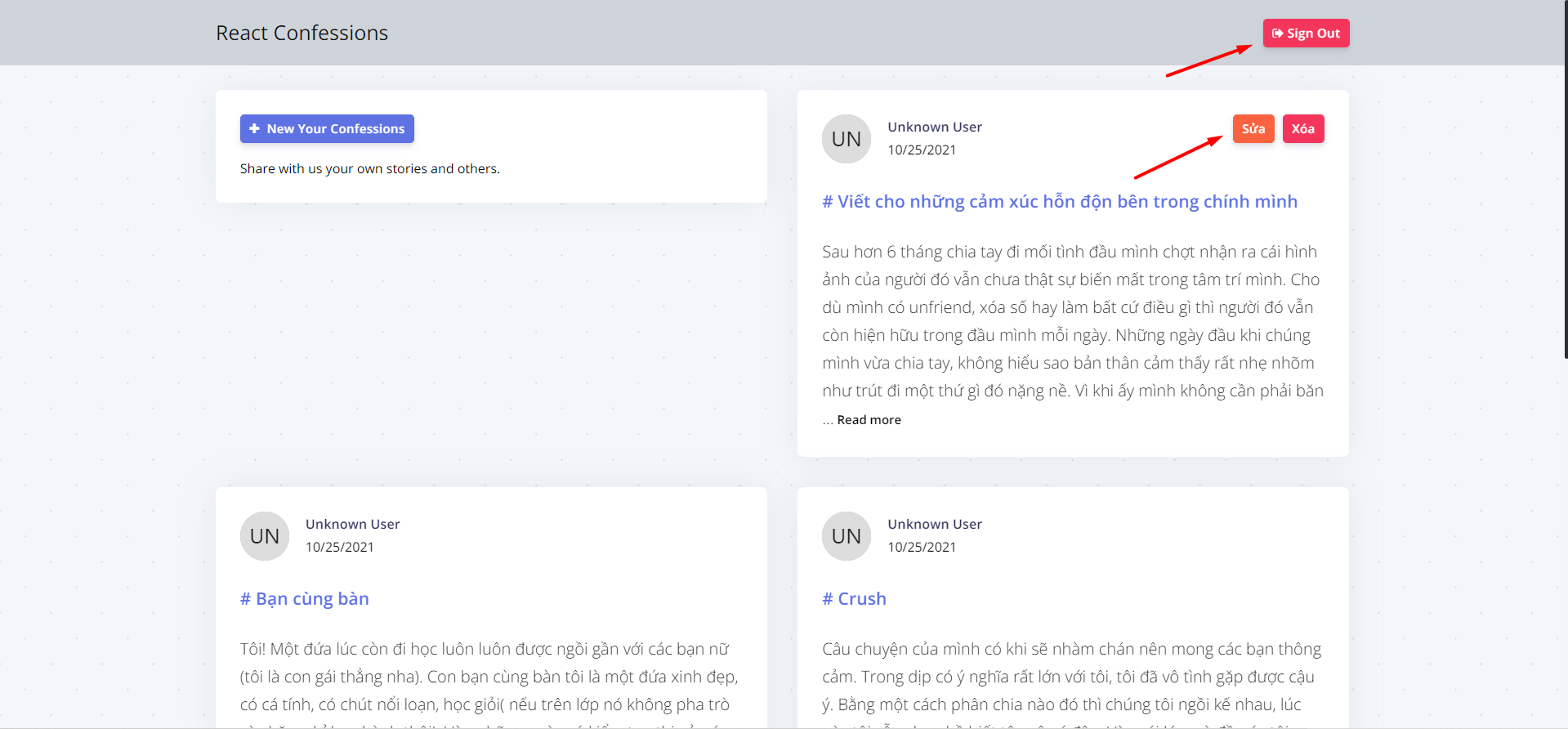
Hình 14. Hiển thị Pagination cho bài viết.



Hình 15. 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.

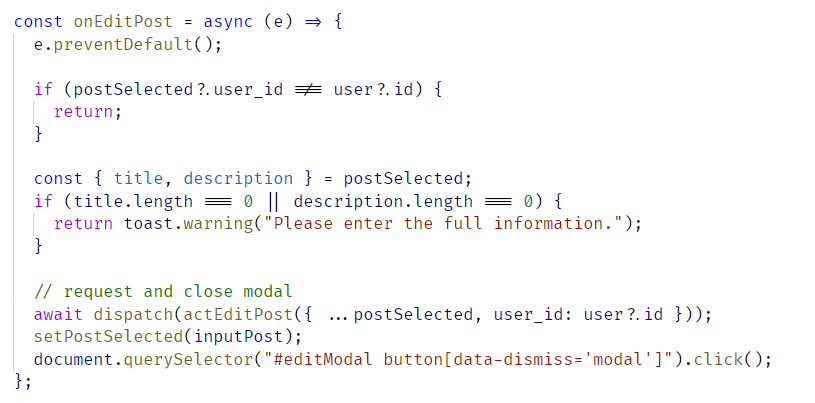


Hình 16. Hiển thị sửa khi người dùng đăng nhập và đúng bài viết.



Hình 17. 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 18. Khi click vào nút sửa sẽ gọi đến function sửa.

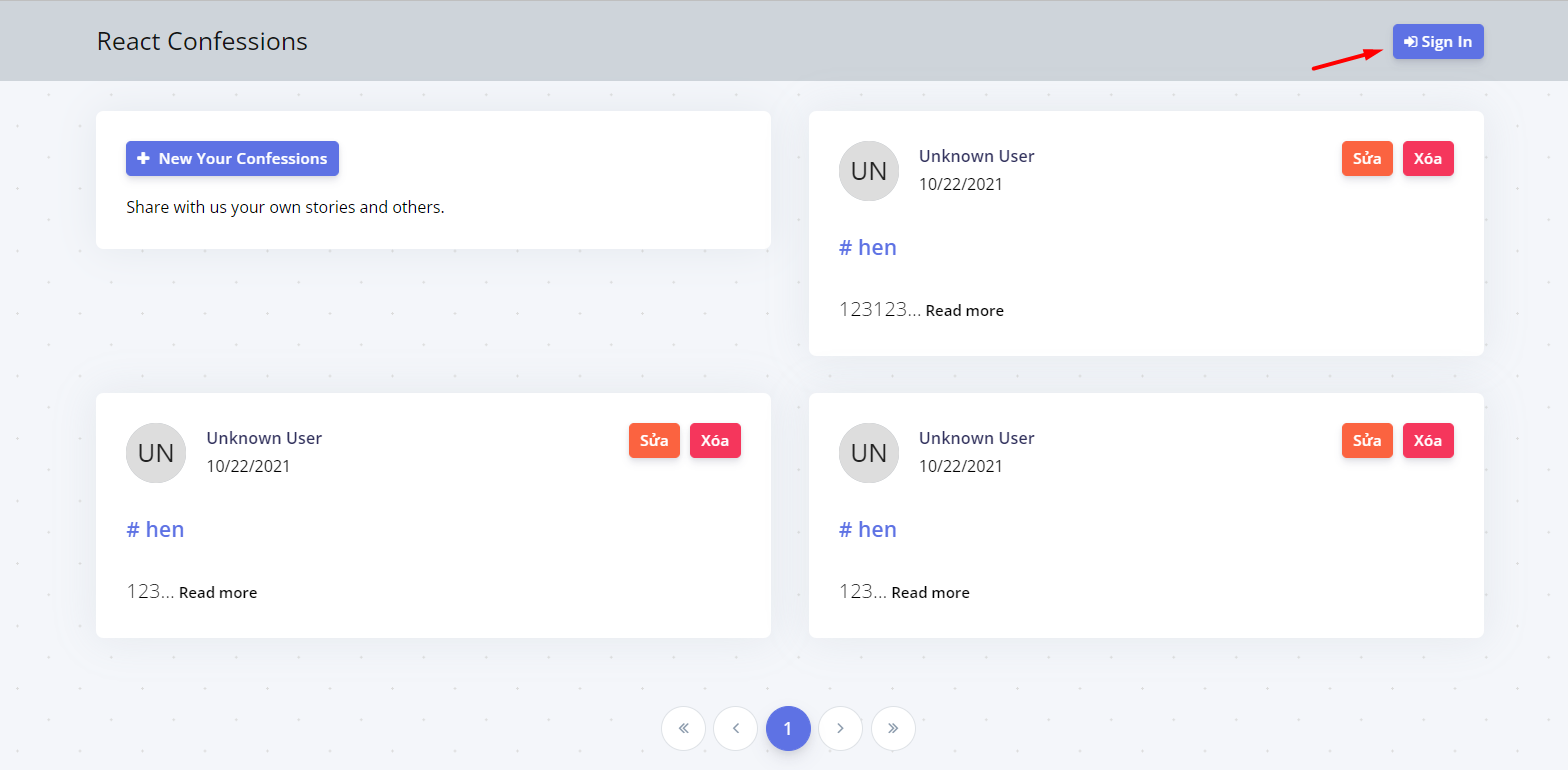
Ở trong Redux sẽ xứ lí chức năng sửa bài viết, và load lại tất cả bài viết hiện tại.



Hình 19. Xử lí sửa bài viết và load lại dữ liệu.

