Report #3: Final report

Food Map

Introduction to Software Engineering

(CSC13002)

Group name: **The Dreamers**

1. **Lê Công Luận**
2. **Huỳnh Kim Ninh**
3. **Phan Anh Phú**
4. **Châu Hoàng Phúc**
5. **Nguyễn Văn Phước**

Ho Chi Minh City, 29/12/2018

Revision History

Version No.	Date of Revision
v.1	27/12/2018

.https://github.com/kim-ninh/ISE_NTMT_04

Individual Contributions Breakdown

	Luận	Ninh	Phúc	Phước	Phú
Summarry of Changes		X			
Customer Statement of Requirments #1					X
Glossary of Terms #1					X
System Requirements #1		X			
Functional Requirements Specification #1		X			
Domain Analysis #1	X				
Class Diagram and Interface Specification #2			X		
System Architecture and System Design #2				X	
Algorithm and Data Structures #2		X			
User Interface Design and implementation #1	X				
Design of Tests #3				X	
History of Work, Current Status, and Future		X			

Bảng phân công các nội dung trong báo cáo

Table of Contents

Summary of Changes

- Cập nhật biểu đồ Use-case (Thêm 1 số Use-case: Xem bản đồ, Cập nhật hình,...)
- Thêm giải thích thuật ngữ “Order” trong app
- Thay đổi định nghĩa Guest và User
- Thêm một loại người dùng mới: Admin - người sẽ kiểm duyệt các yêu cầu đăng ký quán của Owner, từ đó có thể duyệt hoặc hủy bỏ.

Customer Statement of Requirements

Nhu cầu tìm kiếm 1 địa điểm ăn uống phù hợp với túi tiền, hợp vệ sinh ở 1 nơi xa lạ mà bạn lần đầu đặt chân đến hoặc ít khi ghé đến hiện nay là rất phổ biến. Có thể kể đến các trường hợp như là đi phượt, đi dã ngoại, đi du lịch dài ngày, chuyển từ quê lên thành phố của các bạn sinh viên,... Vì thế, Food Map ra đời để giải quyết 1 phần các khó khăn trên, giúp thực khách có thể nhanh chóng tìm kiếm cũng như chủ quán có thể quảng bá được quán họ được nhiều người biết hơn. Food Map sẽ có 2 loại người chính là thực khách (User) và chủ quán (Owner). Ngoài ra sẽ có thêm quản trị viên (Admin) sẽ xét duyệt các đăng ký mở quán của Owner

1. **User**:
   Sử dụng app nhằm tìm được quán ăn tức thời, dù cho đang ở bất cứ đâu. Có thể đươc chỉ dẫn đến tận nơi cần đến nếu không rành đường. Hoàn toàn chủ động trong việc nắm toàn bộ thông tin quán ăn như giá cả, giờ hoạt động, số bàn trống,... Để lại phản hồi về chất lượng quán, góp ý quán,... So sánh 2 quán với nhau.

2. **Owner**:
   Sử dụng app nhằm để được nhiều người biết đến quán họ hơn, cung cấp cho các thực khách 1 sự chọn lựa.

3. **Admin**:
   Xem xét, phê duyệt các yêu cầu đăng ký mở quán của Owner.

Glossary of Terms

user
Người sử dụng là khách, chưa đăng nhập

guest
Người sử dụng là khách, đã đăng nhập

owner
Người sử dụng là chủ quán, đã đăng nhập

admin
Người quản trị hệ thống

discount
Đợt khuyến mãi, ưu đãi, giảm giá

order
Đơn đặt món ăn

System Requirements

1. Enumerated Functional Requirements

Identifier	Priority Weight	Description
R01	1	Người dùng có thể xem các quán ăn ở khu vực xung quanh họ.
R02	1	Người dùng có thể xem thông tin chi tiết của quán ăn
R03	3	Người dùng có thể bình luận đánh giá quán ăn *(sau khi đã đăng nhập)*
R04	2	Người dùng có thể xem hướng dẫn chỉ đường đến quán
R05	3	Người dùng có thể lưu lại 1 quán vào danh sách yêu thích
R06	1	Chủ quán có thể thêm mới 1 quán ăn vào bản đồ
R07	1	Chủ quán có thể thêm thông tin các món ăn của quán họ
R08	2	Chủ quán có thể chỉnh sửa thông tin quán *(Địa chỉ hiện tại, số điện thoại, tên quán, giờ phục vụ,...)*
R09	2	Chủ quán có thể chỉnh sửa thông tin món ăn *(Cập nhật giá tiền, hình ảnh, tên món,...)*
R10	3	Chủ quán có thể trả lời các nhận xét từ phía thực khách
R11	4	Cả chủ quán và người dùng đều có thể gửi feedback cho nhà phát triển.
R12	5	Người dùng có thể đặt món (chỉ khi có discount)
R13	5	Chủ quán có thể tạo discount

2. Enumerated Nonfunctional Requirements

Identifier	Priority Weight	Description
REQ1	3	Thời gian nạp các dữ liệu ban đầu không quá **5s** cho tập dữ liệu **1000 dòng**
REQ2	1	Thời gian cập nhật các chi tiết trên bản đồ khi thu phóng không quá **2s** (khi dùng **offline**)
REQ3	2	Thời gian phản hồi người dùng không vượt quá **1s**

Functional Requirements Specification

1. Stakeholders

- Người tìm kiếm: bình luận, đánh giá
- Chủ quán ăn: thêm địa điểm quán ăn, món ăn
- Người quản trị: phê duyệt các yêu cầu đăng ký quán ăn của Owner
- Người bảo trì: đảm bảo hệ thống ổn định, nâng cấp hệ thống

2. Actors and Goals

	Actor	Role
1.	User	Người tìm kiếm, xem thực đơn quán
2.	Owner	Chủ quán, quản lý toàn bộ thông tin về quán
3.	Guest	Người nhận xét, đánh giá
4.	Admin	Người quản trị hệ thống, phê duyệt các đăng ký mở quán ăn

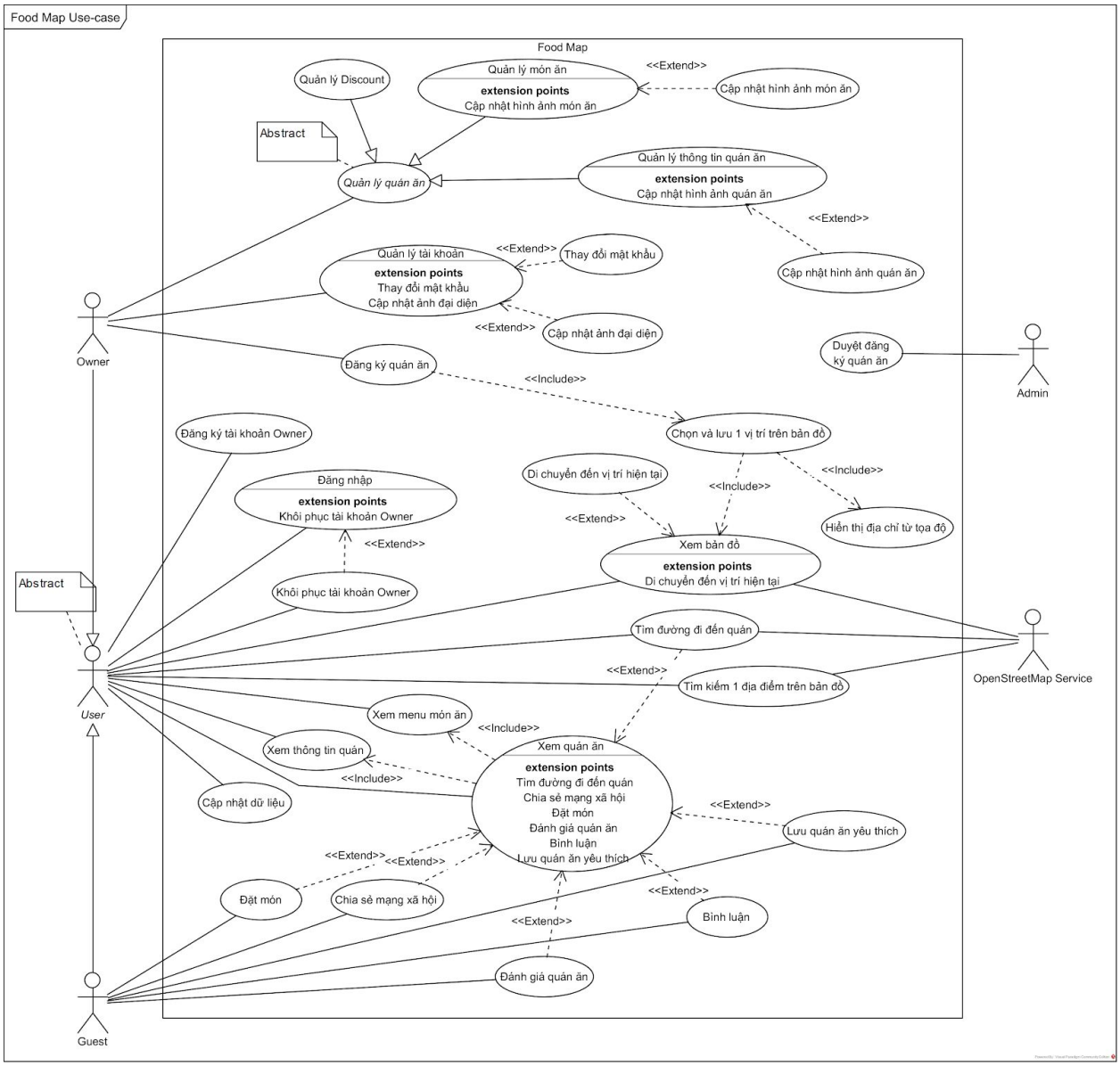
3. Use Cases

a. Casual Description

Use case ID	Tên use case	Tóm tắt ngắn gọn
UC1	Xem các quán ăn xung quanh 1 địa điểm	Cho phép Guest xem vị trí các quán ăn xung quanh vị trí của mình
~~UC2~~	~~Thay đổi bán kính khu vực hiển thị~~	~~Mở rộng hoặc thu nhỏ phạm vi tìm kiếm các quán~~
UC3	Xem thông tin quán ăn	Guest có thể vào xem thông tin của quán ăn
UC4	Xem thông tin món ăn	Use case cho phép Guest xem các menu các món ăn hiện có trong quán

UC5	Đánh giá quán ăn	Đánh giá trên thang tiêu chí 5 sao
UC6	Tìm đường đi đến quán ăn	Chỉ dẫn Guest đi tới quán
UC7	Lưu quán vào danh sách yêu thích	Lưu lại các quán ăn yêu thích vào tài khoản User
UC8	Đăng ký tài khoản	Tạo một tài khoản Owner mới
UC9	Đăng nhập	Đăng nhập vào hệ thống
UC10	Đăng ký thông tin quán	Cho phép Owner đăng ký thông tin của một quán ăn
UC11	Quản lý thông tin quán	Cho phép Owner thêm, xóa, sửa các thông tin và quán của họ
UC12	Quản lý menu quán	Cho phép Owner thêm, xóa, thay đổi thông tin món ăn trong menu
~~UC13~~	~~Quản lý đánh giá~~	~~Cho phép Owner xem thống kê các phản hồi từ phía người dùng~~
UC14	Feedback cải thiện app	Cho phép Guest phản hồi về app
UC15	Quản lý discount	Thêm, xóa, sửa discount định kỳ hàng tháng
UC16	~~Đặt món~~	~~Đặt món ăn khi có discount từ quán~~
UC17	Duyệt yêu cầu đăng ký	Cho phép Admin duyệt các yêu cầu đăng ký quán mới của Owner
UC18	Check-in, share quán ăn	Chia sẻ lên mạng xã hội Facebook
UC19	Xem bản đồ	Xem bản đồ khu vực gồm các quán ăn
UC20	Di chuyển đến vị trí hiện tại	Di chuyển camera về vị trí hiện tại dựa trên GPS
UC21	Chọn và lưu vị trí trên bản đồ	Cho phép Owner lúc đăng ký quán ăn xác định vị trí quán của họ trên bản đồ
UC22	Cập nhật dữ liệu trên Server	Cho phép User cập nhật dữ liệu mới nhất trên Server

b. Use Case Diagram



Biểu đồ Use-case Food Map

c. Traceability Matrix

	UC ID	UC 01	UC 02	UC 03	UC 04	UC 05	UC 06	UC 07	UC 08	UC 09	UC 10	UC 11	UC 12	UC 13	UC 14	UC 15	UC 16
	Priority	1	3	2	2	4	3	4	1	1	2	2	1	4	6	5	5
REQ ID																	
R01		X	X														
R02				X	X												
R03						X				X							
R04							X										
R05								X									
R06									X	X	X						
R07										X			X				
R08												X					
R09										X			X				
R10														X			
R11															X		
R12																	X
R13																X	

d. Use case Specification

Use case ID	UC4
Tên Use case	Xem thông tin món ăn
Tóm tắt Use case	Use case cho phép thực khách(Guest) xem các menu các món ăn hiện có trong quán
Luồng chính	1. Hệ thống hiển thị danh sách các quán 2. Thực khách sẽ chọn ngẫu nhiên 1 quán bất kỳ 3. Hệ thống lấy thông tin quán dựa trên dữ liệu có sẵn dưới local. 4. Menu các món ăn sẽ hiển thị theo dữ liệu đã có dưới local
Luồng phụ	1. Khu vực hiện tại không có quán ăn nào - Hệ thống sẽ hiển thị thông báo yêu cầu mở rộng bán kính tìm kiếm - Hệ thống sẽ hiển thị lại danh sách các quán 4. Quán hiện chưa có món ăn nào - Hệ thống sẽ gợi ý người dùng xem quán khác lân cận
Điều kiện tiên quyết	1. Hệ thống đã tải dữ liệu thành công từ web 2. Dữ liệu đã được lưu xuống Local database
Yêu cầu đặc biệt	Các thông tin hiển thị phải “dễ nhìn”, Thời gian load không quá 0.5s Thông tin về các món ăn phải được lưu offline

