|  | 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 QUÁ TRÌNH 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. PHP Unit Test 5](#_heading=h.3dy6vkm)

[1.1. Function sumb. 7](#_heading=h.1t3h5sf)

[1.2. Function auth. 9](#_heading=h.4d34og8)

[1.3. Function getUsers. 11](#_heading=h.17dp8vu)

[1.4. Function insertUser. 12](#_heading=h.3rdcrjn)

[1.5. Function updateUser. 14](#_heading=h.2jxsxqh)

[1.6. Function findBankById. 16](#_heading=h.z337ya)

[1.7. Function insertBank. 17](#_heading=h.3j2qqm3)

[1.8. Function updateBank. 18](#_heading=h.4i7ojhp)

[1.9. Function deleteBank. 19](#_heading=h.2xcytpi)

[1.10. Function deleteUser. 20](#_heading=h.3as4poj)

[1.11. Function getBanks. 21](#_heading=h.1pxezwc)

[1.12. Function findUserById. 23](#_heading=h.49x2ik5)

[1.13. Function findUser. 24](#_heading=h.2p2csry)

[2. DỰ ÁN CHUYÊN ĐỀ WEB 2 25](#_heading=h.3o7alnk)

[2.1. Mô Tả Công Nghệ Được Sử Dụng 25](#_heading=h.1hmsyys)

[2.1.1. ReactJS là gì? 25](#_heading=h.3fwokq0)

[2.1.2. Redux là gì? 25](#_heading=h.2u6wntf)

[2.1.3. Next JS là gì? 26](#_heading=h.19c6y18)

[2.2. Đặc Tả Dự Án 27](#_heading=h.28h4qwu)

[2.2.1. Hiển thị confessions 28](#_heading=h.nmf14n)

[2.2.2. Phân trang hiển thị confessions 30](#_heading=h.1mrcu09)

[2.2.3. Sửa confessions 34](#_heading=h.2lwamvv)

[2.2.4. Đăng nhập confessions 36](#_heading=h.111kx3o)

[2.2.5. Đăng ký confessions 39](#_heading=h.3l18frh)

[2.2.6. Đăng xuất confessions 42](#_heading=h.206ipza)

[2.2.7. Thêm confessions 44](#_heading=h.4k668n3)

[2.2.8. Xóa confessions 46](#_heading=h.2zbgiuw)

[2.2.9. Loading khi xử lý 48](#_heading=h.1egqt2p)

[2.2.10. Tìm kiếm bài viết 50](#_heading=h.3ygebqi)

[2.2.11. Trang chi tiết người dùng 51](#_heading=h.2dlolyb)

[Hình 1. Coverage tất cả các lớp trong ứng dụng. 6](#_heading=h.30j0zll)

[Hình 2. Coverage của lớp UserModel. 6](#_heading=h.1fob9te)

[Hình 3. Coverage của lớp BankModel. 6](#_heading=h.3znysh7)

[Hình 4. Testdox cho function sumb. 8](#_heading=h.tyjcwt)

[Hình 5. Testdox cho function auth. 10](#_heading=h.26in1rg)

[Hình 6. Testdox cho function getUsers. 11](#_heading=h.35nkun2)

[Hình 7. Testdox cho function insertUser. 13](#_heading=h.44sinio)

[Hình 8. Testdox cho function updateUser. 15](#_heading=h.1ci93xb)

[Hình 9. Testdox cho function findBankById. 16](#_heading=h.2bn6wsx)

[Hình 10. Testdox cho function insertBank. 17](#_heading=h.147n2zr)

[Hình 11. Testdox cho function updateBank. 18](#_heading=h.ihv636)

[Hình 12. Testdox cho function deleteBank. 19](#_heading=h.41mghml)

[Hình 13. Testdox cho function deleteUser. 20](#_heading=h.1v1yuxt)

[Hình 14. . Testdox cho function getBanks. 22](#_heading=h.3tbugp1)

[Hình 15. Testdox cho function findUserById. 23](#_heading=h.46r0co2)

[Hình 16. Testdox cho function findUser. 24](#_heading=h.39kk8xu)

[Hình 17. Hình ảnh Logo của ReactJS. 25](#_heading=h.1302m92)

[Hình 18. Hình ảnh mô tả vòng đời (life cycle) của Redux. 26](#_heading=h.2250f4o)

[Hình 19. Hình ảnh Logo của NextJS. 26](#_heading=h.319y80a)

[Hình 20. Trang chủ hiển thị confessions. 28](#_heading=h.upglbi)

[Hình 21. Gọi useEffect để dispatch actions dữ liệu. 28](#_heading=h.1tuee74)

[Hình 22. Gọi API và đưa vào reducers của Redux. 29](#_heading=h.2szc72q)

[Hình 23. Sử dụng use Selector của Redux để lấy dữ liệu. 29](#_heading=h.184mhaj)

[Hình 24. Hiển thị dữ liệu bằng cách map qua mảng dữ liệu. 29](#_heading=h.meukdy)

[Hình 25. Hiển thị phân trang trên giao diện. 30](#_heading=h.1ljsd9k)

[Hình 26. Lấy số trang hiện tại bằng useRouter, mặc định là 1. 30](#_heading=h.2koq656)

[Hình 27. Sử dụng useEffect để xử lí bài viết hiển thị. 31](#_heading=h.3jtnz0s)

[Hình 28. Hiển thị pagination khi số lượng bài viết lớn hơn 0.. 31](#_heading=h.4iylrwe)

[Hình 29. Hiển thị Pagination cho bài viết. 32](#_heading=h.2fk6b3p)

[Hình 30. Logic xử lý hiển thị pagination. 33](#_heading=h.2y3w247)

[Hình 31. Hiển thị sửa khi người dùng đăng nhập và đúng bài viết. 34](#_heading=h.1d96cc0)

[Hình 32. Hiển thị modal sửa khi người dùng nhấp vào nút sửa. 34](#_heading=h.3x8tuzt)

[Hình 33. Khi click vào nút sửa sẽ gọi đến function sửa. 35](#_heading=h.2ce457m)

[Hình 34. Xử lý sửa bài viết và load lại dữ liệu. 35](#_heading=h.3bj1y38)

[Hình 35. Nơi hiển thị button Sign In. 36](#_heading=h.2pta16n)

[Hình 36. Hiện model để đăng nhập. 36](#_heading=h.3oy7u29)

[Hình 37. Hiện model để đăng ký. 37](#_heading=h.j8sehv)

[Hình 38. Hàm Sign Up action ở Redux. 38](#_heading=h.338fx5o)

[Hình 39. Model Sign In. 39](#_heading=h.wnyagw)

[Hình 40. Sau đăng nhập thành công. 39](#_heading=h.1vsw3ci)

[Hình 41. Hàm Redux xử lý đăng nhập. 40](#_heading=h.2uxtw84)

[Hình 42. Hàm sẽ thực hiện khi người dùng reload trang. 40](#_heading=h.3u2rp3q)

[Hình 43. Hàm Redux loading đăng nhập. 41](#_heading=h.odc9jc)

[Hình 44. Hiện button Sign Out. 42](#_heading=h.2mn7vak)

[Hình 45. Ở hàm auth Reducer. 42](#_heading=h.3ls5o66)

[Hình 46. Hàm Logout ở Redux. 43](#_heading=h.20xfydz)

[Hình 47. HIện thông báo. 44](#_heading=h.3z7bk57)

[Hình 48. Model Thêm Confessions. 44](#_heading=h.2eclud0)

[Hình 49. Khi click vào nút thêm sẽ gọi đến function thêm. 45](#_heading=h.3dhjn8m)

[Hình 50. Xử lý thêm bài viết và load lại dữ liệu. 45](#_heading=h.4cmhg48)

[Hình 51. Hiển thị xóa khi người dùng đăng nhập và đúng bài viết. 46](#_heading=h.l7a3n9)

[Hình 52. Hiển thị Model xóa khi người dùng click vào nút xóa. 46](#_heading=h.356xmb2)

[Hình 53. Khi click vào nút xóa sẽ gọi đến function xóa. 47](#_heading=h.1kc7wiv)

[Hình 54. Xử lý xóa bài viết và load lại dữ liệu. 47](#_heading=h.2jh5peh)

[Hình 55. Giao diện loading. 48](#_heading=h.1xrdshw)

[Hình 56. Tạo 2 case action. 48](#_heading=h.4hr1b5p)

[Hình 57. Dispatch action trong một function xử lý khác. 49](#_heading=h.2wwbldi)

[Hình 58. Gọi giá trị loading từ Redux và sử dụng. 49](#_heading=h.1c1lvlb)

[Hình 59. Giao diện sau khi tìm kiếm. 50](#_heading=h.2b6jogx)

[Hình 60. Khai báo onSearch. 50](#_heading=h.qbtyoq)

[Hình 61. use Effect hiển thị mục tìm kiếm. 50](#_heading=h.3abhhcj)

[Hình 62. Lọc các bài viết của người dùng đã tạo. 51](#_heading=h.2olpkfy)

[Hình 63. Sử dụng redux mà trước đó đã lưu thông tin người dùng. 51](#_heading=h.13qzunr)

[Hình 64. Nhúng thư viện mã hóa md5 vào dự án. 51](#_heading=h.3nqndbk)

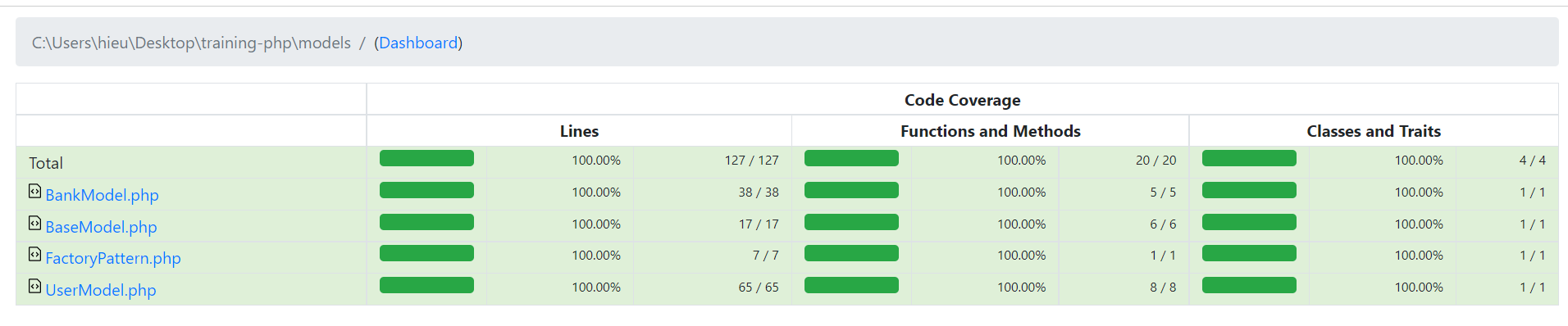
[Hình 65. Hiển thị hình ảnh người dùng sau khi hash ra mã md5. 52](#_heading=h.22vxnjd)

# 1. PHP Unit Test

Phân chia công việc:

| **STT** | **File** | **Function** | **Phân Công** |
| --- | --- | --- | --- |
| 1 | UserModel.php | sumb | Trần Trọng Hiếu |
| 2 | UserModel.php | auth |
| 3 | UserModel.php | getUsers |
| 4 | UserModel.php | insertUser |
| 5 | UserModel.php | updateUser |
| 6 | UserModel.php | deleteUserById | Ngô Bảy Hên |
| 7 | UserModel.php | findUser |
| 8 | UserModel.php | findUserById |
| 9 | BankModel.php | getBanks |
| 10 | BankModel.php | insertBank | Nguyễn Huỳnh Quốc Trung |
| 11 | BankModel.php | updateBank |
| 12 | BankModel.php | deleteBankById |
| 13 | BankModel.php | findBankById |

Coverage:



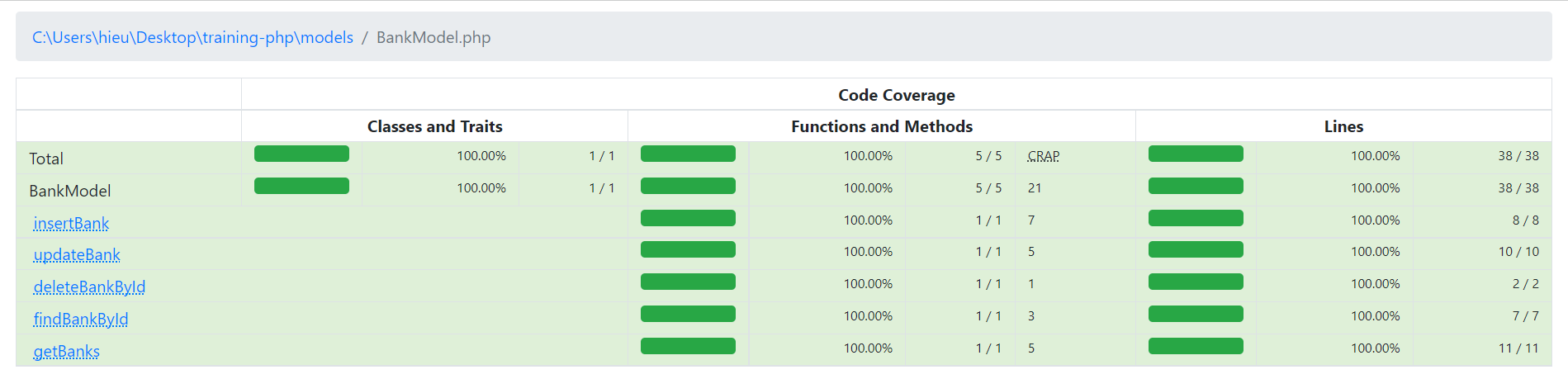
*Hình 1. Coverage tất cả các lớp trong ứng dụng.*

