**LỜI CẢM ƠN**

Được sự hướng dẫn của Thạc sĩ Nguyễn Văn Nam và sự đồng ý của Khoa Công nghệ thông tin trường Đại Học Thủy Lợi, em đã được nhận đề tài *“***Phát triển ứng dụng tìm kiếm địa điểm ăn uống trên mobile***”*cho đồ án tốt nghiệp kết thúc 4,5 năm quá trình học tại trường mình.

Để có được điều này, trước hết cho phép em được gửi lời cảm ơn chân thành đến toàn thể các thầy cô giáo trong trường Đại học Thủy Lợi nói chung và đặc biệt là các thầy cô giáo trong khoa Công nghệ thông tin nói riêng, những người đã tận tình dạy dỗ, truyền đạt cho em những kiến thức, kĩ năng học tập trong suốt thời gian học tập tại trường và tạo mọi điều kiện tốt nhất để em hoàn thành các môn học tại trường cũng như là đồ án tốt ngiệp này. Đặc biệt hơn hết, em xin gửi lời cảm ơn chân thành nhất tới thầy giáo - Ths Nguyễn Văn Nam - Giảng viên khoa Công nghệ thông tin, người đã nhiệt tình chỉ bảo giúp đỡ em rất nhiều trong suốt quá trình hoàn thành đồ án từ lúc phân tích tới thiết kế và hoàn thành đồ án.

Mặc dù nhiều cố gắng để đạt được kết quả tốt cho đề tài, nhằm phản ánh thực lực trong quá trình học tập và rèn luyện tại trường Đại Học Thủy Lợi, song do những hạn chế về kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót. Em rất mong sự góp ý của các Giảng viên để em ngày một cải thiện khả năng hơn trong học tập cũng như trong công việc sau này.

Một lần nữa em xin cảm ơn chân thành tới Ths Nguyễn Văn Nam cùng các thầy cô giáo trong khoa đã giúp em trong suốt quá trình hoàn thành đồ án tốt nghiệp này.

Em xin chân thành cảm ơn!

*Hà Nội, Ngày 01 tháng 11 năm 2017*

Sinh viên

**Phạm Ngọc Hải**

MỤC LỤC

[**CHƯƠNG 1: GIỚI THIỆU CHUNG** 4](#_Toc497443264)

[**1.1** **Giới thiệu đề tài** 4](#_Toc497443265)

[**1.2 Mục tiêu đề tài** 4](#_Toc497443266)

[**CHƯƠNG 2: KIẾN THỨC CHUNG** 5](#_Toc497443267)

[**2.1.1** **Giới thiệu về React Native** 5](#_Toc497443268)

[**2.1.5** **Bắt đầu code React native** 26](#_Toc497443269)

[**2.2 NodeJs** 45](#_Toc497443270)

[**2.2.1 Giới thiệu về NodeJs** 45](#_Toc497443271)

[**2.2.2 Các câu lệnh cơ bản** 45](#_Toc497443272)

[**2.2.3 NPM(Node Package Manager)** 45](#_Toc497443273)

[**2.3. MySQL** 46](#_Toc497443274)

[**2.3.1 Giới thiệu về MySQL.** 46](#_Toc497443275)

[**2.3.2** **Các câu truy vấn dữ liệu, điều kiện, gom nhóm** 48](#_Toc497443276)

[**2.4** **Kết nối NodeJs với Mysql** 49](#_Toc497443277)

[**CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ ĐỀ TÀI** 52](#_Toc497443278)

[**3.1 Định nghĩa lại bài toán** 52](#_Toc497443279)

[**3.1.1** **Xác định vai trò người dùng** 52](#_Toc497443280)

[**3.1.2 Các yêu cầu ứng dụng đối với người dùng** 52](#_Toc497443281)

[**3.2 Phân tích và đặc tả yêu cầu** 53](#_Toc497443282)

[**3.2.1** **Đặc tả yêu cầu** 53](#_Toc497443283)

[**3.2.2** **Phân tích yêu cầu** 65](#_Toc497443284)

[**3.3 Thiết kế** 129](#_Toc497443285)

[**3.3.1 Các chức năng của khách ( guest )** 129](#_Toc497443286)

[**3.3.2 Các chức năng của thành viên ( member )** 137](#_Toc497443287)

[**3.3.3 Các chức năng của admin** 144](#_Toc497443288)

[**3.3.4 Đồng bộ hóa các thực thể và thiết kế cơ sở dữ liệu** 162](#_Toc497443289)

[**CHƯƠNG 4: GIỚI THIỆU GIAO DIỆN ỨNG DỤNG** 178](#_Toc497443290)

[**CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 179](#_Toc497443291)

[**5.1 Kết luận** 179](#_Toc497443292)

[**5.2 Hướng phát triển đề tài** 179](#_Toc497443293)

[**CHƯƠNG 6: TÀI LIỆU THAM KHẢO** 181](#_Toc497443294)

[**6.1 Tài liệu tham khảo online** 181](#_Toc497443295)

[**6.2 Giáo trình** 181](#_Toc497443296)

# **CHƯƠNG 1: GIỚI THIỆU CHUNG**

## **Giới thiệu đề tài**

Xã hội của chúng ta đang ngày càng phát triển và đời sống con người vì thế càng ngày càng được nâng cao. Nhu cầu được sử dụng các thiết bị công nghệ thông minh như smartphone (điện thoại thông minh) ngày càng nhiều. Cùng với đó là nhu cầu về ăn uống. Một địa điểm ăn uống chất lượng tốt và thoải mái sẽ giúp mọi người có một bữa ăn ngon miệng và thư giãn. Nhưng để tìm được một địa điểm ăn uống tốt cũng không dễ dàng. Chính vì điều này, cần phải có một ứng dụng mà khi người dùng cài vào điện thoại của mình, họ có thể tìm kiếm được những địa điểm ăn uống mà họ mong muốn.

## **1.2 Mục tiêu đề tài**

* Tìm kiếm được địa điểm ăn uống theo tên.
* Tìm kiếm được địa điểm theo bán kính tính từ vị trí người dùng.
* Bình luận và đánh giá về địa điểm.
* Chỉ đường từ người dùng đến địa điểm.

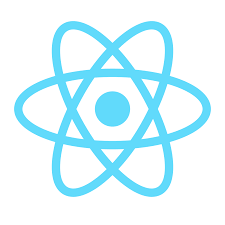
# **CHƯƠNG 2: KIẾN THỨC CHUNG**

Các công nghệ được sử dụng trong phát triển ứng dụng:

* React Native
* NodeJs
* Mysql

**2.1 React Native**

### **2.1.1 Giới thiệu về React Native**



**Giới thiệu về React Native:**

Trước khi cho ra đời React Native, Facebook đã cho ra đời ReactJS(gọi tắt là React). React Native có luồng dữ liệu và thiết kế giống với React, điểm khác biệt giữa React Native và React là React Native dùng trên môi trường mobile, còn React dùng trên môi trường browser.

Trước hết, React là gì? React là một thư viện viết bằng **javascript**, dùng để **xây dựng giao diện người dùng** (User Interface – UI).

Tháng 5/2013, React được facebook giới thiệu tới cộng đồng lập trình viên, từ thời điểm đó cho đến nay, React đã phát triển một cách nhanh chóng không chỉ trong nội bộ facebook mà còn cả cộng đồng bên ngoài. Hiện nay, rất nhiều dự án đã được xây dựng dựa trên React, tỷ lệ các lập trình viên chọn và sử dụng React ngày càng nhiều vì nó giúp tiết kiệm thời gian làm việc với framework và tập trung hơn vào sản phẩm của mình hơn.

React tập trung vào việc chia nhỏ ứng dụng ra thành nhiều thành phần nhỏ, mỗi thành phần thể hiện một view tương ứng. Những thành phần này có thể dễ dàng được sử dụng lại trong hệ thống, khi có yêu cầu thay đổi chúng ta không cần phải thay đổi toàn bộ hệ thống nữa mà thay vào đó chỉ cần thay đổi một phần nhỏ trong đó. Quan trọng hơn React đóng gói các thẻ DOM, API chỉ cần khai báo một lần, giúp tăng tính trừu tượng và đơn giản hóa mô hình lập trình. Xây dựng web với React giúp code của chúng ta sẽ dễ hình dung hơn, dễ đoán hơn. Tính dễ đoán này giúp chúng ta có thể tự tin sử dụng lại và web đáng tin cậy hơn. Ngoài ra, xây dựng web với React không những giúp web có khả năng mở rộng mà còn giúp chúng ta dễ dàng mở rộng qui mô của nhóm hơn.

Với công nghệ web và React, chúng ta có thể xây dựng rất nhiều trang web hoặc web app tuyệt vời. Tuy nhiên, trải nghiệm trên đó không thật sự tuyệt vời cho người dùng. Vì vậy, các ứng dụng android, ios dần dần lên ngôi. Đó là lý do Facebook cho ra đời React Native

Tháng 3 năm 2015 Facebook công bố React native, đến nay phiên bản hiện tại là 0.47, nó 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.

React Native sử dụng ES6/7 để các developers viết ứng dụng trên nó. Tuy nhiên Facebook chỉ lấy 1 phần (subset) của ES6/7 và sử dụng Babel để làm transpiler.

**2.1.2 ES6 và ES7 là gì?**

#### **ES6**

ES6 là chữ viết tắt của **ECMAScript 6,** đây được coi là một tập hợp các kỹ thuật nâng cao của Javascript và là phiên bản mới nhất của chuẩn **ECMAScript**. ECMAScript do hiệp hội các nhà sản xuất máy tính Châu Âu đề xuất làm tiêu chuẩn của ngôn ngữ Javascript

**Một số chức năng mới mà ES6 tích hợp vào ngôn ngữ javascript**

**Block Scoped**: Định nghĩa biến với từ khóa let, cách định nghĩa này thì biến chỉ tồn tại trong phạm vi khối của nó (Block Scope)

Block Scoped là phạm vi trong một khối, nghĩa là chỉ hoạt động trong phạm vi được khai báo bời cặp {}.

Ví dụ:

if(true){ let name = “hai”;} thì biến name chỉ tồn tại trong khối if.

**Arrow function**: có thể tạo hàm bằng cách sử dụng dấu mũi tên =>.

Cú pháp căn bản:

varfunctionName = (var1, var2) => {  // Nội dung function};

Ví dụ:

varhello = (name, message) => {console.log("Chào "+ name + ", bạn là "+ message);};

hello('Hải', 'sinh viên k55');

Trường hợp trong thân của hàm chỉ có một lệnh duy nhất thì có thể bỏ cặp {}, sử dụng theo ví dụ dưới đây.

varhello = (name, message) => console.log("Chào "+ name + ", bạn là "+ message);

Trường hợp truyền vào chỉ một tham số thì có thể bỏ cặp ().

Ví dụ:

varhello = message => {

    console.log(message);

};

hello('ES6');

Trường hợp không có tham số truyền vào thì sử dụng cặp () rỗng, xem ví dụ sau:

varhello = () => {

console.log('Chào mừng bạn đến với freetuts.net');

};

hello();

**Default Parameters:** gán giá trị mặc định cho các tham số.

Ví dụ:

functionsayHello(name = 'Hải')

{

    return name;

}

// Sử dụng

console.log("KHÔNG truyền tham số: "+ sayHello());

//KHÔNG truyền tham số: Hải

console.log("CÓ truyền tham số: "+ sayHello('Long'));

//Có truyền tham số: Long

**Destructuring Assignments**: Bạn có thể khởi tạo các biến từ một mảng  bằng một dòng code đơn giản.

**Destructuring Assignments:**Đơn giản chỉ là cách tách các phần tử của Array hoặc Object thành nhiều biến chỉ bằng một đoạn code duy nhất.

**Ví dụ:**

// Array

let date = [10, 03, 2016]

// Chuyển ba giá trị vào ba biến d, m, y

let [d, m, y] = date;

// In kết quả

console.log("Day: "+ d);   // Day: 10

console.log("Month: "+ m); // Month: 03

console.log("Year: "+ y);  // Year : 2016

Nếu muốn lấy một phần tử thôi thì hãy bỏ trống các phần tử không muốn lấy:

// Array

let date = [10, 03, 2016]

// Chuyển giá trị year vào biến y

let [, , y] = date;

// In kết quả

console.log("Year: "+ y); //Year: 2016

Ngoài mảng ra thì có thể tách biến từ object.

**Ví dụ:**

// Object

let date = {

    day : 10,

    month : 06,

    year : 2016

}

// Chuyển ba giá trị vào ba biến

let {day : d, month : m, year : y} = date;

// In kết quả

console.log("Day: "+ d);//Day: 10

console.log("Month: "+ m);//Month: 06

console.log("Year: "+ y);//Year: 2016

Với trường hợp là object có thể lấy một phần tử bất kì chứ không bắt buộc theo thứ tự như bên mảng.

// Object

let date = {

    day : 10,

    month : 06,

    year : 2016

}

// Chuyển giá trị year vào biến y

let {

    year : y

} = date;

// In kết quả

console.log("Year: "+ y);//Year: 2016

**Rest Parameter**: Tham số không giới hạn

Có thể khai báo một hàm với số lượng tham số không xác định, đây là một tính năng mới khiến Javascript ngày càng trở nên mạnh mẽ hơn.

Để khai báo các tham số còn lại của một function thì đặt 3 dấu chấm . trước biến đại diện.

let functionName = (param1, param2, ...other) => {}

**Ví dụ**: In ra tất giá trị của tham số truyền vào trong hàm.

// Khai báo hàm

let studentList = (name, age, ...other) =>

{

    console.log("Name: "+ name);

    console.log("Age: "+ age);

    console.log("Other");

    console.log(other);

}

// Gọi hàm

studentList('Hải', '22', 'haipn32@wru.vn', 'Thủy Lợi');

//Name: Hải

//Age: 22

//Other

//['haipn32@wru.vn','Thủy Lợi']

Như vậy biến other sẽ lưu một mảng các tham số thuộc …other.

**Set:**các kiểu dữ liệu phức tạp mới

Chúng ta có các thao tác chính khi làm việc với set như sau:

* **Khởi tạo**: let set = new Set();
* **Thêm phần tử**: set.add(value);
* **Xóa phần tử**: set.delete(value);
* **Kiểm tra tồn tại giá trị**: set.has(value);
* **Đếm tổng số phần tử**: set.size;
* **Xóa toàn bộ phần tử**: set.clear();

Đối với Set thì các giá trị không được trùng, vì vậy nếu cố tình thêm vào hai giá trị giống nhau thì nó chỉ lưu một lần.

**Khởi tạo:** Lúc khởi tạo sẽ có một tham số truyền vào, tham số này bắt buộc phải là một mảng.

varnumbers = newSet([1, 2, 3, 4]);

Thêm phần tử:

varnumbers = newSet([1, 2, 3, 4]);

numbers.add(5); // numbers = Set {1, 2, 3, 4 ,5}

Xóa phần tử:

varnumbers = newSet([1, 2, 3, 4]);

numbers.delete(2); // numbers = Set {1, 3, 4}

Kiểm tra sự tồn tại của phần tử:

varnumbers = newSet([1, 2, 3, 4]);

console.log(numbers.has(1)); // True

console.log(numbers.has(5)); // False

Đếm số phần tử:

varnumbers = newSet([1, 2, 3, 4]);

console.log(numbers.size); // 4

Xóa toàn bộ phần tử:

varnumbers = newSet([1, 2, 3, 4]);

numbers.clear(); // numbers = Set()

**Promise trong ES6**

Promise được đưa vào Javascript từ ES6, đây có thể coi là một kỹ thuật nâng cao giúp xử lý vấn đề bất đồng bộ hiệu quả hơn. Trước đây kết quả của một tác vụ đồng bộ và bất đồng bộ sẽ trả về một kiểu dữ liệu nào đó hoặc thực hiện một **Callback Function**. Với trường hợp thực hiện Callback Function thì sẽ dễ xảy ra lỗi **Callback Hell**, nghĩa là gọi callback quá nhiều và lồng nhau nên dẫn đến không kiểm soát được chương trình hoặc bộ nhớ không đủ để hoạt động. Promise, một package được đưa vào từ ES6 giúp giải quyết vấn đề Callback Hell này.

Khi một Promise được khởi tạo thì nó có một trong ba trạng thái sau:

**Fulfilled** Hành động xử lý xông và thành công

**Rejected** Hành động xử lý xong và thất bại

**Pending** Hành động đang chờ xử lý hoặc bị từ chối

Trong đó hai trạng thái **Reject** và **Fulfilled** ta gọi là **Settled**, tức là đã xử lý xong.

**Tạo một Promise:**

Để tạo một Promise sử dụng cú pháp sau:

varpromise = newPromise(callback);

Trong đó callback là một function có hai tham số truyền vào như sau:

varpromise = newPromise(function(resolve, reject){

});

**Trong đó**:

**resolve** là một hàm callback xử lý cho hành động thành công.

**reject** là một hàm callback xử lý cho hành động thất bại.

#### **ES7**

**Async/await một bổ sung mới trong ECMAScript 2016 (ES7) hướng tới cách viết gọn gàng súc tích và dễ dàng maintain, debug hơn trong sử lý bất đồng bộ (Asynchronous) mà trước đó được biết tới là Promise (ES6).**

Ở các phiên bản cũ hơn của ECMAScript (từ Es5 trở về trước) thì việc viết code async là dùng callback function. Về cơ bản thì **callback** chỉ là một function bình thường và khác biệt ở đây là trong khi các function thông thường sẽ trả về kết quả ngay khi kết thúc thì callback lại trì hoãn việc này. Do đó, callback hay được sử dụng trong trường hợp thực thi các lệnh liên quan tới nhập xuất dữ liệu như download, đọc file và giao tiếp với cơ sở dữ liệu,…

Async/await là một cách viết asynchronous code mới mà trước đó ta dùng Promise và cũ hơn là callbacks.

Async/await thực chất “**build on top of promises**”. và nó không thể được sử dụng trong việc gọi tới 1 plain callback, điều này có nghĩa là bạn bắt buộc phải gọi tới một api hoặc 1 action được tạo ra là promise.

Async/await giống promises là **Non-Blocking**(*nếu chúng ta yêu cầu 1 thao tác I/O là non-Blocking thì luồng điều khiển được trả lại ngay cho ứng dụng và ko cần chờ đợi tao tác I/O thực hiện xong.*)

Async/await giúp cho việc viết code bất đồng bộ (asyn) trở nên trông giống với viết code đồng bộ hơn (synchronous code)- *đây chính là ưu điểm nổi bật của nó.*

**Ví dụ**:

Lấy dữ liệu về qua api getJSON();

const makeRequest = async () => {

let data = await getJSON();

console.log(data);

}

makeRequest();

Ở đây từ khóa “**async**” khai báo ở trước function. còn “**await**” trong thân hàm (bắt buộc phải khai báo **asyncở trước hàm**thì bên trong ta mới dùng được **await**). Với keyword await đặt trước function**getJSON()** nó sẽ đợi thằng này cho tới khi nào xong thì mới chạy tiếp xuống câu lệnh**console.log(data)**

Kết hợp **try** và **catch** để bắt lỗi trong trường hợp parse bị lỗi.

**Ví dụ**:

const makeRequest = async () => {

try {

const data = JSON.parse(awaitgetJSON())

console.log(data);

} catch(err){

console.log(err);

}

}

makeRequest();

Trong một ứng dụng React Native trong nhiều trường hợp sẽ sử dụng đến async/await để xử lý dữ liệu.

**2.1.3 Các câu lệnh cơ bản trong React Native**

* import a,{b} from ‘c’ (để import thư viện, object, hoặc component vào 1 componet để sử dụng).
* export default Component\_name (export 1 component để các component khác có thể gọi và sử dụng) ( Tên component phai viết hoa chữ cái đầu)
* render ra nội dung sẽ hiển thị trên ứng dụng (phần nội dung phải được bao phủ bởi một component hoặc thẻ nào đó)

render(){

return(

<View>….</View>

)}

* props : props là một thuộc tính của Component . Có thể truyền dữ liệu qua các component sử dụng props (dữ liệu có thể là text, số, object hay function). this.props.props\_name để truy cập props được truyền.
* state : state giống như 1 biến private của 1 component, nó có thể thay đổi thường xuyên.
  + - Khởi tạo :

this.state = {state1: 0, state2: [], state3: {}}

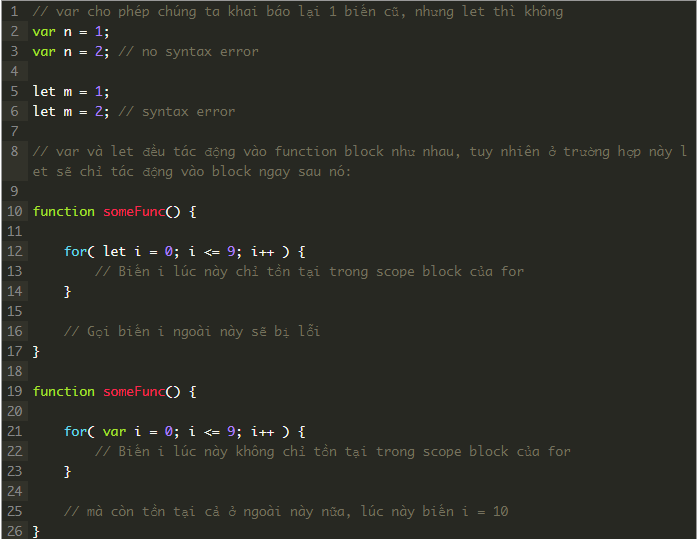
* + - Để thay đổi state dùng hàm

this.setState({state1: new\_value})

* + - Để lấy giá trị state

this.state.tên\_state

* **Khai báo biến trong React Native**.
* Sử dụng từ khóa var, let, const.
* Với var và let, đều có thể khai báo được 1 biến bất kỳ, biến này có thể thay đổi được giá trị. Điểm khác biệt giữ var và let đó là như hình dưới đây:



- const dùng để khai báo một hằng số - là một giá trị không thay đổi được trong suốt quá trình chạy. Nếu đã khai báo 1 biến với từ khóa const rồi sau gán lại giá trị cho biến đó thì sẽ bị lỗi.

* **Viết hàm trong React Native.**
* Hàm sẽ được khai báo và định nghĩa trước hàm render().
* Cú pháp: tên\_hàm() {…nội dung hàm…}
* Cách gọi hàm: this.tên\_hàm(tham số nếu có), this ở đây có nghĩa là đang đứng ở Component này và gọi hàm của Component đó.
* Hàm không tham số: Ví dụ

Thong\_bao() {

Return “Chào mừng bạn đến với React Native”;

}

Hàm trên sẽ trả về 1 đoạn string có nội dung là “Chào mừng bạn đến với React Native”

* Hàm có tham số: Ví dụ

Tinh\_tong(a, b) {

Return a+b;

}

* Nếu a,b là đều là số thì hàm sẽ trả về tổng của 2 số (Tinh\_tong(1,2) = 3).
* Nếu a hoặc b hoặc cả a và b đều là chuỗi thì hàm sẽ trả về chuỗi được nối bởi 2 chuỗi đó (Tinh\_tong(1,”2”) = “12”).
* Hàm không có giá trị trả về : Ví dụ

No\_return(…tham số nếu có…) {

Console.log(“Hàm không có giá trị trả về”);

}

Khi gọi hàm No\_return thì hàm sẽ log ra chuỗi “Hàm không có giá trị trả về”.

* Hàm có giá trị trả về: Ví dụ

Yes\_return(…tham số nếu có…) {

return “Hàm có giá trị trả về”;

}

Khi gọi hàm Yes\_return thì hàm sẽ trả về một chuỗi “Hàm có giá trị trả về”. Có thể dùng biến để lưu giá trị trả về này, ví dụ:

var string = this.Yes\_return(…tham số nếu có…);

khi đó biến string sẽ chứa chuỗi “Hàm có giá trị trả về”.

* Vòng đời của 1 component : gồm 3 thành phần chính là :
* Mounted: khi bắt đầu mở ứng dụng
* Updated: khi ứng dụng được dừng tạm thời sau đó quay lại sử dụng tiếp
* Unmounted: đóng hẳn ứng dụng

**Mounted:**

* Constructor: Phương thức khởi tạo đầu tiên của 1 component và luôn nhận vào 1 props hoặc 1 biến nào đó.
* ComponentWillMount(): hàm này chạy sau constructor và cũng là nơi có thể khai báo biến.
* Render(): hàm này sẽ render các elements của React ra giao diện.
* ComponentDidMount(): Hàm này thường dùng để code phần liên quan tới delay hoặc đồng bộ API.

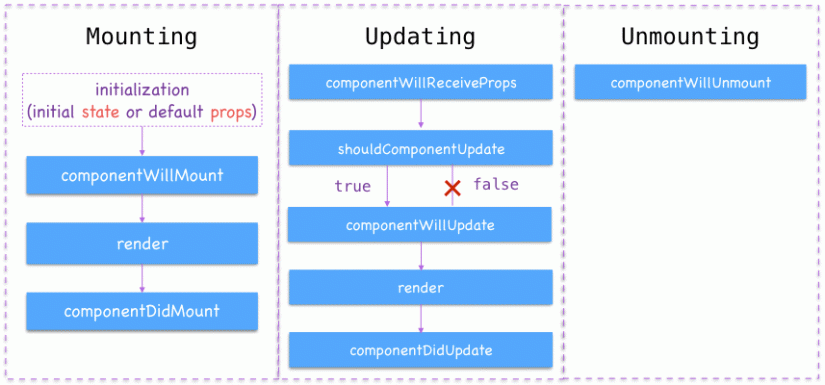
**Updated:**

* *ComponentWillReceiveProps(onject nextProps):* cha của component này sẽ trả về 1 props mới và khởi tạo lại giao diện. Chúng ta có thể cập nhật lại state nội bộ dựa vào this.setState() trước khi render được gọi lại.
* *ShouldComponentUpdate(object nextProps, object nextState):* Phương thức này có giá trị trả về kiểu booleam. Chúng ta dùng hàm này để kiểm tra xem Propsvà State có thay đổi hay không, nếu thay đổi thì trả về true và hàm render() sẽ được chạy lại và không thay đổi sẽ trả về false, hàm render() sẽ không được chạy.
* *render():* hàm này chỉ được chạy khi shouldComponentUpdate() trả về true.
* *componentDidUpdate(object prevProps, object prevState):* hàm này chỉ chạy khi render() đã chạy xong.

**Unmounted:**

* ComponentWillUnmount(): hàm này chạy khi ứng dụng bị đóng hoàn toàn. Component được gỡ khỏi DOM TREE.