Use case ID	UC10
Tên Use case	Đăng ký thông tin quán ăn
Tóm tắt Use case	Use case cho phép Owner đăng ký thông tin quán ăn của mình.
Luồng chính	1. Owner bấm vào icon dấu + để tạo 1 quán ăn mới. 2. Hệ thống hiện thị form đăng ký. Form đăng ký gồm các thông tin bắt buộc, thông tin thêm. 3. Owner nhập các thông tin đăng ký. 4. Owner nhấn Xác nhận. 5. Hệ thống xác nhận các thông tin hợp lệ và đủ các trường bắt buộc 6. Hệ thống thông báo đã đăng ký thành công.
Luồng phụ	1. Owner nhấn quay lại khi đang điền thông tin đăng ký - Hệ thống hỏi xác nhận hủy đăng ký từ người dùng 2. Thông tin cung cấp không hợp lệ hoặc trường bắt buộc để trống. - Hệ thống thông báo lỗi và chỉ cho người dùng thấy vị trí chưa hợp lệ.
Điều kiện tiên quyết	Chủ quán đã đăng nhập thành công vào hệ thống
Yêu cầu đặc biệt	Không có

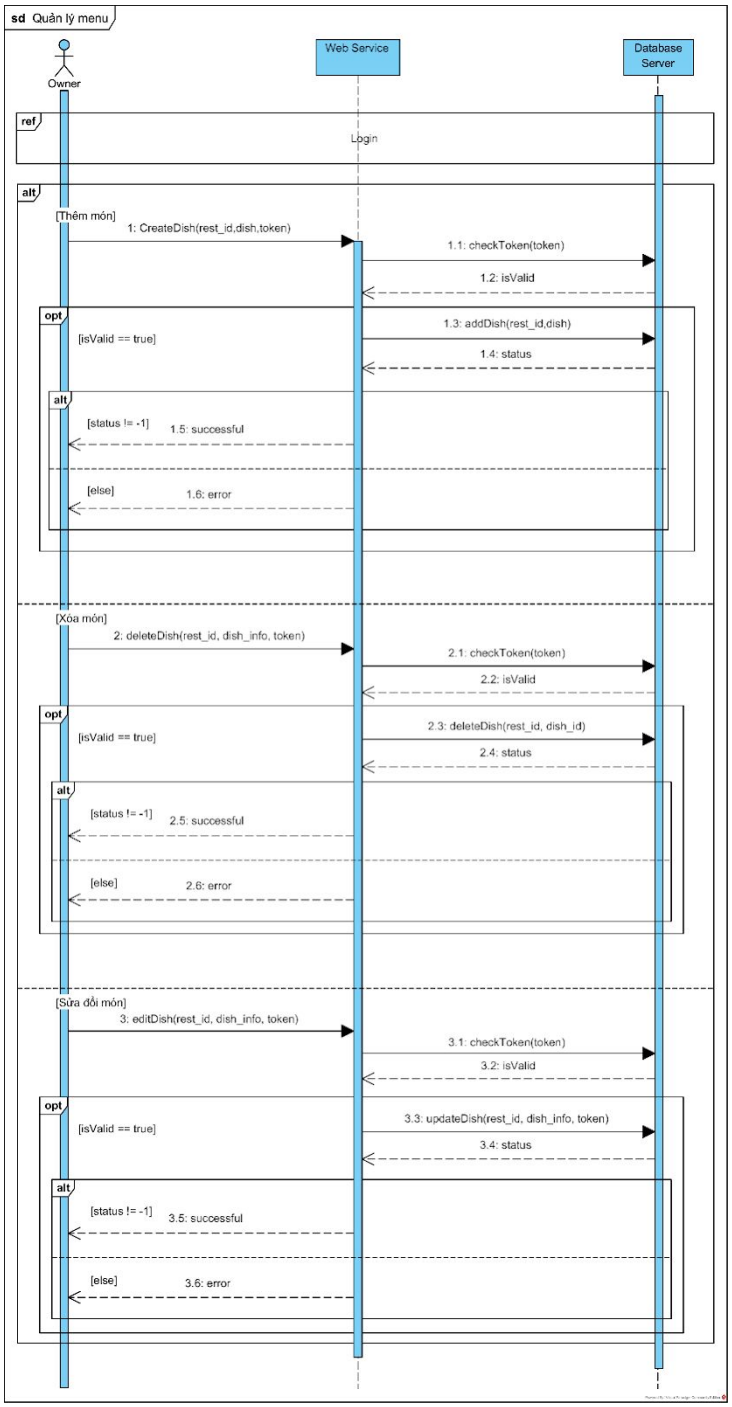
Use case ID	UC12
Tên Use case	Quản lý menu quán
Tóm tắt Use case	Cho phép Owner thêm, xóa, thay đổi thông tin món ăn trong menu
Luồng chính	1. Owner vào mục quản lý quán ăn 2. Sau đó kéo xuống vùng menu: a. Bấm vào món ăn nếu muốn chỉnh sửa thông tin b. Đè lên món ăn khi muốn xóa món ăn ra khỏi menu c. Bấm vào nút **+** để thêm món ăn mới 3. a. Điền thông tin cần chỉnh sửa cho món ăn c. Điền các thông tin cơ bản của món ăn 4. Nhấn vào dấu ✓ ở góc trên bên phải để xác nhận quá trình chỉnh sửa hoặc thêm mới

Luồng phụ	2. b. Hiển thị thông báo xác nhận xóa để người dùng xác nhận 3. Khi người dùng đã nhập thông tin thay đổi, sau đó bấm nút quay lại, thì hệ thống xuất hiện thông báo cho người dùng có muốn quay lại khi đang chỉnh sửa hay không 4. Khi người dùng nhấn ✓ để xác nhận cập nhật thông tin thì kiểm tra xem thông tin người dùng đã hợp lệ chưa. Nếu chưa thì thông báo và cho người dùng nhập lại.
Điều kiện tiên quyết	Đã đăng nhập bằng tài khoản Owner
Yêu cầu đặc biệt	Cần đảm bảo tính đúng đắn của thông tin người dùng nhập vào.

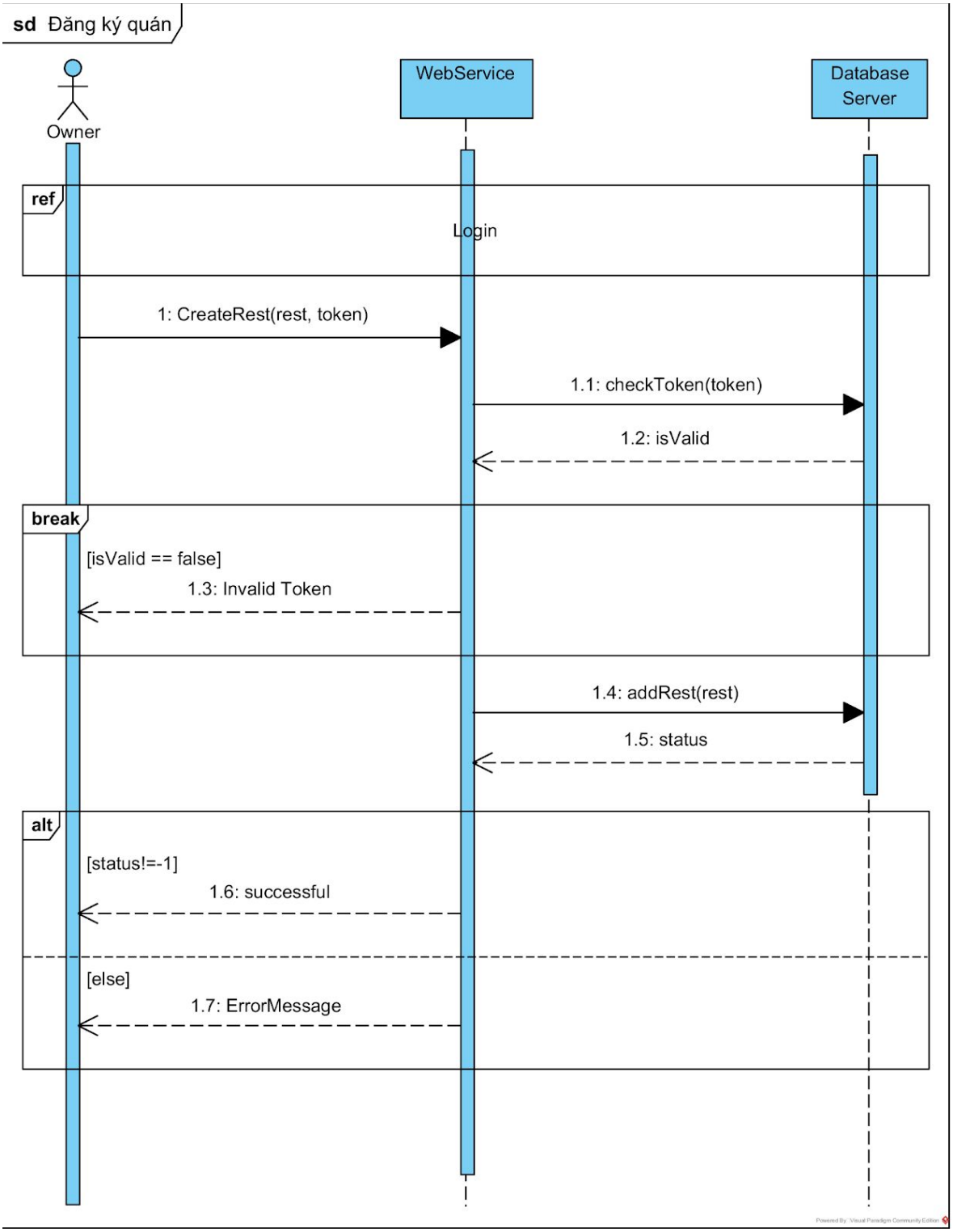
Use case ID	**UC1**
Tên Use case	Xem các quán ăn xung quanh một địa điểm
Tóm tắt Use case	Hiển thị vị trí của các quán ăn xung quanh 1 địa chỉ mà Guest nhập vào từ ô tìm kiếm
Luồng chính	1. Guest nhập địa điểm của khu vực họ muốn xem dựa trên autocomplete search. 2. Guest chọn vào 1 trong các kết quả trả về 3. Bản đồ tự di chuyển đến tọa độ tương ứng của kết quả trả về đó.
Luồng phụ	1. Điện thoại không có kết nối Internet Hệ thống sẽ thông báo người dùng kết nối vào Internet
Điều kiện tiên quyết	Kết nối Internet
Yêu cầu đặc biệt	Không có

Use case ID	UC3
Tên Use case	Xem thông tin quán ăn
Tóm tắt Use case	Xem thông tin chi tiết của 1 quán ăn gồm tên quán, thống kế nhận xét, đánh giá, số lần check-in, giờ hoạt động, danh sách các món ăn,...
Luồng chính	1. Hệ thống hiển thị các quán trên bản đồ 2. Thực khách sẽ chọn ngẫu nhiên 1 quán bất kỳ 3. Hệ thống sẽ tra cứu thông tin về quán đã được chọn. 4. Hiện thị thông tin chi tiết của quán ăn
Luồng phụ	1.1 Dữ liệu sẽ được lấy từ class Restaurant đã load trước đó 1.2 Ứng dụng phải xử lí dữ liệu để hiện thị ra màn hình 3. Ứng dụng sẽ tìm kiếm thông tin của nhà hàng mà người dùng muốn xem, rồi chuyển sang màn hình hiển thị 4. Ứng dụng sẽ nhận dữ liệu từ màn hình hiển thị danh sách quán ăn và hiển thị dữ liệu ra màn hình
Điều kiện tiên quyết	Hệ thống đã tải dữ liệu thành công từ web
Yêu cầu đặc biệt	Quá trình load dữ liệu và hiển thị không quá 0.5s Giao diện thân thiện với người dùng