Coverage UserModel:



*Hình 2. Coverage của lớp UserModel.*

Coverage BankModel:

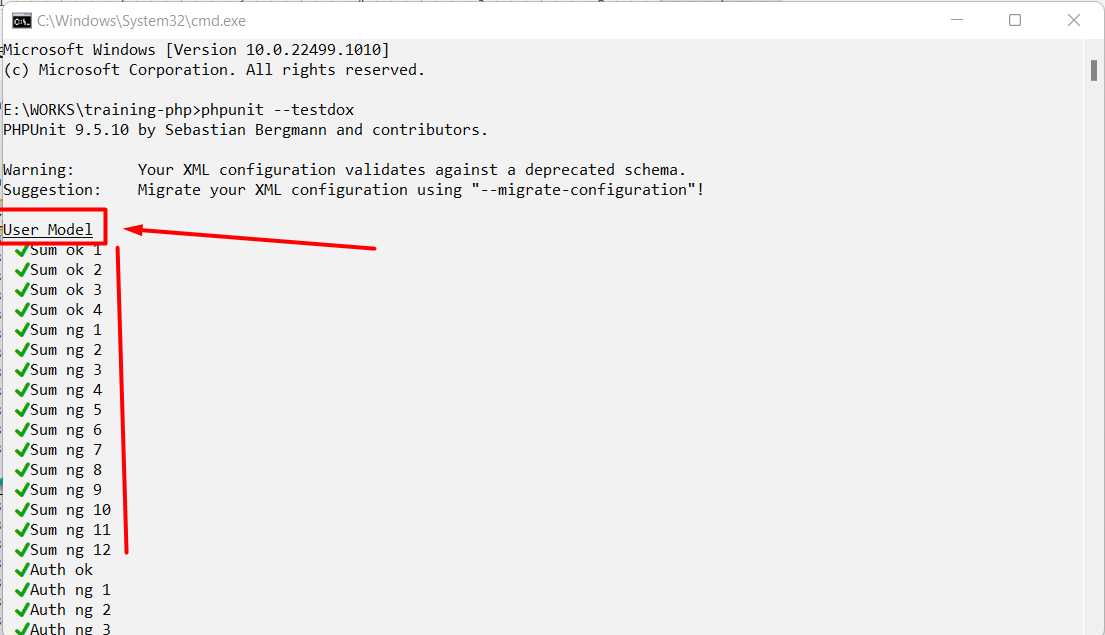


*Hình 3. Coverage của lớp BankModel.*

## 1.1. Function sumb.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | sumb | testSumOk1 | OK | Cộng số dương và số dương |
| 2 | sumb | testSumOk2 | OK | Cộng số âm và số âm |
| 3 | sumb | testSumOk3 | OK | Cộng số dương và số âm |
| 4 | sumb | testSumOk4 | OK | Cộng hai số thập phân |
| 5 | sumb | testSumNg1 | OK | Cộng số và chuỗi |
| 6 | sumb | testSumNg2 | OK | Cộng chuỗi và chuỗi |
| 7 | sumb | testSumNg3 | OK | Cộng số và null |
| 8 | sumb | testSumNg4 | OK | Cộng null và null |
| 9 | sumb | testSumNg5 | OK | Cộng số và chuỗi rỗng |
| 10 | sumb | testSumNg6 | OK | Cộng chuỗi rỗng và chuỗi rỗng |
| 11 | sumb | testSumNg7 | OK | Cộng số và mảng |
| 12 | sumb | testSumNg8 | OK | Cộng mảng và mảng |
| 13 | sumb | testSumNg9 | OK | Cộng số và object |
| 14 | sumb | testSumNg10 | OK | Cộng object và object |
| 15 | sumb | testSumNg11 | OK | Cộng số và (true/false) |
| 16 | sumb | testSumNg12 | OK | Cộng (true/false) và (true/false) |

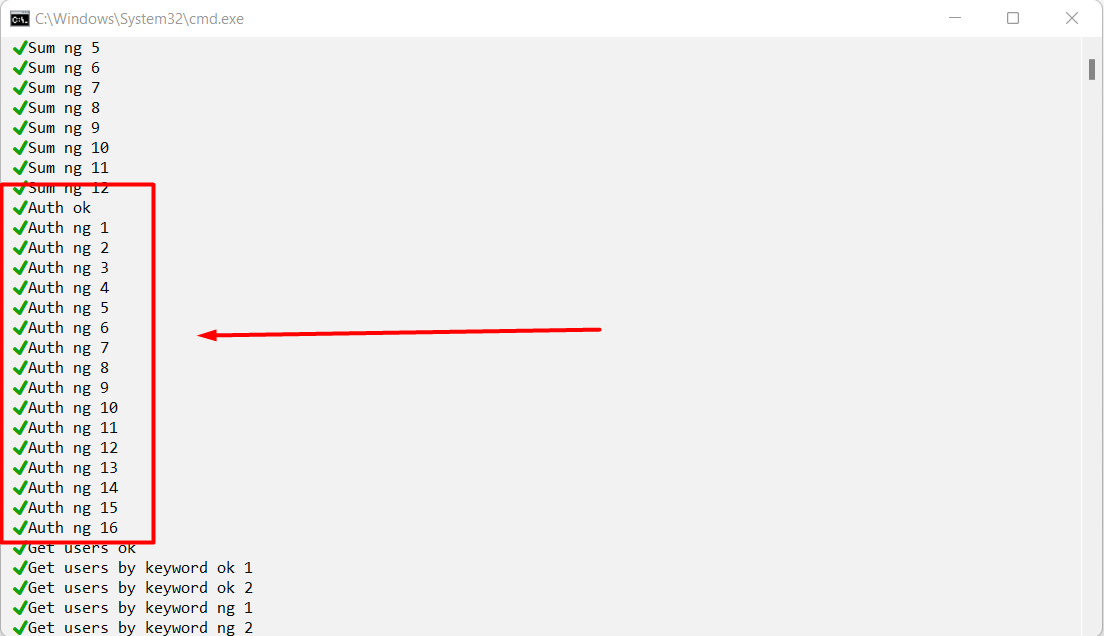


*Hình 4. Testdox cho function sumb.*

## 1.2. Function auth.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | auth | testAuthOk | OK | Kiểm tra xác thực user |
| 2 | auth | testAuthNg1 | OK | Kiểm tra xác thực user khi username không tồn tại |
| 3 | auth | testAuthNg2 | OK | Kiểm tra xác thực user khi username là rỗng |
| 4 | auth | testAuthNg3 | OK | Kiểm tra xác thực user khi username là null |
| 5 | auth | testAuthNg4 | OK | Kiểm tra xác thực user khi username là số |
| 6 | auth | testAuthNg5 | OK | Kiểm tra xác thực user khi username là số thực |
| 7 | auth | testAuthNg6 | OK | Kiểm tra xác thực user khi username là mảng |
| 8 | auth | testAuthNg7 | OK | Kiểm tra xác thực user khi username là object |
| 9 | auth | testAuthNg8 | OK | Kiểm tra xác thực user khi username là (true/false) |
| 10 | auth | testAuthNg9 | OK | Kiểm tra xác thực user khi password không chính xác |
| 11 | auth | testAuthNg10 | OK | Kiểm tra xác thực user khi password là rỗng |
| 12 | auth | testAuthNg11 | OK | Kiểm tra xác thực user khi password là null |
| 13 | auth | testAuthNg12 | OK | Kiểm tra xác thực user khi password là số |
| 14 | auth | testAuthNg13 | OK | Kiểm tra xác thực user khi password là số thực |
| 15 | auth | testAuthNg14 | OK | Kiểm tra xác thực user khi password là mảng |
| 16 | auth | testAuthNg15 | OK | Kiểm tra xác thực user khi password là object |
| 17 | auth | testAuthNg16 | OK | Kiểm tra xác thực user khi password là (true/false) |



*Hình 5. Testdox cho function auth.*

## 1.3. Function getUsers.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | getUsers | testGetUsersOk | OK | Kiểm tra lấy tất cả users hiện tại |
| 2 | getUsers | testGetUsersByKeywordOk1 | OK | Kiểm tra lấy tất cả users bằng keyword |
| 3 | getUsers | testGetUsersByKeywordOk2 | OK | Kiểm tra lấy tất cả users bằng keyword (chuỗi lạ) |
| 4 | getUsers | testGetUsersByKeywordNg1 | OK | Kiểm tra lấy tất cả users với keyword là số |
| 5 | getUsers | testGetUsersByKeywordNg2 | OK | Kiểm tra lấy tất cả users với keyword là mảng |
| 6 | getUsers | testGetUsersByKeywordNg3 | OK | Kiểm tra lấy tất cả users với keyword là object |
| 7 | getUsers | testGetUsersBy Keyword g4 | OK | Kiểm tra lấy tất cả users với keyword là (true/false) |



*Hình 6. Testdox cho function getUsers.*

## 1.4. Function insertUser.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | insertUser | testInsertUserOk | OK | Kiểm tra thêm user mới |
| 2 | insertUser | testInsertUserNg1 | OK | Kiểm tra thêm user với input là rỗng |
| 3 | insertUser | testInsertUserNg2 | OK | Kiểm tra thêm user với input là null |
| 4 | insertUser | testInsertUserNg3 | OK | Kiểm tra thêm user với input là số |
| 5 | insertUser | testInsertUserNg4 | OK | Kiểm tra thêm user với input là mảng |
| 6 | insertUser | testInsertUserNg5 | OK | Kiểm tra thêm user với input là object |
| 7 | insertUser | testInsertUserNg6 | OK | Kiểm tra thêm user với input là (true/false) |
| 8 | insertUser | testInsertUserNg7 | OK | Kiểm tra thêm user với giá trị key input là rỗng |
| 9 | insertUser | testInsertUserNg8 | OK | Kiểm tra thêm user với giá trị key input là null |
| 10 | insertUser | testInsertUserNg9 | OK | Kiểm tra thêm user với giá trị key input là số |
| 11 | insertUser | testInsertUserNg10 | OK | Kiểm tra thêm user với giá trị key input là mảng |
| 12 | insertUser | testInsertUserNg11 | OK | Kiểm tra thêm user với giá trị key input là object |
| 13 | insertUser | testInsertUserNg12 | OK | Kiểm tra thêm user với giá trị key input là (true/false) |



*Hình 7. Testdox cho function insertUser.*

## 1.5. Function updateUser.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | updateUser | testUpdateUserOk1 | OK | Kiểm tra cập nhật user |
| 2 | updateUser | testUpdateUserOk2 | OK | Kiểm tra cập nhật user với giá trị key input là rỗng |
| 3 | updateUser | testUpdateUserNg1 | OK | Kiểm tra cập nhật user với input là rỗng |
| 4 | updateUser | testUpdateUserNg2 | OK | Kiểm tra cập nhật user với input là null |
| 5 | updateUser | testUpdateUserNg3 | OK | Kiểm tra cập nhật user với input là số |
| 6 | updateUser | testUpdateUserNg4 | OK | Kiểm tra cập nhật user với input là mảng |
| 7 | updateUser | testUpdateUserNg5 | OK | Kiểm tra cập nhật user với input là object |
| 8 | updateUser | testUpdateUserNg6 | OK | Kiểm tra cập nhật user với input là (true/false) |
| 9 | updateUser | testUpdateUserNg7 | OK | Kiểm tra cập nhật user với giá trị key input là null |
| 10 | updateUser | testUpdateUserNg8 | OK | Kiểm tra cập nhật user với giá trị key input là số |
| 11 | updateUser | testUpdateUserNg9 | OK | Kiểm tra cập nhật user với giá trị key input là mảng |
| 12 | updateUser | testUpdateUserNg10 | OK | Kiểm tra cập nhật user với giá trị key input là object |
| 13 | updateUser | testUpdateUserNg11 | OK | Kiểm tra cập nhật user với giá trị key input là (true/false) |
| 14 | updateUser | testUpdateUserNg12 | OK | Kiểm tra cập nhật user với giá trị id input là rỗng |
| 15 | updateUser | testUpdateUserNg13 | OK | Kiểm tra cập nhật user với giá trị id input là null |
| 16 | updateUser | testUpdateUserNg14 | OK | Kiểm tra cập nhật user với giá trị id input là số |
| 17 | updateUser | testUpdateUserNg15 | OK | Kiểm tra cập nhật user với giá trị id input là mảng |
| 18 | updateUser | testUpdateUserNg16 | OK | Kiểm tra cập nhật user với giá trị id input là object |
| 19 | updateUser | testUpdateUserNg17 | OK | Kiểm tra cập nhật user với giá trị id input là (true/false) |

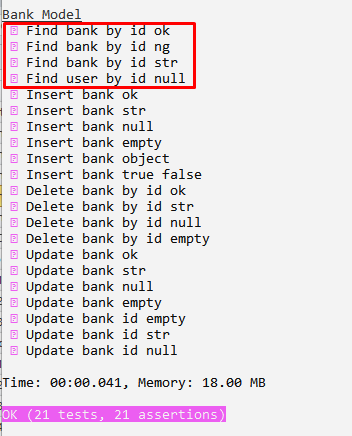


*Hình 8. Testdox cho function updateUser.*

## 1.6. Function findBankById.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | findBankById | testFindBankByIdOk | OK | Kiểm tra tìm bank |
| 2 | findBankById | testFindBankByIdNg | OK | Kiểm tra tìm bank với id quá mức |
| 3 | findBankById | testFindBankByIdStr | OK | Kiểm tra tìm bank với id là chuỗi |
| 4 | findBankById | testFindUserByIdObject | OK | Kiểm tra tìm bank với id là Object |
| 5 | findBankById | testFindUserByIdNull | OK | Kiểm tra tìm bank với id là null |

