1. Khi chạy demo trên máy ảo, không cần thiết phải mở Android Studio.

2. method filter dùng cho search bar.

3. Đối với action bất đồng bộ, ta không thể return dispatch bình thường được (khi dùng middleware thunk). Ta có thể, ví dụ, viết return như code phía dưới. Để ý là return để đầu tiên, sau đó function dispatch() để cuối cùng.

export function updateList() {

return (dispatch) => {

let animals = [];

let listItem = [];

animalRef.once('value', (childSnapshot)=>{

childSnapshot.forEach((doc)=>{

animals.push({

key: doc.key,

animalName: doc.toJSON().animalName

})

})

}).then(()=>{

listItem = JSON.parse(JSON.stringify(animals));

console.log("error");

console.log(listItem);

dispatch({

type: ActionTypes.GET\_FIREBASE\_DATABASE\_LIST,

listItem

});

})

}

}

Ta không thể dùng .then sau .once() được.

animalRef.once(…….).then(() => {return(dispatch) =>{

….

dispatch({….})

}})

Vì nó sẽ báo lỗi action should be plain object …. gì gì đó.

4. Nên lưu ý đến thời diểm render data. Với ví dụ bên dưới, rõ ràng là chương trình chạy render() trước khi listItem, khi này sẽ báo lỗi là listItem = null.

VD:

componentDidMount() {

this.props.updateList(); // nơi cập nhật this.props.listItem

}

class CompA extends Components () {

….

render() {

<View>

<FlatList

data = {this.props.listItem}

……

</View>

}

}

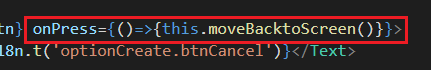
Để khắc phục, ta nên cho điều kiện xét xem ***this.props.listItem != undefined hay không***.

5. Khi chạy mà bị mấy cái error liên quan đến folder build của debug, thay vì phải xóa tay, ta dùng lênh sau:

react-native start --reset-cache

Sau đó chạy lại react-native run-android.

6. Lưu ý khi khai báo hàm thực hiện trong onPress

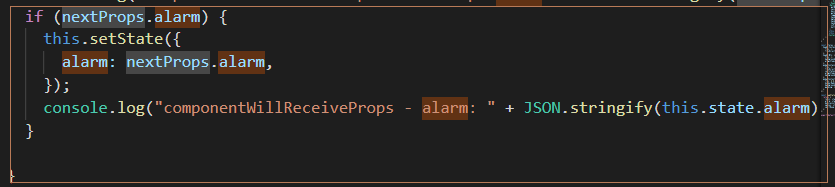


Không nên khai báo như bên dưới:

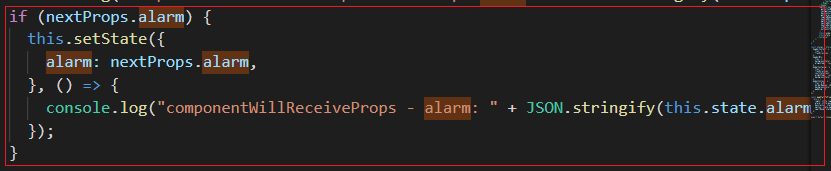
onPress = {this.moveBacktoScreen} vì khi nhảy vào screen chưa button này, nó sẽ thực hiện luôn hàm moveBacktoScreen().

7. this.setState là **async function.**

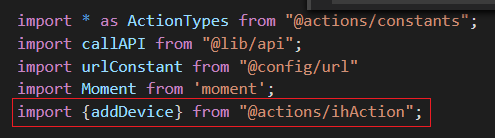
**Giả sử ta đang trong componentWillReceiveProps(), như code bên dưới, kết quả in ra của console.log() sẽ chưa được cập nhật bởi nextProps.alarm.**

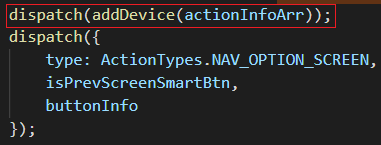


Để khắc phục, ta sẽ đưa console.log() vào tham số thứ 2 của method this.setState(). Tham số thứ 2 là callback function.



