

---

# ĐỀ TÀI

## Quản lí hệ thống nhà thuốc với react-native

---

*Giảng viên Cô Phan Nguyệt Minh*

*Thành viên Nguyễn Thanh Thọ*

**Dương Nhật Huy**



# MỤC LỤC

## CHƯƠNG 1: Giới thiệu đề tài và các chức năng chính

1.1	Giới thiệu đề tài và các chức năng chính .....	4
1.2	Yêu cầu thực hiện .....	5

## CHƯƠNG 2: Phân tích và đặc tả yêu cầu

2.1.	Chức năng đăng nhập bằng gmail .....	6
2.2.	Chức năng quản lí thông tin .....	6
2.2.	Chức năng đăng kí nhà phát triển .....	7
2.3.	Chức năng quản lí sản phẩm .....	7
2.4.	Chức năng search sản phẩm .....	8

## CHƯƠNG 3: Thiết kế chương trình

3.1.	Thiết kế giao diện, thiết kế xử lý.....	9
------	-----------------------------------------	---

## CHƯƠNG 4: CÁC KIẾN TRÚC TRONG PHẦN MỀM

4.1.	Kiến trúc Component trong ReactJS.....	28
4.2.	Kiến trúc tổng thể Redux .....	40

## CHƯƠNG 5: Chỉ tiêu thực hiện và kết quả đạt được

# CHƯƠNG 1:

## Giới thiệu đề tài và các chức năng chính

### 1. Giới thiệu đề tài và các chức năng chính

#### 1.1. Lý do phát triển

##### *Hiện trạng về mặt thực tế:*

Hiện nay có nhiều nhà thuốc hoạt động mang tính chất tự phát, nhỏ lẻ. Nhà thuốc thì nhiều nên vấn đề cạnh tranh luôn rất cao. Những nhà thuốc nhỏ thường bị những nhà thuốc lớn hơn lấn áp do không kinh phí để mở rộng kinh doanh. Nhiều nhà thuốc tuy nhỏ nhưng rất chất lượng bán những loại thuốc rẻ và bổ rất bổ ít nhưng không được người dùng biết đến. Người dùng luôn có tư tưởng là nhà thuốc lớn thì sẽ có thể mua được những loại thuốc tốt hơn và giá tiền thì cũng xứng đáng.

Tuy nhiên, những nhà thuốc nhỏ thì họ vẫn có được những sản phẩm mà khiến cho người dùng có thể an tâm, ngoài ra họ không cần chịu nhiều chi phí về nhân lực, lẫn mặt bằng, các loại thuế khác nên giá cả của họ cũng rất phải chăng.

##### *Cách quản lý hệ thống nhà thuốc hiệu quả*

- Hỗ trợ các nhà thuốc về kho lưu trữ các loại thuốc, thực phẩm chức năng.
- Quảng bá sản phẩm thông qua app tới người dùng.
- Người dùng có thể tìm các loại thuốc hay các loại thực phẩm chức năng thông qua app.
- Người dùng có thể xem được thông tin về các loại thuốc, thông tin địa chỉ của nhà thuốc.

#### 1.2. Yêu cầu chức năng:

##### *1.2.1. Chức năng dành cho người dùng*

- Đăng nhập bằng tài khoản gmail: chức năng này hỗ trợ cho người dùng có thể tạo tài khoản trên firebase và thực hiện đăng nhập bằng tài khoản gmail.
- Tìm kiếm thuốc:
  - Tìm kiếm ra nhà thuốc cung cấp thuốc hoặc thực phẩm chức năng.

- Lọc tất cả sản phẩm từ firebase theo từ khóa. Và hiển thị các thông tin về tên thuốc, hình ảnh, loại thuốc, giá tiền, tên nhà thuốc và địa chỉ nhà thuốc.
- Có thêm chức năng đánh giá loại thuốc của nhà thuốc đó (điều kiện là phải đăng nhập trước).
- Tìm kiếm nhà thuốc: Hiển thị được danh sách đang bán ở nhà thuốc đó.
- Quản lí được thông tin các nhân: người dùng có thể tự do thêm xóa sửa các thông tin về họ tên, ngày sinh, và kích hoạt chế độ nhà phát triển.

#### 1.2.2. **Chức năng dành cho nhà thuốc**

- Người dùng có thể kích hoạt chức năng nhà phát triển và có thể kinh doanh đăng các sản phẩm đang bán trên app.
- Quản lí tài khoản nhà thuốc: ngoài thông tin của người dùng thì nhà thuốc còn có thể thêm xóa sửa các thông tin về nhà thuốc như là địa chỉ, và tên nhà thuốc
- Quản lí kho: nhà thuốc có thể thêm xóa sửa các loại thuốc, thực phẩm chức năng mà nhà thuốc muốn đăng bán

#### 1.2.3. **Bảng phân công công việc nhóm**

Bước 1: Xác định và đặc tả yêu cầu phần mềm

Bước 2: Xây dựng cơ sở dữ liệu

Bước 3: Thiết kế giao diện

Bước 4: Thiết kế xử lý

Bước 5: Kiểm thử phần mềm

➔ Tổng thời gian: 1 tháng rưỡi

#### *Phân chia công việc nhóm*

Nhiệm vụ	Bảng phân công
Thiết kế xử lý	Thanh Thọ
thiết kế giao diện	Thanh Thọ, Nhật Huy
Phân luồng dữ liệu	Thanh Thọ
Thiết kế database	Thanh Thọ, Nhật Huy
Tìm kiếm những plu-in	Thanh Thọ
Kiểm thử phần mềm	Nhật Huy
Viết báo cáo	Thanh Thọ, Nhật Huy

*Kế hoạch làm việc nhóm:*

- Mỗi tuần họp 1 buổi
- Thời gian: Sau buổi học hoặc sáng thứ 7

*Nhiệm vụ:* trưởng nhóm tìm hiểu những công nghệ hỗ trợ cho react-native về việc quản lý app bán hàng sau đó phổ biến thông tin lại cho các thành viên

*Công nghệ xử dụng:* react-native, react-native-element, react-navigation, redux, react-redux, redux-action, firebase, adobe XD (hỗ trợ design app)

## CHƯƠNG 2:

### Phân tích và đặc tả yêu cầu

#### 2.1. Chức năng đăng nhập bằng gmail.

##### THÔNG TIN CHỨC NĂNG ĐĂNG NHẬP BẰNG GMAIL

Tên chức năng	Đăng nhập bằng Gmail.
Mô tả	Người dùng có thể dùng tài khoản gmail để đăng kí tài khoản đăng nhập vào app.
Tác nhân	Người dùng.
Điều kiện trước	Phải có tài khoản gmail mới có thể đăng nhập.
Điều kiện sau	Không có.
Ngoại lệ	Không có.
Các yêu cầu đặc biệt	Không có.

#### 2.2. Chức năng quản lí thông tin người dùng.

##### QUẢN LÝ THÔNG TIN NGƯỜI DÙNG

Tên chức năng	Quản lí thông tin người dùng
Mô tả	Chức năng này hỗ trợ cho người dùng có thể quản lí các thông tin về tài khoản của mình
Tác nhân	Người dùng
Điều kiện trước	Phải đăng nhập vào app bằng tài khoản gmail và.
Điều kiện sau	Không có.
Ngoại lệ	Không có.
Các yêu cầu đặt biệt	Không có.

### 2.3. Chức năng đăng kí nhà phát triển.

#### THÔNG TIN CHỨC NĂNG ĐĂNG KÍ NHÀ PHÁT TRIỂN

Tên chức năng	Đăng kí nhà phát triển bán thuốc
Mô tả	Chức năng này hỗ trợ cho người dùng có thể kích hoạt chế độ nhà phát triển để có thể quản lý các thông tin về nhà thuốc và bán hàng.
Tác nhân	Nhà phát triển.
Điều kiện trước	Phải đăng nhập vào app bằng tài khoản gmail và hoàn thành các thông tin về người dùng.
Điều kiện sau	Không có.
Ngoại lệ	Không có.
Các yêu cầu đặt biệt	Nhà phát triển có thể đăng kí chế độ nhà thuốc nhưng vẫn có thể để product là null.

### 2.4. Chức năng quản lí sản phẩm cho nhà thuốc.

#### THÔNG TIN CHỨC NĂNG QUẢN LÝ SẢN PHẨM

Tên chức năng	Quản lí sản phẩm cho nhà thuốc
Mô tả	Người dùng có thể thêm xóa sửa các thông tin về sản phẩm như là: hình ảnh, tên thuốc, giá tiền, hạn dùng, ngày sản xuất.
Tác nhân	Nhà phát triển.
Điều kiện trước	Phải kích hoạt chế độ nhà thuốc trước thì mới có thể thêm xóa sửa các loại thuốc.
Điều kiện sau	Không có.
Ngoại lệ	Không có.
Các yêu cầu đặt biệt	Không có.

