TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN

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

BÁO CÁO KẾT THÚC MÔN HỌC

Lập trình di động 3

**QUẢN LÝ SINH VIÊN**

Giảng viên hướng dẫn: Trương Bá Thái

Sinh viên thực hiện:

1. Trần Cao Vinh
2. Phạm Minh Hiển
3. Nguyễn Hữu Phú

Ngành: Công nghệ thông tin

Lớp: Lập trình di động 3

Tp. Hồ Chí Minh, ngày 10 tháng 12 năm 2018

**NHẬT KÝ HOẠT ĐỘNG NHÓM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stt** | **Họ và tên** | **Công việc đã thực hiện** | **Tự đánh giá** | **Nhóm đánh giá** | **Chữ ký** |
| 1 | Trần Cao Vinh | * Cài đặt và chạy demo react native. Tìm hiểu var , let. Vẽ màn hình cho project. * Tìm hiểu biến mảng trong ES6 * Tìm hiểu height and width, flex * Demo project. | 9 | 10 |  |
| 2 | Nguyễn Hữu Phú | * Cài đặt và chạy demo react native . Tìm hiểu class Vẽ màn hình cho project. * Tìm hiểu set trong ES6 * Tìm hiểu styles, flexbox * Demo project. Đăng nhập, đăng ký. | 9 | 10 |  |
| 3 | Phạm Minh Hiển | * Cài đặt và chạy demo react native * Tìm hiểu object. Vẽ màn hình cho project Chạy demo react native * Tìm hiểu map trong ES6 * Tìm hiểu state, props * Tim hiểu ListView * Demo project. Thêm sinh viên, thêm giảng viên * Tìm hiểu Service | 9 | 10 |  |

MỤC LỤC

DANH MỤC BẢNG BIỂU, HÌNH VẼ, SƠ ĐỒ 4

CHƯƠNG 1. TỔNG QUAN VỀ REACT NATIVE 5

1. Giới thiệu tổng quan về React native 5

2. Kiến thức cơ bản về ES6 trong React native 5

3. Component trong react native 5

4. API Trong React Native 5

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 6

2.1 Phân tích hệ thống 6

2.2 Thiết kế hệ thống 8

3.1 Cài đặt 10

3.2 Kiểm thử 10

CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC 11

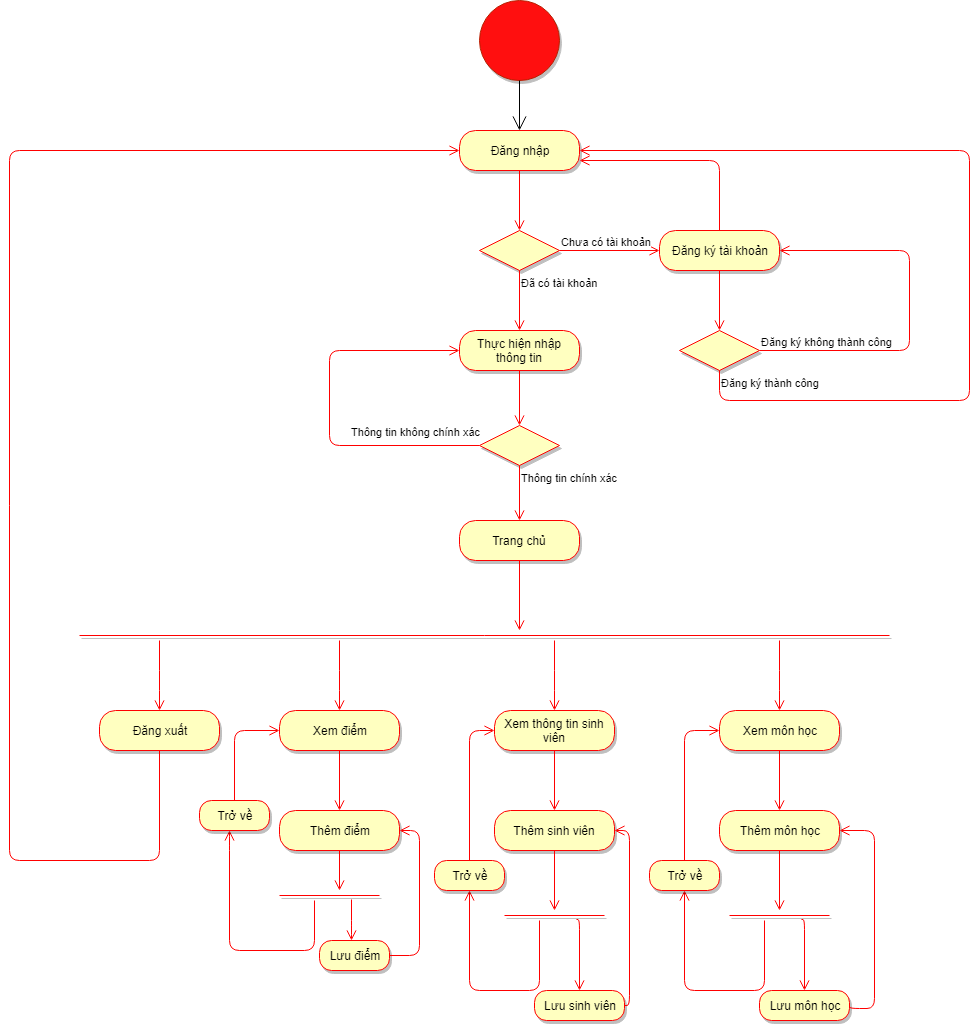
4.1 Kết quả đạt được 11

4.2 Các kết luận và kiến nghị 11

PHỤ LỤC 12

TÀI LIỆU THAM KHẢO 13

# DANH MỤC BẢNG BIỂU, HÌNH VẼ, SƠ ĐỒ



# CHƯƠNG 1. TỔNG QUAN VỀ REACT NATIVE

## Giới thiệu tổng quan về React native

* React Native cho phép bạn xây dựng ứng dụng dành cho thiết bị di động( cả Android và IOS) mà chỉ cần dùng JavaScript. Nó sử dụng cùng một thiết kế như React, bạn có thể tạo một giao diện người dùng (UI) phong phú từ các components được tối ưu cho thiết bị di động.
* Điều gì làm cho React Native trở nên khác biệt với các framework khác cũng sử dụng JavaScript để tạo ứng dụng cross-platform .
* Component của React Native được kế thừa từ React, và nó đóng vai trò như hạt nhân thúc đẩy React và React Native phát triển, với khả năng tái sử dụng và quản lý giao diện, Component API được xem như là tương lai của thiết kế UI trên web và mobile.
* Với React Native, nó cho phép bạn render các View của mình bằng chính **true native API**, bằng cách gọi chính SDK theo từng nền tảng để build ứng dụng.
* Trong một số tính năng đặc thù cần phải tương tác nhiều với phần cứng của thiết bị như: tính năng camera, tính năng Bluetooth… thì bạn cần phải sử dụng code native(Java hoặc Swift). React Native cho bạn code native ngay chính trong project và có thể tương tác với code Javascript bên ngoài.
* **Ưu điểm:**
  + Hiệu quả về mặt thời gian khi mà bạn muốn phát triển một ứng dụng nhanh chóng.
  + Hiệu năng tương đối ổn định.
  + Cộng đồng phát triển mạnh.
  + Tiết kiệm tiền.
  + Team phát triển nhỏ.
  + Ứng dụng tin cậy và ổn định.
  + Xây dựng cho nhiều hệ điều hành khác nhau với ít native code nhất.
  + Trải nghiệm người dùng tốt hơn là hybrid app.
* **Nhược điểm:**
  + Vẫn đòi hỏi native code.
  + Hiệu năng sẽ thấp hơn với app thuần native code.
  + Bảo mật không cao do dựa trên JS.
  + Quản lý bộ nhớ.
  + Khả năng tùy biến cũng không thực sự tốt đối với một vài module