Xem hình dưới đây



* **Cách lấy dữ liệu từ 1 api: Sử dụng hàm fetch**

Ví dụ:

fetch(api)

.then((response)=> response.json())

.then((responseJson)=>{

return responseJson.movies; (dữ liệu cần lấy)

})

.catch((error)=>{

console.error(error);

});

***- Tạo và chạy một ứng dụng React Native***

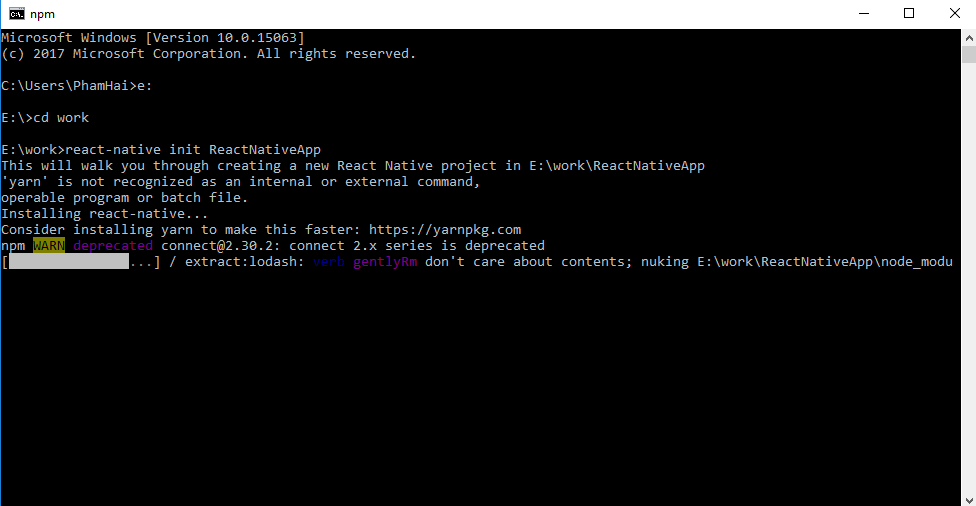
Tạo một ứng dụng React Native

* Bước 1: Mở cửa sổ cmd lên thực hiện lệnh sau:

npm install -g react-native-cli

* Bước 2: Trong cửa sổ cmd cd đến thư mục sẽ chứa ứng dụng thực hiện lệnh sau:

react-native init tên\_ứng\_dụng

Ví dụ như hình dưới (tên ứng dụng là ReactNativeApp)  


Hình 2. 1 Tạo ứng dụng React Native

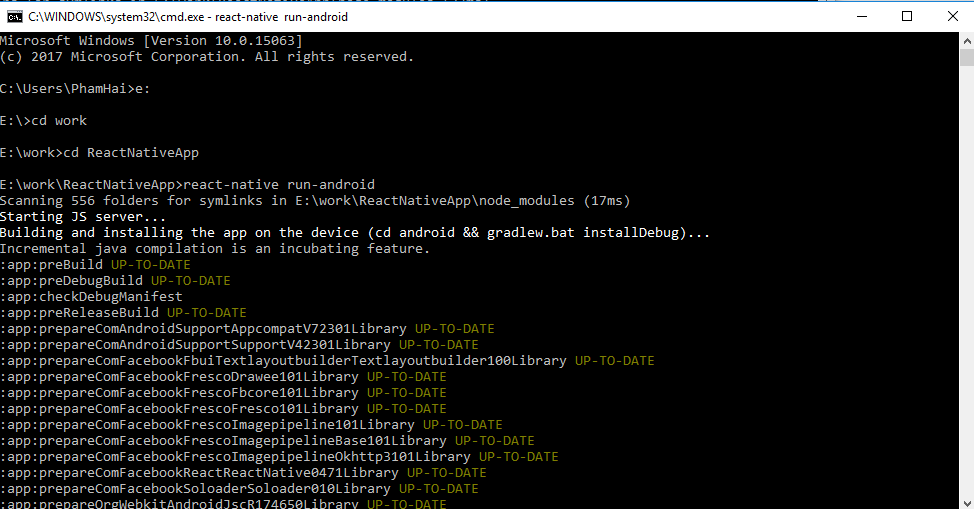
Sau khi câu lệnh được thực hiện xong thì việc tạo 1 ứng dụng React Native hoàn thành.

* Chạy một ứng dụng React Native
  + - Nếu chạy trên môi trường android thì cần có một chiếc điện thoại android hoặc máy ảo android.
    - Nếu có điện thoại android thì chỉ cần kết nối điện thoại với máy tính qua cáp usb rồi chạy ứng dụng thì ứng dụng sẽ tự động build và chạy trên chiếc máy android đó
    - Nếu không có điện thoại android thì sử dụng máy ảo Genymotion
    - Đây là giao diện máy ảo android của Genymotion



Mở cửa sổ cmd lên rồi cd vào thư mục ứng dụng

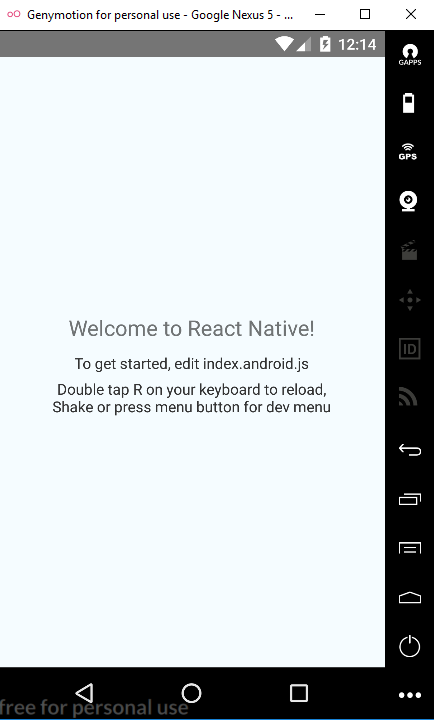
* Ví dụ như hình dưới đây



* Sau đó thực hiện lệnh

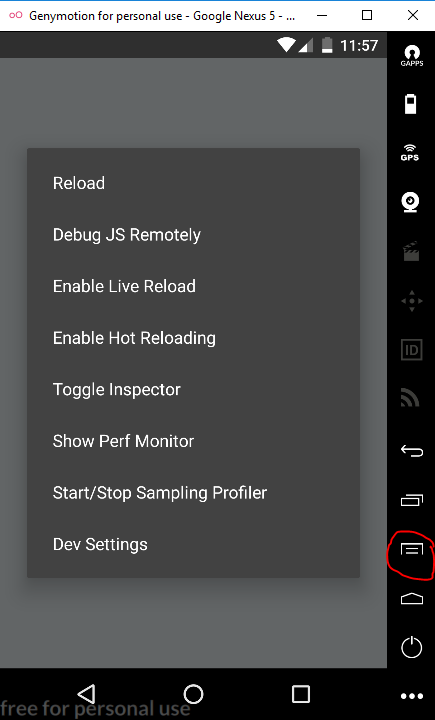
react-native run-android/run-ios(tùy theo nền tảng muốn chạy)

* Sau khi chạy xong thì ứng dụng được cài đặt trên máy ảo.
* Đây là giao diện ban đầu của ứng dụng trên android



* Sau này nếu cập nhập hoặc bổ sung code thì nhấn double vào phím R thì ứng dụng sẽ được cập nhật trên máy ảo.
* ***Cách xem log trong React Native***

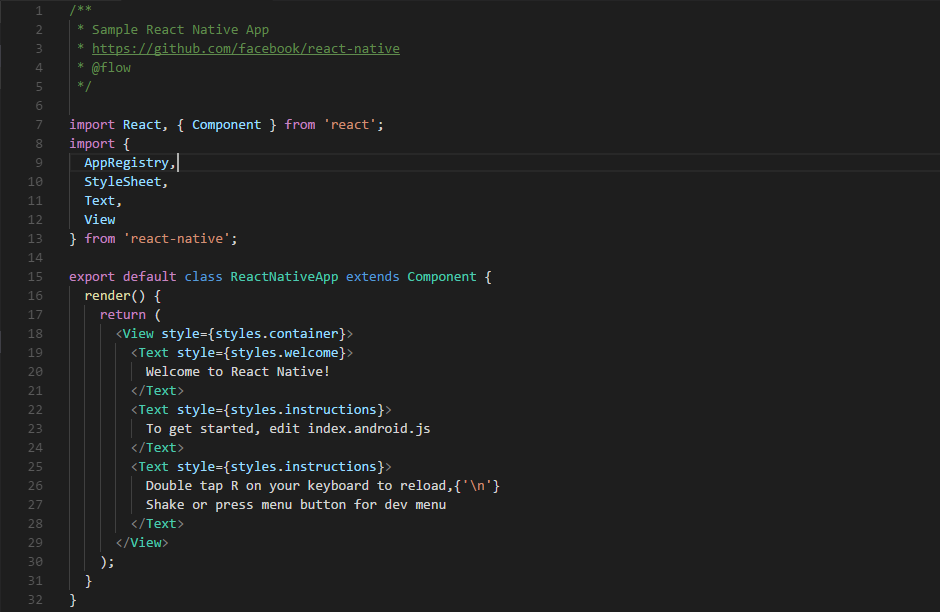
- Xem trên trình duyệt: Trên máy ảo nhấn tổ hợp phím Ctrl + M hoặc nhấn vào biểu tượng Menu (được khoanh đỏ ở hình dưới)



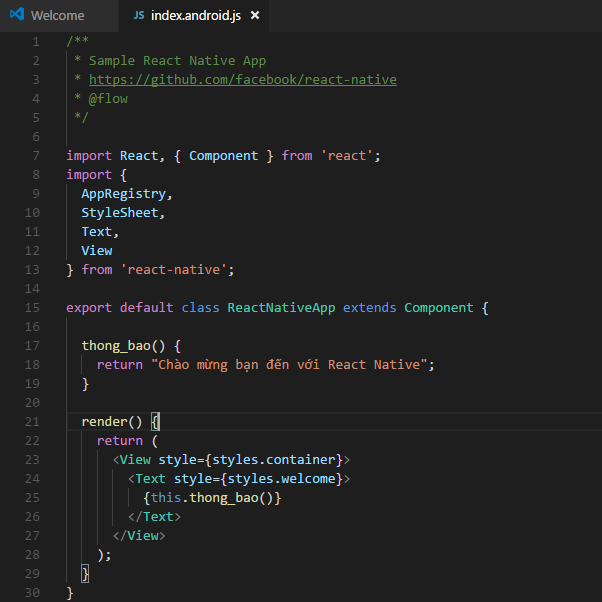
* + Sau đó chọn Debug JS Remotely, chờ một chút nó sẽ tự động mở trình duyệt, nhấn F12 và log sẽ được xem như xem log trên web.
  + Xem trên cmd: Mở một cửa sổ cmd mới và cd đến project rồi chạy câu lệnh **react-native run-android.** Log sẽ được hiển thị ở màn hình cmd đó.

### **2.1.5 Bắt đầu code React native**

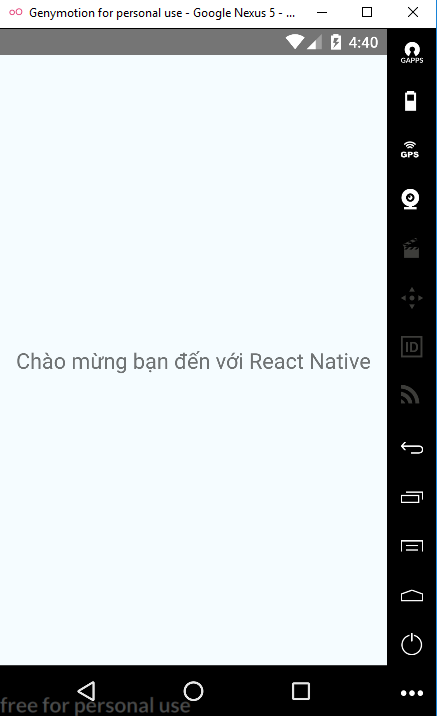
* Nếu chạy trên hệ điều hành android thì ứng dụng sẽ chạy file index.android.js, còn ở ios thì sẽ chạy file index.ios.js.
* Mở file index.android.js và sẽ có nội dung như hình dưới đây



* Nội dung được return trong hàm render sẽ chính là giao diện của người sử dụng.
* Chỉnh sửa file index.android.js : Viết hàm không tham số thong\_bao() trả về một chuỗi rồi gọi hàm đó trong thẻ <Text>{this.thong\_bao()}</Text>



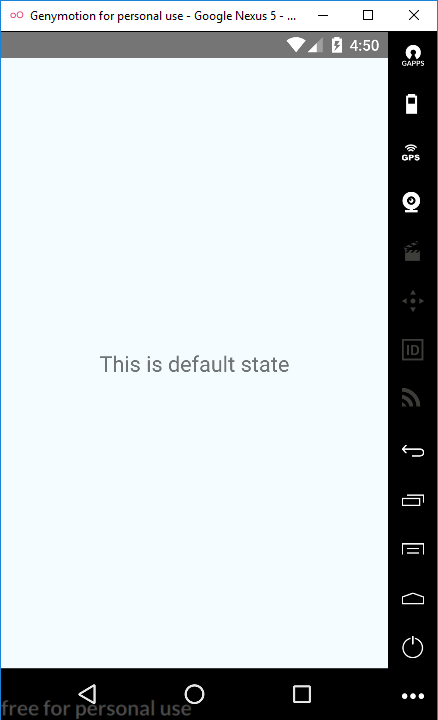
* Sau khi chạy kết quả thu được là :



* Thử dùng props và state trong React Native.
* **State**
* Tiếp tục chỉnh sửa file index.android.js như hình sau:



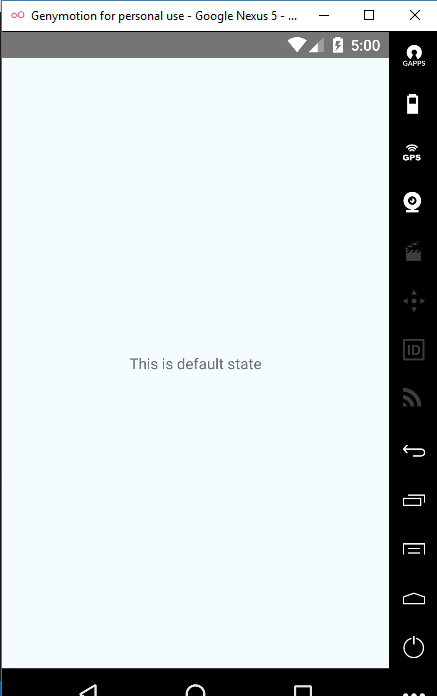
* Sau khi reload lại máy ảo thu được kết quả:



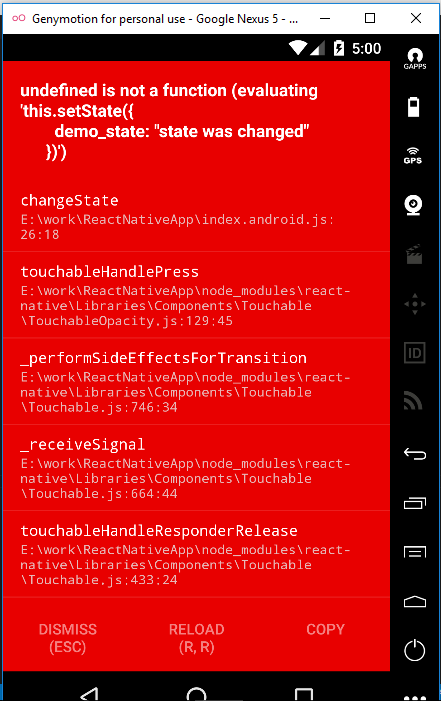
* Nội dung hiển thị chính là giá trị của state demo\_state. Như vậy là đã khởi tạo và gọi state thành công.
* Tiếp theo thử sự kiện thay đổi state:
* Tiếp tục sửa file index.android.js như sau:



* Ở lần sửa này có import thêm component TouchableOpacity. Component này có tác dụng như một button. Khi người dùng tác động vào component này thì sự kiện onPress được gọi và hàm changeState thực thi. Hàm changeState sẽ set lại giá trị cho state demo\_sate thành “state was changed”. Và đây là kết sau khi reload lại máy ảo



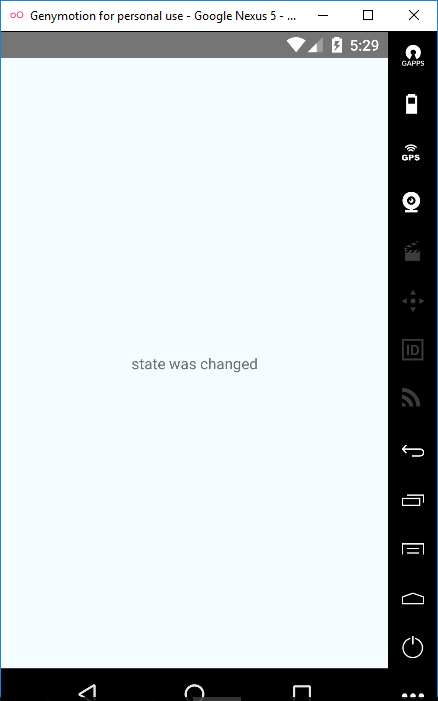
* Sau khi bấm vào dòng *This is default state* thì kết quả như sau:



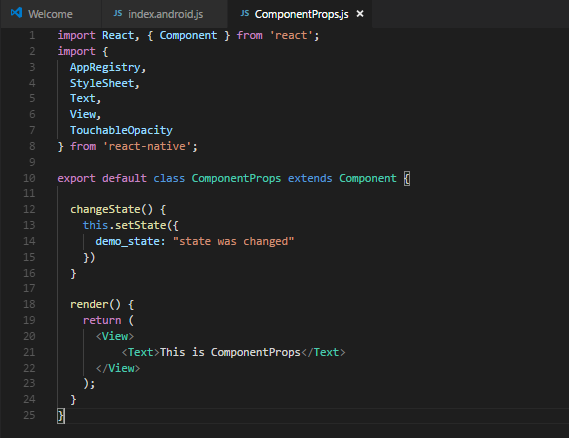
* Chương trình báo lỗi ở dòng 26 trong file index.android.js và lỗi là nó không hiểu this.setState là gì. Để sử dụng được this trong hàm changeState, cần phải bind(this) khi gọi đến nó.



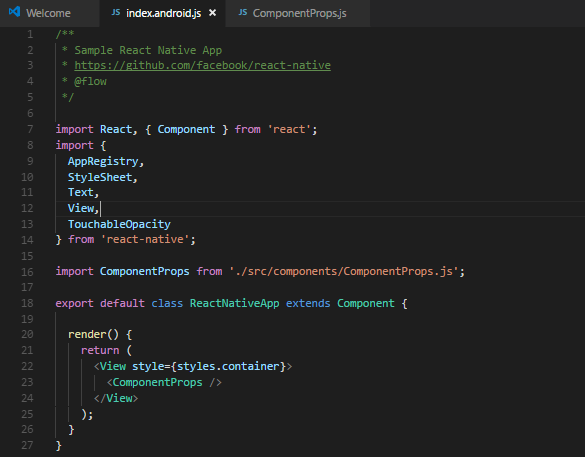
* Và kêt quả sau khi reload máy ảo và bấm vào *This is default state* là:



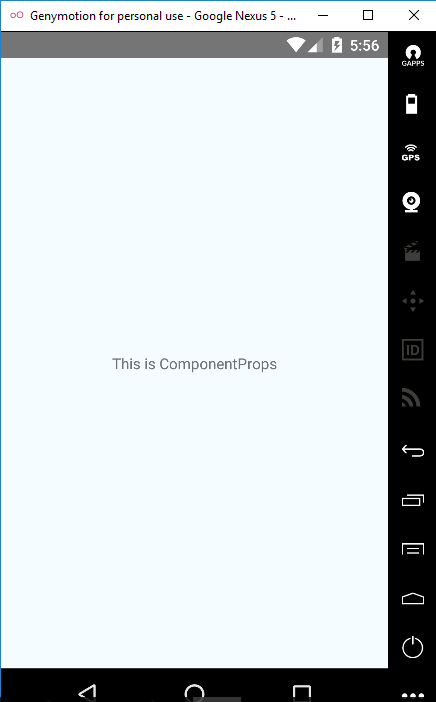
* Như vậy là đã thay đổi được state. Sau khi state thay đổi thì hàm render() sẽ chạy lại và nội dung mới của state được hiển thị.
* **Props**
  + Trước hết tạo file src/components/ComponentProps.js với nội dung như sau:



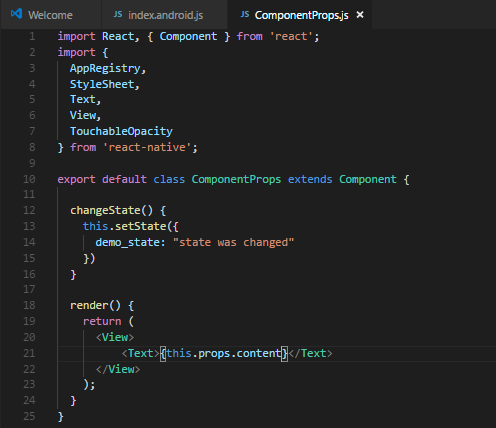
* Tiếp tục sửa file index.android.js
* Để sử dụng được component ComponentProps vừa tạo, cần phải import component đó vào.

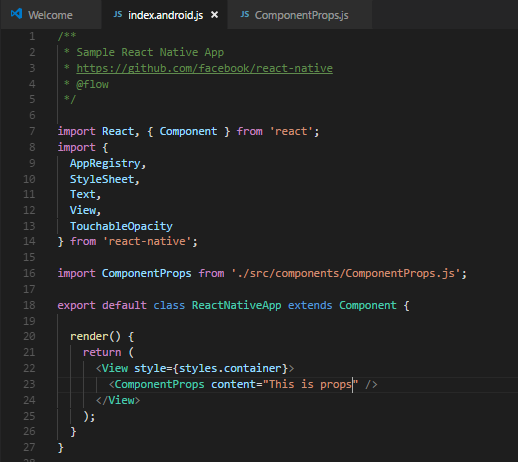


* Kết quả sau khi reload lại máy ảo như sau :

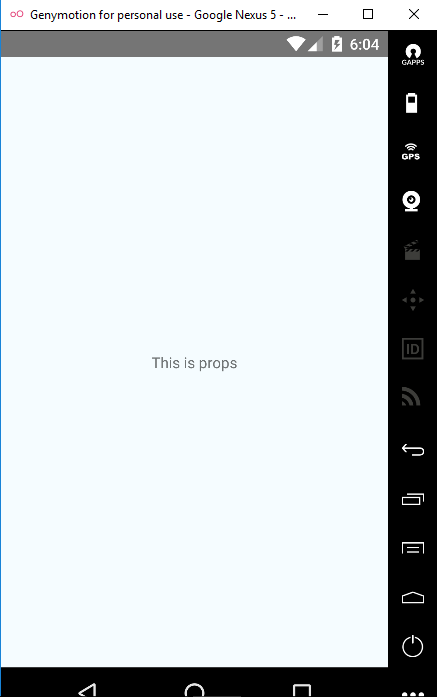


* Như vậy là đã import và sử dụng thành công được 1 component tự tạo bên ngoài.
* Bây giờ, để xem cách dùng props. Sửa file ComponentProps, index.android.js như sau :

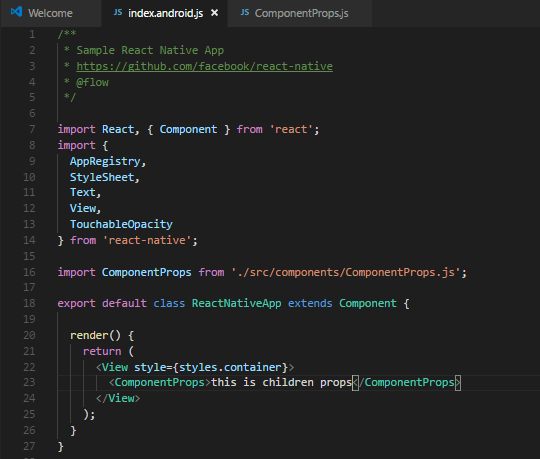


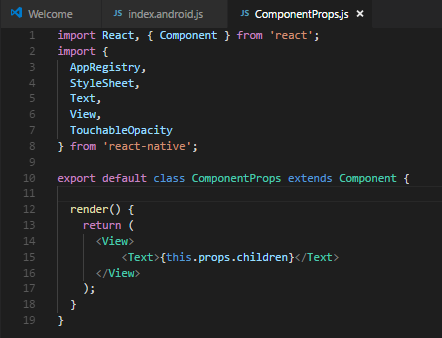


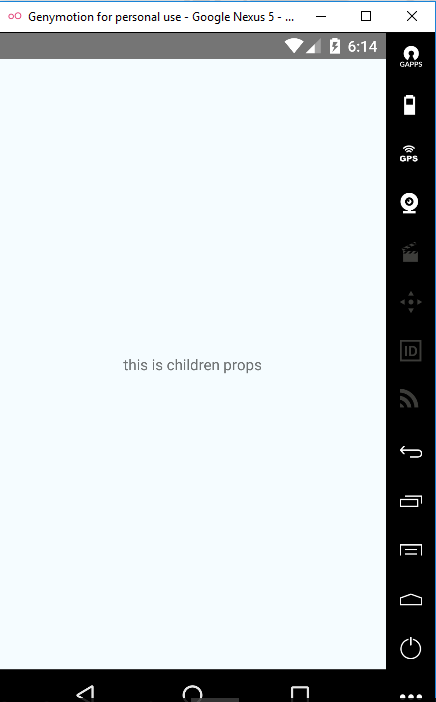
* Và kết quả sau khi reload máy ảo :



* Như vậy có thể thấy props content được truyền vào component ComponentProps. Từ đó ComponentProps có thể lấy được giá trị của content thông qua props và hiển thị nó.
* Ngoài ra còn có this.props.children để lấy giá trị nằm giữa thẻ đóng và thẻ mở cửa 1 component. Ví dụ :
* Sửa file index.android.js và ComponentProps.js như sau :







* Như vậy ComponentProps đã lấy được this.props.children là đoạn văn bản nằm giữa thẻ đóng và thẻ mở của nó.

## **2.2 NodeJs**

### **2.2.1 Giới thiệu về NodeJs**

NodeJS là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

V8 Engine là gì? V8 là một mã nguồn mở JavaScript engine viết bằng C++, phát triển bởi Google như là một phần trong dự án Chromium (dự án mã nguồn mở và là nền tảng cho Google Chrome), phát hành lần đầu cùng với phiên bản đầu tiên của trình duyêt Chrome. V8 compile trực tiếp JavaScript thành mã máy (native machine code ) thay vì sử dụng interpreting bytecode theo cách truyền thống.