## 2.5. Chức năng search sản phẩm.

### THÔNG TIN CHỨC NĂNG SEARCH SẢN PHẨM

<b>Tên chức năng</b>	Tìm kiếm các loại thuốc được đăng bán bởi các nhà thuốc.
<b>Mô tả</b>	Người dùng có thể tìm kiếm các thông tin về các loại thuốc được đăng bán như: hình ảnh, tên thuốc, giá tiền, tên nhà thuốc và địa chỉ
<b>Tác nhân</b>	Người dùng.
<b>Điều kiện trước</b>	Không có.
<b>Điều kiện sau</b>	Không có.
<b>Ngoại lệ</b>	Không có.
<b>Các yêu cầu đặc biệt</b>	Người dùng có thể không cần đăng nhập vẫn có thể sử dụng chức năng này



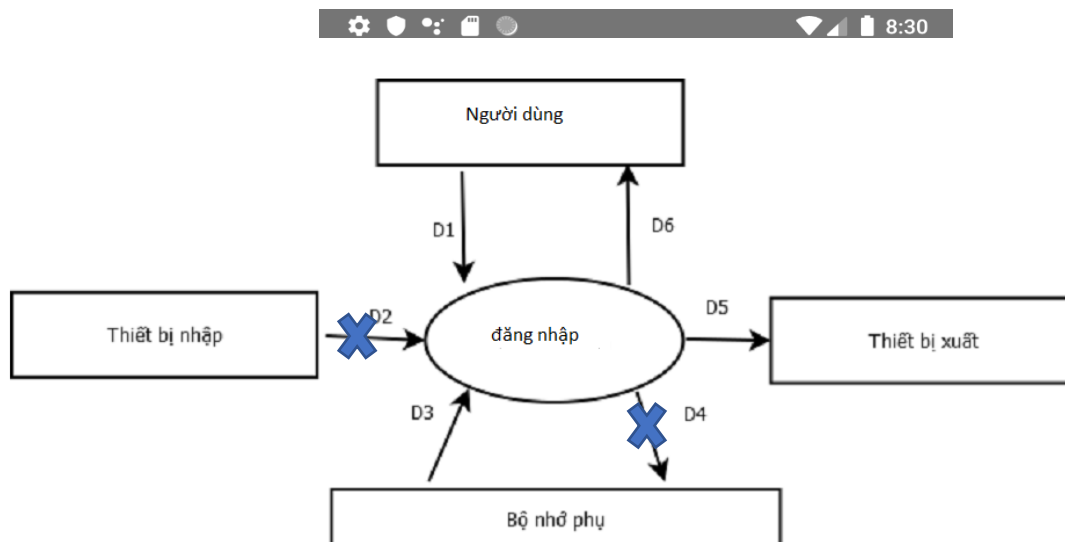
## CHƯƠNG 3:

### Thiết kế chương trình Quản lí tiệc cưới

#### 3.1 Thiết kế giao diện và thiết kế xử lý

##### 3.1.1. Form Login

##### a) Màn hình xử lý



##### b) Biểu đồ luồng xử lý chức năng

**D1: Nhập tài khoản, mật khẩu**

**D2: Không có**

**D3: Trả về thông tin đầy đủ username, birthday, email**

**D4: không có**

**D5: không có**

**D6: không có**

**Thuật toán**

**B1: Nhập thông tin từ D1**

**B2: Kết nối database**

**B3: Database nhận D1 từ người dùng**

**B4: Kiểm tra tài khoản có đúng hay không**

**B5: Nếu sai chuyển tới B7**

**B6: Nếu đúng trả về D3**

**B7: Đóng database**

**B8: Kết thúc**

c) Danh sách xử lý

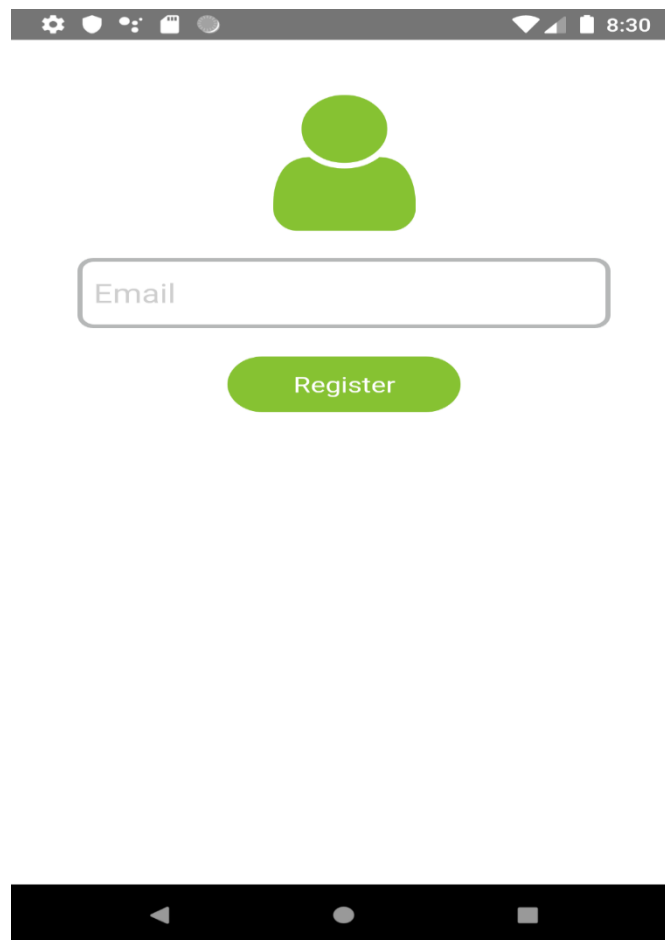
Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nhập email	Điền thông tin email đã đăng ký	Không có	Bắt buộc điền
Nhập password	Điền thông tin password đã đăng ký	Không có	Bắt buộc điền
Nút login	Kiểm tra thông tin email và password có hợp lệ hay không	Nếu email và password chính xác sẽ chuyển sang màn hình user. Nếu sai sẽ báo nhập email hoặc password sai	Bắt buộc ấn
Nút register	Chuyển qua màn hình đăng ký	Không có	Không có

3.1.2. Màn hình đăng kí

a) Thông tin chức năng đăng kí tài khoản:

- Người dùng có thể đăng kí tài khoản mới bằng cách nhập vào tài khoản gmail hiện tại. Tất cả thông tin về gmail và password phải hoàn toàn chính xác

b) Màn hình xử lý



c) Danh sách xử lý

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nhập email	Điền thông tin về tài khoản gmail để có thể đăng ký	Gmail phải hoàn toàn xác thực	Bắt buộc điền
Nhập password	Điền thông tin về password muốn đăng kí đăng nhập vào ứng dụng	Phải nhập thông tin gmail xong thì mới có thể nhập đc password	Bắt buộc điền
Nút register	Xác thực tài khoản đăng kí trên app	Gmail phải tồn tại	Nếu đăng kí thành công sẽ xuất ra thông báo đăng kí thành công. Nếu không sẽ báo đăng kí thất bại

### 3.1.2. Màn hình quản lí User

#### a) Thông tin chức năng quản lí user

- Chức năng quản lí user giúp quản lí các thông tin có liên quan tới người dùng.