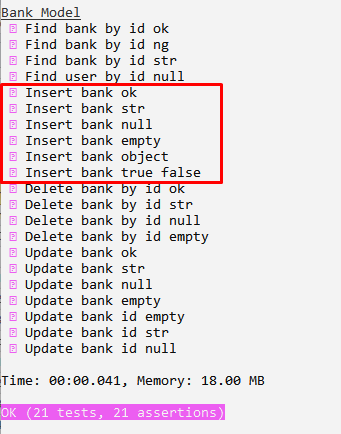


*Hình 9. Testdox cho function findBankById.*

## 1.7. Function insertBank.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | insertBank | testInsertBankEmpty | OK | Kiểm tra thêm bank với input rỗng |
| 2 | insertBank | testInsertBankObject | OK | Kiểm tra thêm bank với input là Object |
| 3 | insertBank | testInsertBankTrueFalse | OK | Kiểm tra thêm bank với input là True/False |
| 4 | insertBank | testInsertBankOk | OK | Kiểm tra thêm bạn mới |
| 5 | insertBank | testInsertBankStr | OK | Kiểm tra thêm bank với input là chuỗi |
| 6 | insertBank | testInsertBankNull | OK | Kiểm tra thêm bank với input là null |

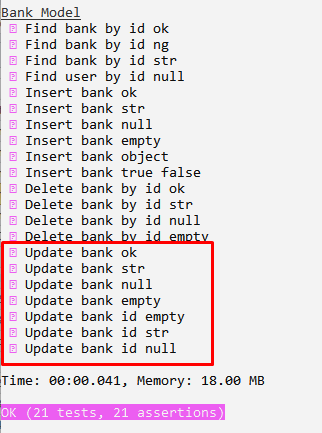


*Hình 10. Testdox cho function insertBank.*

## 1.8. Function updateBank.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | updateBank | testUpdateBankEmpty | OK | Kiểm tra cập nhật bank với input là rỗng |
| 2 | updateBank | testUpdateBankIdEmpty | OK | Kiểm tra cập nhật bank với id là rỗng |
| 3 | updateBank | testUpdateBankIdStr | OK | Kiểm tra cập nhật bank với id là chuỗi |
| 4 | updateBank | testUpdateBankOk | OK | Kiểm tra cập nhật bank |
| 5 | updateBank | testUpdateBankStr | OK | Kiểm tra cập nhật bank với input là chuỗi |
| 6 | updateBank | testUpdateBankNull | OK | Kiểm tra cập nhật bank với input là null |

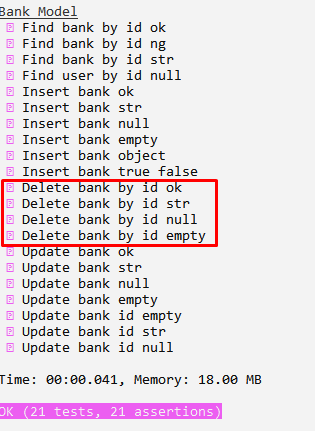


*Hình 11. Testdox cho function updateBank.*

## 1.9. Function deleteBank.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | deleteBankById | testDeleteBankByIdOk | OK | Kiểm tra xóa bank |
| 2 | deleteBankById | testDeleteBankByIdStr | OK | Kiểm tra xóa bank với id là chuỗi |
| 3 | deleteBankById | testDeleteBankByIdNull | OK | Kiểm tra xóa bank với id là null |
| 4 | deleteBankById | testDeleteBankByIdEmpty | OK | Kiểm tra xóa bank với id rỗng |
|  |  |  |  |  |
|  |  |  |  |  |



*Hình 12. Testdox cho function deleteBank.*

## 1.10. Function deleteUser.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | deleteUserById | testDeleteUserByIdOk | OK | Kiểm tra xóa user |
| 2 | deleteUserById | testDeleteUserByIdStr | OK | Kiểm tra xóa user với id là kiểu chuỗi |
| 3 | deleteUserById | testDeleteUserByIdNull | OK | Kiểm tra xóa user với id là kiểu null |
| 4 | deleteUserById | testDeleteUserByIdNE | OK | Kiểm tra xóa user với id là kiểu không tồn tại |
| 5 | deleteUserById | testDeleteUserEmpty | OK | Kiểm tra xóa user với id là kiểu rỗng |
| 6 | deleteUserById | testDeleteUserByArray | OK | Kiểm tra xóa user với id là kiểu mảng |
| 7 | deleteUserById | testDeleteUserByObject | OK | Kiểm tra xóa user với id là kiểu object |
|  |  |  |  |  |
|  |  |  |  |  |



*Hình 13. Testdox cho function deleteUser.*

## 1.11. Function getBanks.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | getBanks | testGetBankOk | OK | Kiểm tra lấy tất cả banks |
| 2 | getBanks | testGetBankNull | OK | Kiểm tra param chuyền vào lấy bank là kiểu null |
| 3 | getBanks | testGetBankStr | OK | Kiểm tra param chuyền vào lấy bank là kiểu string |
| 4 | getBanks | testGetBankNE | OK | Kiểm tra param chuyền vào lấy bank là kiểu NE |
| 5 | getBanks | testGetBankTrue | OK | Kiểm tra param chuyền vào lấy bank là kiểu true |
| 6 | getBanks | testGetBankFalse | OK | Kiểm tra param chuyền vào lấy bank là kiểu false |
| 7 | getBanks | testGetBankObject | OK | Kiểm tra param chuyền vào lấy bank là kiểu object |
| 8 | getBanks | testGetBankEmpty | OK | Kiểm tra param chuyền vào lấy bank là kiểu empty |
| 9 | getBanks | testGetBankKeywordEmpty | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu empty |
| 10 | getBanks | testGetBankKeywordNull | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu null |
| 11 | getBanks | testGetBankKeywordInt | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu int |
| 12 | getBanks | testGetBank Keywords Array | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu mảng |
| 13 | getBanks | testGetBankKeywordTrueFalse | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu bool |
| 14 | getBanks | testGetBankKeywordObject | OK | Kiểm tra lấy tất cả bank bằng keyword kiểu object |



*Hình 14. . Testdox cho function getBanks.*

## 1.12. Function findUserById.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | findUserById | testFindUserByIdOk | OK | Kiểm tra tìm user bằng id |
| 2 | findUserById | testFindUserByIdNE | OK | Kiểm tra tìm user bằng id không tồn tại |
| 3 | findUserById | testFindUserByIdStr | OK | Kiểm tra tìm user bằng id là kiểu string |
| 4 | findUserById | testFindUserByIdNull | OK | Kiểm tra tìm user bằng id là kiểu null |
| 5 | findUserById | testFindUserByIdObject | OK | Kiểm tra tìm user bằng id là kiểu object |
| 6 | findUserById | testFindUserByIdEmpty | OK | Kiểm tra tìm user bằng id là kiểu empty |
| 7 | findUserById | testFindUserByIdArray | OK | Kiểm tra tìm user bằng id là kiểu mảng |
|  |  |  |  |  |
|  |  |  |  |  |



*Hình 15. Testdox cho function findUserById.*

## 1.13. Function findUser.

Bảng mô tả test case:

| **STT** | **Function** | **Function Test** | **Kết quả** | **Mô tả** |
| --- | --- | --- | --- | --- |
| 1 | findUser | testFindUserOk | OK | Kiểm tra tìm user name và email |
| 2 | findUser | testFindUserKeywordInt | OK | Kiểm tra tìm user name và email keyword kiểu int |
| 3 | findUser | testFindUserKeywordTrue | OK | Kiểm tra tìm user name và email keyword kiểu bool |
| 4 | findUser | testFindUserKeywordFalse | OK | Kiểm tra tìm user name và email keyword kiểu bool |
| 5 | findUser | testFindUserKeywordArray | OK | Kiểm tra tìm user name và email keyword kiểu array |
| 6 | findUser | testFindUserKeywordObject | OK | Kiểm tra tìm user name và email keyword kiểu object |
| 7 | findUser | testFindUserKeywordNull | OK | Kiểm tra tìm user name và email keyword kiểu null |
| 8 | findUser | testFindUserKeywordEmpty | OK | Kiểm tra tìm user name và email keyword kiểu empty |



*Hình 16. Testdox cho function findUser.*

# 2. DỰ ÁN CHUYÊN ĐỀ WEB 2

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.

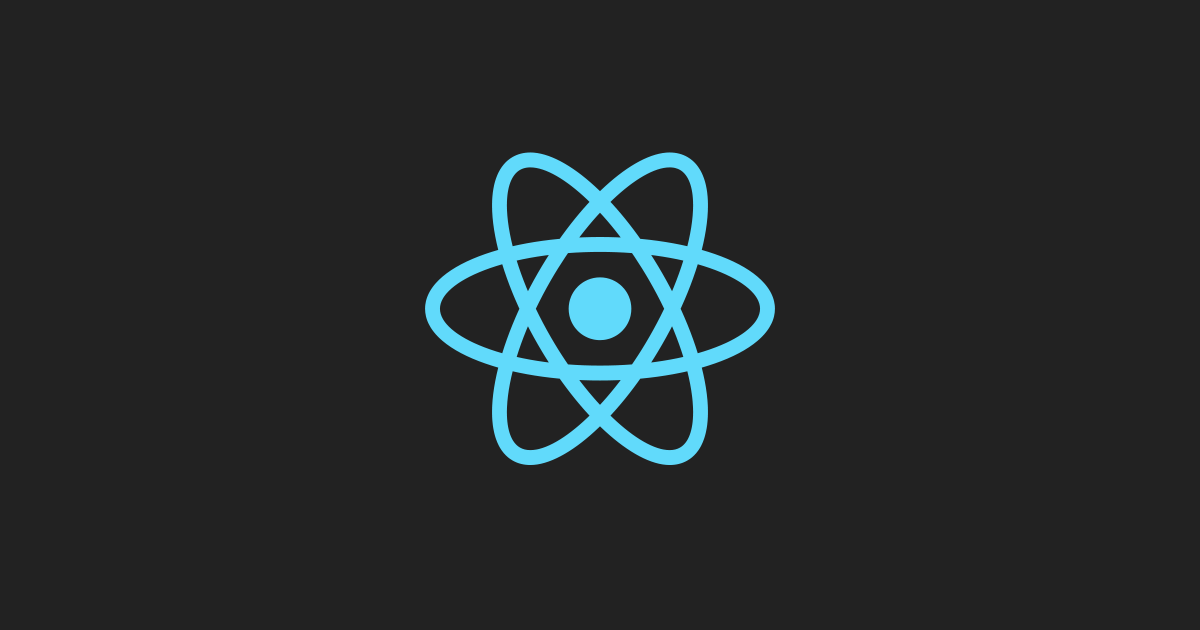
## 2.1. Mô Tả Công Nghệ Được Sử Dụng

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**.

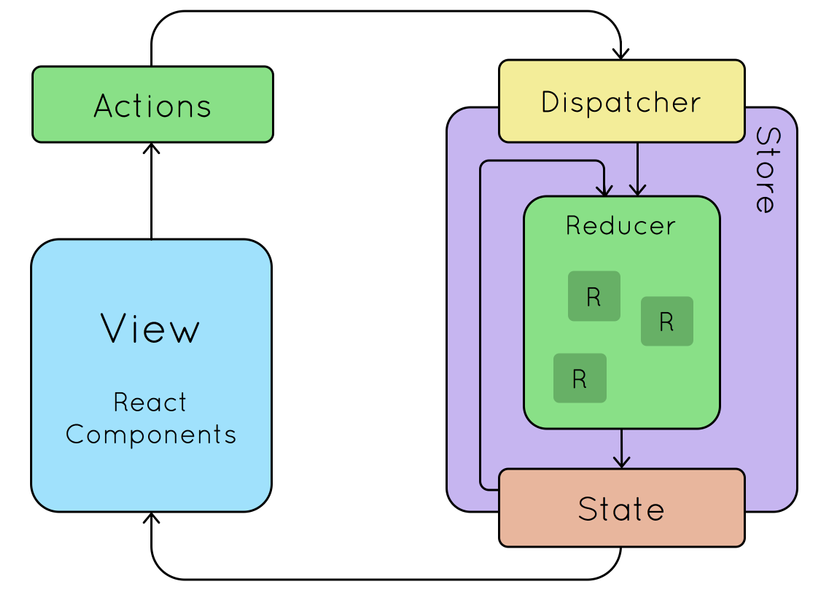
ReactJS sử dụng để tạo giao diện và quản lý cấu trúc cực tốt, và được đứng sau bởi một ông lớn trong ngành IT. Ngoài ra ReactJS được kế thừa một kho thư viện đồ sộ trên NPM. Không thể không kể đến Redux.