## 2. Kiến thức cơ bản về ES6 trong React native

**Var – Let – Const**

### const

const dùng để khai báo một hằng số - là một giá trị không thay đổi được trong suốt quá trình chạy.

const A = 5;

A = 10; *// Lỗi Uncaught TypeError: Assignment to constant variable*

### let – var

**Var** và **let** đều là từ khoá – keyword. V**ar** và **let** cùng được dùng để khai báo biến trong JavaScript

Let tạo ra một biến chỉ có thể truy cập được trong block bao quanh nó, khác với var - tạo ra một biến có phạm vi truy cập xuyên suốt function chứa nó. Phạm vi của biến số sử dụng **let** là phạm vi một khối (Block), xác định bởi cặp {}. Phạm vi của biến số sử dụng **var** là phạm vi hàm số hoặc bên ngoài hàm số, toàn cục.

Sử dụng var:

function foo() {

var x = 10;

if (true) {

var x = 20; *// x ở đây cũng là x ở trên*

console.log(x); *// in ra 20*

}

console.log(x); *// vẫn là 20*

}

Sử dụng let:

function foo() {

let x = 10;

if (true) {

let x = 20; *// x này là x khác rồi đấy*

console.log(x); *// in ra 20*

}

console.log(x); *// in ra 10*

}

Ngoài ra, khi ở global scope (tức là không nằm trong một function nào cả), từ khóa var tạo ra thuộc tính mới cho global object (this), còn let thì không:

var x = 'global';

let y = 'global';

console.log(this.x); *// "global"*

console.log(this.y); *// undefined*

**CLASS - OBJECT ES6**

**OBJECT**

**Đối tượng (Object) có thể chứa nhiều giá trị :**

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**Đối tượng cũng có thể chứa phương thức hàm – function :**

var person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};

**Truy cập thuộc tính đối tượng, có 2 cách :**

* *objectName.propertyName :* person.lastName;
* *objectName["propertyName"] :* person["lastName"];

**Truy cập hàm phương thức đối tượng :**

* *objectName.methodName() :* name = person.fullName();

**This – từ khóa trong đối tượng : là đại diện cho chính nó .**

var **person** = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return **this**.firstName + " " + **this**.lastName;  
  }  
};

**CLASS**

**Đây là cách tạo đối tượng cũ trong javascipt :**

var employee = {

name: null,

age: null,

setName: function (name) {

this.name = name;

},

getName: function () {

return this.name;

},

setAge: function (age) {

this.age = age;

},

getAge: function () {

return this.age;

}

};

**Từ ES6 , đối tượng có thể tạo bằng cách này :**

class Employee {

setName (name) {

this.name = name;

}

getName () {

return this.name;

}

setAge (age) {

this.age = age;

}

getAge () {

return this.age;

}

constructor(name,age,wifeName)

{

super(name,age);

this.wifeName = wifeName;

}

};

Và với ES6, nó cũng hỗ trợ chúng ta một phương thức đặc biệt mà bất kỳ ngôn ngữ lập trình nào cũng có đối với class đó là constructor - phương thức khởi tạo. constructor trong ES6 cũng có tác dụng tương tự, nó sẽ tự động được gọi khi đối tượng được khởi tạo. Để khai báo constructor trong ES6 thì các bạn chỉ cần khai báo một phương thức có tên là **constructor**.

**ARRAY – MẢNG TRONG ES6**

Một mảng là một biến đặc biệt, có thể chứa nhiều hơn một giá trị tại một thời điểm.

Cú pháp tạo mảng : var array\_name = [item1, item2, ...];

Ví dụ : var cars = ["Saab", "Volvo", "BMW"];

Sử dụng từ khóa new để tạo mảng ( tuy nhiên , 2 kiểu này đều giống nhau , nên để đơn giản, dễ đọc và tốc độ thực thi cao , hãy sử dụng phương thức đầu tiên ) :

var cars = new Array("Saab", "Volvo", "BMW");

Truy cập phần tử mảng :

var x = cars[0]; // => Truy cập phần tử mảng bắt đầu từ 0 .

Với array hoặc object trong ES6 cũng có sự đột phá giúp ích rất nhiều cho việc code. Phục vụ cho code dễ hiểu hiểu hơn, mà không gây nhầm lẫn .

VD: Với javascript thông thường

var arr = [1, 2, 3, 4];

var a = arr[0];

var b = arr[1];

var c = arr[2];

var d = arr[3];

đây là cách lấy biến trong mảng mà ta thường dùng để gán vào biến khác Tuy nhiên ES6 có 1 cách đơn giản ngăn hơn

let [a, b, c, d] = [1, 2, 3, 4];

**MAP VÀ SET**

Được giới thiệu từ ES6 : Map, Set là những cấu trúc dữ liệu giúp thao tác trên tập hợp.

**MAP**

Map, mảng kết hợp (associate arrays) hay từ điển (dictionary/dict) là những thuật ngữ dùng để chỉ một cấu trúc dữ liệu, cho phép bạn ánh xạ từ một khóa (key) tương ứng với một giá trị (value). Trong JavaScript, chúng ta có thể sử dụng object để thể hiện cấu trúc này.

const dict = {

hello: 'Xin chào',

bye: 'Tạm biệt',

}

console.log(dict['hello']) // Xin chào

Tuy nhiên, nếu dùng object thì bạn chỉ có thể dùng chuỗi làm khóa. Ngoài ra, cách này cũng có một số hạn chế khác. Lớp Map do ES6 giới thiệu sẽ giúp giải quyết những vấn đề này. Với Map, bạn có thể sử dụng bất cứ dạng dữ liệu nào để làm khóa.

const obj = { bar: 2 }

const dict = new Map()

dict

.set('foo', 123)

.set(obj, 'hello world')

dict.get('foo') // 123

dict.get(obj) // 'hello world'

// Lấy giá trị của một khóa không tồn tại

dict.get('wat') // undefined

Bạn cũng có thể truyền vào hàm dựng của Map một mảng các cặp giá trị dạng [key, value], ví dụ như sau:

const dict = new Map([

['foo', 123],

[obj, 'hello world']

])

Như đã nói ở trên, bạn có thể dùng bất cứ dạng dữ liệu gì để làm khóa cho Map, kể cả mảng, object, hàm, hay NaN.

const arr = [1]

const f = () => {}

dict

.set(arr, 'an array')

.set(f, 'a function')

.set(NaN, 'not a number')

Nếu trong map đã có sẵn *khóa*, dữ liệu mới sẽ bị ghi đè lên.