#### b) Màn hình xử lý



Manager User >

DrugStore Mode >

LogOut



#### c) danh sách xử lý

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nút Manager User	Chuyển qua màn hình Manager User	Đã đăng nhập thành công	Không có
Nút DrugStore Mode	Chuyển qua màn hình DrugStore Mode	Đã đăng nhập thành công	Không có

Nút LogOut	Đăng xuất tài khoản	Đã đăng nhập thành công	Không có
------------	---------------------	-------------------------	----------

### 3.1.3. Màn hình Manager User

- a) Thông tin chức năng quản lý thông tin người dùng
- Giúp người dùng có thể quản lý tất cả thông tin của mình: Người dùng có thể cập nhật lại các thông tin như UserName, BirthDay
- b) Màn hình xử lý



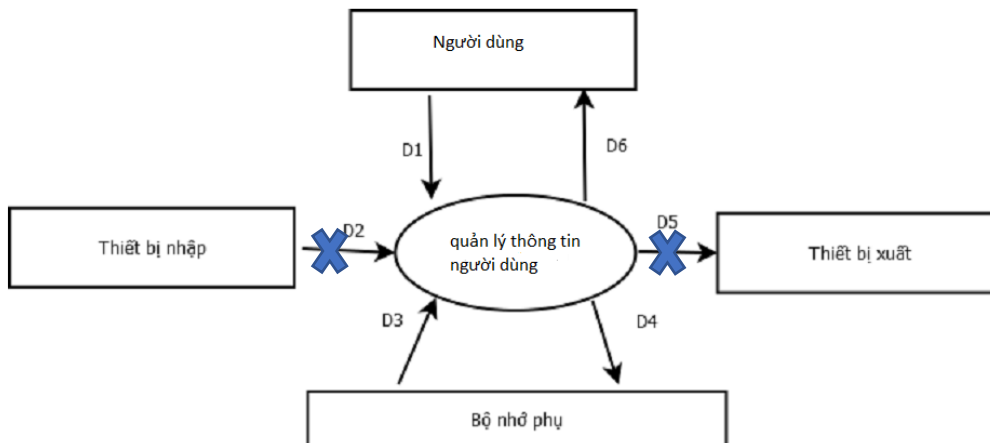
User Name: ngoc hien

BrithDay: 05/06/99

Email: thotpbank050699@gmail.com

Update Info

c) Sơ đồ luồng



**D1: Nhập thông tin người dùng**

**D2: Không có**

**D3: kiểm tra thông tin từ người dùng nhập lên**

**D4: D1**

**D5: không có**

**D6: không có**

**Thuật toán**

**B1: Nhập thông tin từ D1**

**B2: Kết nối database**

**B3: Kiểm tra thông tin từ người dùng theo D1. Nếu không hợp lệ sẽ chuyển tới bước 5**

**B4: Lưu D1 vào database**

**B5: Đóng database**

**B6: Kết thúc**

d) Danh sách xử lí

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nút back	Quay lại màn hình user	Không có	Không có
Nhập User Name	Điền user name muốn cập nhập	Không có	Không có
Nhập Birthday	Điền birthday muốn cập nhập	Không có	Không có
Nút Update Info	Cập nhập lại thông tin tài khoản người dùng	Không có	Không có

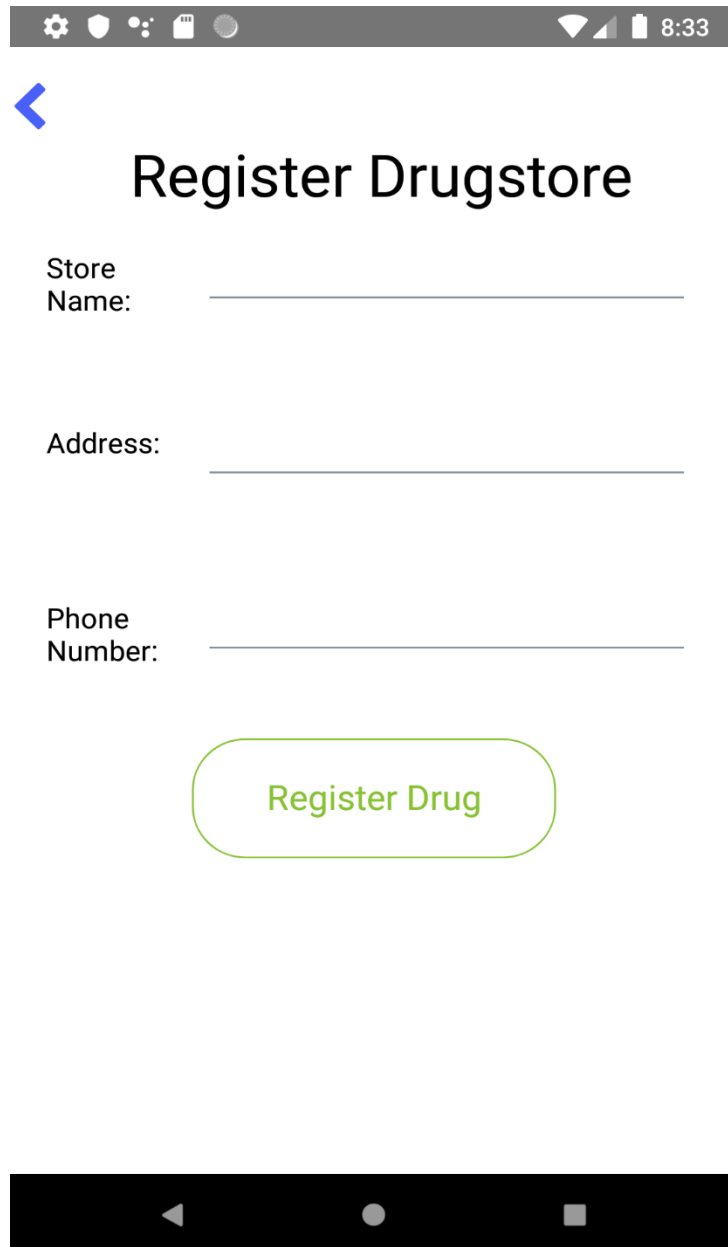
### 3.1.4. Màn hình DrugStore

#### I. Màn hình DrugStore chưa đăng kí chế độ nhà phát triển

a) Thông tin về DrugMode

- Người dùng có thể đăng ký thông tin về nhà thuốc

b) Màn hình xử lý



c) Danh sách xử lý

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nút back	Quay lại màn hình User	Không có	Không có
Nhập Store Name	Điền tên nhà thuốc muốn đăng ký	Không có	Không có
Nhập Address	Điền địa chỉ nhà thuốc muốn đăng ký	Không có	Không có
Nhập Phone Number	Điền SDT nhà thuốc muốn đăng ký	Không có	Không có
Nút Register Drug	Đăng ký thông tin nhà thuốc	Không có	Không có

## II. Màn hình DrugStore đã đăng kí chế độ nhà phát triển

### a) Thông tin về DrugMode

- Người dùng có thể cập nhập lại thông tin nhà thuốc

### b) Màn hình xử lý

Update Drugstore

Store Name:

Address:

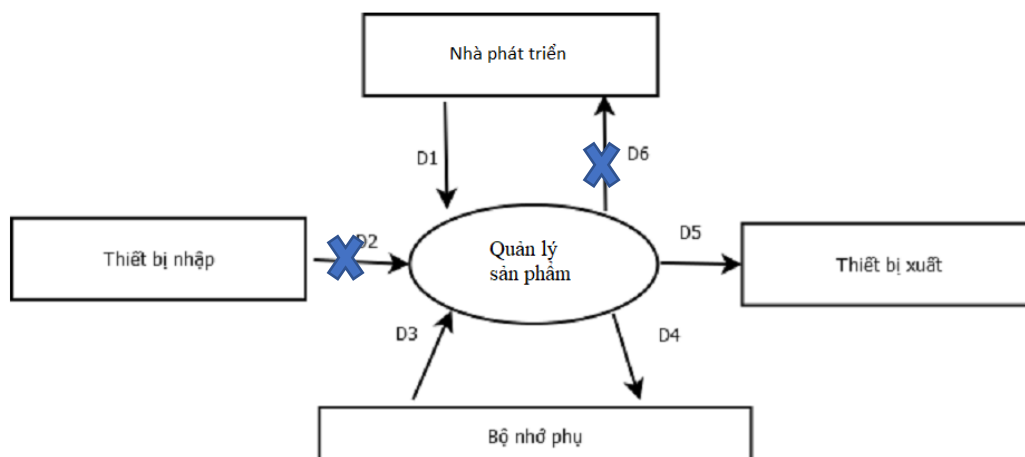
Phone Number:



c) Danh sách xử lý

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nút back	Quay lại màn hình User	Không có	Không có
Nhập Store Name	Điền tên nhà thuốc muốn cập nhật	Không có	Không có
Nhập Address	Điền địa chỉ nhà thuốc muốn cập nhật	Không có	Không có
Nhập Phone Number	Điền SDT nhà thuốc muốn cập nhật	Không có	Không có
Nút Update Drug	Cập nhật thông tin nhà thuốc	Không có	Không có
Nút Manager Product	Chuyển qua màn hình Manager Product	Không có	Không có

### III. Sơ đồ luồng phân tích



**D1:** Nhập thông tin sản phẩm  
**D2:** Không có  
**D3:** Danh sách thông tin sản phẩm  
**D4:** D1  
**D5:** không có  
**D6:** D3

#### Thuật toán

**B1:** Nhập thông tin từ D1  
**B2:** Kết nối database  
**B3:** Database nhận D1 từ người dùng  
**B4:** Lưu D1 vào database  
**B5:** Trả về D3 cho nhà phát triển  
**B5:** Đóng database  
**B6:** Kết thúc