*Hình 17. 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 18. Hình ảnh mô tả vòng đời (life cycle) của Redux.*

### 2.1.3. Next JS 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 19. Hình ảnh Logo của NextJS.*

## 2.2. Đặc Tả Dự Án

Các chức năng và phân công công việc CDSPTW1:

| **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 CDSPTW2:

| **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 | 50% |
| 2 | Trần Trọng Hiếu | Đa Ngôn Ngữ  Trang Profile người dùng | 14 ngày | 50% |
| 3 | Nguyễn Huỳnh Quốc Trung | Tìm kiếm bài viết  Trang chi tiết bài viết (nội dung) | 10 ngày | 50% |

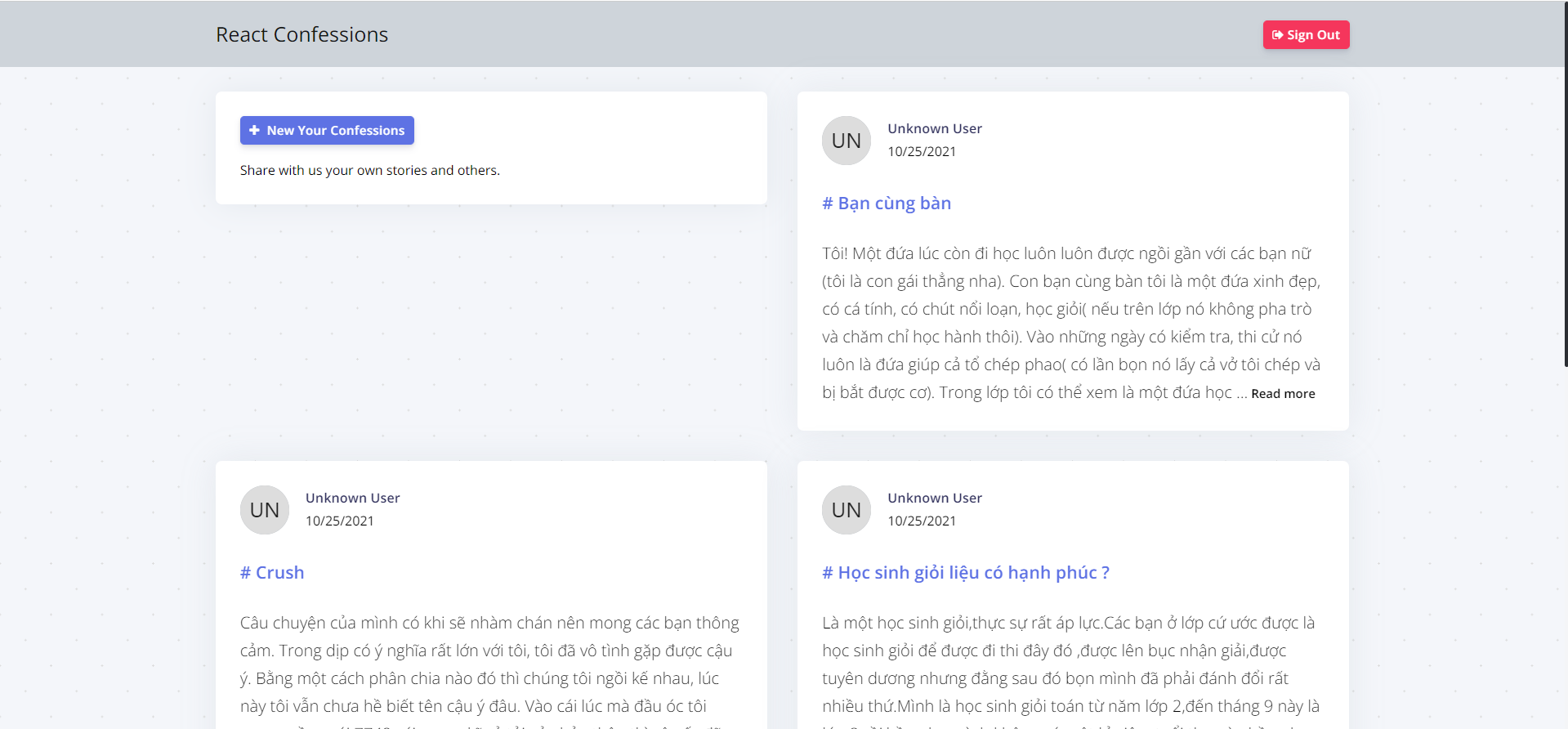
Các chức năng và phân công công việc CDSPTW2 (BCQT):

| **STT** | **Thành Viên** | **Nội Dung Được Phân Công** | **Bắt Đầu** | **Kết Thúc** | **Hoàn Thành** |
| --- | --- | --- | --- | --- | --- |
| 1 | Ngô Bảy Hên | Loading khi xử lý API (9) | 14/11 | 18/11 | 100% |
| 2 | Trần Trọng Hiếu | Trang Profile người dùng (11) | 15/11 | 189/11 | 100% |
| 3 | Nguyễn Huỳnh Quốc Trung | Tìm kiếm bài viết (10) | 13/11 | 16/11 | 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.

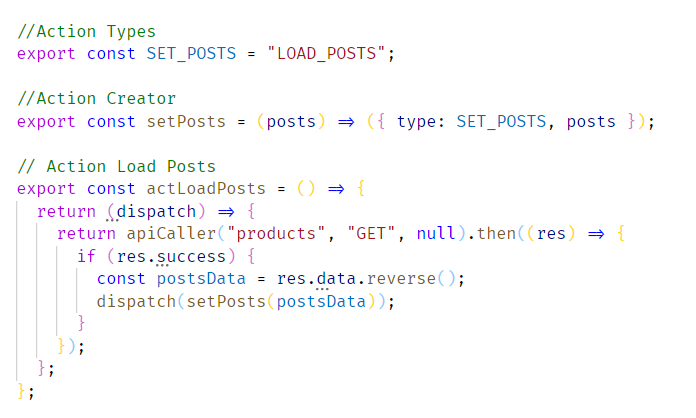


*Hình 20. Trang chủ hiển thị confessions.*

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.

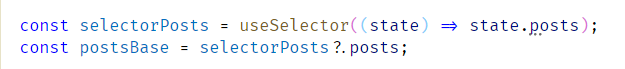


*Hình 21. Gọi useEffect để dispatch actions dữ liệu.*

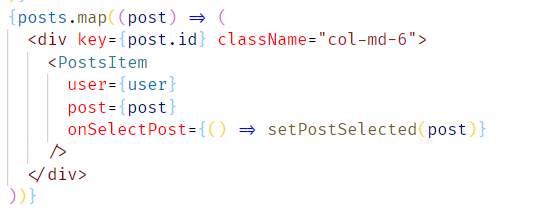


*Hình 22. Gọi API và đưa vào reducers của Redux.*

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 23. Sử dụng use Selector của Redux để lấy dữ liệu.*

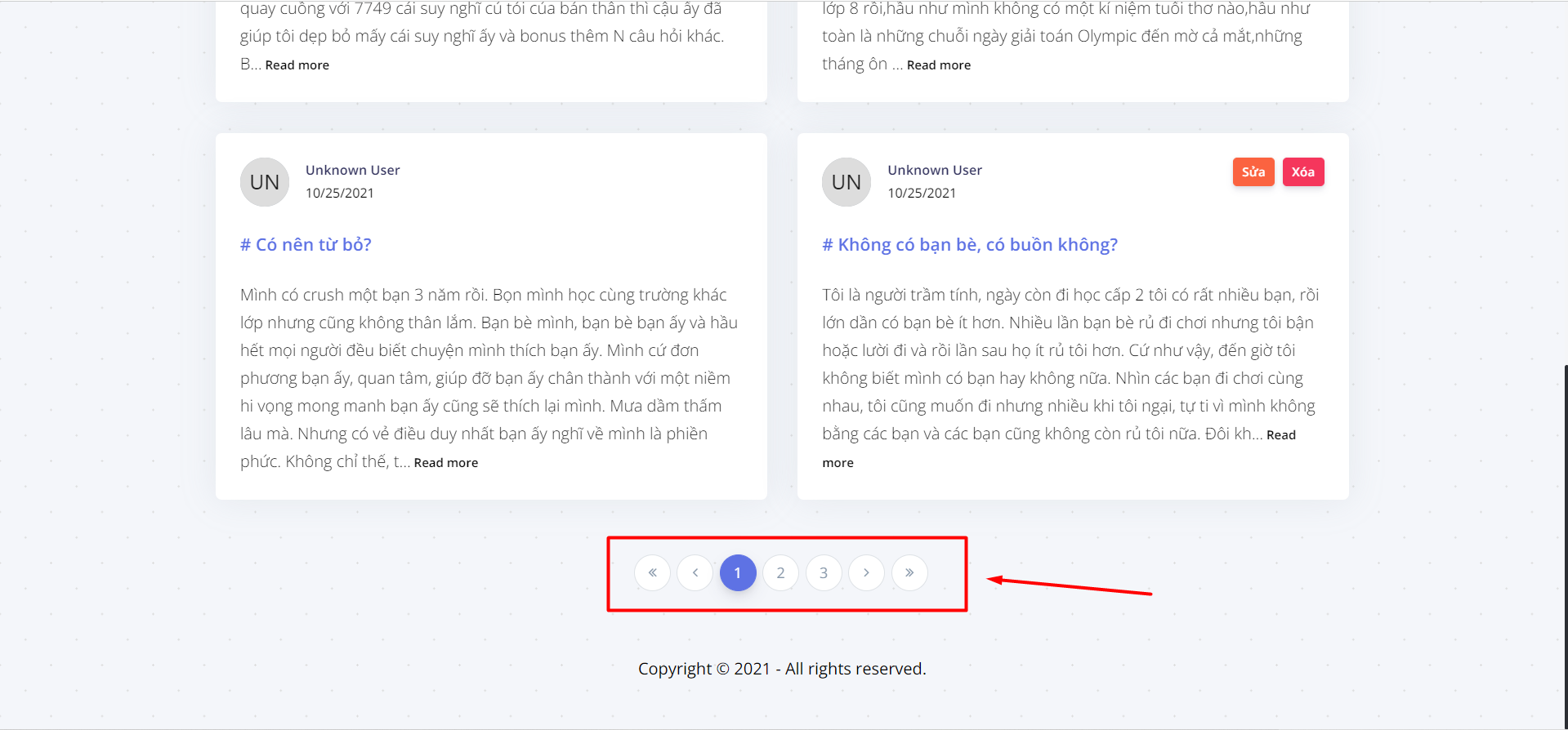


*Hình 24. Hiển thị dữ liệu bằng cách map qua mảng dữ liệu.*

### 2.2.2. Phân trang hiển thị confessions

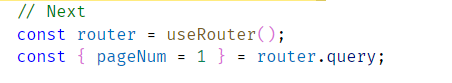
Hiển thị phân trang ở phía dưới của bài viết, giao diện bao gồm:

* 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 25. Hiển thị phân trang trên giao diện.*

Để phân trang ta cần lấy số trang hiện tại, vì sử dụng extJS nên ta có useR router.



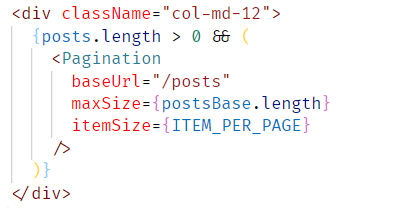
*Hình 26. Lấy số trang hiện tại bằng usb Router, mặc định là 1.*

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 27. Sử dụng useEffect để xử lí bài viết hiển thị.*

Hiển thị Pagination ở trang chủ bằng components Pagination.