### **2.2.2 Các câu lệnh cơ bản**

- Khai báo biến : var a = 8;

- Khai báo hàm : function test(có thể có tham số hoặc không){…nội dung hàm…}

- Gọi hàm : test(…tham số…);

- Vòng lặp :

for (var i = 0; i < 5; i++) { console.log(i); }

while (x < 5) { console.log(x++); }

-Điều khiển lựa chọn:

var a = 5;

if(a<10) { ...nội dung… }

### **2.2.3** **NPM(Node Package Manager)**

Khi nói về NodeJS thì không thể bỏ qua một chương trình được tích hợp sẵn trong trình cài đặt của NodeJS, hỗ trợ việc quản lý các gói sử dụng với NodeJS. NPM giúp quản lý các module, dự án, các thành phần có thể tái sử dụng thông qua mạng internet bằng các dòng lệnh(command line).

Một số những module của NPM dùng phổ biến hiện nay là:

* **Express** Một web framework của NodeJS, cung cấp nhiều tiện ích để phát triển một ứng dụng với NodeJS.
* **Connect** Là một HTTP server framework mở rộng, cung cấp các hỗ trợ thực thi mạnh mẽ cho Express.
* **Mongo & Mongojs** Là một thư viện bọc bên ngoài của MongoDB, cung cấp các hàm API để truy vấn dữ liệu từ NodeJS đến MongoDB.
* **Socket IO** Thành phần để xây dựng các ứng dụng thời gian thực.Ví dụ như Chat Server.
* **NodeMailer** Gửi mail với NodeJS.
* **Jade** Một trong những dạng template phổ biến, đơn giản hơn HTML, được tích hợp sẵn với Express.
* **Passport-facebook** Kết nối và chứng thực với facebook.

Ngoài ra còn rất nhiều module khác nữa do cá nhân và tổ chức chia sẻ và sử dụng với NPM, số lượng vào khoảng 37.000 module, mỗi ngày có hàng triệu lượt tải về và sử dụng.Nó chứng tỏ NodeJS đang ngày càng được sử dụng rộng rãi, là một nền tảng đầy tiềm năng để các lập trình viên có thể học tập và phát triển ứng dụng.

## **2.3.** **MySQL**

### **2.3.1 Giới thiệu về MySQL.**

MySQL là một trong những hệ thống quản lý cơ sở dữ liệu mã nguồn mở quan hệ SQL sử dụng trên web phổ biến nhất hiện nay.

MySQL được sử dụng cho việc bổ trợ PHP, Perl và nhiều ngôn ngữ khác, nó làm nơi lưu trữ những thông tin trên các trang web viết bằng PHP, Perl,…

**Một số đặc điểm của MySQL:**

* + Tốc độ: MySQL có tốc độ nhanh, được các nhà phát triển nhận xét rằng có thể MySQL là hệ thống quản lý cơ sở dữ liệu nhanh nhất.
  + Dễ sử dụng: MySQL là hệ thống quản lý cơ sở dữ liệu đơn giản, dễ cài đặt, sử dụng và quản lý.
  + Giá thành: MySQL được sử dụng miễn phí.
  + Năng lực, hiệu suất hoạt động: Nhiều client có thể truy cập đến server trong cùng một thời gian. Các client có thể sử dụng nhiều cơ sở dữ liệu một cách đồng thời. Bạn có thể truy cập MySQL tương tác với sử dụng một vài giao diện để bạn có thể đưa vào các truy vấn và xem các kết quả: Các dòng yêu cầu của khách hàng, các trình duyệt Web,…
  + Kết nối và bảo mật: MySQL được kết nối đầy đủ, người sử dụng có thể chia sẻ dữ liệu với bất kỳ ai, tới bất kỳ nơi nào có internet. Nhưng MySQL cũng có thể kiểm soát quyền truy cập nên bảo mật dữ liệu.
  + Tính linh động: MySQL chạy được trên nhiều hệ thống, nhiều phần cứng.
  + Ngoài ra MySQL được phân phối, sử dụng rộng rãi và được hỗ trợ từ cộng đồng và nhà phát triển.

Những định nghĩa cơ bản.

1. **Định nghĩa cơ sở dữ liệu, bảng, cột.**

CSDL là tên của CSDL chúng ta muốn dùng, có thể bao gồm 1 hay nhiều bảng.

Bảng là 1 bảng giá trị nằm trong CSLD, có thể bao gồm nhiều cột.

Cột là 1 giá trị nằm trong bảng, dùng để lưu trữ các trường dữ liệu, có thể có hoặc không có thuộc tính.

1. **Định nghĩa một số thuật ngữ.**

NULL: giá trị cho phép rỗng.

AUTO\_INCREMENT: Cho phép giá trị tự động tăng.

UNSIGNED: Phải là số nguyên dương.

PRIMARY KEY: Khóa chính trong bảng.

1. **Các kiểu dữ liệu trong MySQL.**

* Kiểu số:
* Int: Số có chiều dài -2147483648 đến 2147483647 hoặc 0 đến 4294967215.
* Float: Số thập phân có chiều dài nhỏ.
* Double: Số thập phân có chiều dài lớn.
* Kiểu chuỗi:
* Char: Chuỗi có độ dài tối đá 255 ký tự.
* Varchar: Chứa một biến chuỗi có độ dài tối đa 255 ký tự.
* Text: Chứa một văn bản có độ dài tối đa 65.535 ký tự.
* MediumText: Chứa một đoạn văn bản có độ dài tối đa là 16.777.215 ký tự.
* LongText: Chứa một đoạn văn bản có độ dài tối đa là 4.294.967.295 ký tự.
* Kiểu dữ liệu date time:
* Date: Trả về ngày tháng năm theo định dạng yyyy – mm – dd (năm – tháng – ngày).
* DateTime: Trả về ngày tháng năm và giờ phút giây theo định dạng “yyyy – mm – dd hh:mm:ss”.
* Time: Trả về một thời điểm theo định dạng “HH:MM:SS”.

### **2.3.2** **Các câu truy vấn dữ liệu, điều kiện, gom nhóm**

***Các câu truy vấn :***

* Select \* from tên\_bảng (lấy dữ liệu ở bảng).
* Insert into tên\_bảng(danh sách các trường dữ liệu) values (danh sách dữ liệu tương ứng các trường) (thêm dữ liệu cho bảng).
* update tên\_bảng set column1 = value1,….where điều\_kiện (cập nhật dữ liệu cho bảng với điều kiện).
* delete from tên\_bảng where điều\_kiện (xóa dữ liệu ở bảng với điều kiện, nếu k có điều kiện sẽ xóa hết dữ liệu ở bảng đó).

***Gom nhóm:***

* Sử dụng mệnh đề group by: select class\_name from sinhvien group by class\_name (sẽ lấy ra danh sách tên các class).

## **Kết nối NodeJs với Mysql**

-Bước 1: Tạo database Mysql.

Tạo một database test\_db có bảng user

-Bước 2: Tạo một project node js

Tạo một thư mục rỗng chứa project

Mở cmd và link đến thư mục đó

Viết dòng npm init và ấn enter

Cứ nhấn enter liên tục đến khi hỏi yes thì nhập y và ấn enter.

Lúc này vào thư mục chứa project sẽ thấy có file package.json.

Cài đặt express framework. Vẫn cmd ở thư mục chứa project và viết dòng npm install express

Tạo file server.js sau đó copy và dán đoạn sau vào

var express = require(“express”);

var app = express();

app.get(“/user”, function(req, res)){

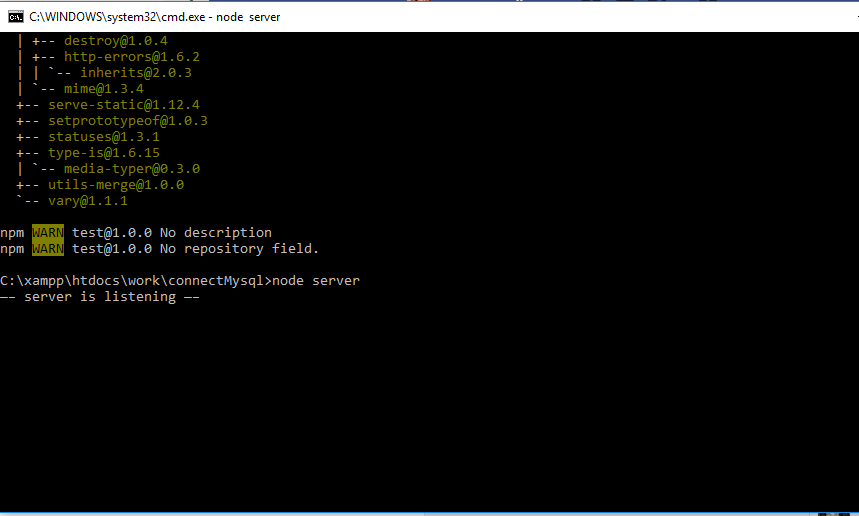
res.send(“Hello!!!”);

});

app.listen(3000);

console.log(*“—– server is listening —–“*);

Chạy project bằng cách mở cmd và link đến thư mục chứa project rồi viết dòng **node server.js** sau đó ấn enter.



Sau khi chạy sẽ được kết quả như hình.

-Bước 3: Kết nối với Mysql

* Cài đặt gói mysql : chạy dòng **npm install mysql**
* Sau đó mở file server.js và sửa thành như sau:

var express = require(*“express”*);  
var mysql = require(*“mysql”*); *// include thêm module mysql*  
var app = express();  
*// Tạo kết nối với Database*  
var connection= mysql.createConnection ({  
host: *“localhost”*,  
user: *“root”,*  
password: *null*,  
database: *“test\_db”(*tên database*)*,  
});  
app.get(*“/user”*, function(req, res){

*// Viết câu truy vấn sql*  
var sql = *“SELECT \* FROM ` user`”*;*// Thực hiện câu truy vấn và show dữ liệu*  
connection.query(sql, function(error, result){  
if (error) throw error;  
console.log(*“– USER TABLE — “* , result);  
res.json(result); *// Trả kết quả về cho client dưới dạng json*  
});

});  
app.listen(3000);  
console.log(*“—– server is listening —–“*);

* Sau đó chạy lại project rồi mở trình duyệt vào địa chỉ <http://localhost:3000/user>.
* Và đây là kết quả



* Dữ diệu trong bảng user được trả về dưới dạng json.
* Như vậy là đã hoàn thành việc kết nối NodeJs với Mysq

# **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ ĐỀ TÀI**

## **3.1 Định nghĩa lại bài toán**

### **3.1.1 Xác định vai trò người dùng**

* Hệ thống cho 3 nhóm người dùng : quản trị viên, thành viên, khách.
* Quản trị viên: Nhóm này có vai trò quản lý địa điểm, quản lý comment, quản lý danh mục món ăn,, quản lý món ăn.
* Khách: Nhóm này có vai trò sử dụng hệ thống để tìm kiếm, xem thông tin về địa điểm và chỉ đường tới địa điểm.
* Thành viên: Nhóm này có các vai trò của nhóm khách và thêm vai trò của thành viên ( đánh giá, nhận xét về địa điểm, quản lý danh sách địa điểm của mình ).

### **3.1.2 Các yêu cầu ứng dụng đối với người dùng**

* Vai trò quản lý địa điểm: đăng nhập để xác định quyền admin,thêm, sửa, xóa địa điểm.
  + Quản lý danh mục kiểu địa điểm: Thêm, sửa, xóa dah mục kiểu địa điểm.
  + Quản lý danh mục món ăn: Thêm, sửa, xóa danh mục món ăn.
  + Quản lý món ăn: Thêm, sửa, xóa món ăn.
  + Quản lý ảnh: Thêm ảnh của địa điểm.
  + Quản lý bình luận: xem, xóa bình luận.
  + vai trò sử dụng hệ thống để tìm kiếm và xem thông tin về địa điểm : tìm kiếm địa điểm theo từ khóa, theo bán kính.
  + Hiển thị danh sách địa điểm theo kiểu địa điểm hoặc trong bán kish 5 km từ vị trí người dùng hoặc mới nhất.
  + Xem thông tin về địa điểm : Xem thông tin chi tiết của địa điểm, xem bình luận, xem đánh giá, xem ảnh, xem thực đơn.
  + Xem chỉ đường tới địa điểm : Chỉ đường từ vị trí người dùng tới địa điểm.
  + Thành viên : đăng nhập để xác định quyền thành viên, đăng xuất để xóa quyền thành viên.
  + Đánh giá : đánh giá về địa điểm thông qua số sao/ 5 sao.
  + Nhận xét : bình luận về địa điểm.
  + Quản lý danh sách địa điểm của mình : lưu địa điểm, xóa địa điểm đã lưu.

## **3.2 Phân tích và đặc tả yêu cầu**

### **Đặc tả yêu cầu**

**1. Đối với khách truy cập vào ứng dụng :**

**Đăng nhập**

|  |  |
| --- | --- |
| Mô tả | Để sử dụng được chức năng này, người dùng phải có tài khoản facebook. |
| Đầu vào | Tài khoản, mật khẩu facebook. |
| Xử lý | Người dùng đăng nhập vào ứng dụng thông qua 1 tài khoản facebook, sau đó facebook sẽ trả về một access\_token, từ access\_token đó có thể lấy được một số thông tin của người dùng.  Kiểm tra dữ liệu được trả về sau khi người dùng đăng nhập thông qua facebook. Lưu lại thông tin người dùng vào cơ sở dữ liệu ( facebook id, facebook name, link avatar, link ảnh bìa ( nếu có ), ). |
| Đầu ra | Đăng nhập thành công. |

**Tìm kiếm địa điểm ăn uống theo bán kính ( tính từ vị trí người dùng )**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng tìm kiếm các địa điểm ăn uống xung quanh vị trí của mình theo bán kính được chọn (250m, 500m, 1km, 2km, 5km) và từ khóa do người dùng nhập. |
| Đầu vào | Bán kính được chọn, từ khóa tìm kiếm, vị trí người dùng hiện tại. |
| Xử lý | Lấy danh sách địa điểm có tên phù hợp với từ khóa và có khoảng cách đến vị trí người dùng không lớn hơn bán kính. |
| Đầu ra | Hiển thị danh sách tìm kiếm. |

**Tìm kiếm địa điểm ăn uống theo dữ liệu nhập vào**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng tìm kiếm địa điểm có tên phù hợp với dữ liệu do người dùng nhập vào. |
| Đầu vào | Dữ liệu cần tìm kiếm do người dùng nhập vào ( là một chuỗi có thể gồm chữ, số, kí tự đặc biệt ). |
| Xử lý | Lấy danh sách địa điểm có tên phù hợp với dữ liệu người dùng nhập vào. |
| Đầu ra | Hiển thị danh sách kết quả tìm kiếm. |

**Xem chi tiết địa điểm**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng xem được thông tin của địa điểm. |
| Đầu vào | Địa điểm mà người dùng đã chọn. |
| Xử lý | Lấy trong cơ sở dữ liệu thông tin địa điểm mà người dùng chọn |
| Đầu ra | Hiển thị thông tin của địa điểm. |

**Xem chỉ đường tới địa điểm**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng xem được đường đi từ vị trí hiện tại đến địa điểm đã chọn. |
| Đầu vào | Địa điểm đã chọn và vị trí hiện tại của người dùng. |
| Xử lý | Gửi vị trí hiện tại của người dùng và vị trí của địa điểm ( latitude, longitude ) lên api của google map.  Google map nhận được dữ liệu gửi lên sẽ trả về dữ liệu để chỉ đường đi từ vị trí hiện tại tới địa điểm đã chọn.  Client nhận được dữ chỉ đường sẽ vẽ trên bản đồ một đường màu đỏ từ vị trí người dùng đến địa điểm đã chọn. |
| Đầu ra | Hiển thị chỉ đường từ vị trí người dùng tới địa điểm. |

**Xem toàn bộ bình luận về địa điểm**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng xem được các bình luận về địa điểm. |
| Đầu vào | Địa điểm người dùng đã chọn. |
| Xử lý | Lấy trong cơ sở dữ liệu danh sách bình luận của địa điểm đã chọn. |
| Đầu ra | Hiển thị danh sách bình luận của địa điểm đã chọn. |

**Xem thực đơn của quán**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng xem được thực đơn của quán đã chọn. |
| Đầu vào | Địa điểm đã chọn. |
| Xử lý | Lấy trong cơ sở dữ liệu danh sách các kiểu món ăn và món ăn của địa điểm. |
| Đầu ra | Hiển thị thực đơn của địa điểm. |

**Xem ảnh của quán**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng xem được các ảnh của địa điểm đã chọn. |
| Đầu vào | Địa điểm đã chọn. |
| Xử lý | Lấy trong cơ sở dữ liệu danh sách ảnh của địa điểm |
| Đầu ra | Hiển thị danh sách ảnh của địa điểm đã chọn. |

**Hiển thị danh sách địa điểm theo kiểu địa điểm hoặc trong bán kính 5 km từ vị trí người dùng hoặc mới nhất**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng chọn danh sách hiển thị địa điểm theo kiểu địa điểm hoặc trong bán kính 5 km từ vị trí người dùng hoặc mới nhất. |
| Đầu vào | Kiểu hiển thị mà người dùng muốn. |
| Xử lý | Lấy trong cơ sở dữ liệu danh sách địa điểm phù hợp với kiểu hiển thị mà người dùng chọn. |
| Đầu ra | Danh sách địa điểm hiển thị theo sự lựa chọn của người dùng. |

1. **Đối với thành viên ( người có tài khoản đăng nhâp vào ứng dụng ) sẽ có đầy đủ chức năng của khách và thêm các chức năng sau**

**Bình luận địa điểm**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép thành viên bình luận về một địa điểm |
| Đầu vào | Nội dung bình luận, thành viên và địa điểm |
| Xử lý | Thêm bình luận của thành viên vào cơ sở dữ liệu và cập nhật lại danh sách bình luận. |
| Đầu ra | Hiển thị bình luận của thành viên. |

**Đánh giá địa điểm**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép thành viên đánh giá về địa điểm thông qua số ngôi sao. |
| Đầu vào | Mức đánh giá, thành viên, địa điểm. |
| Xử lý | Thêm đánh giá của thành viên vào cơ sở dữ liệu và cập nhật lại đánh giá của địa điểm. |
| Đầu ra | Hiển thị đánh giá của người dùng và đánh giá trung bình về địa điểm đó. |

**Sửa đánh giá**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép người dùng sửa đánh giá về địa điểm. |
| Đầu vào | Mức đánh giá. |
| Xử lý | Cập nhật lại đánh giá của thành viên về địa điểm trong cơ sở dữ liệu. |
| Đầu ra | Hiển thị đánh giá của người dùng và đánh giá trung bình về địa điểm đó. |

**Lưu địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép thành viên lưu lại các địa điểm muốn lưu. |
| Đầu vào | Địa điểm, thành viên. |
| Xử lý | Thêm địa điểm vào danh sách địa điểm đã lưu của thành viên. |
| Đầu ra | Hiển thị danh sách địa điểm đã lưu của thành viên. |

**Xem lại các địa điểm đã lưu**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép thành viên xem lại các địa điểm mà họ đã lưu lại. |
| Đầu vào | Thành viên. |
| Xử lý | Lấy trong cơ sở dữ liệu danh sách địa điểm đã lưu của thành viên. |
| Đầu ra | Hiển thị danh sách địa điểm đã lưu của thành viên. |

**Xóa địa điểm đã lưu**

|  |  |
| --- | --- |
| Mô tả | Chức năng cho phép thành viên xóa địa điểm đã lưu trong danh sách địa điểm đã lưu của mình. |
| Đầu vào | Địa điểm, thành viên. |
| Xử lý | Xóa địa điểm trong danh sách địa điểm đã lưu của người dùng trong cơ sở dữ liệu. |
| Đầu ra | Danh sách địa điểm đã lưu sau khi xóa. |

**Xem thông tin cá nhân**

|  |  |
| --- | --- |
| Mô tả | Cho phép thành viên xem thông tin cá nhân của mình. |
| Đầu vào | Thành viên. |
| Xử lý | Lấy thông tin của thành viên trong cơ sở dữ liệu. |
| Đầu ra | Hiển thị thông tin cá nhân của thành viên đó. |

**Đăng xuất**

|  |  |
| --- | --- |
| Mô tả | Cho phép thành viên đăng xuất khỏi ứng dụng. |
| Đầu vào | Thành viên chọn đăng xuất. |
| Xử lý | Xóa phiên đăng nhập. |
| Đầu ra | Đăng xuất thành viên khỏi ứng dụng. |

1. **Đối với quản trị**

**Đăng nhập**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị đăng nhập vào hệ thống quản trị.  Để sử dụng chức năng này, người dùng cần có một tài khoản có quyền admin . |
| Đầu vào | Tài khoản và mật khẩu. |
| Xử lý | Kiểm tra dữ liệu tài khoản và mật khẩu trong cơ sở dữ liệu |
| Đầu ra | Nếu thông tin đăng nhập đúng thì cho phép đăng nhập vào trang quản trị, ngược lại thông báo lỗi |

**Đăng xuất**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị đăng xuất khỏi hệ thống quản trị. |
| Đầu vào | Người quản trị chọn đăng xuất. |
| Xử lý | Xóa phiên đăng nhập và đăng xuất khỏi hệ thống. |
| Đầu ra | Đăng xuất thành công và hiển thị màn hình đăng nhập. |

**Quản lý địa điểm**

**Thêm thông tin địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị thêm thông tin địa điểm mới vào hệ thống |
| Đầu vào | Thông tin địa điểm muốn thêm. |
| Xử lý | Kiểm tra và thêm địa điểm mới với thông tin do người quản trị đã nhập vào cơ sở dữ liệu. |
| Đầu ra | Nếu thêm thành công thì thông báo thêm thành công và cập nhật lại danh sách địa điểm hoặc thông báo lỗi nếu có lỗi. |

**Sửa thông tin địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị sửa lại thông tin của các địa điểm trong hệ thống |
| Đầu vào | Địa điểm, thông tin sửa. |
| Xử lý | Kiểm tra và cập nhật thông tin cho địa điểm với thông tin do người quản trị đã nhập vào cơ sở dữ liệu. |
| Đầu ra | Nếu sửa thành công thì thông báo sửa thành công và cập nhật lại danh sách địa điểm hoặc thông báo lỗi nếu có lỗi. |

**Xóa địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xóa một địa điểm muốn xóa. |
| Đầu vào | Địa điểm. |
| Xử lý | Xóa địa điểm đã chọn trong cơ sở dữ liệu. |
| Đầu ra | Thông báo xóa thành công và cập nhật lại danh sách địa điểm. |

**Quản lý bình luận**

**Xem bình luận**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xem bình luận của địa điểm đã chọn. |
| Đầu vào | Địa điểm. |
| Xử lý | Lấy danh sách bình luận của địa điểm trong cơ sở dữ liệu. |
| Đầu ra | Hiển thị danh sách bình luận của địa điểm. |

**3.3.2. Xóa bình luận**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xóa bình luận. |
| Đầu vào | Bình luận. |
| Xử lý | Xóa bình luận đã chọn trong cơ sở dữ liệu. |
| Đầu ra | Thông báo xóa thành công và cập nhật lại danh sách bình luận. |

**Quản lý danh mục món ăn**

**Thêm danh mục món ăn**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị thêm danh mục món ăn mới vào hệ thống. |
| Đầu vào | Thông tin danh mục muốn thêm. |
| Xử lý | Kiểm tra và thêm thông tin danh mục món ăn mới vào cơ sở dữ liệu. |
| Đầu ra | Nếu thêm thành công thì thông báo thêm thành công và cập nhật lại danh sách danh mục món ăn hoặc báo lỗi nếu có. |

**Sửa thông tin danh mục món ăn**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị sửa lại thông tin của các danh mục món ăn. |
| Đầu vào | Danh mục, thông tin sửa. |
| Xử lý | Kiểm tra và cập nhật lại thông tin cho danh mục món ăn. |
| Đầu ra | Nếu sửa thành công thì thông báo sửa thành công và cập nhật lại danh sách hiển thị hoặc thông báo lỗi nếu có. |

**Xóa danh mục**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xóa một danh mục món ăn. |
| Đầu vào | Danh mục món ăn. |
| Xử lý | Xóa danh mục món ăn trong cơ sở dữ liệu. |
| Đầu ra | Thông báo xóa thành công và cập nhật lại danh sách danh mục. |

**Quản lý món ăn**

**Thêm món ăn**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị thêm món ăn mới cho địa điểm. |
| Đầu vào | Thông tin món ăn, địa điểm, danh mục món ăn. |
| Xử lý | Kiểm tra và thêm thông tin món ăn mới vào cơ sở dữ liệu. |
| Đầu ra | Nếu thêm thành công thì thông báo thêm thành công và cập nhật lại danh sách món ăn hoặc thông báo lỗi nếu có. |

**Sửa thông tin món ăn**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị sửa lại thông tin của các món ăn. |
| Đầu vào | Món ăn, thông tin sửa. |
| Xử lý | Kiểm tra và cập nhật thông tin cho món ăn trong cơ sở dữ liệu. |
| Đầu ra | Nếu sửa thành công thì thông báo sửa thành công và cập nhật lại danh sách hiển thị hoặc thông báo lỗi nếu có. |

**Xóa món ăn**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xóa một món ăn. |
| Đầu vào | Món ăn. |
| Xử lý | Xóa dữ liệu món ăn trong cơ sở dữ liệu. |
| Đầu ra | Thông báo xóa thành công và cập nhật lại danh sách món ăn. |

**Quản lý ảnh**

**Thêm ảnh**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị thêm ảnh mới cho địa điểm. |
| Đầu vào | Ảnh, địa điểm. |
| Xử lý | Thêm ảnh vào danh sách ảnh của địa điểm trong cơ sở dữ liệu |
| Đầu ra | Nếu thêm thành công thì thông báo thêm thành công, cập nhật lại danh sách hiển thị hoặc thông báo lỗi nếu có. |

**Quản lý danh mục kiểu địa điểm**

**Thêm danh mục kiểu địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị thêm danh mục kiểu địa điểm mới. |
| Đầu vào | Thông tin kiểu địa điểm muốn thêm. |
| Xử lý | Kiểm tra và thêm thông tin danh mục kiểu địa điểm mới vào cơ sở dữ liệu. |
| Đầu ra | Nếu thêm thành công thì thông báo thêm thành công và cập nhật lại danh sách kiểu địa điểm hoặc thông báo lỗi nếu có. |

