**ĐẠI HỌC QUỐC GIA TP. HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG**

**ĐỀ TÀI : Ứng dụng xem phim với React Native**

**Lớp :** SE346.J11

**GVHD :** ThS. Huỳnh Tuấn Anh

**Nhóm**

**:** 13521007 – Phan Văn Tùng

14520683 – Nguyễn Hoài Phú

Hồ Chí Minh. Ngày 24 tháng 12 năm 2018

**LỜI NÓI ĐẦU**

Cuộc sống bận rộn hàng ngày, bạn đã dành bao nhiêu thời gian cho bản thân? Sau những lo toan của cuộc sống, lắm lúc bạn chẳng muốn ra đường nữa, chỉ muốn tìm một thứ gì đó để giải trí tại nhà. Và tôi tin rằng phim là 1 trong những sự lựa chọn ưu tiên cho đa số các bạn,và chính tôi cũng vậy. Tôi đã tứng phát điên với những link, app mà khi xem phim thì tốc độ load cực kỳ chậm, hình ảnh không sắc nét, mang nhiều virus và nhiều vấn đề mà đa số app nào cũng gặp phải. Với mong muốn tạo ra một ứng dụng xem phim có thể khắc phục các thiếu sót và tổng hợp những ưu điểm của các app trước để phục vụ nhu cầu xem phim ngày càng mở rộng của các bạn, và cũng là một điều thật tuyệt vời cho một dân IT như tôi. Sau hàng tá thời gian để dạo 1 lượt thử nghiệm từng app xem phim, tôi đã đúc kết được rất nhiều bài học, kinh nghiệm cho bản thân, và do đó, tôi đã tạo ra một app mà tôi tin nó sẽ mang đến cho bạn điều bạn muốn. Bạn có tin vào điều đó không, nếu có thì tiếp theo hãy lắng nghe tôi xin giới thiệu cho các bạn rõ hơn về ứng dụng.... ,cũng chính là đứa con tinh thần mà chúng tôi đã tạo ra.

**LỜI CẢM ƠN**

Trong quá trình thực hiện đề tài với những khó khăn nhất định gặp phải, chúng em đã nhận được sự chỉ đạo tận tình của thầy giáo hướng dẫn giúp đỡ vượt qua khó khăn cũng như nhà trường đã tạo điều kiện để chúng em hoàn thành đề tài này.

Chúng em xin gửi lời cảm ơn chân thành tới thầy **Huỳnh Tuấn Anh**, người đã tận tình giảng dạy chúng em trong thời gian vừa qua. Chúng tôi xin gửi lời cảm ơn chân thành tới toàn thể các bạn sinh viên lớp Lập trình trên thiết bị di động đã động viên giúp đỡ tạo điều kiện thuận lợi cho chúng tôi trong suốt quá trình thực hiện đề tài này.

Với sự cố gắng và nỗ lực của mình, chúng em đã cùng nhau xây dựng và hoàn thành đề tài này. Tuy nhiên do trình độ còn có phần hạn chế nên trong quá trình thực hiện đồ án không tránh khỏi những thiếu sót về kiến thức, nên kính mong nhận được sự góp ý của thầy và các bạn.

Chúng em xin trân thành cảm ơn !

*Hồ Chí Minh, ngày 24 tháng 12 năm 2018*

# **NHẬN XÉT CỦA GIẢNG VIÊN**

**MỤC LỤC**

**[LỜI NÓI ĐẦU](#_Toc533557933)** [2](#_Toc533557933)

[**LỜI CẢM ƠN** 3](#_Toc533557934)

[**NHẬN XÉT CỦA GIẢNG VIÊN** 4](#_Toc533557935)

[**I.** **THÔNG TIN CHUNG** 8](#_Toc533557936)

[**II.** **TÌM HIỂU VỀ REACT NATIVE** 8](#_Toc533557937)

[**1.** **WHAT is React Native?** 8](#_Toc533557938)

[**2.** **HOW does React Native work?** 8](#_Toc533557939)

[**3.** **WHY is React Native?** 10](#_Toc533557940)

[**4.** **Cài đặt môi trường (Android)** 11](#_Toc533557941)

[**5.** **Props và State** 12](#_Toc533557942)

[*a.* *Props (viết tắt của Properties)* 12](#_Toc533557943)

[*b.* *State* 13](#_Toc533557944)

[**6.** **Một số component thông dụng** 14](#_Toc533557945)

[*a.* *Hiển thị thông báo đang tải nội dung* 14](#_Toc533557946)

[*b.* *Button (Phím bấm)* 15](#_Toc533557947)

[*c.* *DatePickerIOS ( Chế độ chọn ngày trên IOS)* 16](#_Toc533557948)

[*d.* *Image (Hình ảnh)* 17](#_Toc533557949)

[*e.* *ResizeMode(Xác định thay đổi kích thước khung ảnh, phù hợp với ảnh thô)* 20](#_Toc533557950)

[*f.* *ListView (Danh sách)* 22](#_Toc533557951)

[*g.* *Slider(Dạng chuyển động hình ảnh)* 35](#_Toc533557952)

[*h.* *SelectionList* 36](#_Toc533557953)

[*i.* *ToolbarAndroid* 41](#_Toc533557954)

[*j.* *View* 43](#_Toc533557955)

[*k.* *Text* 46](#_Toc533557956)

[**III.** **PHÁT BIỂU BÀI TOÁN** 48](#_Toc533557957)

[**1.** **Khảo sát hiện trạng** 48](#_Toc533557958)

[**2.** **Tiện ích ứng dụng** 48](#_Toc533557959)

[**IV.** **SƠ ĐỒ ĐẶC TẢ** 49](#_Toc533557960)

[**1.** **Use-case diagram** 49](#_Toc533557961)

[**1.1.** **Đăng nhập** 49](#_Toc533557962)

[**1.2.** **Đăng xuất** 49](#_Toc533557963)

[**1.3.** **Tìm kiếm phim** 49](#_Toc533557964)

[**1.4.** **Xem phim** 49](#_Toc533557965)

[**1.5.** **Xem lịch sử phim đã xem** 50](#_Toc533557966)

[**1.6.** **Lưu phim yêu thích** 50](#_Toc533557967)

[**2.** **Đặc tả Use-case** 50](#_Toc533557968)

[**2.1.** **Đăng nhập** 50](#_Toc533557969)

[**2.2.** **Đăng xuất** 52](#_Toc533557970)

[**2.3.** **Tìm kiếm phim** 53](#_Toc533557971)

[**2.4.** **Xem phim** 54](#_Toc533557972)

[**2.5.** **Xem lịch sử phim đã xem** 54](#_Toc533557973)

[**2.6.** **Lưu phim yêu thích** 55](#_Toc533557974)

[**3.** **Sequence diagram** 57](#_Toc533557975)

[**3.1.** **Đăng nhập** 57](#_Toc533557976)

[**3.2.** **Đăng xuất** 58](#_Toc533557977)

[**3.3.** **Tìm kiếm phim** 58](#_Toc533557978)

[**3.4.** **Xem phim** 59](#_Toc533557979)

[**3.5.** **Xem lịch sử phim đã xem** 60](#_Toc533557980)

[**3.6.** **Lưu phim yêu thích** 60](#_Toc533557981)

[**V.** **THIẾT KẾ DỮ LIỆU** 62](#_Toc533557982)

[**1.** **Mô hình ERD** 62](#_Toc533557983)

[**2.** **Class Diagram** 63](#_Toc533557984)

[**VI.** **GET LINK VIDEO PHIM** 63](#_Toc533557985)

[**VII.** **CÁC COMPONENT ĐƯỢC SỬ DỤNG** 64](#_Toc533557986)

[**1.** **Banner** 64](#_Toc533557987)

[**2.** **Film & BigFilm** 64](#_Toc533557988)

[**3.** **ListFilms** 65](#_Toc533557989)

[**4.** **SlideFilms** 65](#_Toc533557990)

[**5.** **Episode** 66](#_Toc533557991)

[**6.** **ListEpisodes** 66](#_Toc533557992)

[**7.** **Actor & Director** 66](#_Toc533557993)

[**8.** **SlideActors** 66](#_Toc533557994)

[**9.** **SlideDirectors** 67](#_Toc533557995)

[**10.** **SlideShow** 67](#_Toc533557996)

[**11.** **Player (dùng thư viện react-native-video-controls)** 68](#_Toc533557997)

[**VIII.** **NAVIGATION** 68](#_Toc533557998)

[**IX.** **GIAO DIỆN** 69](#_Toc533557999)

[**1.** **Trang chủ (Home)** 69](#_Toc533558000)

[**2.** **Danh mục (Thể loại)** 69](#_Toc533558001)

[**3.** **Tìm kiếm** 70](#_Toc533558002)

[**4.** **Cá nhân** 70](#_Toc533558003)

[**5.** **Danh sách phim** 71](#_Toc533558004)

[**6.** **Xem phim** 71](#_Toc533558005)

[**7.** **Đăng nhập** 72](#_Toc533558006)

[**X.** **KẾT LUẬN VÀ TỰ ĐÁNH GIÁ** 72](#_Toc533558007)

[**1.** **Kết luận** 72](#_Toc533558008)

[**2.** **Tự đánh giá** 72](#_Toc533558009)

[**XI.** **TÀI LIỆU THAM KHẢO** 72](#_Toc533558010)

1. **THÔNG TIN CHUNG**

* Tên đề tài : **Ứng dụng xem phim với React Native**
* Môi trường phát triển ứng dụng : Android OS
* Thông tin Nhóm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** | **Điện thoại** | **Email** |
| 1 | 14520683 | Nguyễn Hoài Phú | 0389 230 860 | [14520683@gm.uit.edu.vn](mailto:14520683@gm.uit.edu.vn) |
| 2 | 13521007 | Phan Văn Tùng |  | 13521007@gm.uit.edu.vn |

1. **TÌM HIỂU VỀ REACT NATIVE**
2. **WHAT is React Native?**

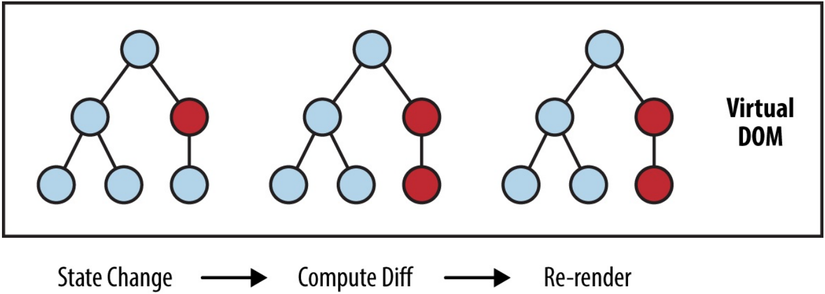
React Native là một framework cho phép các lập trình viên xây dựng các ứng dụng native mà chỉ sử dụng ngôn ngữ lập trình javascript.

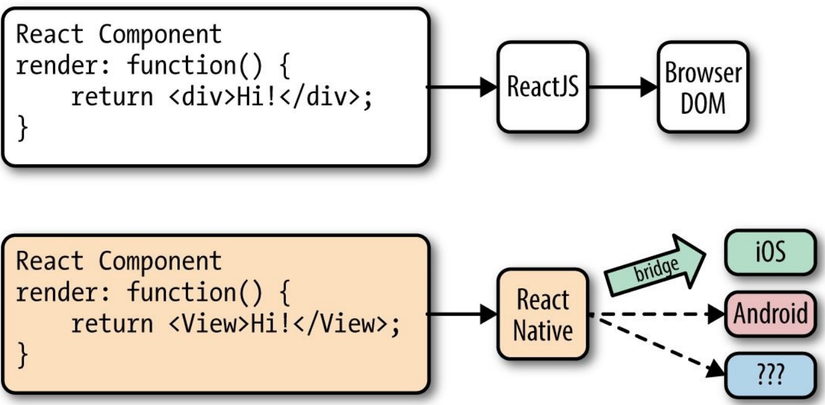
React Native cho phép bạn xây dựng các ứng dụng trên Android và iOS chỉ với một ngôn ngữ thống nhất là javascript nhưng mang lại trải nghiệm native app thực sự. Không như các framework hybrid khác (viết một lần, triển khai nhiều nơi), React Native tập trung vào việc một lập trình viên làm việc hiệu quả trên môi trường đa nền tảng như thế nào.Điều này không những có lợi cho lập trình viên web mà nó giúp cho các doanh nghiệp phát triển sản phẩm đầu cuối với ít nhân lực hơn.

1. **HOW does React Native work?**

Nhắc lại: một trong những tính năng của React đó là Virtual DOM (Document Object Model)

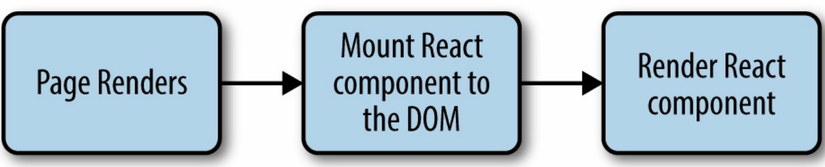
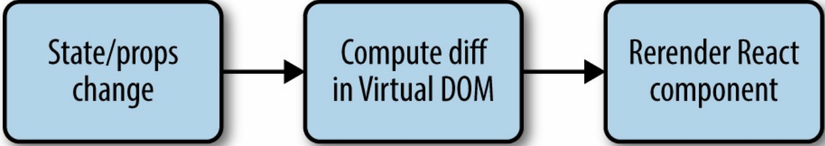
Trong React, Virtual DOM hoạt động như một layer giữa mô tả về cách hiển thị mọi thứ và công việc được thực hiện để hiển thị trên page. Để hiển thị UI trên browser, developer phải sửa DOM(Document Object Model) của browser. Đây là cách tốn hiệu năng vì viết lại DOM nhiều lần sẽ tác động đáng kể đến performance. Thay vì thay đổi trực tiếp trên page, React tính toán những thay đổi cần thiết bằng cách sử dụng một phiên bản bộ nhớ của DOM hay nói cách khác là một bản sao của DOM và hiển thị lại những thay đổi cần thiết.



Và sau đây, là cách React Native hoạt động. Thay vì render DOM của browser, Reat Native gọi Objective-C API để hiển thị lên iOS component, hoặc Java API để hiển thị Android component.

**Bridge** là cầu nối giữa các React Component với native UI element tương ứng ví dụ như View có thể là iOS UIView. React Native hiện tại support cho Android và iOS. Bởi vì abtraction layer được cung cấp bởi Vitural DOM. React Native cũng có thể nhắm đến nhiều platform khác. "Target platform là gì không quan trọng, quan trọng là cần bridge"

* **Vòng đời:**

Khi React chạy trong browser, vòng đời bắt đầu khi mounting các React Component. Sau đó React xử lý việc rendering và rerendering các component nào cần thiết. Đối với trạng thái render, dev trả về HTML markup từ method render của React component mà React sau đó hiển thị trực tiếp vào page khi cần thiết.