const m = new Map()

m.set('foo', 1)

m.set('foo', 2)

m.get('foo') // 2

Để duyệt qua các *khóa* và giá trị trong Map, bạn có thể dùng:

const dict = new Map([

['foo', 1], ['bar', 2]

])

dict.keys() // ['foo', 'bar']

dict.values() // [1, 2]

dict.entries() // [ ['foo', 1], ['bar', 2] ]

dict.forEach(function(value, key, map) {

console.log(`${key} has ${value}`)

}, /\* thisArgs bạn có thể truyền vào tham chiếu cho `this` ở đây \*/)

// Sử dụng for..of cùng với destructuring

for (let [key, value] of dict) {

console.log(`${key} has ${value}`)

}

Bạn cũng có thể dùng toán tử spread ... với Map

const dict = new Map([

['foo', 1], ['bar', 2]

])

console.log([

['wut', 3],

...dict

])

// [ [ 'wut', 3 ], [ 'foo', 1 ], [ 'bar', 2 ] ]

Một số thao tác khác với Map.

const dict = new Map([

['foo', 1], ['bar', 2]

])

// Đếm số cặp giá trị trong map

dict.size // 2

// Kiểm tra trong map có khóa "foo" hay không

dict.has('foo') // true

dict.has('wut') // false

// Xóa một khóa, trả về boolean nếu thành công, false nếu thất bại

dict.delete('wut') // false

dict.delete('foo') // true

// Xóa hết các cặp giá trị của map

dict.clear()

**SET**

Set là tập hợp các giá trị không bị trùng lặp, nghĩa là trong một set không thể có hai giá trị bằng nhau.

const s = new Set()

set

.add('red')

.add('blue')

.add('sweet')

.add('you')

s.size // 4

Bạn cũng có thể truyền một mảng vào hàm dựng của Set.

const s = new Set(['red', 'blue', 'sweet', 'red', 'you'])

console.log(s) // Set (4) {'red', 'blue', 'sweet', 'you'}

Bạn cũng có thể thấy giá trị 'red' bị trùng lặp đã được loại bỏ. Chúng ta có thể áp dụng Set để tạo ra một mảng chứa những phần tử duy nhất.

const a = ['red', 'blue', 'sweet', 'red', 'you']

const b = [...new Set(a)]

console.log(b) // [ 'red', 'blue', 'sweet', 'you' ]

Để duyệt qua các phần tử của Set, bạn có thể dùng các phương thức như với Map.

const s = new Set([1, 2, 3, 4, 5])

// Vì Set không có khái niệm keys nên kết quả của `s.keys()` và `s.values()` là như nhau.

s.keys()

s.values()

s.entries()

s.forEach(function(value, key, setReference) {

}, thisArg)

for (let el of s) {

console.log(el)

}

Một số thao tác khác trên Set.

const s = new Set([1, 2, 3, 4, 5])

// Xóa một phần tử trong set

s.delete(3) // Set (4) {1, 2, 4, 5}

// Xóa hết phần tử trong set

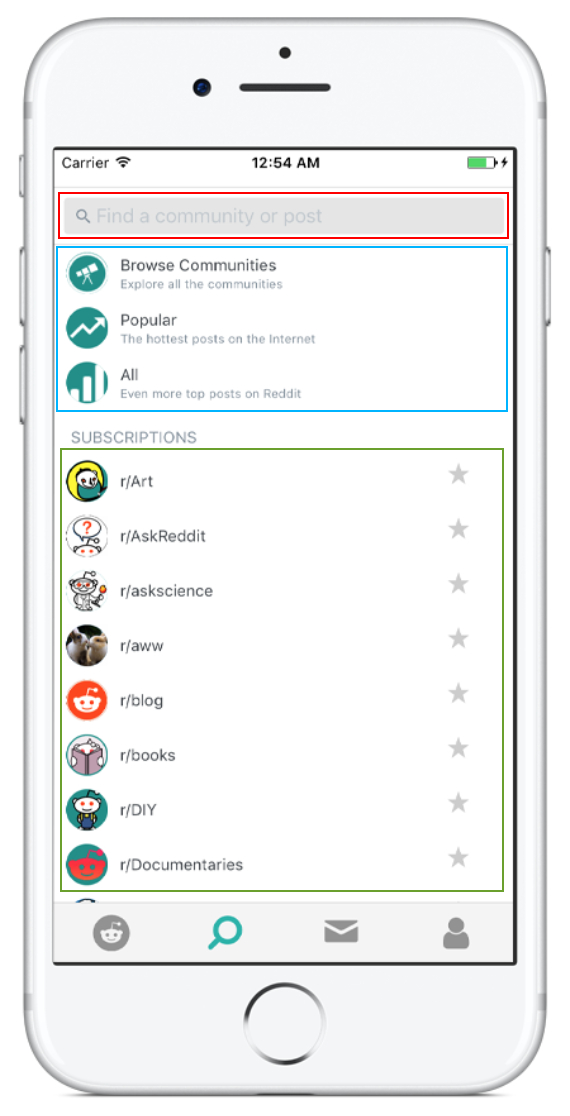
s.clear()

Với những cải tiến so với object thông thường, Map sẽ là công cụ hữu hiệu để lưu trữ dữ liệu dạng (khóa, giá trị). Trong khi đó, Set giúp bạn lưu trữ chuỗi dữ liệu mà không lo lắng về việc trùng lắp giá trị.

## 3. Component trong react native

Component là những thành phần hỗ trợ , phân ra nhiều loại khác nhau có khả năng tái sử dụng và tăng khả năng mở rộng .

Ví dụ : Những vùng khoanh đỏ , xanh , vàng … đều là component , nó là tập hợp những component giống nhau . Đi sâu hơn nữa thì mỗi thành phần như 1 danh sách ở vùng màu xanh gồm tập hợp nhiều Component khác như : Image , Text , Flex , View , …..



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

**PROPS**

* props là viết tắt của Properties.
* Một điều mà bạn cần phải nhớ khi sử dụng props đó là không bao giờ nên thay đổi giá trị của nó, hay nói cách khác, đây là một dữ liệu immutable.
* Các component nhận **props** từ component cha. Bạn 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.
* Bạn 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.

import React, { Component } from 'react';

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

class Greeting extends Component {

render() {

return (

<View style={{alignItems: 'center'}}>

<Text>Hello {this.**props**.name}!</Text>

</View>

);

}

}

export default class LotsOfGreetings extends Component {

render() {

return (

<View style={{alignItems: 'center'}}>

<Greeting name='Rexxar' />

<Greeting name='Jaina' />

<Greeting name='Valeera' />

</View>

);

}

}

**STATE**

* state thì hoạt động khác với props. state là dữ liệu nội bộ của một Component, trong khi props là dữ liệu được truyền cho Component.
* Chính vì vậy hoàn toàn có thể thay đổi state, và coi nó là một kiểu dữ liệu **mutable**.
* Tuy vậy, đừng bao giờ thay đổi trực tiếp biến this.state. Thay vào đó hãy dùng hàm setState để cập nhật giá trị.
* Sở dĩ 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.
* **setState** chạy bất đồng bộ, vậy nên nếu tiến hành đọc ra giá trị state ngay **sau khi setState** thì chưa chắc giá trị sẽ được update .