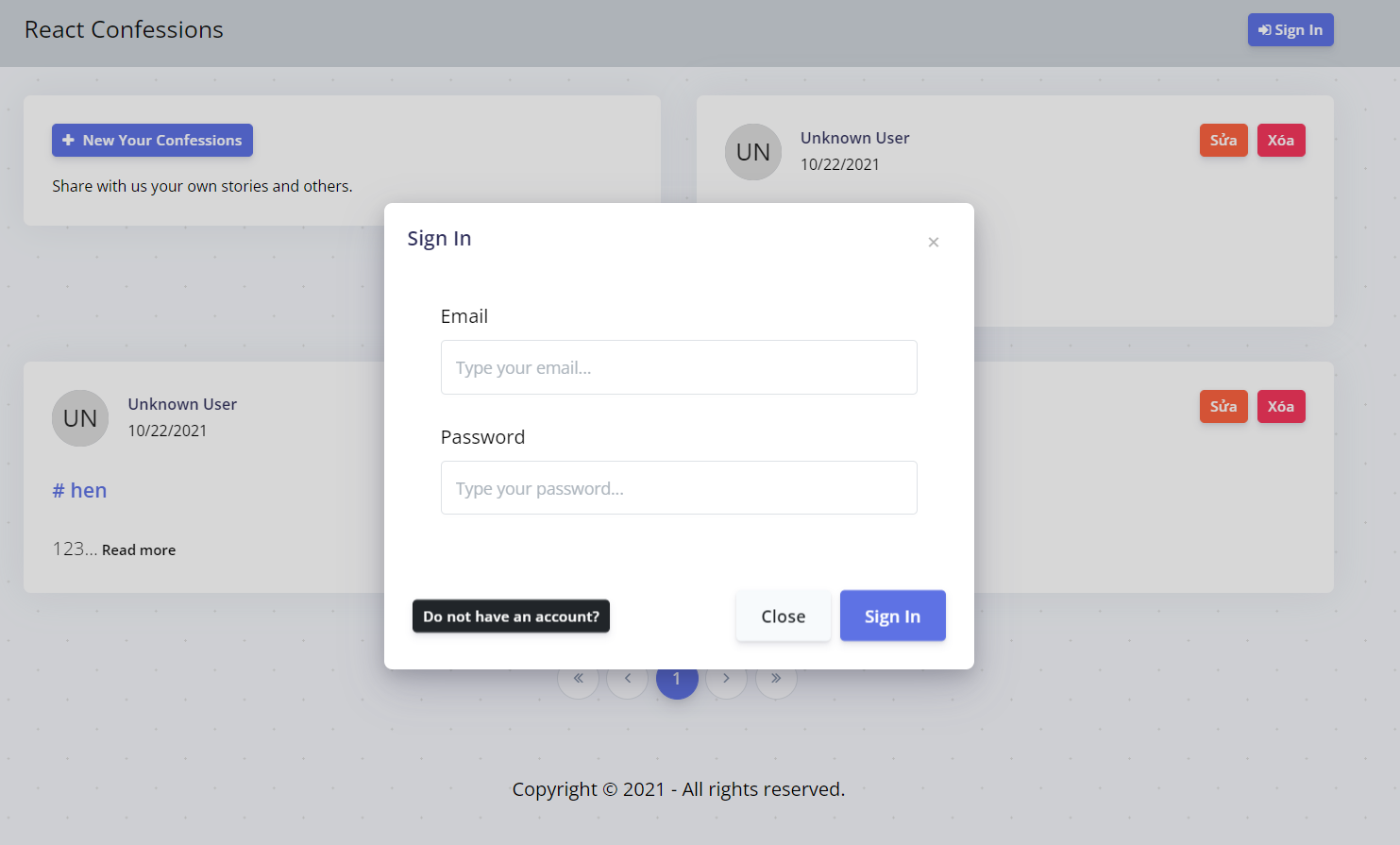
### 2.2.4. Đăng nhập confessions

Ở giao diện chính của web confessions ở trên cùng bên phải của web sẽ có một button Sign In để cho người dùng có thể đăng ký tài khoản.



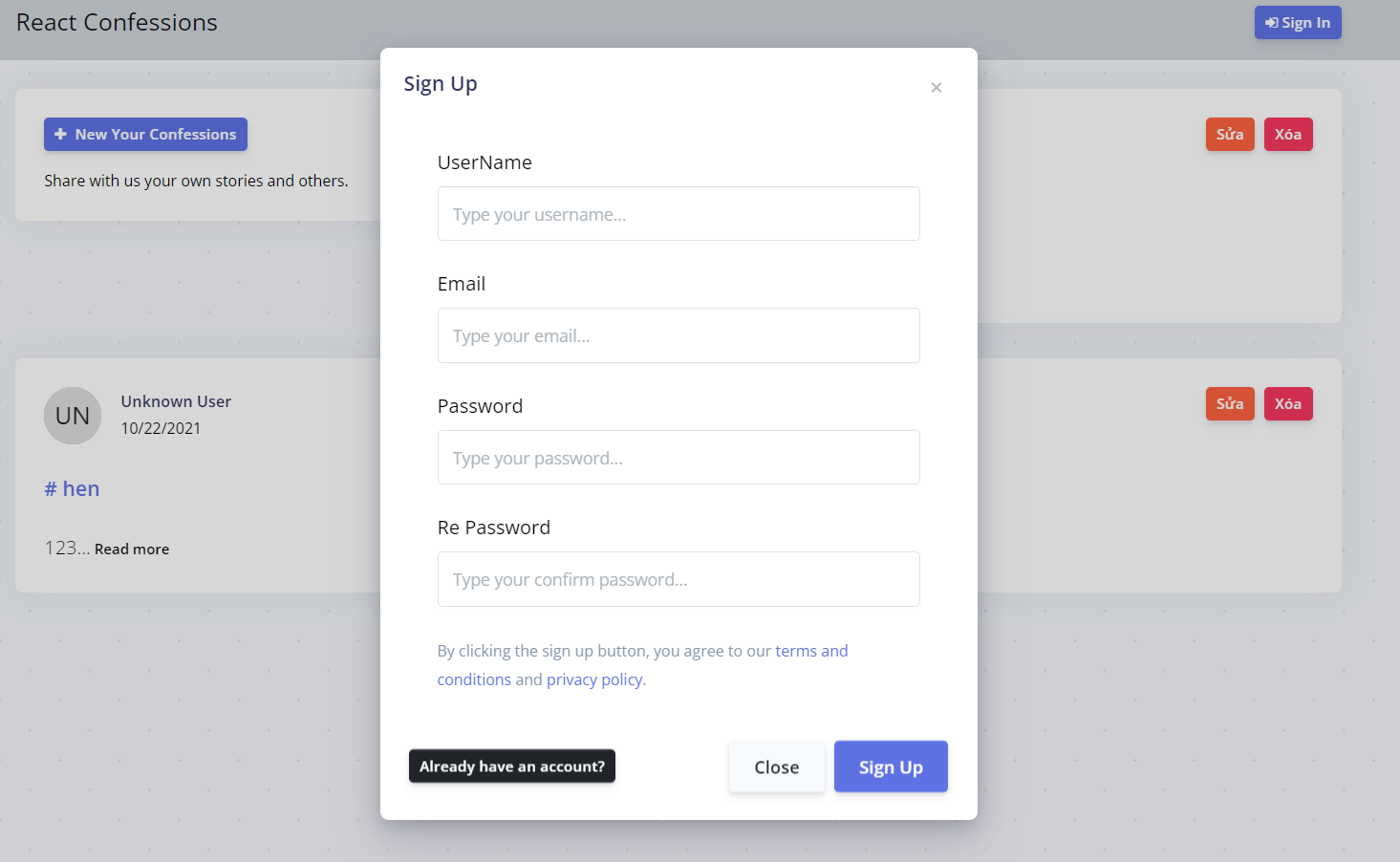
Hình 20. Nơi hiển thị button Sign In.

Khi click vào button Sign In web sẽ hiện lên cho người dùng một model để có thể đăng nhập hoặc đăng ký tài khoản.



Hình 21. Hiện model để đăng nhập.

Ở đây khi người dùng chưa có tài khoản thì học sẽ click vào button “Do not have an account” để web xử lý là hiện ra model đăng ký để người dùng có thể đăng ký tài khoản. Ở model này nếu người dùng có tài khoản thì chỉ cần nhấn button “Already have an account ? ” để có thể đăng nhập.



Hình 22. Hiện model để đăng ký.

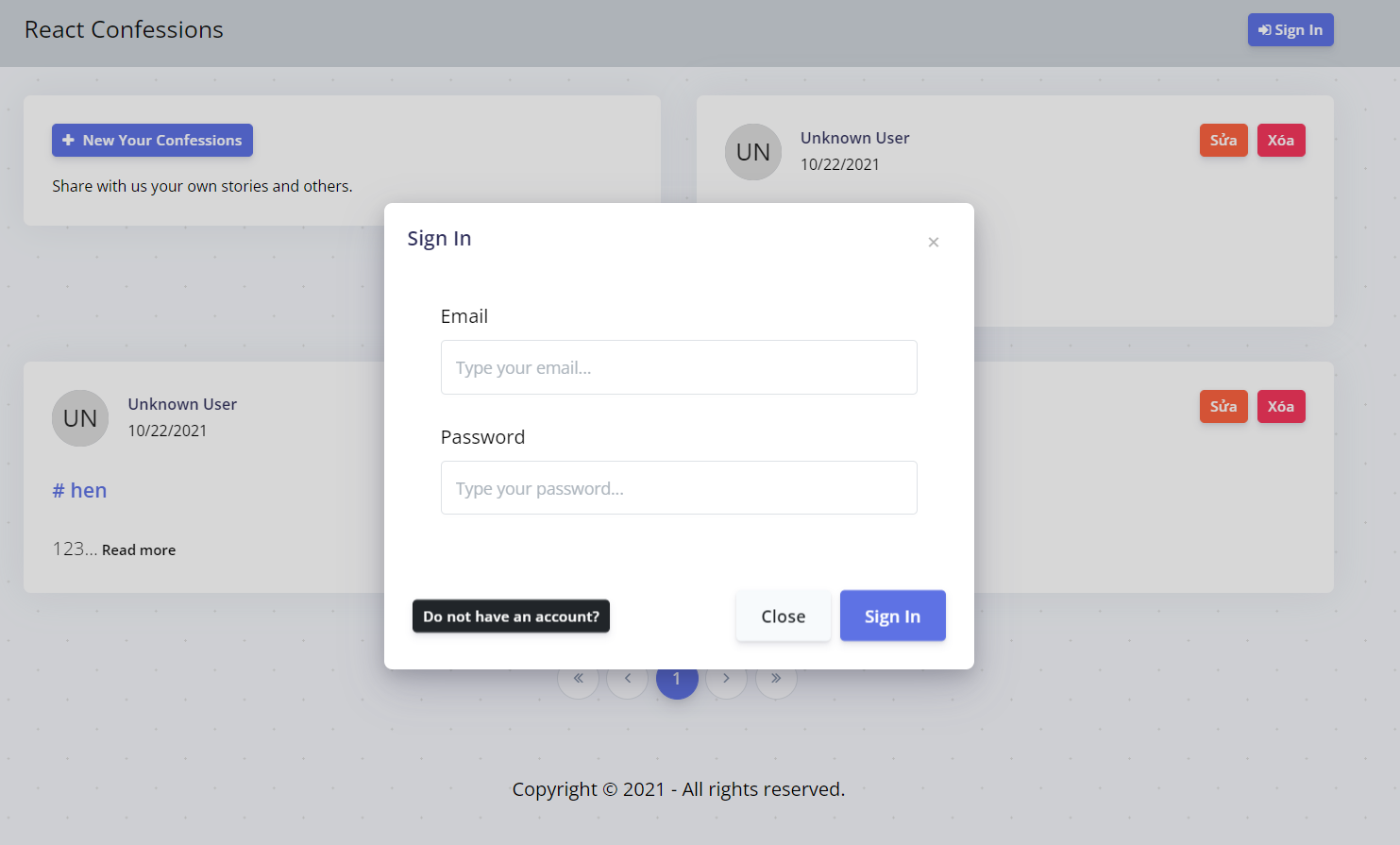
Ở model này người dùng chỉ nhập đầy đủ và đúng thông tin và nhấn button Sign Up web sẽ vào Redux và callApi để thực hiện chức năng đăng ký và thông báo kết quả. Nếu đăng ký thành công thì sẽ web sẽ navigation qua model Sign In.



Hình 23. Hàm Sign Up action ở Redux.