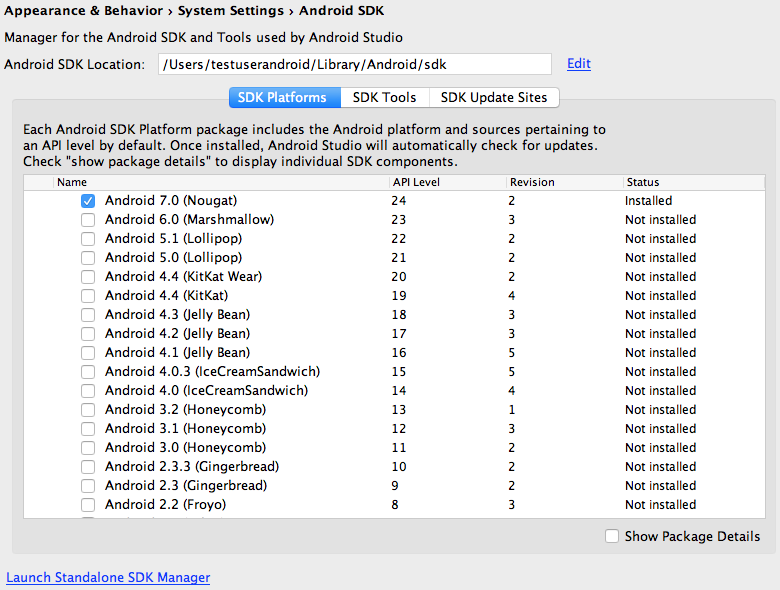
Với React Native thì vòng đời cũng tương tự, nhưng quá trình render hơi khác một chút vì React Native phụ thuộc vào bridge. Bidge biên dịch các mã JS và gọi platform của host, các API và các UI elements. Vì React Native không chạy trên main thread, nó có thể chạy bất đồng bộ nên nó không ảnh hưởng đến trải nghiệm của người dùng.

1. **WHY is React Native?**

Lý do đầu tiên và quan trọng nhất phải sử dụng React Native Framework là nhà phát triển phải viết một bộ mã JavaScript, để tận dụng được hiệu suất của các REACT Native Mobile Applications.

* **Ưu điểm:**
* Sử dụng nền tảng chéo: code 1 được 2. Chỉ 1 codebase đơn cho cả 2 nền tảng cùng lúc (Android & iOS)
* Class performance: Các REACT Native apps được biên dịch thành mã nguyên bản, cho phép nó không chỉ hoạt động trên cả hai hệ điều hành mà còn hoạt động theo cùng một cách trên cả hai nền tảng mà không có độ trễ nào cả.
* Khả năng phát triển nhanh
* Có khả năng tái sử dụng các thành phần và dễ bảo trì
* **Nhược điểm:**
* Một số mô-đun tùy chỉnh trong framework bị thiếu, điều này có thể khiến chúng ta mất thêm thời gian trong việc xây dựng và tạo mô-đun của riêng mình.
* Sự tồn tại của nó là không chắc chắn vì Facebook có tất cả các quyền để hủy bỏ dự án bất cứ lúc nào.

1. **Cài đặt môi trường (Android)**

* Cài đặt Android Studio.
* Xác nhận đã cài đặt Android SDK: click vào **Configure** ở cuối màn hình trong Android Studio Setup Wizard hoặc mở **Preferences** từ menu của Android Studio, sau đó lựa chọn **Appearance and Behavior** **→ System Settings** → \*\***Android SDK** →chọn **SDK Platforms** trong **SDK Manager** và bạn sẽ nhìn thấy dấu tích xanh ở Android 7.0 (Nougat). Nếu như không nhìn thấy bạn hãy click vào checkbox sau đó click vào **Apply**.
* Cài đặt đường dẫn: Mở một Terminal mới, hãy tạo (hoặc chỉnh sửa) ~/.bashrc bằng cách thêm dòng bên dưới vào file đó.

export ANDROID\_HOME=~/Library/Android/sdk

export PATH={PATH}:*PATH*:{ANDROID\_HOME}/tools

* Cài đặt máy ảo Android (AVD) (Có thể sử dụng Genymotion): Để xem danh sách có sẵn AVDs, khởi động "AVD Manager" từ bên trong Android Studio hoặc chạy lệnh sau trong Terminal:

android avd

1. **Props và State**

Trong React, chúng ta điều khiển component bằng cách sử dụng props và state.

1. *Props (viết tắt của Properties)*

Các component nhận props từ component cha. Chúng ta không được thay đổi giá trị của props trong các component này mà chỉ được phép đọc giá trị ra thôi. Trong React thì dữ liệu sẽ đi theo một chiều, có nghĩa là từ component cha => các component con.

Chúng ta có thể tạo ra component sử dụng props. Ý tưởng của props đó là việc trừu tượng hoá các component để có thể sử dụng được ở nhiều chỗ khác nhau trong app.

Ví dụ:

*// Parent*

*export default class ScreenOne extends React.Component {*

*render () {*

*return (*

*<View>*

*<Heading message={'Custom Heading for Screen One'}/>*

*</View>*

*)*

*}*

*}*

*// Child component*

*export default class Heading extends React.Component {*

*render () {*

*return (*

*<View>*

*<Text>{this.props.message}</Text>*

*</View>*

*)*

*}*

*}*

*Heading.propTypes = {*

*message: PropTypes.string*

*}*

*Heading.defaultProps = {*

*message: 'Heading One'*

*}*

1. *State*

State là dữ liệu nội bộ của một Component, trong khi props là dữ liệu được truyền cho Component. Chính vì vậy chúng ta hoàn toàn có thể thay đổi state.

***Lưu ý:*** đừng bao giờ thay đổi trực tiếp biến *this.state*. Thay vào đó hãy dùng hàm *setState* để cập nhật giá trị. Sở dĩ chúng ta cần dùng hàm này là do nó sẽ kích hoạt việc render lại component và tất cả component con nằm trong nó, còn thay đổi *this.state* thì không.

Ví dụ:

*class Form extends React.Component {*

*constructor (props) {*

*super(props)*

*this.state = {*

*input: ''*

*}*

*}*

*handleChangeInput = (text) => {*

*this.setState({ input: text })*

*}*

*render () {*

*const { input } = this.state*

*return (*

*<View>*

*<TextInput style={{height: 40, borderColor: 'gray', borderWidth: 1}}*

*onChangeText={this.handleChangeInput}*

*value={input}*

*/>*

*</View>*

*)*

*}*

*}*

1. **Một số component thông dụng**
2. *Hiển thị thông báo đang tải nội dung*

* Màn hình hiển thị thông báo biểu tượng đang tải dữ liệu vòng tròn

Code mẫu:

*import React, { Component } from 'react'*

*import {*

*ActivityIndicator,*

*AppRegistry,*

*StyleSheet,*

*Text,*

*View,*

*} from 'react-native'*

*export default class App extends Component {*

*render() {*

*return (*

*<View style={[styles.container, styles.horizontal]}>*

*<ActivityIndicator size="large" color="#0000ff" />*

*<ActivityIndicator size="small" color="#00ff00" />*

*<ActivityIndicator size="large" color="#0000ff" />*

*<ActivityIndicator size="small" color="#00ff00" />*

*</View>*

*)*

*}*

*}*

*const styles = StyleSheet.create({*

*container: {*

*flex: 1,*

*justifyContent: 'center'*

*},*

*horizontal: {*

*flexDirection: 'row',*

*justifyContent: 'space-around',*

*padding: 10*

*}*

*})*

*AppRegistry.registerComponent('App', () => App)*

* **Props( kiểu hiển thị)**

+ Đúng, hoặc mặc định (Vòng trong nhỏ quay vòng)

+ Kiểu dữ liệu: bool

+ Cần thiết: không

* **Màu sắc:**

+ Mặc đinh là màu xám

+ Cần thiết: không

* **Kích thước:**

+ Mặc đỉnh là kiểu kích thước nhỏ, thay đổi số liệu kích thước chỉ hệ điều hành android mới hỗ trợ

+ Cú pháp: enum(“small”, “large”),, number

+ Cần thiết : không

* **Ẩn hiển thị khi tải nội dung hoàn tất:**

+ Khi đã tải xong nội dung thì kiểu hiển thị quay tròn biến mất

+ kiểu dữ liệu: bool

+ Cần thiết: Không

+ Nền tảng: IOS

1. *Button (Phím bấm)*

* Các button (phím bấm) sẽ hiển thị mặc định với mỗi nền tảng hệ điều hành
* Nếu như button không hoạt động hay phù hợp với ứng dụng muốn tạo, có thể thay đổi được bằng cách sử dụng:

+ TouchableOpacity

+ TouchableNativeFeedback

Ví dụ:

*import { Button } from 'react-native';*

*...*

*<Button*

*onPress={onPressLearnMore}*

*title="Learn More"*

*color="#841584"*

*accessibilityLabel="Learn more about this purple button"*

*/>*

**Các thức sửa dụng : Bấm nút (click):**

**+** Khi người dùng bấm nút hệ thống sẽ tiến hành xử lý

+ Kểu: chức năng

+ Cần thiết : có

**Dòng tiêu đề:**

+ Hiển thị tiêu đề, giới thiệu

+ Kiểu dữ liệu: String

+ Cần thiết: không

**Mã ID:**

+ Được sử dụng để định vị chế độ xem trong các chế độ thử

+ Kiểu dữ liệu: string

**Tiêu điểm ưa thích cho Tv:**

+ Chỉnh dành cho các Tv của Apple

+ Kiểu bool

+ Nền tảng: IOS

1. *DatePickerIOS ( Chế độ chọn ngày trên IOS)*

* Chế độ chọn ngày trên IOS, thực hiên thao tác vuốt và chọn, dùng cập nhật ngày giờ, hoặc chọn thời gian báo thức, vòng lặp

Ví dụ:

*import React, { Component } from 'react'*

*import {*

*DatePickerIOS,*

*View,*

*StyleSheet,*

*} from 'react-native'*

*export default class App extends Component {*

*constructor(props) {*

*super(props);*

*this.state = { chosenDate: new Date() };*

*this.setDate = this.setDate.bind(this);*

*}*

*setDate(newDate) {*

*this.setState({chosenDate: newDate})*

*}*

*render() {*

*return (*

*<View style={styles.container}>*

*<DatePickerIOS*

*date={this.state.chosenDate}*

*onDateChange={this.setDate}*

*/>*

*</View>*

*)*

*}*

*}*

*const styles = StyleSheet.create({*

*container: {*

*flex: 1,*

*justifyContent: 'center'*

*},*

*})*

**Date (Kiểu ngày tháng):**

+ Kiểu: Date

+ Cần thiết: có

**OnDateChange(Xử lý thay đổi ngày)**

+ Kiểu: Date

+ Cần thiết: có

**MaximumDate (Ngày tối đa cho phép)**

**+** Kiều: Date

+ Cần thiết: có

**MinimumDate (Ngày tối thiểu cho phép)**

**+** Kiểu: Date

+ Cần thiết: có

**MinuteInterval (Lựa chọn phút)**

**+** Kiểu: enum(‘date’, ‘time’, ‘datetime’)

+ Cần thiết: không

**Locale (Biến cục bộ)**

+ Kiểu: string

+ Cần thiết: không

**TimeZoneOffsetInminutes (Chênh lệch múi giời tính bằng phút)**

**+** Kiểu: Number

+ Cần thiết: không

1. *Image (Hình ảnh)*

* Hình ảnh dùng để hiển thị các hình ảnh trên thiết bị, các hình ảnh này có thể được lưu ở đĩa cứng hoặc lưu trên mạng internet
* Lưu ý đối với React native hình ảnh sẽ phải chỉnh thủ công
* Ví dụ:

*import React, { Component } from 'react';*

*import { AppRegistry, View, Image } from 'react-native';*

*export default class DisplayAnImage extends Component {*

*render() {*

*return (*

*<View>*

*<Image*

*source={require('/react-native/img/favicon.png')}*

*/>*

*<Image*

*style={{width: 50, height: 50}}*

*source={{uri: 'https://facebook.github.io/react-native/docs/assets/favicon.png'}}*

*/>*

*<Image*

*style={{width: 66, height: 58}}*

*source={{uri: 'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADMAAAAzCAYAAAA6oTAqAAAAEXRFWHRTb2Z0d2FyZQBwbmdjcnVzaEB1SfMAAABQSURBVGje7dSxCQBACARB+2/ab8BEeQNhFi6WSYzYLYudDQYGBgYGBgYGBgYGBgYGBgZmcvDqYGBgmhivGQYGBgYGBgYGBgYGBgYGBgbmQw+P/eMrC5UTVAAAAABJRU5ErkJggg=='}}*

*/>*

*</View>*

*);*

*}*

*}*

*// skip this line if using Create React Native App*

*AppRegistry.registerComponent('DisplayAnImage', () => DisplayAnImage);*

Có thể thêm style vào một hình ảnh:

*import React, { Component } from 'react';*

*import { AppRegistry, View, Image, StyleSheet } from 'react-native';*

*const styles = StyleSheet.create({*

*stretch: {*

*width: 50,*

*height: 200*

*}*

*});*

*export default class DisplayAnImageWithStyle extends Component {*

*render() {*

*return (*

*<View>*

*<Image*

*style={styles.stretch}*

*source={require('/react-native/img/favicon.png')}*

*/>*

*</View>*

*);*

*}*

*}*

*// skip these lines if using Create React Native App*

*AppRegistry.registerComponent(*

*'DisplayAnImageWithStyle',*

*() => DisplayAnImageWithStyle*

*);*

Hỗ trợ ảnh gif và WebP trên android

Khi xây dựng mã nguồn gốc của riêng bạn, GIF và WebP không được hỗ trợ theo mặc định trên Android.

Bạn sẽ cần phải thêm một số mô-đun tùy chọn android/app/build.gradle, tùy thuộc vào nhu cầu của ứng dụng của bạn.

**Style(Kiểu phong cách)**

+ Kiểu: style

+ Cần thiết: không

• borderTopRightRadius: con số

• backfaceVisibility: enum ('có thể nhìn thấy', 'ẩn')

• borderBottomLeftRadius: con số

• borderBottomRightRadius: con số

• borderColor: màu sắc

• borderRadius: con số

• borderTopLeftRadius: con số

• backgroundColor: màu sắc

• borderWidth: con số

• opacity: con số

• overflow: enum ('có thể nhìn thấy', 'ẩn')

• resizeMode: Object.keys (ImageResizeMode)

• tintColor: màu sắc

**OverlayColor**

Khi hình ảnh đã làm tròn các góc, việc chỉ định overlayColor sẽ làm cho khoảng trống còn lại trong các góc được lấp đầy bằng một màu đồng nhất. Điều này hữu ích trong các trường hợp không được hỗ trợ bởi việc triển khai các góc tròn được Android hỗ trợ:

• Một số chế độ thay đổi kích thước nhất định, chẳng hạn như 'chứa'

• GIF động

Một cách điển hình để sử dụng prop này là với các hình ảnh hiển thị trên một nền tảng vững chắc và thiết lập các overlayColormàu sắc tương tự như nền.

**blurRadius(Làm mờ bộ lọc được thêm vào hình ảnh)**

**+**  Kiểu: number

+ Cần thiết : không

**Onlayout (Trên lớp giao diện)**

**+** Cú pháp : {nativeEvent: {layout: {x, y, width, height}}};

+ Kiểu: function

+ Cần thiết : Không

**OnLoad(Lớp tải trên)**

**+** Kiểu: funtion

+ Cần thiết: không

**OnloadEnd (Tải thành không hoặc không thành công)**

+ Kiểu : funtion

+ Cần thiết: không

**OnloadStart (Bắt đầu tải)**

**+** Cú pháp: onLoadStart={(e) => this.setState({loading: true})}

**+** Kiểu: funtion

+ Cần thiết: không

1. *ResizeMode(Xác định thay đổi kích thước khung ảnh, phù hợp với ảnh thô)*

• cover: Quy mô hình ảnh thống nhất (duy trì tỷ lệ khung hình của hình ảnh) sao cho cả hai chiều (chiều rộng và chiều cao) của hình ảnh sẽ bằng hoặc lớn hơn kích thước tương ứng của chế độ xem (dấu trừ).