Một ví dụ thiết lập hàm **setState** sẽ ẩn - hiện văn bản sau 1s thời gian :

class Blink extends Component {

constructor(props) {

super(props);

**this.state = { isShowingText: true };**

setInterval(() => ( // Toggle the state every second

**this.setState**(previousState => (

{ isShowingText: !previousState.isShowingText }

))

), 1000);

}

render() {

if (!this.**state**.isShowingText) {

return null;

}

return (

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

);

}

}

// Class chứa hiển thị từ class - **Blink**

export default class BlinkApp extends Component {

render() {

return (

<View>

<Blink text='I love to blink' />

<Blink text='Yes blinking is so great' />

</View>

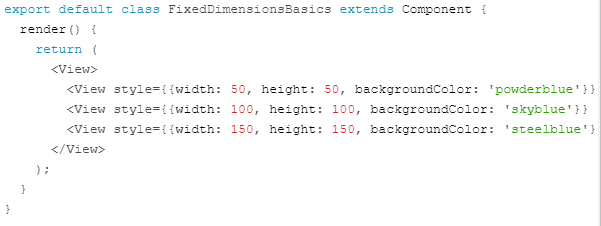
);

}

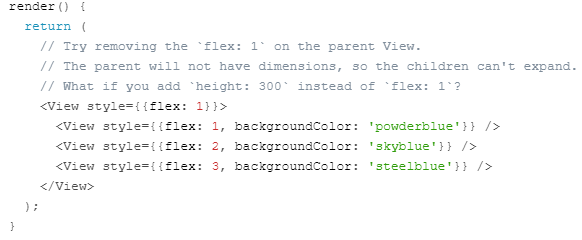
}

**BỐ CỤC THIẾT KẾ - WIDTH – HEIGHT – FLEX**

**Fixed Dimension** – kích thước cố định : Cách đơn giản nhất để thiết lập kích thước của một thành phần là thêm một kiểu cố định **width và height** kiểu. Tất cả các thứ nguyên trong React Native đều không có đơn vị và biểu thị các pixel độc lập với mật độ.



**Flex :** Sử dụng flex theo kiểu của một thành phần để có thành phần mở rộng và thu nhỏ động dựa trên không gian có sẵn .



**FLEX BOX**

* Flexbox được thiết kế để cung cấp bố cục nhất quán trên các kích thước màn hình khác nhau.
* Flexbox hoạt động theo cách tương tự trong React Native giống như trong CSS trên web, với một vài ngoại lệ. Mặc định là khác nhau, với flexDirection mặc định column thay vì row, và flex tham số chỉ hỗ trợ một số duy nhất.

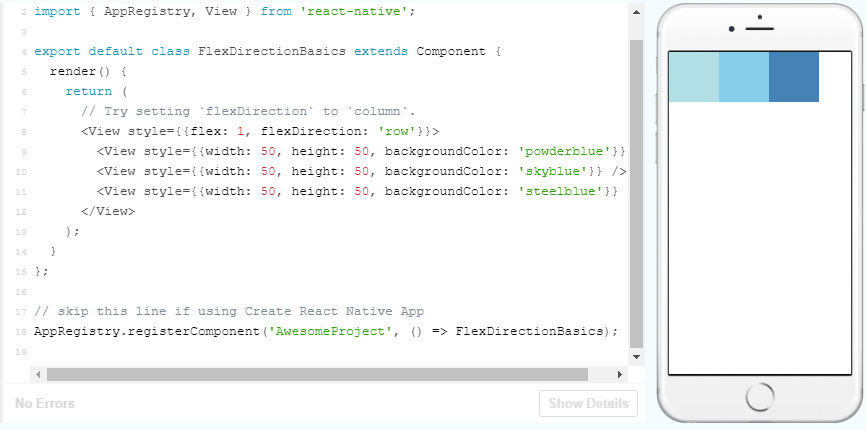
Bạn thường sẽ sử dụng một sự kết hợp của flexDirection, alignItems và justifyContent để đạt được bố trí đúng.

* flexDirection – Hướng dể phân chia bố cục



* justifyContent – căn đều nội dung
  + Tùy chọn sẵn có flex-start, center, flex-end, space-around, space-betweenvà space-evenly.
* alignItems – căn chỉnh các mục
  + Tùy chọn sẵn có flex-start, center, flex-end, và stretch.

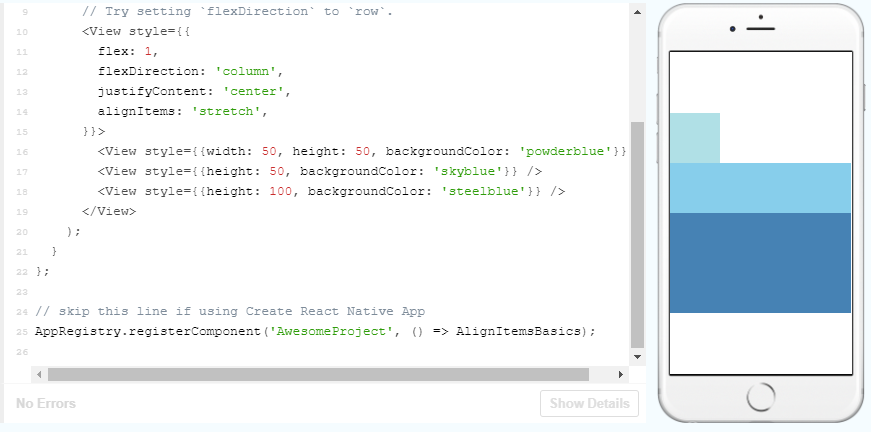
Flex – hiển thị theo chiều ngang – rows



Flex – hiển thị theo chiều dọc – column cột :