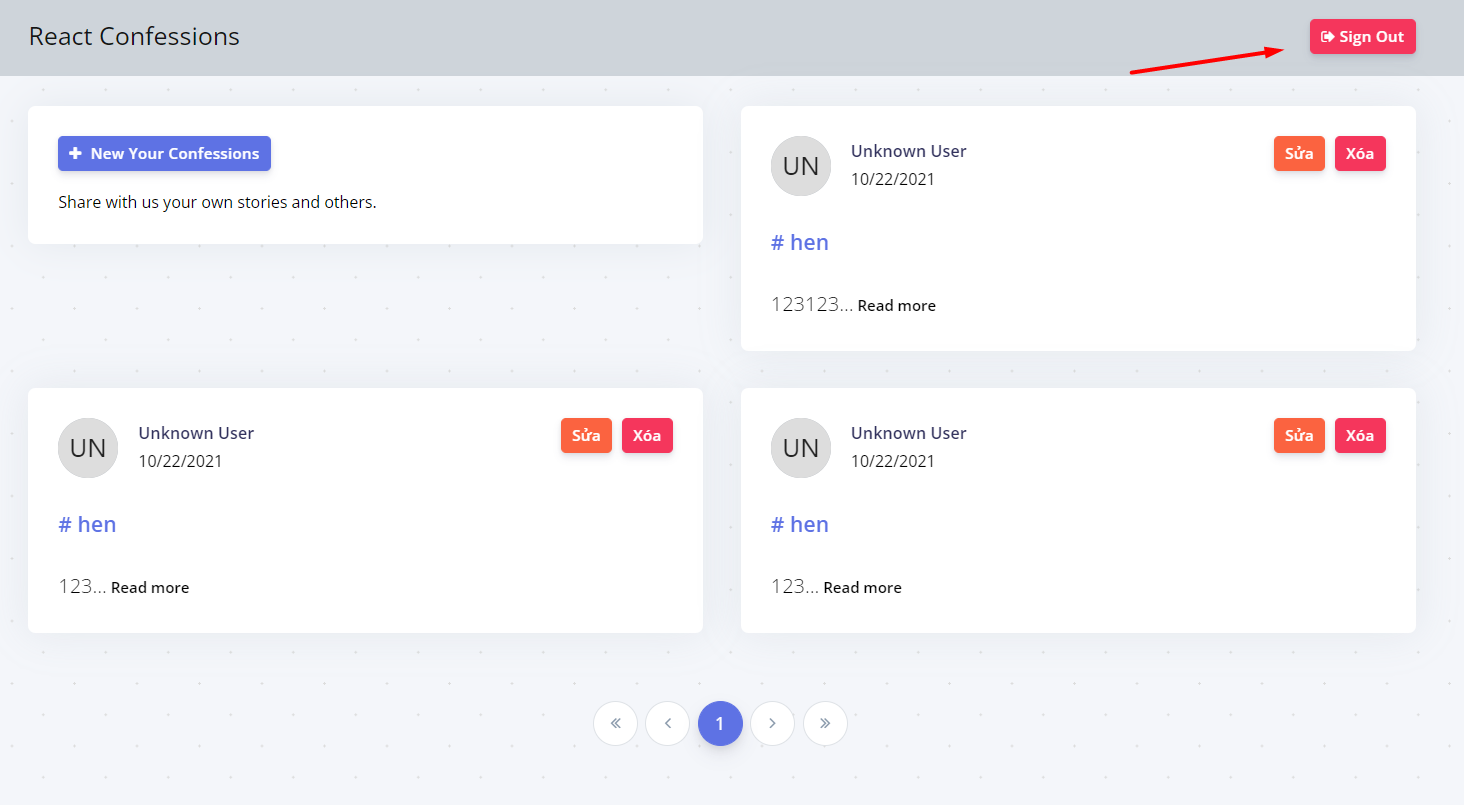
### 2.2.5. Đăng ký confessions

Ở model đăng nhập sau khi click vào button Sign In để hiện ra thì người dùng nếu đã có tài khoản thì chỉ cần đăng nhập điền đầy đủ thông tin và bấm button Sign In để đăng nhập.



Hình 24. Model Sign In.

Ở đây khi người dùng điền thông tin đầy đủ và nhấn button Sign In thì web sẽ vào Redux và callApi để xử lý đăng nhập nếu đăng nhập thành công sẽ đóng model và thông báo đăng nhập thành công chuyển button Sign In thành Sign Out.



Hình 25. Sau đăng nhập thành công.

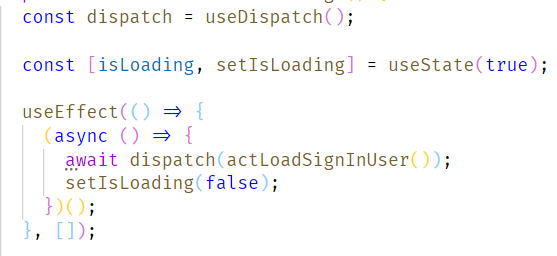


Hình 26. Hàm Redux xử lý đăng nhập.

Sau khi chuyển button thành Sign Out ta phải ghi thêm hàm loading để có thể lưu được trạng thái đăng nhập.



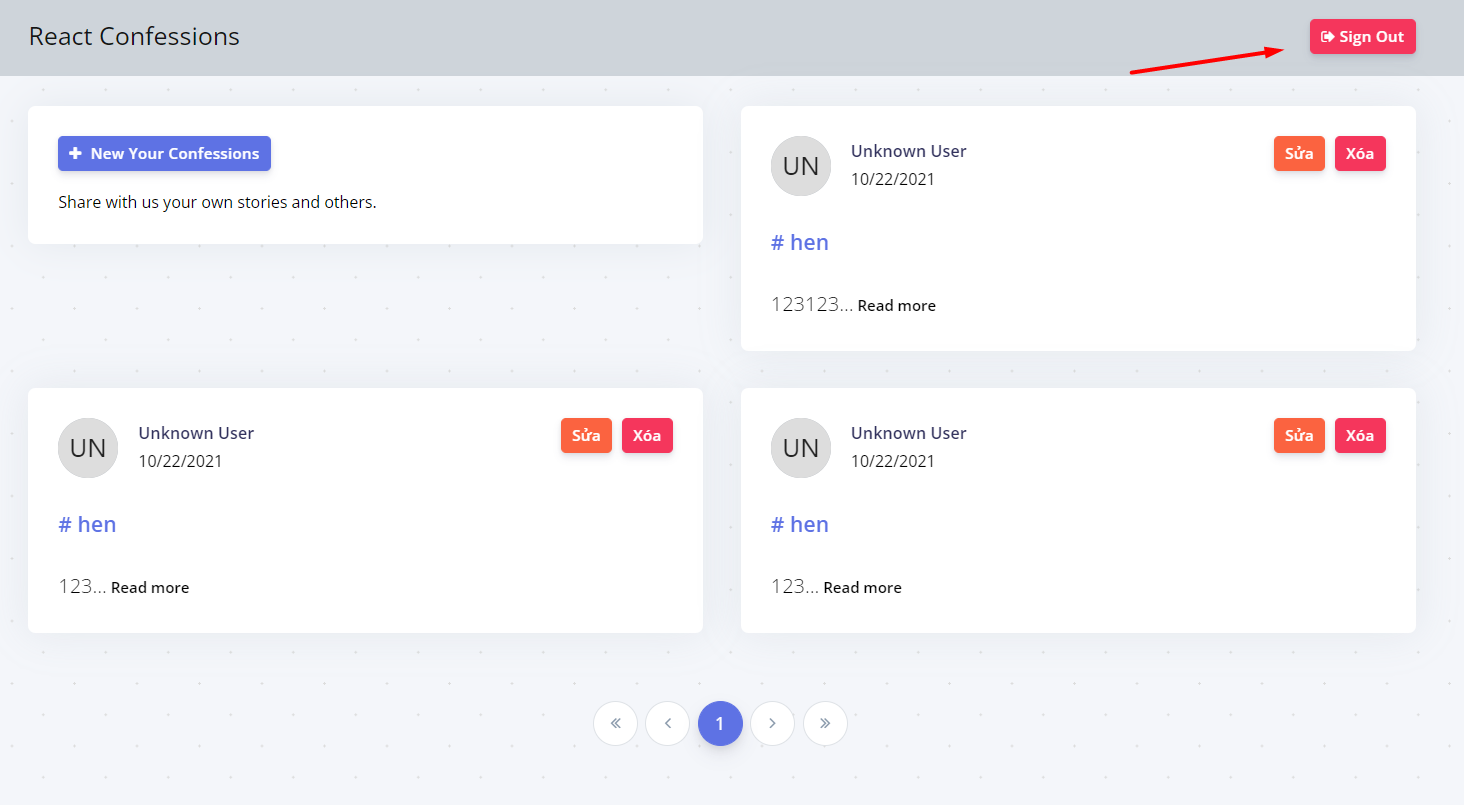
Hình 27. Hàm sẽ thực hiện khi người dùng reload trang.



Hình 28. Hàm Redux loading đăng nhập.

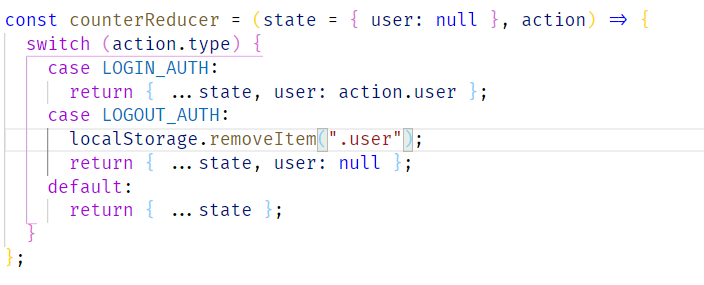
### 2.2.6. Đăng xuất confessions

Khi người dùng đã đăng nhập và muốn Sign out tài khoản của mình ra ngoài thì người dùng sẽ click vào nút button Sign Out để có thể đăng xuất tài khoản ra khỏi web confessions.

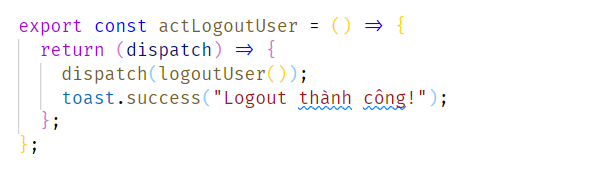


Hình 29. Hiện button Sign Out.

Sau khi click vào button Sign Out thì token ở LocalStorage sẽ bị xóa và button sẽ chuyển về lại thành Sign In và để thực hiện được chuyện đó chúng ta sẽ viết hàm ở Redux tạo action để có thể xóa token ở LocalStorage.



Hình 30. Ở hàm authReducer.

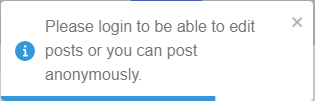


Hình 31. Hàm Logout ở Redux.