• contain: Chia tỷ lệ hình ảnh thống nhất (duy trì tỷ lệ khung hình của hình ảnh) sao cho cả hai chiều (chiều rộng và chiều cao) của hình ảnh sẽ bằng hoặc nhỏ hơn kích thước tương ứng của chế độ xem (dấu trừ).

• stretch: Quy mô chiều rộng và chiều cao một cách độc lập, Điều này có thể thay đổi tỷ lệ khung hình của src.

• repeat: Lặp lại hình ảnh để che khung của chế độ xem. Hình ảnh sẽ giữ kích thước và tỷ lệ khung hình của nó, trừ khi nó lớn hơn khung nhìn, trong trường hợp đó nó sẽ được thu nhỏ lại đồng đều sao cho nó được chứa trong khung nhìn.

• center: Căn giữa hình ảnh trong chế độ xem dọc theo cả hai chiều. Nếu hình ảnh lớn hơn chế độ xem, hãy giảm tỷ lệ đó xuống đồng nhất để hình ảnh được chứa trong chế độ xem.+ Kiểu : enum

+ Cần thiết : Không

**Source (Nguồn ảnh)**

Nguồn hình ảnh (hoặc URL từ xa hoặc tài nguyên tệp cục bộ).

Prop này cũng có thể chứa một số URL từ xa, được chỉ định cùng với chiều rộng và chiều cao của chúng và có khả năng với các đối số quy mô / URI khác. Phía gốc sau đó sẽ chọn tốt nhất uriđể hiển thị dựa trên kích thước được đo của vùng chứa hình ảnh. Một thuộc cachetính có thể được thêm vào để kiểm soát cách Cần thiết nối mạng tương tác với bộ đệm cục bộ.

Các định dạng hỗ trợ hiện tại là png, jpg, jpeg, bmp, gif, webp(chỉ dành cho Android), psd(chỉ iOS).

+ Kiểu: ImageSourcePropType

+ Cần thiết : Không

**loadingIndicatorSource**

Tương tự source, thuộc tính này đại diện cho tài nguyên được sử dụng để hiển thị chỉ báo tải cho hình ảnh, được hiển thị cho đến khi hình ảnh sẵn sàng hiển thị, thường sau khi nó được tải xuống từ mạng. + Kiểu: array

+ Cần thiết : Không

**OnError(Hiển thị lỗi tải)**

Cú pháp: {nativeEvent: {error}}

+ Kiểu : String

+ Cần thiết: không

**ResizeMethod(Phương pháp thay đổi kích thước)**

**+** Thay đổi kích thước hình ảnh khác với kích thước của chế độ xem ảnh, mặc định là auto

+ Auto có thể chọn giữa resize và scale

+ Resize: Một hoạt động phần mềm thay đổi hình ảnh được mã hóa trong bộ nhớ trước khi nó được giải mã. Điều này nên được sử dụng thay vì scale khi hình ảnh lớn hơn nhiều so với chế độ xem.

**+** Scale: Hình ảnh được vẽ xuống hoặc được nâng cấp. So với resize, scalelà nhanh hơn (thường là phần cứng tăng tốc) và tạo ra hình ảnh chất lượng cao hơn. Điều này sẽ được sử dụng nếu hình ảnh nhỏ hơn chế độ xem. Nó cũng nên được sử dụng nếu hình ảnh lớn hơn một chút so với chế độ xem.

+ Kiểu : enum(‘auto’, ‘resize’, ‘scale’)

+ Cần thiết: không

+ Nền tảng: Android

**ImageBackground (Hình ảnh nền)**

* Dùng để xử lý ảnh lên đầu trang ,muốn thêm hình ảnh vào đâu trang chèn hoặc sửa đổi, cần chỉnh thông số thuộc tính chiều cao và chiều rộng
* Ví dụ:

*return (*

*<ImageBackground source={...} style={{width: '100%', height: '100%'}}>*

*<Text>Inside</Text>*

*</ImageBackground>*

*);*

* **Style (Kiểu phong cách)**

**+** Kiểu: View styles

**+** Cần thiết: không

* **ImageStyle(Kiểu ảnh)**

**+** Kiểu: image styles

**+** Cần thiết: Không

* **ImageRef (ảnh Ref)**

**+** Kiểu: Ref

**+** Cần thiết : Không

1. *ListView (Danh sách)*

* Có hai phần:

+ DEPRECATED - sử dụng một trong các thành phần danh sách mới, chẳng hạn như FlatListhoặc SectionListsử dụng bộ nhớ bị hạn chế, ít lỗi hơn, hiệu suất tốt hơn, dễ sử dụng API hơn và nhiều tính năng hơn. Hãy xem bài đăng trên blog này để biết thêm chi tiết.

+LISTVIEW - Một thành phần cốt lõi được thiết kế để hiển thị hiệu quả danh sách các dữ liệu thay đổi theo chiều dọc. API tối thiểu là tạo một ListView.DataSource, điền nó với một mảng dữ liệu đơn giản, và khởi tạo một ListViewthành phần với nguồn dữ liệu đó và một renderRowcuộc gọi lại lấy một blob từ mảng dữ liệu và trả về một thành phần có thể hiển thị.

Ví dụ:

*class MyComponent extends Component {*

*constructor() {*

*super();*

*const ds = new ListView.DataSource({rowHasChanged: (r1, r2) => r1 !== r2});*

*this.state = {*

*dataSource: ds.cloneWithRows(['row 1', 'row 2']),*

*};*

*}*

*render() {*

*return (*

*<ListView*

*dataSource={this.state.dataSource}*

*renderRow={(rowData) => <Text>{rowData}</Text>}*

*/>*

*);*

*}*

*}*

ListView cũng hỗ trợ các tính năng nâng cao hơn, bao gồm các phần có tiêu đề phần dính, hỗ trợ đầu trang và chân trang, gọi lại vào cuối dữ liệu có sẵn ( onEndReached) và trên tập hợp các hàng hiển thị trong thay đổi chế độ xem thiết bị ( onChangeVisibleRows) và một số hiệu suất tối ưu hóa.

Có một vài hoạt động hiệu suất được thiết kế để làm cho ListView cuộn trơn tru trong khi tải dữ liệu có khả năng rất lớn (hoặc khái niệm vô hạn):

• Chỉ hiển thị lại các hàng đã thay đổi - hàm rowHasChanged được cung cấp cho nguồn dữ liệu cho ListView biết nếu nó cần tái kết xuất một hàng vì dữ liệu nguồn đã thay đổi - hãy xem ListViewDataSource để biết thêm chi tiết.

• Hiển thị hàng có giới hạn theo tỷ lệ - Theo mặc định, chỉ có một hàng được hiển thị trên mỗi vòng lặp sự kiện (có thể tùy chỉnh với giá pageSizetreo). Điều này chia nhỏ công việc thành các phần nhỏ hơn để giảm khả năng giảm khung trong khi hiển thị các hàng.Các công cụ:

**Datasource (Dữ liệu Code)**

* Một thể hiện của ListView.DataSource để sử dụng

+ Kiểu: ListViewDataSource

+ Cần thiết: có

initialListSize (cài đặt kích thước )

- Có bao nhiêu hàng để hiển thị trên thành phần ban đầu. Sử dụng điều này để làm cho nó để giá trị màn hình đầu tiên của dữ liệu xuất hiện cùng một lúc thay vì trong quá trình nhiều khung hình.

+ Kiểu: Number

+ Cần thiết: có

onEndReachedThreshold

- Tính theo pixel để gọi trên EndReached

+ Kiểu: Number

+ Cần thiết: có

pageSize(Kích thước trang)

- Số hàng để hiển thị cho mỗi vòng lặp sự kiện. Lưu ý: nếu 'hàng' của bạn thực sự là ô, tức là chúng không kéo dài toàn bộ chiều rộng của chế độ xem của bạn (như trong ListViewGridLayoutExample), bạn nên đặt pageSize thành bội số của mỗi ô, nếu không bạn ' có khả năng thấy các khoảng trống ở cạnh của ListView khi các trang mới được tải.

+ Kiểu: Number

+ Cần thiết: có

renderRow(Tạo cột)

- (rowData, sectionID, rowID, highlightRow) => có thể hiển thị

- Lấy một mục nhập dữ liệu từ nguồn dữ liệu và các id của nó và sẽ trả về một thành phần có thể hiển thị được hiển thị dưới dạng hàng. Theo mặc định dữ liệu là chính xác những gì đã được đưa vào nguồn dữ liệu, nhưng nó cũng có thể cung cấp các trình giải nén tùy chỉnh. ListView có thể được thông báo khi một hàng đang được đánh dấu bằng cách gọi highlightRow(sectionID, rowID). Điều này đặt giá trị boolean của neighborRowHighlighted trong renderSeparator, cho phép bạn kiểm soát các dấu tách ở trên và bên dưới hàng được đánh dấu. Trạng thái được đánh dấu của một hàng có thể được đặt lại bằng cách gọi highlightRow (null).

+ Kiểu: Function

+ Cần thiết: có

renderScrollComponent(Tạo thành phần nút cuận)

- (props) => renderable

Hàm trả về thành phần có thể cuộn trong đó các hàng trong danh sách được hiển thị. Mặc định trả về một ScrollView với các đạo cụ đã cho.

+ Kiểu: Function

+ Cần thiết: có

scrollRenderAheadDistance(Cách tính bắt đầu để hiện trên màn hình)

+ Kiểu : number

+ Cần thiết: Có

stickyHeaderIndices(Hiện ghi chú)

- Làm cho phần tử con ở trên đầu danh sách listview

+ Kiểu: array of number

+ Cần thiết: không

enableEmptySections(Găn cờ, chú thích)

+ Kiểu: bool

+ Cần thiết: không

RenderHeader(dựng đầu mục)

+ Kiểu: function

+ Cần thiết: không

OnEndReached(Liên quan đến sự quận chuột)

+ Kiểu: Function

+ Cần thiết: không

stickySectionHeadersEnabled(các phần tử đính kèm trên đỉnh màn hình)

+ Kiểu: bool

+ Cần thiết: không

renderSeparator

(sectionID, rowID, edgesRowHighlighted) => renderable

- Highlight , nhấn mạnh

+ Kiểu: function

+ Cần thiết : không

onChangeVisibleRows

(visibleRows, changedRows) => void

- Hiển thị hàng, tập hợp các hàng và hiển thị chúng ví dụ như bản đồ

- changedRowsbản đồ {sectionID: {rowID: true | false}}

+ Kiểu: function

+Cần thiết: không

renderFooter

- Hiển thị cuối trang, với listview nằm ngang thì hiển thị cuối trang bên phải, đầu trang bên trái

Phương pháp:

getMetrics()

- Xuất một số dữ liệu, ví dụ như cho các nghiên cứu hoặc phân tích hoàn hảo

scrollTo()

- Cú pháp: scrollTo(...args: Array)

- Cuộn đến một x, y bù đắp, hoặc ngay lập tức hoặc với một hình ảnh động trơn tru.

scrollToEnd()

- Cú pháp: scrollToEnd(([options]: object));

- Nếu đây là một ListView dọc cuộn xuống đáy. Nếu đây là một ListView ngang cuộn sang phải.

- Sử dụng scrollToEnd({animated: true})để cuộn động mượt mà, scrollToEnd({animated: false})để cuộn ngay lập tức. Nếu không có tùy chọn nào được chuyển, animatedmặc định là đúng.

flashScrollIndicators()

- Cú pháp: flashScrollIndicators();

- Hiển thị các thông báo trong giây lát và cuộn xuống rồi biến mất

a. Modal(Hiển thị hay trình bày một nội dung kèm theo)

- Thành phần hiển thị trong khung

- Ví dụ:

import React, {Component} from 'react';

import {Modal, Text, TouchableHighlight, View, Alert} from 'react-native';

class ModalExample extends Component {

state = {

modalVisible: false,

};

setModalVisible(visible) {

this.setState({modalVisible: visible});

}

render() {

return (

<View style={{marginTop: 22}}>

<Modal

animationType="slide"

transparent={false}

visible={this.state.modalVisible}

onRequestClose={() => {

Alert.alert('Modal has been closed.');

}}>

<View style={{marginTop: 22}}>

<View>

<Text>Hello World!</Text>

<TouchableHighlight

onPress={() => {

this.setModalVisible(!this.state.modalVisible);

}}>

<Text>Hide Modal</Text>

</TouchableHighlight>

</View>

</View>

</Modal>

<TouchableHighlight

onPress={() => {

this.setModalVisible(true);

}}>

<Text>Show Modal</Text>

</TouchableHighlight>

</View>

);

}

}

}

Các Công cụ

- Visible(Hiển thị)

- Hiển thị các phương thức có thể nhìn thấy

+ Kiểu: Bool

+ Cần thiết: Không

- SupportedOrientations(Hỗ trợ hiển thị)

- Chỉ mục cho phép xoay theo chiều mong muốn và vẫn bị hạn chế được chỉ định trong trường UISupportedInterfaceOrientations

+ Kiểu: Array

+ Cần thiết: Không

+ Nền tảng: IOS

- OnRequestClose(Yêu cầu với các thiết bị ngoại vi kết nối)

- Như phát ra tín hiệu điều khiển cho TV và nhận trở lại như TV của Apple

+ Kiểu: Function

+ Cần thiết: Có

+Nền tảng: Android, Platform.is TVOS

- OnShow(Hiển thị)

- Các Onshow cho phép truyền đi một hàm khi phương thức đã được chứng minh

+ Kiểu: Function

+ Cần thiết: không

- Transparent (Làm hiệu ứng trong suốt)

- Các transparent sẽ bao phủ lên nền người xem, nếu để chế độ true, nó sẽ trong suốt

+ Kiểu : Bool

+ Cần thiết: Không

- AnimationType

- Các animationTypeđiều khiển prop cách phương thức sinh động.

• slide trang trình bày từ dưới cùng

• fade mờ dần vào chế độ xem

• none xuất hiện không có hoạt ảnh

+ Kiểu: enum(‘none’ ‘slide’ ‘fade’ )

+ Cần thiết: Không

- HardwareAccelerated(Tăng tốc phần cứng)

- Các hardwareAcceleratedđiều khiển prop liệu để buộc tăng tốc phần cứng cho cửa sổ phía dưới.

+ Kiểu: bool

+ Cần thiết: Không

+ Nền tảng: Android

- OnDismiss(Bỏ qua phương thức)

- Các onDismissprop cho phép truyền một hàm sẽ được gọi khi phương thức đã bị loại bỏ

+ Kiểu: Function

+ Cần thiết: Không

+ Nền tảng: IOS

- OnOrientationChage(Các định hướng thay đổi)

- Các onOrientationChangecallback được gọi khi định hướng thay đổi trong khi phương thức đang được hiển thị. Định hướng được cung cấp chỉ là 'chân dung' hoặc 'phong cảnh'. Lệnh gọi lại này cũng được gọi khi hiển thị ban đầu, bất kể định hướng hiện tại.