8. Sử dung action của 1 component khác. Ta phải **import** action đó, sau đó **dispatch** cái hàm ta muốn dùng.





**9. (Redux)this.props.[tên reducer].[tên prop] = [value]**

Khi ta gán giá trị cho props của 1 component nào đó bằng cách trên, nhưng component khi sử dụng prop trên cũng sẽ bi ảnh hưởng.

VD:

**Tại CompB:**

this.props.compA.propA = 2;

. . .

function mapPropToState() {

compA: state.compReducer

}

**Tại CompC:**

console.log(this.props.compA.propA); // print 2

. . .

function mapPropToState() {

compA: state.compReducer

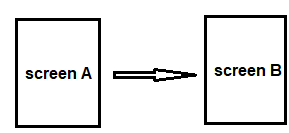
}

10. Moment

Moment.utc(time).format : Chỉnh lại thời gian theo múi giở local

**11. Navigation**

11.1



Khi từ screenA đến screenB, lifeCycle của screen B là:

constructor -> WillMount -> render -> DidMount

**Khi quay lại screenA, component của screenB sẽ bị UnMount**.

11.2 Cũng từ screenA tới screenB, nhưng trước khi nhảy vào screenB, ta thay đổi state redux của component screenB. Lifecycle khi vào screenB:

constructor -> WillMount -> render -> DidMount

Ta thấy cho dù state của compB có thay đổi nhưng cũng **không vào ComponentWillReceiveProps.**

11.3 Cũng từ screenA tới screenB, khi ta dùng lệnh:

this.props.navigation.navigate( ‘ScreenB’ );

thì cũng **bằng với this.props.navigation.goBack();**

Và lưu ý là với goBack() thì screenA sẽ **ko render lại.**

**11.4 Modal**

Khi muốn tạo 1 modal có background màn hình sau mờ, ta dùng format code như bên dưới.

<Modal

transparent={true}

visible={this.state.modalVisible}

onRequestClose={()=>{this.setState({modalVisible: !this.state.modalVisible})}}

>

<View style={styles.modalContainer} >

<View style={styles.modalContent}>

<TouchableOpacity onPress={()=>{this.handleAddMenu()}}>

<Text>Add</Text>

</TouchableOpacity>

<TouchableOpacity onPress={()=>{this.handleDeleteMenu()}}>

<Text>Delete</Text>

</TouchableOpacity>

<TouchableOpacity onPress={()=>{this.handleModifyMenu()}}>

<Text>Modify</Text>

</TouchableOpacity>

</View>

</View >

</Modal>

const styles = StyleSheet.create({

modalContainer: {

flex: 1,

flexDirection: "row",

justifyContent: "center",

alignItems: "flex-end",

backgroundColor: "rgba(0, 0, 0, 0.2)"

},

modalContent: {

flex: 1,

backgroundColor: "white",

height: 100,

flexDirection: 'column',

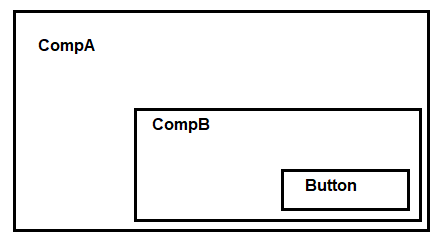
justifyContent: 'space-around',

paddingLeft: 15

},

});

**12.**

****

**Tại CompA:**

handleAction = ((var1, var2) => {

// process code with var1, var2

}

render(){

return(

<CompB

handleAction = {(var1, var2) => {this.handleAction(var1, var2)}}

/>

)

}

**Tại CompB:**

render(){

return(

<View>

<Button onPress={()=>{this.props.handleAction(var1,var2)}}

</View>

)

}

Ta thấy, từ bên trong compB, khi nhấn button, hàm sẽ truyền thẳng lên handleAction() của compA. Phía trên là cách thực hiện event từ 1 component bên trong 1 component khác.