### 3.1.5. Màn hình quản lí sản phẩm

#### a) Thông tin chức năng quản lí sản phẩm

- Người dùng với chế độ nhà thuốc có thể quản lí tất cả các sản phẩm có trong nhà thuốc và có thể thực hiện các thao tác thêm xóa sửa

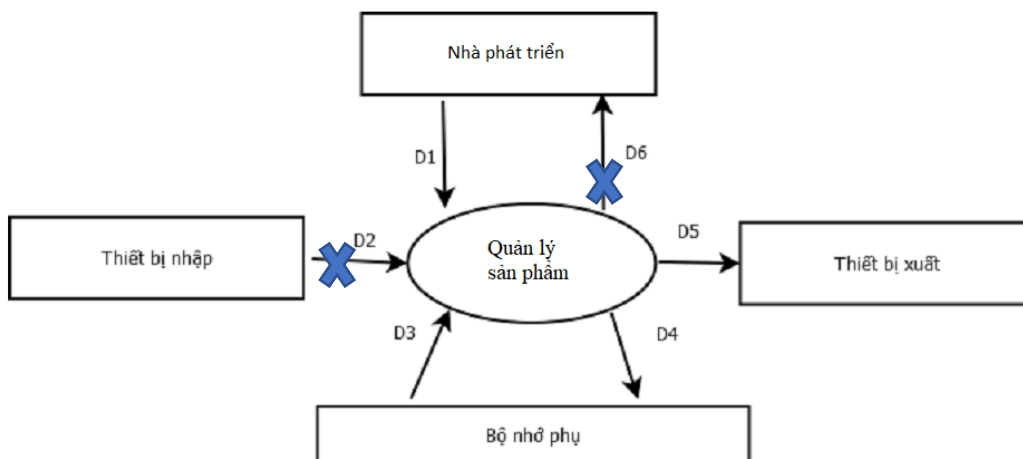
#### b) Màn hình chức năng



c) Danh sách xử lý

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nút Add Product	Chuyển sang màn hình Add Product	Không có	Không có
Nút Delete	Xóa 1 Product khỏi nhà thuốc	Trượt Product muốn xóa sang phía bên phải	Không có
Nút Update	Chuyển sang màn hình Update Product	Trượt Product muốn update sang phía bên trái	Không có

d) Sơ đồ luồng



**D1:** Nhập thông tin sản phẩm

**D2:** Không có

**D3:** Danh sách thông tin sản phẩm

**D4:**D1

**D5:** không có

**D6:**D3

**Thuật toán**

**B1:** Nhập thông tin từ D1

**B2:** Kết nối database

**B3:** Database nhận D1 từ người dùng

**B4:** Lưu D1 vào database

**B5:** Trả về D3 cho nhà phát triển

**B5:** Đóng database

**B6:** Kết thúc

### 3.2.6. Màn hình Add Product

#### a) Thông tin chức năng Add Product

- Cho phép người dùng thêm sản phẩm vào danh sách các sản phẩm của nhà thuốc

#### b) Màn hình xử lý



<



Name Product : \_\_\_\_\_

Price : \_\_\_\_\_ VND

Dead date : \_\_\_\_\_ NMD

Package : \_\_\_\_\_ Lô 1

Shape : \_\_\_\_\_

free for personal use

c) Danh sách xử lý

<b>Tên xử lý</b>	<b>Ý nghĩa</b>	<b>Điều kiện</b>	<b>Ghi chú</b>
Nút back	Quay lại màn hình Manager Product	Đã đăng nhập thành công	Không được bỏ trống
Nhập Name Product	Điền tên thuốc muốn thêm	Đã đăng nhập thành công	Không được bỏ trống
Nhập Price	Điền giá thuốc muốn thêm	Đã đăng nhập thành công	Không được bỏ trống
Nhập Dead Date	Điền HSD thuốc muốn thêm	Đã đăng nhập thành công	Không được bỏ trống
Nhập Package	Điền gói muốn thêm	Đã đăng nhập thành công	Không được bỏ trống
Nhập Shape	Điền lô thuốc muốn thêm	Đã đăng nhập thành công	Không được bỏ trống
Nút Add Product	Thêm 1 Product vào nhà thuốc	Đã đăng nhập thành công	Không được bỏ trống

3.2.7. Màn hình Update Product

a) Thông tin chức năng update product

- Cho phép người dùng có thể thay đổi các thông tin chung của sản phẩm

b) Màn hình xử lý

8:35

Add Product

<

Name  
Product : thuoc chong mat

Price : 15000 VND

Dead date :  NMD

Package :  Lô 1

Grid icon Home icon User icon

c) Danh sách xử lý

<b>Tên xử lý</b>	<b>Ý nghĩa</b>	<b>Điều kiện</b>	<b>Ghi chú</b>
Nút back	Quay lại màn hình Manager Product	Đã đăng nhập thành công	Không được bỏ trống
Nhập Name Product	Điền tên thuốc muốn sửa	Đã đăng nhập thành công	Không được bỏ trống
Nhập Price	Điền giá thuốc muốn sửa	Đã đăng nhập thành công	Không được bỏ trống
Nhập Dead Date	Điền HSD thuốc muốn sửa	Đã đăng nhập thành công	Không được bỏ trống
Nhập Package	Điền gói muốn sửa	Đã đăng nhập thành công	Không được bỏ trống
Nhập Shape	Điền lô thuốc muốn sửa	Đã đăng nhập thành công	Không được bỏ trống

### 3.2.8. *Màn hình Home*

#### a) Thông tin chức năng Home

- Màn hình mặc định của app, giúp người có thể tìm kiếm sản phẩm

#### b) Màn hình xử lý



Hello,

What can I do for  
you ?

Searching...

Prescription

Searching





c) Danh sách xử lý

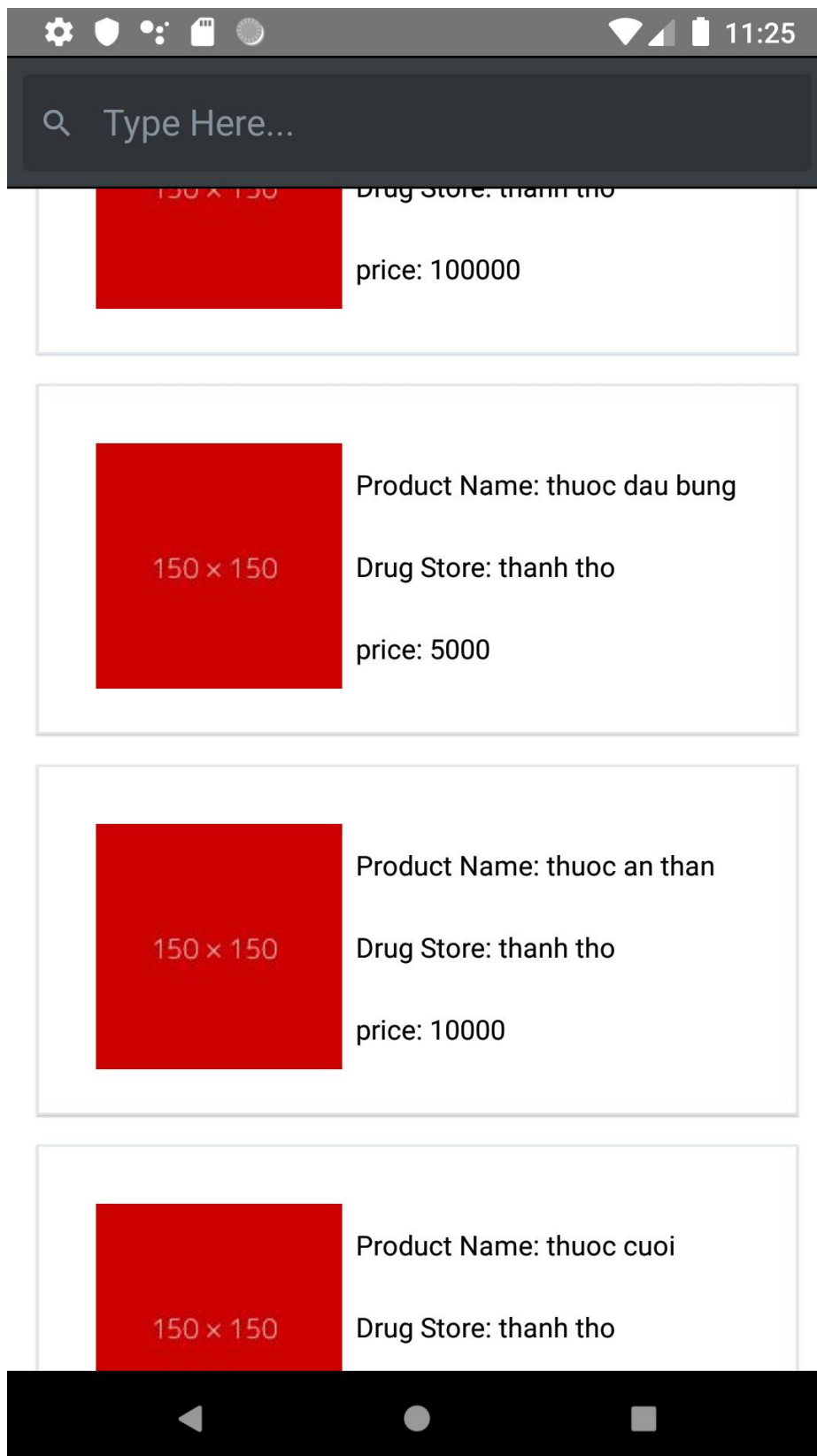
Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nhập searching	Điền tên thuốc muốn tìm kiếm	Không có	Không có
Nút searching	Tìm kiếm những loại thuốc trong nhập searching và chuyển sang màn hình searching product	Bắt buộc phải nhập thông tin trong ô nhập searching	Không có
Nút Prescription	Nhập danh sách các loại thuốc và thực hiện tìm kiếm	Không có	Hiện tại chưa hoàn thành do chưa hiểu nghiệp vụ