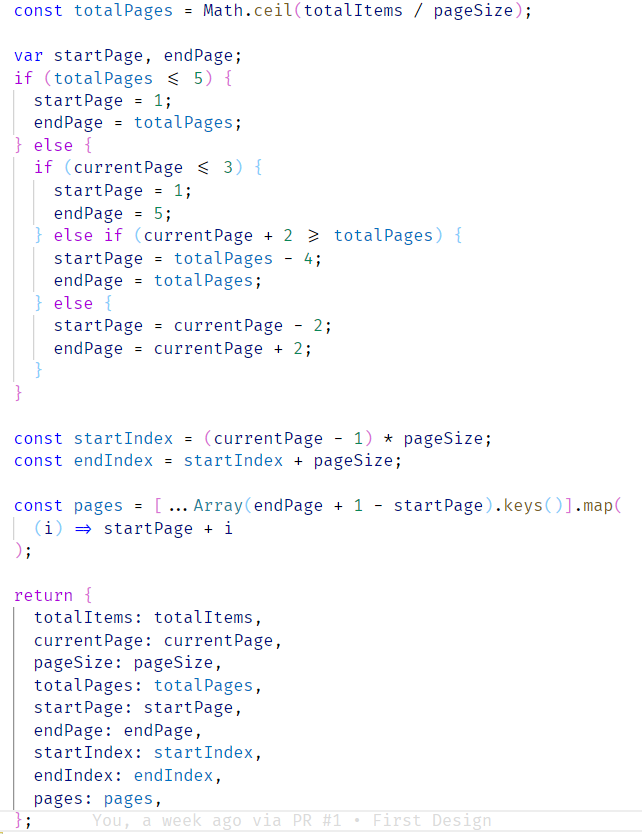


*Hình 28. Hiển thị pagination khi số lượng bài viết lớn hơn 0..*

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



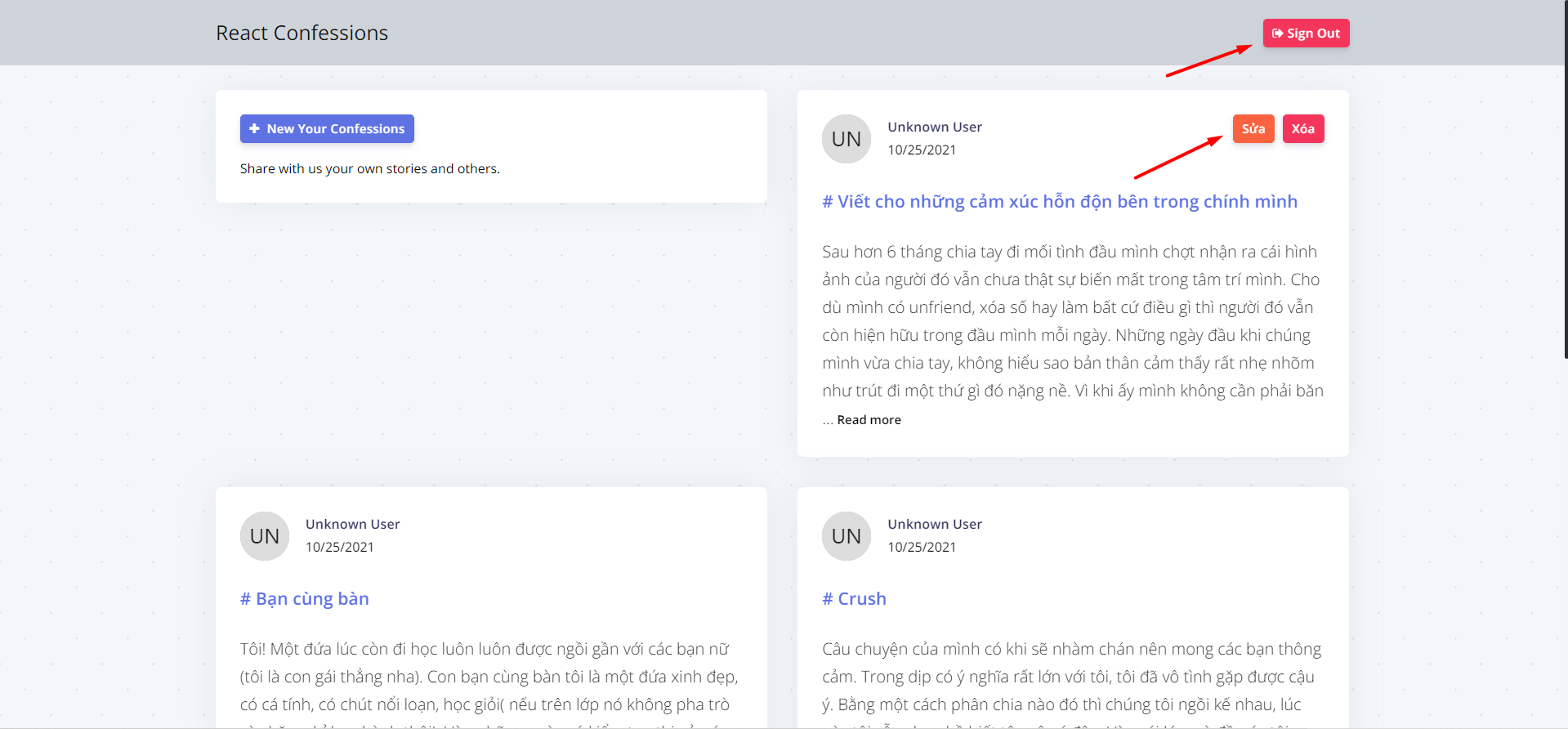
*Hình 29. Hiển thị Pagination cho bài viết.*



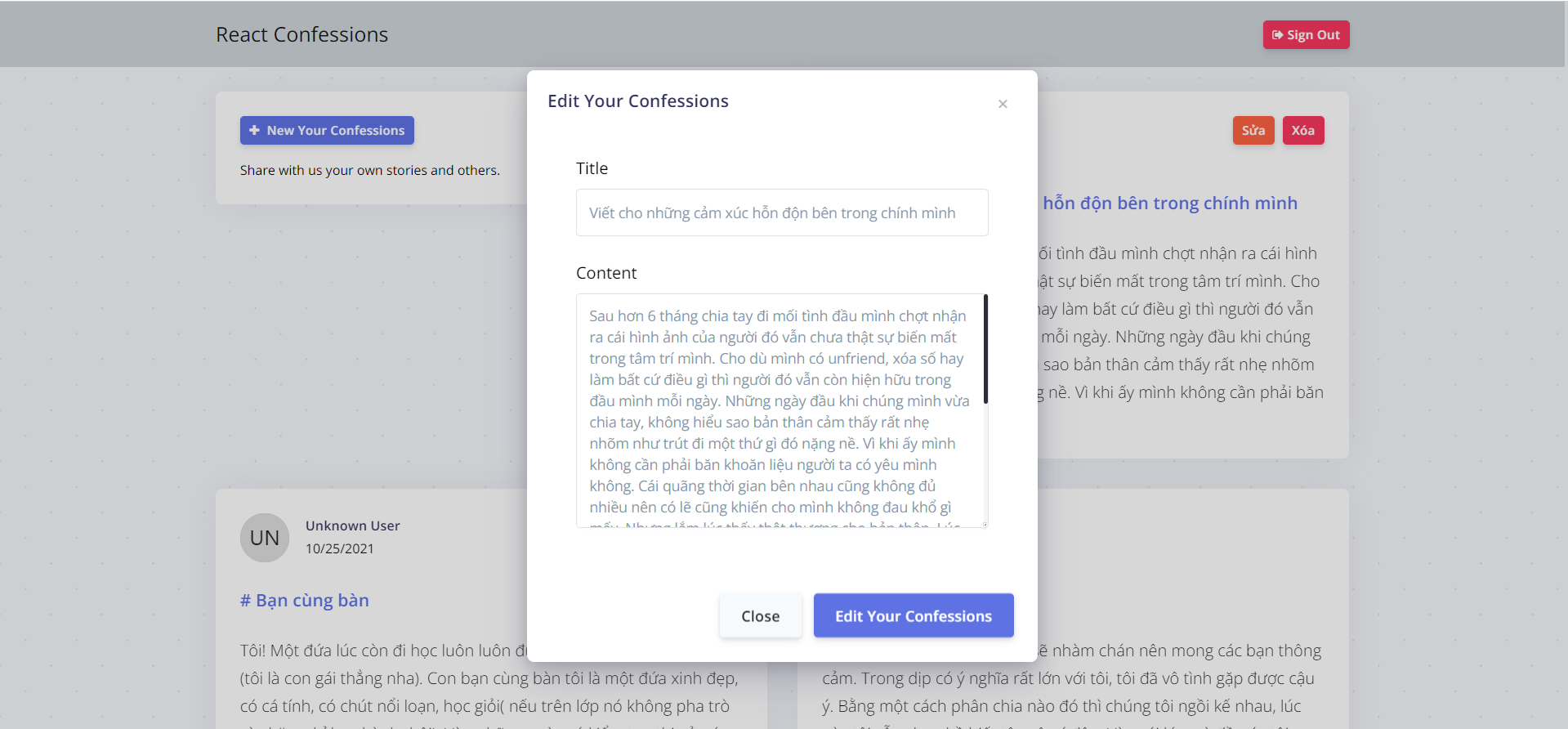
*Hình 30. Logic xử lý hiển thị pagination.*

### 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.

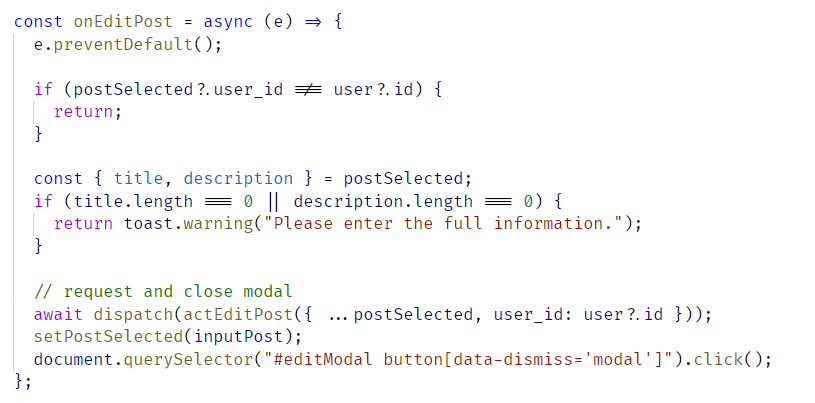


*Hình 31. Hiển thị sửa khi người dùng đăng nhập và đúng bài viết.*



*Hình 32. 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 33. Khi click vào nút sửa sẽ gọi đến function sửa.*

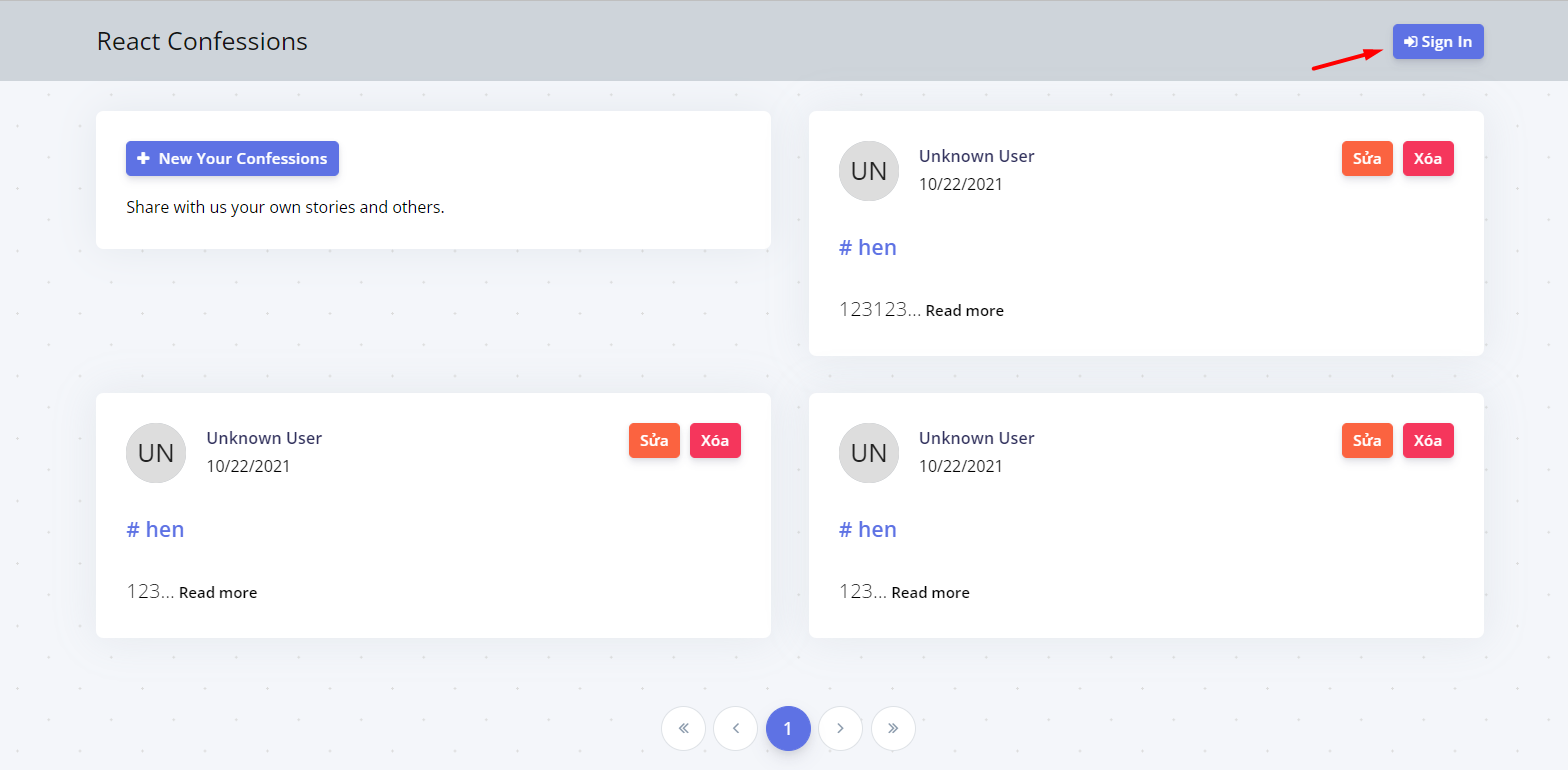
Ở trong Redux sẽ xử lý chức năng sửa bài viết, và load lại tất cả bài viết hiện tại.