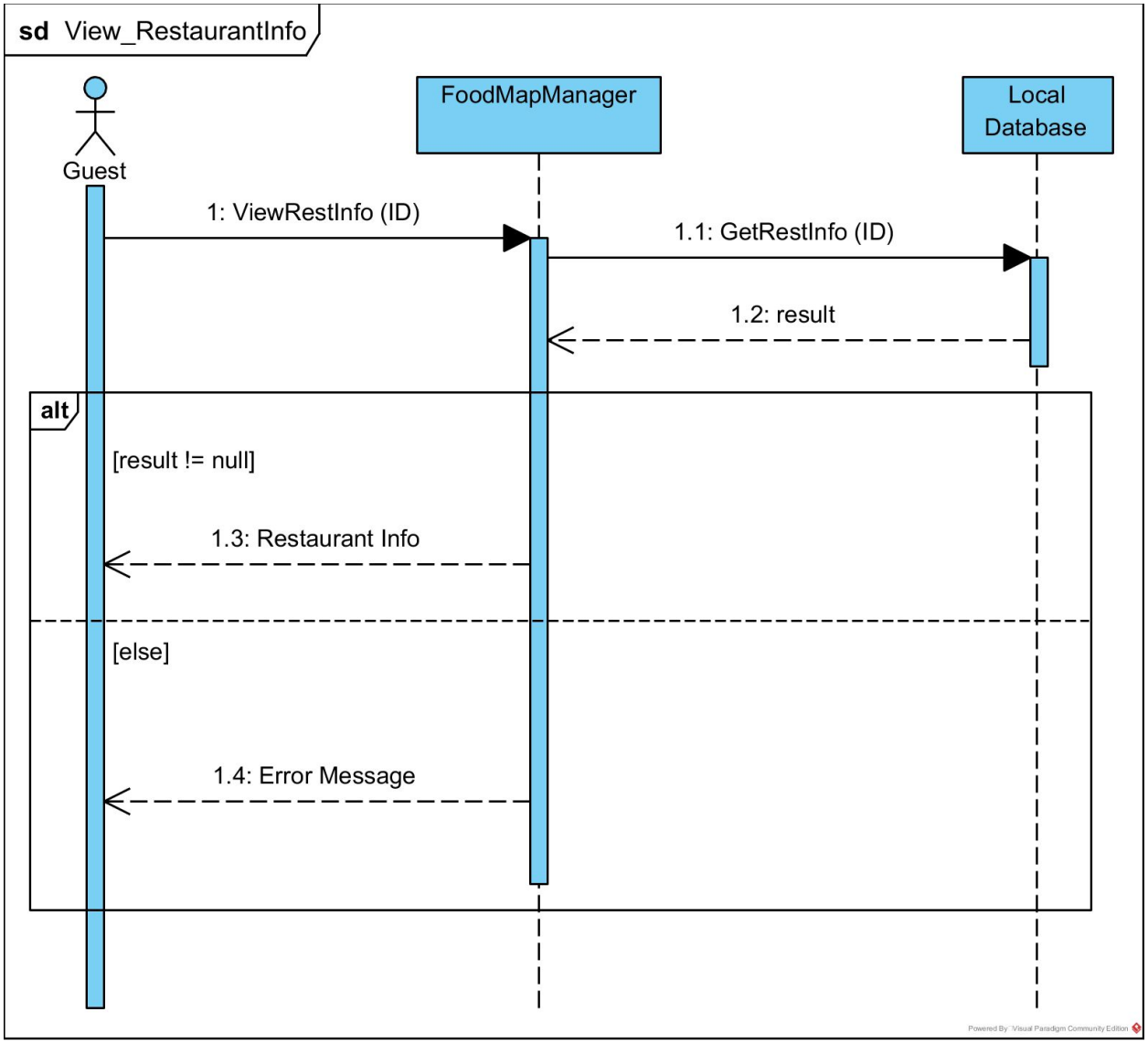
4. System Sequence Diagrams



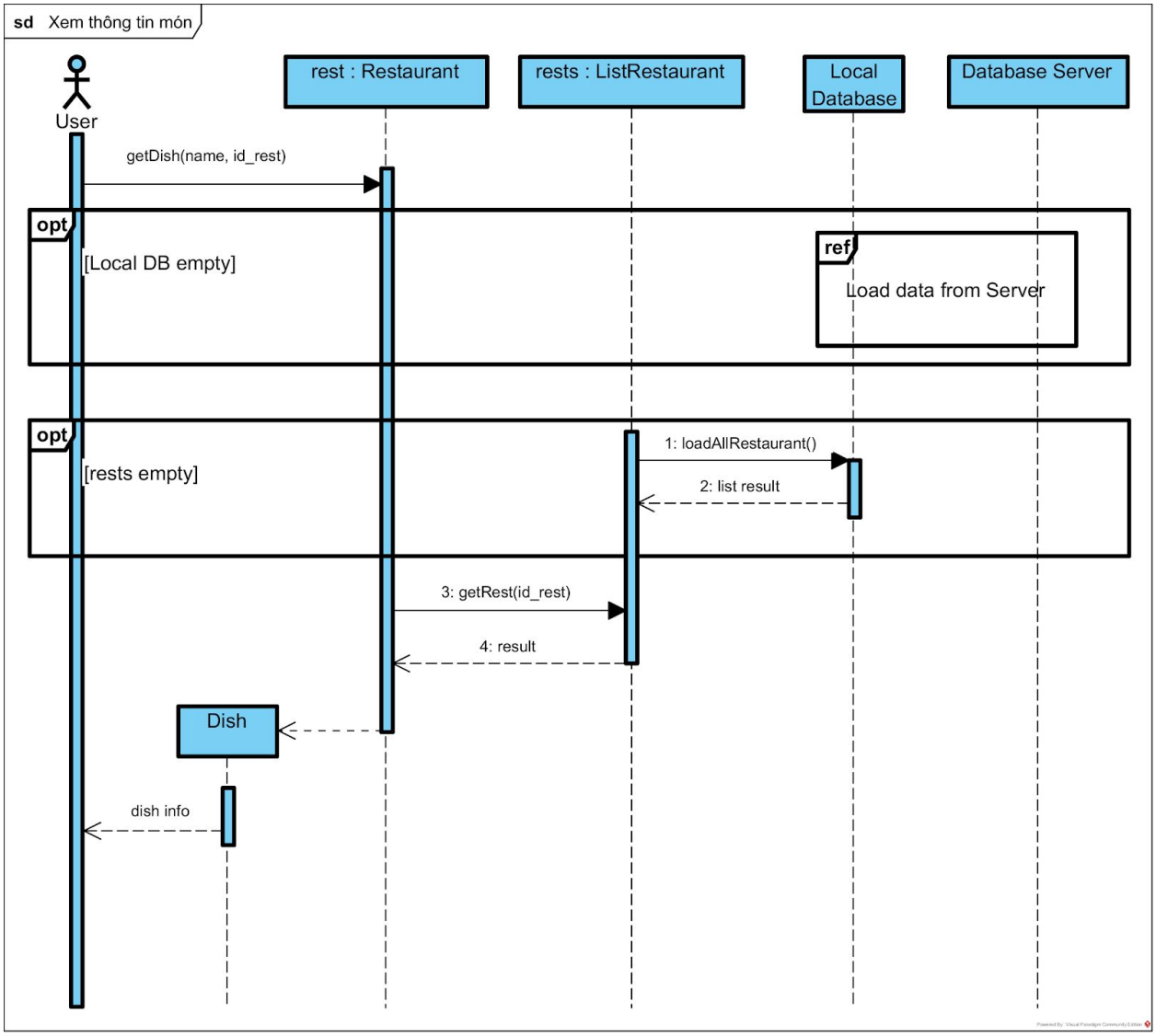
Hình SD1. Quản lý menu quán



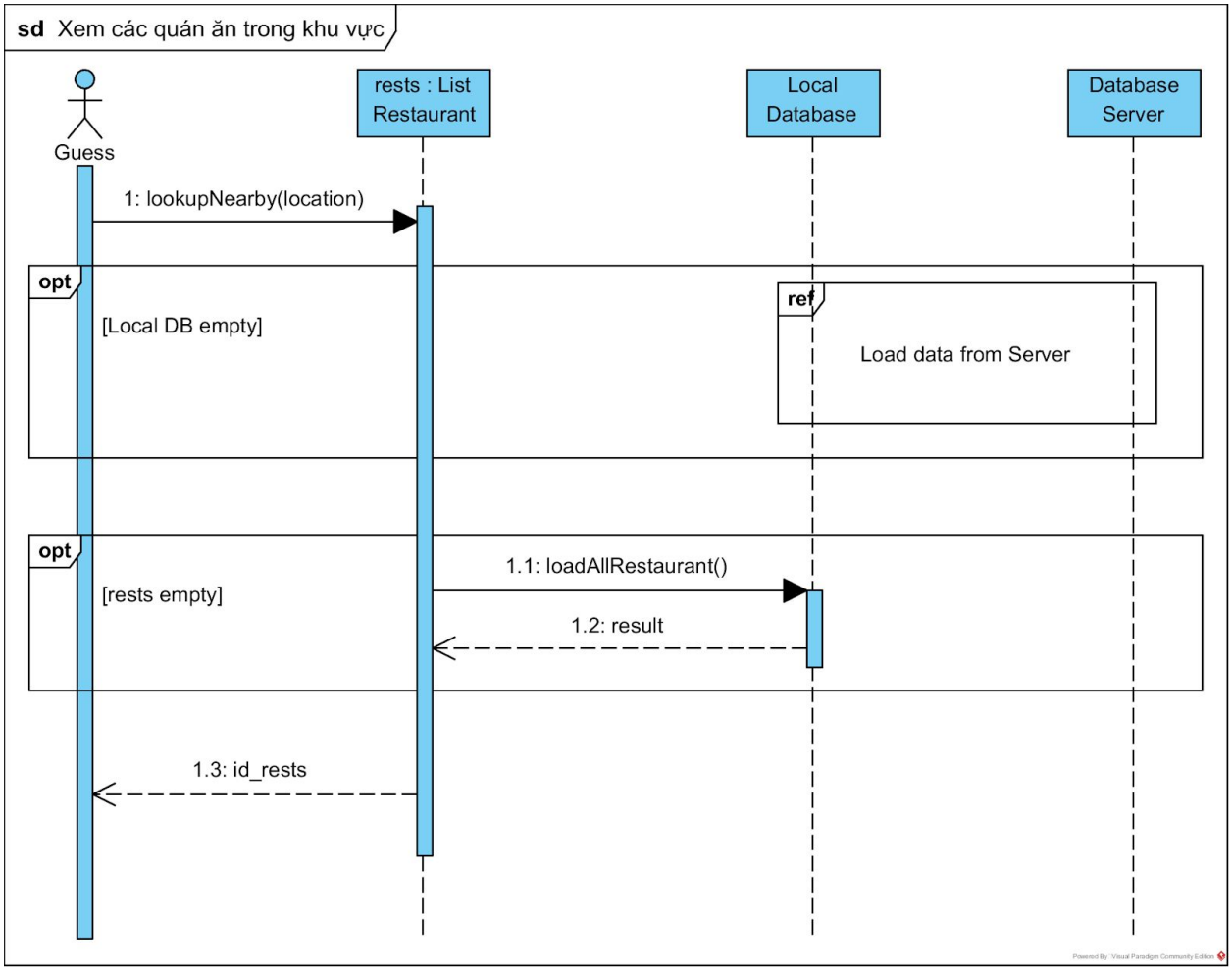
Hình SD2. Đăng ký quán



Hình SD3. Xem thông tin quán



Hình SD4. Xem thông tin món



Hình SD5. Xem các quán xung quanh

Domain Analysis

1. Concept definitions

Food Map là ứng dụng tích hợp giữa hai nhóm ứng dụng bản đồ - ăn uống.

Các chức năng cơ bản của một ứng dụng bản đồ:

ID	Chức năng
M1	Cung cấp thông tin, vị trí về các địa điểm
M2	Hỗ trợ chỉ đường người dùng giữa 2 địa điểm
M3	Xác định vị trí người dùng trên bản đồ
M4	Tìm kiếm 1 địa điểm trong ô tìm kiếm
M5	Lưu lịch sử tìm kiếm
M6	Lấy ra danh sách của 1 loại địa điểm (ăn uống, cây xăng, khách sạn,...) trong khu vực xung quanh người dùng
M7	Có thể hỗ trợ xem bản đồ offline

Các chức năng cơ bản của một ứng dụng dịch vụ ăn uống

ID	Chức năng
E1	Cung cấp thông tin cơ bản về giờ hoạt động, địa chỉ, nhóm ẩm thực phục vụ,...
E2	Cung cấp thông tin về menu các món
E3	Tương tác với quán ăn (Thích, chia sẻ, bình luận,...)
E4	Đặt món từ xa.

2. Traceability matrix

		Map Service							Eating Service			
		M1	M2	M3	M4	M5	M6	M7	E1	E2	E3	E4
Use Case	UC1			X			X	X				
	UC2			X			X	X				
	UC3	X						X	X			
	UC4							X		X		
	UC5										X	
	UC6		X					X				
	UC15											X

UC1: Xem quán ăn xung một địa điểm

- Map Service:
  - Xác định vị trí người dùng trên bản đồ
  - Lấy ra danh sách của 1 loại địa điểm (ăn uống, cây xăng, khách sạn,...) trong khu vực xung quanh người dùng
  - Có thể hỗ trợ xem bản đồ offline

UC2: Thay đổi bán kính khu vực hiển thị

- Map Service:
  - Xác định vị trí người dùng trên bản đồ
  - Lấy ra danh sách của 1 loại địa điểm (ăn uống, cây xăng, khách sạn,...) trong khu vực xung quanh người dùng
  - Có thể hỗ trợ xem bản đồ offline

UC3: Xem thông tin quán ăn

- Map Service:
  - Cung cấp thông tin, vị trí về các địa điểm
  - Có thể hỗ trợ xem bản đồ offline
- Eating Service:
  - Cung cấp thông tin cơ bản về giờ hoạt động, địa chỉ, nhóm ẩm thực phục vụ,...