3.2.9. Màn hình search product

a) Thông tin chức năng search product

- Tìm kiếm các loại sản phẩm được các nhà thuốc đăng bán

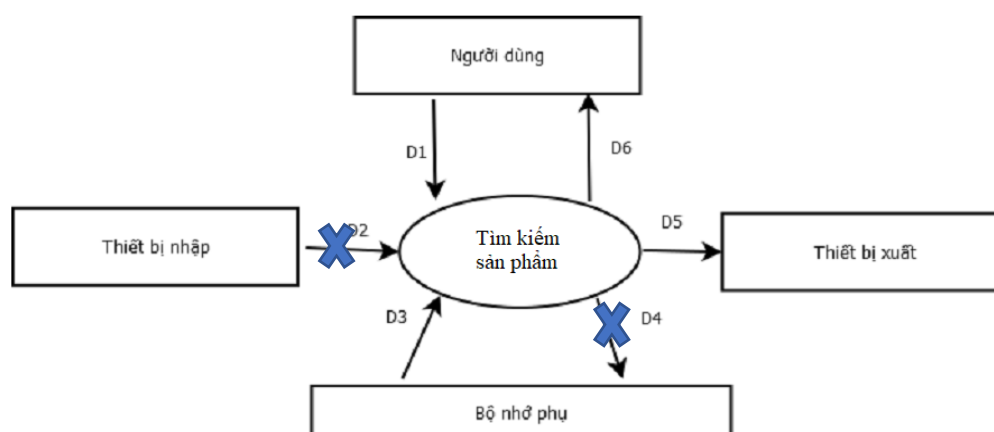
b) Màn hình xử lý



c) Danh sách chức năng

Tên xử lý	Ý nghĩa	Điều kiện	Ghi chú
Nhập searching	Điền tên thuốc muốn tìm kiếm	Không có	Không có
Item	Hiển thị thông tin chi tiết của từng loại sản phẩm như là hình ảnh, tên sản phẩm, giá tiền, tên nhà thuốc	Không có	Không có

d) Sơ đồ luồng



**D1: Nhập tên sản phẩm**

**D2: Không có**

**D3: Danh sách sản phẩm phù hợp**

**D4: không có**

**D5: không có**

**D6: D3**

**Thuật toán**

**B1: Nhập thông tin từ D1**

**B2: Kết nối database**

**B3: Database nhận D1 từ người dùng**

**B4: Kiểm tra sản phẩm phù hợp với D1**

**B5: Trả về D3 cho người dùng**

**B6: Đóng database**

**B7: Kết thúc**

## CHƯƠNG 4:

### Các kiến trúc thiết kế phần mềm

#### 4.1. Kiến trúc Component trong ReactJS

##### 4.1.1. Component là gì?

- Một component là một tập hợp module di động, có thể thay thế, có thể tái sử dụng các chức năng được đóng gói và có thể export thành các interface ở mức độ cao hơn.
- Một software component có thể được định nghĩa là một đơn vị bố cục với giao diện được quy định rõ ràng và chỉ phụ thuộc vào bối cảnh. Nó là một phần mềm có thể triển khai độc lập hoặc cũng có thể phụ thuộc vào bên thứ ba.
- Một component có 3 cách nhìn khác nhau là: xem hướng đối tượng, xem thông thường và xem quá trình liên quan
- Xem hướng đối tượng: Một thành phần được xem như là một tập hợp của một hoặc nhiều lớp hợp tác. Mỗi lớp miền vấn đề (phân tích) và lớp cơ sở hạ tầng (thiết kế) được giải thích để xác định tất cả các thuộc tính và hoạt động áp dụng cho việc thực hiện. Nó cũng bao gồm việc xác định các giao diện cho phép các lớp học để giao tiếp và hợp tác.
- Xem thông thường: Nó được xem như là một yếu tố chức năng hoặc một module của một chương trình tích hợp các xử lý logic, các cấu trúc dữ liệu nội bộ được yêu cầu để thực hiện các xử lý logic và một giao diện cho phép các thành phần được gọi và dữ liệu được truyền cho nó.
- Xem quá trình liên quan: Một thành phần giao diện người dùng (UI) bao gồm các lưới, các nút gọi là kiểm soát, và các thành phần tiện ích phơi bày một tập hợp

cụ thể của các chức năng được sử dụng trong các thành phần khác

#### **4.1.2.Đặc điểm của Component :**

- **Tính tái sử dụng:** Các thành phần thường được thiết kế để được tái sử dụng trong các tình huống khác nhau trong các ứng dụng khác nhau. Tuy nhiên, một số thành phần có thể được thiết kế cho một công việc cụ thể.
- **Thay thế:** Các thành phần có thể được tự do thay thế bằng các thành phần tương tự khác.
- **Không có bối cảnh cụ thể:** Các thành phần được thiết kế để hoạt động trong môi trường và bối cảnh khác nhau.
- **Khả năng mở rộng:** Một thành phần có thể được mở rộng từ các thành phần hiện có để cung cấp cho hành vi mới.
- **Tính đóng gói:** Một thành phần A mô tả các giao diện, cho phép người gọi để sử dụng chức năng của nó, và không để lộ các chi tiết của các quy trình nội bộ hoặc bất kỳ biến nội bộ.
- **Các thành phần được thiết kế để có phụ thuộc tối thiểu vào các thành phần khác.**

#### **4.1.3.Kiến trúc Component:**

- a) Component-based software engineering (CBSE) - Kiến trúc dựa trên Component tập trung vào việc phân chia thiết kế thành các thành phần chức năng hoặc các logic độc lập đại diện cho giao diện truyền thông có chứa các method, event và thuộc tính. Nó cũng cung cấp một mức độ trừu tượng cao hơn và chia các vấn đề thành các vấn đề nhỏ hơn kết hợp với các phân vùng thành phần.
- b) Mục tiêu chính của CBSE là đảm bảo sự tái sử dụng của các thành phần. Mỗi thành phần đóng gói một chức năng và hành vi của 1 yếu tố phần mềm thành một đơn vị nhị phân giúp sử dụng lại và tự triển khai