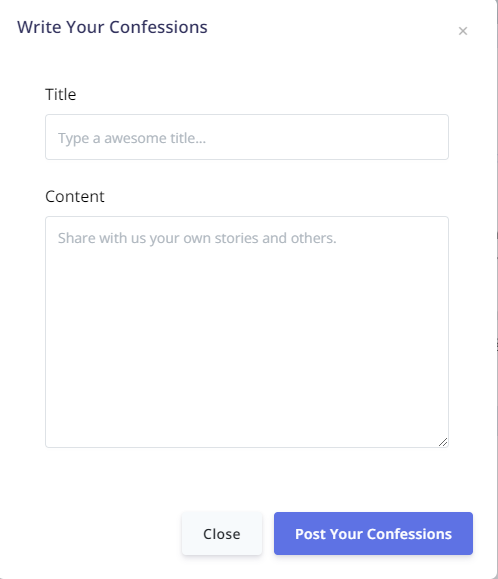
### 2.2.7. Thêm confessions

Khi người dùng Click vào button “New Your Confessions” thì một model sẽ hiện ra cho phép người dùng thêm bài viết.

Trường hợp không đăng nhập thì sẽ có thông báo:

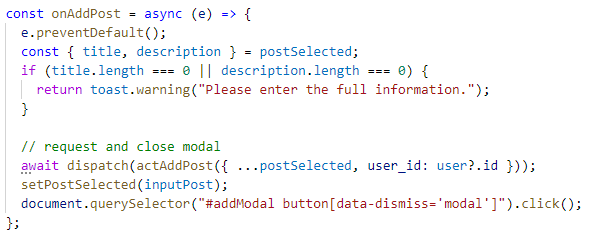


Hình 32. HIện thông báo.



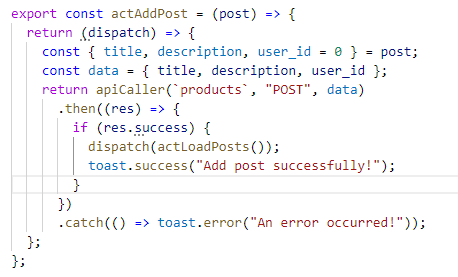
Hình 33. Model Thêm Confessions.

Khi click vào nút “Post Your Confessions”sẻ gọi đến function thêm. Đầu tiên check đang input hiện tại có rỗng hay không. Tiếp tục gọi dispatch actions thêm bài viết của Redux, sau đó đóng model lại.



Hình 34. Khi click vào nút thêm sẽ gọi đến function thêm.

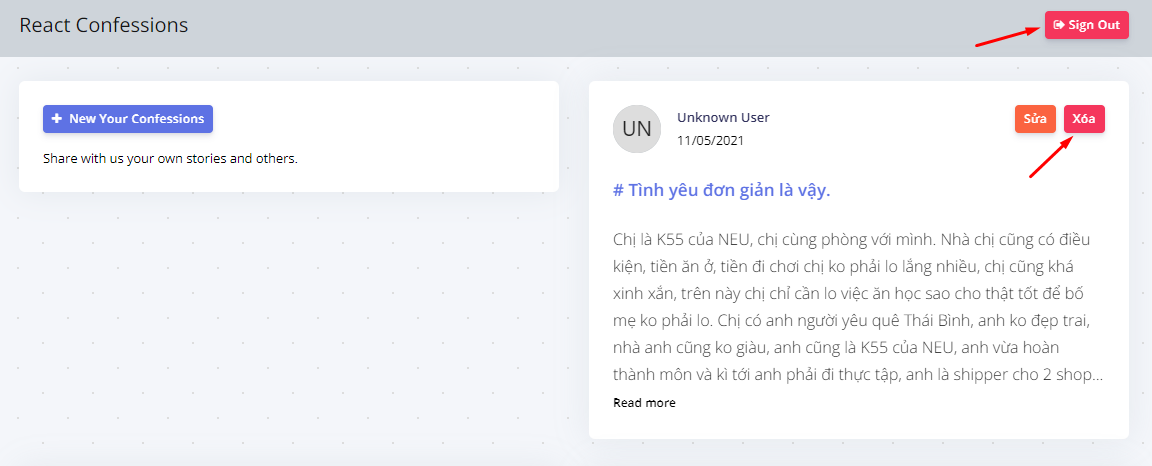
Ở trong Redux sẽ xử lý chức năng thêm bài viết, và load lại tất cả bài viết hiện tại.



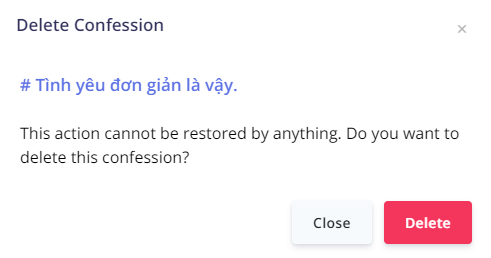
Hình 35. Xử lý thêm bài viết và load lại dữ liệu.

### 2.2.8. Xóa confessions

Giao diện xóa 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 xóa bài viết.

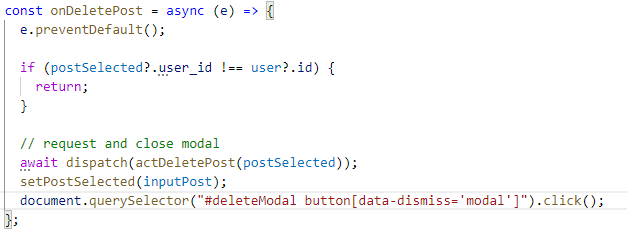


Hình 36. Hiển thị xóa khi người dùng đăng nhập và đúng bài viết.



Hình 37. Hiển thị Model xóa khi người dùng click vào nút xóa.

Khi click vào nút xóa sẻ gọi đến function xóa, 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 xóa bài viết của Redux, sau đó đóng modal lại.



Hình 38. Khi click vào nút xóa sẽ gọi đến function xóa.

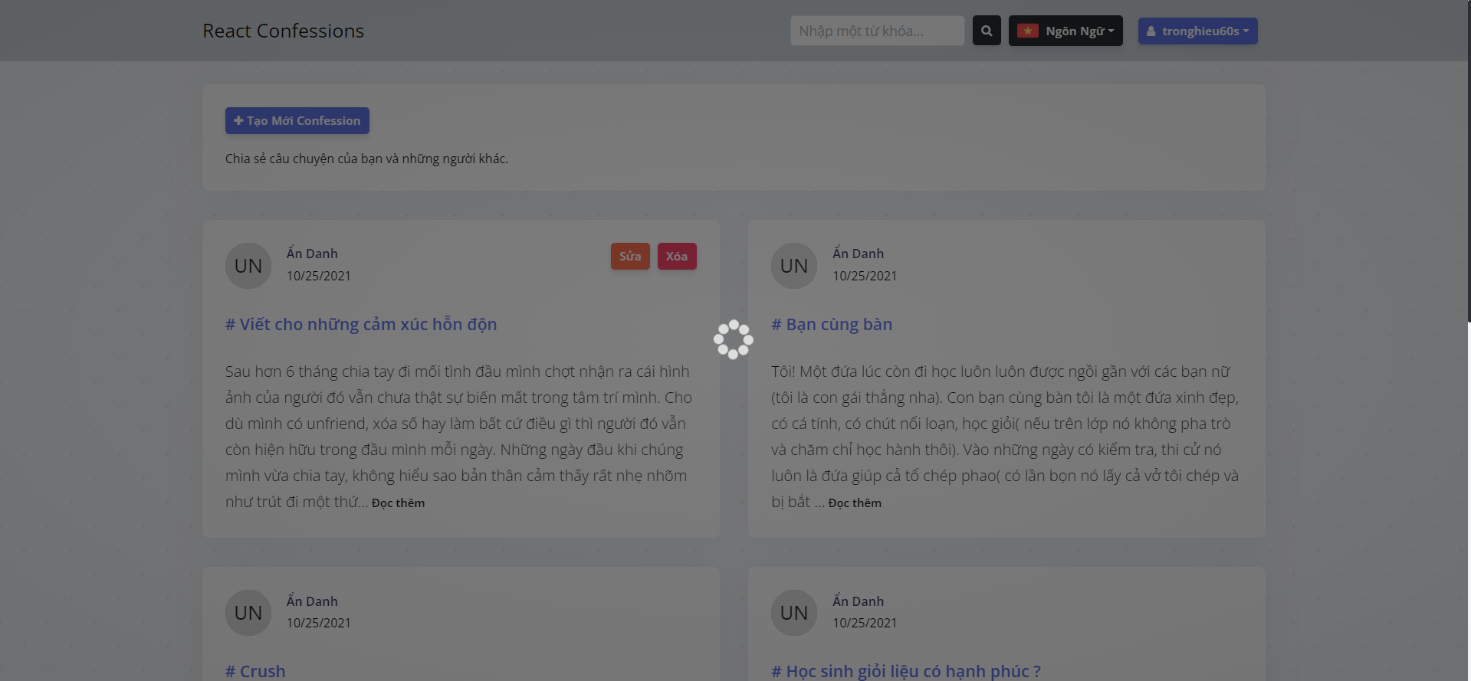
Ở trong Redux sẽ xử lý chức năng xóa bài viết, và load lại tất cả bài viết hiện tại.



Hình 39. Xử lý xóa bài viết và load lại dữ liệu.

### 2.2.9. Loading khi xử lý

Loading sẽ được hiện lên khi người dùng có hành động xử lý và gọi đến API của ứng dụng, tăng UI/UX người dùng.



Hình 40. Giao diện loading.

Để có thể xử lý được loading ở đây chúng ta sẽ dùng Redux để xử lý đầu tiên chúng ta vào commonReducer để set state loading bằng false và tạo 2 case action START\_LOADING VÀ STOP\_LOADING để có thể thay đổi giá trị của loading khi chúng ta gọi làm để chạy action.



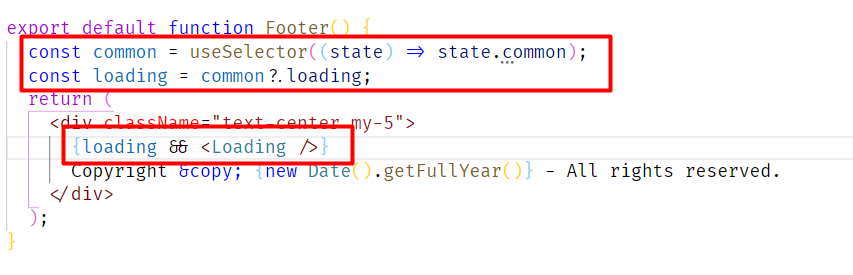
Hình 41. Tạo 2 case action.

Sau khi đã thêm xong commonReducer thì chúng ta sẽ đem xử lý này để gắn vào những nơi có gọi API để hiển thị loading trong thời gian gọi API



Hình 42. Dispatch action trong một function xử lý khác.