13. Sử dụng điểu kiện trong return()

VD:

render() {

return(

**{**

this.state.isShowing **?**

<View> <Text>aaaaaaaa</Text> </View>

**:**

<View><Text>bbbbbbbbb</Text> </View>

**}**

)

}

hoặc ta cũng có thể dùng hàm:

**\_renderView** = () => {

if(this.state.isShowing){

return(

<View> <Text>aaaaaaaa</Text> </View>

)

}

else {

return(

<View><Text>bbbbbbbbb</Text> </View>

)

}

}

render() {

return(

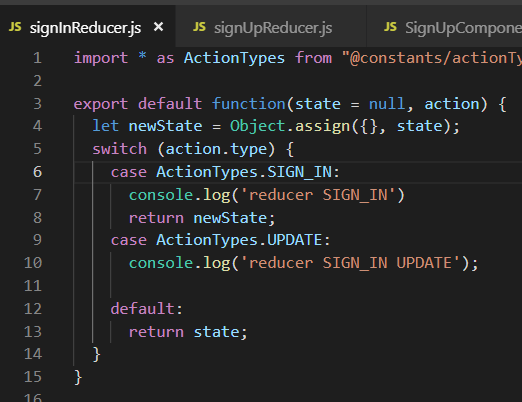
**{this.\_renderView()}**

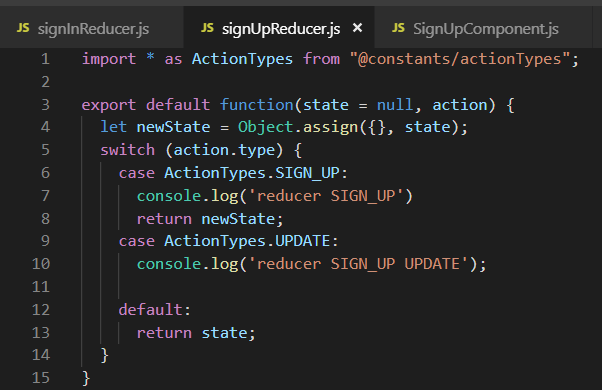
)

}

**14.** Nhiều reducer có thể cùng thực hiện khi dispatch 1 action type giống nhau.

VD:





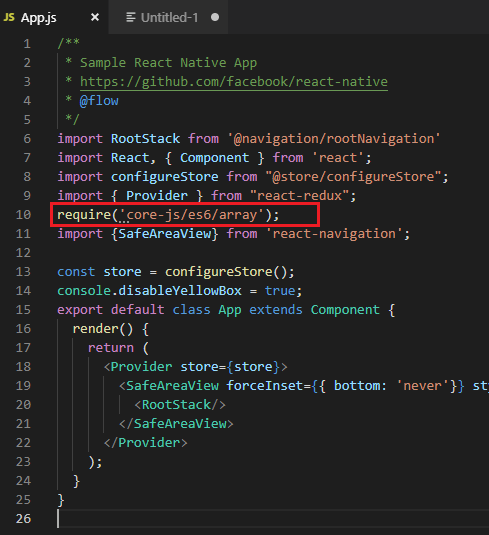
Ta thấy cả 2 reducer trên có cùng chung actionType là UPDATE. Giả sử ta đang ở component SignIn, ta gọi dispatch action type là UPDATE, nó sẽ ra kết quả sau:



Kết quả là nó cũng in ra luôn dòng của thằng Sign Up.

**15.** Nên cover trường hợp Back với **phím cứng** trên android.

TH dùng BackHandler trong release mode mà không có tác dụng, thêm dòng này vào file App.js.