+ Kiểu : Function

+ Cần thiết: Không

+ Nền tảng: IOS

- PresentationStyle(Phong cách)

- Thay đổi các kích thước cho phù hợp với các thiết bị, như Ipad khác smart phone

- Các chế độ xem:

o Full screen : Toàn màn hình

o PageSheet : bao gồm chế độ xem dọc theo chiều dọc (chỉ trên các thiết bị lớn hơn)

o FormSheet : bao gồm chế độ xem chiều rộng hẹp ở giữa (chỉ trên các thiết bị lớn hơn)

o OverFullScreen: bao gồm toàn bộ màn hình, nhưng cho phép minh bạch

+ Kiểu: enum('fullScreen', 'pageSheet', 'formSheet', 'overFullScreen')

+ Cần thiết: không

+ Nền tảng: IOS

a. ProgressBarAndroid

- Thanh tiến trình trong Android dùng để hiển thị các ứng dụng đang tải hay các ứng dụng hoạt động

Ví dụ:

*import React, { Component } from "react";*

*import {*

*ProgressBarAndroid,*

*AppRegistry,*

*StyleSheet,*

*View*

*} from "react-native";*

*export default class App extends Component {*

*render() {*

*return (*

*<View style={styles.container}>*

*<ProgressBarAndroid />*

*<ProgressBarAndroid styleAttr="Horizontal" />*

*<ProgressBarAndroid styleAttr="Horizontal" color="#2196F3" />*

*<ProgressBarAndroid*

*styleAttr="Horizontal"*

*indeterminate={false}*

*progress={0.5}*

*/>*

*</View>*

*);*

*}*

*}*

*Const styles = StyleSheet.create({*

*container: {*

*flex: 1,*

*justifyContent: "space-evenly",*

*padding: 10*

*}*

*});*

*AppRegistry.registerComponent("App", () => App);*

Các công cụ:

- Animating (Hoạt ảnh)

- Cho hiển thị thanh processBar hiển thị nó true, hoặc ẩn đi là fasle

+ Kiểu: bool

+ Cần thiết: không

- Color(Màu sắc của thanh processBar)

+ Kiểu: color

+ Cần thiết: Không

- Indeterminate

- Nếu thanh tiến trình sẽ hiển thị tiến trình không xác định. Lưu ý rằng điều này chỉ có thể sai nếu styleAttr là Ngang.

+ Kiểu: indeterminate Type

+ Yêu cầu: Không

- Progress (Giá trị 0 hoặc 1)

- Giá trị tiến trình (0 đến 1)

+Kiểu: Number

+ Cần thiết: Không\

- StyleAttr(Phong cách của processBar):

+ Ngang

+ Bình thường (Mặc định)

+ Nhỏ

+ Lớn

+ Nghịch đảo

+ Kiểu: enum

+ Cần thiết: Không

- TestID

- Được sử dụng để định vị chế độ xem này trong các thử nghiệm từ đầu đến cuối.

+ Kiểu: String

+ Cần thiết: Không

a. RefreshControl

- Thành phần này được sử dụng bên trong một ScrollView hoặc ListView để thêm chức năng kéo để làm mới. Khi ScrollView đang ở scrollY: 0, việc vuốt xuống sẽ kích hoạt onRefreshsự kiện.

- Ví dụ:

*class RefreshableList extends Component {*

*constructor(props) {*

*super(props);*

*this.state = {*

*refreshing: false,*

*};*

*}*

*\_onRefresh = () => {*

*this.setState({refreshing: true});*

*fetchData().then(() => {*

*this.setState({refreshing: false});*

*});*

*}*

*render() {*

*return (*

*<ScrollView*

*refreshControl={*

*<RefreshControl*

*refreshing={this.state.refreshing}*

*onRefresh={this.\_onRefresh}*

*/>*

*}*

*...*

*/>*

*);*

*}*

*...*

*}*

- Lưu ý: refreshing là một điều khiển chống đỡ, đây là lý do tại sao nó cần phải được đặt thành true trong onRefreshchức năng nếu không chỉ báo làm mới sẽ dừng ngay lập tức.

- Công cụ:

- Refreshing (Chế độ làm mới)

- Chế độ xem cho biết chế độ làm mới có hoạt động hay không

+ Kiểu: bool

+ Cần thiết: có

- Onfresh

- Được gọi khi chế độ Refrshing bắt đầu

+ Kiểu: Functiom

+ Cần thiết: không

- Colors(Màu sắc)

- Các màu sẽ được vẽ khi làm mới

+ Kiểu : Colors

+ Cần thiết: Không

+ Nền tảng: Android

* **Enabled (Cho phép )**
* Cho phép hiển thị dù chức năng Refreshing được bật hay chưa

+ Kiểu: Bool

+ Cần thiết: Không

+ Nền tảng: Android

* **ProgressBackgroundColor**
* Màu nền của thanh Progress

+ Kiểu: Color

+ Cần thiết :Không

+ Nền tảng: Android

* **ProgressViewOffset**
* Chế độ xem trang tiến độ

+ Kiểu: Number

+ Cần thiết: Không

+ Nền tảng: Android

* **Size(Kích thước)**

**+** Kiểu: enum

**+** Cần thiết: Không

**+** Nền tảng: Android

* **TintColor**
* Màu của chỉ báo làm mới

+ Kiểu: Color

+ Cần thiết: Không

+ Nền tảng: IOS

* **Title(Tiêu đề)**

**+** Kiểu: String

**+** Cần thiết: Không

**+** Nền tảng: IOS

* **TitleColor**
* Màu của tiêu đề

+ Kiểu: String

+ Cần thiết: Không

+ Nền tảng: IOS

1. *Slider(Dạng chuyển động hình ảnh)*

**Công cụ:**

* **Style(Phong cách)**

**-** Được sử dụng để tạo kiểu và bố cục Slider. Xem StyleSheet.jsvà ViewStylePropTypes.jsđể biết thêm thông tin.

+ Kiểu: View.style

+ Cần thiết: Không

- Disabled

- Nếu giá trị là true người dùng sẽ không thể di chuyển thanh slider, còn là False thì có thể

**+** Kiểu:Bool

**+** Cần thiết: Không

* **MaximumValue(Gia trị tối đa của thanh Slider, mặc định là 1)**

**+** Kiểu: Con số

**+** Cần thiết: Không

* **MinimumTrackTintColor**
* Màu được sử dụng cho bên trái của nút bấm, bên hệ điều hành IOS là ghi đè lên màu xanh mặc định

+ Kiểu: Color

+ Cần thiết: Không

* **MinimumValue(Giá trị tối thiểu của thanh Slider, Giá trị mặc định là bằng 0)**

**+** Kiểu: Number

**+** Cần thiết: Không

- onSlidingComplete

- Gọi lại được gọi khi người dùng nhả thanh trượt, bất kể giá trị đã thay đổi hay chưa. Giá trị hiện tại được chuyển như một đối số cho trình xử lý gọi lại.

+Kiểu: Function

+ Cần thiết: Không

- onValueChange

- Gọi lại liên tục được gọi trong khi người dùng đang kéo thanh trượt.

+ Kiểu: Number

+ Cần thiết: Không

- maximumTrackTintColor

- Màu được sử dụng cho bản nhạc ở bên phải của nút. Ghi đè hình ảnh màu xám mặc định trên iOS

+ Kiểu: Color

+ Cần thiết: Không

- maximumTrackImage

- Chỉ định hình ảnh bản nhạc tối đa. Chỉ hỗ trợ hình ảnh tĩnh. Pixel ngoài cùng bên trái của hình ảnh sẽ được kéo dài để lấp đầy đường đi.

+ Kiểu: Image.propTypes.source

+ Cần thiết: Không

+ Nền tảng: IOS

- thumbImage

- Chỉ định một hình ảnh cho bản nhạc đó. Chỉ hỗ trợ hình ảnh tĩnh. Pixel trung tâm của hình ảnh sẽ được kéo dài để lấp đầy đường đi.

+Kiểu: Image.propTypes.source

+ Cần thiết: Không

+ Nền tảng: IOS

1. *SelectionList*

- Giao diện trình diễn để hiển thị danh sách được phân đoạn, hỗ trợ các tính năng tiện dụng nhất:

• Hoàn toàn đa nền tảng.

• Callback khả năng xem có thể định cấu hình.

• Hỗ trợ tiêu đề danh sách.

• Hỗ trợ chân trang danh sách.

• Hỗ trợ phân tách mục.

• Hỗ trợ tiêu đề mục.

• Hỗ trợ phân cách phần.

• Dữ liệu không đồng nhất và hỗ trợ hiển thị mục.

• Kéo để làm mới.

• Cuộn tải.

* **Ví Dụ:**

// Example 1 (Homogeneous Rendering)

<SectionList

renderItem={({item, index, section}) => <Text key={index}>{item}</Text>}

renderSectionHeader={({section: {title}}) => (

<Text style={{fontWeight: 'bold'}}>{title}</Text>

)}

sections={[

{title: 'Title1', data: ['item1', 'item2']},

{title: 'Title2', data: ['item3', 'item4']},

{title: 'Title3', data: ['item5', 'item6']},

]}

keyExtractor={(item, index) => item + index}

/>

// Example 2 (Heterogeneous Rendering / No Section Headers)

const overrideRenderItem = ({ item, index, section: { title, data } }) => <Text key={index}>Override{item}</Text>

<SectionList

renderItem={({ item, index, section }) => <Text key={index}>{item}</Text>}

sections={[

{ title: 'Title1', data: ['item1', 'item2'], renderItem: overrideRenderItem },

{ title: 'Title2', data: ['item3', 'item4'] },

{ title: 'Title3', data: ['item5', 'item6'] },

]}

/>

Đây là một wrapper tiện lợi xung quanh <VirtualizedList>, và do đó thừa hưởng đạo cụ của nó (cũng như những <ScrollView>cái mà không được liệt kê rõ ràng ở đây, cùng với những điều sau đây hãy cẩn thận:

• Trạng thái nội bộ không được giữ nguyên khi nội dung cuộn ra khỏi cửa sổ hiển thị. Đảm bảo tất cả dữ liệu của bạn được ghi lại trong dữ liệu mục hoặc các cửa hàng bên ngoài như Flux, Redux hoặc Relay.

• Đây là một PureComponentđiều có nghĩa là nó sẽ không tái hiện lại nếu propsvẫn còn nông bằng nhau. Đảm bảo rằng tất cả mọi thứ mà renderItemchức năng của bạn phụ thuộc vào được truyền dưới dạng bản sao (ví dụ extraData) không phải ===sau khi cập nhật, nếu không giao diện người dùng của bạn có thể không cập nhật các thay đổi. Điều này bao gồm các datatrạng thái prop và cha mẹ thành phần.

• Để hạn chế bộ nhớ và cho phép cuộn trơn tru, nội dung được hiển thị không đồng bộ trên màn hình không đồng bộ. Điều này có nghĩa là có thể cuộn nhanh hơn tỷ lệ lấp đầy và trong giây lát nhìn thấy nội dung trống. Đây là một sự cân bằng có thể được điều chỉnh cho phù hợp với nhu cầu của từng ứng dụng và chúng tôi đang nỗ lực cải thiện nó sau hậu trường.

• Theo mặc định, danh sách tìm kiếm giá trị cho keymỗi mục và sử dụng nó cho khóa Phản hồi. Ngoài ra, bạn có thể cung cấp một tùy chỉnh keyExtractorchống đỡ.

- Các Công cụ:

- sections

- Các dữ liệu thực tế để render, giống như các dataprop trong FlatList.

+ Kiểu: Array of Sections

+ Cần thiết: Có

- initialNumToRender

- Có bao nhiêu mục để hiển thị trong lô ban đầu. Điều này là đủ để lấp đầy màn hình nhưng không nhiều hơn nữa. Lưu ý rằng các mục này sẽ không bao giờ được bỏ gắn kết như một phần của hiển thị cửa sổ để cải thiện hiệu suất cảm nhận của các hành động cuộn lên đầu.

+ Kiểu: Number

+ Cần thiết: Có

- renderItem

- Trình kết xuất mặc định cho mọi mục trong mỗi phần. Có thể bị ghi đè trên cơ sở từng phần. Nên trả về một phần tử React.

+Kiểu: Function

+ Cần thiết: Có

- Hàm render sẽ được truyền một đối tượng với các phím sau:

• 'item' (đối tượng) - đối tượng mục như được chỉ định trong datakhóa của phần này

• 'index' (số) - Chỉ mục của mục trong phần.

• 'section' (đối tượng) - Đối tượng phần đầy đủ như được chỉ định trong sections.

• 'separators' (đối tượng) - Một đối tượng với các phím sau:

o 'highlight' (chức năng) - () => void

o 'unhighlight' (chức năng) - () => void

o 'updateProps' (hàm) - (select, newProps) => void

 'select' (enum) - các giá trị có thể là 'hàng đầu', 'dấu'

 'newProps' (đối tượng)

- onEndReached

- Được gọi một lần khi vị trí cuộn nằm trong onEndReachedThresholdnội dung được hiển thị.

+ Kiểu: [(info: {distanceFromEnd: number}) => void]

+Yêu cầu: Không

- extraData

- Thuộc tính điểm đánh dấu để cho biết danh sách sẽ hiển thị lại (vì nó được thực hiện PureComponent). Nếu bất kỳ renderItemchức năng nào của bạn , Tiêu đề, Chân trang, v.v. phụ thuộc vào bất kỳ thứ gì ngoài giá trị data, hãy dán nó vào đây và xử lý nó một cách bất biến.

+ Kiểu: Any

+ Cần thiết: Không

- inverted

- Đảo ngược hướng cuộn. Sử dụng các biến đổi tỷ lệ của -1.

+ Kiểu: Boolean

+Cần thiết: Không

- ListFooterComponent

- Hiển thị ở cuối danh sách. Có thể là Lớp thành phần phản ứng, hàm trả về hoặc phần tử được hiển thị.

+Kiểu: [component, function, element]

+ Cần thiết: Không

- legacyImplementation

+ Kiểu: Boolean

+Cần thiết: Không

- ListEmptyComponent

- Hiển thị khi danh sách trống. Có thể là Lớp thành phần phản ứng, hàm trả về hoặc phần tử được hiển thị.

+ Kiểu: [component, function, element]

+ Cần thiết: Không

- onEndReachedThreshold

- Cách xa kết thúc (trong đơn vị độ dài nhìn thấy được của danh sách) cạnh dưới của danh sách phải từ cuối nội dung để kích hoạt onEndReachedgọi lại. Do đó, giá trị 0,5 sẽ kích hoạt onEndReachedkhi kết thúc nội dung nằm trong khoảng một nửa độ dài hiển thị của danh sách.

+ Kiểu: Number

+ Cần thiết: Không

- onRefresh

- Nếu được cung cấp, một RefreshControl tiêu chuẩn sẽ được thêm vào cho chức năng "Kéo để làm mới". Đảm bảo đặt cũng refreshingchính xác.

+ Kiểu: [() => void]

+ Cần thiết: Không

- onViewableItemsChanged

- Được gọi khi khả năng xem của các hàng thay đổi, như được xác định bởi giá viewabilityConfig đỡ.

+Kiểu: Function

+ Cần thiết: Không

Hàm sẽ được truyền một đối tượng với các phím sau:

• 'viewableItems' (mảng của ViewTokens)

• 'đã thay đổi' (mảng của ViewTokens)

Các ViewTokenloại được xuất khẩu bởi ViewabilityHelper.js:

|  |  |  |
| --- | --- | --- |
| **Tên** | **Kiểu** | **Cần Thiết** |
| Item | any | Yes |
| Key | String | Yes |
| Index | Number | No |
| isViewable | Boolean | Yes |
| Section | Any | No |

- refreshing

- Đặt điều này đúng trong khi đợi dữ liệu mới từ làm mới.