Flex – thay đổi các giá trị : justifyContent …



## 4. API Trong React Native

## CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1 Phân tích hệ thống

### 2.1.1. #1: Màn Hình Đăng Nhập và Đăng Ký(Nguyễn Hữu Phú)

#### 2.1.1.1 Giao diện người dùng

#### 

Image1: Đăng Nhập

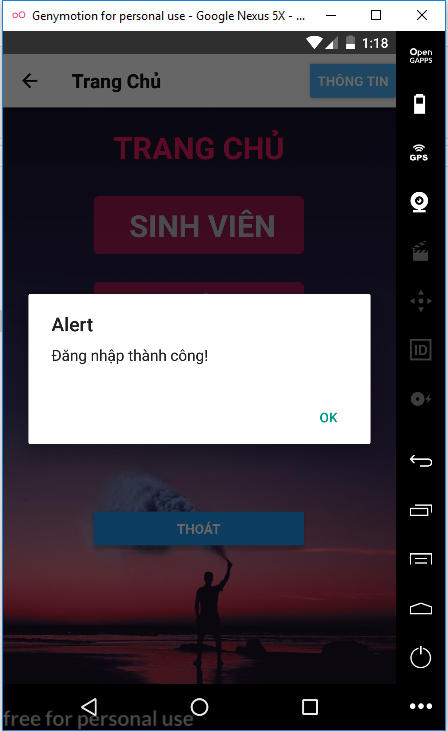


Image2: Đăng Nhập Thành Công

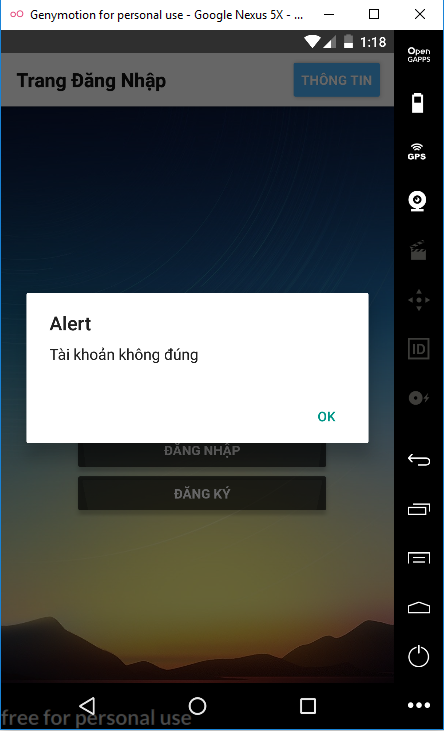


Image3: Đăng Nhập Không Thành Công



Image4: Đăng Ký

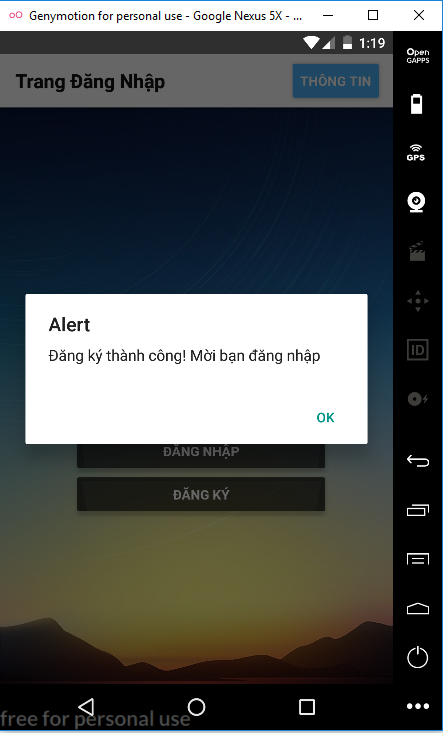


Image5: Đăng Ký Thành Công

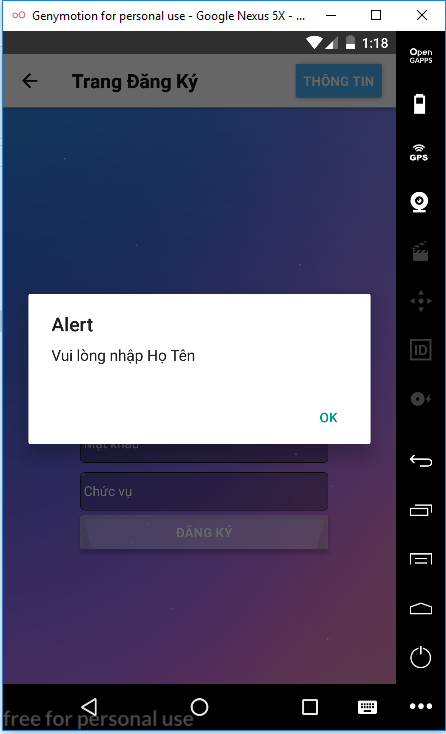


Image6: Đăng Ký Không Thành Công

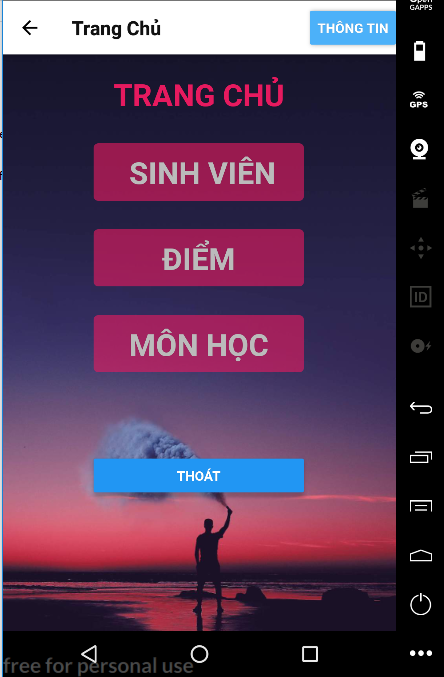
#### 2.1.1.2. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Mục** | **Mô tả** | **Hoạt động** | **Phản Ứng** |
| **Đăng Nhập** | Screen login have:   * 2 TextInput * 2 Button * Navigation(2 Button + 1 Label) | Acction Screen login:   * 2 TextInput: Get input value * “Đăng Nhập” - Send the data to the server * “Đăng Ký” - Navigate to the registration page | * 2 TextInput: Only retrieve data when entered * “Đăng Nhập” - Check and navigate to the “Home” page |
| **Đăng Ký** | Screen registration have:   * 4 TextInput * 1 Button   Navigation(2 Button + Label) | Acction Screen registration:   * 4 TextInput: Get input value * “Đăng Ký” - Check and send data on the server | * 4 TextInput: Only retrieve data when entered * “Đăng Ký” - Check and navigate to the “login” page |

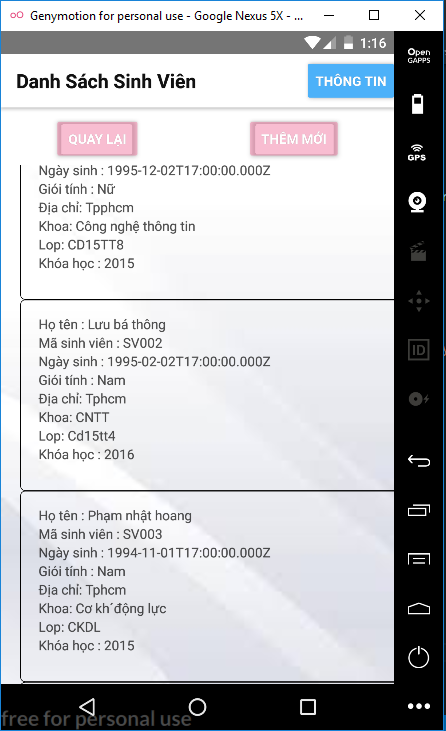
## 2.2 Thiết kế hệ thống

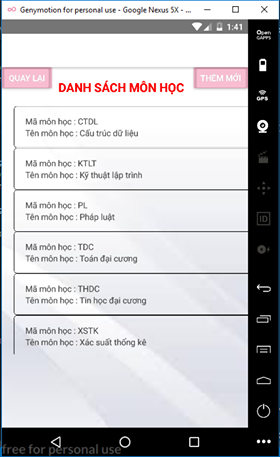
### 2.2.1. #2 Màn Hình Điều Hướng – Xem Thông Tin (Trần Cao Vinh)