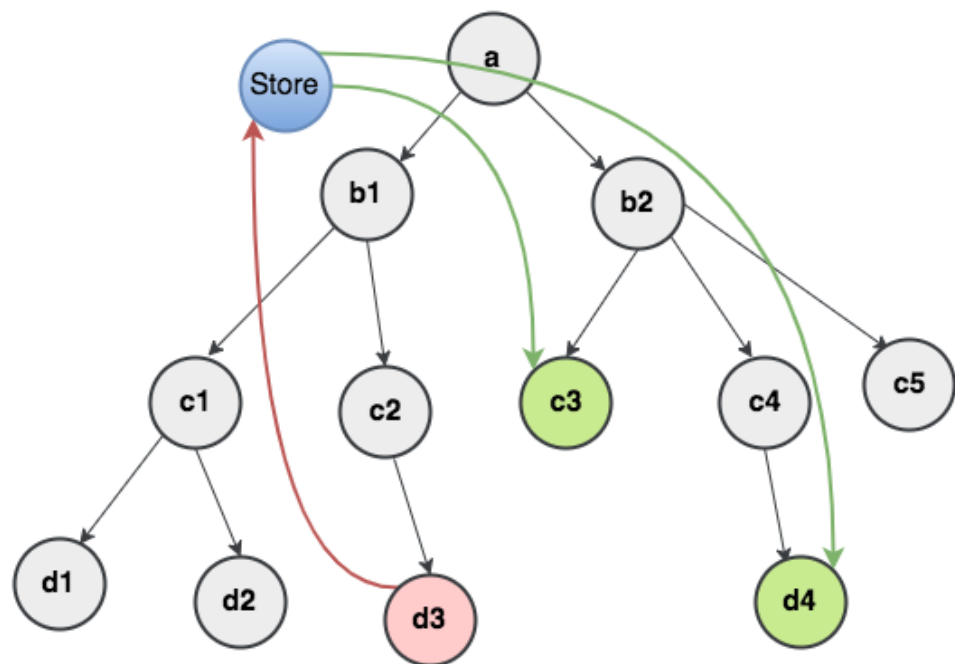
c) CBSE có nhiều ưu điểm so với các phương pháp tiếp cận đối tượng kiểu truyền thống như:

- Giảm thời gian phát triển do sử dụng lại được các thành phần có sẵn.
- Tăng độ tin cậy với việc sử dụng lại các thành phần đã có.

#### 4.2. Kiến trúc Tổng thể Redux trong ReactJS

✚ Khái niệm về redux: Redux là một thư viện nhỏ với API đơn giản, giới hạn được thiết kế để trở thành vùng chứa dự đoán cho trạng thái ứng dụng. Nó hoạt động theo kiểu tương tự như hàm khử, một khái niệm lập trình hàm.

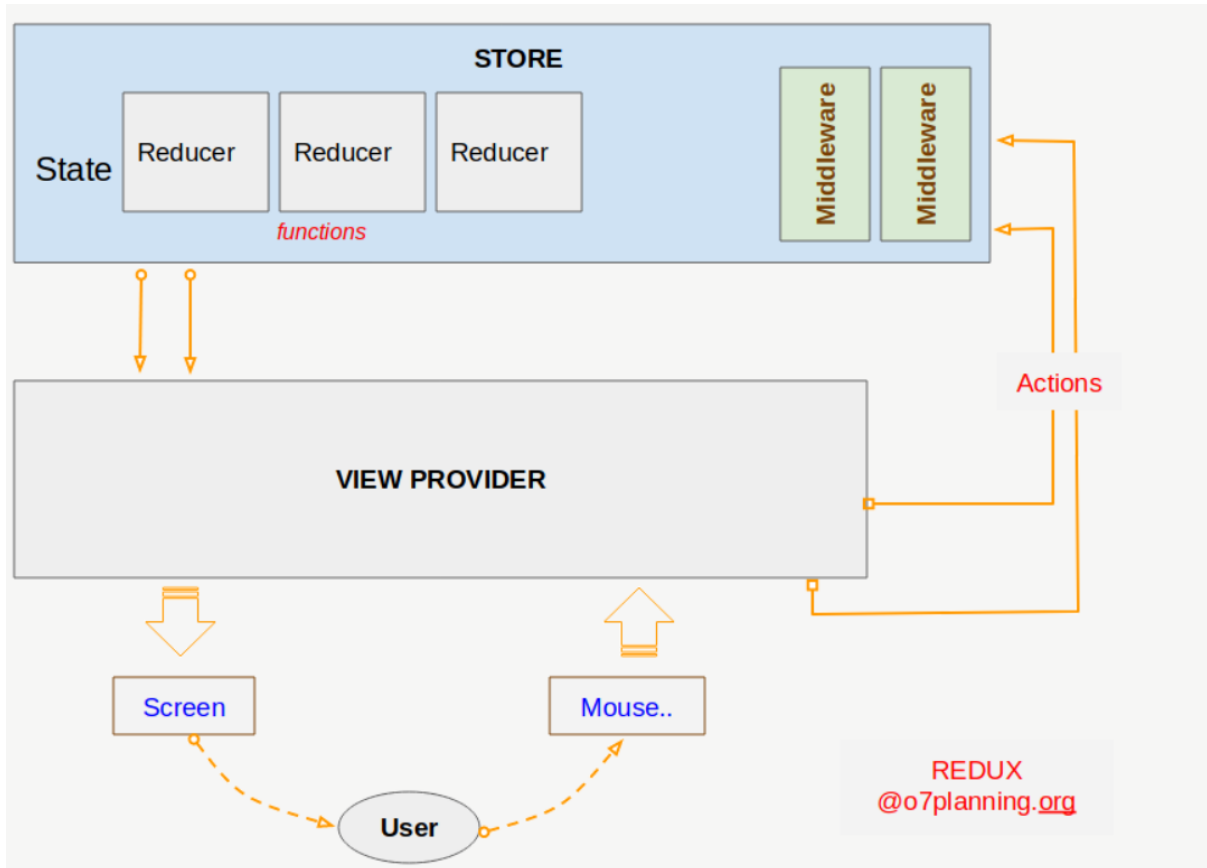
- Ta có thể hiểu 1 cách đơn giản: Redux sẽ chứa State của toàn bộ ứng dụng và được lưu trong 1 store duy nhất là 1



#### object trong mô hình tree

- Chỉ có 1 cách thay đổi state đó là tạo ra 1 action (object mô tả những gì xảy ra)
- Để chỉ rõ state tree được thay đổi bởi 1 action bạn phải viết case reducers

- ✚ Kiến trúc của Redux thực chất là sự lược bỏ những thứ phức tạp của kiến trúc Flux trong ReactJS
- ✚ Sơ đồ kiến trúc của Redux



- ✚ Chi tiết các khái niệm trong kiến trúc của Redux
  - VIEW PROVIDER: đại diện cho 1 View Framework để đăng ký với STORE

```
const AppContainer = createAppContainer(AppNavigation);
```

```
const initialState = {};
```

```
const store = createStore(MyReducer, initialState);
```

You, 25 days ago | 2 authors (You and others)

```
export default class App extends Component {
```

```
  render() {
```

```
    return (You, 25 days ago • su dung redux va goi api
```

```
      <Provider store={store}>
```

```
        <AppContainer />
```

```
      </Provider>
```

```
    );
```

```
  }
```

```
}
```

- **ACTION:** là 1 đối tượng được tạo ra để lưu trữ thông tin liên quan tới sự kiện của người dùng, nó bao gồm cả thông tin như hành động. Chức năng của action dùng để truyền tải mục đích của người dùng truyền tới store

```
import * as CONST from './constant'; You, 14 days ago • fix logic
```

```
export function loaddata(data) {
```

```
  return {
```

```
    type: CONST.LOADDATA,
```

```
    data: data,
```

```
  };
```

```
}
```

```
export function searchdata(data) {
```

```
  return {
```

```
    type: CONST.SEARCHDATA,
```

```
    data: data,
```

```
  };
```

```
}
```

- **STORE:** là nơi quản lý trạng thái của ứng dụng và có thể truy cập để lấy trạng thái (state). Store lắng nghe action thông qua hàm dispatch() và cập nhập qua views



```
import {combineReducers} from 'redux';
import Login from '../component/login/reducer';
import Home from '../screen/home/reducer';
import ManagerDrug from '../screen/ManagerDrug/reducer';

const MyReducer = combineReducers({
  Home,
  Login,
  ManagerDrug,
});

export default MyReducer;
```