+ Kiểu: Boolean

+ Cần thiết: Không

- removeClippedSubviews

- Lưu ý: có thể có lỗi (thiếu nội dung) trong một số trường hợp - sử dụng có nguy cơ của riêng bạn.

- Điều này có thể cải thiện hiệu suất cuộn cho các danh sách lớn.

+ Kiểu: Boolean

+ Cần thiết: Không

- ListHeaderComponent

- Hiển thị ở đầu danh sách. Có thể là Lớp thành phần phản ứng, hàm trả về hoặc phần tử được hiển thị.

+Kiểu: component, function, element

+Cần thiết: Không

- renderSectionFooter

- Hiển thị ở cuối mỗi phần.

+Kiểu: [(info: {section: SectionT}) => ?React.Element]

+ Cần thiết: Không

- SectionSeparatorComponent

- Hiển thị ở trên cùng và dưới cùng của mỗi phần (lưu ý điều này khác với ItemSeparatorComponent chỉ được hiển thị giữa các mục). Chúng được thiết kế để phân tách các phần khỏi các tiêu đề ở trên và dưới và thường có cùng một phản hồi nổi bật như ItemSeparatorComponent. Cũng nhận được highlighted, [leading/trailing][Item/Separator]và bất kỳ đạo cụ tùy chỉnh từ separators.updateProps.

+Kiểu: ReactClass

+ Cần thiết: Không

- stickySectionHeadersEnabled

- Làm cho tiêu đề phần dính vào đầu màn hình cho đến khi tiêu đề tiếp theo đẩy nó ra. Chỉ được bật theo mặc định trên iOS vì đó là tiêu chuẩn nền tảng ở đó.

+Kiểu:Boolean

+ Cần thiết: Không

- Định nghĩa

- Selection

- Một đối tượng xác định dữ liệu được hiển thị cho một Selection nhất định.

+Kiểu: Any

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **Data** | Array | Dữ liệu để hiển thị các mục trong phần này. Mảng các đối tượng, giống như FlatListdữ liệu của prop |
| **Key** | String | Phím tùy chọn để theo dõi việc sắp xếp lại phần. Nếu bạn không có kế hoạch đặt lại các phần, chỉ mục mảng sẽ được sử dụng theo mặc định. |
| **Renderltem** | Function | Tùy chọn xác định trình kết xuất mục tùy ý cho phần này, ghi đè mặc định renderItemcho danh sách. |
| **ItemSeparatorComponent** | component, function, element | Tùy ý xác định một dấu phân cách mục tùy ý cho phần này, ghi đè mặc định ItemSeparatorComponentcho danh sách. |
| **KeyExtractor** | Function | Tùy chọn xác định một bộ giải mã khóa tùy ý cho phần này, ghi đè mặc định keyExtractor. |

1. *ToolbarAndroid*

* Toolbar tiện ích chỉ dành cho Android. Thanh công cụ hiển thị biểu tượng, biểu tượng điều hướng, tiêu đề, phụ đề và các danh sách tác vụ
* Tiêu đề và phụ đề được mở rộng để biểu trưng và biểu tượng, điều hướng được hiển thị phí bên trái và , tiêu đề và phụ đề giữa các hành động
* Nếu thanh công cụ chỉ chứa 1 thì sẽ hiển thị ở giữa

Ví dụ:

*render: function() {*

*return (*

*<ToolbarAndroid*

*logo={require('./app\_logo.png')}*

*title="AwesomeApp"*

*actions={[{title: 'Settings', icon: require('./icon\_settings.png'), show: 'always'}]}*

*onActionSelected={this.onActionSelected} />*

*)*

*},*

*onActionSelected: function(position) {*

*if (position === 0) { // index of 'Settings'*

*showSettings();*

*}*

*}*

* **Các công cụ:**
* **OverflowIcon**
* Biểu tượng

+ Kiểu: optionallmageSoure

+ Cần thiết: Không

* **Actions**
* Đặt các hành động trên thanh công cụ trong phần menu tác vụ, dạng hiển thị là văn bản tiện ích, nếu kích thước không vừa thì sẽ được đặt trong menu “overflow”
* Title: Bắt buộc, là tiêu đề của hành động
* Icon: biểu tượng vidu: (*require('./some\_icon.png')*)
* Show: Thời gian hiển thị hành động này như một biểu tượng hoặc giấu nó trong menu “overflow”: always, ifrom, hoặc never
* ShowWithText: boolean. Có hiển thị văn bản

+ Kiểu: Array or object: {title: string,icon: optionalImageSource,show: enum('always', 'ifRoom', 'never'),showWithText: bool}

+ Cần thiết: không

- contentInsetStart

- Đặt phần tử nội dung cho cạnh bắt đầu của thanh công cụ.

- Chèn nội dung ảnh hưởng đến khu vực hợp lệ cho nội dung của Thanh công cụ ngoài nút điều hướng và menu. Insets xác định tỷ lệ tối thiểu cho các thành phần này và có thể được sử dụng để căn chỉnh hiệu quả nội dung Thanh công cụ dọc theo các đường lưới nổi tiếng.

+ Kiểu: Number

+ Cần thiết: không

* **Logo(Đặt biểu tượng cho thanh công cụ)**

+ Kiểu: optionallmageSoure

+ Cần thiết: không

* **Navicon**
* Đặt biểu tượng điều hướng

+ Kiểu:Function

+ Cần thiết: Không

* onActionSelected
* Gọi tới hàm được chọn, đối số là gọi tới vị trí của hàm
* + Kiểu: Function
* + Cần thiết: Không
* onIconClicked
* Gọi lại khi biểu tượng được chọn
* + Kiểu: Function
* + Cần thiết: không
* onIconClicked
* Đặt nội dung cần thêm cho cạnh của thanh công cụ, chèn nội dung sẽ ảnh hưởng đến nội dung của thanh Toolbar
* + Kiểu: Number
* + Cần thiết: Không
* **Rtl**
* Được sử dụng để đặt hướng thanh công cụ thành RTL. Ngoài thuộc tính này, bạn cần phải thêm
* android: supportRtl = "true"
* vào ứng dụng AndroidManifest.xml của bạn và sau đó gọi setLayoutDirection(LayoutDirection.RTL)trong onCreatephương thức MainActivity của bạn .

+ Kiểu: bool

+ Cần thiết: không

* **Subtitle**
* Đặt phụ đề thanh công cụ

+ Kiểu: String

+ Cần thiết: không

* **SubtitleColor**
* Đặt màu phụ đề thanh công cụ

+ Kiểu: Color

+ Cần thiết : không

* **TestID**
* Được sử dụng để định vị chế độ xem này trong các thử nghiệm từ đầu đến cuối.

+Kiểu: String

+ Cần thiết: Không

* **Title**
* Đặt tiêu đề thanh công cụ

+ Kiểu: String

+ Cần thiết: Không

* TitleColor
* Đặt màu cho tiêu đề thanh công cụ

+ Kiểu: Color

+ Cần thiết: Không

1. *View*

* Thành phần cơ bản nhất dùng để xây dựng giao diện người dùng là View,
* View được thế kế để lồng vào các chế độ xem khác nhau và có chứa nhiều lớp con
* Ví dụ: hiển thị 2 màu với View trong hàng (row)

*class ViewColoredBoxesWithText extends Component {*

*render() {*

*return (*

*<View*

*style={{*

*flexDirection: 'row',*

*height: 100,*

*padding: 20,*

*}}>*

*<View style={{backgroundColor: 'blue', flex: 0.3}} />*

*<View style={{backgroundColor: 'red', flex: 0.5}} />*

*<Text>Hello World!</Text>*

*</View>*

*);*

*}*

*}*

* View được sử dụng chung với StyleSheet để rõ ràng hơn về hiệu suất
* **Synthetic Touch Events**
* Tổng hợp với các sự kiện cảm ứng
* nativeEvent
* changedTouches
* identifier
* locationX
* locationY
* pageX
* pageY
* target
* timestamp
* touches
* **Công cụ:**
* onStartShouldSetResponder
* Chế độ xem muốn phản hồi khi bắt đầu một lần chạm
* View.props.onStartShouldSetResponder: (event) => [true | false], event

+ Kiểu: Function

+ Cần thiết: Không

* accessibilityHint
* Gợi ý trợ năng giúp người dùng hiểu điều gì sẽ xảy ra khi họ thực hiện hành động trên phần tử trợ năng khi kết quả đó không rõ ràng từ nhãn trợ năng.

+ Kiểu: String

+ Cần thiết: Không

* hitSlop
* Điều này xác định khoảng cách mà sự kiện chạm có thể bắt đầu từ chế độ xem. Nguyên tắc giao diện điển hình đề xuất các mục tiêu cảm ứng có ít nhất 30 - 40 điểm / pixel độc lập với mật độ.
* For example, if a touchable view has a height of 20 the touchable height can be extended to 40 with hitSlop={{top: 10, bottom: 10, left: 0, right: 0}}

+ Kiểu: object: {top: number, left: number, bottom: number, right: number}

+ Cần thiết : Không

* nativeID
* Được sử dụng để định vị chế độ xem này từ các lớp gốc.

+Kiểu: String

+ Cần thiết: Không

* onAccessibilityTap
* Khi accessibleđúng, hệ thống sẽ cố gắng gọi hàm này khi người dùng thực hiện thao tác chạm trợ năng.

+ Kiểu: Function

+ Cần thiết: Không

* onLayout
* Cú pháp: {nativeEvent: { layout: {x, y, width, height}}}
* Sự kiện này được kích hoạt ngay lập tức khi bố cục đã được tính toán, nhưng bố cục mới có thể chưa được phản ánh trên màn hình tại thời điểm sự kiện được nhận, đặc biệt nếu một hoạt ảnh bố cục đang được tiến hành.

+ Kiểu: Function

+ Cần thiết: Không

* **OnMagicTap**
* Khi accessible là true, hệ thống sẽ gọi hàm này

+ Kiểu: Function

+ Cần thiết : Không

* onMoveShouldSetResponder
* Chế độ xem này có muốn "xác nhận quyền sở hữu" phản hồi cảm ứng không? Điều này được gọi cho mọi cử động cảm ứng Viewkhi nó không phải là người trả lời.
* Cú pháp: View.props.onMoveShouldSetResponder: (event) => [true | false], event

+ Kiểu: Function

+ Cần thiết: Không

* onMoveShouldSetResponderCapture
* Nếu lớp cha View muốn ngăn chặn lớp con View yêu cầu phản hồi khi bị di duyển nó sẽ trả về True
* Cú pháp: View.props.onMoveShouldSetResponderCapture: (event) => [true | false], event

+ Kiểu: Function

+ Cần thiết: Không

* onResponderGrant
* Chế độ xem hiện đang phản hồi cho các sự kiện chạm. Đây là thời điểm để làm nổi bật và hiển thị cho người dùng những gì đang xảy ra.
* Cú pháp: View.props.onResponderGrant: (event) => {}, event

+ Kiểu: Function

+ Cần thiết: Không

* onResponderMove
* Di chuyển reponder
* Cú pháp: View.props.onResponderMove: (event) => {}, event

+ Kiểu: Function

+ Cần thiết: Không

* onResponderReject
* Cú pháp: View.props.onResponderRelease: (event) => {}, event

+ Kiểu: Function

+ Cần thiết: Không

* onResponderTerminate
* Trả lời đã được lấy từ View. Có thể được thực hiện bởi các chế độ xem khác sau khi gọi onResponderTerminationRequesthoặc có thể được hệ điều hành thực hiện mà không hỏi (ví dụ: xảy ra với trung tâm điều khiển / trung tâm thông báo trên iOS)
* Cú pháp: View.props.onResponderTerminate: (event) => {}, where event

+ Kiểu: Function

+ Cần thiết: Không

* onResponderTerminationRequest
* Cú pháp : View.props.onResponderTerminationRequest: (event) => {}, where event

+ Kiểu: Function

+ Cần thiết: Không

* accessible
* Khi true, cho biết chế độ xem là yếu tố trợ năng. Theo mặc định, tất cả các yếu tố có thể chạm đều có thể truy cập được.

+ Kiểu: Bool

+ Cần thiết: Không

* onStartShouldSetResponderCapture
* Nếu Nếu lớp cha View muốn chặn lớp con View trình xử lý sẽ trả về true
* Cú pháp: View.props.onStartShouldSetResponderCapture: (event) => [true | false], event

+ Kiểu: Function

+ Cần thiết: Không

1. *Text*

* Văn bản là hiển thị cần thiết nhất và quan trọng nhất đối với React
* Hỗ trợ từ bàn phím hoặc cảm ứng
* Ví dụ về văn bản:

*import React, { Component } from 'react';*

*import { AppRegistry, Text, StyleSheet } from 'react-native';*

*export default class TextInANest extends Component {*

*constructor(props) {*

*super(props);*

*this.state = {*

*titleText: "Bird's Nest",*

*bodyText: 'This is not really a bird nest.'*

*};*

*}*

*render() {*

*return (*

*<Text style={styles.baseText}>*

*<Text style={styles.titleText} onPress={this.onPressTitle}>*

*{this.state.titleText}{'\n'}{'\n'}*

*</Text>*

*<Text numberOfLines={5}>*

*{this.state.bodyText}*

*</Text>*

*</Text>*

*);*

*}*

*}*

*const styles = StyleSheet.create({*

*baseText: {*

*fontFamily: 'Cochin',*

*},*

*titleText: {*

*fontSize: 20,*

*fontWeight: 'bold',*

*},*

*});*

*// skip this line if using Create React Native App*

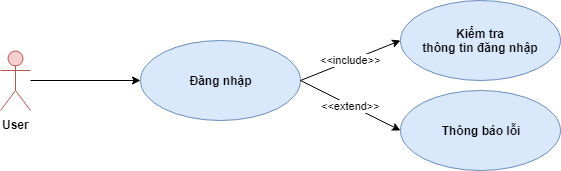
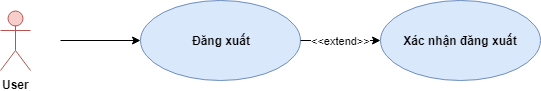
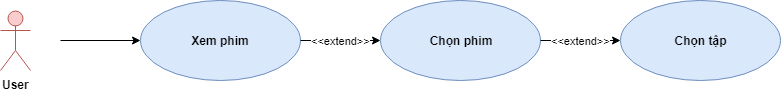
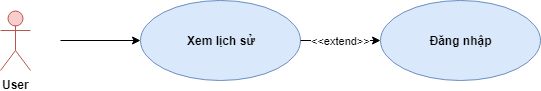
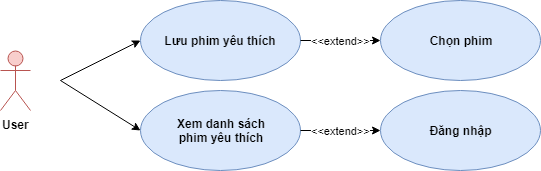
*AppRegistry.registerComponent('TextInANest', () => TextInANest);*

1. **PHÁT BIỂU BÀI TOÁN**
2. **Khảo sát hiện trạng**

Hiện nay, trong một xã hội hiện đại và phát triển nhanh chóng, đòi hỏi con người phải làm việc ngày một nhiều hơn, do vậy nhu cầu giải trí cũng tăng cao. Sau một ngày làm mệt mỏi, mọi người đều muốn chọn cho mình một sở thích riêng, có người thì thích xem phim, nghe nhạc, đọc sách, chơi game,… Hiện nay, sống trong một thời đại công nghệ hóa ngày càng rõ rệt, các nhu cầu đơn giản, thiết thực ấy hầu như đều có thể đáp ứng được bằng công nghệ thông tin, đặc biệt là thiết bị di động, điển hình là smartphone. Do vậy, ứng dụng xem phim sẽ đáp ứng được nhu cầu giải trí phần nào, vì thế, nhóm chọn đề tài Ứng dụng xem phim với React Native để làm đồ án cuối kỳ.