**16.** Cẩn thận với trường hợp gán mảng với mảng, vì sẽ liên quan đến trường hợp **Shadow copy.**

- Nếu chỉ với mảng generic (string, int, float,…) ta có thể dùng hàm **.slice()** để clone mảng

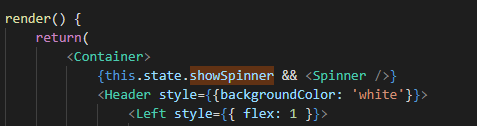
- Nhưng với mảng chứa object thì ta nên dùng **JSON.parse(JSON.stringtify(array))**

**17.** Các component có thể dùng chung typeAction trong redux.

GIả sử ta có 2 comp A và B. Cả 2 comp đều có type action là isLoading.

Trong action của compA khi dispatch với type IS\_LOADING, reducer của compB cũng sẽ dc gọi.

**18.** Trong trường hợp show Spinner, comp Spinner **nên để đầu dòng** trong return của hàm render.



**19. FlatList**

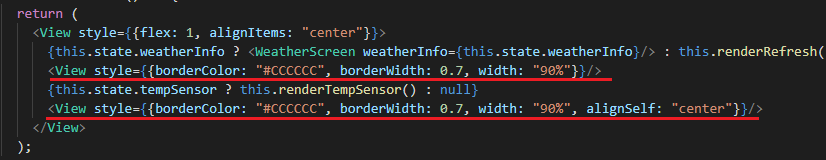
Muốn render lại Flatlist, dùng extraData.

Có 2 loại:

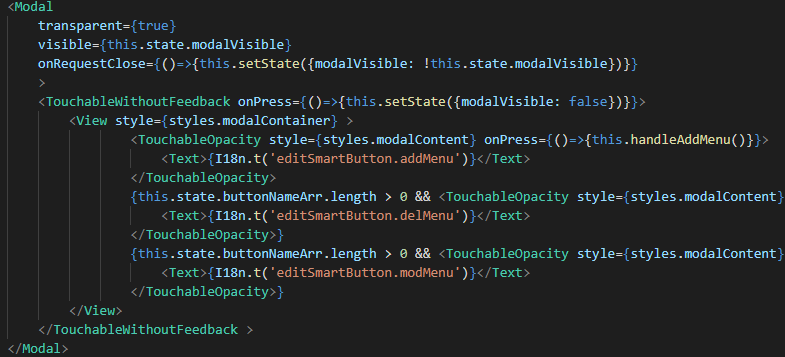
+ extraData: {this.state}

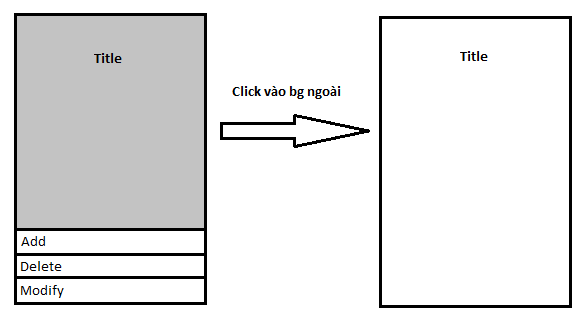
+ extraData: {this.props}

**20**. **Tạo border đường ngang,** ta dùng <View> và kết hợp borderWidth và borderColor.



**21. Tạo modal với background ngoài được làm mờ, và có thể thoát ra khi click vào nó.**

****

****

**22. Array không có phần tử không phải null.**

Luu ý 2 câu lệnh so sánh sau:

+ if(array) => check array có null không

+ if(array.length > 0) => check array có phần tử không

**23. Cách dùng style với styleSheet kết hợp với style riêng.**

<View style={ [ style.title, {color: ‘red’} ] } >

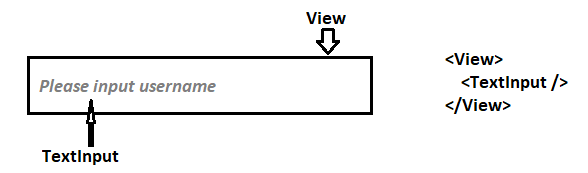
**24. Các hotkey với iOS:**

Refresh: command + R

Dev Setting menu: command + D

**25.** ListItem có thể gắn onPress.

**26.**

****

+ Để TextInput ở giữa, dùng **alignItems** với View.