*Hình 34. Xử lý sửa bài viết và load lại dữ liệu.*

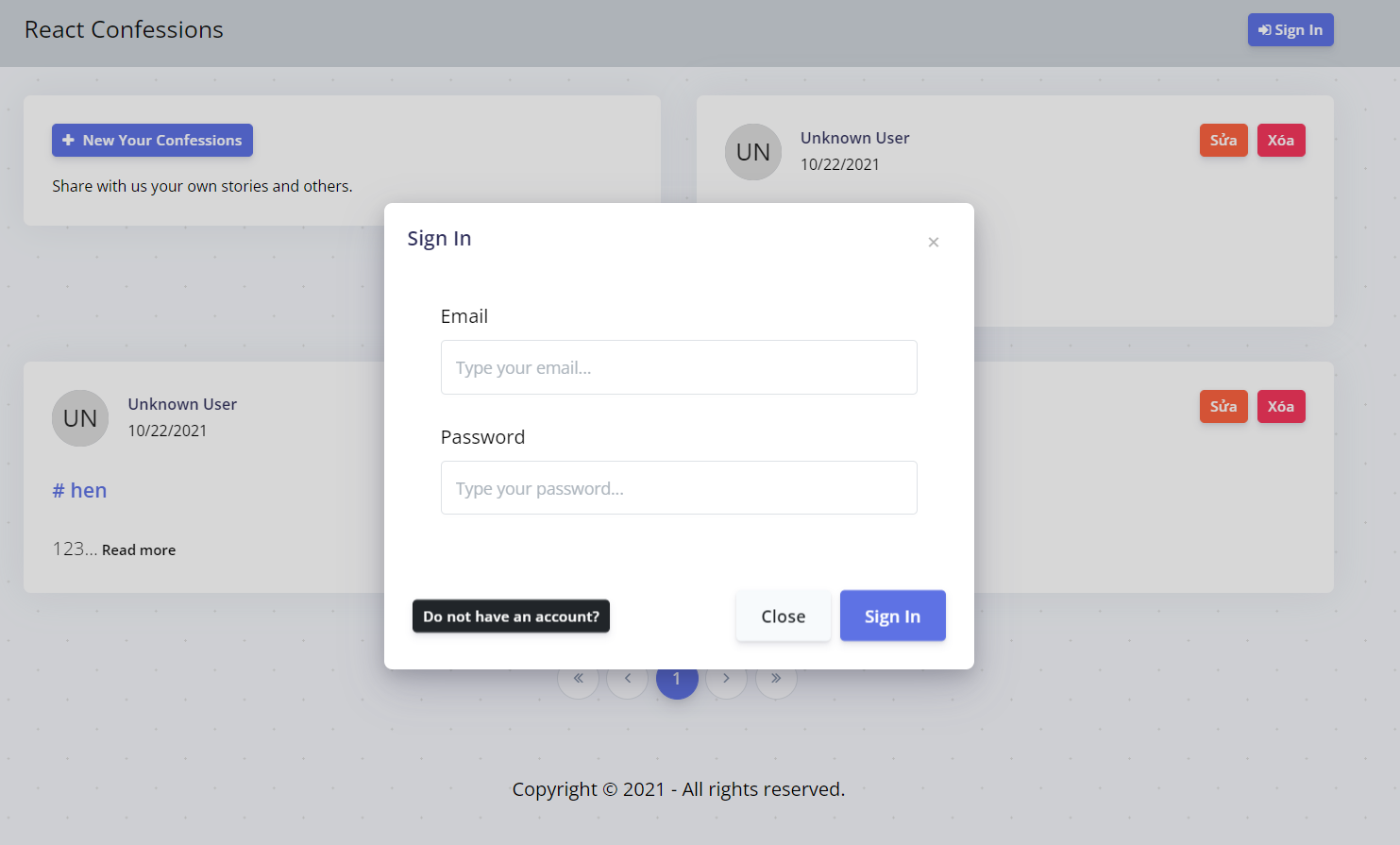
### 2.2.4. Đăng nhập confessions

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.



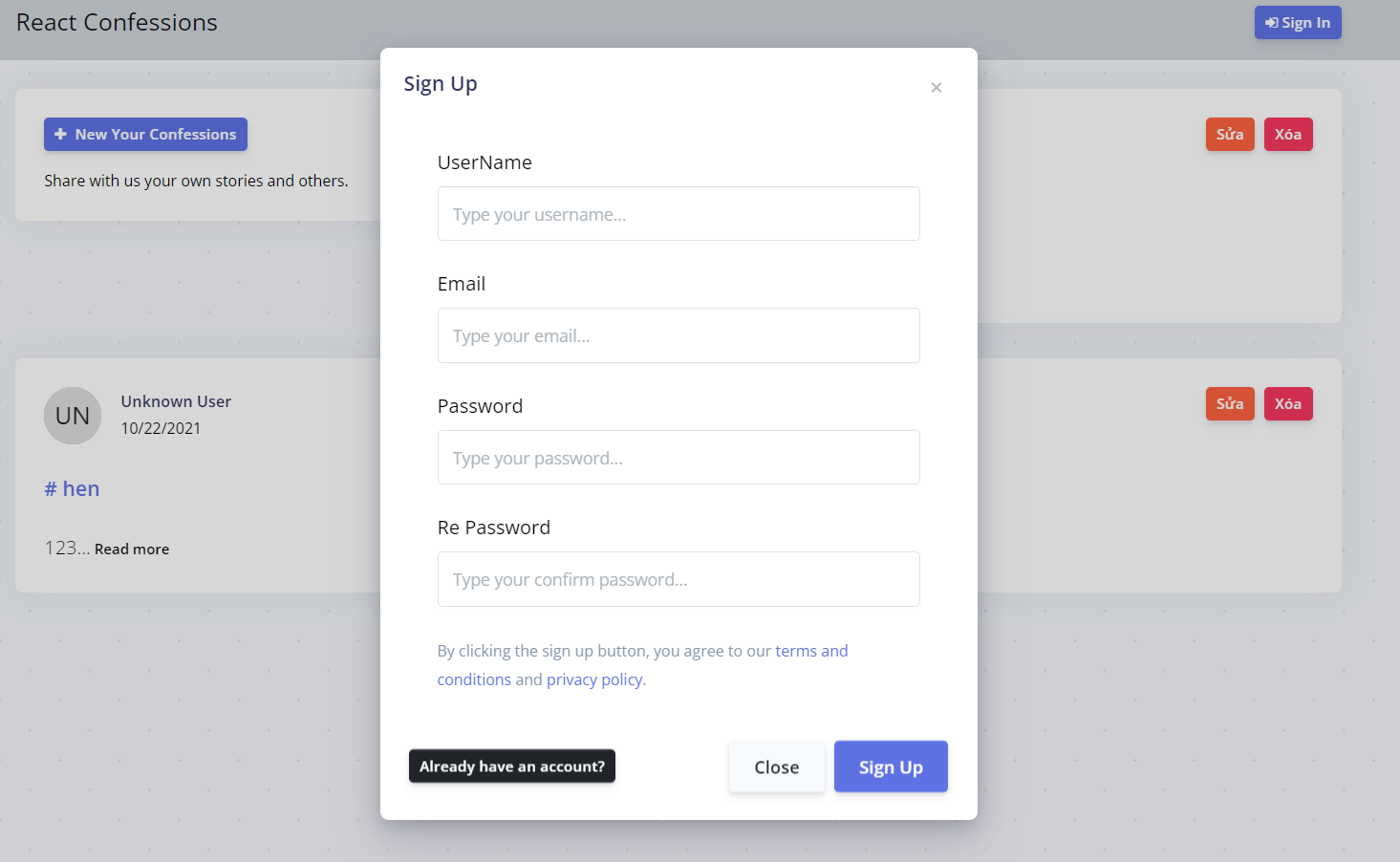
*Hình 35. Nơi hiển thị button Sign In.*

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.



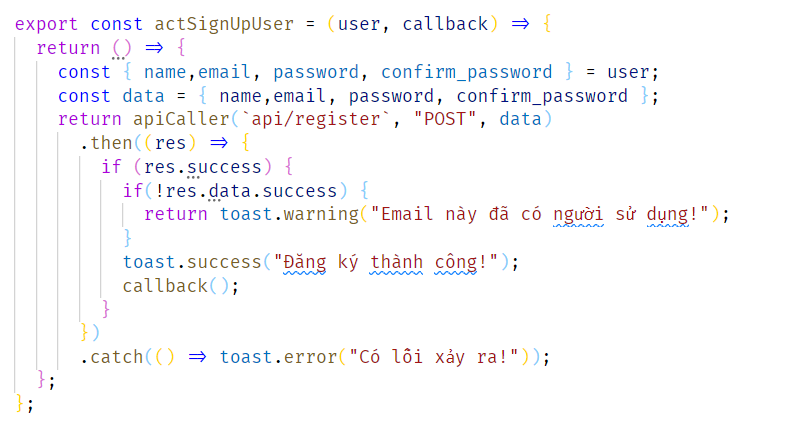
*Hình 36. Hiện model để đăng nhập.*

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 37. Hiện model để đăng ký.*

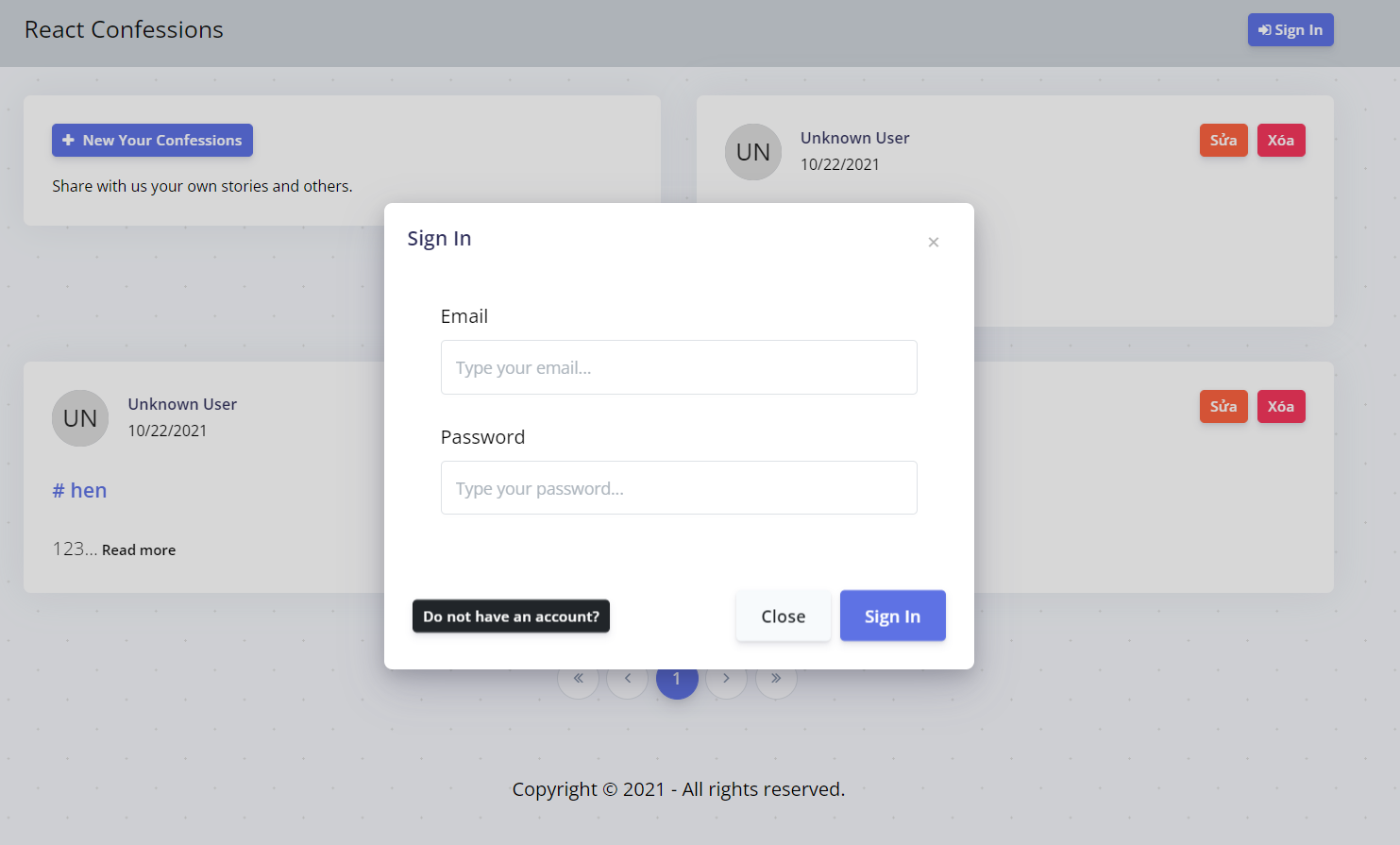
Ở model này người dùng chỉ nhập đầy đủ và đúng thông tin và nhấn button Sign Up web sẽ vào Redux và callApi để thực hiện chức năng đăng ký và thông báo kết quả. Nếu đăng ký thành công thì sẽ web sẽ navigation qua model Sign In.