UC4: Xem thông tin món ăn

- Map Service:
  - Có thể hỗ trợ xem bản đồ offline
- Eating Service:
  - Cung cấp thông tin về menu các món

UC5: Đánh giá quán ăn

- Eating Service:
  - Tương tác với quán ăn (Thích, chia sẻ, bình luận,...)

UC6: Tìm đường đi đến quán ăn

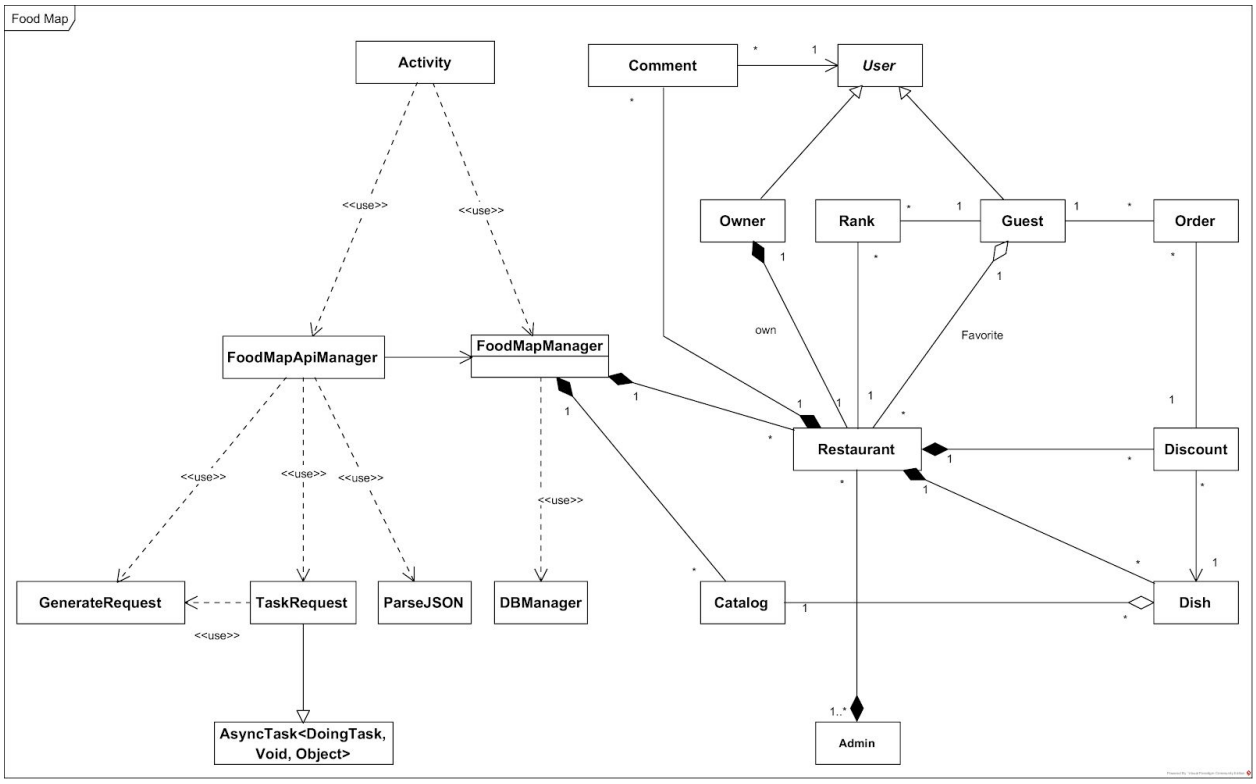
- Map Service:
  - Hỗ trợ chỉ đường người dùng giữa 2 địa điểm
  - Có thể hỗ trợ xem bản đồ offline

UC15: Quản lý discount

- Eating Service:
  - Đặt món từ xa.

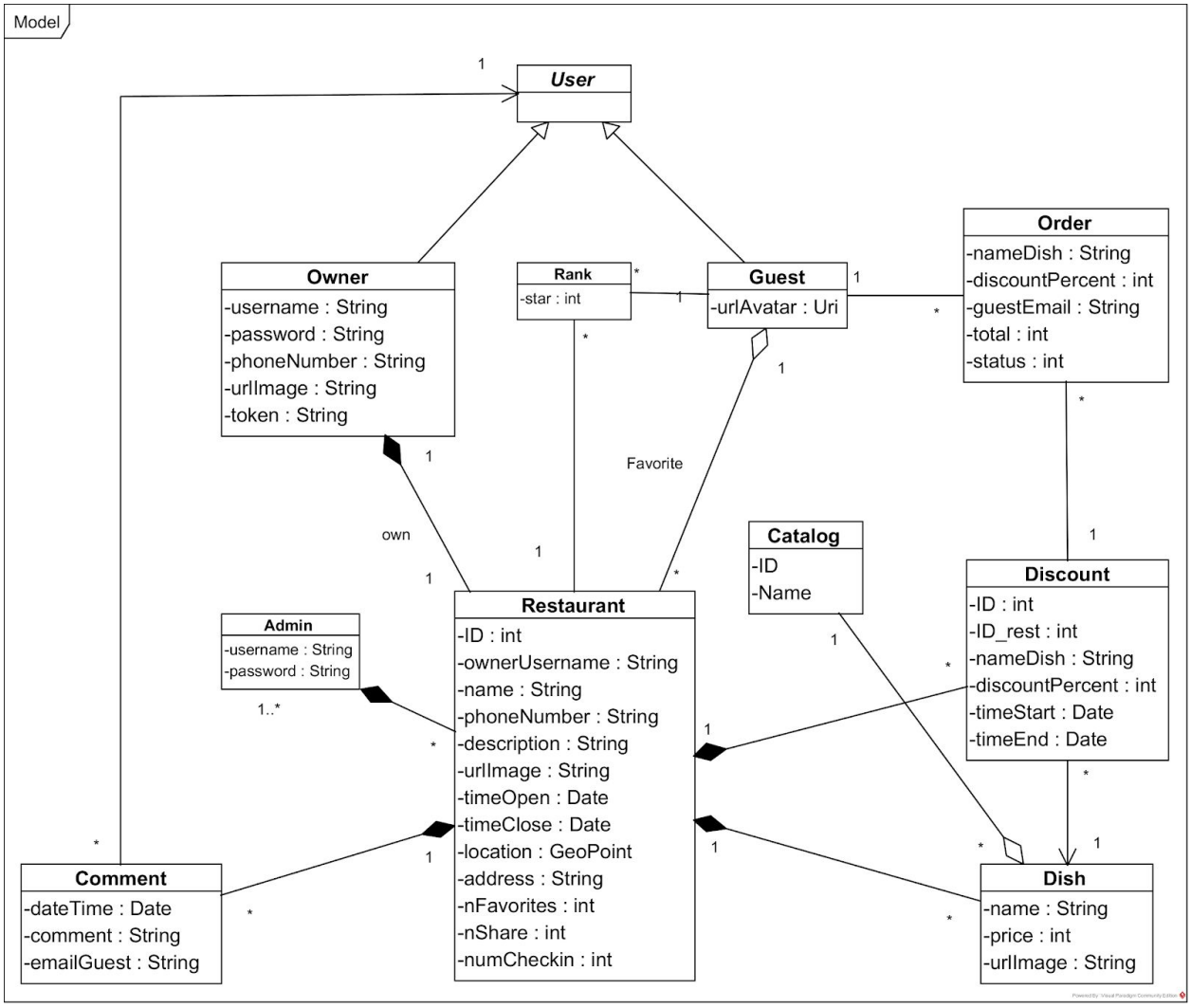
Class Diagram and Interface Specification

1. Class Diagram



Hình CD1. Tổng quát các lớp

Nhìn chung thì sẽ có 2 nhóm, 1 nhóm là các lớp Model dùng làm vật chứa đối tượng, 1 nhóm còn lại là gồm những phương thức liên quan đến các lớp Model này.



Hình CD2. Các lớp Model

Hình CD3. Lớp DBManager

Hình CD4. Lớp ParseJSON

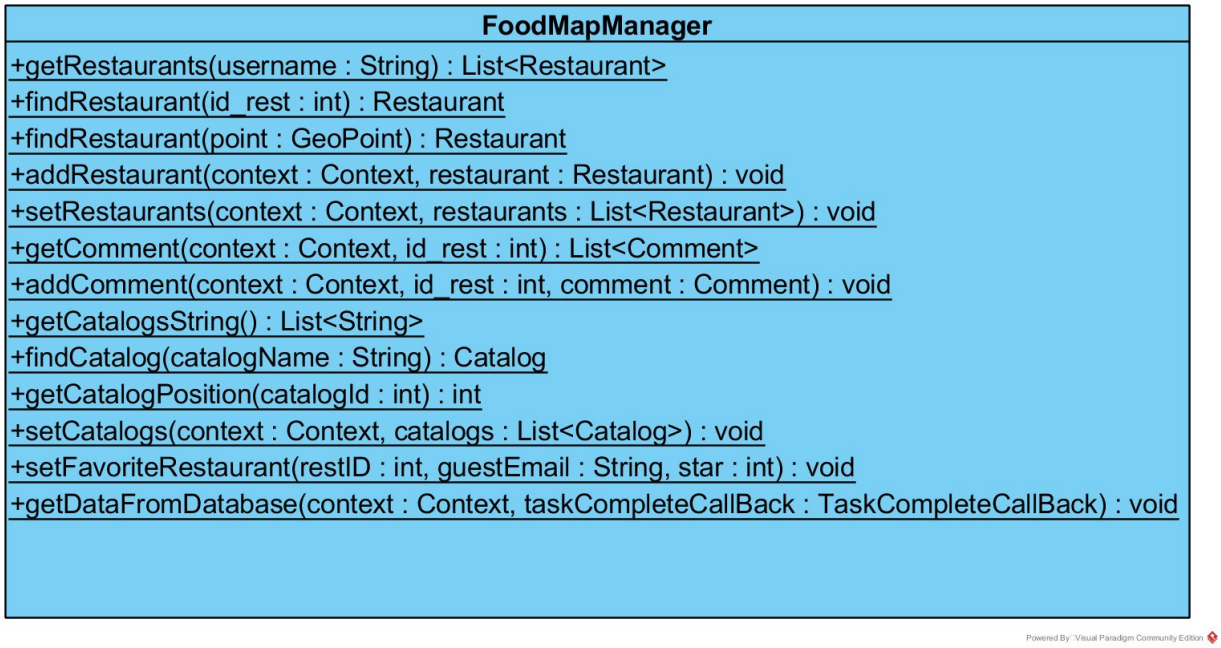
Lớp ParseJSON dùng để lấy các đối tượng có trong chuỗi JSON.

Hình CD5. Lớp GenerateRequest

Lớp GenerateRequest có nhiệm vụ là tạo các yêu cầu GET/POST để gửi lên webservice

Hình CD6. Lớp FoodMapApiManager

Đây là lớp gồm những phương thức cấp cao thực hiện thao tác với webservice API, phần cài đặt sẽ tiến hành gọi phương thức cấp thấp từ lớp GenerateRequest.



Hình CD7. Lớp FoodMapManager

Lớp này là 1 lớp trung gian, là cầu nối giữa FoodMapApiManager và DBManager. Nó chứa danh sách các nhà hàng lấy được từ trên Server và tích hợp một số phương thức hữu ích liên quan đến danh sách nhà hàng này.