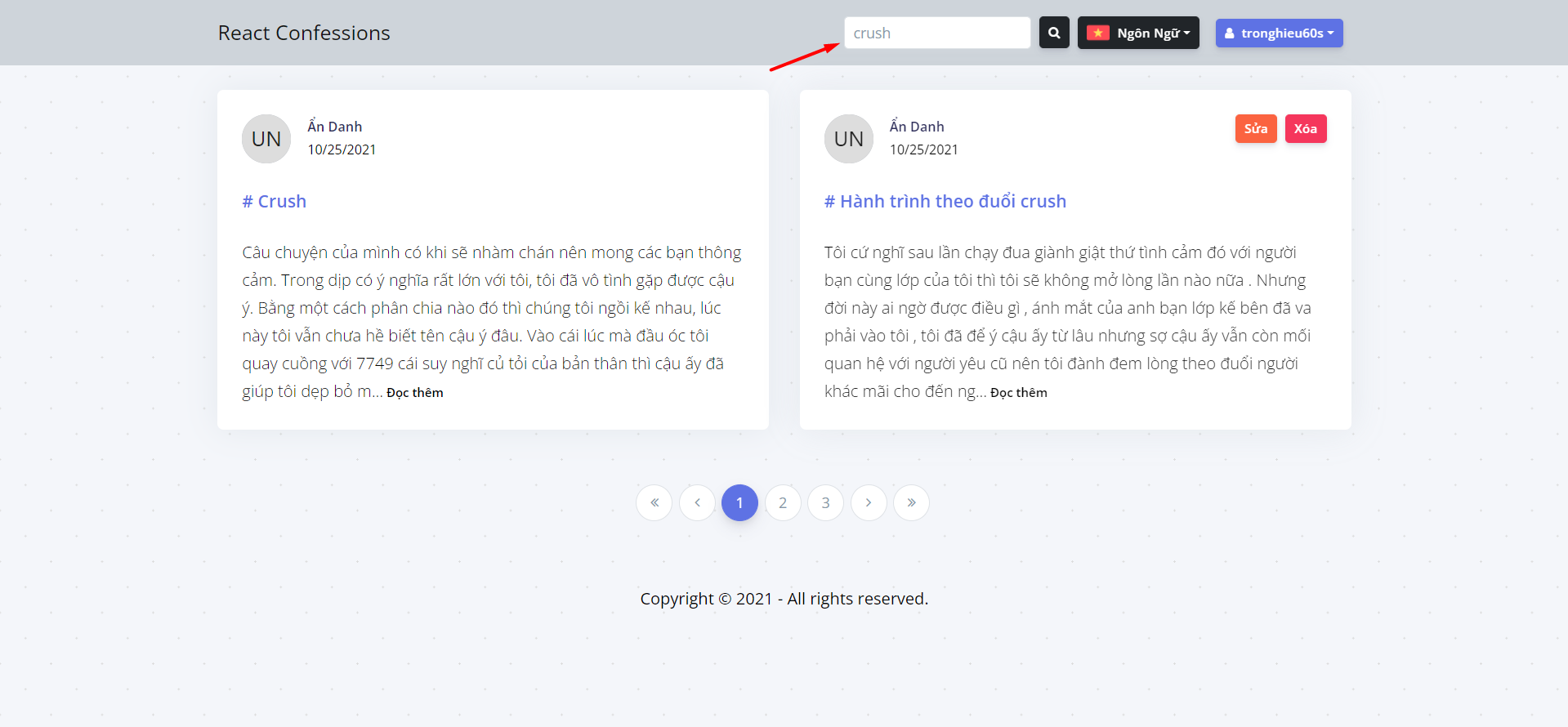
Khi đã gắn vào trong hàm call API thì sau đó chúng tra sẽ ghi làm để set trạng thái cho loading là gì để có thể hiển thị ra màn hình



Hình 43. Gọi giá trị loading từ Redux và sử dụng.

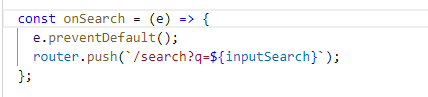
### 2.2.10. Tìm kiếm bài viết

Người dùng có thể tìm kiếm được cái bài viết trong ứng dụng nhanh chóng.



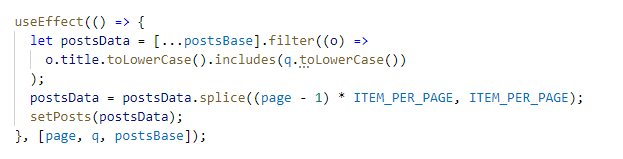
Hình 44. Giao diện sau khi tìm kiếm.

Khai báo nút Search:



Hình 45. Khai báo onSearch.

Khi Search thì màn hình sẽ chỉ hiển thị những mục mà ta tìm kiếm:



Hình 46. useEffect hiển thị mục tìm kiêm.

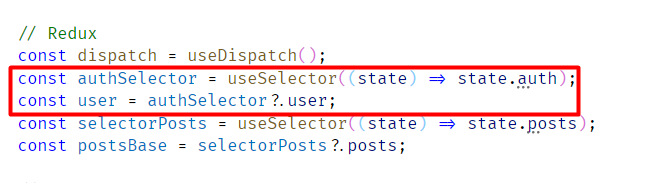
### 2.2.11. Trang chi tiết người dùng

Ở trang chi tiết người dùng, phần bên trái là hiển thị danh sách bài viết, cũng giống như trang chủ, ta chỉ cần lọc đi phần hiển thị cho người dùng hiện tại.



Hình 47. Lọc các bài viết của người dùng đã tạo.

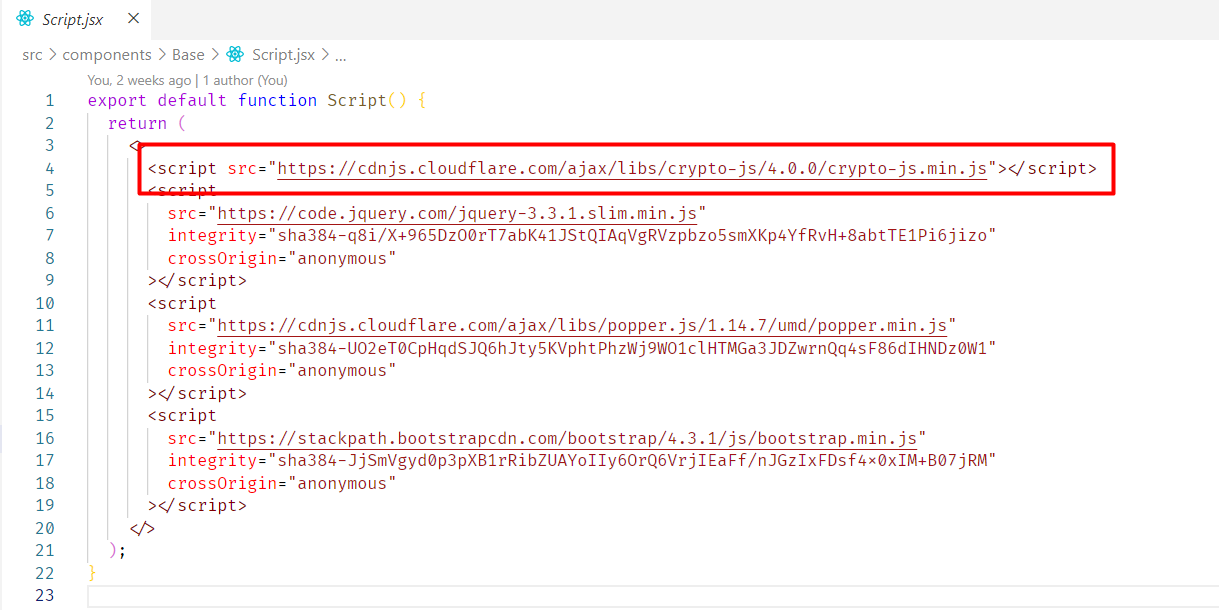
Để hiển thị thông tin cá nhân người dùng ra ta chỉ cần gọi redux mà đã lưu người dùng đăng nhập trước đó.



Hình 48. Sử dụng redux mà trước đó đã lưu thông tin người dùng.

Ảnh đại diện được sử dụng thông qua gravatar, ta chỉ cần lấy thông tin email người dùng hash md5 để lấy đường dẩn hình ảnh.

Để mã hóa md5, hãy nhúng một thư viện khác.



Hình 49. Nhúng thư viện mã hóa md5 vào dự án.

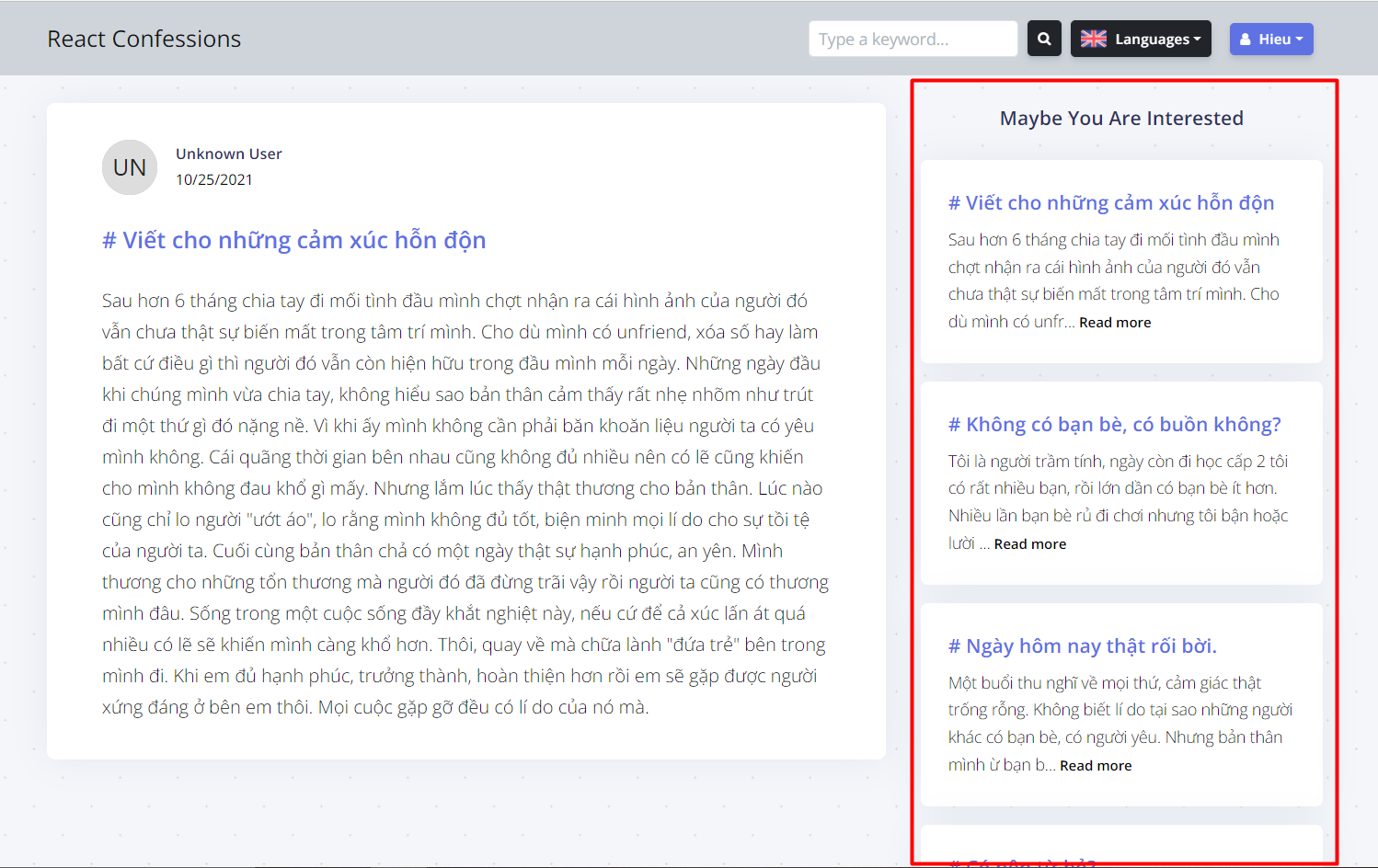
Sau đó ta sẽ hash email người dùng bằng useEffect hook, vậy là đã có đường dẫn hình ảnh rồi, chỉ cần hiển thị ra là xong.