*Hình 38. Hàm Sign Up action ở Redux.*

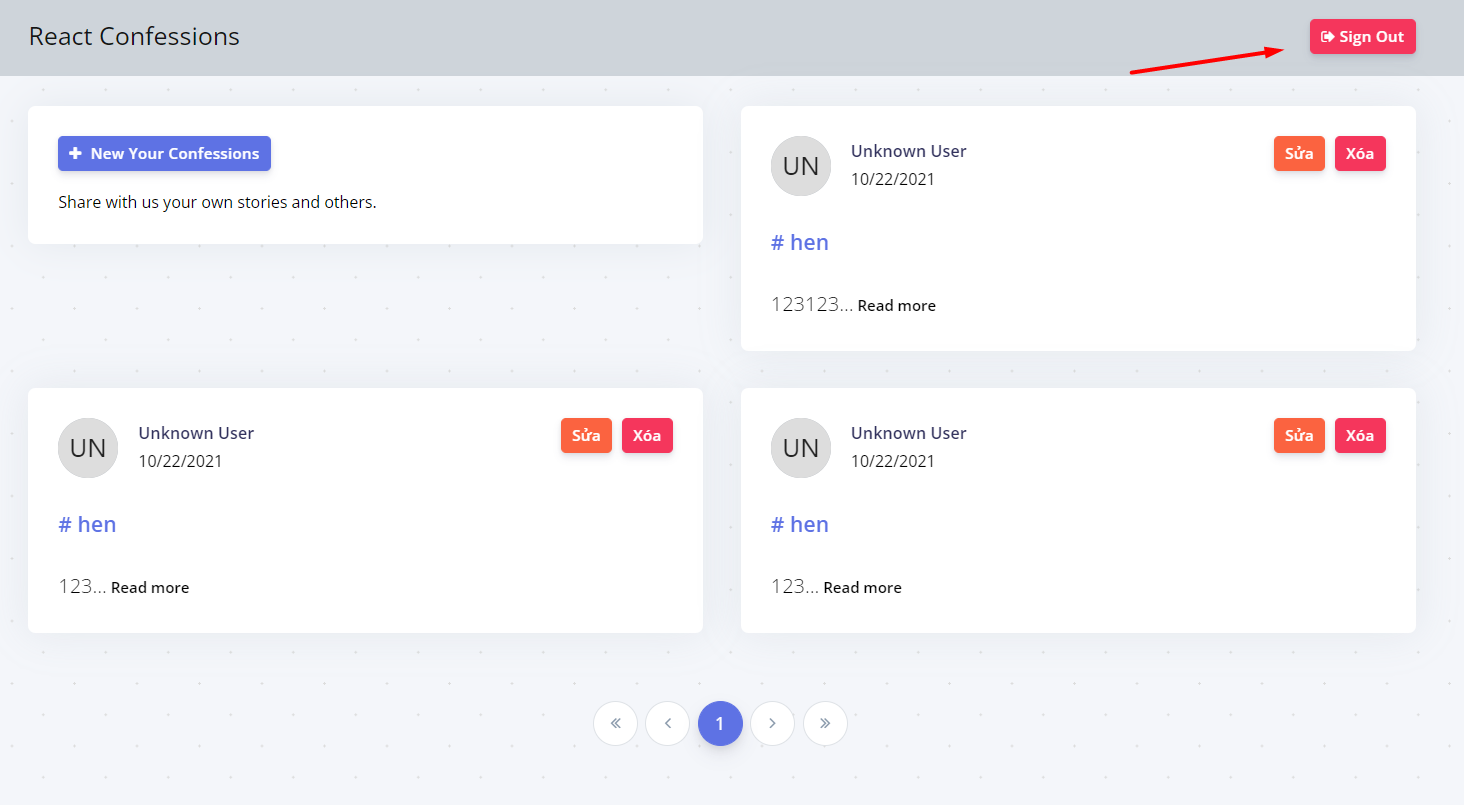
### 2.2.5. Đăng ký confessions

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.

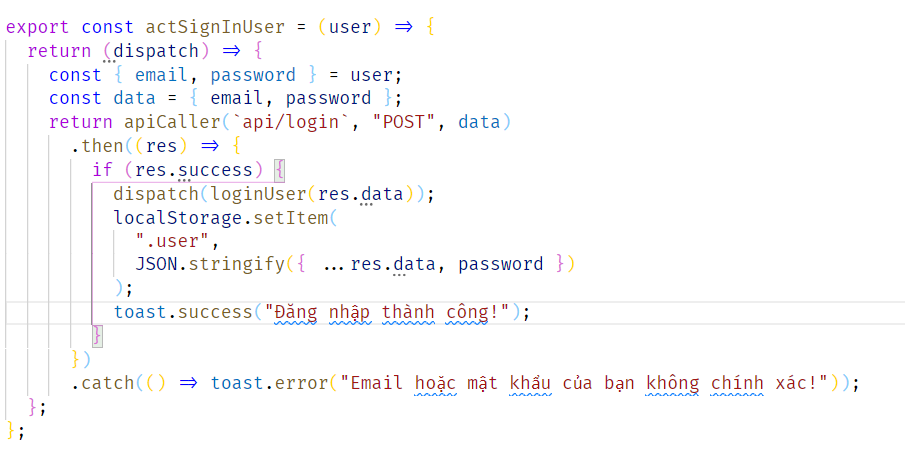


*Hình 39. Model Sign In.*

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 40. Sau đăng nhập thành công.*

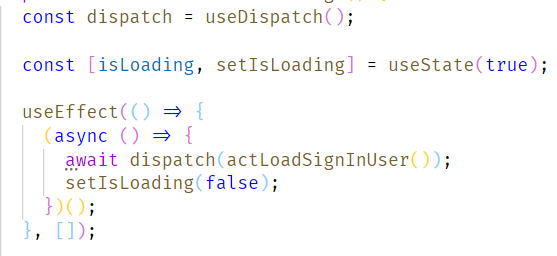


*Hình 41. Hàm Redux xử lý đăng nhập.*

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



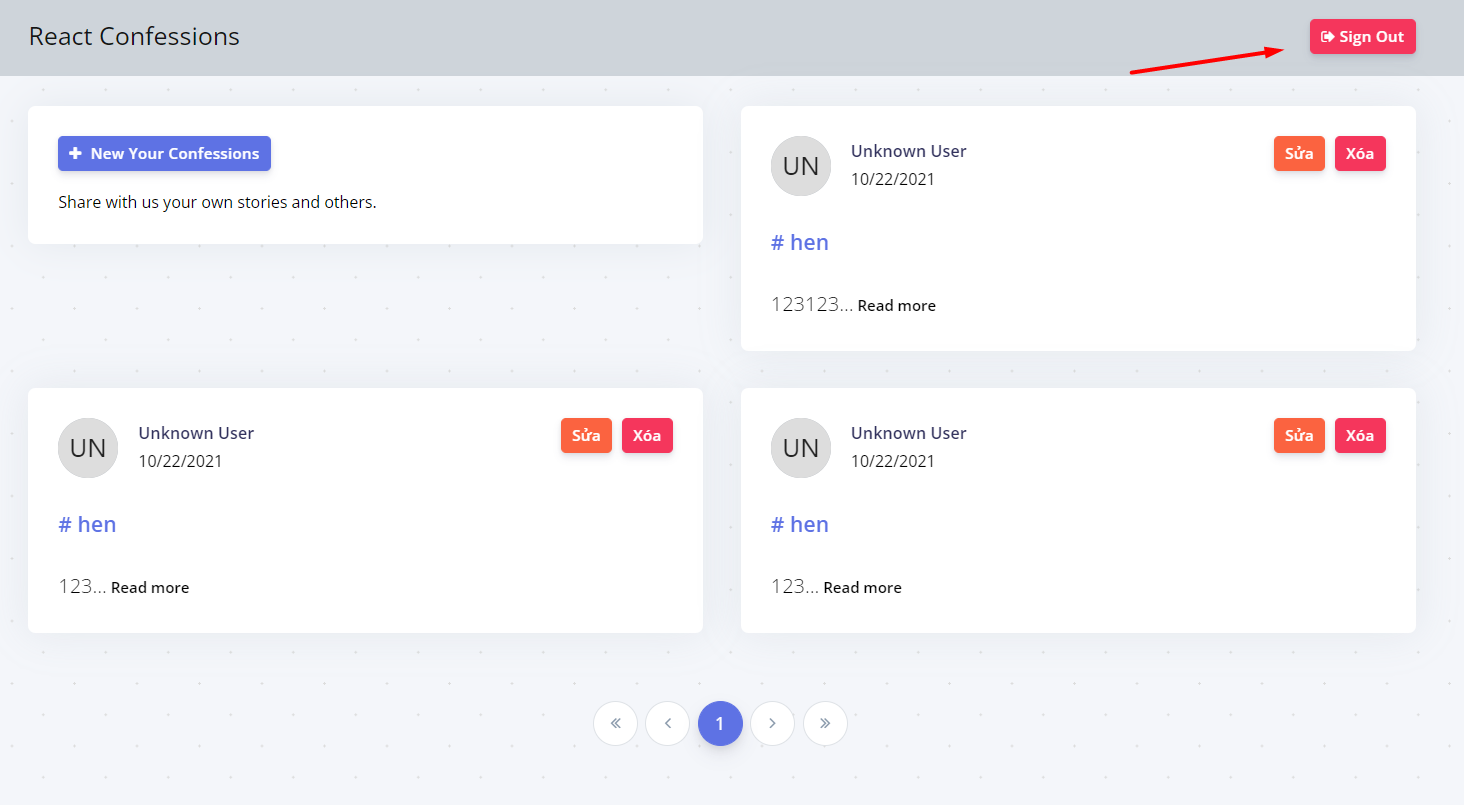
*Hình 42. Hàm sẽ thực hiện khi người dùng reload trang.*



*Hình 43. Hàm Redux loading đăng nhập.*

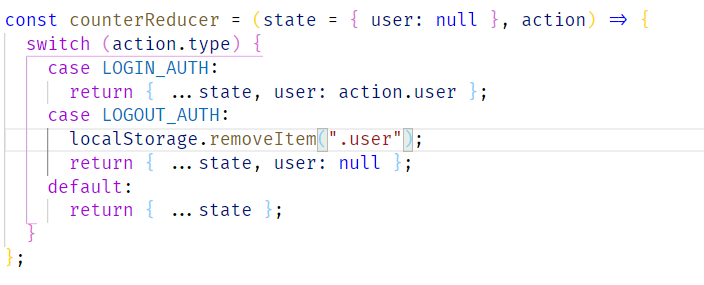
### 2.2.6. Đăng xuất confessions

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.

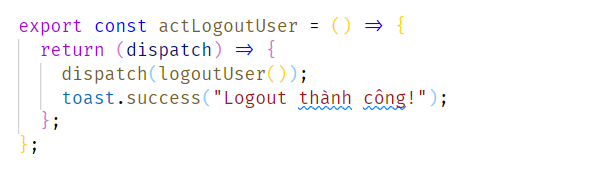


*Hình 44. Hiện button Sign Out.*

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 45. Ở hàm auth Reducer.*

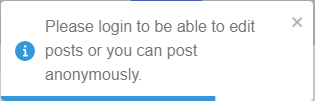


*Hình 46. Hàm Logout ở Redux.*

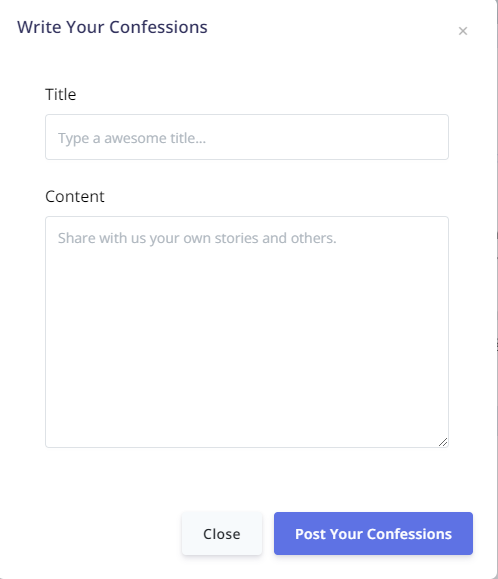
### 2.2.7. Thêm confessions

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:

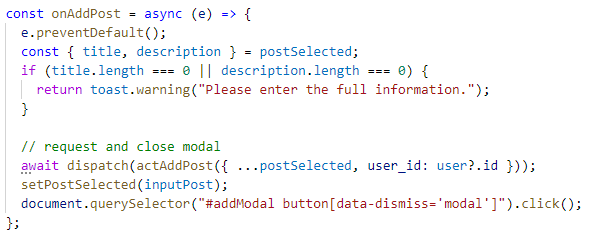


*Hình 47. HIện thông báo.*



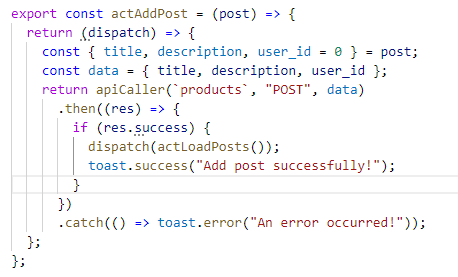
*Hình 48. Model Thêm Confessions.*

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 49. Khi click vào nút thêm sẽ gọi đến function thêm.*

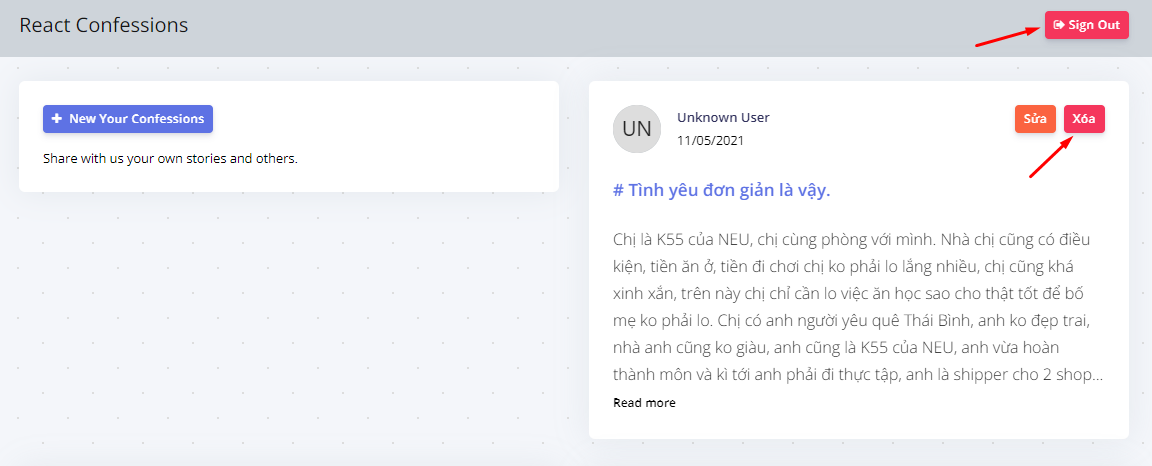
Ở trong Redux sẽ xử lý chức năng thêm bài viết, và load lại tất cả bài viết hiện tại.