+ Để chỉnh màu cho placeholder, dùng **placeHolderTextColor**. (có thể dùng màu rgba)

+ Để bỏ underline của TextInput, chỉnh màu transparent trong prop **underlineColorAndroid**.

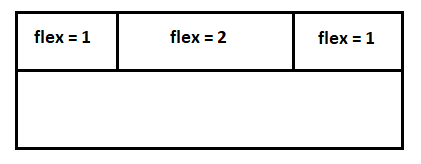
+ Không nên dùng <TextInput>{this.state.text}<TextInput> mà thay vào đó là:

<TextInput value: this.state.text />

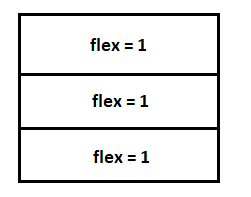
**27.**

Chia flex cho view con.

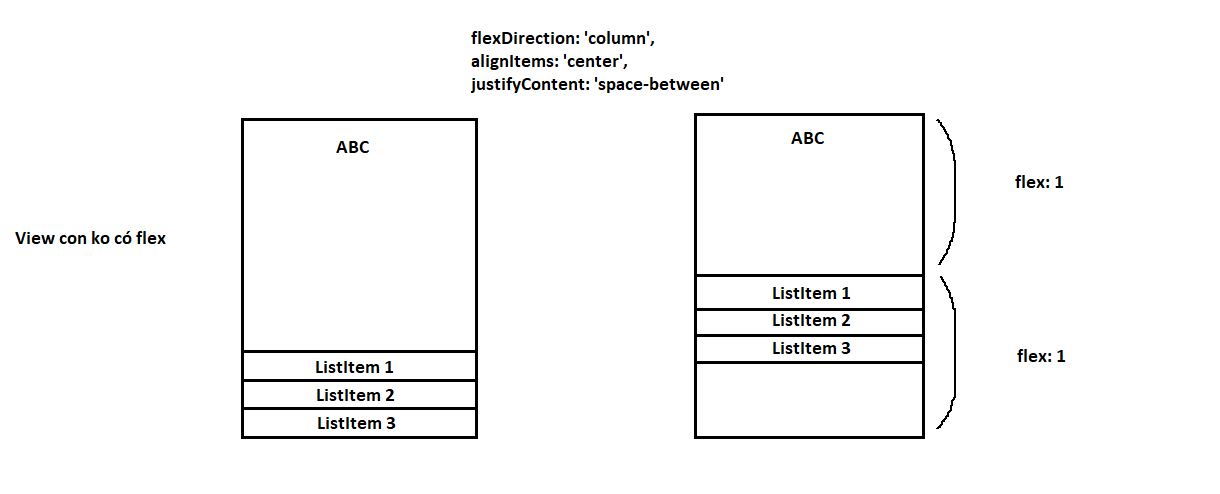
Với flexDirection = ‘row’:



Với flexDirection = ‘column’:



**28.**

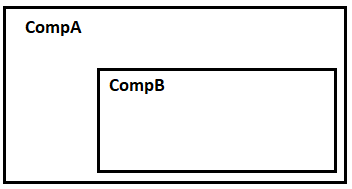


Xem trường hợp trên, có những trường hợp ko nhất thiết phải dùng flex cho view con.

Ở hình 2, ta thấy nếu gán flex cho view con thì List sẽ không nằm dưới.

**29**. Để code chạy đúng nhất, đôi khi ta nên Reload lại app. Vì khi reload thì các redux state sẽ reset lại như ban đầu.

**30.**

****

Giả sử ta chỉ thay đổi state trong compB, chỉ có compB render lại, ko render compA.

**31. setNativeProps kết hợp ref.**

Trong VD dưới, khi nhấn vào button, thì button đó sẽ thay đổi background.