2. Data Types and Operation Signatures

```
/*  Lớp Owner dùng để chứa thông tin của chủ quán ăn
 *
 *   @instance: Owner            //một thể hiện cho toàn bộ ứng dụng
 *   @username: String           //chứa tên đăng nhập
 *   @password: String           //chứa mật khẩu đăng nhập
 *   @phoneNumber: String        //chứa số điện thoại chủ quán
 *   @urlImage: String           //chứa đường dẫn hình đại diện trên server
 *   @token: String              //chứa chuỗi thông tin xác thực khi thực hiện một
thao tác với server
 *   @listRestaurant: List       //chứa danh sách nhà hàng của chủ quán
 *
 * */
```

```
public class Owner extends User {

	private static Owner instance;

	private String username;

	private String password;

	private String phoneNumber;

	private String urlImage;

	private String token;


	private List<Restaurant> listRestaurant;

	// ...

}
```

```
	/*
	*  Lớp có chức năng cung cấp các phương thức cấp cao nhằm tương tác với
Webservice và DB Server.
	*  Mỗi phương thức đều có tham số kiểu TaskCompleteCallBack, đây chính là 1
call-back function
	*  được truyền từ các Activity, cụ thể thì các Activity sẽ phải tạo mới 1 đối
tượng kèm theo
	*  call-back function chứa các việc cần làm của Activity đó sau khi kết thúc
truyền/nhận dữ liệu
	*  từ Webservice
	*
	* */
public class FoodMapApiManager {

	public static final int SUCCESS = -1;
	public static final int PARSE_FAIL = -2;
	public static final int FAIL_INFO = -3;


	/*
	*  Hàm thực hiện việc gửi yêu cầu đăng nhập lên server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @username: String                              //Tên đăng nhập của người dùng
	*  @password: String                              //Mật khẩu người dùng
	*  @taskCompleteCallBack: TaskCompleteCallBack    //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
	public static void Login(String username, String password, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
```

```
	/*
	*  Hàm thực hiện việc xóa tài khoản owner, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, FAIL_INFO, NOTINTERNET
	*
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện xóa tài khoản
	*
	*  @return void
	* */
	public static void deleteAcount(final TaskCompleteCallBack
taskCompleteCallBack){
	//...
	}


	/*  Hàm thực hiện việc tạo restaurant, nhận phản hồi và gọi callback-function
đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, FAIL_INFO, NOTINTERNET
	*
	*   @restaurant: Restaurant                          //Restaurant chứa thông
tin về nhà hàng mới
	*   @taskCompleteCallBack: TaskCompleteCallBack      //callback-function chứa
các việc cần làm sau khi tạo nhà hàng
	*
	*   @return void
	* */
	public static void createRestaurant(final Restaurant restaurant,final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*  Hàm thực hiện việc update thông tin tài khoản lên Host server, nhận phản
hồi và gọi callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND, NOTINTERNET
	*
	*   @owner: Owner                                    //Owner chứa thông tin cần
update của tài khoản
	*   @taskCompleteCallBack: TaskCompleteCallBack      //callback-function chứa
các việc cần làm sau khi update tài khoản
	*
	*   @return void
	* */
	public static void updateAccount(final Owner owner, final TaskCompleteCallBack
taskCompleteCallBack){
	//...}
```

```
	/*  Hàm thực hiện việc tạo tài khoản, nhận phản hồi và gọi callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND, NOTINTERNET
	*
	*   @username: String                                   //Username của account
	*   @password: String                                   //Password của account
	*   @name: String                                       //name của account
	*   @phoneNumber: String                                //PhoneNumber của account
	*   @email: String                                      //Email của account
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa các việc cần làm sau khi tạo Account
	*
	*   @return void
	* */
	public static void createAccount(String username, String password, String name, String phoneNumber, String email, final TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*  Hàm thực hiện việc tạo tài khoản Guest khi người dùng đăng nhập bằng facebook, nhận phản hồi và gọi callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND, NOTINTERNET
	*
	*   @guest: Guest                                       //Guest chứa thông tin của user (name, email)
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa các việc cần làm sau khi thêm tài khoản Guest
	*
	*   @return void
	* */
	public static void addGuest(Guest guest, final TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
```

```
	/*  Hàm thực hiện việc xóa Dish từ server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND, NOTINTERNET
	*
	*   @id_rest: int                                   //ID của nhà hàng chứa
Dish.
	*   @dishName: String                               //Tên Dish
	*   @taskCompleteCallBack: TaskCompleteCallBack     //callback-function chứa
các việc cần làm sau khi xóa Dish
	*
	*   @return void
	* */
	public static void deleteDish(int id_rest, final String dishName, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*
	*  Hàm thực hiện việc gửi yêu cầu thêm 1 quán ăn vào danh sách nhà hàng yêu
thích, nhận phản hồi và gọi callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @guest_email: String                         //Địa chỉ email của khách đăng
nhập từ FB
	*  @id_rest: int                                //ID của nhà hàng muốn thêm vào
danh sách
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
	public static void addFavorite(String guest_email, int id_rest, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
```

```
	/*
	*  Hàm thực hiện việc thêm 1 comment mới vào DB Server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @id_rest: int                                    //ID của nhà hàng hiện đang
comment
	*  @comment: Comment                                //Object comment chứa các thông
tin về lần comment hiện tại
	*  @token: String                                   //token của phiên đăng nhập
hiện tại
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
	public static void addComment(int id_rest, Comment comment, String token, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*
	*  Hàm thực hiện việc cập nhật 1 món ăn trên DB Server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @id_rest: int                                    //ID của nhà hàng hiện đang
comment
	*  @dish: Dish                                      //Object dish chứa các thông
tin về món ăn mới cần cập nhật
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
	public static void updateDish(int id_rest, final Dish dish, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
```

```
	/*
	 *  Hàm thực hiện việc gửi yêu cầu xóa 1 quán ăn khỏi danh sách nhà hàng yêu
thích, nhận phản hồi và gọi callback-function đính kèm theo phàn hồi
	 *  nhận được.
	*
	 *  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	 *
	 *  @guest_email: String                          //Địa chỉ email của khách đăng
nhập từ FB
	 *  @id_rest: int                                 //ID của nhà hàng muốn xóa khỏi
danh sách
	 *  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	 *
	 *  @return void
	 * */
	public static void deleteFavorite(String guest_email, int id_rest, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*
	 *  Hàm thực hiện việc gửi yêu cầu lấy danh sách nhà hàng yêu thích của khách,
nhận phản hồi và gọi callback-function đính kèm theo phàn hồi
	 *  nhận được. Khi nhận được phản hồi SUCCESS sẽ lập tức gán vào danh sách nhà
hàng yêu thích của Guest.
	 *
	 *  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	 *
	 *  @guest_email: String                          //Địa chỉ email của khách đăng
nhập từ FB
	 *  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	 *
	 *  @return void
	 * */
	public static void getFavorite(String guest_email, final TaskCompleteCallBack
taskCompleteCallBack){
	//...
    }
```

```
	/*
	*  Hàm thực hiện việc gửi yêu cầu xóa 1 quán ăn khỏi danh sách nhà hàng của chủ
quán, nhận phản hồi và gọi callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @restaurant: Restaurant                        //Object nhà hàng cần phải xóa
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
    public static void deleteRestaurant(final Restaurant restaurant, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}


	/*
	*  Hàm thực hiện việc gửi yêu cầu cập nhật 1 quán ăn trên DB Server, nhận phản
hồi và gọi callback-function đính kèm theo phản hồi
	*  nhận được.
	*
	*  Mã phản hồi gồm có SUCCESS, PARSE_FAIL, FAIL_INFO
	*
	*  @restaurant: Restaurant                        //Object nhà hàng cần phải cập
nhật lại
	*  @taskCompleteCallBack: TaskCompleteCallBack  //callback-function chứa các
việc cần làm sau khi thực hiện đăng nhập
	*
	*  @return void
	* */
	public static void updateRestaurant(final Restaurant rest, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
```

```
	/*  Hàm thực hiện việc upload ảnh lên server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, FAIL_INFO
	*
	*   @context: Context                                   //Context giao diện upload
	*   @id_rest: int                                       //id của nhà hàng
	*   @name: String                                       //tên file image để lưu
trên server
	*   @url: Stringq                                       //đường dẫn của file
upload
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void uploadImage(Context context, int id_rest, String name,
String url, final TaskCompleteCallBack taskCompleteCallBack)
	{
	//...
	}


	/*  Hàm thực hiện việc lấy dữ liệu catalog từ server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND, PARSE_FAIL, NOTINTERNET
	*
	*   @context: Context                                   //Context giao diện get
catalog
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void getCatalog(final Context context, final TaskCompleteCallBack
taskCompleteCallBack){
	//...
	}
```

```
	/*  Hàm thực hiện việc delete ảnh trên server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi gồm có SUCCESS, NOTFOUND
	*
	*   @imageURL: String                                   //Đường dẫn file image
	*   @taskCompleteCallBack: TaskCompleteCallBack      //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void deleteImage(String imageURL, final TaskCompleteCallBack
taskCompleteCallBack)
	{
	//...
	}


	/*  Hàm thực hiện việc lấy offer trên server về, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Chuỗi phản hồi sẽ chứa mã thực hiện, message và data
	*
	*   @id_rest: int                                       //id của nhà hàng
	*   @taskCompleteCallBack: TaskCompleteCallBack      //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void getOffer(int id_rest, final TaskCompleteCallBack
taskCompleteCallBack)
	{
	//...
	}


	/*  Hàm thực hiện việc thêm món ăn lên trên server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi có thể chứa SUCCESS, PARSE_FAIL,  NOTINTERNET
	*
	*   @id_rest: int                                       //id của nhà hàng
	*   @dish: Dish                                         //dứ liệu của món ăn
	*   @taskCompleteCallBack: TaskCompleteCallBack      //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void addDish(int id_rest, Dish dish, final TaskCompleteCallBack
taskCompleteCallBack){//...}
```

```
	/*  Hàm thực hiện việc thêm sao đánh giá nhà hàng lên trên server, nhận phản
hồi và gọi callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi có thể chứa SUCCESS, INVALIDREQUEST,  NOTINTERNET
	*
	*   @guestEmail: String                                 //chứa email của người
đánh giá
	*   @id_rest: int                                       //id của nhà hàng
	*   @star: int                                          //số sao đánh giá
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa
các việc cần làm sau khi upload
	 *
	*   @return void
	* */
	public static void addRank(String guestEmail, int restID, int star, final
TaskCompleteCallBack taskCompleteCallBack)
	{
	//...
	}


	/*  Hàm thực hiện việc thêm order lên trên server, nhận phản hồi và gọi
callback-function đính kèm theo phản hồi nhận được.
	*
	*   Mã phản hồi có thể chứa SUCCESS, NOTFOUND,  NOTINTERNET
	*
	*   @offer: Offer                                       //chứa dữ liệu của order
	*   @id_discount: int                                   //id của discount
	*   @taskCompleteCallBack: TaskCompleteCallBack         //callback-function chứa
các việc cần làm sau khi upload
	*
	*   @return void
	* */
	public static void addOrder(final Offer offer, int id_discount, final
TaskCompleteCallBack taskCompleteCallBack){
	//...
	}
}
```

```
/*  Lớp Restaurant dùng để chứa thông tin của quán ăn nhà hàng
 *
 *   @ownerUsername: String           //Username của chủ nhà hàng
 *   @name: String                    //Tên nhà hàng
 *   @phoneNumber: String             //chứa số điện thoại chủ quán
 *   @description: String             //Mô tả của quán ăn
 *   @address: String                 //Địa chỉ của quán ăn
 *   @urlImage: String                //chứa đường dẫn hình đại diện trên server
 *   @timeOpen: Date                  //Thời gian mở cửa
 *   @timeClose: Date                 //Thời gian đóng cửa
 *   @location: GeoPoint              //Tọa độ vị trí của quán ăn
 *   @dishes: List                    //Danh sách món ăn
 *   @comments: List                  //Danh sách các comments của quán ăn
 *   @nFavorites: int                 //Số lượt yêu thích của quán ăn
 *   num_checkin: int                 //Số lượt check in tại quán ăn
 *   @nShare: int                     //Số lượt share quán ăn trên facebook
 *   ranks: HashMap                   //Chứa email và số sao của khách hàng đánh giá
quán ăn.
 *
 * */
public class Restaurant implements Serializable {
	private int id;
	private String ownerUsername;
	private String name;
	private String phoneNumber;
	private String description;
	private String urlImage;
	private Date timeOpen;
	private Date timeClose;
	private GeoPoint location;
	private List<Dish> dishes;
	private List<Comment> comments;
	private String address;
	//so luong guest da yeu thich
	private int nFavorites;
	private int nShare;
	// bảng lưu thông tin người đánh giá
	// keyvalue: <email, star>
	private HashMap<String, Integer> ranks;

	private int num_checkin;
	//...
}
```

```java
/*
 * Lớp DBManager dùng để quản lý cơ sở dữ liệu offline cho ứng dụng.
 * Hỗ trợ các thao tác tạo, xóa (bảng), thêm, cập nhật, lấy (dữ liệu)
 *
 * @DATABASE_VERSION: int        // phiên bản của database.
 * @DATABASE_NAME: String        // tên của database
 */
public class DBManager extends SQLiteOpenHelper {

    // Database version
    private static final int DATABASE_VERSION = 3;       // old = 2

    // Database name
    private static final String DATABASE_NAME = "FOODMAP_DATABASE";

    /*
    * Hàm thêm dữ liệu thông tin của một thể loại (Catalog) vào local database
    *
    * @catalog: Catalog         // đối tượng chứa thông tin catalog cần thêm vào.
    *
    * @return void
    */
    public void addCatalog(Catalog catalog) {
    // ...
    }


    /*
    * Hàm thêm dữ liệu thông tin của một bình luận vào local database
    *
    * @id_rest: int            // id của quán ăn mà comment cần thêm vào thuộc về.
    * @comment: Comment        // đối tượng chứa thông tin của comment cần thêm vào.
    *
    * @return void
    */
    public void addComment(int id_rest, Comment comment) {
    // ...
    }


    /*
    * Hàm thêm dữ liệu thông tin của một món ăn (Dish) vào local database
    *
    * @id_rest: int            // id của quán ăn mà món ăn cần thêm vào thuộc về.
    * @dish: Dish              // đối tượng chứa thông tin của món ăn cần thêm vào.
    *
    * @return void
    */
    public void addDish(int id_rest, Dish dish) {
       // ...}
```

```
        /*
        * Hàm thêm dữ liệu thông tin của một địa điểm (của quán ăn) vào local database
        *
        * @id_rest: int            // id của quán ăn mà địa điểm cần thêm vào thuộc về.
        * @location: GeoPoint      // đối tượng lưu thông tin của comment cần thêm vào.
        *
        * @return void
        */
        public void addLocation(int id_rest, GeoPoint location) {
        // ...
        }


        /*
        * Hàm thêm dữ liệu thông tin của một điểm đánh giá của người dùng (guest)
        * đối với một quán ăn vào local database
        *
        * @id_rest: int            // id của quán ăn mà điểm đánh giá cần thêm vào thuộc
về.
        * @email_guest: String     // email của khách đánh giá.
        * @star: int               // số điểm đánh giá (1 -> 5).
      *
        * @return void
        */
        public void addRank(int id_rest, String email_guest, int star) {
        // ...
        }


        /*
        * Hàm thêm dữ liệu thông tin của một quán ăn vào local database
        *
        * @restaurant: Restaurant      // Đối tượng chứa thông tin quán ăn cần thêm
vào.
        *
        * @return void
        */
        public void addRestaurant(Restaurant restaurant) {
        // ...
        }


        /*
        * Hàm cập nhật dữ liệu của một thể loại (Catalog) trong local database.
        *
        * @catalog: Catalog  // đối tượng lưu thông tin của Catalog cần được cập nhật.
        *
        * @return void
        */
        public void updateCatalog(Catalog catalog) {
```

```
        // ...}
        /*
        * Hàm cập nhật dữ liệu của một bình luận trong local database.
        *
        * @id_rest: int            // id của quán ăn mà comment cần được cập nhật thuộc
về.
        * @comment: Comment        // đối tượng lưu thông tin của một Comment cần được
cập nhật.
        *
        * @return void
        */
        public void updateComment(int id_rest, Comment comment) {
        // ...
        }


        /*
        * Hàm cập nhật dữ liệu của một món ăn trong local database.
        *
        * @id_rest: int            // id của quán ăn mà món ăn cần được cập nhật thuộc
về.
        * @dish: Dish              // đối tượng lưu thông tin của món ăn cần được cập
nhật.
        *
        * @return void
        */
        public void updateDish(int id_rest, Dish dish) {
        // ...
        }


        /*
        * Hàm cập nhật dữ liệu của một địa điểm (của một quán ăn) trong local database.
        *
        * @id_rest: int            // id của quán ăn cần được cập nhật địa điểm.
        * @location: GeoPoint      // đối tượng chứa thông tin địa điểm cần được cập
nhật.
        *
        * @return void
        */
        public void updateLocation(int id_rest, GeoPoint location) {
        // ...
        }
```

```
	/*
	* Hàm cập nhật dữ liệu của một điểm đánh giá của người dùng (guest)
	* đối với một quán ăn trong local database.
	*
	* @id_rest: int            // id của quán ăn mà điểm đánh giá cần cập nhật thuộc
về.
	* @email_guest: String     // email của người dùng.
	* @star: int               // số điểm mà người dùng đánh giá (1 -> 5)
	*
	* @return void
	*/
	public void updateRank(int id_rest, String email_guest, int star) {
	// ...
	}


	/*
	* Hàm lấy dữ liệu của một Catalog từ local database
	*
	* @id: int          // id của Catalog cần lấy.
	*
	* @return: Catalog  // đối tượng chứa thể loại món ăn.
	*/
	public Catalog getCatalog(int id)  {

	}


	/*
	* Hàm lấy dữ liệu của tất cả các thể loại món ăn (Catalog) từ local database
	*
	* @return: List<Catalog>         // danh sách các đối tượng chứa thể loại các
món ăn (Catalog)
	*/
	public List<Catalog> getAllCatalog() {
	// ...
	}


	/*
	* Hàm lấy dữ liệu số lần được ghé vào của một quán ăn từ local database
	*
	* @id_rest: int                  // id của quán ăn.
	*
	* @return: int                   // số lượng khách check in tại quán ăn.
	*/
	public int getNumCheckin(int id_rest) {
	// ...
	}
```

```
        /*
        * Hàm lấy dữ liệu số lượng khách check in tại một quán ăn từ local database
        *
        * @id_rest: int     // id của quán ăn.
        *
        * @return: int      // số lượng khách check in tại quán ăn
        */
        public int getNumFavorite(int id_rest) {
        // ...
        }


        /*
        * Hàm lấy dữ liệu của một Catalog từ local database
        *
        * @id: int          // id của Catalog cần lấy.
        *
        * @return void
        */
        public int getNumShare(int id_rest) {
        // ...
        }


        /*
        * Hàm lấy dữ liệu thông tin của tất cả các bình luận (của một quán ăn) từ local
database.
        *
        * @id_rest: int                // id của quán ăn.
        *
        * @return: List<Comment>         // danh sách tất cả các bình luận của quán ăn.
        */
        public List<Comment> getComments(int id_rest) throws ParseException {
        // ...
        }


        /*
        * Hàm lấy dữ liệu thông tin của một món ăn từ local database
        *
        * @id_rest: int                  // id của quán ăn.
        *
        * @return: List<Dish>            // danh sách các món ăn của quán ăn.
        */
        public List<Dish> getDishes(int id_rest) {
        // ...
        }
```

```
        /*
        * Hàm lấy dữ liệu thông tin địa điểm (của một quán ăn) từ local database
        *
        * @id_rest: int                  // id của quán ăn.
        *
        * @return: GeoPoint              // đối tượng lưu thông tin địa điểm cửa quán
ăn.
      */
        public GeoPoint getLocation(int id_rest) {
        // ...
        }


        /*
        * Hàm lấy dữ liệu thông tin tất cả các điểm đánh giá của người dùng
        * đối với một quán ăn từ local database
        *
        * @id_rest: int                         // id của quán ăn.
        *
        * @return: HashMap<String, Integer>     // danh sách các điểm đánh giá.
        */
        public HashMap<String, Integer> getRanks(int id_rest) {
        // ...
        }


        /*
        * Hàm lấy dữ liệu thông tin của một quán ăn từ local database
        *
        * @id_rest: int                  // id của quán ăn.
        *
        * @return: Restaurant            // đối tượng lưu thông tin cửa quán ăn.
        */
        public Restaurant getRestaurant(int id_rest) throws ParseException {
        // ...
        }


        /*
        * Hàm lấy dữ liệu thông tin của tất cả các quán ăn từ local database
        *
        * @return: List<Restaurant>             // danh sách các đối tượng lưu thông
tin của tất cả các quán ăn.
        */
        public List<Restaurant> getAllRestaurant() throws ParseException {
        // ...
        }
}
```