1. **Tiện ích ứng dụng**

|  |  |  |
| --- | --- | --- |
| STT | Nhóm tiện ích | Tên tiện tích |
| 1 | Authentication | Đăng nhập |
| Đăng xuất |
| 2 | Xem phim | Tìm kiếm phim |
| Xem phim |
| Xem lịch sử phim đã xem |
| Lưu phim yêu thích |

1. **SƠ ĐỒ ĐẶC TẢ**
2. **Use-case diagram**
   1. **Đăng nhập**
   2. **Đăng xuất**
   3. **Tìm kiếm phim**
3.  Tìm kiếm theo danh mục
4. **** Tìm kiếm theo tên
   1. **Xem phim**
   2. **Xem lịch sử phim đã xem**
   3. **Lưu phim yêu thích**
5. **Đặc tả Use-case**
   1. **Đăng nhập**
6. Đăng nhập bằng số điện thoại

* Tóm tắt: Người dùng có thể đăng nhập bằng số điện thoại.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Đăng nhập” trên giao diện Cá nhân, hiển thị giao diện Đăng nhập
* Người dùng cần nhập số điện thoại vào TextInput rồi click vào button “Đăng nhập”

+ Các dòng sự kiện khác

* Nếu người dùng nhập đúng số điện thoại, sẽ chuyển tới giao diện nhập mã xác nhận (đồng thời hệ thống gửi 1 mã xác nhận tới số điện thoại vừa nhập)
* Nếu người dùng nhập đúng mã xác nhận thì đăng nhập thành công và chuyển tới giao diện Cá nhân
* Nếu người dùng không nhập đầy đủ thông tin, hệ thống sẽ hiển thị thông báo.
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng chưa đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Đăng nhập.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : Trở về giao diện Cá nhân.
2. Thực hiện thất bại : Ở lại giao diện Đăng nhập và xuất hiện thông báo.
3. Đăng nhập bằng Facebook

* Tóm tắt: Người dùng có thể đăng nhập bằng tài khoản Facebook.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Đăng nhập” trên giao diện Cá nhân, hiển thị giao diện Đăng nhập
* Người dùng click vào button “Đăng nhập bằng Facebook”

+ Các dòng sự kiện khác

* Nếu người dùng đã đăng nhập tài khoản Facebook trên thiết bị thì đăng nhập thành công và chuyển tới giao diện Cá nhân
* Nếu người dùng chưa đăng nhập tài khoản Facebook trên thiết bị thì hiển thị giao diện đăng nhập Facebook
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng chưa đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Đăng nhập.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : Trở về giao diện Cá nhân.
2. Thực hiện thất bại : Ở lại giao diện Đăng nhập và xuất hiện thông báo.
3. Đăng nhập bằng Google+

* Tóm tắt: Người dùng có thể đăng nhập bằng tài khoản Google+.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Đăng nhập” trên giao diện Cá nhân, hiển thị giao diện Đăng nhập
* Người dùng click vào button “Đăng nhập bằng Google+”

+ Các dòng sự kiện khác

* Nếu người dùng đã đăng nhập tài khoản Google+ trên thiết bị thì đăng nhập thành công và chuyển tới giao diện Cá nhân
* Nếu người dùng chưa đăng nhập tài khoản Google+ trên thiết bị thì hiển thị giao diện đăng nhập Google+
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng chưa đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Đăng nhập.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : Trở về giao diện Cá nhân.
2. Thực hiện thất bại : Ở lại giao diện Đăng nhập và xuất hiện thông báo.
   1. **Đăng xuất**

* Tóm tắt: Người dùng có thể đăng xuất.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Đăng xuất” trên giao diện Cá nhân, hiển thị thông báo xác nhận đăng xuất gồm 2 button “Xác nhận” và “Hủy bỏ”
* Chọn “Xác nhận” để thực hiện thao tác đăng xuất, hoặc chọn “Hủy bỏ” để hủy bỏ thao tác đăng xuất
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đã đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Cá nhân.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : Trở về giao diện Cá nhân.
2. Thực hiện thất bại : Ở lại giao diện Cá nhân và xuất hiện thông báo.
   1. **Tìm kiếm phim**
   2. Tìm kiếm theo danh mục

* Tóm tắt: Người dùng có thể tìm kiếm phim theo danh mục.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Danh mục”, sau đó click chọn danh mục trên giao diện Danh mục, hiển thị giao diện danh sách các phim theo danh mục vừa chọn
* Người dùng click vào phim muốn xem
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đang ở giao diện Danh mục.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : chuyển tới giao diện xem phim.
2. Thực hiện thất bại : Ở lại giao diện Danh mục và xuất hiện thông báo.
   1. Tìm kiếm theo tên

* Tóm tắt: Người dùng có thể tìm kiếm phim theo tên.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Tìm kiếm”, sau đó nhập từ khóa muốn tìm kiếm vào TextInput trên giao diện Tìm kiếm, hiển thị danh sách các phim phù hợp với từ khóa
* Người dùng click vào phim muốn xem
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đang ở giao diện Tìm kiếm.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : Chuyển tới giao diện xem phim.
2. Thực hiện thất bại : Ở lại giao diện Tìm kiếm và xuất hiện thông báo.
   1. **Xem phim**

* Tóm tắt: Người dùng có thể xem phim.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click chọn phim muốn xem trên giao diện Home hoặc thông qua tìm kiếm, hiển thị giao diện Xem phim.
* Chọn tập muốn xem.
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đang ở giao diện Home hoặc giao diện Danh mục/Tìm kiếm

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : chuyển tới giao diện Xem phim.
2. Thực hiện thất bại : Ở lại giao diện trước đó và xuất hiện thông báo.
   1. **Xem lịch sử phim đã xem**

* Tóm tắt: Người dùng có thể xem lịch sử phim đã xem.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Lịch sử” trên giao diện Cá nhân, hiển thị giao diện Lịch sử gồm danh sách các phim mà người dùng đã xem theo thứ tự thời gian từ gần nhất đến xa nhất
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đã đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Cá nhân.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : chuyển tới giao diện Lịch sử.
2. Thực hiện thất bại : Ở lại giao diện Cá nhân và xuất hiện thông báo.
   1. **Lưu phim yêu thích**
   2. Lưu phim yêu thích

* Tóm tắt: Người dùng có thể lưu lại phim mình yêu thích.
* Dòng sự kiện

+ Dòng sự kiện chính

* Khi người dùng click vào button “Like” hình trái tim rỗng trên poster phim mình thích, hệ thống sẽ lưu lại phim này vào danh sách yêu thích của người dùng, đồng thời thay đổi button “Like” thành hình trái tim màu đỏ
* Người dùng có thể click button “Like” lần nữa để xóa phim khỏi danh sách yêu thích
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

1. Người dùng đã đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Home/Tìm kiếm/Lịch sử hoặc danh sách phim thuộc danh mục tương ứng.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : ở lại giao diện hiện tại.
2. Thực hiện thất bại : Ở lại giao diện hiện tại và xuất hiện thông báo.
   1. Xem danh sách phim yêu thích

* Tóm tắt: Người dùng có thể xem danh sách phim yêu thích.
* Dòng sự kiện

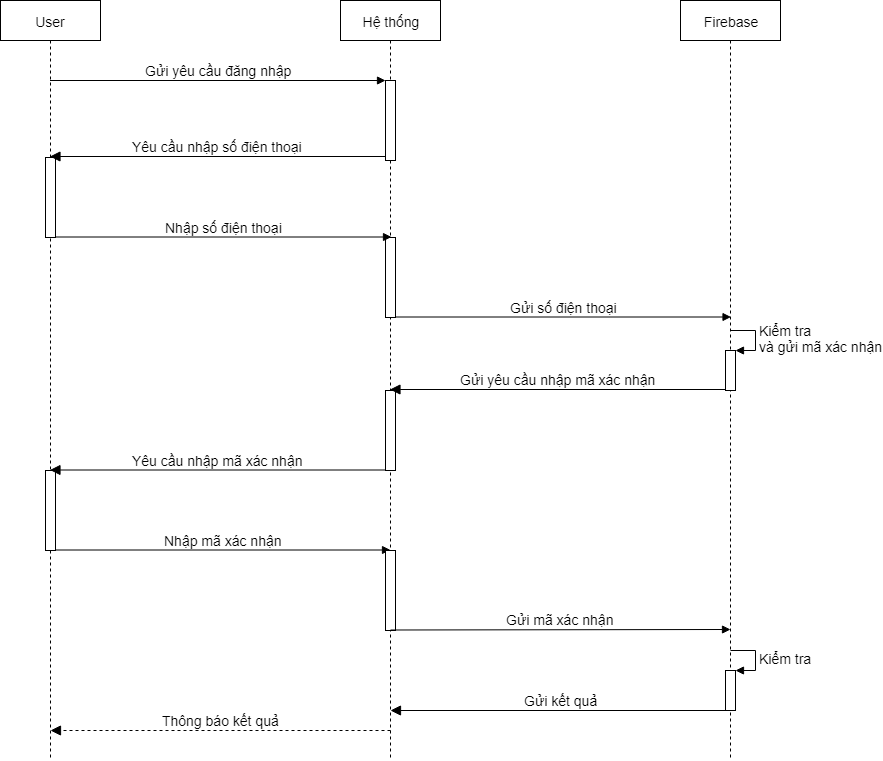
+ Dòng sự kiện chính

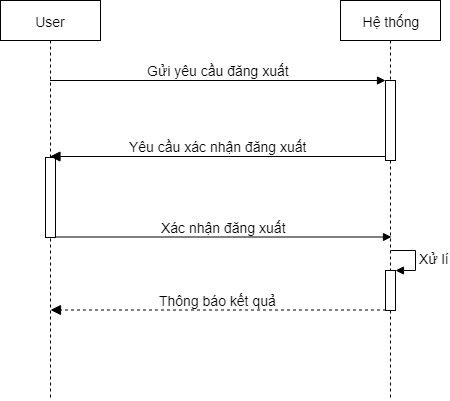
* Khi người dùng click vào tab “Cá nhân”, sau đó click vào “Yêu thích” trên giao diện Cá nhân, hiển thị giao diện Yêu thích gồm danh sách các phim mà người dùng đã lưu
* Click chọn phim muốn xem
* Trạng thái hệ thống khi bắt đầu thực hiện Use-case