**Sửa thông tin danh mục kiểu địa điểm**

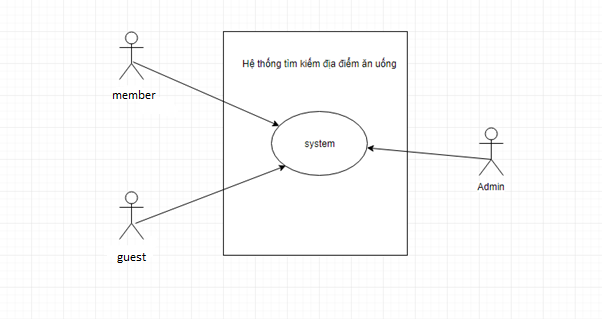
|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị sửa lại thông tin của kiểu địa điểm. |
| Đầu vào | Danh mục kiểu địa điểm, thông tin sửa. |
| Xử lý | Kiểm tra và cập nhật thông tin cho địa điểm trong cơ sở dữ liệu. |
| Đầu ra | Nếu sửa thành công thì thông báo sửa thành công và cập nhật lại danh sách hiển thị hoặc thông báo lỗi nếu có. |

**Xóa danh mục kiểu địa điểm**

|  |  |
| --- | --- |
| Mô tả | Cho phép người quản trị xóa danh mục kiểu địa điểm. |
| Đầu vào | Danh mục kiểu địa điểm. |
| Xử lý | Xóa danh mục kiểu địa điểm đã chọn trong cơ sở dữ liệu. |
| Đầu ra | Thông báo xóa thành công và cập nhật lại danh sách danh mục. |

### **3.2.2 Phân tích yêu cầu**

Usercase tổng quát



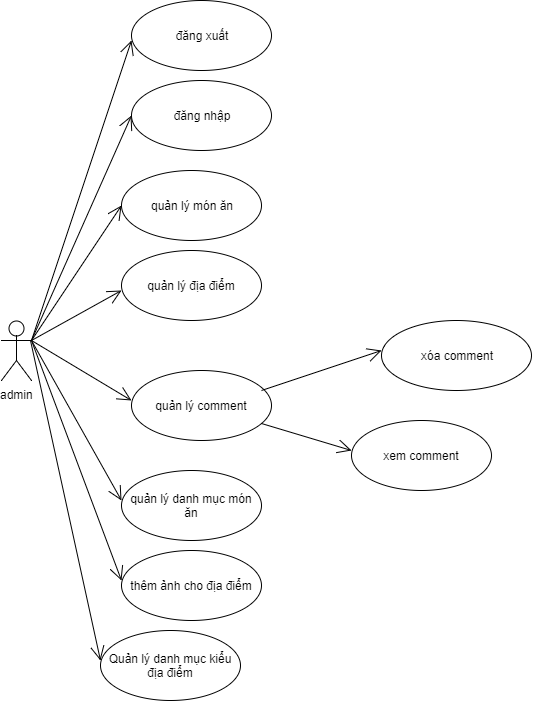
1. Phía người dùng

Usercase



1. Phía quản trị

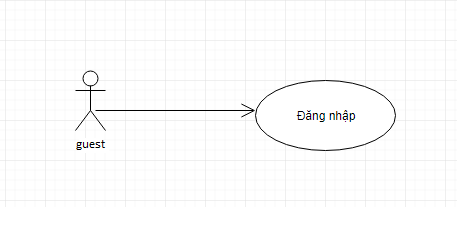
Usercase



#### **3.2.2.1 Chức năng của khách ( guest )**

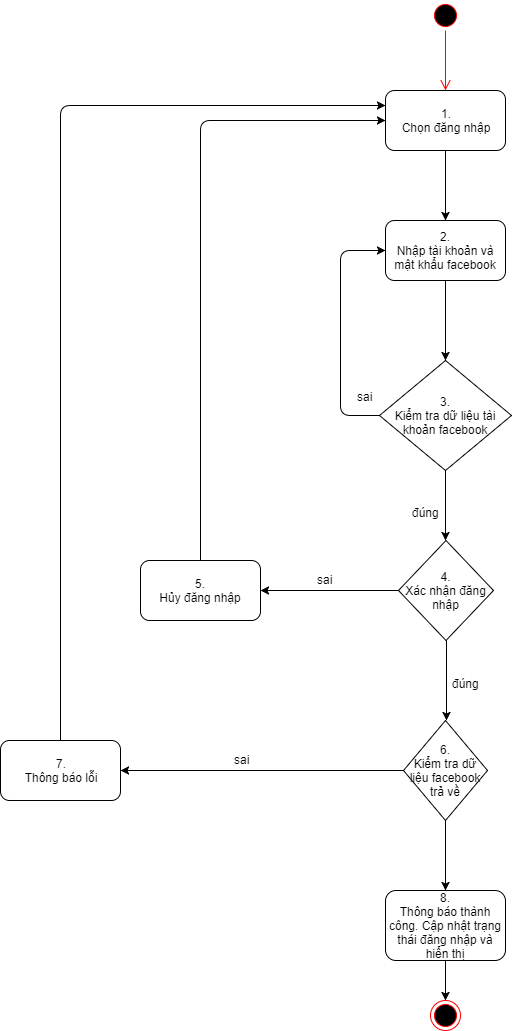
##### **3.2.2.1.1. Đăng nhập**

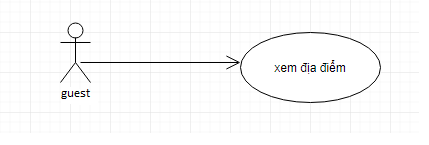
Mô tả: Chức năng cho phép người dùng đăng nhập vào hệ thống thông qua đăng nhập facebook



* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng chọn đăng nhập bằng facebook.
  + **Bước 2**: Nhập tài khoản và mật khẩu facebook.
  + **Bước 3**: Kiểm tra dữ liệu tài khoản facebook. Nếu đúng đến bước 4, nếu sai quay lại bước 2
  + **Bước 4**: Xác nhận đăng nhập, nếu đồng ý chuyển đến bước 6, nếu không chuyển đến bước 5.
  + **Bước 5**: Hủy việc đăng nhập.
  + **Bước 6**: Hệ thống kiểm tra dữ liệu facebook trả về, nếu đúng chuyển đến bước 8, nếu sai chuyển đến bước 7.
  + **Bước 7**: Thông báo lỗi.
  + **Bước 8**: Đăng nhập thành công, cập nhật lại trạng thái đăng nhập và trạng thái hiển thị.
  + **Bước 9**: Kết thúc.

Biểu đồ hoạt động

****

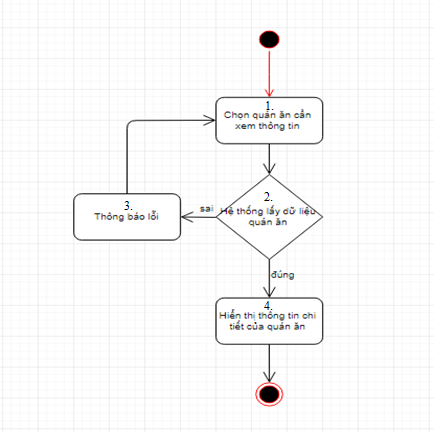


##### **3.2.2.1.2. Xem thông tin địa điểm**

Mô tả : Chức năng cho phép người dùng xem thông tin chi tiết của địa điểm

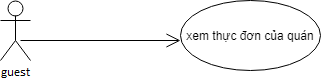
* **Bước 0**: Bắt đầu
* **Bước 1**: Người dùng chọn địa điểm cần xem thông tin
* **Bước 2**: Hệ thống lấy dữ liệu thông tin của địa điểm đã chọn, nếu lấy được thông tin thì chuyển đến bước 4, nếu không thì chuyển đến bước 3
* **Bước 3**: Hiển thị thông báo lỗi
* **Bước 4**: Hiển thị chi tiết thông tin địa điểm đã chọn (ảnh đại diện, tên, giá nhỏ nhất, giá lớn nhất, địa chỉ, tên tỉnh, quận, phường , đường, mức đánh giá, giờ mở cửa, giờ đóng cửa, khoảng cách từ vị trí người dùng đến địa điểm).
* **Bước 5**: Kết thúc

Biểu đồ hoạt động



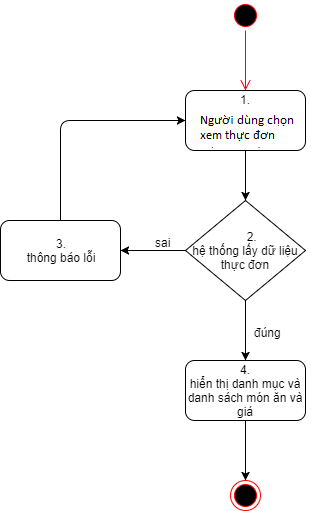
##### **Xem thực đơn của địa điểm**

Mô tả: Chức năng cho phép người dùng xem thực đơn của địa điểm dưới dạng danh sách ( tên món và giá tương ứng )



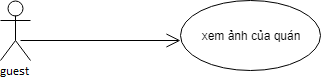
* **Bước 0**: Bắt đầu
* **Bước 1**: Người dùng chọn xem thực đơn của địa điểm.
* **Bước 2**: Client gửi yêu cầu lấy thực đơn của quán lên server và nhận kết quả trả về. Nếu lấy được thực đơn đến bước 4, nếu không đến bước 3
* **Bước 3**: Thông báo lỗi
* **Bước 4**: Hiển thị danh mục các món ăn cùng danh sách các món ăn thuộc danh mục của địa điểm dưới dạng danh sách, mỗi hàng là 1 món gồm tên và đơn giá.
* **Bước 5**: Kết thúc

Biểu đồ hoạt động



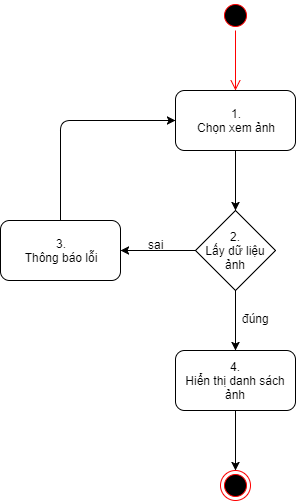
##### **Xem ảnh của địa điểm**

Mô tả: Chức năng cho phép người dùng xem được danh sách ảnh của địa điểm đã chọn



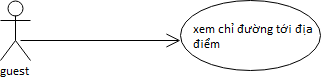
* **Bước 0**: Bắt đầu
* **Bước 1**: Người dùng chọn xem ảnh của địa điểm.
* **Bước 2**: Client gửi yêu cầu lấy ảnh của địa điểm lên server và nhận kết quả trả về. Nếu lấy được danh sách ảnh đến bước 4, nếu không đến bước 3.
* **Bước 3**: Thông báo lỗi.
* **Bước 4**: Hiển thị danh sách ảnh của địa điểm, mỗi hàng 3 ảnh.
* **Bước 5**: Kết thúc.

Biểu đồ hoạt động



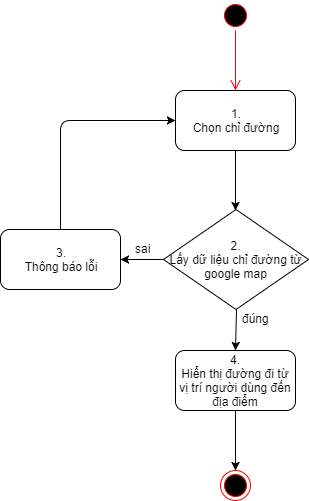
##### **Xem chỉ đường từ vị trí người dùng tới địa điểm**

Mô tả: Chức năng cho phép người dùng xem đường đi từ vị trí người dùng đến địa điểm.



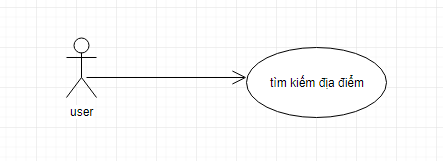
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng chọn chỉ đường tới địa điểm
  + **Bước 2**: Client gửi vị trí người dùng và vị trí địa điểm lên api của google map, nếu nhận được dữ liệu chỉ đường trả về thì chuyển đến bước 4, nếu nhận về dữ liệu lỗi đến bước 3
  + **Bước 3**: Thông báo lỗi
  + **Bước 4**: Hiển thị đường đi từ vị trí người dùng đến địa điểm trên bản đồ, đường đi được tô màu đỏ
  + **Bước 5**: Kết thúc

Biểu đồ hoạt động



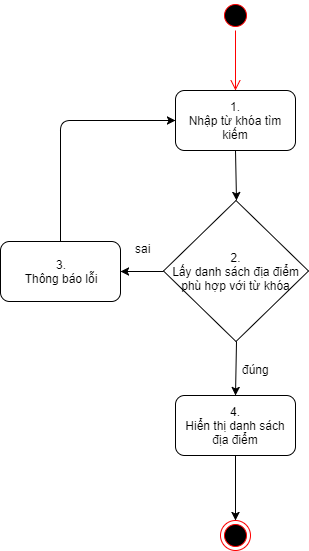
##### **Tìm kiếm địa điểm theo từ khóa**

Mô tả: Chức năng được sử dụng khi người dùng muốn tìm kiếm địa điểm theo tên địa điểm



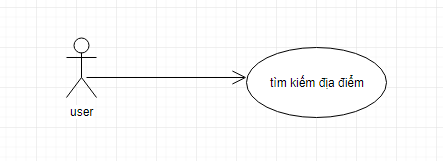
* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Người dùng nhập từ khóa tìm kiếm.
  + **Bước 2**: Hệ thống kiểm tra trong cơ sở dữ liệu và lấy ra những địa điểm có tên phù hợp với từ khóa người dùng đã nhập, nếu lấy được dữ liệu thì chuyển đến bước 4, nếu không thì chuyển đến bước 3.
  + **Bước 3**: Thông báo lỗi.
  + **Bước 4**: Hiển thị danh sách địa điểm.
  + **Bước 5**: Kết thúc.

Biểu đồ hoạt động



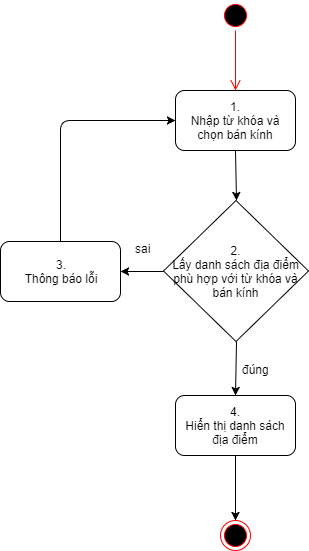
##### **Tìm kiếm địa điểm theo bán kính tính từ vị trí người dùng**

Mô tả: Chức năng cho phép người dùng tìm kiếm địa điểm nằm trong bán kính đã chọn tính từ vị trí người dùng



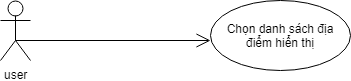
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng nhập từ khóa và chọn bán kính ( bán kính gồm 250m, 500m, 1km, 2km, 5km ).
  + **Bước 2**: Hệ thống kiểm tra trong cơ sở dữ liệu và lấy ra những địa điểm có tên phù hợp với từ khóa và có khoảng cách từ người dùng đến địa điểm không lớn hơn bán kính, nếu lấy được dữ liệu thì chuyển đến bước 5, nếu không thì chuyển đến bước 4
  + **Bước 3**: Thông báo lỗi
  + **Bước 4**: Hiển thị danh sách địa điểm.
  + **Bước 5**: Kết thúc

Biểu đồ hoạt động



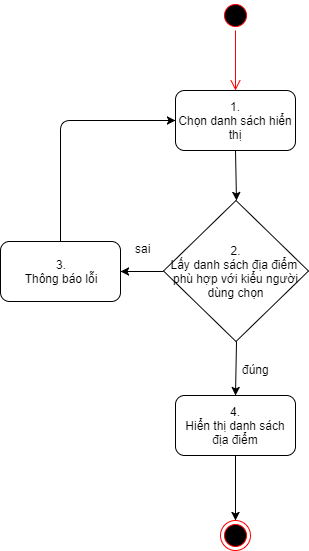
##### **Hiển thị danh sách địa điểm theo kiểu địa điểm hoặc trong khoảng cách bán kính 5 km từ vị trí người dùng hoặc danh sách mới nhất**

Mô tả: Chức năng cho phép người dùng chọn hiển thị danh sách địa điểm theo kiểu địa điểm hoặc trong bán kính 5 km từ vị trí người dùng hoặc danh sách mới nhất.



* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn danh sách hiển thị.
  + **Bước 2**: Hệ thống kiểm tra trong cơ sở dữ liệu và lấy ra những địa điểm phù hợp với kiểu hiển thị mà người dùng chọn, nếu lấy được dữ liệu thì đến bước 4, nếu có lỗi đến bước 3.
  + **Bước 3**: Thông báo lỗi.
  + **Bước 4**: Hiển thị danh sách địa điểm phù hợp với dữ liệu người dùng đã chọn.
  + **Bước 5**: Kết thúc.

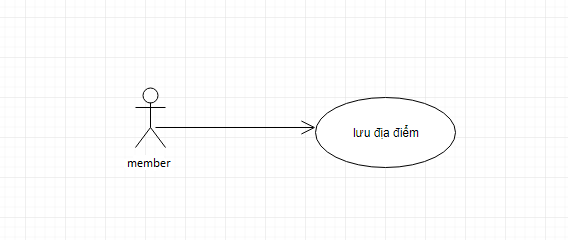
Biểu đồ hoạt động



#### **3.2.2.2 Chức năng của thành viên (member) ( người đăng nhập vào ứng dụng ) sẽ có tất cả các chức năng của khách ( guest) và thêm các chức năng sau.**

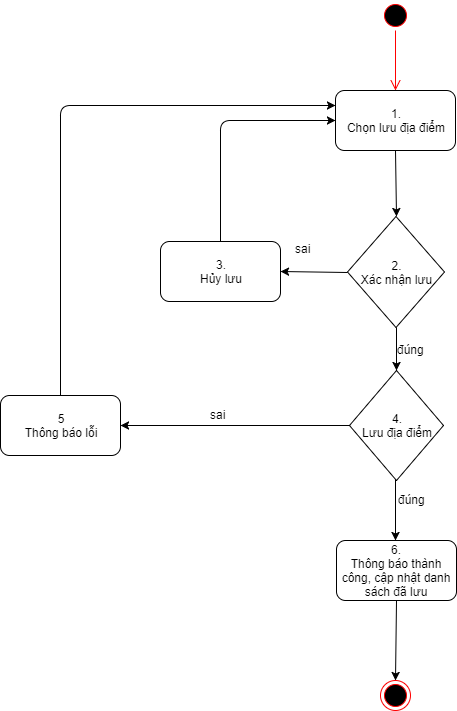
##### **3.2.2.2.1. Lưu địa điểm**

Mô tả: Chức năng cho phép thành viên lưu lại địa điểm ăn uống vào danh sách riêng của mình.



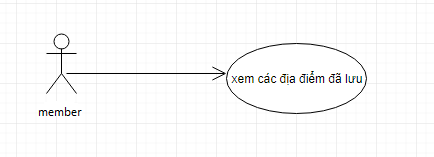
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn lưu địa điểm.
  + **Bước 2**: Xác nhận lưu. Nếu đồng ý đến bước 4, nếu không đến bước 3.
  + **Bước 3**: Hủy việc lưu
  + **Bước 4**: Hệ thống lưu lại địa điểm vào danh sách địa điểm đã lưu của thành viên trong cơ sở dữ liệu. Nếu lưu thành công chuyển đến bước 6, nếu không chuyển sang bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Thông báo thành công và cập nhật lại danh sách địa điểm đã lưu của thành viên.
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động



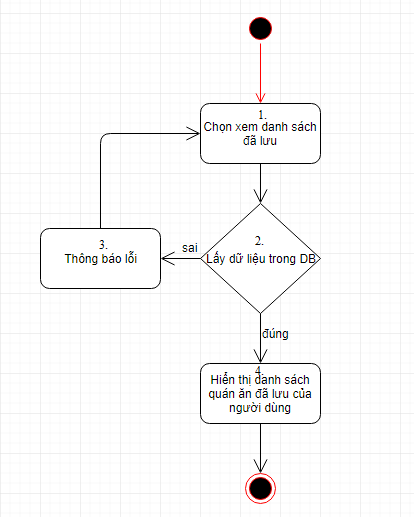
##### **Xem lại các địa điểm đã lưu**

Mô tả: Chức năng cho phép thành viên xem lại danh sách địa điểm đã lưu của mình



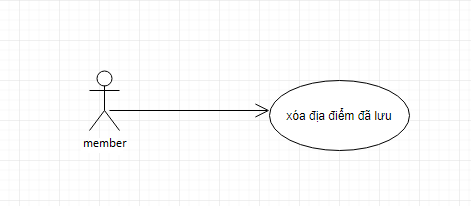
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn xem danh sách địa điểm đã lưu.
  + **Bước 2**: Hệ thống lấy danh sách địa điểm đã lưu của người dùng trong cơ sở dữ liệu, nếu đúng đến bước 4, nếu sai đến bước 3
  + **Bước 3**: Thông báo lỗi
  + **Bước 4**: Hiển thị danh sách địa điểm đã lưu của người dùng
  + **Bước 5**: Kết thúc

Biểu đồ hoạt động



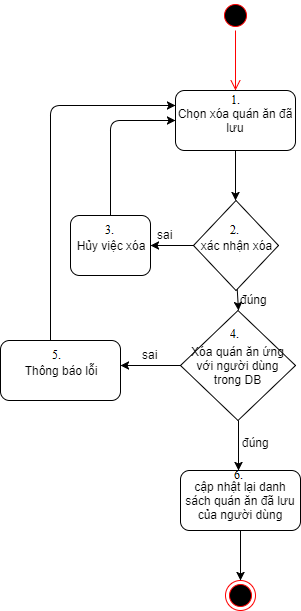
##### **Xóa địa điểm đã lưu**

Mô tả: Chức năng cho phép người dùng xóa đi địa điểm trong danh sách địa điểm đã lưu của mình



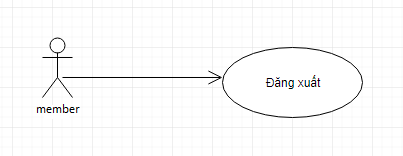
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn xóa địa điểm đã lưu.
  + **Bước 2**: Xác nhận xóa. Nếu đồng ý xóa đến bước 4, nếu không đến bước 3.
  + **Bước 3**: Hủy việc xóa.
  + **Bước 4**: Hệ thống xóa địa điểm trong danh sách địa điểm đã lưu của thành viên. Nếu xóa thành công thì đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Cập nhật lại danh sách địa điểm đã lưu
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động



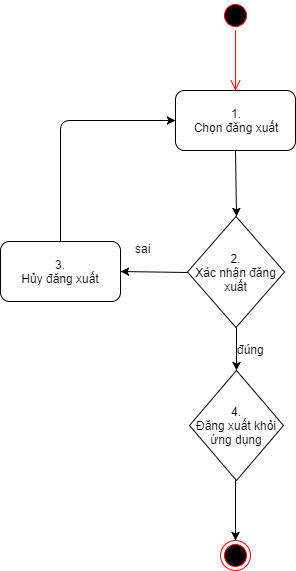
##### **Đăng xuất**

Mô tả: Chức năng cho phép người dùng đăng xuất khỏi hệ thống

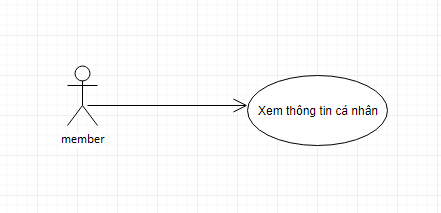


* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn đăng xuất
  + **Bước 2**: Xác nhận đăng xuất. Nếu đồng ý đến bước 4, nếu không đến bước 3.
  + **Bước 3**: Hủy việc đăng xuất.
  + **Bước 4**: Đăng xuất thành viên khỏi ứng dụng và cập nhật trạng thái hiển thị.
  + **Bước 5**: Kết thúc.

Biểu đồ hoạt động

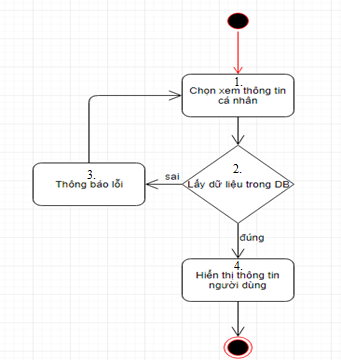


##### **Xem thông tin cá nhân**

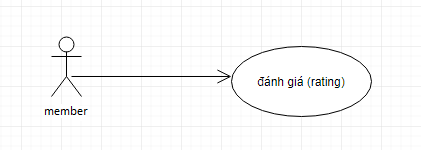


* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xem thông tin cá nhân.
  + **Bước 2**: Hệ thống lấy dữ liệu thành viên trong cơ sở dữ liệu, nếu đúng đến bước 4, sai đến bước 3.
  + **Bước 3**: Thông báo lỗi.
  + **Bước 4**: Hiển thị thông tin thành viên.
  + **Bước 5**: Kết thúc.