#### 2.2.1.1. Màn hình chính – Trang điều hướng



#### 2.2.1.2. Màn Hình Xem Thông Tin

****

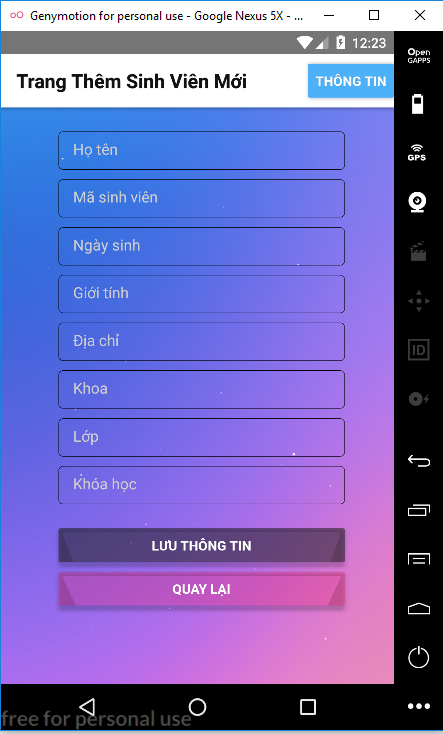


|  |  |  |  |
| --- | --- | --- | --- |
| **Mục** | **Mô tả** | **Hoạt động** | **Phản Ứng** |
| **Trang Chủ** | Screen Home have:   * 4 Button | Acction Screen login:   * “Sinh Viên” - Next to Screen “Danh Sách Sinh Viên” * “Điểm” - Next to Screen “Danh Sách Điểm” * the registration page * “Môn Học” – Next to Screen “Đăng ký môn học” * “Thoát” – Back to Screen “Đăng Nhập” | * Check Data Sign of Admin |
| **Danh Sách Sinh Viên** | Screen registration have:   * 1 FlatList * 2 Button   Navigation(2 Button + Label) | Acction Screen registration:   * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Sinh Viên” | * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Mới” |
| **Danh Sách Điểm** | Screen registration have:   * 1 FlatList * 2 Button   Navigation(2 Button + Label) | Acction Screen registration:   * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Điểm” | * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Điểm” |
| **Danh Sách Môn Học** | Screen registration have:   * 1 FlatList * 2 Button   Navigation(2 Button + Label) | Acction Screen registration:   * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Môn Học” | * 1 FlatList: Load data in the webserver * “Quay Lại” - Back to Screen “Trang Chủ” * “Thêm Mới” – Move in Screen “Thêm Điểm” |

## 2.3 Thiết kế hệ thống

### 2.3.1. #3 Màn Hình Thêm Thông Tin ( Phạm Minh Hiển )

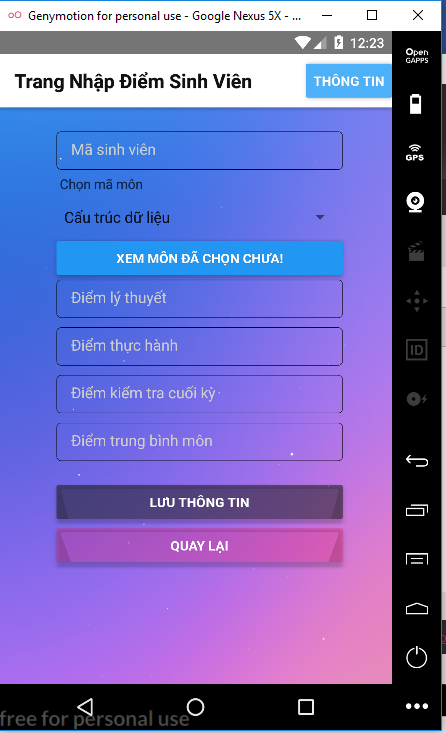
#### 2.3.1.1. Màn hình thêm sinh viên



#### 2.3.1.2. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **The screen adds new student information** | Screen add new student information have:   * 8 TextInput * 3 Button * Navigation (2 Button ) | N/A | N/A |
| **“Lưu thông tin” button** | Use to display a pop up check. | Tap on the Button | A pop up is showed with :  - “ Nhập đầy đủ thông tin trước khi thêm ”  - “ Sinh viên đã đăng ký ” - “Lỗi phần mềm ”  - “Thêm sinh viên mới thành công” |
| A button as a condition to navigate to home screen , if the information entered correctly . | Tap on the Button | Redirect to home screen |
| **“Quay lại” button** | A button as a condition to navigate to home screen | Tap on the Button | Redirect to home screen |
| **“Thông tin” button** | Button used to display information about the application | Tap on the Button | A pop up is showed with : “Ứng dụng quản lý sinh viên thuộc nhóm 6 ….” |

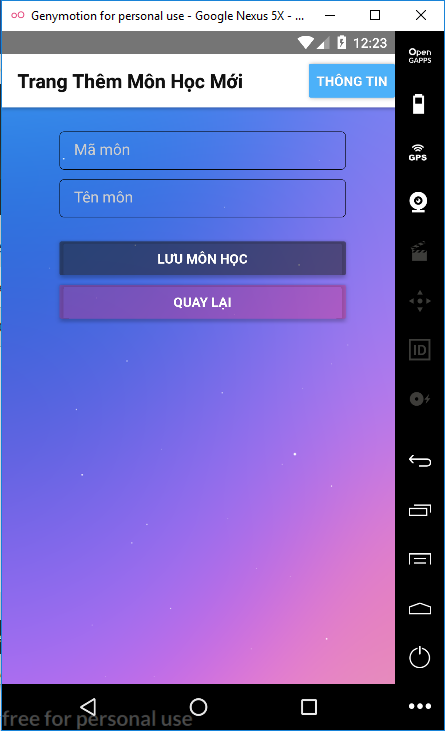
#### 2.3.2.1. Màn hình thêm điểm



#### 2.3.2.2. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Screen adds new student score information** | Screen add new student information have:   * 5 TextInput * 1 Text * 4 Button * Navigation (2 Button ) * 1 Picker | N/A | N/A |
| **“Lưu điểm” button** | Use to display a pop up check. | Tap on the Button | A pop up is showed with :  - “ Nhập đầy đủ thông tin trước khi thêm ”  - “ Chưa có thông tin về sinh viên này ”  - “Sinh viên đã nhập điểm ở môn học này ” - “Lỗi phần mềm ”  - “Thêm điểm cho sinh viên có mã … thành công” |
| A button as a condition to navigate to home screen , if the information entered correctly . | Tap on the Button | Redirect to home screen |
| **“Quay lại” button** | A button as a condition to navigate to previous screen | Tap on the Button | Redirect to previous screen |
| **“Xem môn đã chọn chưa” button** | The button checks whether the selected value is in the Picker | Tap on the Button | Show “Mã môn ” in a popup . |
| **“Thông tin” button** | Button used to display information about the application | Tap on the Button | A pop up is showed with : “Ứng dụng quản lý sinh viên thuộc nhóm 6 ….” |