Hình 50. Hiển thị hình ảnh người dùng sau khi hash ra mã md5.

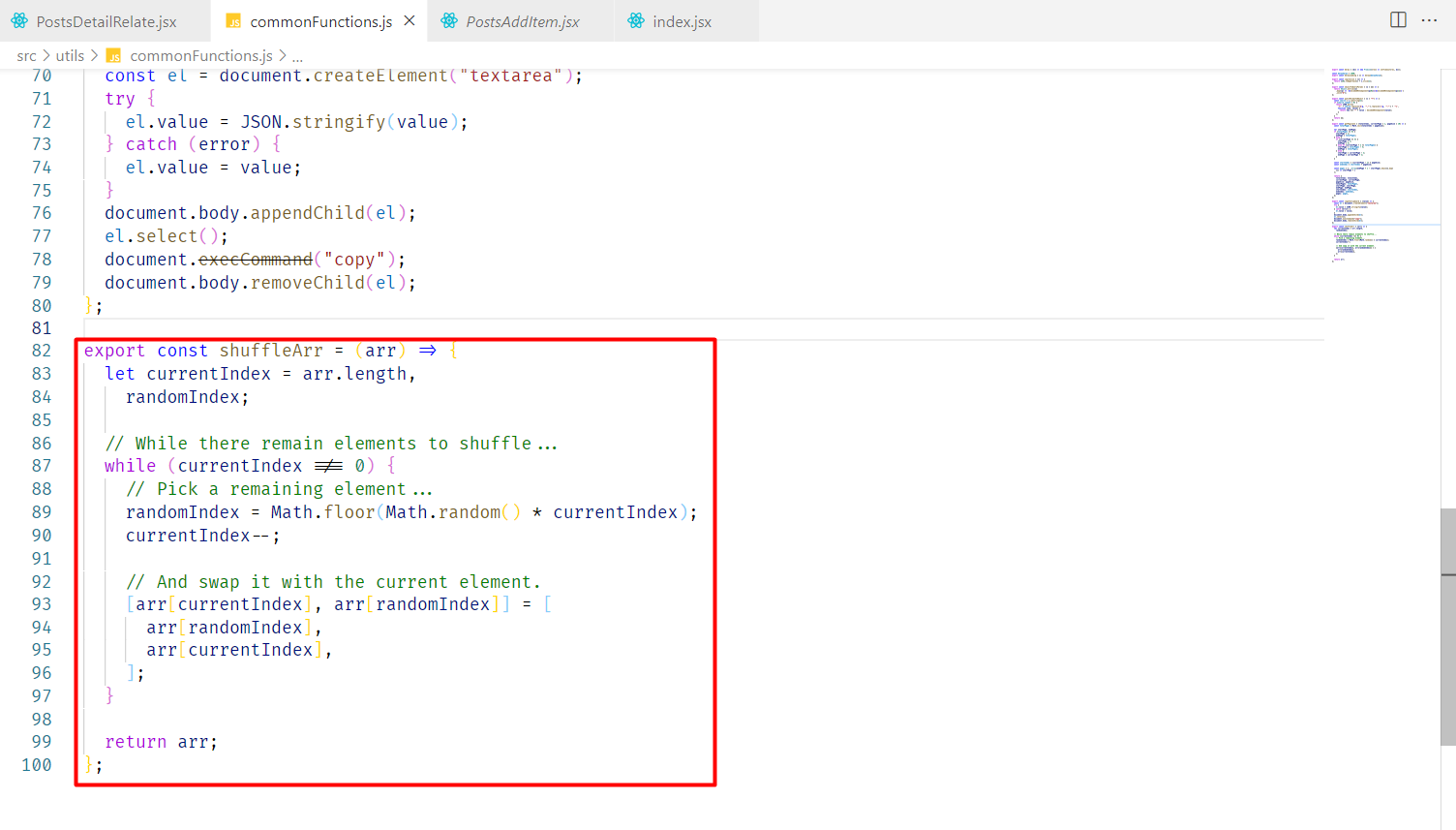
### 2.2.12. Trang chi tiết bài viết (bài viết liên quan)

Khi người dùng click vào một bài viết để xem nội dung bài viết thì ở page này chúng ta sẽ có thêm một cột bài viết liên quan để có thể gợi ý thêm cho người đọc những bài viết khác.



Hình . Giao diện hiển thị của bài viết liên quan.

Để làm bài viết liên quan đầu tiên ta phải viết một function để trộn thứ tự hiển thị của các bài viết.



Hình . Function trộn thứ tự của các bài viết.

Sau khi tạo xong hàm trộn các bài viết thì chúng ta sẽ khai báo useState và dùng Redux và useEffect để có thể sử dụng và hiển thị nó ra màn hình.



Hình . Khai báo và gọi hàm bài viết liên quan.

Sau cùng khi đã gọi hàm, khai báo và xử lý ở thằng cha xong việc cuối cùng là mình sẽ cho thằng con kế thừa thằng cha là mọi chuyện đã xong.



Hình . Kế thừa từ thằng cha của bài viết liên quan.

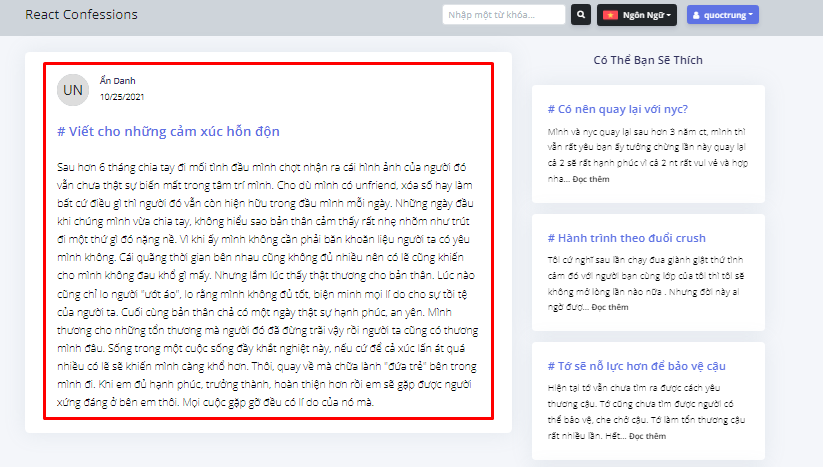
### 2.2.13. Trang chi tiết bài viết (nội dung)

Muốn xem đầy đủ nội dung bài viết nào thì ta cần click vào tiêu đề của bài viết đó:



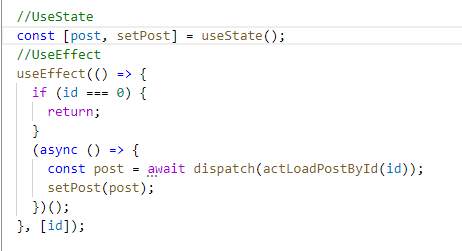
Hình . Click vào tiêu đề bài viết.

Sau khi click vào tiêu đề, trang sẽ chuyển sang giao diện mới:



Hình . Giao diện trang chi tiết bài viết.

Để hiện thị ra, ta cần dùng useEffect và useState:



Hình . useEffect và UseState trang chi tiết bài viết.

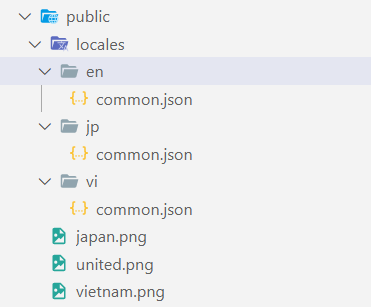
Sau đó gọi qua để hiển thị là xong:



Hình . HIển thị thông tin trong trang chi tiết bài viết.