Biểu đồ hoạt động

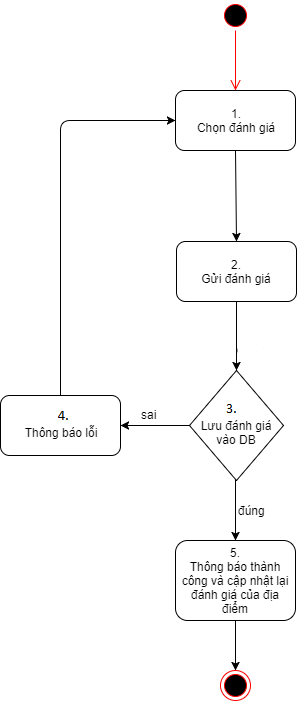


##### **Đánh giá**

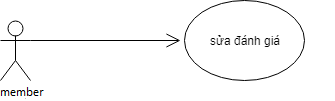


* + **Bước 0**: Bắt đầu
  + **Bước 1**: Đánh giá địa điểm.
  + **Bước 2**: Gửi đánh giá.
  + **Bước 3**: Lưu đánh giá của thành viên vào cơ sở dữ liệu, nếu đúng đến bước 5, nếu sai đến bước 4.
  + **Bước 4**: Thông báo lỗi
  + **Bước 5**: Thông báo thành công và cập nhật lại mức độ đánh giá của địa điểm
  + **Bước 6**: Kết thúc

Biểu đồ hoạt động

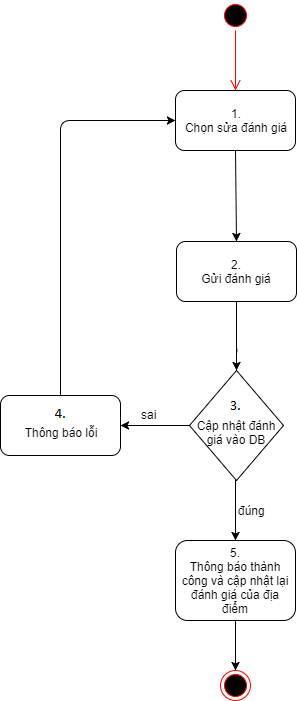


##### **Sửa đánh giá**

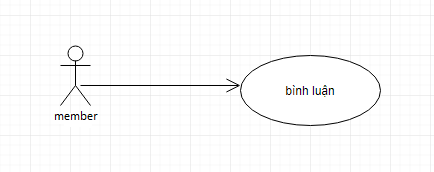


* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn sửa đánh giá.
  + **Bước 2**: Gửi đánh giá.
  + **Bước 3**: Cập nhật lại đánh giá của thành viên vào cơ sở dữ liệu, nếu đúng đến bước 5, nếu sai đến bước 4.
  + **Bước 4**: Thông báo lỗi
  + **Bước 5**: Thông báo thành công và cập nhật lại mức độ đánh giá của địa điểm**.**
  + **Bước 6**: Kết thúc

Biểu đồ hoạt động

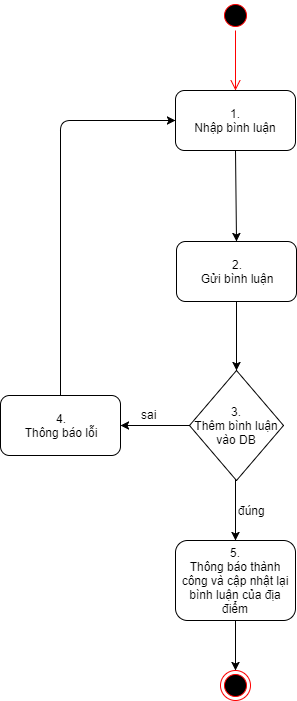


##### **Bình luận**

****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Nhập bình luận.
  + **Bước 2**: Gửi bình luận.
  + **Bước 3**: Thêm bình luận vào cơ sở dữ liệu, nếu thành công đến bước 5, không thành công đến bước 4
  + **Bước 4**: Thông báo lỗi
  + **Bước 5**: Cập nhật lại danh sách bình luận của địa điểm
  + **Bước 6**: Kết thúc

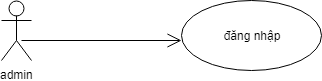
Biểu đồ hoạt động

****

#### **3.2.2.3 Chức năng của người quản trị (admin)**

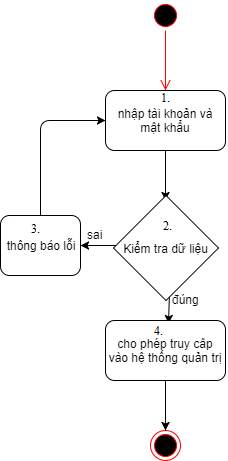
##### **3.2.2.3.1 Đăng nhập**

Mô tả: Đăng nhập vào hệ thống quản trị. Để truy cập vào trang quản trị, người dùng phải đăng nhập bằng tài khoản có quyền admin



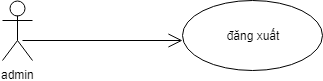
* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Nhập tài khoản và mật khẩu.
  + **Bước 2**: Kiểm tra dữ liệu tài khoản và mật khẩu. Nếu đúng đến bước 4, nếu không đến bước 3
  + **Bước 3**: Thông báo tài khoản không hợp lệ
  + **Bước 4**: Đăng nhập vào hệ thống quản trị và sử dụng các chức năng của người quản trị.
  + **Bước 5**: Kết thúc

Biểu đồ hoạt động



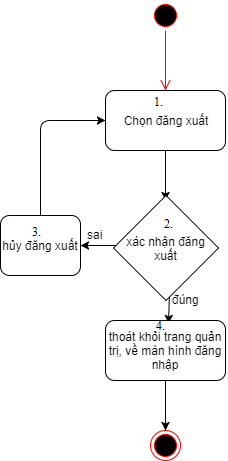
##### **3.2.2.3.2 Đăng xuất**

Mô tả: Cho phép người dùng thoát khỏi trang quản trị



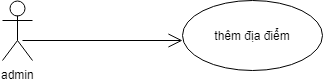
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Người dùng chọn đăng xuất
  + **Bước 2**: Xác nhận đăng xuất. Nếu đồng ý đăng xuất chuyển đến bước 4, nếu không chuyển đến bước 3
  + **Bước 3**: Hủy việc đăng xuất
  + **Bước 4**: Đăng xuất người dùng khỏi trang quản trị, trở về màn hình đăng nhập
  + **Bước 5**: Kết thúc

Biểu đồ hoạt động



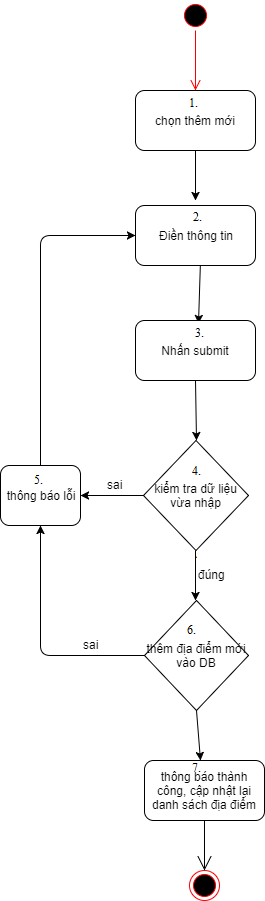
##### **3.2.2.3.3 Thêm địa điểm**

Mô tả: Cho phép người quản trị thêm mới một địa điểm



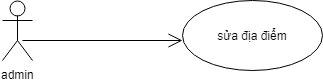
* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn thêm địa điểm.
  + **Bước 2**: Nhập thông tin địa điểm ( tên, tỉnh, quận, phường, đường, địa chỉ, kinh độ, vĩ độ, số điện thoại, giá thấp nhất, giá cao nhất, giờ mở cửa, giờ đóng cửa, ảnh đại diện ).
  + **Bước 3**: Gửi dữ liệu để thêm địa điểm.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Gửi dữ liệu lên server để thêm mới 1 địa điểm vào cơ sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo thêm dữ liệu thành công và cập nhật lại danh sách địa điểm
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động



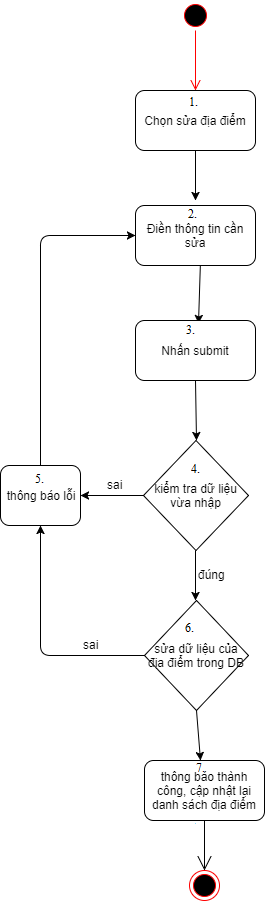
##### **3.2.2.3.4 Sửa thông tin địa điểm**

Mô tả: Cho phép người quản trị sửa thông tin của một địa điểm



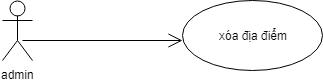
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn sửa địa điểm.
  + **Bước 2**: Nhập thông tin cần sửa.
  + **Bước 3**: Gửi thông tin muốn sửa.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Gửi dữ liệu lên server để sửa địa điểm đã chọn trong cơ sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo sửa thành công và cập nhật lại danh sách địa điểm
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động



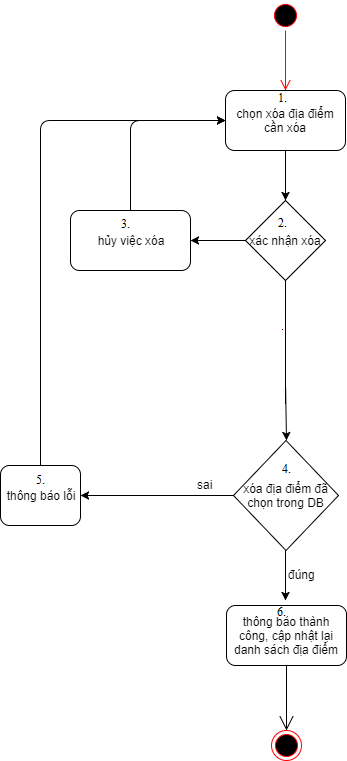
##### **3.2.2.3.5 Xóa địa điểm**

Mô tả: Cho phép người quản trị xóa một địa điểm khỏi cơ sở dữ liệu



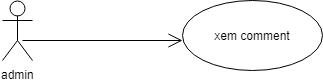
* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xóa địa điểm.
  + **Bước 2**: Xác nhận xóa. Nếu đồng ý đến bước 4, nếu không đến bước 3.
  + **Bước 3**: Hủy việc xóa.
  + **Bước 4**: Xóa địa điểm trong cơ sở dữ liệu. Nếu xóa thành công đến bước 6, nếu không đến bước 7.
  + **Bước 5**: Thông báo lỗi.
  + **Bước 6**: Thông báo xóa thành công và cập nhật lại danh sách địa điểm.
  + **Bước 7**: Kết thúc.

Biểu đồ hoạt động



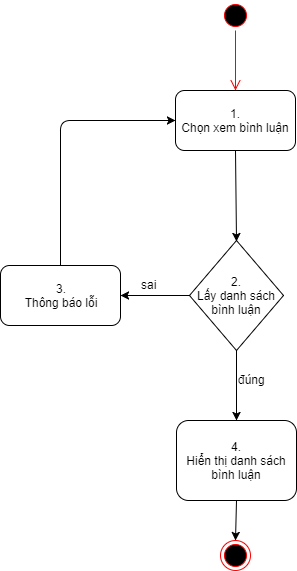
##### **3.2.2.3.6 Xem bình luận**

Mô tả: Cho phép người quản trị xem được danh sách các bình luận của địa điểm



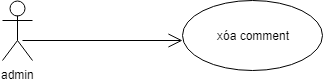
* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn xem bình luận.
  + **Bước 2**: Hệ thống lấy dữ liệu danh sách bình luận của địa điểm đã chọn. Nếu lấy được dữ liệu thành công chuyển đến bước4, ngược lại đến bước 3
  + **Bước 3**: Thông báo lỗi
  + **Bước 4**: Hiển thị danh sách bình luận của địa điểm đã chọn
  + **Bước 6**: Kết thúc

Biểu đồ hoạt động



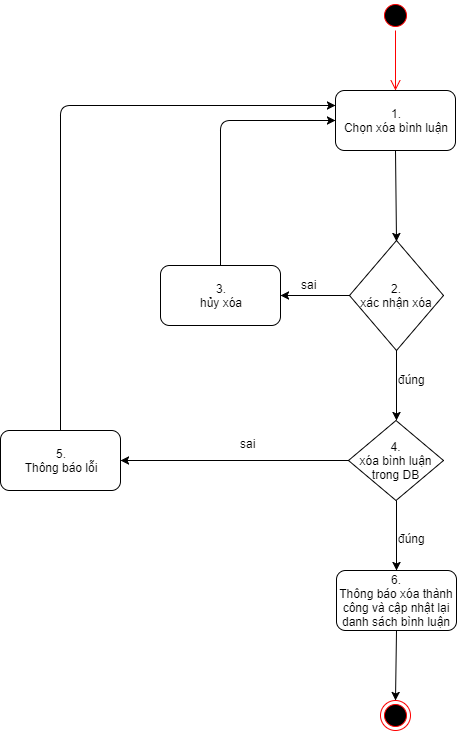
##### **3.2.2.3.7 Xóa bình luận**

Mô tả: Cho phép người quản trị xóa được bình luận của địa điểm



* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xóa bình luận.
  + **Bước 2**: Xác nhận xóa, nếu đồng ý xóa đến bước 3, nếu không đến bước 4.
  + **Bước 3**: hủy việc xóa.
  + **Bước 4**: Xóa bình luận đã chọn trong cơ sở dữ liệu. Nếu xóa thành công chuyển đến bước 6, ngược lại đến bước 5.
  + **Bước 5**: thông báo lỗi.
  + **Bước 6**: thông báo xóa thành công và cập nhật lại danh sách bình luận.
  + **Bước 7**: Kết thúc.

Biểu đồ hoạt động

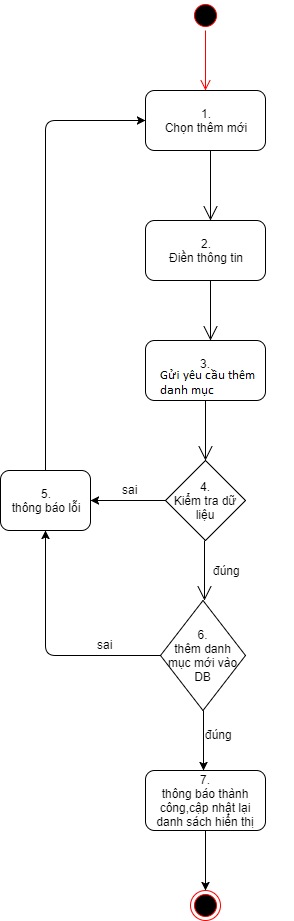


##### **3.2.2.3.8 Thêm danh mục món ăn**



* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn thêm danh mục món ăn.
  + **Bước 2**: Nhập thông tin danh mục (tên danh mục).
  + **Bước 3**: Gửi yêu cầu thêm danh mục với thông tin đã nhập.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5.
  + **Bước 5**: Thông báo lỗi.
  + **Bước 6**: Thêm danh mục mới vào cơ sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5.
  + **Bước 7**: Thông báo thêm dữ liệu thành công và cập nhật lại danh sách danh mục.
  + **Bước 8**: Kết thúc.

Biểu đồ hoạt động

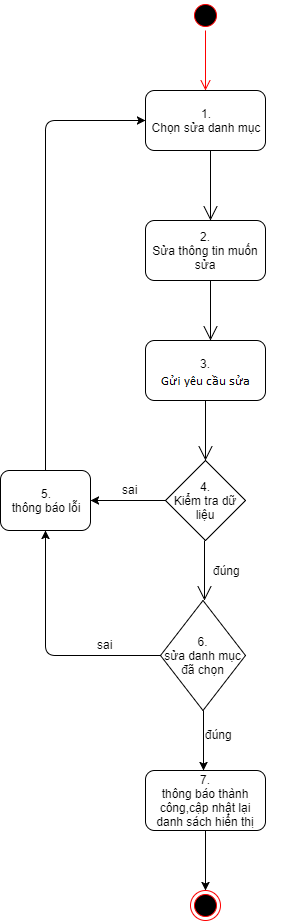


##### **3.2.2.3.9 Sửa danh mục món ăn**

****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn sửa danh mục.
  + **Bước 2**: Nhập thông tin cần sửa.
  + **Bước 3**: Gửi thông tin muốn sửa.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Cập nhật dữ liệu danh mục trong cơ sở dữ liệu. Nếu thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo sửa thành công và cập nhật lại danh sách danh mục
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động

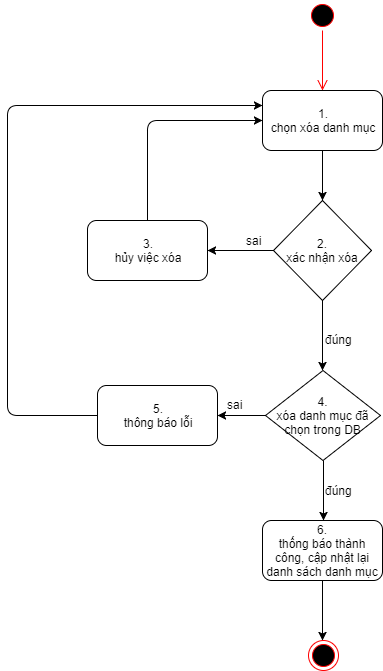


##### **3.2.2.3.10 Xóa danh mục món ăn**

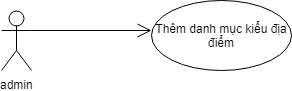
****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xóa danh mục.
  + **Bước 2**: Xác nhận xóa. Nếu đồng ý xóa chuyển đến bước 4, ngược lại chuyển đến bước 3
  + **Bước 3**: Hủy việc xóa
  + **Bước 4**: Xóa dữ liệu danh mục trong cơ sở dữ liệu. Nếu xóa thành công chuyển đến bước 6, ngược lại chuyển đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Thông báo xóa thành công và cập nhật lại danh sách danh mục
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động

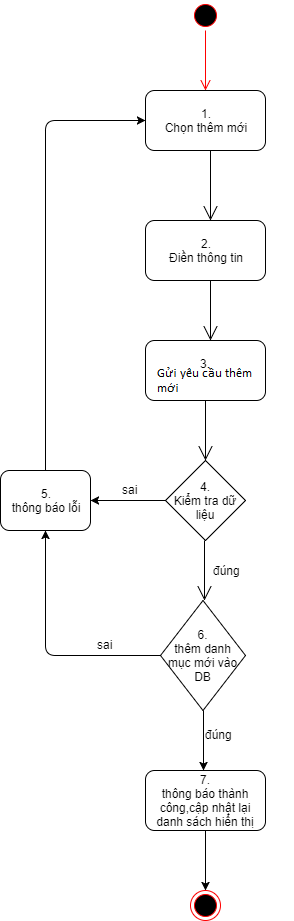


##### **3.2.2.3.11 Thêm danh mục kiểu địa điểm**

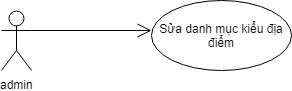


* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn thêm danh mục kiểu địa điểm.
  + **Bước 2**: Nhập thông tin danh mục kiểu địa điểm ( tên kiểu địa điểm ).
  + **Bước 3**: Gửi yêu cầu thêm danh mục kiểu địa điểm.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Gửi dữ liệu lên server để thêm mới 1 danh mục vào cơ sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo thêm dữ liệu thành công và cập nhật lại danh sách danh mục
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động

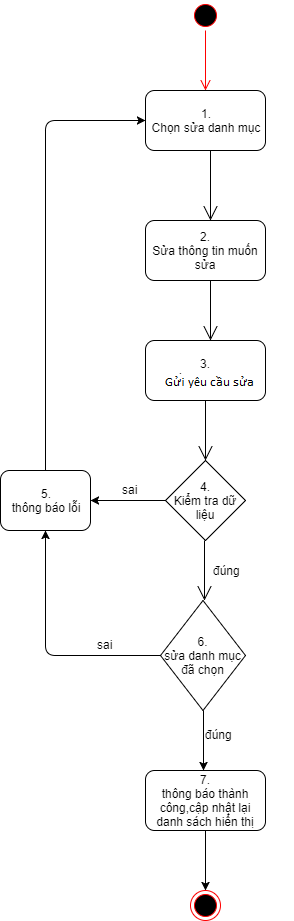


##### **3.2.2.3.12 Sửa danh mục kiểu địa điểm**

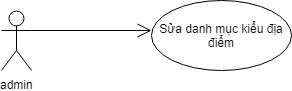
****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn sửa danh mục kiểu địa điểm.
  + **Bước 2**: Nhập thông tin cần sửa.
  + **Bước 3**: Gửi yêu cầu sửa.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Sửa dữ liệu danh mục kiểu địa điểm trong cơ sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo sửa thành công và cập nhật lại danh sách danh mục
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động

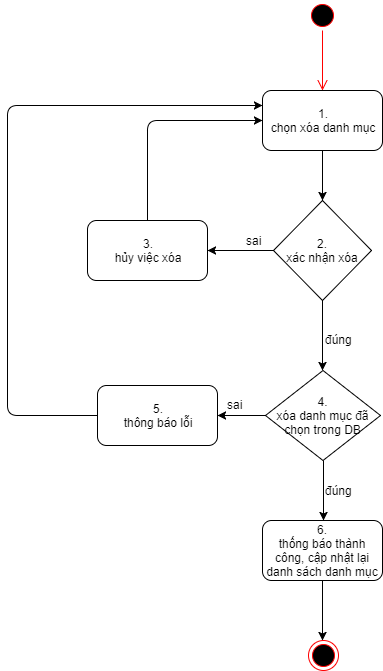


##### **3.2.2.3.13 Xóa danh mục kiểu địa điểm**

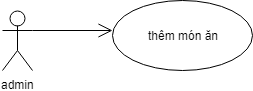
****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xóa danh mục kiểu địa điểm.
  + **Bước 2**: Xác nhận xóa. Nếu đồng ý xóa chuyển đến bước 4, ngược lại chuyển đến bước 3
  + **Bước 3**: Hủy việc xóa
  + **Bước 4**: Xóa danh mục kiểu địa điểm trong cơ sở dữ liệu. Nếu xóa thành công chuyển đến bước 6, ngược lại chuyển đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Thông báo xóa thành công và cập nhật lại danh sách danh mục
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động

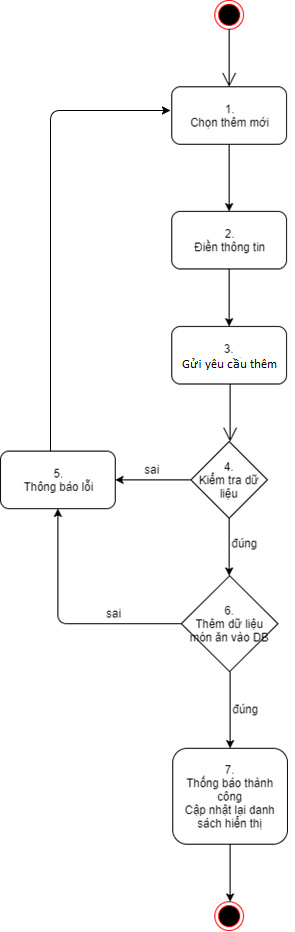


##### **3.2.2.3.14 Thêm món ăn**

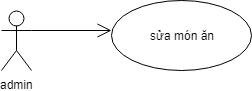
****

* + **Bước 0**: Bắt đầu
  + **Bước 1**: Chọn thêm món ăn.
  + **Bước 2**: Nhập thông tin cho món ăn (tên, thuộc danh mục đồ ăn, giá),
  + **Bước 3**: Gửi yêu cầu thêm món ăn.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Thêm món ăn mới vào cở sở dữ liệu. Nếu thêm thành công đến bước 7. Nếu không đến bước 5
  + **Bước 7**: Thông báo thêm dữ liệu thành công và cập nhật lại danh sách món ăn của địa điểm
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động

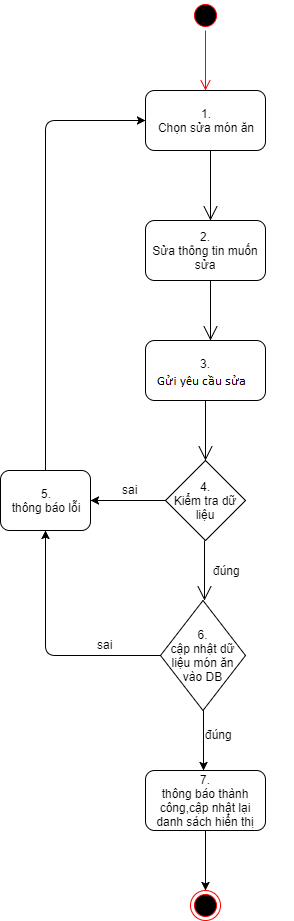


##### **3.2.2.3.15 Sửa món ăn**

****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn sửa món ăn.
  + **Bước 2**: Nhập thông tin muốn sửa.
  + **Bước 3**: Gửi yêu cầu sửa.
  + **Bước 4**: Hệ thống kiểm tra dữ liệu vừa nhập xem có hợp lệ không. Nếu có đến bước 6, nếu không đến bước 5.
  + **Bước 5**: Thông báo lỗi.
  + **Bước 6**: Cập nhật lại món ăn. Nếu sửa thành công đến bước 7. Nếu không đến bước 5.
  + **Bước 7**: Thông báo sửa thành công và cập nhật lại danh sách món ăn.
  + **Bước 8**: Kết thúc

Biểu đồ hoạt động

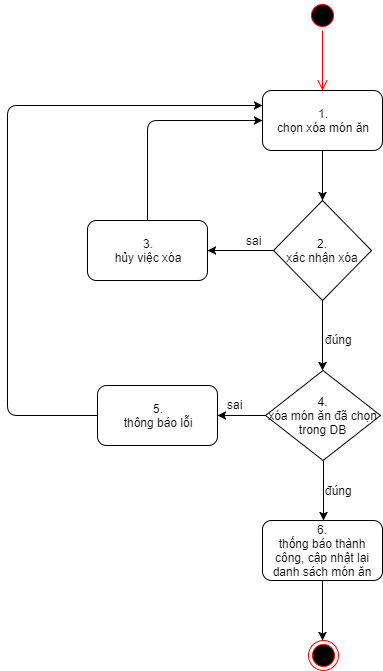


##### **3.2.2.3.16 xóa món ăn**

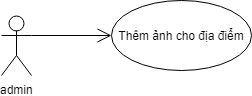
****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn xóa món ăn.
  + **Bước 2**: Xác nhận xóa. Nếu đồng ý xóa chuyển đến bước 4, ngược lại chuyển đến bước 3.
  + **Bước 3**: Hủy việc xóa.
  + **Bước 4**: Xóa món ăn đã chọn trong cơ sở dữ liệu. Nếu xóa thành công chuyển đến bước 6, ngược lại chuyển đến bước 5.
  + **Bước 5**: Thông báo lỗi
  + **Bước 6**: Thông báo xóa thành công và cập nhật lại danh sách món ăn
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động

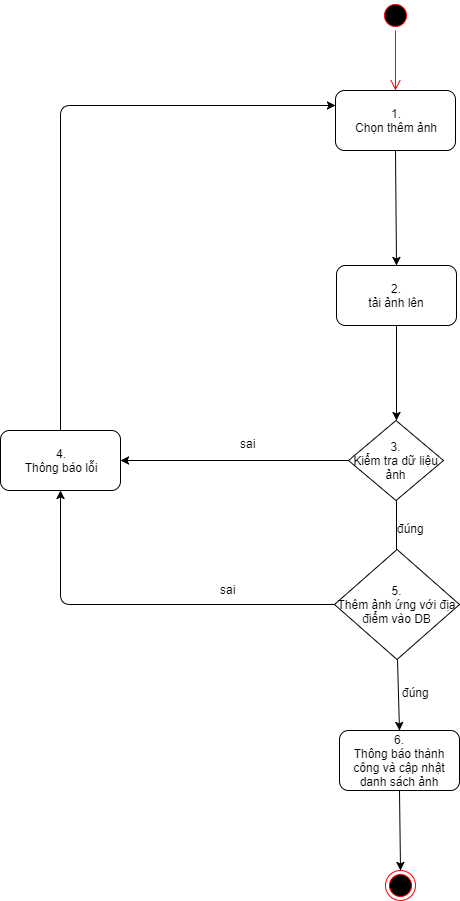


##### **3.2.2.3.17 Thêm ảnh cho địa điểm**

****

* + **Bước 0**: Bắt đầu.
  + **Bước 1**: Chọn thêm ảnh.
  + **Bước 2**: Tải ảnh lên.
  + **Bước 3**: Hệ thống kiểm tra dữ liệu ảnh xem có hợp lệ không. Nếu có đến bước 5, nếu không đến bước 4
  + **Bước 4**: Thông báo lỗi
  + **Bước 5**: Thêm ảnh đã chọn ứng với địa điểm vào cơ sở dữ liệu. Nếu thêm thành công đến bước 6. Nếu không đến bước 4
  + **Bước 6**: Thông báo thành công và cập nhật lại danh sách ảnh của địa điểm
  + **Bước 7**: Kết thúc