****

****

**32. Chỉnh <Text> vào vị trí vertical center => Dùng <View>**

<View style={{position: ‘absolute’, top: 0, botto: 0}}>

<Text>abc<Text>

</View>

**33. <Content>**

+ Có chức năng scroll

+ <List> phải đi với <Content>, nếu không sẽ ko chỉnh được height cho ListItem.

**34. <Seperator> kết hợp với <List>**

=> ListItem nên để **borderBottom** thay vì border.

**35. Nên để scrollView ở mọi màn hình.**

**36. Vysor: phần mềm đưa màn hình điện thoại lên laptop**

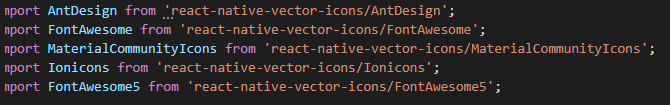
**37. ip4.me: web xem địa chỉ ip public của máy.**

**38. Cách dùng vector icon**

**B1: Check thư viện icon trong trang này:**

[**https://oblador.github.io/react-native-vector-icons/**](https://oblador.github.io/react-native-vector-icons/)

**B2: Import thư viện**

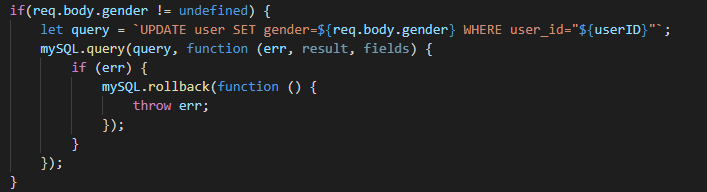
****

**B3: Dùng tên tìm trong B1 đưa vào code:**

****

**DATABASE:**

**1. Trong câu query, với các biến chứa true, false, không được dùng dấu “ “.**

****

**req.body.gender là biến chứa true hoặc false, nên không có dấu “ “ như userID.**

**2. Kiểu dữ liệu tinyInt trong database, với true sẽ là 1, false là 0.**

**Node JS Server (server.js)**

1. Dùng firebase-admin cho server thay vì dùng firebase cho app.

2. Nên install body-parser, vì đây là package dành cho body của request từ client gửi tới server. Nếu không cài package này, ở server, req.body sẽ là **undefined**.

<https://stackoverflow.com/questions/9177049/express-js-req-body-undefined>

3. Khi sửa source Server, nên reset lại (tại cmd window -> Ctrl C -> node server.js)

4. Cách để tạo user bằng Authentical: Có 4 cách.

<https://firebase.google.com/docs/auth/admin/>

5. Khi dùng console.log() để debug, nó sẽ hiện nội dung trên **cmd prompt**.

**adb**

**Sau đây sẽ trình bày các bước thực hiện chay react native trên real device**

**I. Thiết lập adb**

B1: Mở chế độ “Cài đặt cho người phát triển” trong Settings => Mở “Chế độ USB debug”

B2: Kết nối smartphone vs laptop thông qua công USB

B3: Mở cmd prompt => gõ lệnh adb devices. Nếu xuất hiện thiết bị có chữ device bên phải là ok.

B4: Chay lệnh react-native run-android

**II. Kết nối development server**

B1: Làm theo link dưới:

<https://facebook.github.io/react-native/docs/running-on-device.html#method-2-connect-via-wi-fi>

B2: Để kiểm tra có reload được không, gõ lệnh sau:

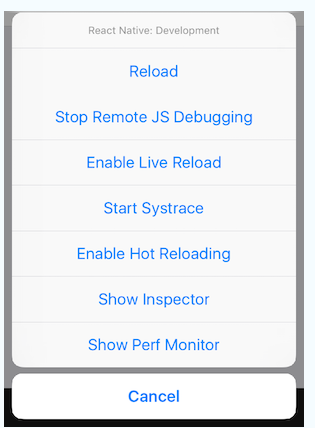
adb shell input text "RR"

**III. Mở chế độ Debug JS**

B1: Mở cmd prompt khác và gõ lệnh sau để mở Developer menu:

adb shell input keyevent 82

B2: Chọn Remote JS Debugging.