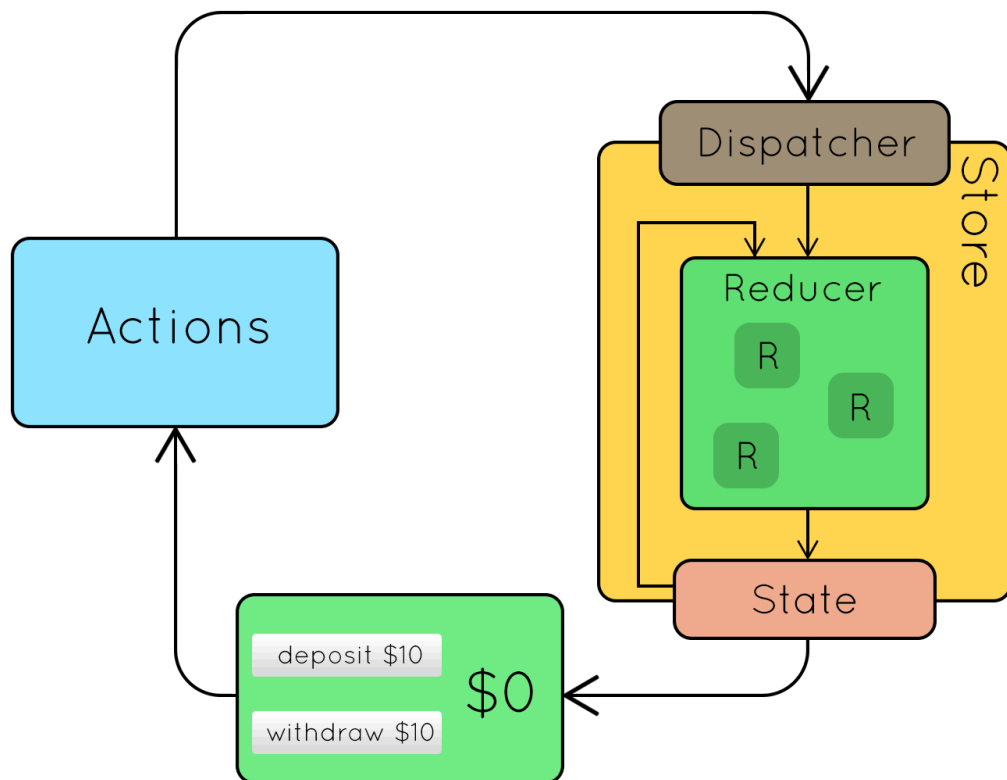
- REDUCER: Khác với actions có chức năng là mô tả những thứ gì đã xảy ra, nó không chỉ rõ state nào của response thay đổi, mà việc này là do REDUCER đảm nhiệm, nó là nơi xác định state sẽ thay đổi như thế nào, sau đó trả ra một

```
let initialState = {
  isCheck: false,
  data: [],
  isAccount: false,
  loadAPI: false,
};

export default function handleAction(state = initialState, action) {
  switch (action.type) {
    case actionType.LOADDATA: {
      return {
        loadAPI: true,
        ...state,
      };
    }
    default:
      return state;
  }
}
```

state mới

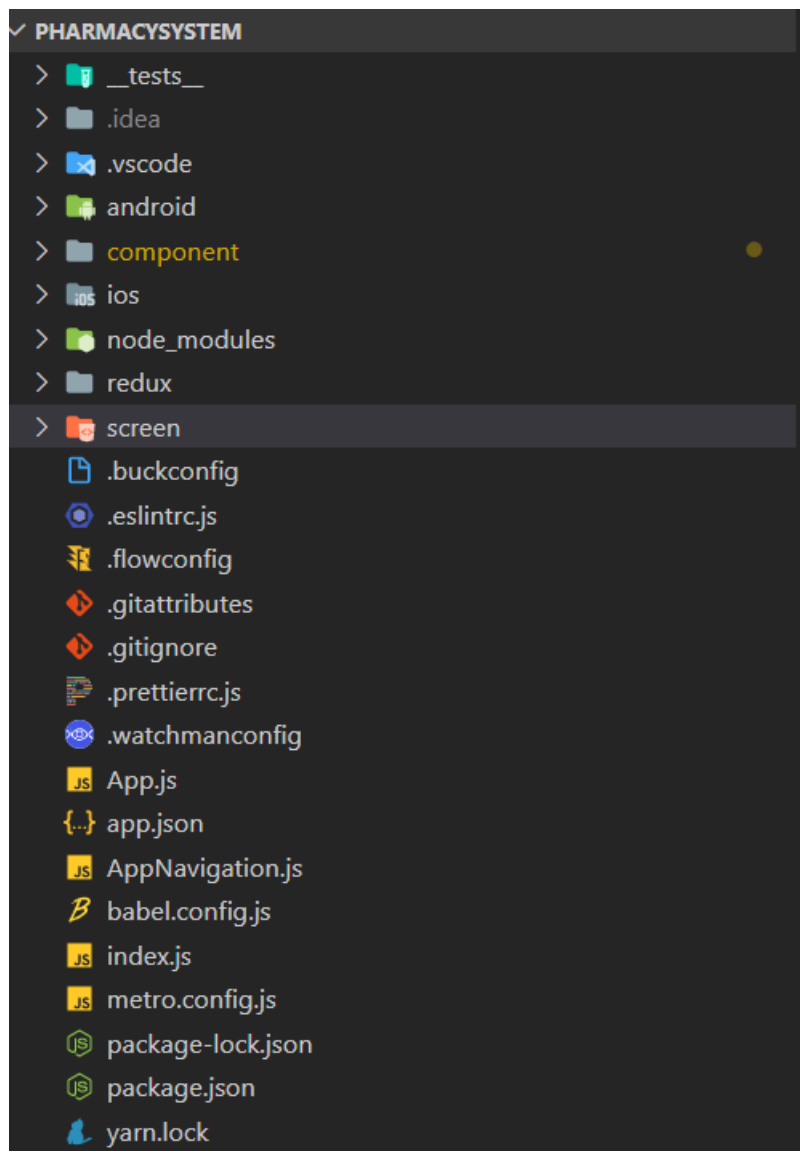
- Nguyên lý hoạt động của redux là action->reducer->store->view



### 4.3. Phân tích đồ án

#### 4.3.1. Phân tích thiết kế cây thư mục:

- Ứng dụng được khởi tạo theo mô hình chuẩn của react-native: khởi tạo ứng dụng bằng câu lệnh `npx react-native init PHARMACYSYSTEM`
- Cấu trúc cây thư mục cho project



- Cấu hình cho project:
  - File index.js sẽ là file chứa node root (App.js).
  - App.js mang vai trò là node root và thực hiện gọi các node con. Ngoài ra App.js còn đóng vai trò như một store để chứa tất cả các state của các node con. Cấu hình cho file App.js

```

You, a month ago | 2 authors (You and others)
NumbersZeros, 2 months ago

import React, {Component} from 'react';
import {createAppContainer} from 'react-navigation';
import AppNavigation from './AppNavigation';
import {Provider} from 'react-redux';
import {createStore} from 'redux';
import MyReducer from './redux/MyReducer';

const AppContainer = createAppContainer(AppNavigation);

const initialState = {};
const store = createStore(MyReducer, initialState);

You, a month ago | 2 authors (You and others)
export default class App extends Component {
  render() {
    return (
      <Provider store={store}>
        <AppContainer />
      </Provider>
    );
  }
}