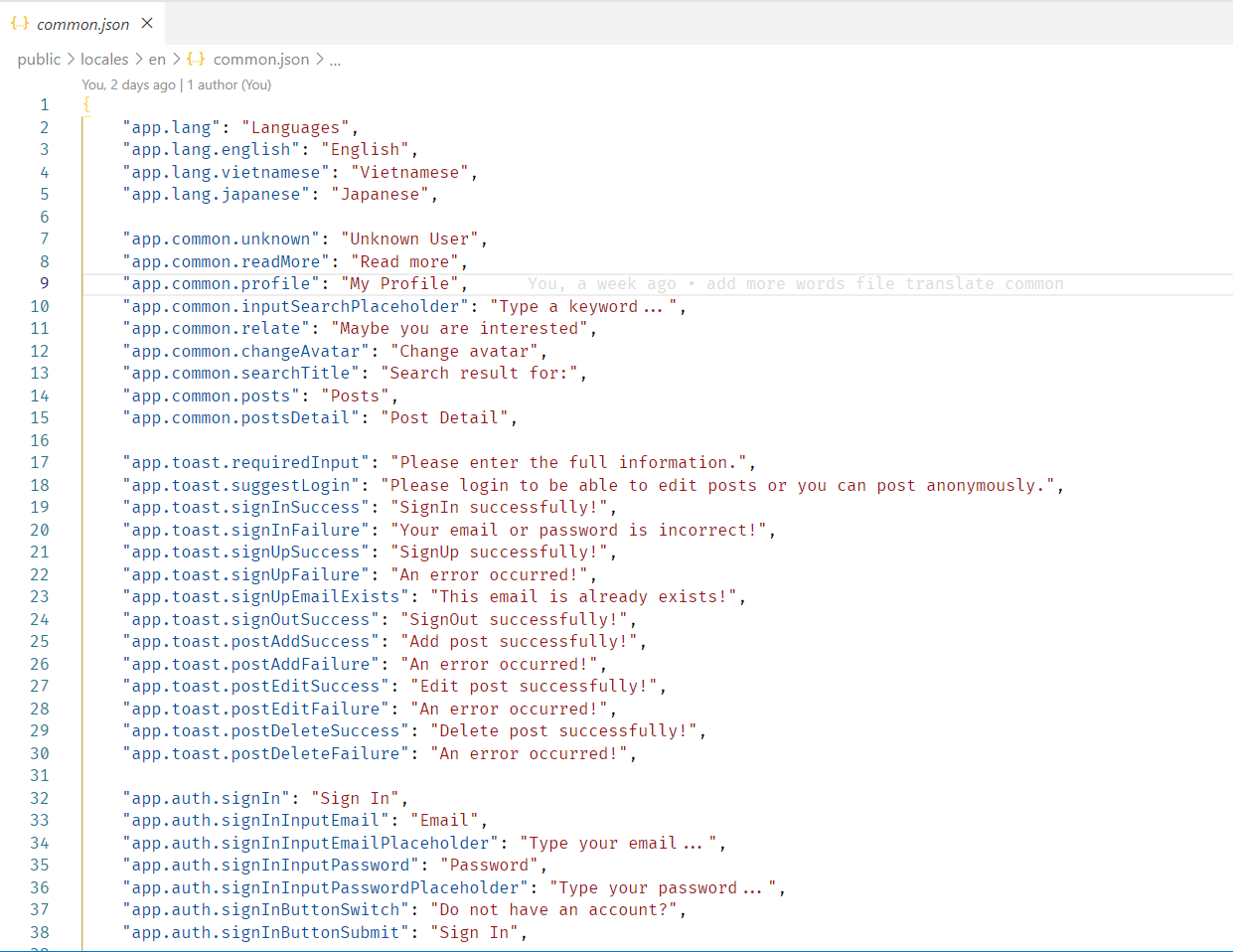
### 2.2.14. Đa ngôn ngữ

Ta cần chuẩn bị một số tài nguyên để hiển thị đa ngôn ngữ, thư mục ngôn ngữ chứa các loại ngôn ngữ khác nhau, ở đây ta có 3 ngôn ngữ Anh, Việt, Nhật. Ta cũng cần một vài hình ảnh để hiển thị.



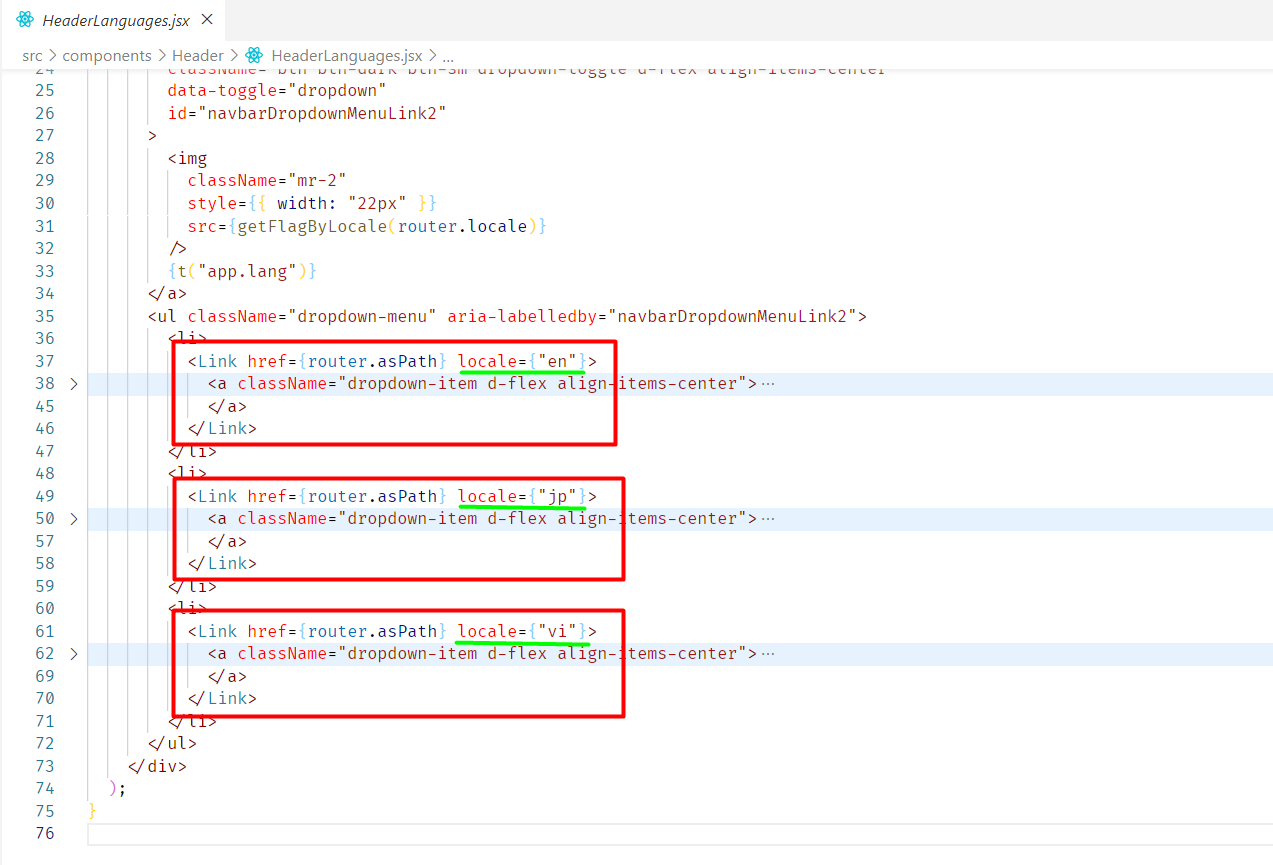
Hình . Thư mục chứa nội dung bản dịch và hình ảnh quốc kỳ.

Mỗi ngôn ngữ có file common chứa ngôn ngữ đó, ta sẽ cho hiển thị ngôn ngữ bằng mã code, ví dụ về một file như sau:



Hình . File nội dung của bản dịch Tiếng Anh.

Ở phần giao diện, ta cho hiển thị thẻ Link, thẻ này dùng để set lại locale của ứng dụng, từ đó có thể nhận biết được ngôn ngữ nào đang sử dụng.



Hình . Sử dụng thẻ Link của NextJS để set locale của ứng dụng.

Ở ứng dụng này ta sẽ sử dụng thư viện next-i18next để cài đặt hãy gõ câu lệnh sau: npm install –save-dev next-i18next

Sau khi cài đặt ta sẽ kiểm tra tại file package.json:



Hình . Kiểm tra thư viện đã cài đặt trong package.json.

Tại file next-i18next.config.js ta cấu hình như sau (nếu chưa có hãy tạo ở thư mục gốc):



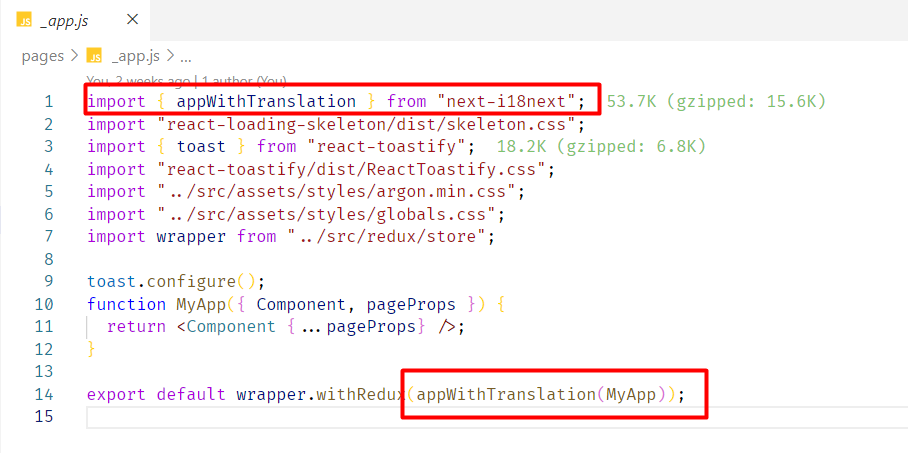
Hình . Cấu hình tại file next-i18next.config.js.

Tiếp tục nhúng file config vào file next.config.js:



Hình . Import file cấu hình vào next.config.js.

Tại file pages/\_app.js tiếp tục cấu hình thêm appWithTranslation.



Hình . Thêm cấu hình vào file \_app.js.

Tại từng files từng trang trên folder pages cấu hình thêm serverSideTranslations.



Hình . Sử dụng serverSideTranslations cho từng trang.

Để sử dụng ta gọi hook useTranslation, và truyền vào tên file đã config trước đó.



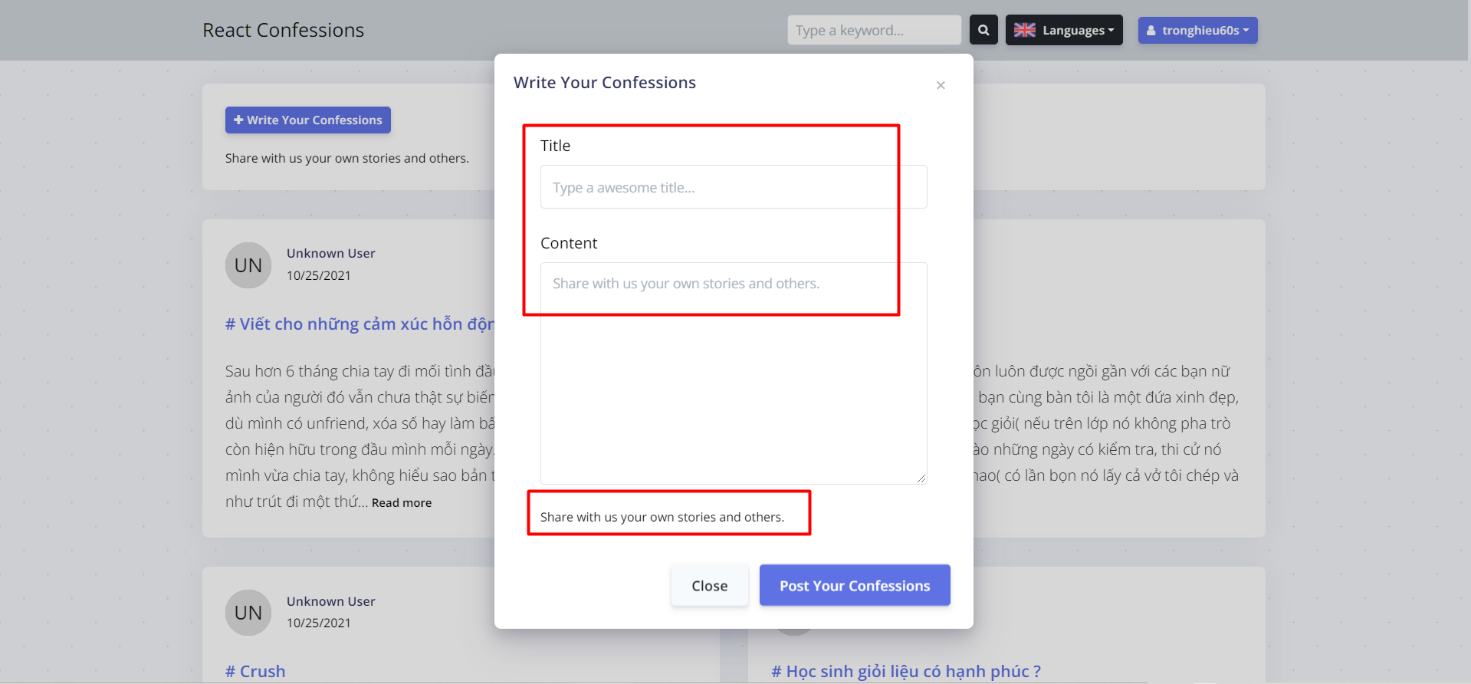
Hình . Sử dụng hook useTranslation để gọi bản dịch.