Biểu đồ hoạt động

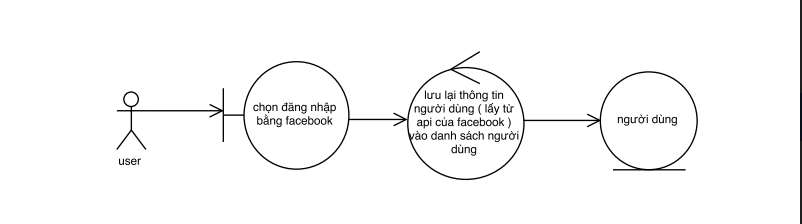
****

## **3.3 Thiết kế**

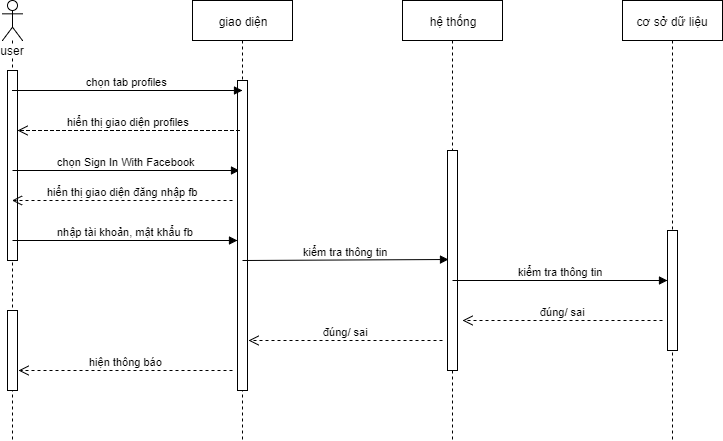
### **3.3.1 Các chức năng của khách ( guest )**

#### **3.3.1.1 Đăng nhập**

Biểu đồ lớp

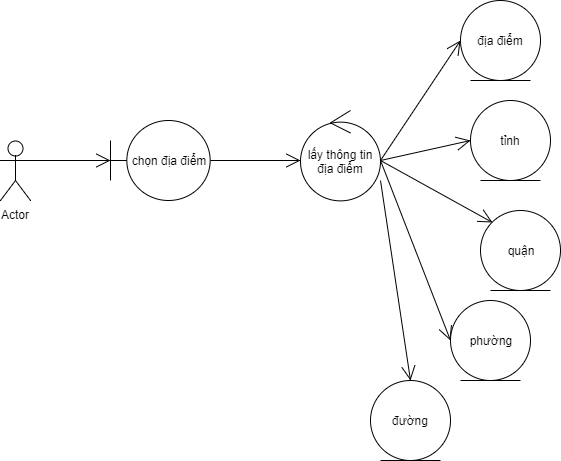


Biểu đồ tuần tự

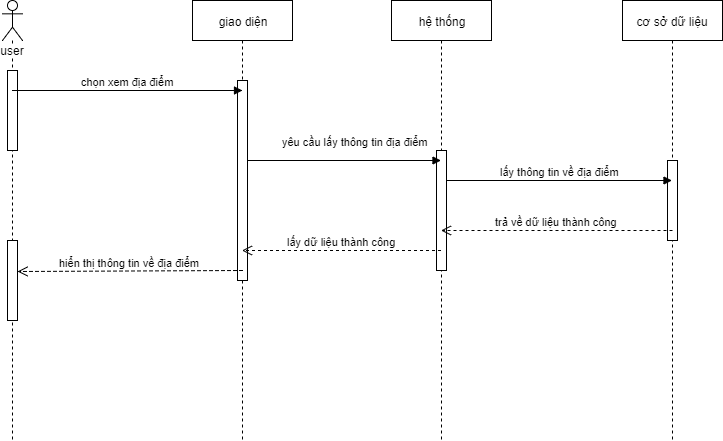


#### **3.3.1.2 Xem địa điểm**

Biểu đồ lớp

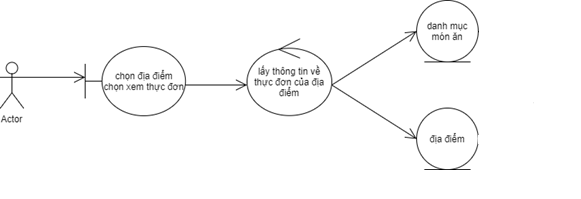


Biểu đồ tuần tự

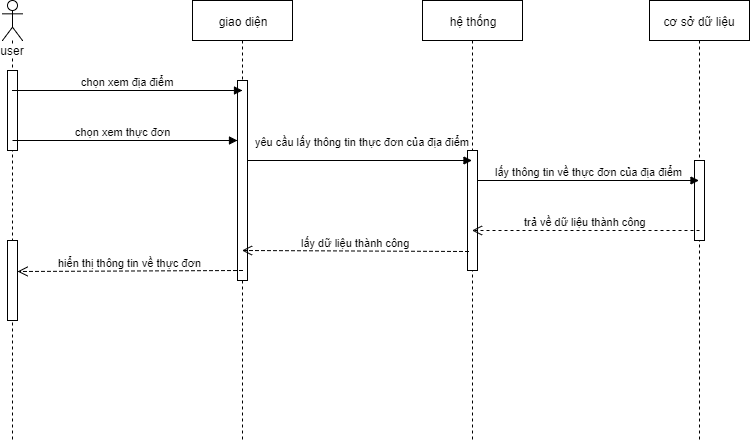


#### **3.3.1.3 Xem thực đơn**

Biểu đồ lớp

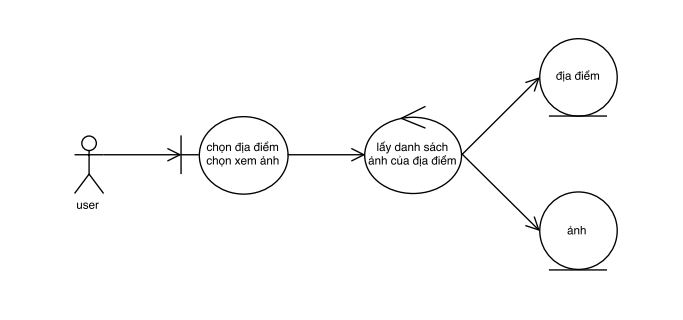


Biểu đồ tuần tự

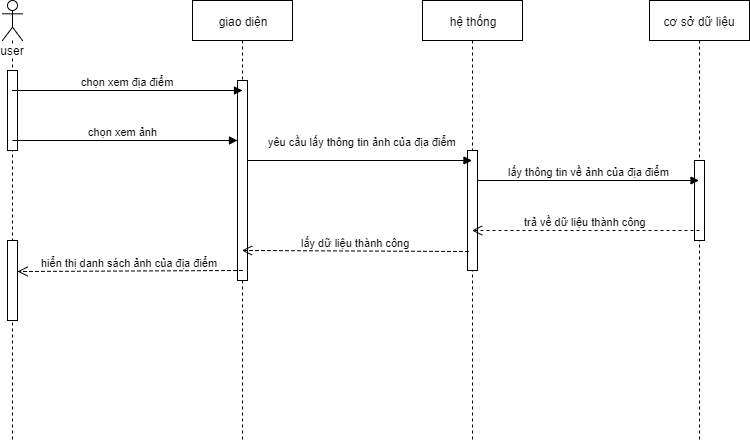
****

#### **3.3.1.4 Xem ảnh**

Biểu đồ lớp

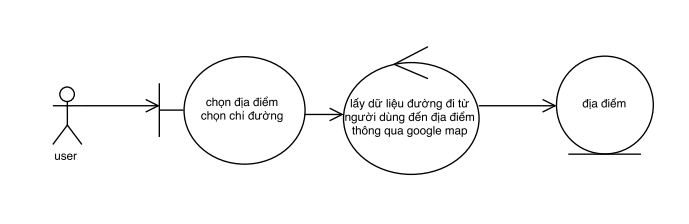


Biểu đồ tuần tự

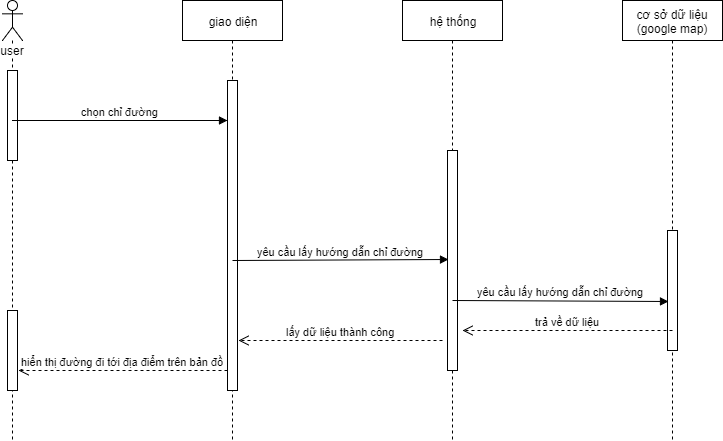
****

#### **3.3.1.5 Xem chỉ đường**

Biểu đồ lớp

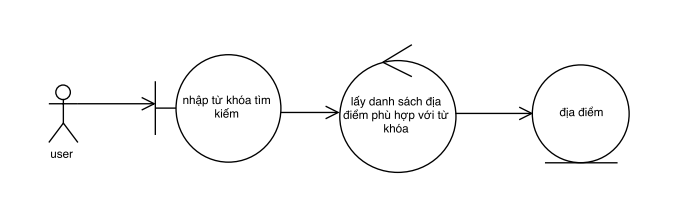


Biểu đồ tuần tự

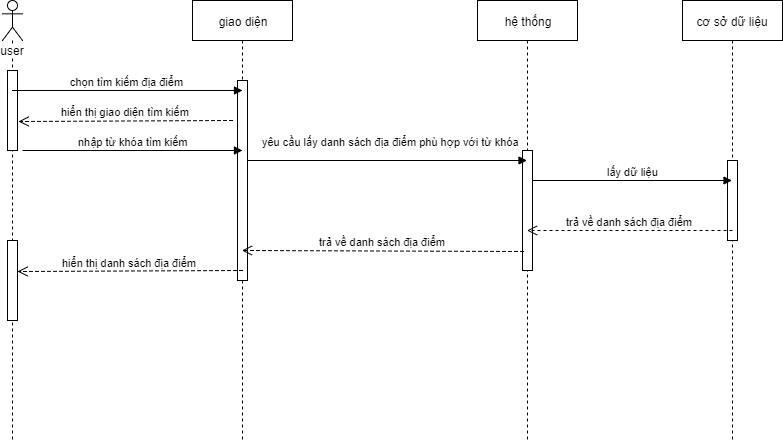
****

#### **3.3.1.6 Tìm kiếm theo tên**

Biểu đồ lớp

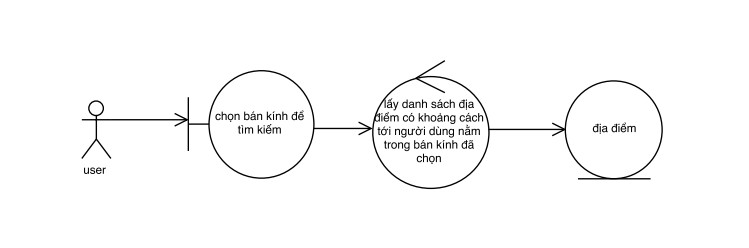


Biểu đồ tuần tự

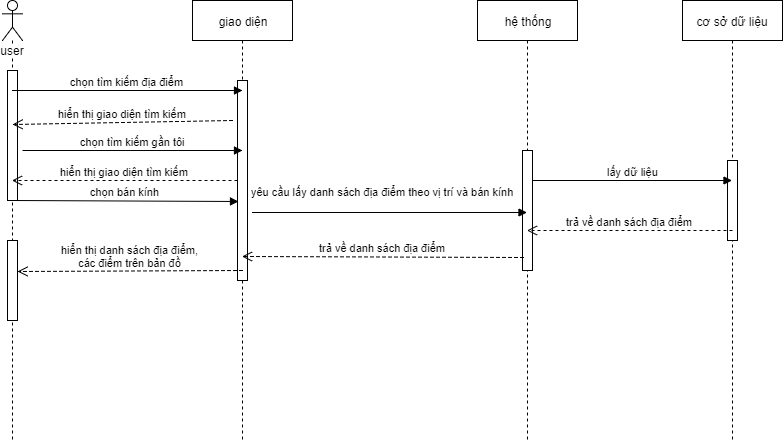
****

#### **3.3.1.7 Tìm kiếm theo bán kính**

Biểu đồ lớp

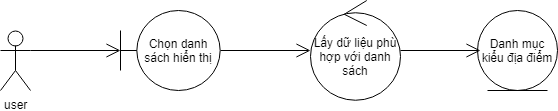


Biểu đồ tuần tự

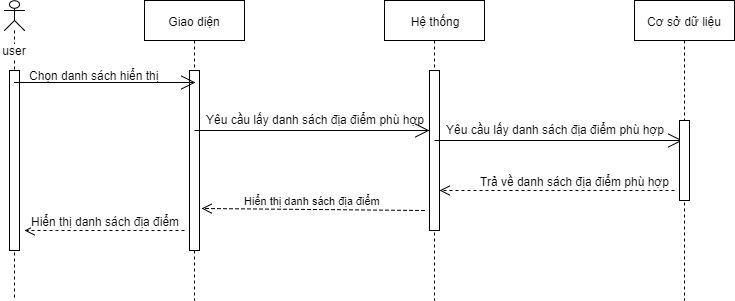
****

#### **3.3.1.8 Chọn danh sách địa điểm hiển thị**

Biểu đồ lớp

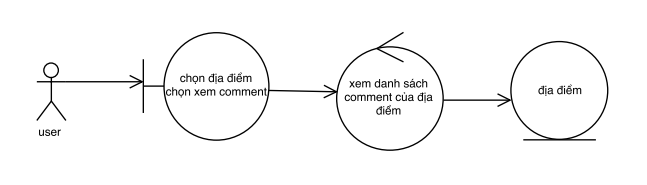
****

Biểu đồ tuần tự

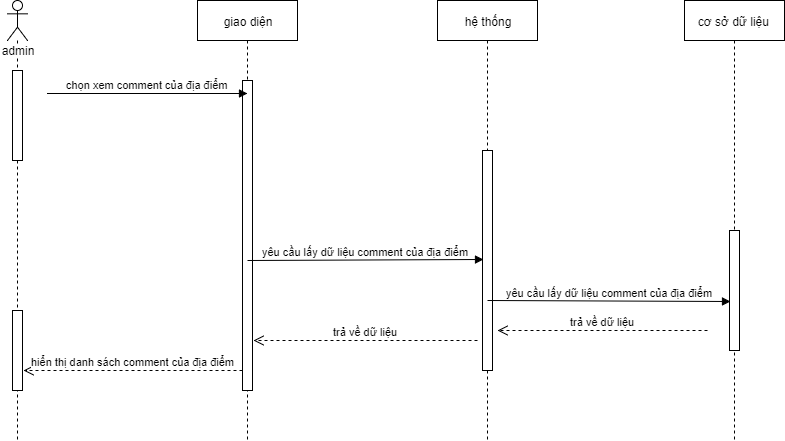


#### **3.3.1.9 xem bình luận của địa điểm**

Biểu đồ lớp

****

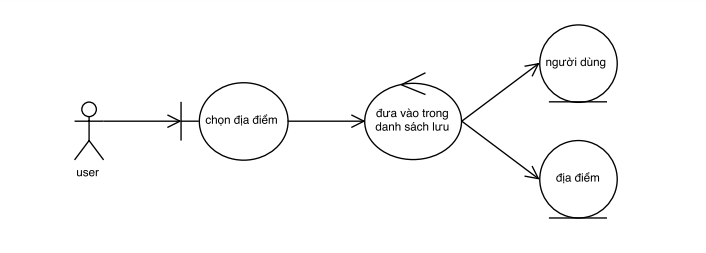
Biểu đồ tuần tự

****

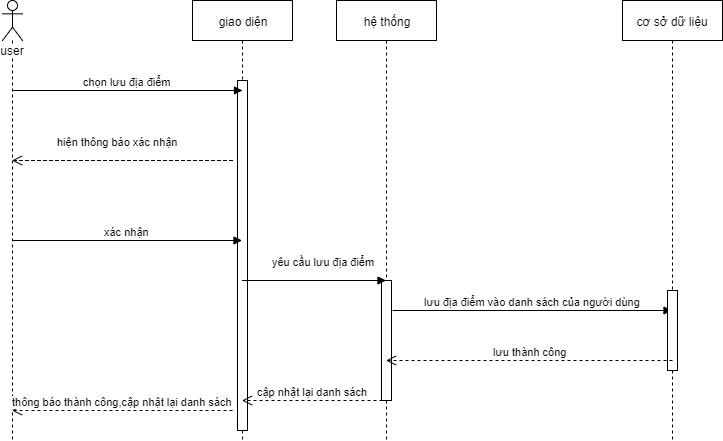
### **3.3.2 Các chức năng của thành viên ( member )**