#### 2.3.3.1. Màn hình thêm môn học



#### 2. 3.3.2. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **The screen adds new student information** | Screen add new student information have:   * 2 TextInput * 3 Button * Navigation (2 Button ) | N/A | N/A |
| **“Lưu môn học” button** | Use to display a pop up check. | Tap on the Button | A pop up is showed with :  - “ Vui lòng nhập đầy đủ thông tin trước khi thêm ”  - “ Môn học đã đăng ký ” - “Lỗi phần mềm ”  - “Thêm môn học thành công” |
| A button as a condition to navigate to home screen , if the information entered correctly . | Tap on the Button | Redirect to home screen |
| **“Quay lại” button** | A button as a condition to navigate to previous screen | Tap on the Button | Redirect to previous screen |
| **“Thông tin” button** | Button used to display information about the application | Tap on the Button | A pop up is showed with : “Ứng dụng quản lý sinh viên thuộc nhóm 6 ….” |

# CHƯƠNG 3. CÀI ĐẶT VÀ KIỂM THỬ

## 3.1 Cài đặt

Mô tả cách thức hiện thực hoá cho từng Activity, Service… của hệ thống

## 3.2 Kiểm thử

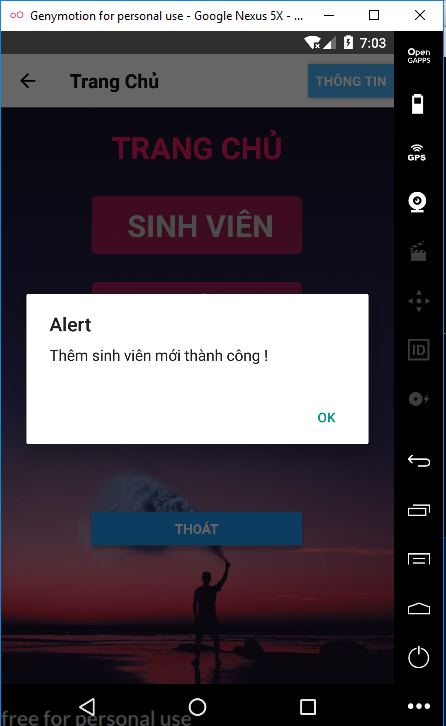
Các Test case của hệ thống là gì?

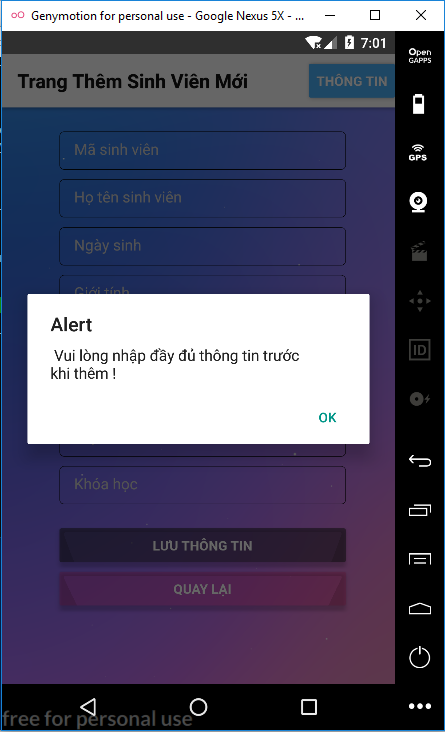
Kết quả kiểm thử ra sao?

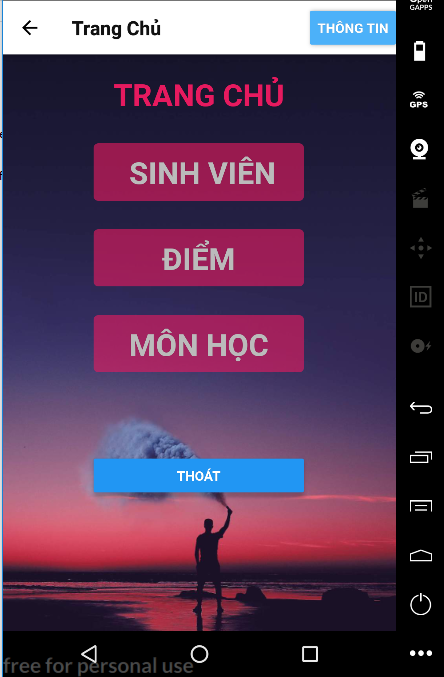
**CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC**

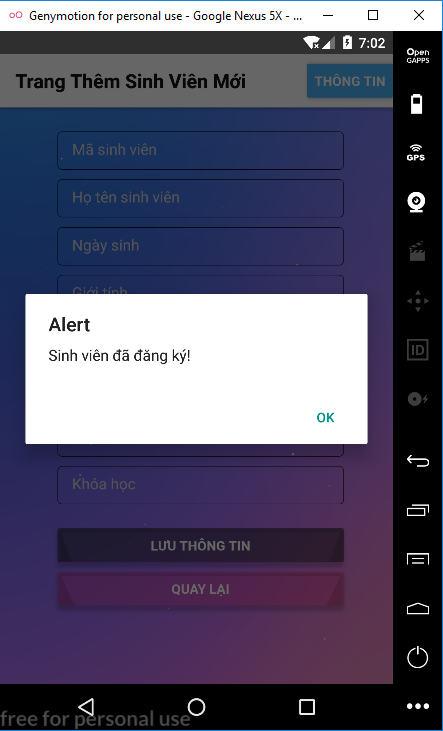
## 4.1 Kết quả đạt được

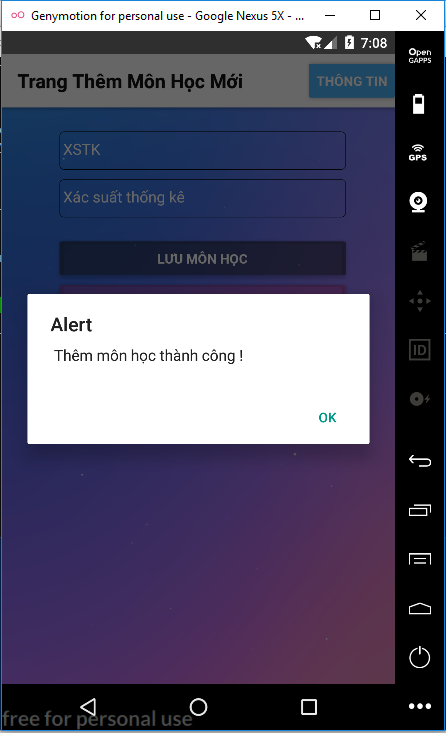
Mô tả các kết quả khi chạy ứng dụng (chụp các màn hình khi chương trình chạy, mô tả cách chạy, kết quả ra sao…)