System Architecture and System Design

1. Architectural Styles

Kiểu kiến trúc sử dụng là phân tầng và client-server:

Kiến trúc phân tầng:

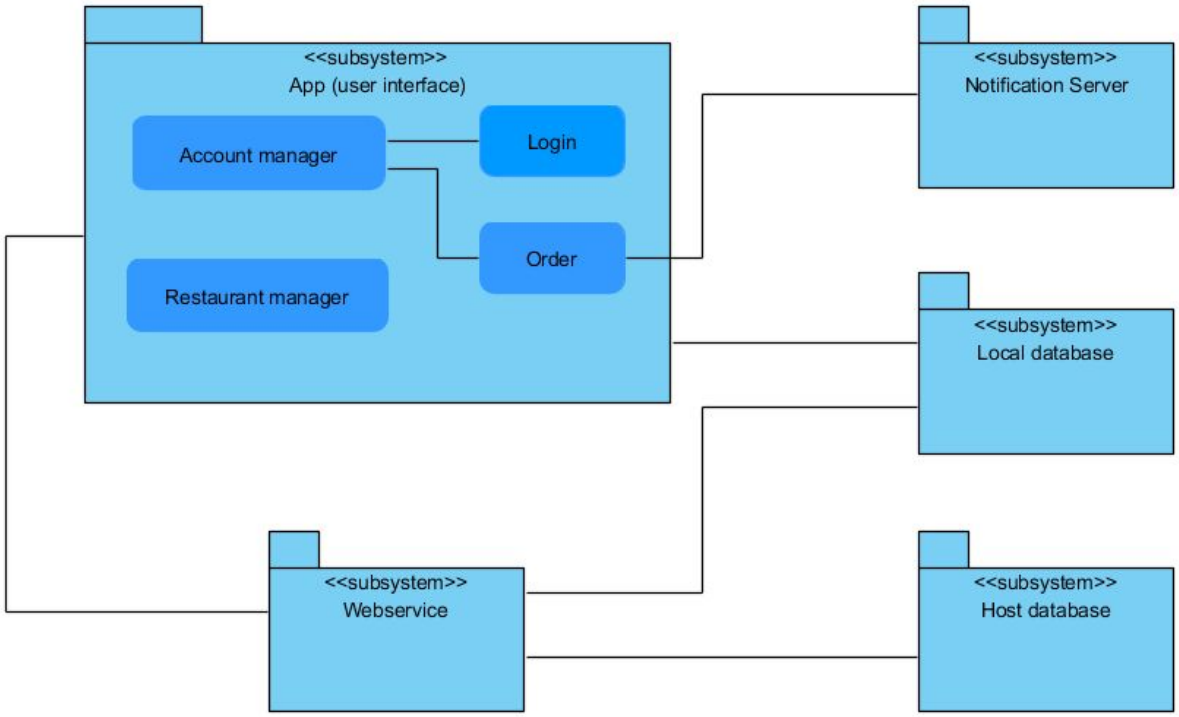
Gồm các tầng:

- User interface
- User communication
- Application functionality
- Host database

Kiến trúc client-server:

Vì client có thể truy cập dữ liệu từ nhiều nơi khác nhau và có thể tự nâng cấp dữ liệu khi server thay đổi.

2. Identifying Subsystems



Hình H1. Sơ đồ UML FoodMap Package

Hệ thống gồm 4 hệ thống con:

- App (user interface): Nơi nhận các giao tiếp của người dùng vào hệ thống.
- Notification server: Nơi quản lý các order từ người dùng gửi lên và chuyển order đó đến cho owner của nhà hàng.
- Local database: Nơi lưu trữ dữ liệu trên thiết bị của người dùng. Nó thực hiện lấy dữ liệu từ Host database thông qua Webservice.
- Host database: Nơi lưu trữ dữ liệu trung tâm của hệ thống.
- Webservice: cung cấp các dịch vụ giao tiếp giữa App với Host database.

3. Mapping Subsystems to Hardware

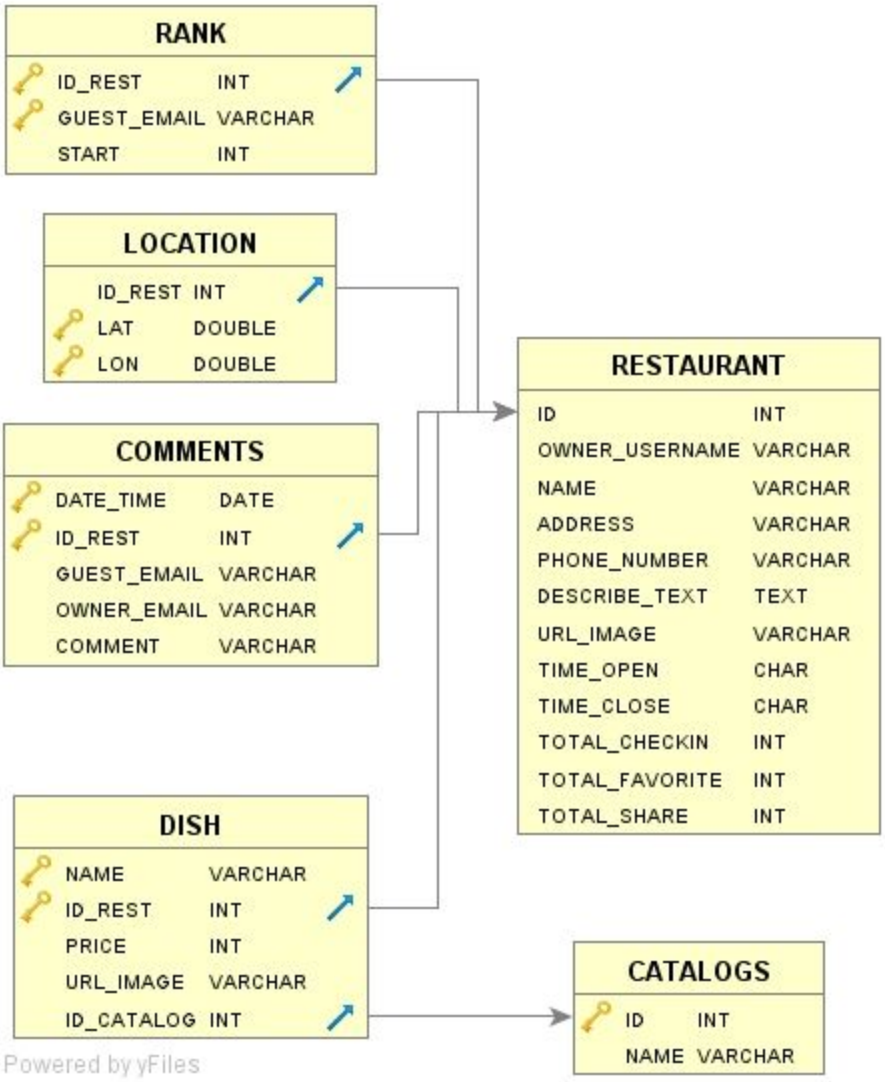
- Webservice sử dụng cơ sở dữ liệu mysql được chạy ở hosting 000webhost.com
- Nodejs server chạy trên host herokuapp được sử dụng để thông báo khi có khách đặt hàng.

4. Persistent Data Storage

Hệ thống có sử dụng cơ sở dữ liệu (được lưu trong bộ nhớ máy) để người dùng sử dụng offline.