#### **3.3.2.1 Lưu địa điểm**

Biểu đồ lớp

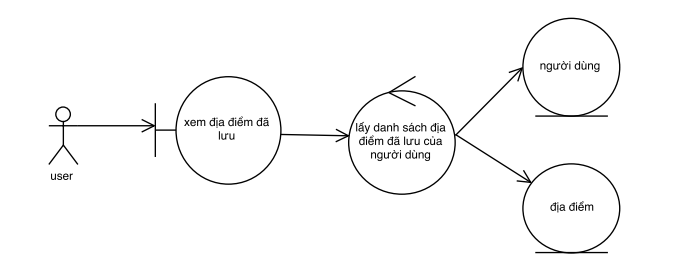


Biểu đồ tuần tự

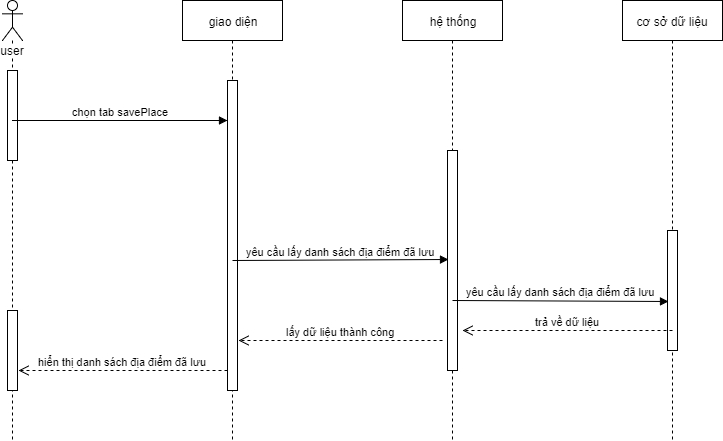
****

#### **3.3.2.2 Xem danh sách đã lưu**

Biểu đồ lớp

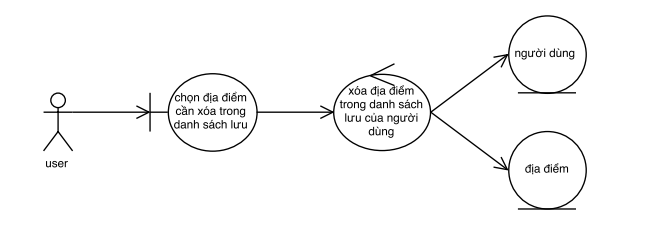


Biểu đồ tuần tự

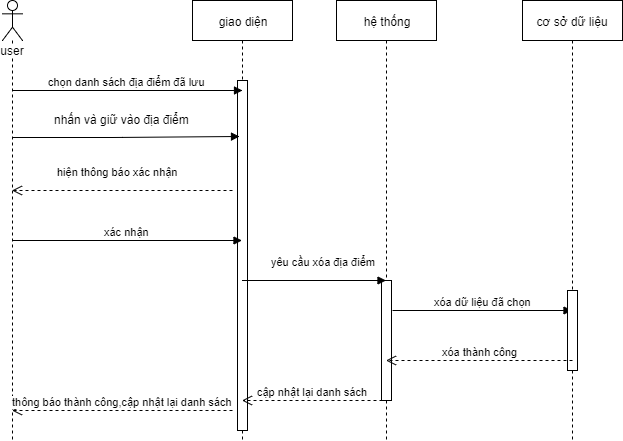
****

#### **3.3.2.3 Xóa địa điểm đã lưu**

Biểu đồ lớp

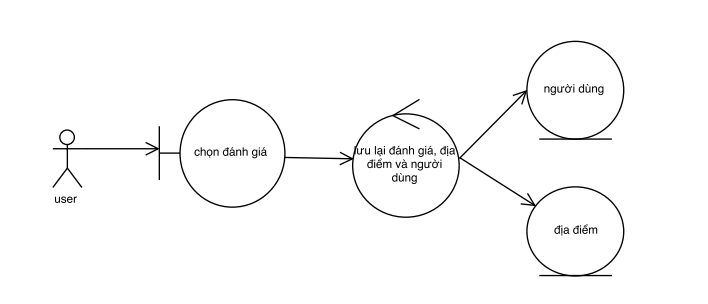


Biểu đồ tuần tự

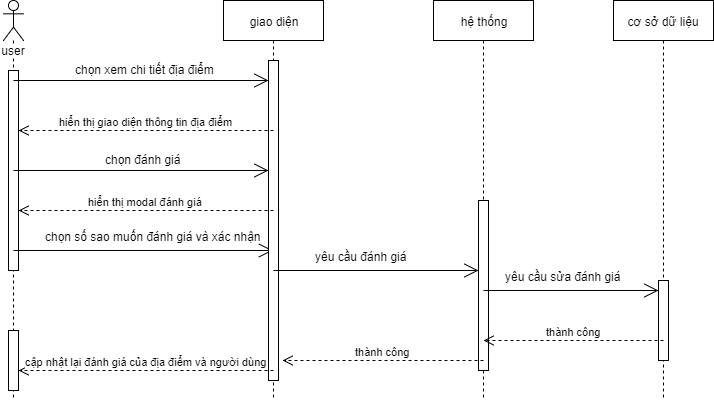
****

#### **3.3.2.4 Đánh giá ( ratting )**

Biểu đồ lớp

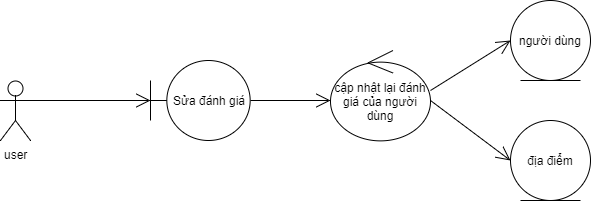


Biểu đồ tuần tự

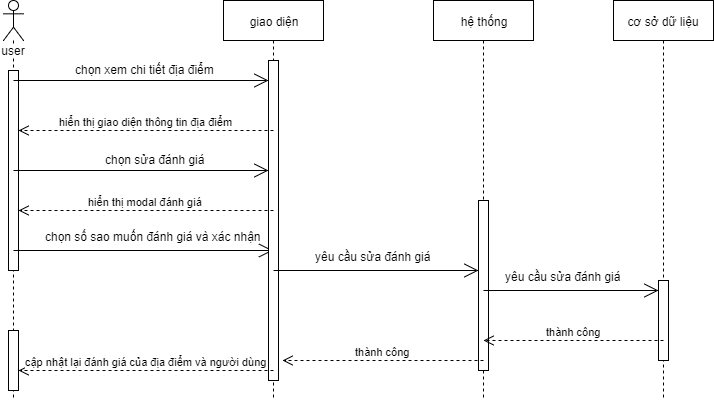
****

#### **3.3.2.5 Sửa đánh giá**

Biểu đồ lớp

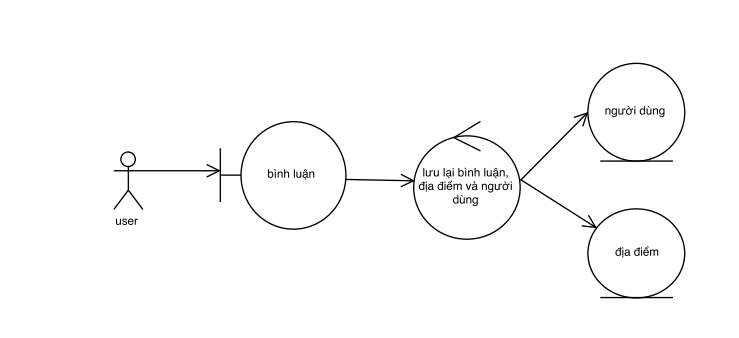


Biểu đồ tuần tự

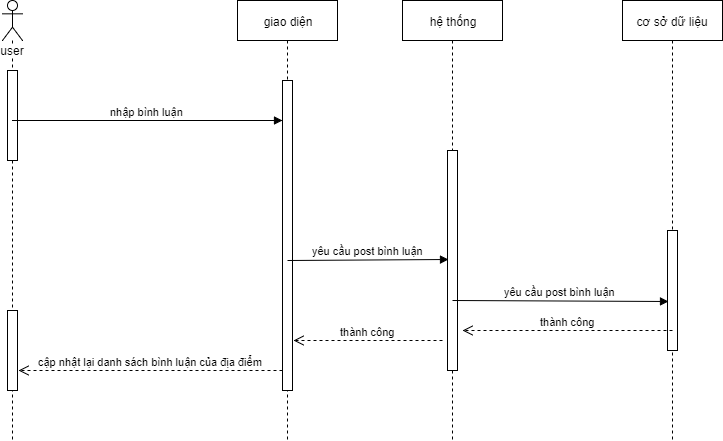


#### **3.3.2.6 Bình luận**

Biểu đồ lớp

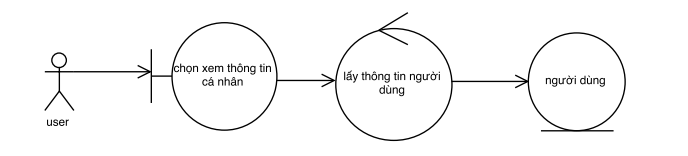


Biểu đồ tuần tự

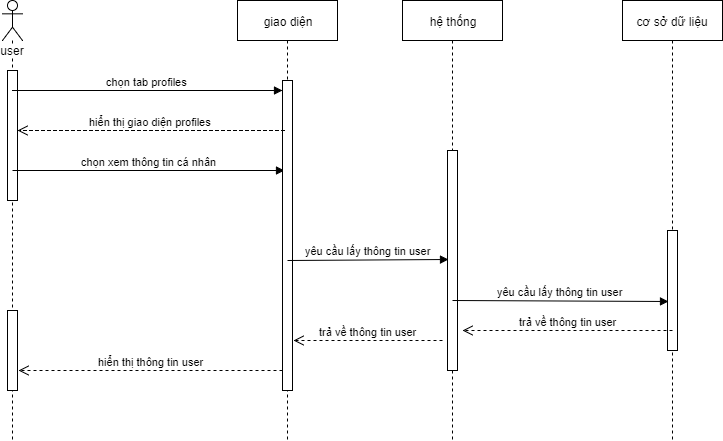
****

#### **3.3.2.7 Xem profiles**

Biểu đồ lớp

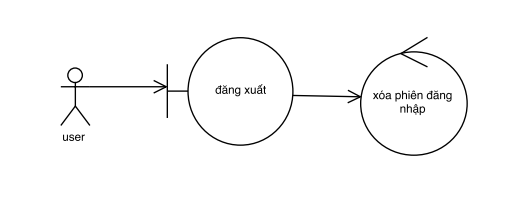


Biểu đồ tuần tự

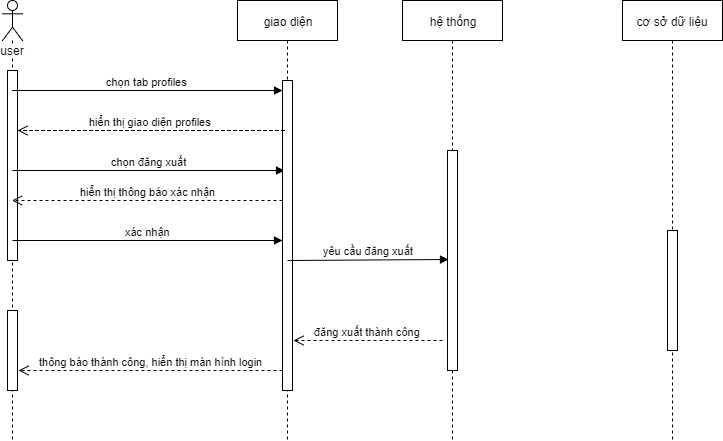
****

#### **3.3.2.8 Đăng xuất**

Biểu đồ lớp



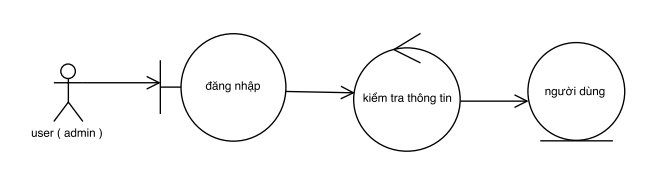
Biểu đồ tuần tự

****

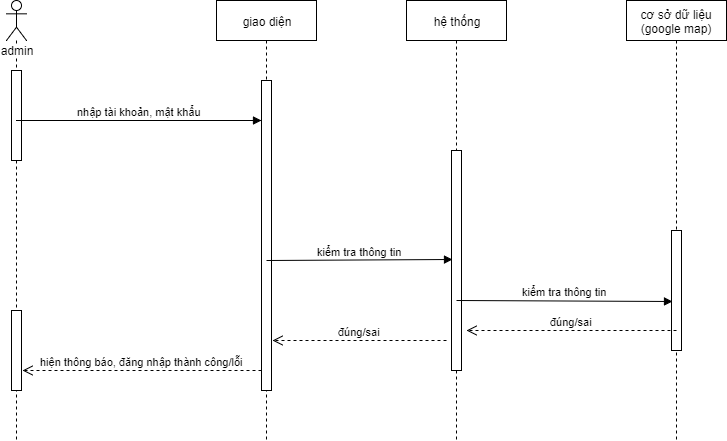
### **3.3.3 Các chức năng của admin**

#### **3.3.3.1 Đăng nhập**

Biểu đồ lớp

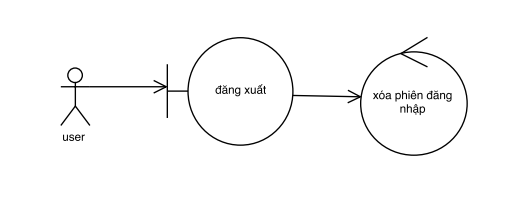


Biểu đồ tuần tự

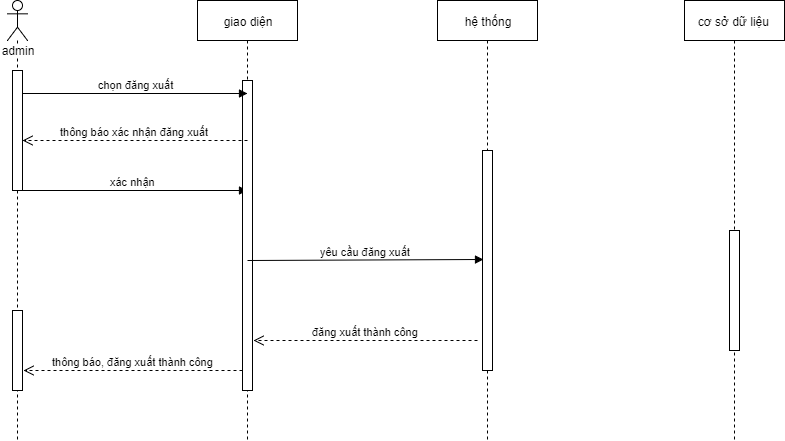
****

#### **3.3.3.2 Đăng xuất**

Biểu đồ lớp

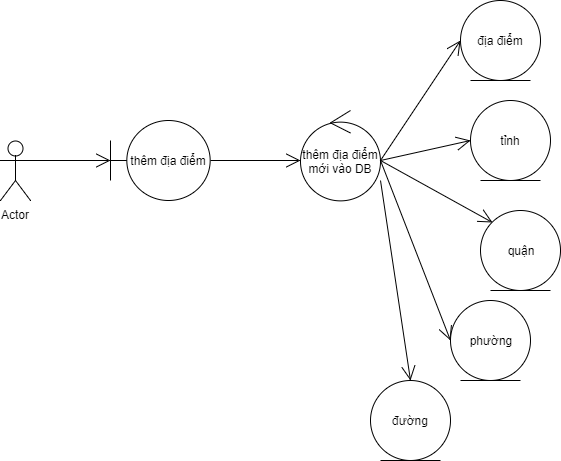


Biểu đồ tuần tự

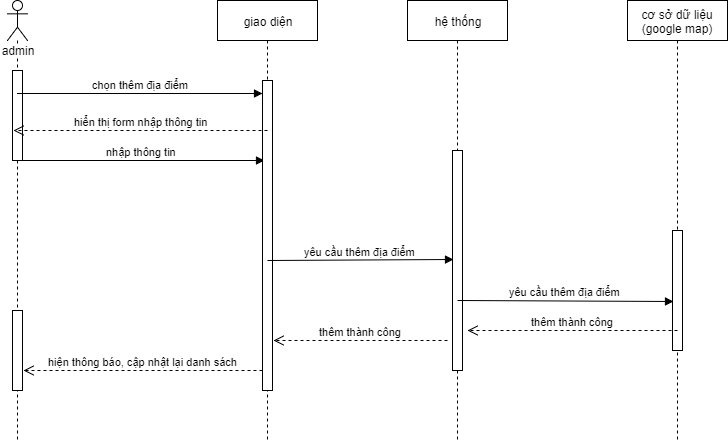
****

#### **3.3.3.3 Thêm địa điểm**

Biểu đồ lớp

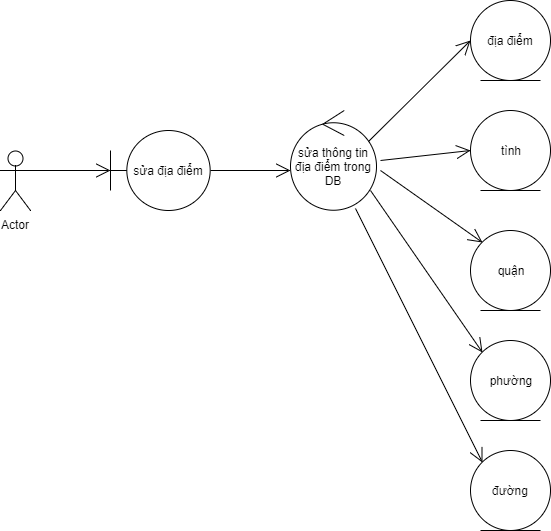


Biểu đồ tuần tự

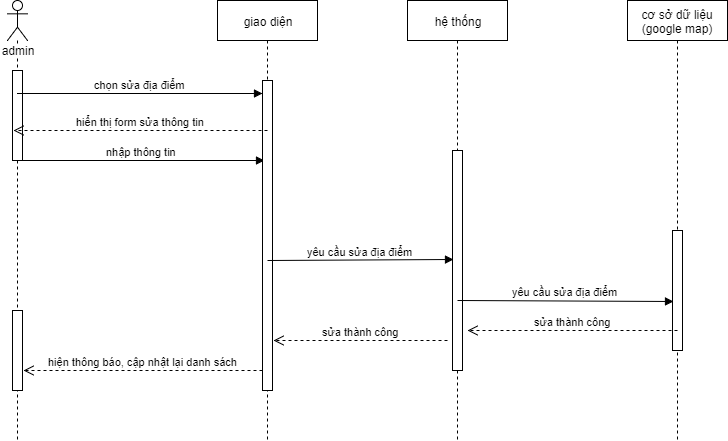
****

#### **3.3.3.4 Sửa địa điểm**

Biểu đồ lớp

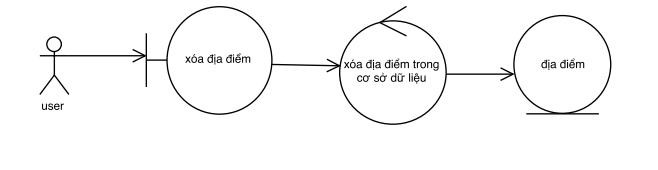


Biểu đồ tuần tự

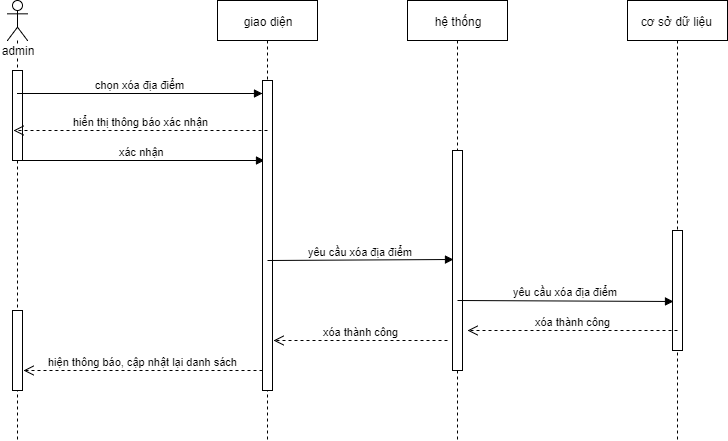
****

#### **3.3.3.5 xóa địa điểm**

Biểu đồ lớp

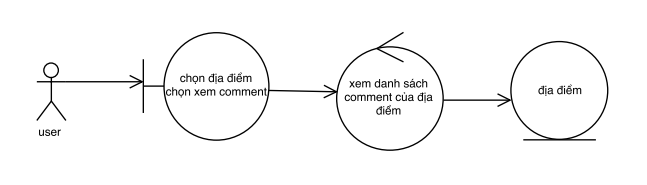


Biểu đồ tuần tự

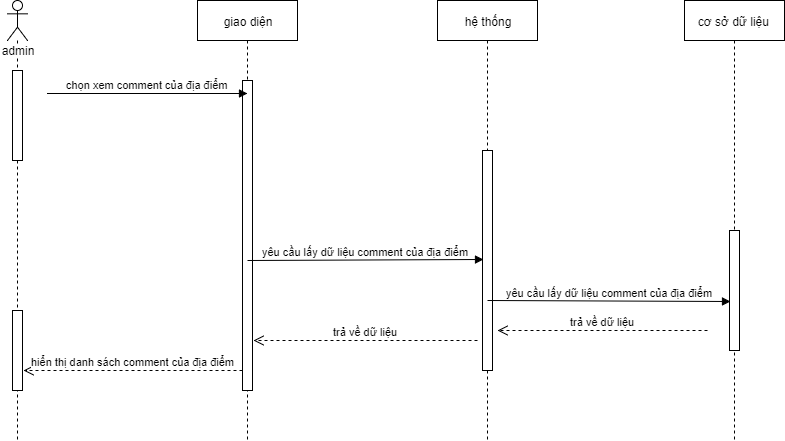
****

#### **3.3.3.6 xem bình luận**

Biểu đồ lớp

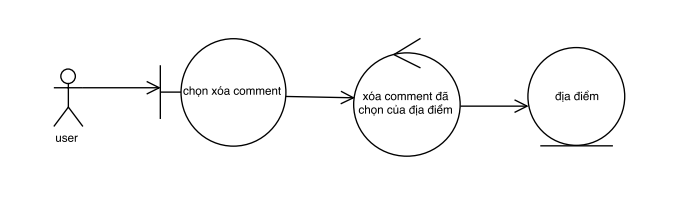
****

Biểu đồ tuần tự

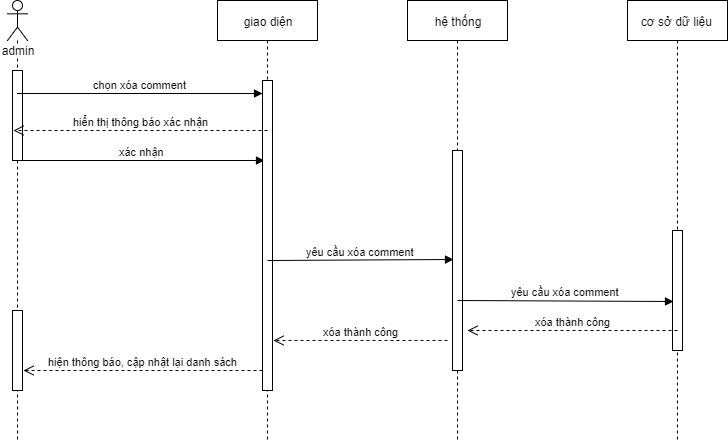
****

#### **3.3.3.7 Xóa bình luận**

Biểu đồ lớp

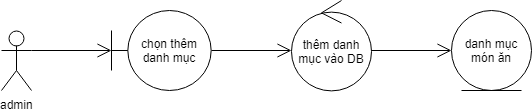


Biểu đồ tuần tự

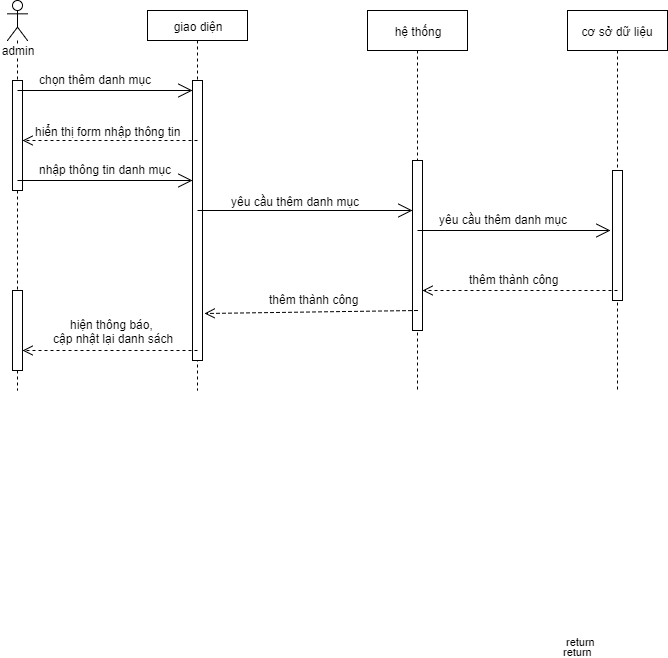
****

#### **3.3.3.8 Thêm danh mục món ăn**

Biểu đồ lớp

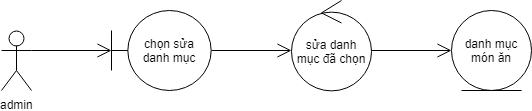


Biểu đồ tuần tự

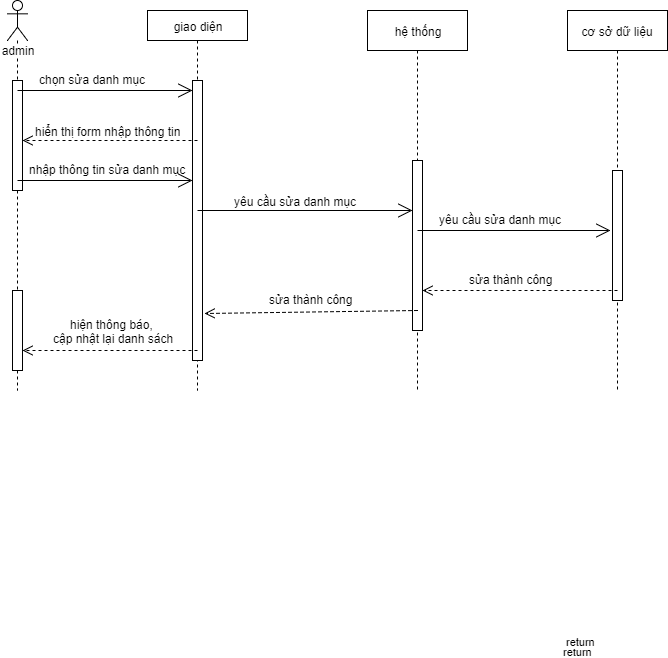


#### **3.3.3.9 Sửa danh mục món ăn**

Biểu đồ lớp

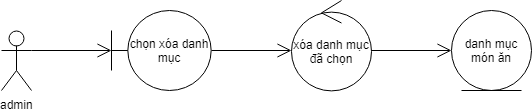


Biểu đồ tuần tự

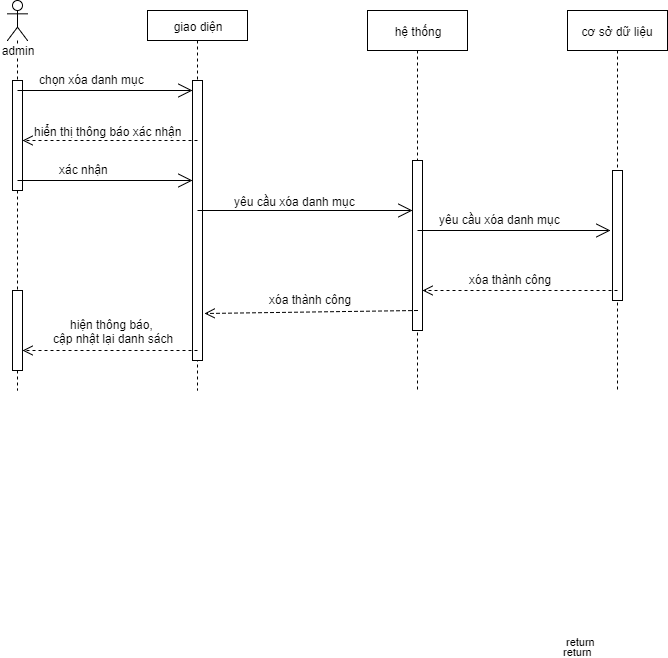


#### **3.3.3.10 Xóa danh mục món ăn**

Biểu đồ lớp

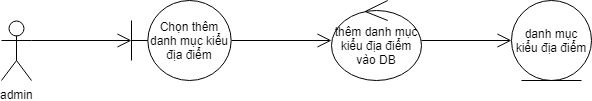


Biểu đồ tuần tự

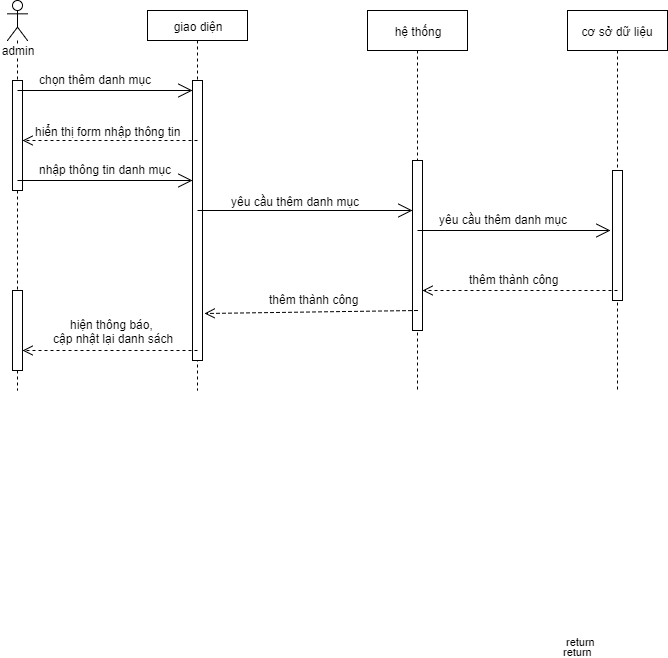


#### **3.3.3.11 Thêm danh mục kiểu địa điểm**

Biểu đồ lớp

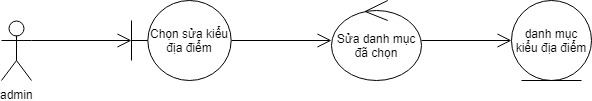


Biểu đồ tuần tự

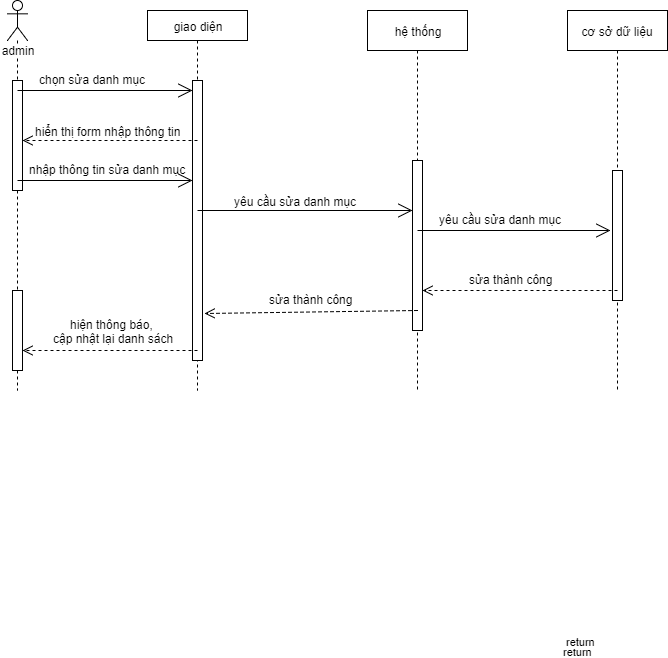


#### **3.3.3.12 Sửa danh mục kiểu địa điểm**

Biểu đồ lớp

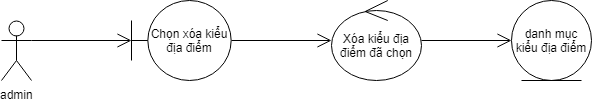


Biểu đồ tuần tự

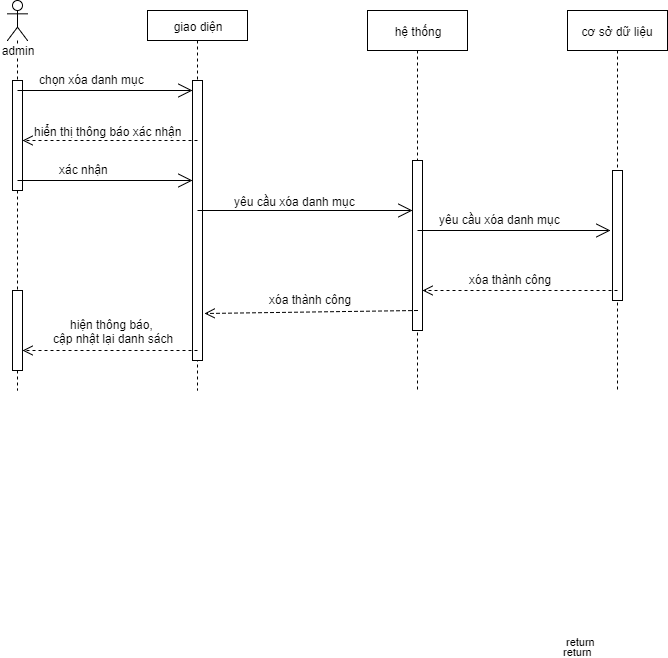


#### **3.3.3.13 Xóa kiểu địa điểm**

Biểu đồ lớp



Biểu đồ tuần tự

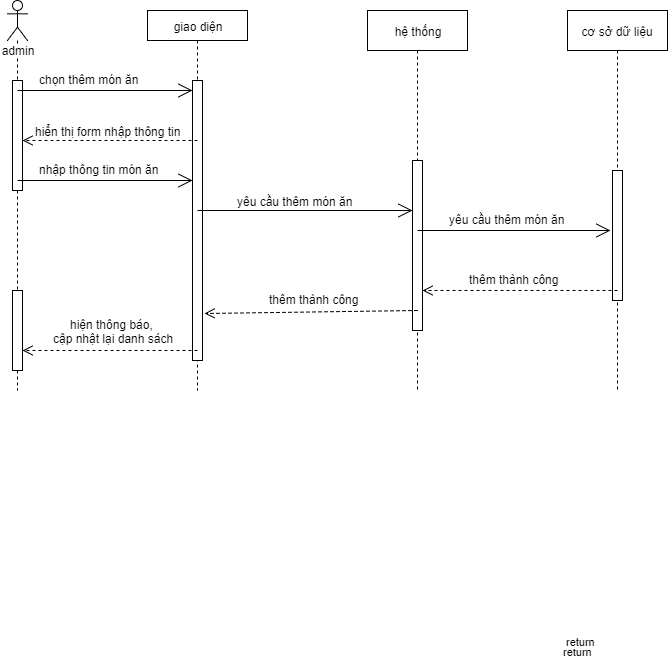


#### **3.3.3.14 Thêm món ăn**

Biểu đồ lớp

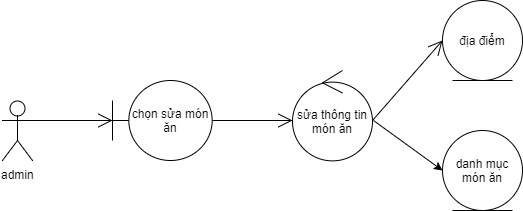


Biểu đồ tuần tự

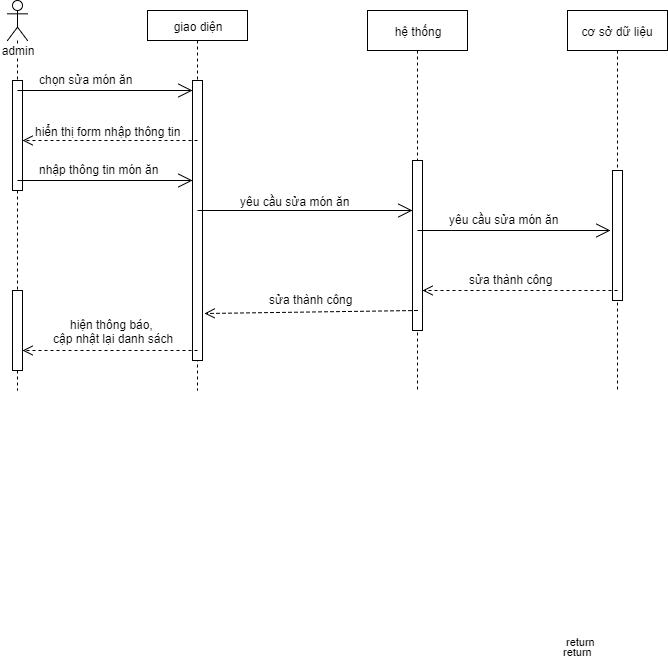


#### **3.3.3.15 Sửa món ăn**

Biểu đồ lớp

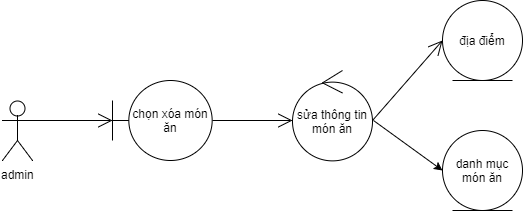


Biểu đồ tuần tự

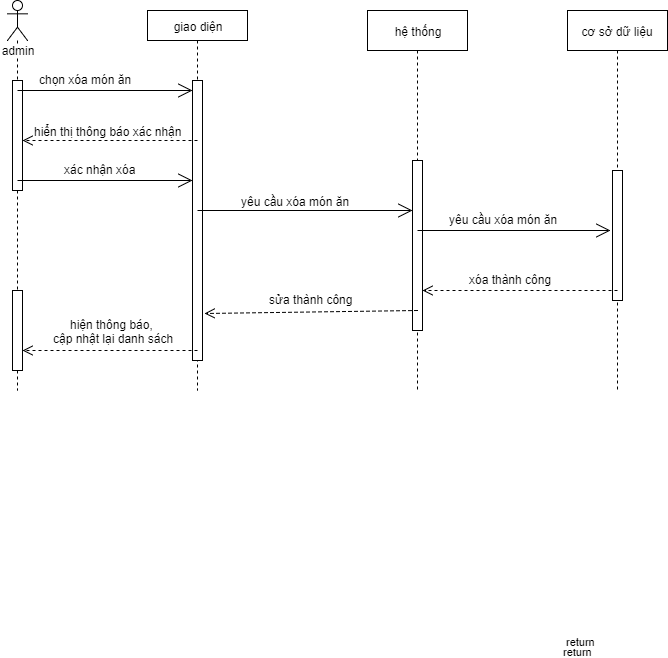


#### **3.3.3.16 Xóa món ăn**

Biểu đồ lớp

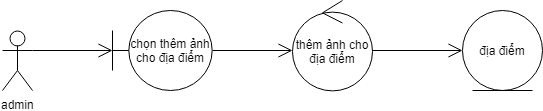


Biểu đồ tuần tự

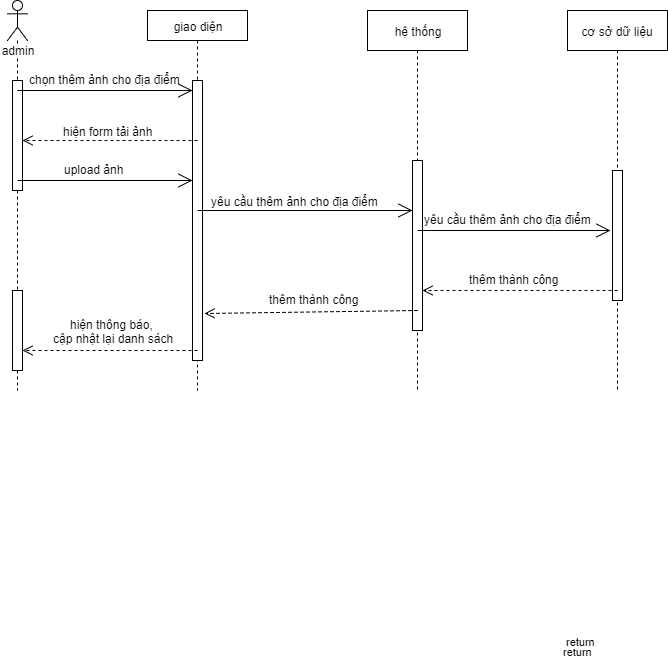


#### **3.3.3.17 Thêm ảnh cho địa điểm**

Biểu đồ lớp



Biểu đồ tuần tự



### **3.3.4 Đồng bộ hóa các thực thể và thiết kế cơ sở dữ liệu**

- Các thực thể còn lại sau khi đồng bộ là

*Thành viên (member), địa điểm(restaurant), ảnh(images), danh mục món ăn(category), danh mục kiểu địa điểm(categoryofrestaurant), tỉnh(province), quận(district), phường(ward), đường(street).*

#### **3.3.4.1 Các đối tượng chính trong hệ thống**

*+ Thành viên: chứa thông tin các tài khoản đăng nhập vào hệ thống*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Id của thành viên |
| Username | Tên tài khoản |
| Password | Mật khẩu của tài khoản |
| Fullname | Tên hiển thị của thành viên trong hệ thống |
| Phone | Số điện thoại thành viên |
| Avatar | ảnh đại diện của thành viên |
| fbId | Id facebook của thành viên |
| fbName | Name facebook của thành viên |
| Picture | Link ảnh avatar facebook của thành viên |
| Cover | Link ảnh bìa facebook của thành viên |
| Type | Kiểu tài khoản ( 0: user, 1: admin ) |
| createAt | Ngày tạo tài khoản |
| updateAt | Ngày cập nhật tài khoản gần nhất |

Ghi chú: fbId, fbName, picture, cover được lấy tự động khi người dùng đăng nhập bằng facebook

*+ Địa điểm ( quán ăn ) ( restaurant ) : chứa thông tin của các địa điểm*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Id của địa điểm |
| Name | Tên của địa điểm |
| Latitude | Vĩ độ của địa điểm |
| Longitude | Kinh độ của địa điểm |
| minPrices | Giá thấp nhất trong thực đơn |
| maxPrices | Giá cao nhất trong thực đơn |
| Open | Giờ mở cửa |
| Close | Giờ đóng cửa |
| Address | Địa chỉ của địa điểm |
| photoUrl | Ảnh đại diện của địa điểm |
| Phone | Số điện thoại của địa điểm |
| createAt | Thời gian tạo địa điểm |
| updateAt | Thời gian cập nhật địa điểm gần nhất |

*+ Ảnh*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Mã của hình ảnh |
| Link | Nguồn của hình ảnh |
| createAt | Ngày tạo ảnh |

*+ Danh mục*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Mã của danh mục |
| Name | Tên của danh mục |
| createdAt | Ngày tạo danh mục |
| updatedAt | Ngày cập nhật danh mục |

*+ Danh mục kiểu địa điểm*

|  |  |
| --- | --- |
| Tên | Mô tả |
| id | Mã của kiểu địa điêm |
| name | Tên của kiểu địa điểm |
| createdAt | Ngày tạo danh mục |
| updatedAt | Ngày cập nhật danh mục |

*+ Tỉnh*

|  |  |
| --- | --- |
| Tên | Mô tả |
| id | Mã của tỉnh |
| name | Tên tỉnh |

*+ Quận*

|  |  |
| --- | --- |
| Tên | Mô tả |
| id | Mã của quận |
| name | Tên quận |

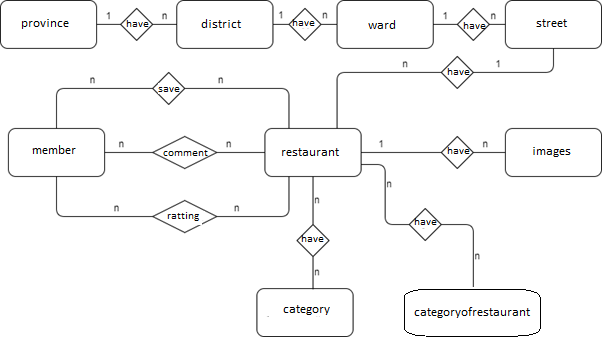
*+ Phường*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Mã của phường |
| Name | Tên phường |

*+ Đường*

|  |  |
| --- | --- |
| Tên | Mô tả |
| Id | Mã của đường |
| Name | Tên đường |

#### **3.3.4.2 Sơ đồ quan hệ thực thể ( ERD )**



* **Phân tích mối quan hệ các thực thể:**
* Xét quan hệ bình luận ( comment ) giữa thành viên và địa điểm là quan hệ n-n ( một thành viên bình luận nhiều địa điểm, một địa điểm có nhiều thành viên bình luận )
* Mối quan hệ này tạo ra một bảng mới

**Bảng bình luận ( comment )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của comment |
| 2 | commentDescription | Text |  | Nội dung comment |
| 3 | uId | int |  | Mã người bình luận |
| 4 | resId | int |  | Mã địa điểm được bình luận |
| 5 | createdAt | Time-stamp |  | Ngày viết bình luận |

* Xét quan hệ đánh giá ( ratting ) giữa thành viên và địa điểm là quan hệ n-n ( một thành viên đánh giá nhiều địa điểm, một địa điểm có nhiều thành viên đánh giá, một thành viên chỉ được đánh giá 1 địa điểm 1 lần )
* Mối quan hệ này tạo ra một bảng mới

**Bảng ratting**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của đánh giá |
| 2 | resId | int |  | Mã địa điểm được đánh giá |
| 3 | uId | int |  | Mã người đánh giá |
| 4 | Star | int |  | Số lượng ngôi sao đánh giá |
| 5 | createdAt | dateTime |  | Ngày đánh giá |
| 6 | updatedAt | Time-stamp |  | Ngày cập nhật đánh giá |

* Xét quan hệ lưu ( save ) giữa thành viên và địa điểm là quan hệ n-n ( một thành viên lưu nhiều địa điểm, một địa điểm có nhiều thành viên lưu )
* Mối quan hệ này tạo ra một bảng mới

**Bảng lưu ( save )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của lần lưu |
| 2 | uId | int |  | Mã người lưu |
| 3 | resId | int |  | Mã địa điểm được lưu |
| 4 | savedAt | Time-stamp |  | Ngày lưu địa điểm |

* Xét quan hệ giữa địa điểm và danh mục món ăn là quan hệ n-n ( một địa điểm có nhiều danh mục, 1 danh mục món ăn thuộc nhiều địa điểm )
* Tạo ra bảng mới

**Bảng món ăn ( food )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | Int |  | Mã của món ăn |
| 2 | resId | Int |  | Mã của địa điểm |
| 3 | cateId | Int |  | Mã của danh mục |
| 4 | name | Varchar | 200 | Tên món ăn |
| 5 | price | Int |  | Giá của món ăn |
| 6 | createdAt | Date-time |  | Ngày tạo món ăn |
| 7 | updatedAt | Time-stamp |  | Ngày cập nhật món ăn |

* Xét quan hệ giữa địa điểm và danh mục kiểu địa điểm là quan hệ n-n ( một địa điểm có nhiều danh mục, 1 danh mục thuộc nhiều địa điểm )