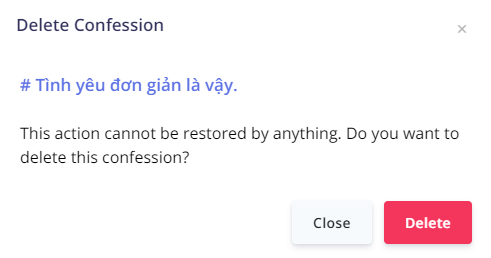
*Hình 50. Xử lý thêm bài viết và load lại dữ liệu.*

### 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.

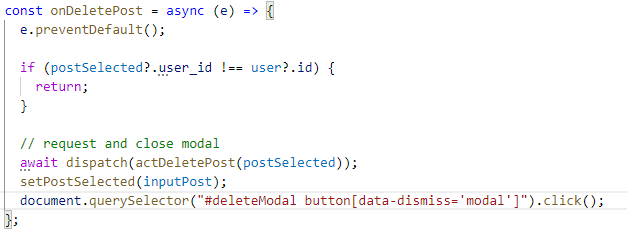


*Hình 51. Hiển thị xóa khi người dùng đăng nhập và đúng bài viết.*



*Hình 52. 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 53. Khi click vào nút xóa sẽ gọi đến function xóa.*

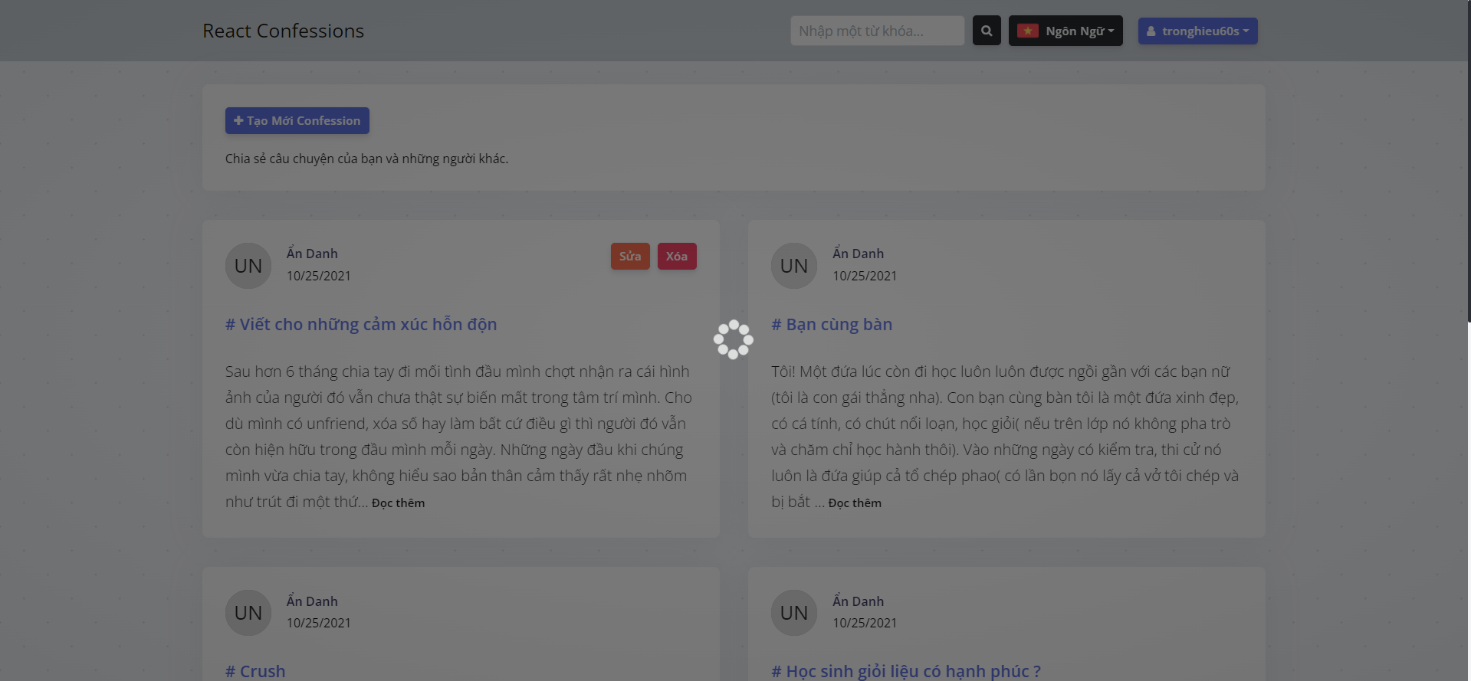
Ở trong Redux sẽ xử lý chức năng xóa bài viết, và load lại tất cả bài viết hiện tại.



*Hình 54. 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 55. 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 common Reducer 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.



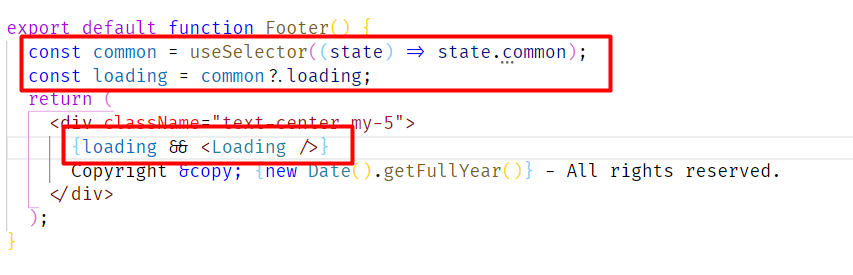
*Hình 56. Tạo 2 case action.*

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 57. Dispatch action trong một function xử lý khác.*

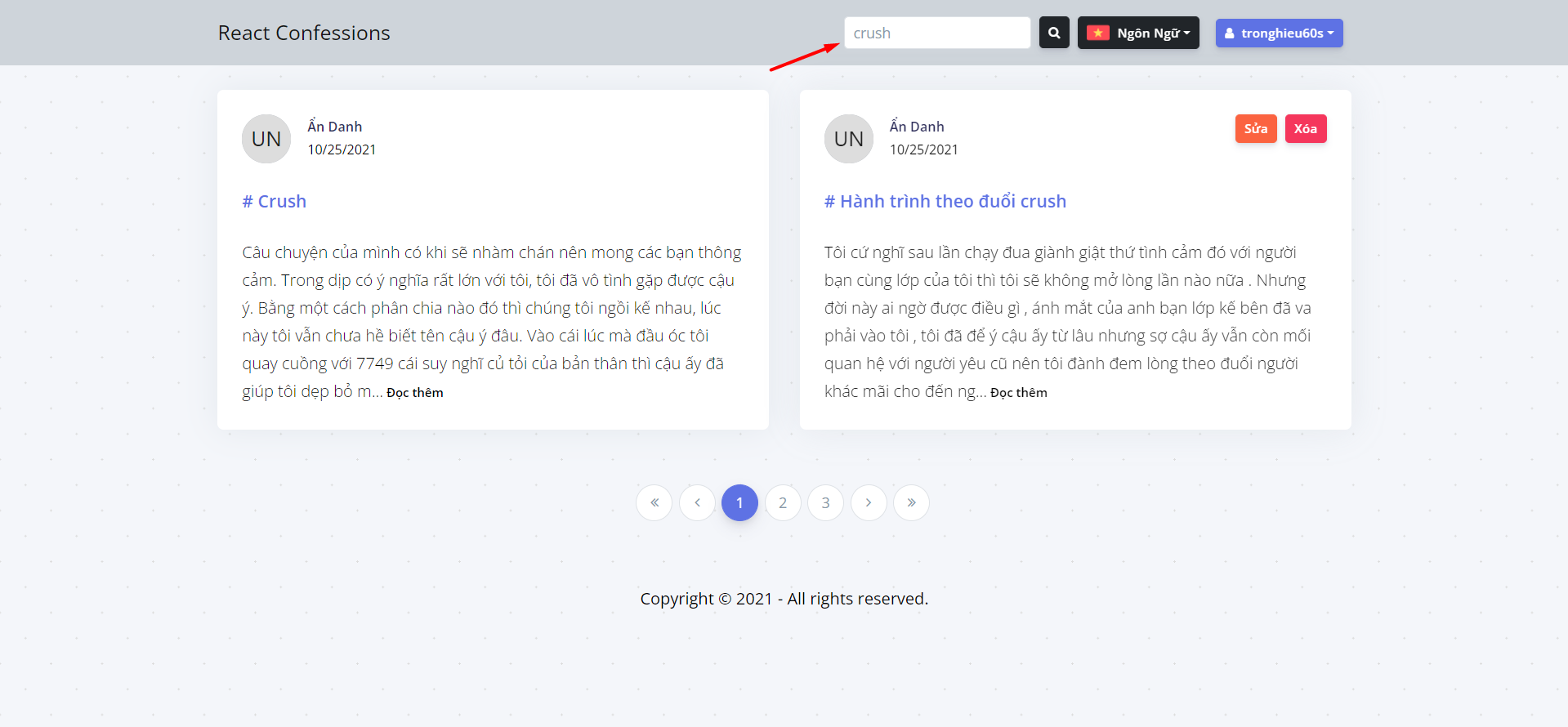
Khi đã gắn vào trong hàm call API thì sau đó chúng tra sẽ ghi làm để set trạng thái cho loading là gì để có thể hiển thị ra màn hình



*Hình 58. Gọi giá trị loading từ Redux và sử dụng.*

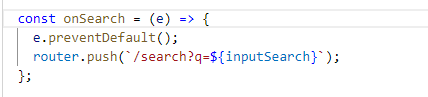
### 2.2.10. Tìm kiếm bài viết

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



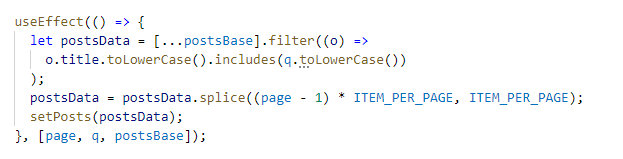
*Hình 59. Giao diện sau khi tìm kiếm.*

Khai báo nút Search:



*Hình 60. Khai báo onSearch.*

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



*Hình 61. use Effect hiển thị mục tìm kiếm.*

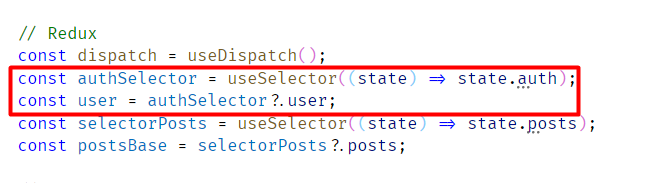
### 2.2.11. Trang chi tiết người dùng

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 62. Lọc các bài viết của người dùng đã tạo.*

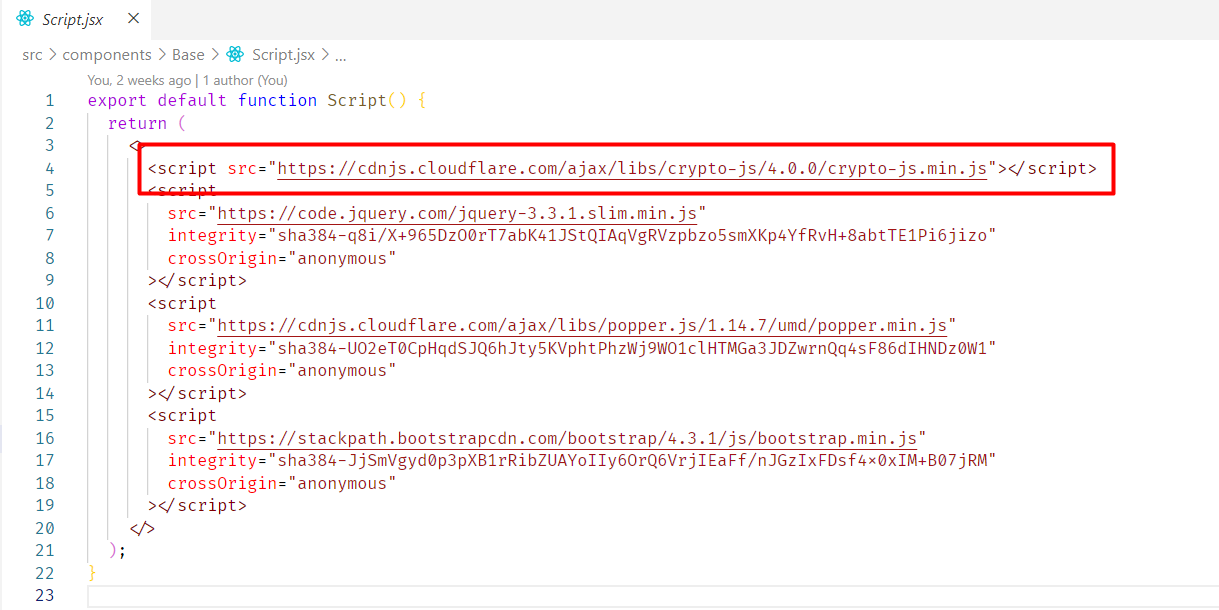
Để hiển thị thông tin cá nhân người dùng ra ta chỉ cần gọi redux mà đã lưu người dùng đăng nhập trước đó.



*Hình 63. 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 dẫn hình ảnh.

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



*Hình 64. Nhúng thư viện mã hóa md5 vào dự án.*

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



*Hình 65. Hiển thị hình ảnh người dùng sau khi hash ra mã md5.*