Các đối tượng được lưu bao gồm: Catalogs, Dish, Comments, Location, Rank, Restaurant.

Hệ thống sử dụng cơ sở dữ liệu quan hệ (relational database) được cài đặt bằng hệ quản trị SQLite để tổ chức và lưu trữ dữ liệu.


RANK
ID_REST INT
GUEST_EMAIL VARCHAR
START INT
LOCATION
ID_REST INT
LAT DOUBLE
LON DOUBLE
COMMENTS
DATE_TIME DATE
ID_REST INT
GUEST_EMAIL VARCHAR
OWNER_EMAIL VARCHAR
COMMENT VARCHAR
RESTAURANT
ID INT
OWNER_USERNAME VARCHAR
NAME VARCHAR
ADDRESS VARCHAR
PHONE_NUMBER VARCHAR
DESCRIBE_TEXT TEXT
URL_IMAGE VARCHAR
TIME_OPEN CHAR
TIME_CLOSE CHAR
TOTAL_CHECKIN INT
TOTAL_FAVORITE INT
TOTAL_SHARE INT
DISH
NAME VARCHAR
ID_REST INT
PRICE INT
URL_IMAGE VARCHAR
ID_CATALOG INT
CATALOGS
ID INT
NAME VARCHAR
Powered by yFiles

Sơ đồ quan hệ

5. Network Protocol

Webservice

- Sử dụng giao thức HTTP để trao đổi dữ liệu, thông qua cái params và server trả về dữ liệu ở định dạng Json có cấu trúc chung gồm các phần sau:

    + status chứa trạng thái của request
        - 200 thực hiên thao tác thành công
        - 400 request không hợp lệ
        - 404 thực hiện thao tác không thành công
        - 444 chứng thực token thất bại

    + message là chuỗi string thông báo thông điệp ứng với các status ở trên

    + data nếu request thành công và cấu request có dữ liệu trả về sẽ được trả qua trường này. Tùy thuộc vào từng loại dữ liệu trả về mà phần data có cấu trúc khác nhau.

Websocket

- Sử dụng giao thức Websocket (một dạng nâng cao của giao thức HTTP) để truyền dữ liệu giữa các thiết bị hay còn được hiểu là truyền dữ liệu thời gian thực. Websocket server lưu trữ các kết nối từ các thiết bị, khi nhận một sự kiện từ một thiết bị bất kì thì sẽ xử lí và trả dữ liệu về một thiết bị thích hợp, dữ liệu ở định dạng JSON có cấu trúc như sau:

    + **status** chứa trạng thái của request
        - 200 thực hiện thành công
        - 404 thiết bị cần hiện đang offline

    + **message** là chuỗi string chứa yêu cầu cần được xử lí hoặc là kết quả của xử lí

6. Global Control Flow

- ***Execution orderness*:**

  Hệ thống hoạt động theo cấu trúc *event-driven*, theo đó dòng chảy của chương trình sẽ được xác định bởi các thao tác của người dùng.

- ***Time dependency*:**

  Hệ thống Food Map hoạt động theo hướng phản hồi sự kiện (event-response) từ người dùng. Ứng với mỗi tương tác từ phía người dùng sẽ luôn có kết quả thông báo trả về tương ứng

- ***Concurrency*:**

  - Hệ thống có sử dụng đa luồng trong việc lấy dữ liệu từ trên webservice về app thông qua lớp đối tượng AsynTask, ứng dụng chia làm 2 luồng cơ bản: frontend và backend:
    - Luồng frontend được xử lí trên luồng chính, cụ thể luồng này sẽ được xử lí liên tục khi ứng dụng bắt đầu hoạt động.
    - Luồng backend được xử lí song song trên luồng phụ mỗi khi có sự kiện lấy dữ liệu từ server về, luồng phụ không được xử lí liên tục mà chỉ hoạt động khi ứng dụng có yêu cầu để lấy data từ trên server về lúc này ứng dụng sẽ tạo ra một đối tượng AsynTask thông qua lớp static FoodMapApiManager.
  - Ứng dụng không có sự xung đột giữa các luồng nên sẽ không có xử lí đồng bộ.

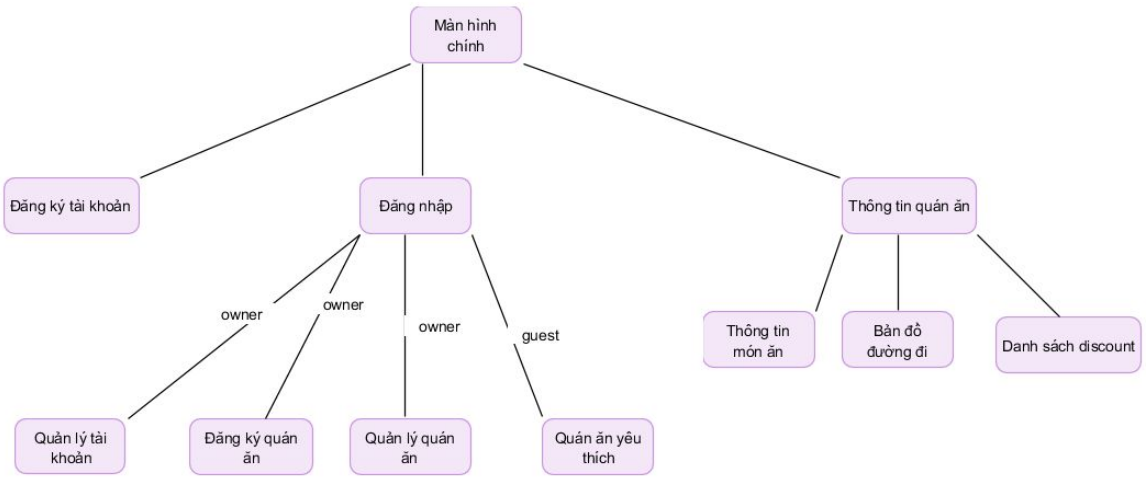
Algorithms and Data Structures

1. Algorithms
2. Data Structures

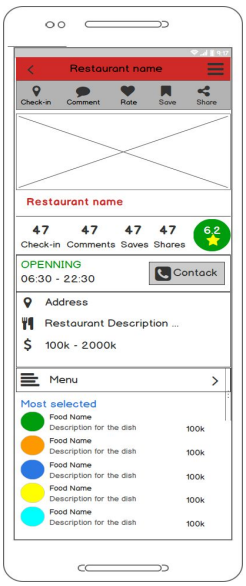
Hệ thống sử dụng ArrayList để chứa toàn bộ các nhà hàng, các món ăn,...
Nhóm chọn cấu trúc dữ liệu này là vì để đạt được hiệu năng cao, khi người dùng chọn bất kỳ 1 nhà hàng, món ăn trong danh sách.

User Interface Design and Implementation

1. Structure and Navigation

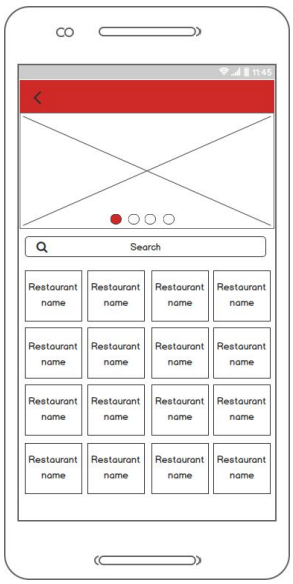


Màn hình	Mô tả
Thông tin quán ăn	Hiển thị các thông tin cơ bản về quán ăn như tên, địa chỉ, thời gian mở cửa, menu, giá cả,..
Bản đồ đường đi	Hiển thị đường đi từ vị trí người dùng đến quán ăn hoặc từ một vị trí người dùng nhập đến quán ăn.
Quán ăn yêu thích	Hiển thị những quán ăn mà khách hàng đã yêu thích
Đăng ký tài khoản	Người dùng đăng ký tài khoản chủ quán
Đăng ký quán ăn	Sau khi người dùng đã đăng nhập với tài khoản chủ quán thì “Đăng ký quán ăn” giúp chủ quán có thể đưa thông tin về quán ăn của mình lên hệ thống

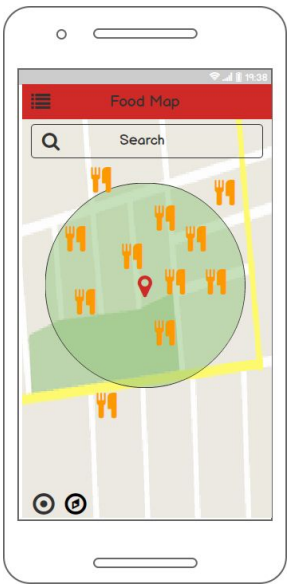


Xem thông tin quán ăn

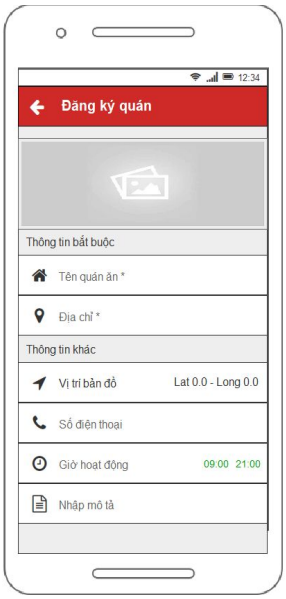
Đăng ký tài khoản

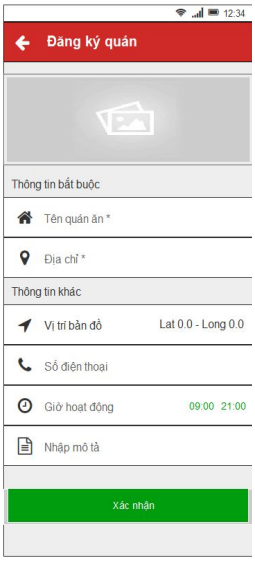


Quán ăn yêu thích



Bản đồ đường đi





Đăng ký quán ăn

2. Specification

Main screen	Purpose	Navigation
Thông tin quán ăn	Xem thông tin quán ăn	click vào maker trên bản đồ
Bản đồ đường đi	Xem đường đi đến quán ăn	Từ thông tin quán ăn, chọn vào mục chỉ dẫn đường đi ở góc dưới bên phải màn hình
Quán ăn yêu thích	Xem danh sách quán ăn người dùng đã yêu thích	Từ màn hình chính chọn mục “Quán ăn yêu thích”
Đăng ký tài khoản	Đăng ký tài khoản chủ quán trên hệ thống	Từ màn hình chính chọn đăng nhập, sau đó chọn đăng ký tài khoản chủ quán
Đăng ký quán ăn	Đưa quán ăn lên hệ thống	Vào mục “Quản lý quán ăn”, chọn mục thêm quán ăn “+”.

Design of Tests

1. Danh sách 10 đặc tả test-case:

Tên	TestCase 1
Use case tương ứng	Đăng nhập
Ngữ cảnh	Đăng nhập thành công
Dữ liệu đầu vào	Username: thedreamers Password: doanfoodmap
Đầu ra mong muốn	Đăng nhập thành công
Các bước thực hiện	B1: Nhập username, password B2: Bấm nút đăng nhập B3: Nhận thông báo đăng nhập thành công

Tên	*TestCase 2*
Use case tương ứng	Đăng nhập
Ngữ cảnh	Đăng nhập thất bại với username, password sai
Dữ liệu đầu vào	Username: thedreamers Password: 123123
Đầu ra mong muốn	Thông tin đăng nhập không chính xác
Các bước thực hiện	B1: Nhập username, password B2: Bấm nút đăng nhập B3: Nhận thông báo thông tin đăng nhập không chính xác

Tên	TestCase 3
Use case tương ứng	Đăng nhập
Ngữ cảnh	Đăng nhập thất bại quá 3 lần
Dữ liệu đầu vào	Username: thedreamers Password: 123123
Đầu ra mong muốn	Thông báo vượt quá số lần đăng nhập cho phép
Các bước thực hiện	B1: Nhập username, password B2: Bấm nút đăng nhập B3: Nhận thông báo đăng nhập thất bại B4: Tiếp tục đăng nhập thêm 2 lần tiếp theo B5: Nhận thông báo vượt quá số lần đăng nhập cho phép

Tên	TestCase 4
Use case tương ứng	Tạo tài khoản
Ngữ cảnh	Tạo tài khoản thành công
Dữ liệu đầu vào	Username: phuocpr1998.test Password: testcase4 Re-Password: testcase4 Name: Nguyễn Văn Phước Email: phuocpr1998@gmail.com SĐT: 0373788807
Đầu ra mong muốn	Đăng ký thành công
Các bước thực hiện	B1: Nhập thông tin đăng ký B2: Bấm nút đăng ký B3: Nhận thông báo đăng ký thành công B4: Xử dụng tài khoản vừa đăng ký, đăng nhập thành công

Tên	TestCase 5
Use case tương ứng	Tạo tài khoản
Ngữ cảnh	Tạo tài khoản thất bại với tên tài khoản đã được sử dụng
Dữ liệu đầu vào	Username: thedreamers Password: testcase4 Re-Password: testcase4 Name: Nguyễn Văn Phước Email: phuocpr1998@gmail.com SĐT: 0373788807
Đầu ra mong muốn	Tạo tài khoản thất bại
Các bước thực hiện	B1: Nhập username, password B2: Bấm nút đăng nhập B3: Nhận thông báo đăng ký thất bại B4: Xử dụng tài khoản vừa đăng ký để đăng nhập, và nhận thông báo thông tin đăng nhập không đúng