* Tạo ra bảng mới

**Bảng nhóm địa điểm ( grouprestaurant )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của món ăn |
| 2 | resId | int |  | Mã của địa điểm |
| 3 | CategoryTypeId | int |  | Mã của kiểu địa điểm |
| 4 | createdAt | Time-stamp |  | Ngày tạo nhóm địa điểm |

* Xét quan hệ giữa tỉnh/thành phố và quận là quan hệ 1-n ( 1 tỉnh/thành phố có nhiều quận, 1 quận chỉ thuộc 1 tỉnh/thành phố )
* Đưa khóa chính của bảng bên 1 ( tỉnh/thành phố ) sang làm khóa ngoại của bảng bên n ( quận )
* Xét quan hệ giữa quận và phường là quan hệ 1-n ( 1 quận có nhiều phường, 1 phường chỉ thuộc 1 quận )
* Đưa khóa chính của bảng bên 1 ( quận ) sang làm khóa ngoại của bảng bên n ( phường )
* Xét quan hệ giữa phường và đường là quan hệ 1-n ( 1 phường có nhiều đường, 1 đường chỉ thuộc 1 phường )
* Đưa khóa chính của bảng bên 1 ( phường ) sang làm khóa ngoại của bảng bên n ( đường )
* Xét quan hệ giữa đường và địa điểm là quan hệ 1-n ( 1 đường có nhiều địa điểm, 1 địa điểm chỉ thuộc 1 đường )
* Đưa khóa chính của bảng bên 1 ( đường ) sang làm khóa ngoại của bảng bên n ( địa điểm )
* Xét quan hệ giữa địa điểm và ảnh là quan hệ 1-n ( 1 địa điểm có nhiều ảnh, 1 ảnh chỉ thuộc 1 địa điểm )
* Đưa khóa chính của bảng bên 1 ( địa điểm ) sang làm khóa ngoại của bảng bên n ( ảnh )

**3.3.4.3 Danh sách các bảng cùng thuộc tính sau khi chuẩn hóa là**

**Bảng member ( thành viên )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Kích thước | Mô tả |
| 1 | id | Int |  | Mã của thành viên trong hệ thống |
| 2 | username | Varchar | 200 | Tên tài khoản |
| 3 | password | Varchar | 50 | Mật khẩu của tài khoản |
| 4 | fullname | Varchar | 200 | Tên hiển thị trong hệ thống |
| 5 | phone | Varchar | 20 | Số điện thoại của thành viên |
| 6 | avatar | Varchar | 200 | Đường dẫn ảnh đại diện của thành viên |
| 7 | fbId | Bigint | 20 | Id facebook của thành viên ( nếu đăng nhập bằng facebook ) |
| 8 | fbName | Varchar | 200 | Tên facebook của thành viên (nếu đăng nhập bằng facebook ) |
| 9 | picture | Varchar | 500 | Đường dẫn ảnh đại diện facebook của người dùng |
| 10 | cover | Varchar | 500 | Đường dẫn ảnh bìa facebook của người dùng |
| 11 | Type | Int | 2 | Kiểu tài khoản ( 0: user, 1: admin ) |
| 12 | createdAt | Date-time |  | Ngày tạo tài khoản |
| 13 | updatedAt | timestamp |  | Ngày cập nhật tài khoản gần nhất |

**Bảng Tỉnh(Thành phố)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của tỉnh |
| 2 | Name | varchar | 100 | Tên của tỉnh |

**Bảng Quận**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của quận |
| 2 | Name | varchar | 100 | Tên của quận |
| 3 | provinceId | Int |  | Mã của tỉnh(thành phố ) |

**Bảng Phường**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của phường |
| 2 | Name | varchar | 100 | Tên của phường |
| 3 | districtId | int |  | Mã của quận |

**Bảng Đường**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | int |  | Mã của đường |
| 2 | Name | varchar | 100 | Tên của đường |
| 3 | wardId | int |  | Mã của phường |

**Bảng địa điểm**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | Int |  | Mã của địa điểm |
| 2 | name | varchar | 200 | Tên của địa điểm |
| 3 | latitude | double |  | Vĩ độ của địa điểm |
| 4 | longitude | double |  | Kinh độ của địa điểm |
| 5 | minPrices | Int |  | Giá thấp nhất trong thực đơn |
| 6 | maxPrices | Int |  | Giá cao nhất trong thực đơn |
| 7 | open | Time |  | Giờ mở cửa |
| 8 | close | Time |  | Giờ đóng cửa |
| 9 | address | Varchar | 200 | Địa chỉ của địa điểm |
| 10 | photoUrl | Varchar | 200 | Đường dẫn của ảnh đại diện của địa điểm |
| 11 | phone | Varchar | 20 | Số điện thoại của địa điểm |
| 12 | createdAt | Date-time |  | Ngày tạo địa điểm |
| 13 | updatedAt | time-stamp |  | Ngày cập nhật địa điểm |
| 14 | streetId | Int |  | Mã đường |

**Bảng Images**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | Int |  | Mã của ảnh |
| 2 | resId | Int |  | Mã của địa điểm chứa ảnh |
| 3 | link | Varchar | 200 | Đường dẫn của ảnh |
| 4 | createdAt | Time-stamp |  | Ngày tải ảnh lên |

**Bảng danh mục ( category )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | int |  | Mã của danh mục |
| 2 | name | Varchar | 200 | Tên của danh mục |
| 3 | createdAt | Date-time |  | Ngày tạo danh mục |
| 4 | updatedAt | Time-stamp |  | Ngày cập nhật danh mục |

**Bảng danh mục kiểu địa điểm ( categoryofrestaurant )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | int |  | Mã của danh mục |
| 2 | name | Varchar | 200 | Tên của danh mục |
| 3 | createdAt | datetime |  | Ngày tạo |
| 4 | updatedAt | Timestamp |  | Ngày cập nhật |

**Bảng nhóm địa điểm ( grouprestaurant )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | Int |  | Mã nhóm |
| 2 | resId | Int |  | Mã địa điểm |
| 3 | categoryTypeId | Int |  | Mã kiểu địa điểm |
| 4 | createdAt | time-stamp |  | Ngày tạo |

**Bảng món ăn ( food )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | Id | Int |  | Mã của món ăn |
| 2 | name | Varchar | 200 | Tên của món ăn |
| 3 | price | Int |  | Giá của món ăn |
| 4 | resId | Int |  | Mã của địa điểm |
| 5 | cateId | Int |  | Mã của danh mục |
| 6 | createdAt | Datetime |  | Ngày tạo món ăn |
| 7 | updatedAt | Timestamp |  | Ngày cập nhật món ăn |

**Bảng save**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | Int |  | Mã của lần lưu |
| 2 | uId | Int |  | Mã người lưu |
| 3 | resId | Int |  | Mã địa điểm |
| 4 | savedAt | Time-stamp |  | Ngày lưu địa điểm |

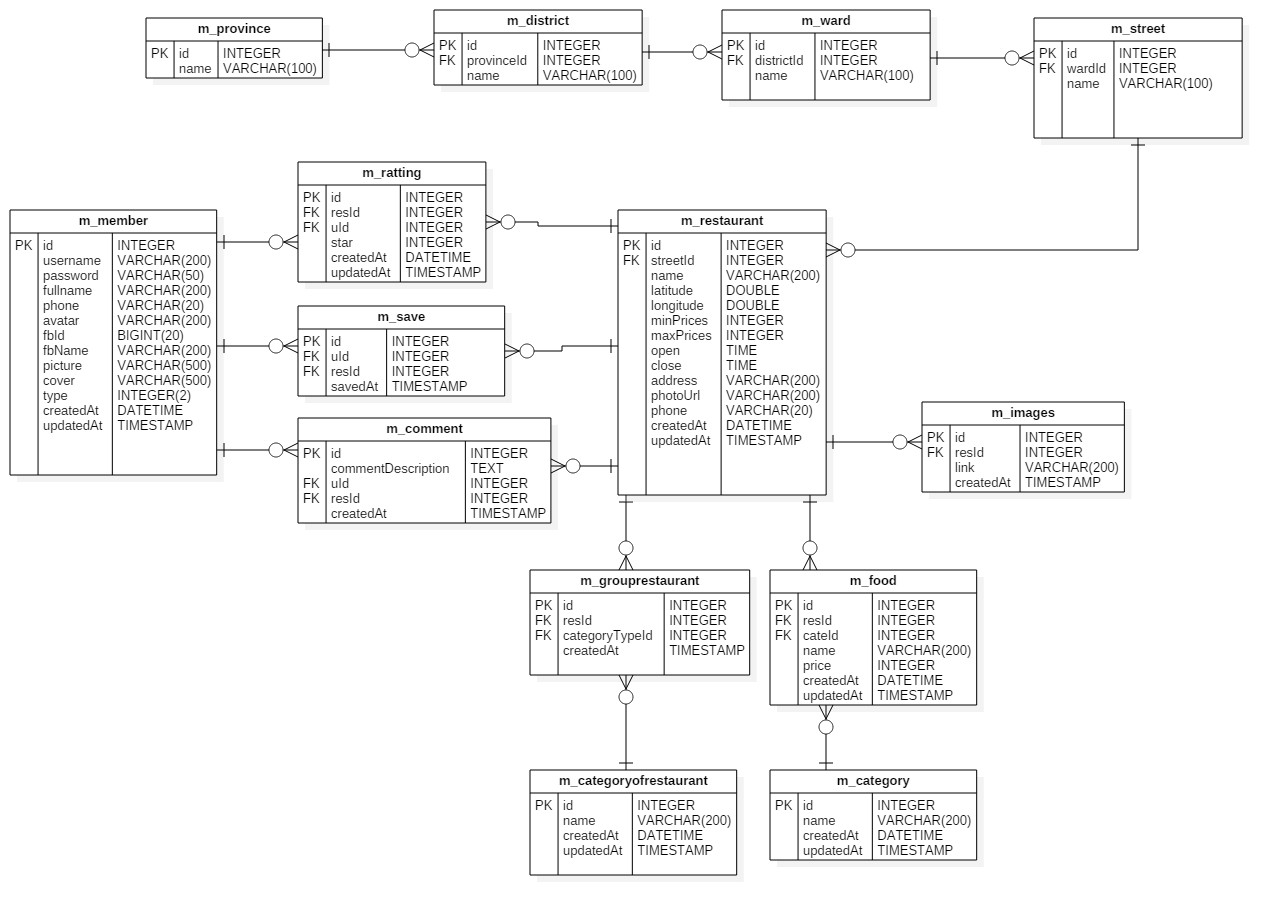
**Bảng ratting**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | Int |  | Mã của đánh giá |
| 2 | resId | Int |  | Mã địa điểm được đánh giá |
| 3 | uId | Int |  | Mã người đánh giá |
| 4 | star | Int |  | Số đánh giá |
| 5 | createdAt | datetime |  | Ngày đánh giá |
| 6 | updatedAt | timestamp |  | Ngày cập nhật đánh giá |

**Bảng bình luận ( comment )**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | Tên trường | Kiểu dữ liệu | Độ dài | Mô tả |
| 1 | id | int |  | Mã của comment |
| 2 | commentDescription | Text |  | Nội dung comment |
| 3 | uId | int |  | Mã người bình luận |
| 4 | resId | int |  | Mã địa điểm được bình luận |
| 5 | createdAt | Timestamp |  | Ngày viết bình luận |

Mô hình quan hệ giữa các bảng



## **CHƯƠNG 4: GIỚI THIỆU GIAO DIỆN ỨNG DỤNG**

## **CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **5.1 Kết luận**

Với sự hướng dẫn nhiệt tình của các thầy cô trong Khoa Công nghệ thông tin, đặc biệt là sự chỉ bảo tận tình của thầy Nguyễn Văn Nam trong thời gian làm đồ án của mình, nay em đã hoàn thành đồ án và đạt được một số kết quả sau:

* Nắm được cơ bản về một số quy trình khi tiền hành xây dựng một hệ thống trong thực tế (khảo sát thực tế, thu thập dữ liệu,…, triển khai, thử nghiệm).
* Tìm hiểu về phân tích và thiết kế cơ sở dữ liệu. Từ đó, áp dụng vào việc phân tích bài toán “*Xây dựng ứng dụng tìm địa điểm ăn uống*”.
* Tìm hiểu được các công nghệ phát triển platform và tìm hiểu được các công nghệ mới như React Native, NodeJS, Google Map API.
* Xây dựng ứng dụng tìm kiếm địa điểm ăn uống có các chức năng sau:
* Xem thông tin, bình luận, đánh giá về địa điểm ăn uống.
* Quản trị các địa điểm ăn uống.
* Tìm kiếm dễ dàng với giao diện thân thiện với người sử dụng.

Mặc dù đã có nhiều cố gắng trong suốt quá trình làm đồ án, nhưng do thời gian có hạn và thiếu kinh nghiệm trong thực tế nên cũng không thể tránh khỏi những nhược điểm và thiếu sót. Vì vậy, em mong nhận được sự góp ý của quý thầy cô và các bạn để em có thể phát triển hệ thống này hoàn thiện hơn.

### **5.2 Hướng phát triển đề tài**

Sau khi hoàn thành đồ án và xây dựng được ứng dụng tìm kiếm địa điểm ăn uống , em sẽ cố gắng tìm hiểu và bổ sung thêm các chức năng để ứng dụng mang lại nhiều tiện ích với người dùng hơn.

Một số hướng để em có thể phát triển hoàn thiện hơn đó là:

* + Tìm hiểu sâu hơn về NodeJS, React Native, MySQL, Google Map API để có thể đáp ứng nhiều hơn nữa nhu cầu của người sử dụng, phát triển và tối ưu hóa hệ thống.
* Khảo sát chi tiết hơn nhu cầu của người dùng để hoàn thiện các chức năng của hệ thống.

## **CHƯƠNG 6: TÀI LIỆU THAM KHẢO**

### **6.1 Tài liệu tham khảo online**

1. <https://facebook.github.io/react-native/>

2. <https://github.com/>

3. <https://www.npmjs.com/>

4. <https://www.w3schools.com/>

5. <https://stackoverflow.com/>

### **6.2 Giáo trình**

1. Đặng Đức Hạnh – Trương Ninh Thuận, *Phân tích và thiết kế hướng đối tượng.*

2. Phạm Thị Hoàng Nhung, *Cơ sở dữ liệu I.*

3. Phạm Thị Hoàng Nhung, *Cơ sở dữ liệu II.*

4. Phạm Thị Hoàng Nhung, *Hệ quản trị cơ sở dữ liệu SQL Server 2000.*

5. Đặng Văn Đức, Nguyễn Thị Phương Trà, *Giao diện Người – Máy.*