```

- AppNavigation: đóng vai trò như là router dùng để thực hiện gọi các node con. Cấu hình cho file AppNavigation.

```

import Home from './component/home/container';
import Categories from './component/home/categories';

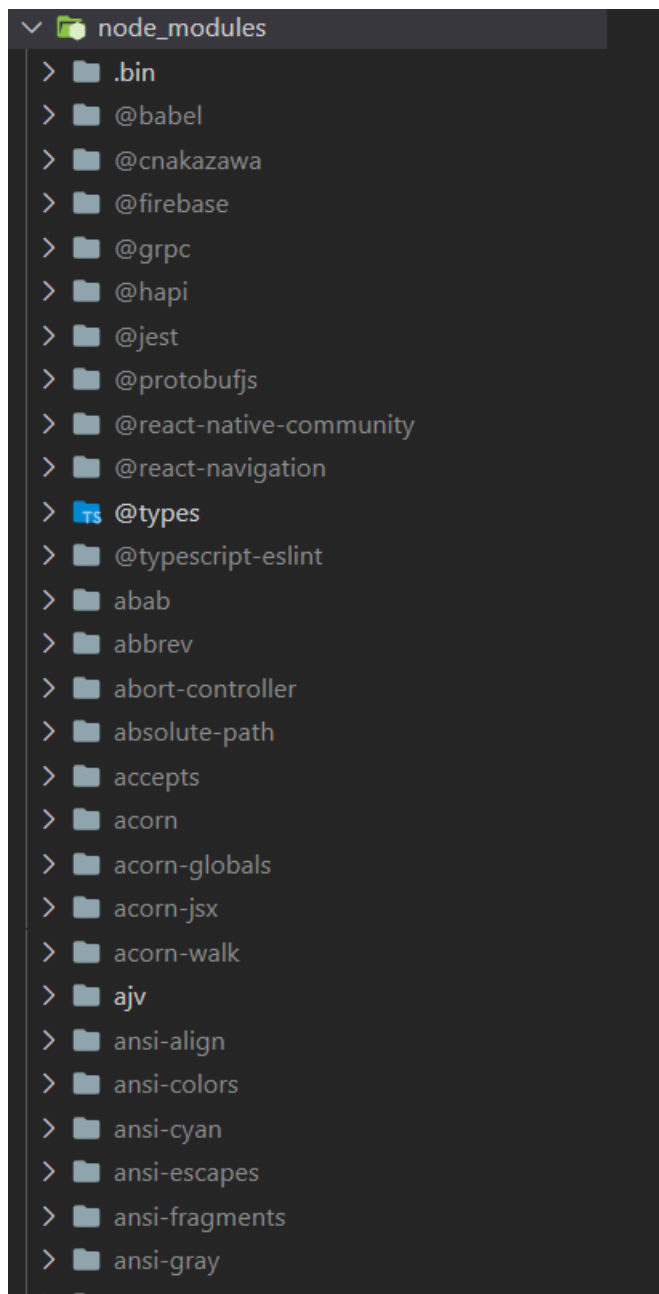
import User from './component/login/container';
import Login from './component/login/Login.user';
import Register from './screen/Register';
import ManagerUser from './component/login/Manager.user';
import ManagerDrug from './component/login/Manager.drug';

const RootStack = createBottomTabNavigator({
  Manager: {screen: Manager},
  Home: {screen: Home},
  User: {screen: User},
}, {
  defaultNavigationOptions: ({navigation}) => ({
    tabBarIcon: ({focused, tintColors}) => {
      const {routeName} = navigation.state;
      let iconName;
      if (routeName === 'Manager') {
        iconName = 'md-apps';
      } else if (routeName === 'Home') {
        iconName = 'md-home';
      } else if (routeName === 'User') {
        iconName = 'md-contact';
      }
      return <Icons name={iconName} size={40} color={tintColors} />;
    },
  }),
  tabBarOptions: {
    activeTintColor: 'light',
    activeBackgroundColor: '#61892F',
    inactiveBackgroundColor: '#474B4F',
    labelStyle: {
      fontSize: 0,
    },
  },
});

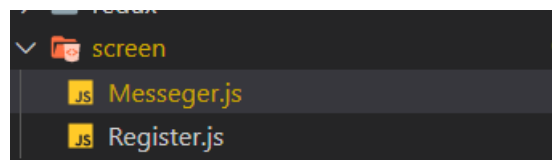
export default createStackNavigator({
  Home: {screen: RootStack},
  Details: {screen: Categories},
  AddDrug: {screen: AddDrug},
  Login: {screen: Login},
  Register: {screen: Register},
  ManagerUser: {screen: ManagerUser},
  ManagerDrug: {screen: ManagerDrug},
}, {
  headerMode: 'none',
}, {
  initialRouteName: 'Home',
});

```

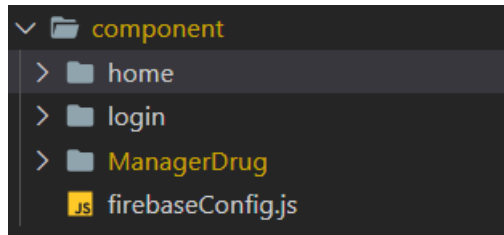
- thư mục node\_modules đóng vai trò như 1 trình quản lý các thư viện được import vào project



- Thư mục screen sẽ là nơi chứa các màn hình phụ trong quá trình thực hiện.



- Thư mục quan trọng nhất là Component: Nó đóng vai trò quan trọng nhất trong việc kết hợp các module lại với nhau. Component sẽ quản lý tất cả các node con có trong chương trình và từng node con như: Home, Login, ManagerDrug sẽ đóng vai trò như những smart component. Điều này rất thuận lợi trong việc truy xuất state của từng node con. Ngoài ra Component còn chứa file firebaseConfig.js, Nó giúp ta có thể tái sử dụng việc kết nối với FirebaseCloud mà không cần phải viết



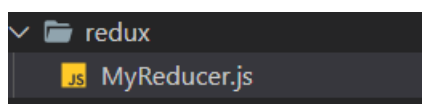
đi viết lại nhiều lần

#### ○ Cấu hình cho firebaseConfig.js

```
import * as firebase from 'firebase';
import 'firebase/firestore';

var firebaseConfig = {
  apiKey: 'AIzaSyDCGACH5sPsoJIH8NNQin6RhVw3dyDzxA',
  authDomain: 'testfirebase-dbd0c.firebaseio.com',
  databaseURL: 'https://testfirebase-dbd0c.firebaseio.com',
  projectId: 'testfirebase-dbd0c',
  storageBucket: 'testfirebase-dbd0c.appspot.com',
  messagingSenderId: '1083535192825',
  appId: '1:1083535192825:web:1e03f5146a93ce9cd76828',
  measurementId: 'G-HTL9SXL7J',
};
// Initialize Firebase
var firebaseApp = firebase.initializeApp(firebaseConfig);
// export var db = firebase.firestore();
export default firebaseApp;
```

- Thư mục redux: đóng vai trò như 1 kho chứa. Nó thực hiện combine hết tất cả state trong từng smart component và truyền đến tất cả các component khác trong chương trình. Nó là bài toán giải quyết conflict state trong reactjs.



- Cấu hình cho file MyReducer.js

```
import {combineReducers} from 'redux';
    You, a month ago • get data từ server về
import Login from '../component/login/reducer';
import Home from '../component/home/reducer';
import ManagerDrug from '../component/ManagerDrug/reducer';

const MyReducer = combineReducers({
  Home,
  Login,
  ManagerDrug,
});

export default MyReducer;
```



## CHƯƠNG 5:

### Chỉ tiêu thực hiện và kết quả đạt được

#### 5. Mục tiêu đề ra

- Hoàn thành thành ứng dụng với những chức năng cơ bản và tập trung vào phía người dùng là nhà thuốc
- Thực hiện đồ án dựa trên kiến trúc phổ biến nhất trong react-native là:
  - Nắm được cách vận hành truyền từ store sang các smart component và truyền xuống các component con
  - Hiểu rõ những nội dung cơ bản của Framework ReactJS
  - Tìm hiểu những thuận lợi cũng như những thách thức của lập trình trên android
  - Tìm hiểu về kiểu dữ liệu mới là NoSQL. Đặc biệt là firebase cloud

#### 6. Thành quả đạt được

- Tất cả thành viên đã học tập các kiến thức mới và các kiến thức mở rộng trong javascript, reactjs và react-native. Trong đó có một số framework tìm hiểu trong quá trình làm việc như lodash, axios.
- Hiểu được cách thức vận hành của redux và áp dụng vào trong chương trình. Ngoài ra còn tìm hiểu thêm các thư viện tùy biến cho redux như là react-redux.
- Nắm được các kiến thức cơ bản của javascript
- Nắm được các cách thức quản lý các màn hình thông qua thư viện react-navigation
- Hiểu được các khái niệm về NoSQL thông qua giáo trình Cơ sở dữ liệu nâng cao
- Nắm được cách thức kết nối từ app react-native với database Firebase. Ngoài ra việc có thể thao tác thêm xóa sửa trên database mà còn có thể kết hợp đăng nhập vào app bằng tài khoản gmail đã được đăng kí trên firebase

#### 7. Những khó khăn trong quá trình làm việc

- Việc chuyển đổi từ một ngôn ngữ hướng đối tượng như C# về ngôn ngữ javascript đối với chúng em gặp nhiều khó khăn. Chúng em phải tốn nhiều thời gian trong việc học lại các nội dung thiết yếu mới có thể sử dụng được reactjs

- Việc tích hợp module mới vào app react-native trên JDK cũng tương đối khó khăn và có nhiều lỗi phát sinh phải thực hiện debug nhiều lần
- Tìm hiểu mô hình kiến trúc redux cũng tương đối lớn nên vận hành không được thuận lợi và phải chỉnh sửa lại nhiều lần mới có thể vận hành tốt
- Áp dụng mô hình redux để gọi api diễn ra không đồng bộ do chưa có nhiều kiến thức về backend và không xử lý được độ trễ của firebase. Tuy nhiên vấn đề đã được khắc phục nhưng hiệu suất của app thì trở nên kém hơn và chậm chạp
- Khi khảo sát thực tế thì người dùng không chịu hợp tác. Nên về vấn đề nghiệp vụ vẫn chưa nắm rõ

#### 8. **Định hướng mở rộng**

- Hiện tại App mới chỉ dừng lại ở mức cơ bản và chỉ chủ yếu hướng tới nhà thuốc vẫn chưa có nhiều chức năng cho người dùng. Trong tương lai sẽ phát triển thêm nhiều tính năng cho người dùng tương tự như các app lazada, tiki
- Trong tương lai sẽ tìm hiểu kỹ hơn về các vấn đề nghiệp vụ để cải thiện tính logic cho các chức năng
- Tự phát triển 1 web server chứa toàn bộ data của app và tiến hành cải thiện hiệu suất cho app