**Tên Nhóm: Nhóm 7**

**Thành Viên:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ Và Tên | Email | SDT |
| 1 | Nguyễn Đăng Khoa | jeremi220497@gmail.com | 0338001520 |
| 2 | Nguyễn Đức Lai | nguyenduclai1997@gmail.com | 0388102525 |
| 3 | Phan Văn Khải | khajphan8@gmail.com | 0326027536 |
| 4 | Bùi Đức Thảo | thaobui290896@gmail.com | 0798775789 |

Email nhóm: didong3nhom7@gmail.com

Pass: Didong123465

---------------------------------------------------------------------------------

**1 Tìm hiểu về list view (Đăng Khoa)**

ListView là một thành phần cốt lõi được thiết kế để hiện thị hiệu quả đối với dữ liệu cần hiển thị dạng danh sách các dữ liệu sẽ được hiển thị theo chiều dọc.

Có hai thuộc tính quan trong list view đó là dataSource và renderRow. Datasource là dữ liệu sẽ được hiển thị lên. RenderRow sẽ trả về giao diện giữa các list item.

VD

class **MyComponent** extends **Component** {

constructor() {

super();

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

this.state = {

*// Tạo một đối tượng data source với dữ liệu là một mảng*

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

};

}

**render**() {

return (

<**ListView**

*// Set nguồn dữ liệu của list view*

*dataSource*={this.state.dataSource}

*// Lấy từng dữ liệu từ mảng hiển thị lên mỗi dòng sẽ là một thẻ text*

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

/>

);

}

}

**Các props (Các thuộc tính trong thẻ list view)**

- dataSource

Đây là thuộc tính bắt buộc với giá trị là ListViewDataSource

var ds = new ListView.DataSource({rowHasChanged: this.\_rowHasChanged});

- initialListSize (Số)

Quy định số hàng hiển thị trên màn hình lúc đầu.

- onEndReachedThreshold (Số)

Quy định số pixel hiển thị trên thiết bị vật lý

- pageSize (Số)

Số dòng sẽ được render cho mỗi lần loop (Mỗi lần kéo). Đây là thuộc tính yêu cầu

- renderRow (Function)

(rowData, sectionID, rowID, highlightRow) => renderable

Lấy một mục dữ liệu từ nguồn và các id của nó và sẽ trả về một thành phần có thể hiển thị dưới dạng hàng. Theo mặc định dữ liệu chính là những gì được đưa vào nguồn dữ liệu. List view có thể đánh dấu hàng được chọn bằng highlighRow(sectionID, rowID).

Những thiết lập giá trị boolean này của adjacentRowHighlighted in renderSeparator cho phép bạn điều khiển các dòng riêng biệt trên và dưới highlightedrow.

- renderScrollComponent (function)

(props)=> renderable

Một phương thức trả về thành phần được cuộn trong danh sách các hàng đã được render. Mặc định để để trả về một ScrollView với giá trị được gắn vô props.

- scrollRenderAheadDistance (Số)

Hiển thị sớm các hàng trên màn hình trước khi nó sẵn sàng hiện dữ liệu

- stickyHeaderIndices

**Các methods**

- getMetrics()

Mở rộng một vài dữ liệu.

- scrollTo()

scrollTo(...args: Array)

Cuộn tới x, y với hiệu ứng

- scrollToEnd()

scrollToEnd(([options]: object));

- flashScrollIndicators()

flashScrollIndicators();

**2 Tìm hiểu về navigation (Đăng Khoa , Văn Khải)**

Naviagation đây là thành phần giúp chúng ta chuyển đổi giữa các màn hình cũng là thành phần giúp chúng ta chuyển đổi dữ liệu giữa các màn hình với nhau.

**Để sử dụng navigation chúng ta phải trải qua các bước sau:**

- Bước 1: Cài gói navigation vào project thông qua lệnh cmd trỏ vào project

npm install --save react-navigation

npm install --save react-native-gesture-handler

react-native link react-native-gesture-handler

Bước 2: Tiến hành tạo stack navigator có bao nhiêu màn hình thì chúng ta đều khai báo tại đây

VD

import React from 'react';

import { View, Text, Button } from 'react-native';

import { createAppContainer, createStackNavigator, StackActions, NavigationActions } from 'react-navigation'; *// Version can be specified in package.json*

class **HomeScreen** extends **React**.**Component** {

**render**() {

return (

<**View** *style*={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<**Text**>Home Screen</**Text**>

<**Button**

*title*="Go to Details"

*onPress*={() => {

this.props.navigation.**dispatch**(StackActions.**reset**({

index: 0,

actions: [

NavigationActions.**navigate**({ routeName: 'Details' })

],

}))

}}

/>

</**View**>

);

}

}

class **DetailsScreen** extends **React**.**Component** {

**render**() {

return (

<**View** *style*={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<**Text**>Details Screen</**Text**>

</**View**>

);

}

}

const AppNavigator = **createStackNavigator**({

Home: {

screen: HomeScreen,

},

Details: {

screen: DetailsScreen,

},

}, {

initialRouteName: 'Home',

});

<https://reactnavigation.org/docs/en/hello-react-navigation.html>

Bước 3: Sau khi khai báo đây là cách di chuyển giữa các màn hình

- Để di chuyển sang màn hình khác tại sự kiện của component nào đó ta gọi cú pháp như sau:

this.props.navigation.**navigate**('TenManHinhDaDinhNghia')

// Nếu muốn load lại màn hình sau khi đã chuyển đến màn hình đó rồi thì theo cú pháp sau:

his.props.navigation.**push**('ManHinhMuonKhoiTaoLai')

// Trường hợp muốn back lại trường hợp trước đó

this.props.navigation.**goBack**()

// Trường hợp muốn chuyển dữ liệu giữa các màn hình

this.props.navigation.**navigate**('TenManHinhDaDinhNghia', {key: value})

Đây chỉ là một phần về navigation để nắm rõ kiến thức này cần tham khảo nhiều tài liệu bên ngoài