**Lưu ý:**

1. Nếu có warning: Debugger and device times had drifted by more than 60s. Please correct this by running adb shell "date `date +%m%d%H%M%Y.%S`" on your debugger machine

thì chình lại thời gian của điện thoại giống 100% với laptop.

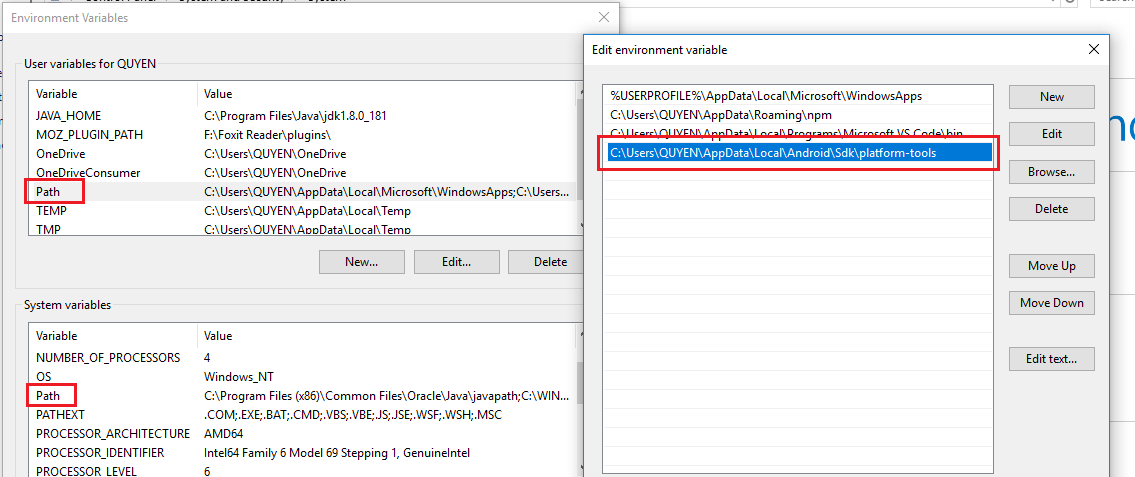
2. Nếu bị Bachedbridge gì đó, dùng lệnh dưới:

react-native start --reset-cache

react-native run-android

sau đó vào Dev menu -> xóa bỏ IP laptop và chạy lại lệnh react-native run-android

3. Trong TH gõ lệnh adb devices nhưng hệ thống không nhận dạng được lệnh thì ta phải thiết lập Environment Variable.



**Một số lệnh:**

adb shell input text "RR" : reload lại app nếu source thay đổi.

adb shell input keyevent 82 : Mở Develop menu

**ERROR**

**1. Attempted to transition from state `RESPONDER\_INACTIVE\_PRESS\_IN` to `RESPONDER\_ACTIVE\_LONG\_PRESS\_IN`, which is not supported**.

=> Chỉnh lại timezone và thời gian cho điện thoại giống với PC

**2. Lỗi google-service.json khi build react native**

=> Mở Task Manager và end task Java

**3. (Remocon) Lỗi khi không load được button và weather**

=> Kiểm tra lai url kết nối với server

# 4. [Unable to connect with remote debugger](https://stackoverflow.com/questions/40898934/unable-to-connect-with-remote-debugger)

# =>

* Press Cmd + M on emulator screen
* Go to Dev settings > Debug server host & port for device
* Set localhost:8081
* Rerun the android app: react-native run-android

5. Khi dùng lệnh npm install mà bị lỗi node gyp thì nên xem lại version của node, ko nên cài bản mới nhất mà nên cài recommended version

**6. Lỗi config.h not found trong thư viện third\_party.**

**=>**

* In the Terminal, navigate to the react-native/third-party/glog folder inside node\_modules (for me, this was cd node\_modules/react-native/third-party/glog-0.3.4)
* Once actively in this folder, run ../../scripts/ios-configure-glog.sh
* Glog is configured and the required config.h header file is created for Xcode to find