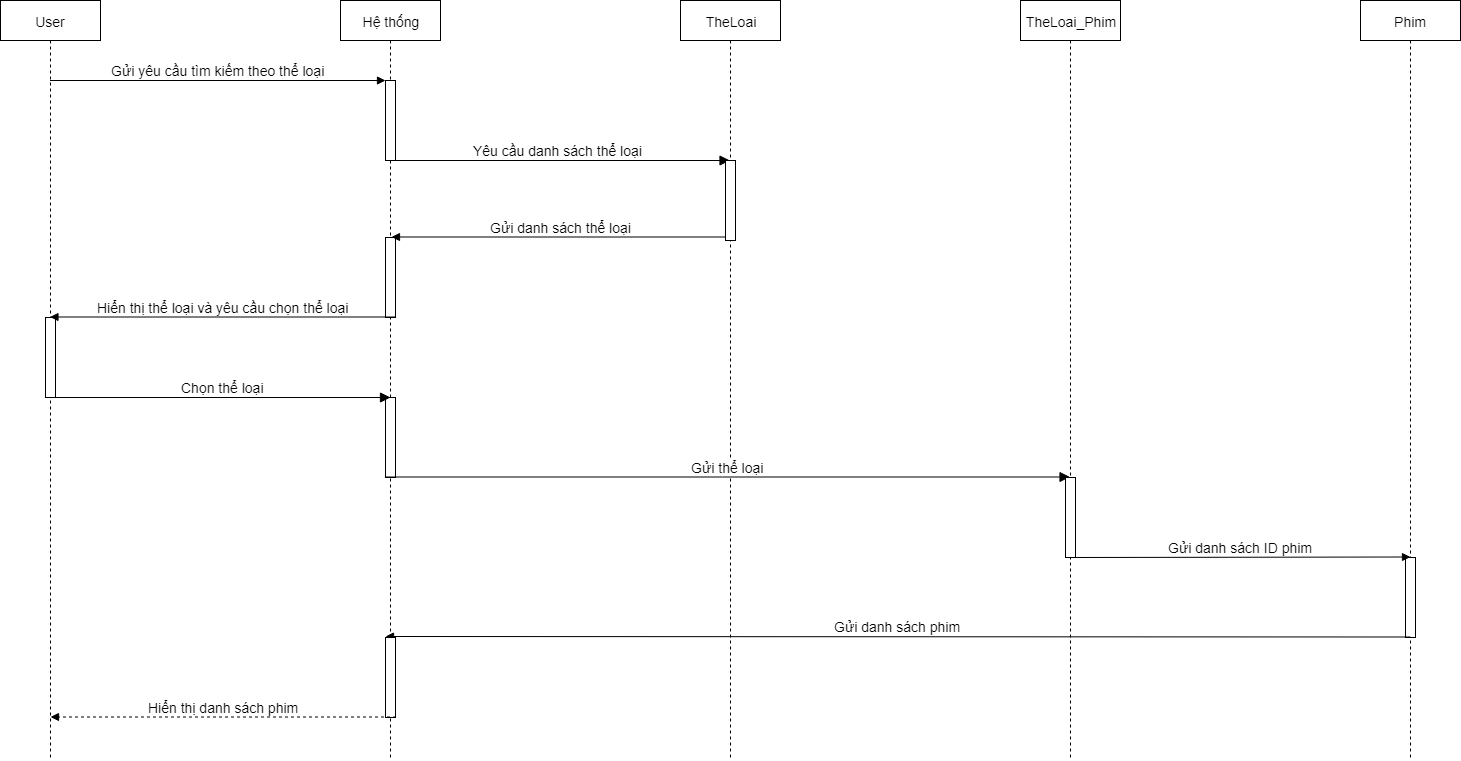
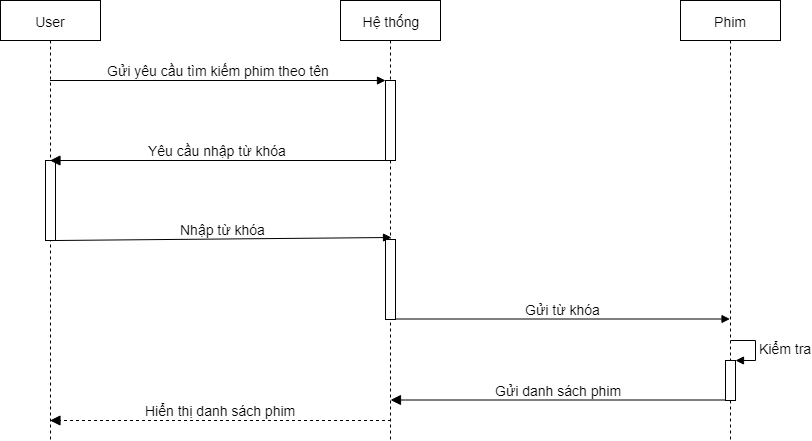
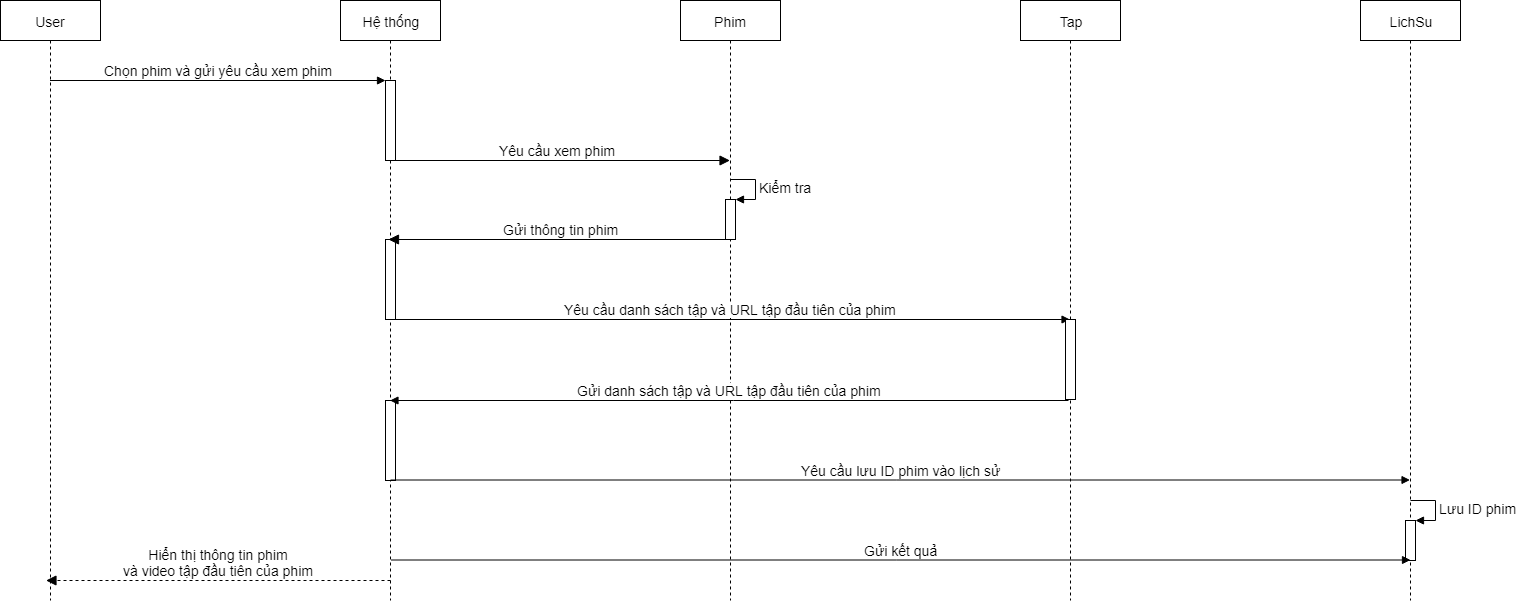
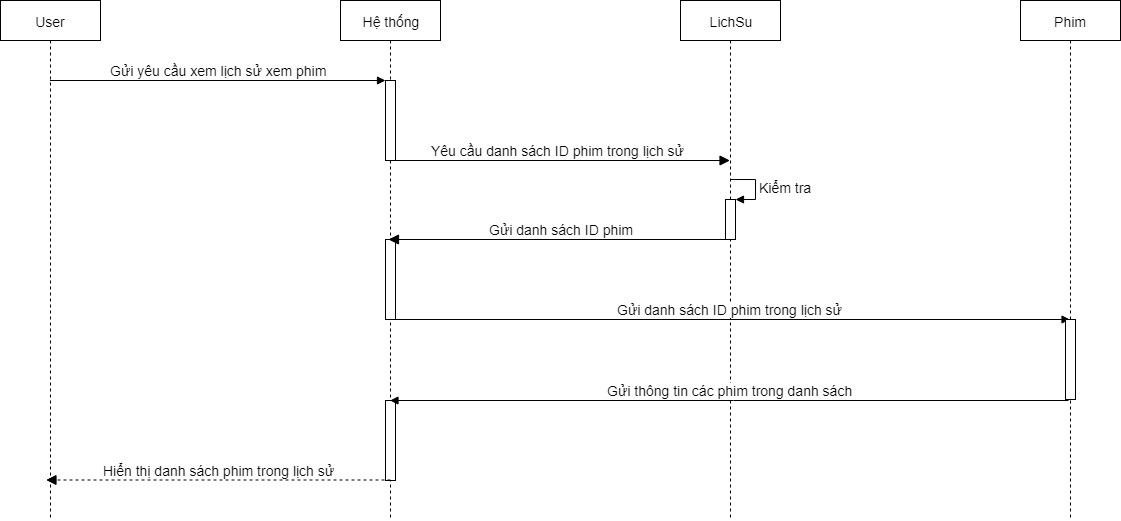
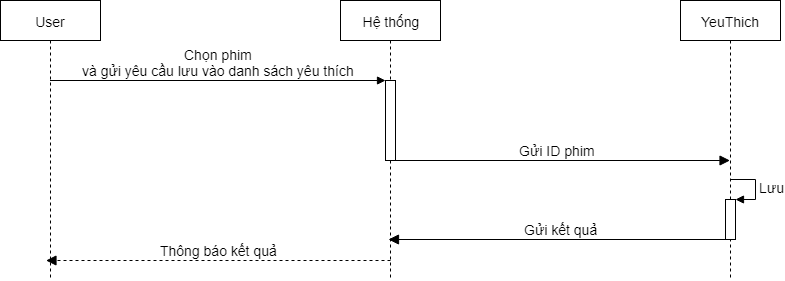
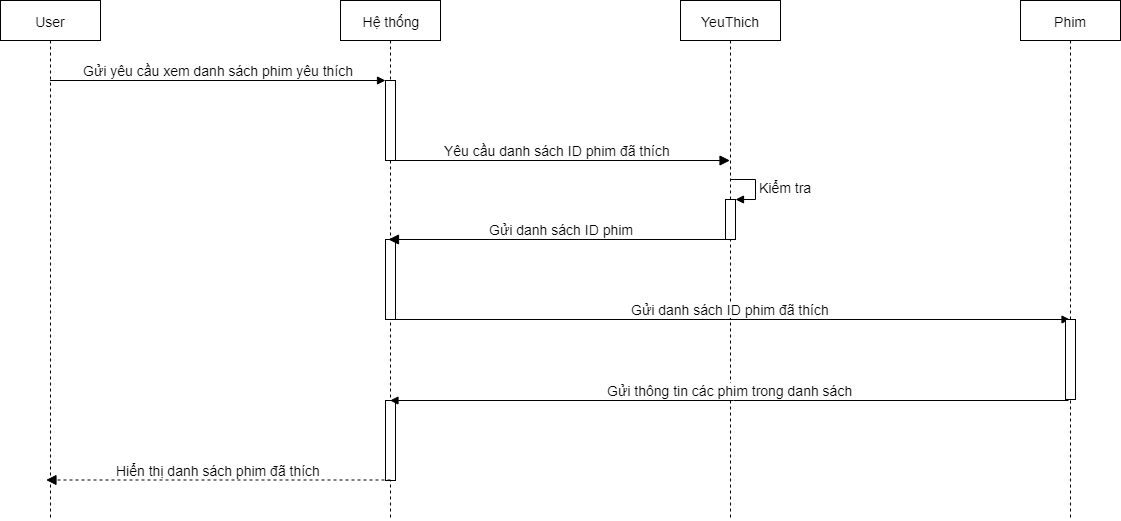
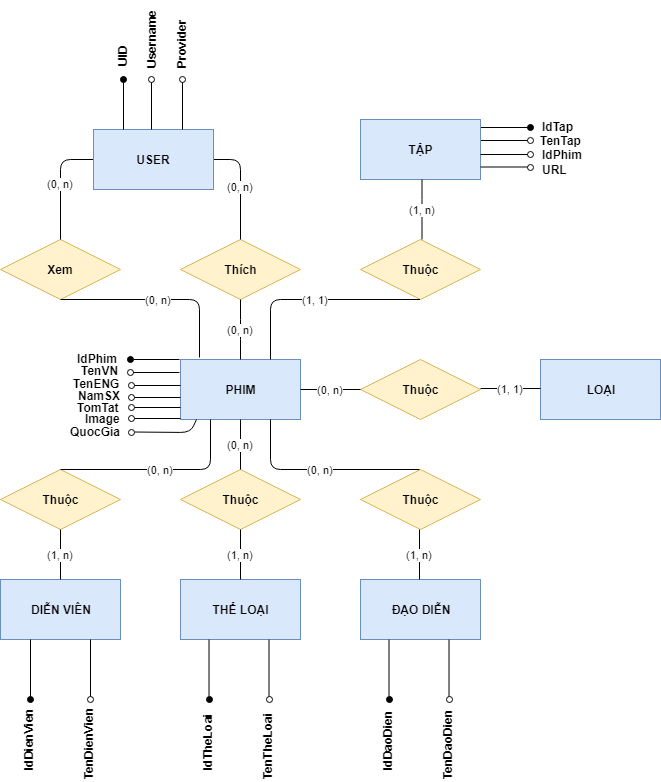
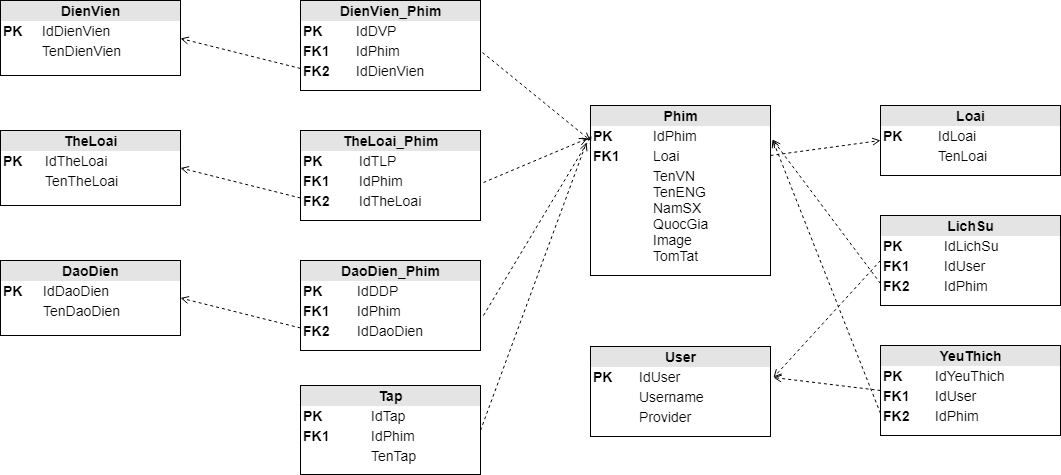
1. Người dùng đã đăng nhập vào tài khoản.
2. Người dùng đang ở giao diện Cá nhân.

* Trạng thái hệ thống sau khi thực hiện Use-case

1. Thực hiện thành công : chuyển tới màn hình Xem phim.
2. Thực hiện thất bại : Ở lại giao diện Cá nhân và xuất hiện thông báo.
3. **Sequence diagram**
   1. **Đăng nhập**
   2. Đăng nhập bằng số điện thoại



* 1. Đăng nhập bằng Facebook
  2. Đăng nhập bằng Google+
  3. **Đăng xuất**
  4. **Tìm kiếm phim**

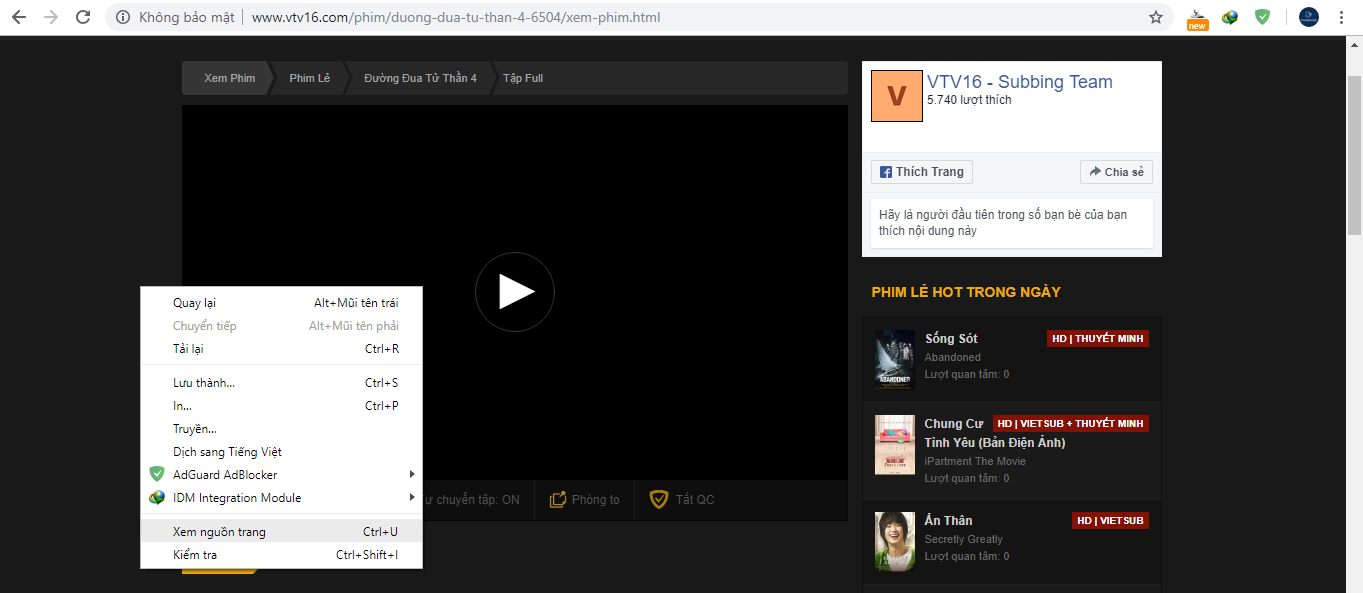
1. Tìm kiếm theo danh mục
2. **** Tìm kiếm theo tên
   1. **Xem phim**
   2. **Xem lịch sử phim đã xem**
   3. **Lưu phim yêu thích**
   4. Lưu phim yêu thích
   5. Xem danh sách phim yêu thích
3. **THIẾT KẾ DỮ LIỆU**
4. **Mô hình ERD**
5. **Class Diagram**
6. **GET LINK VIDEO PHIM**

* Các web phim có thể get link video: xemphimso, vtv16, ok.ru, rapidvideo, hdvip.tv
* Cách get: Thông qua website phụ sử dụng hàm curl($page) với ngôn ngữ PHP (Gọi hàm curl() 1 lần hoặc nhiều phụ thuộc vào độ phức tạp và tính bảo mật của website phim)
* Công dụng của hàm curl($page): tự động return source của $page dưới dạng string.
* Hướng dẫn lấy source page thủ công:

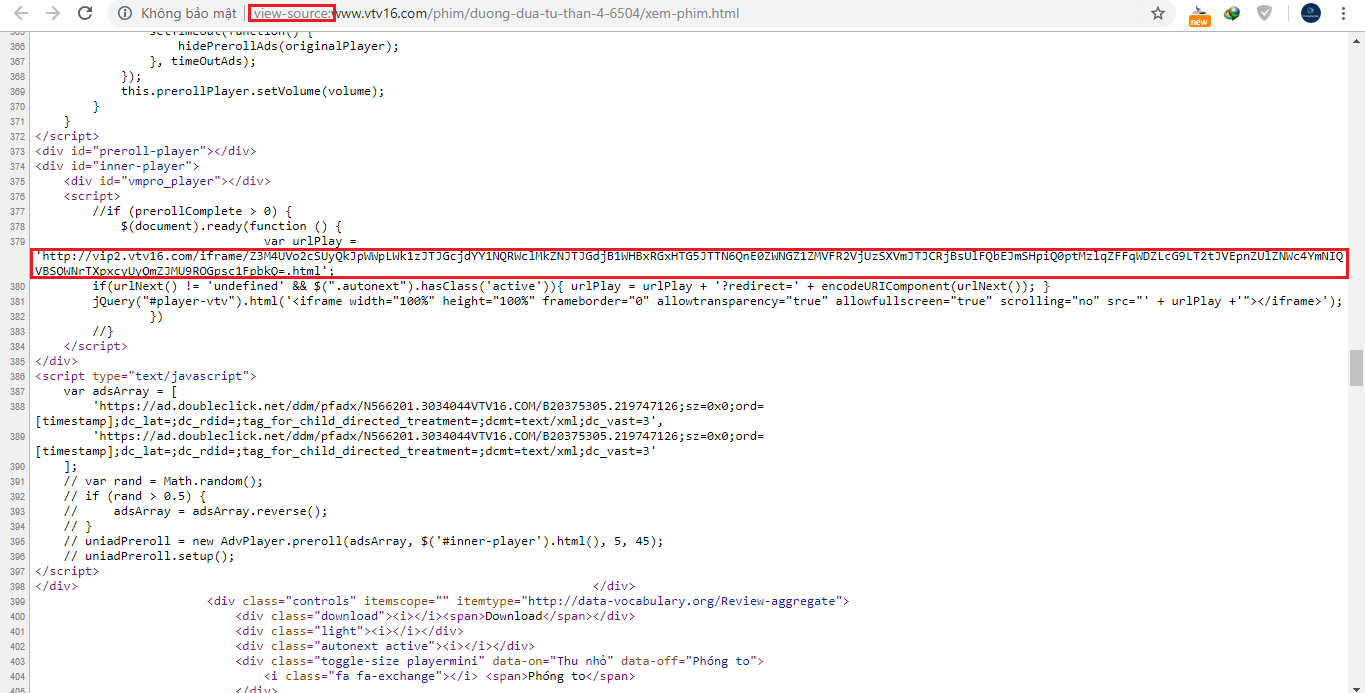
+ Bước 1: Truy cập page cần lấy source

+ Bước 2: Click chuột phải vào bất cứ đâu trên page, chọn “Xem nguồn trang”, hoặc thêm vào đầu URL “view-source:”

+ Bước 3: Xem kết quả

+ Ví dụ:

*Click chuột phải, chọn “Xem nguồn trang”*

*Trong một vài trường hợp, với các trang web bảo mật kém (trong hình là VTV16.com) thì có thể thấy ngay link video*

1. **CÁC COMPONENT ĐƯỢC SỬ DỤNG**

* Ngoài các component có sẵn thông dụng như View, ScrollView, TouchableOpacity, TextInput, Text, Image, Button thì nhóm em còn dùng một số component tự tạo như sau.

1. **Banner**

<TouchableOpacity style={styles.bannerContainer} onPress={() => this.props.navigation.navigate('Watching', {id: this.props.id})}>

<Image source={{uri: this.state.image}} style={styles.bannerStyle} />

</TouchableOpacity>

1. **Film & BigFilm**

<View style={styles.filmContainer}>

<TouchableOpacity onPress={() => this.props.navigation.navigate('Watching', {id: this.props.id})}>

<Image source={{ uri: this.state.image }} style={styles.posterStyle} />

<View style={styles.bottomContainer}>

<View style={styles.title}>

<Text numberOfLines={1} style={styles.enTitle}>{this.state.tenVn} ({this.state.namSX})</Text>

<Text numberOfLines={1} style={styles.vnTitle}>{this.state.tenEng} ({this.state.namSX})</Text>

</View>

<TouchableOpacity onPress={this.like} style={styles.likeButton}>

{

this.state.isLiked ? <Image source={srcLiked} style={styles.iconStyle} /> : <Image source={srcDisliked} style={styles.iconStyle} />

}

</TouchableOpacity>

</View>

</TouchableOpacity>

</View>

1. **ListFilms**

<ScrollView style={styles.slideFilms}>

<ListView

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <BigFilm id={rowData}/>}

/>

</ScrollView

1. **SlideFilms**

<View style={{marginTop: 10}}>

<View style={styles.slideHeader}>

<Text style={styles.category}>{this.state.title}</Text>

<TouchableOpacity onPress={() => this.props.navigation.navigate('TypeContent', {id: this.props.id})}>

<Text style={styles.seeMore}>Xem thêm</Text>

</TouchableOpacity>

</View>

<ScrollView horizontal={true} style={styles.slideFilms}>

<ListView

horizontal={true}

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <Film id={rowData} style={styles.filmItem}/>}

/>

</ScrollView>

</View>

1. **Episode**

<View style={styles.container}>

<TouchableOpacity onPress={ this.props.updateVideo } style={styles.episodeContainer}>

<Text style={styles.text}>{this.state.tenTap}</Text>

</TouchableOpacity>

</View>

1. **ListEpisodes**

<ScrollView style={styles.listEpisodes}>

<ListView

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <Episode id={rowData} style={styles.episode} />}

/>

</ScrollView>

1. **Actor & Director**

<View style={styles.personContainer}>

<TouchableOpacity>

<Image source={srcAvatar} style={styles.avatarStyle} />

<View style={styles.bottomContainer}>

<Text numberOfLines={1} style={styles.title}>{this.state.tenDV}</Text>

</View>

</TouchableOpacity>

</View>

1. **SlideActors**

<View>

<View style={styles.slideHeader}>

<Text style={styles.title}>Diễn viên</Text>

</View>

<ScrollView horizontal={true} style={styles.slideFilms}>

<ListView

horizontal={true}

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <Actor id={rowData} style={styles.personItem}/>}

/>

</ScrollView>

</View>

1. **SlideDirectors**

<View>

<View style={styles.slideHeader}>

<Text style={styles.title}>Đạo diễn</Text>

</View>

<ScrollView horizontal={true} style={styles.slideFilms}>

<ListView

horizontal={true}

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <Actor id={rowData} style={styles.personItem}/>}

/>

</ScrollView>

</View>

1. **SlideShow**

<View style={{marginTop: 10}}>

<ScrollView horizontal={true} style={styles.slideFilms}>

<ListView

horizontal={true}

autoplay={true}

dataSource={ds.cloneWithRows(arrId)}

renderRow={(rowData) => <Banner id={rowData}/>}

/>

</ScrollView>

</View>

1. **Player (dùng thư viện react-native-video-controls)**

import VideoPlayer from 'react-native-video-controls';

<VideoPlayer source={srcVideo}

navigator={ this.props.navigator }

style={styles.player}

/>

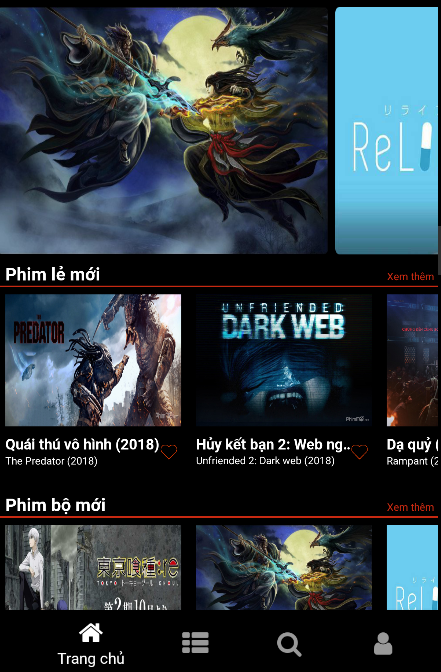
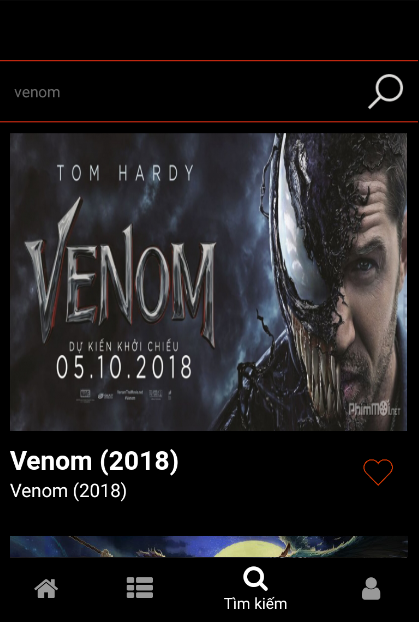
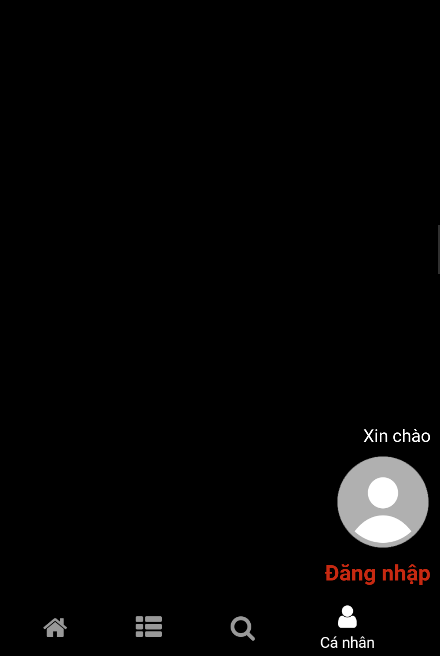
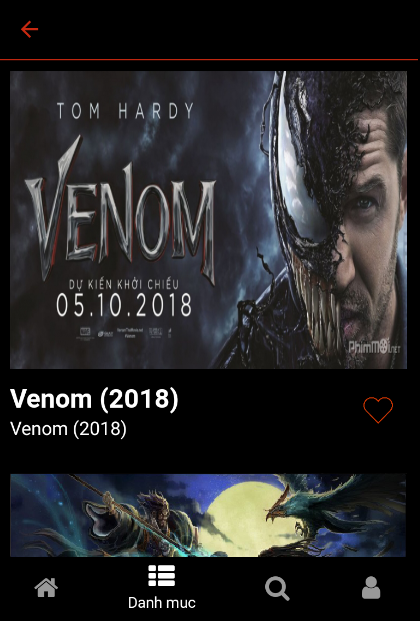
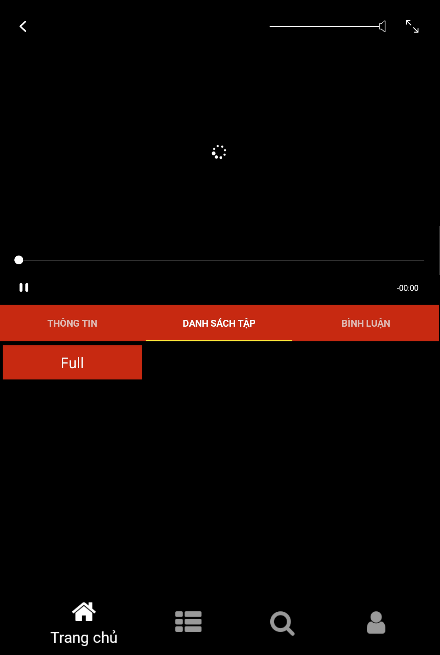
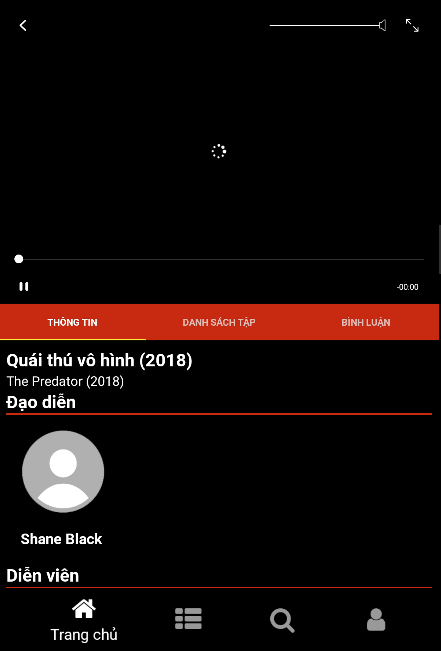
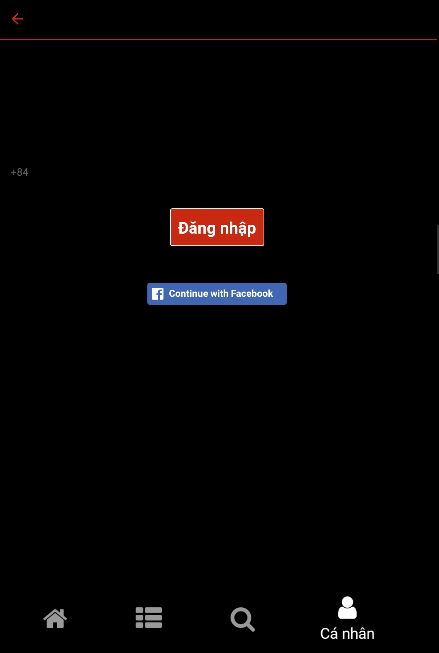
1. **NAVIGATION**

* Dùng thư viện react-native-navigation, react-navigation-material-bottom-tabs và react-navigation-tabs

+ createMaterialBottomTabNavigator ở màn hình chính

+ createStackNavigator cho tất các màn hình

+ createMaterialTopTabNavigator ở màn hình Xem phim

1. **GIAO DIỆN**
2. **Trang chủ (Home)**
3. **Danh mục (Thể loại)**
4. **Tìm kiếm**
5. **Cá nhân**
6. **Danh sách phim**
7. ******Xem phim**
8. **Đăng nhập**
9. **KẾT LUẬN VÀ TỰ ĐÁNH GIÁ**
10. **Kết luận**

* Đã nắm được kiến thức cơ bản về lập trình ứng dụng di động với ngôn ngữ React Native
* Biết cách sử dụng một số thư viện cần thiết hỗ trợ cho việc tập trình React Native
* Hiểu được khả năng và ứng dụng của Firebase

1. **Tự đánh giá**

* Hoàn thành được phần lớn các tình năng cần có của 1 app xem phim
* Tuy vậy, vẫn chưa hoàn thành được một số tính năng như dự kiến
* Định hướng trong tương lai sẽ hoàn thành nốt các tính năng còn thiếu và tối ưu hóa trải nghiệm người dùng

1. **TÀI LIỆU THAM KHẢO**

* Facebook’s Github React Native Docs: <http://facebook.github.io/react-native/docs>
* React Native Firebase Docs: <https://rnfirebase.io>
* List video khóa học React Native từ A đến Z của Trung tâm đào tạo tin học Khoa Phạm: <https://www.youtube.com/watch?v=AOowjW6_kco&list=PLzrVYRai0riQsPYaBX-aNz8YCmtDsg17A>
* List video sử dụng Firebase cho React Native của Trung tâm đào tạo tin học Khoa Phạm: <https://www.youtube.com/watch?v=cm7edAX7jHk&list=PLzrVYRai0riT4WOQXE-B2IDEkeugarXqM>