Tên	TestCase 6
Use case tương ứng	Tạo quán ăn
Ngữ cảnh	Tạo quán ăn thành công
Dữ liệu đầu vào	Tên quán: TestCreateRest1 Địa chỉ: 123, Nguyễn Huệ, Thành Phố Hồ Chí Minh Giờ mở cửa: 7h Giờ đóng cửa: 22h Mô tả: Nhà hàng Nguyễn Huệ Hình ảnh: đã chọn hình ảnh
Đầu ra mong muốn	Đăng ký quán ăn thành công
Các bước thực hiện	B1: Nhập thông tin quán ăn B2: Bấm nút đăng ký quán ăn

	B3: Nhận thông báo đăng ký thành công B4: Vào trang danh sách quán ăn thấy quán ăn vừa mới đăng ký

Tên	TestCase 7
Use case tương ứng	Tạo quán ăn
Ngữ cảnh	Tạo quán ăn thất bại với tên quán ăn đã trùng
Dữ liệu đầu vào	Tên quán: TestCreateRest1 (đã đăng ký thành công trước đó) Địa chỉ: 123, Nguyễn Huệ, Thành Phố Hồ Chí Minh Giờ mở cửa: 7h Giờ đóng cửa: 22h Mô tả: Nhà hàng Nguyễn Huệ Hình ảnh: đã chọn hình ảnh
Đầu ra mong muốn	Thông báo đăng ký quán ăn thất bại
Các bước thực hiện	B1: Nhập thông tin quán ăn cần đăng ký B2: Bấm nút đăng ký quán ăn B3: Nhận thông báo đăng ký thất bại B4: Vào trang danh sách quán ăn không có tên quán ăn vừa đăng ký thất bai

Tên	TestCase 8
Use case tương ứng	Thêm món ăn
Ngữ cảnh	Thêm món ăn thành công
Dữ liệu đầu vào	Tên món ăn: Cá chép hóa rồng Giá: 999000 Type: Cơm Hình ảnh: đã được chọn

Đầu ra mong muốn	Thêm món ăn thành công
Các bước thực hiện	B1: Nhập thông tin món ăn B2: Bấm nút thêm món ăn B3: Nhận thông báo thêm món ăn thành công B4: Vào menu quán ăn thấy món ăn vừa được thêm vào

Tên	TestCase 9
Use case tương ứng	Thêm món ăn
Ngữ cảnh	Thêm món ăn thất bại với tên món ăn đã tồn tại
Dữ liệu đầu vào	Tên món ăn: Cá chép hóa rồng (đã được thêm trước đó) Giá: 1999000 Type: Cơm Hình ảnh: đã được chọn
Đầu ra mong muốn	Thêm món ăn thất bại
Các bước thực hiện	B1: Nhập thông tin món ăn B2: Bấm nút thêm món ăn B3: Nhận thông báo thêm món ăn thất bại B4: Vào menu quán ăn không thấy món ăn vừa được thêm thất bại

Tên	TestCase 10
Use case tương ứng	Xóa món ăn
Ngữ cảnh	Xóa thành công
Dữ liệu đầu vào	Món ăn cần xóa
Đầu ra mong muốn	Xóa thành công

Các bước thực hiện	B1: Chọn món ăn cần xóa trong menu B2: Bấm nút xóa món ăn B3: Nhận thông báo xóa thành công B4: Vào lại menu không còn thấy món ăn vừa xóa trong danh sách món ăn

2. Phạm vi kiểm tra :

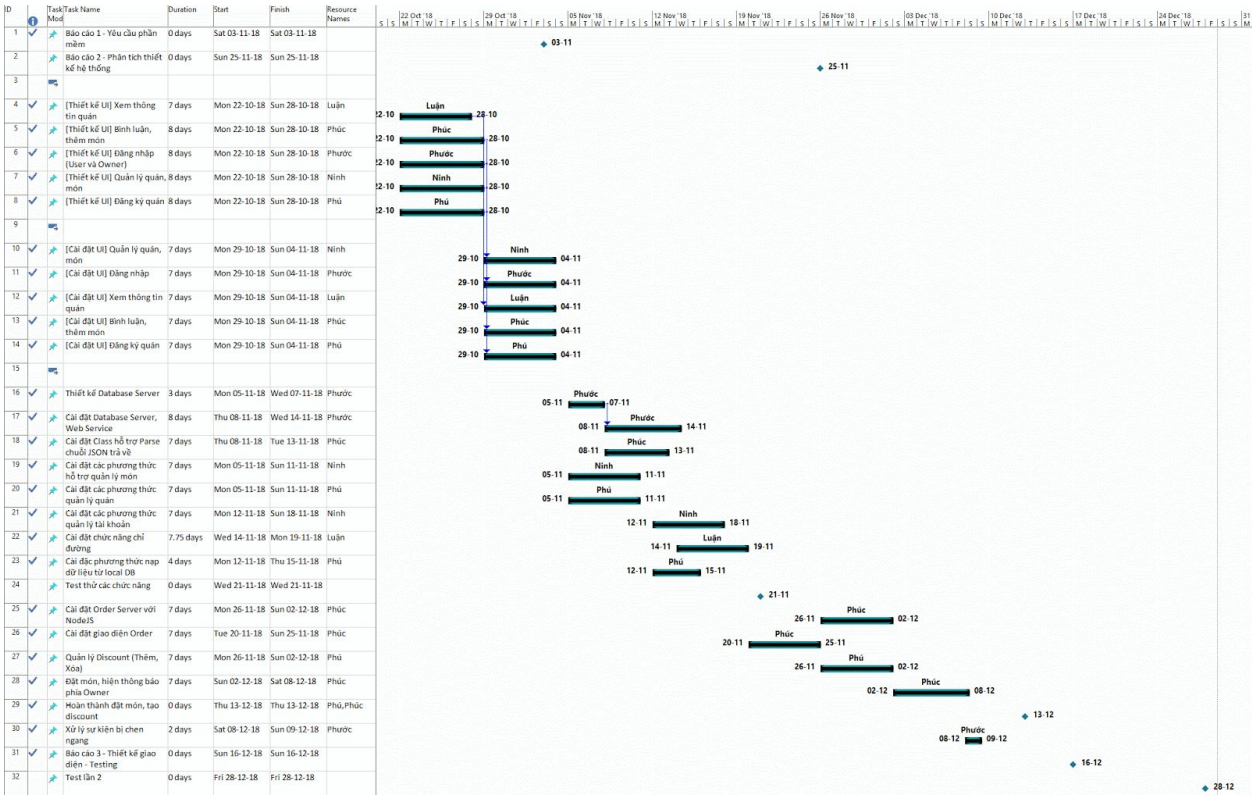
- Kiểm tra toàn bộ các hàm ở các component riêng lẻ
- Kiểm tra toàn bộ các interface của các component
- Kiểm tra toàn bộ hệ thống khi đã gắn các component lại với nhau
- Kiểm tra về mặt user interface và trả nghiệm người dùng

3. Chiến lược kiểm thử:

- Kiểm thử đơn vị: sử dụng các công cụ và framework có sẵn để kiểm thử tự động
    - JUnit: để kiểm thử unit cho component sử dụng ngôn ngữ java
    - PHPUnit: để kiểm thử unit cho component sử dụng ngôn ngữ php
    - Mocha: để kiểm thử unit cho component sử dụng ngôn ngữ nodejs
- Kiểm thử component: sử dụng một số công cụ và tự viết app để test các interface của các component
    - Webservice: sử dụng phần mềm PostMan để kiểm thử các api cung cấp bởi webservice
    - Viết app giả lập gửi lệnh request tới server nodejs để kiểm thử các thông điệp trả lời của server
- Kiểm thử toàn hệ thống: sử dụng các test-case để kiểm thử bằng tay

History of Work, Current Status, and Future Work

History of Work



ID	Task Mod	Task Name	Duration	Start	Finish	Resource Names
1		Báo cáo 1 - Yêu cầu phần mềm	0 days	Sat 03-11-18	Sat 03-11-18	
2		Báo cáo 2 - Phân tích thiết kế hệ thống	0 days	Sun 25-11-18	Sun 25-11-18	
3						
4		[Thiết kế UI] Xem thông tin quán	7 days	Mon 22-10-18	Sun 28-10-18	Luận
5		[Thiết kế UI] Bình luận, thêm món	8 days	Mon 22-10-18	Sun 28-10-18	Phúc
6		[Thiết kế UI] Đăng nhập (User và Owner)	8 days	Mon 22-10-18	Sun 28-10-18	Phước
7		[Thiết kế UI] Quản lý quán, món	8 days	Mon 22-10-18	Sun 28-10-18	Ninh
8		[Thiết kế UI] Đăng ký quán	8 days	Mon 22-10-18	Sun 28-10-18	Phú
9						
10		[Cài đặt UI] Quản lý quán, món	7 days	Mon 29-10-18	Sun 04-11-18	Ninh
11		[Cài đặt UI] Đăng nhập	7 days	Mon 29-10-18	Sun 04-11-18	Phước
12		[Cài đặt UI] Xem thông tin quán	7 days	Mon 29-10-18	Sun 04-11-18	Luận
13		[Cài đặt UI] Bình luận, thêm món	7 days	Mon 29-10-18	Sun 04-11-18	Phúc
14		[Cài đặt UI] Đăng ký quán	7 days	Mon 29-10-18	Sun 04-11-18	Phú
15						
16		Thiết kế Database Server	3 days	Mon 05-11-18	Wed 07-11-18	Phước
17		Cài đặt Database Server, Web Service	8 days	Thu 08-11-18	Wed 14-11-18	Phước
18		Cài đặt Class hỗ trợ Parse chuỗi JSON trả về	7 days	Thu 08-11-18	Tue 13-11-18	Phúc
19		Cài đặt các phương thức hỗ trợ quản lý món	7 days	Mon 05-11-18	Sun 11-11-18	Ninh
20		Cài đặt các phương thức quản lý quán	7 days	Mon 05-11-18	Sun 11-11-18	Phú
21		Cài đặt các phương thức quản lý tài khoản	7 days	Mon 12-11-18	Sun 18-11-18	Ninh
22		Cài đặt chức năng chỉ đường	7.75 days	Wed 14-11-18	Mon 19-11-18	Luận
23		Cài đặc phương thức nạp dữ liệu từ local DB	4 days	Mon 12-11-18	Thu 15-11-18	Phú
24		Test thử các chức năng	0 days	Wed 21-11-18	Wed 21-11-18	
25		Cài đặt Order Server với NodeJS	7 days	Mon 26-11-18	Sun 02-12-18	Phúc
26		Cài đặt giao diện Order	7 days	Tue 20-11-18	Sun 25-11-18	Phúc
27		Quản lý Discount (Thêm, Xóa)	7 days	Mon 26-11-18	Sun 02-12-18	Phú
28		Đặt món, hiện thông báo phía Owner	7 days	Sun 02-12-18	Sat 08-12-18	Phúc
29		Hoàn thành đặt món, tạo discount	0 days	Thu 13-12-18	Thu 13-12-18	Phú,Phúc
30		Xử lý sự kiện bị chen ngang	2 days	Sat 08-12-18	Sun 09-12-18	Phước
31		Báo cáo 3 - Thiết kế giao diện - Testing	0 days	Sun 16-12-18	Sun 16-12-18	
32		Test lần 2	0 days	Fri 28-12-18	Fri 28-12-18	

Các task được giao đa phần đều được thực hiện đúng theo dự định ban đầu, ít có chậm trễ deadline.

Current Status

Các chức năng đã hoàn thành:

- ❖ Xem thông tin của quán ăn, món ăn
- ❖ Tương tác với quán ăn: Bình luận, đánh giá, yêu thích, chia sẻ lên MXH, Check-in
- ❖ Chỉ đường đến quán ăn
- ❖ Đặt món ăn khi có đợt giảm giá
- ❖ Nhận thông báo có đơn đặt hàng từ Guest
- ❖ Đăng ký quán ăn (Chờ duyệt từ phía admin)
- ❖ Duyệt quán ăn đã đăng ký thông qua app phụ (Đăng nhập admin)
- ❖ Quản lý thông tin quán ăn (Thay đổi, xóa)
- ❖ Tạo món ăn

- Quán lý thông tin món ăn (Thay đổi, xóa)
- Quản lý thông tin tài khoản (Thay đổi mật khẩu, Cập nhật avatar)

Future Work

- Tạo một trang web cho phép quản lý, duyệt các yêu cầu đăng ký quán ăn từ phía Owner, tránh việc Owner đăng lung tung thông tin
- Cho phép Owner quản lý, phân loại, thống kê các phản hồi, comment từ phía Guest
- Cho phép Guest bình luận kèm hình ảnh, tương tác với các bình luận của Guest - Owner

References

- Visual Paradigm Community Edititon
- Microsoft Project
- Software Engineering 10 Edition
- https://www.uml-diagrams.org/use-case-diagrams.html
- https://www.uml-diagrams.org/sequence-diagrams.html
- https://www.uml-diagrams.org/package-diagrams-overview.html
- https://courses.cs.washington.edu/courses/cse331/11wi/conceptual-info/specifications.html