Để lấy nội dung, ta sẽ truyền mã nội dung mà ta cần hiển thị.

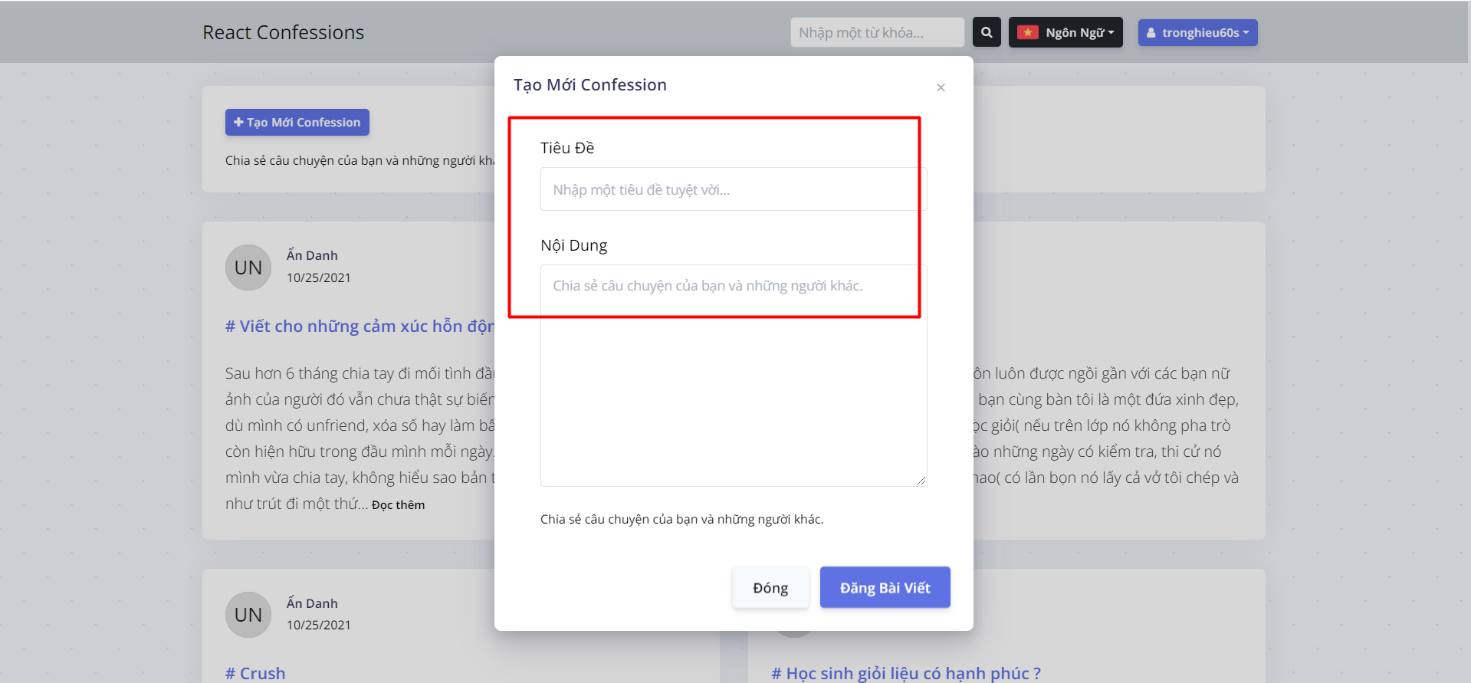


Hình . Để hiển thị ta chỉ cần gọi ra mã nội dung của nó.

Nội dung hiển thị đã xuất hiện, khi đổi ngôn ngữ, ta cũng sẽ lấy được ngôn ngữ đó.



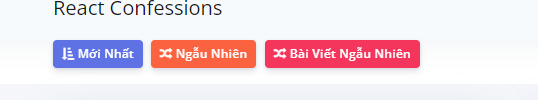
Hình . Nội dung được hiển thị bằng Tiếng Anh.



Hình . Nội dung được hiển thị bằng Tiếng Việt.

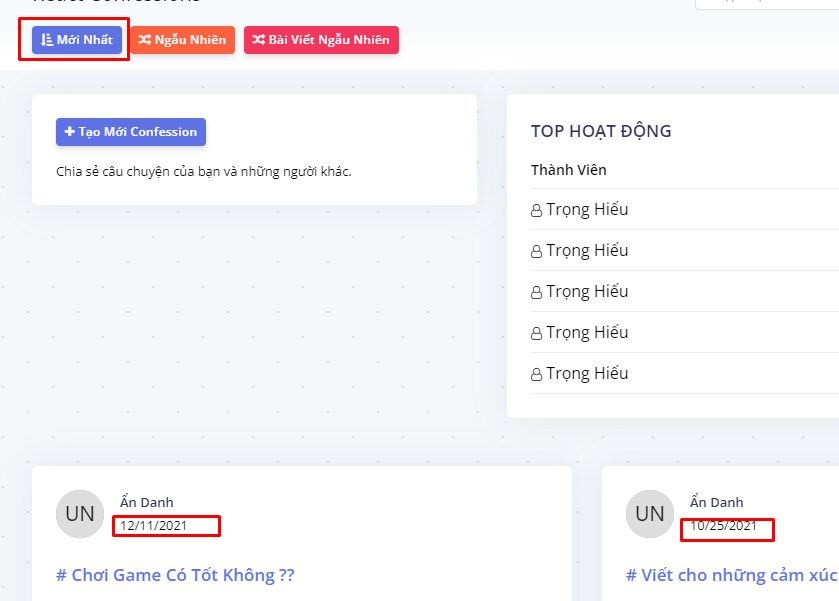
### 2.2.15. Lọc bài viết:

Có 3 cách lọc bài viết: “Mới Nhất” “Ngẫu Nhiên” “Bài Viết Ngẫu Nhiên”



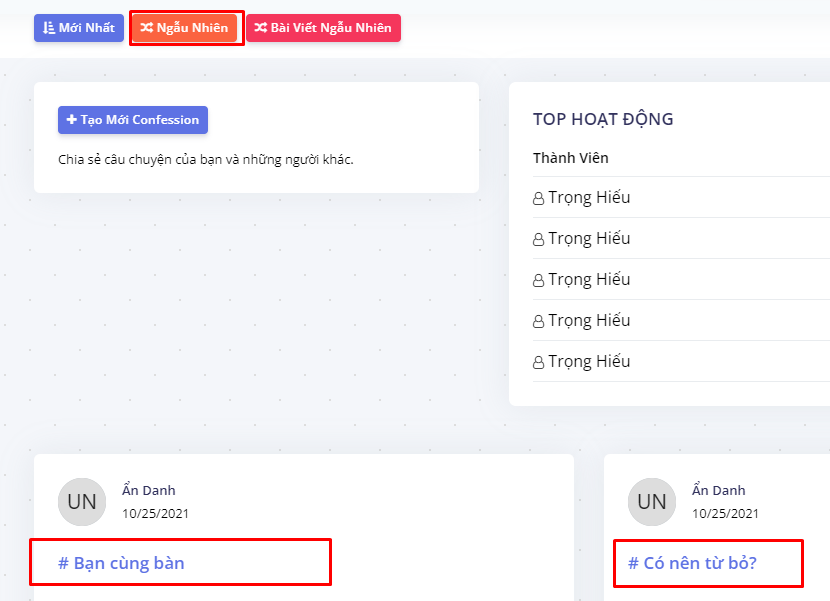
Hình 71. Lọc bài viết.

Khi Click vào “Mới Nhất” thì trang sẽ load lại những confessions mới nhất lên đầu:



Hình 72. Load bài viết mới nhất.

Khi Click vào “Ngẫu Nhiên” thì trang sẽ load lại những confessions ngẫu nhiên:



Hình 73. Load bài viết ngẫu nhiên.

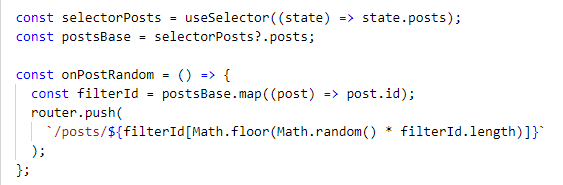
Muốn lọc bài viết như trên, chỉ cần đưa Link vào:



Hình 74. Code mới nhất và ngẫu nhiên.

Khi Click vào “Bài Viết Ngẫu Nhiên” thì sẽ chuyển sang trang chi tiết nội dung bài viết, tất nhiên là theo ngẫu nhiên.

Khai báo “onPostRandom”:



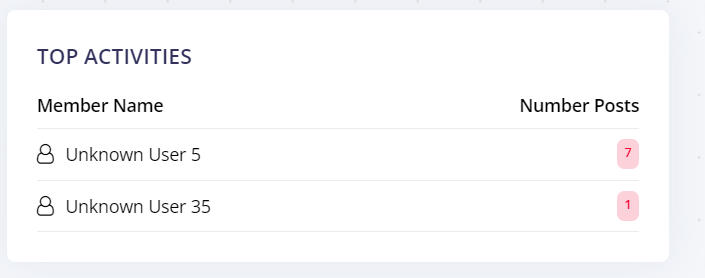
Hình 75. Khai bào hàm random.



Hình 76. Gọi hàm random.

### 2.2.16. Ranking User:

Chúng ta sẽ tạo ra một bảng để hiển thị người có số lượng bài viết cao và đây là giao diện



Hình 71. Nội dung được hiển thị của Ranking User.

Việt đầu tiên sẽ viết hàm để đếm số lượng bài viết và rút gắn lại tìm ra top 5 người có bài viết nhiều nhất.



Hình 72. Hàm đém và Sort bài post của Ranking User.

Sau khi chúng ta ghi xong hàm ở trên thì chúng ta sẽ gọi hàm ở PostsUserRanking.jsx và đồng thời sẽ khỏi tạo biến để dùng và gán nó vào hàm trên.



Hình 72. Nơi được gọi hàm countAndSort.

Sau khi thêm và xử lý PostsUserRanking.jsx thì chúng ra sẽ gọi hàm ra lớp cha để nó có thể hiển thị ra giao diện



Hình 73. Ở lớp cha gọi PostsUserRanking.