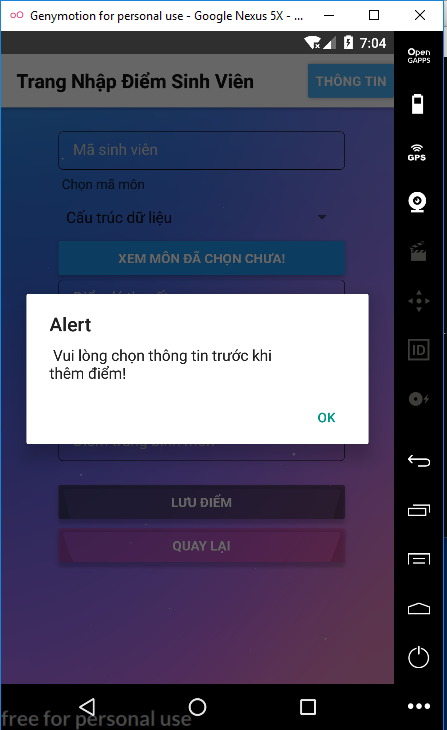


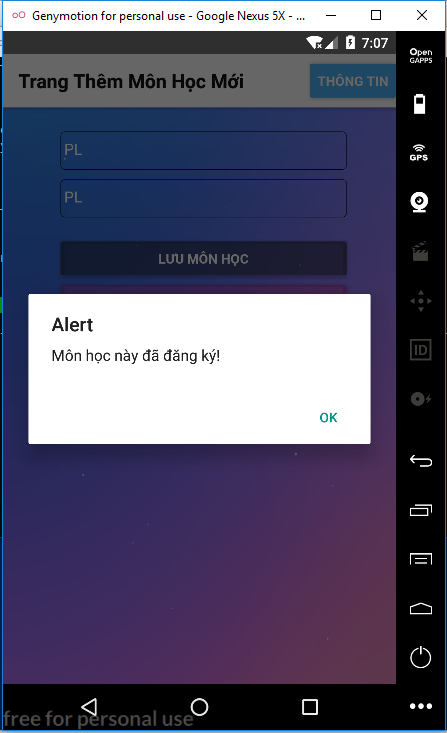


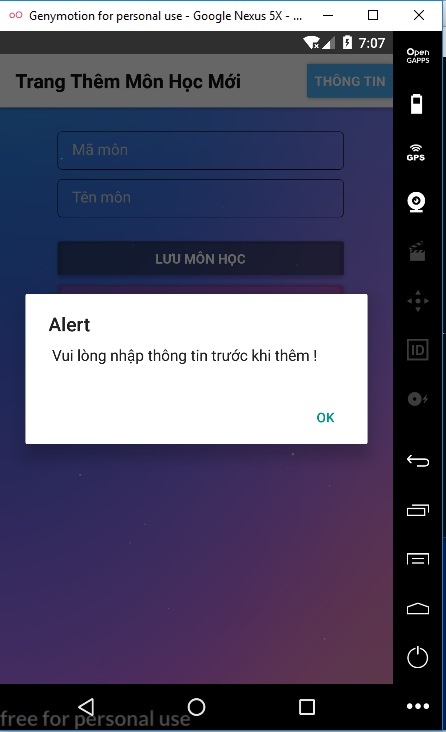




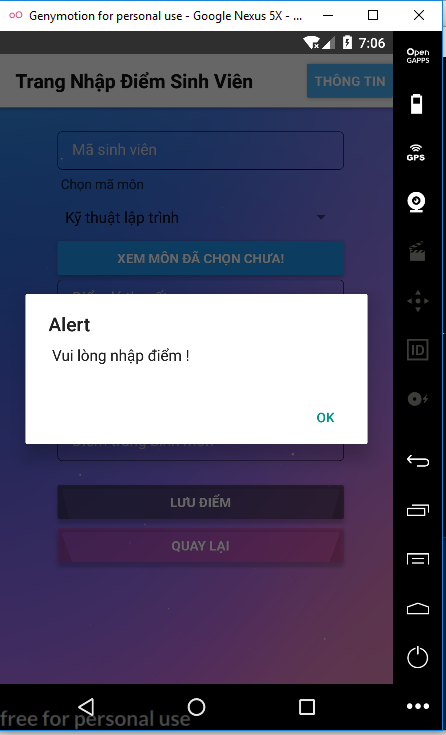


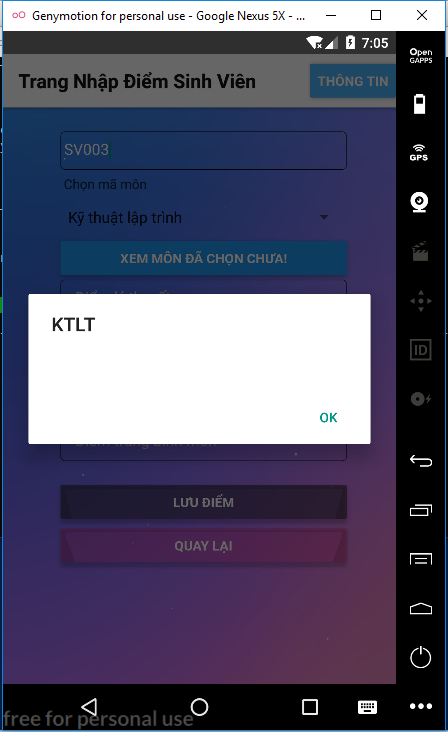


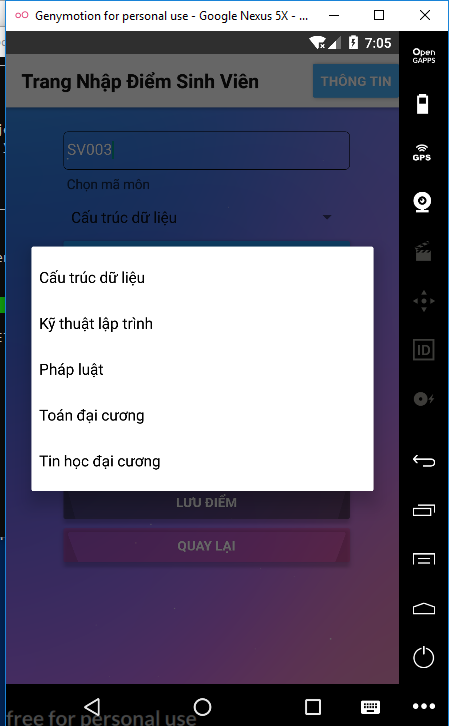


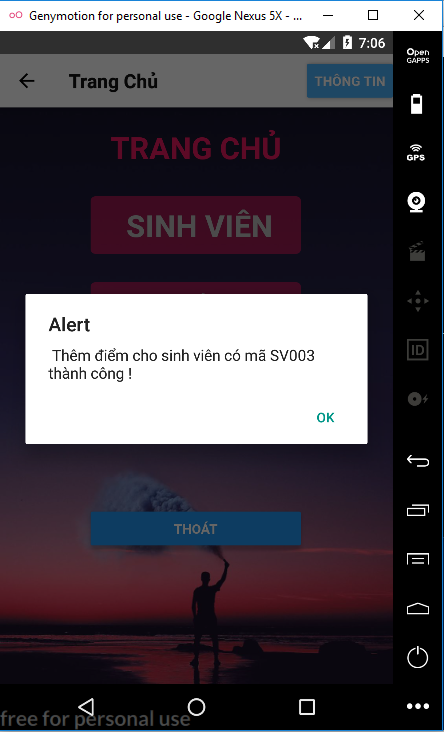












## 4.2 Các kết luận và kiến nghị

- Những điểm đã làm được

- Những điểm chưa làm được

- Các chức năng bổ sung nếu có thêm thời gian…

# PHỤ LỤC

# TÀI LIỆU THAM KHẢO

[1] https://facebook.github.io/react-native/docs/using-a-listview

[2] https://viblo.asia/p/overview-es6-part-2-L4x5xgNwlBM

[3] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object\_prototypes