Report #2: System Design

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, 25/11/2018

Revision History

Date	Version	Description	Author
25/11/2018	3.0	Cập nhật kế hoạch	Ninh

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

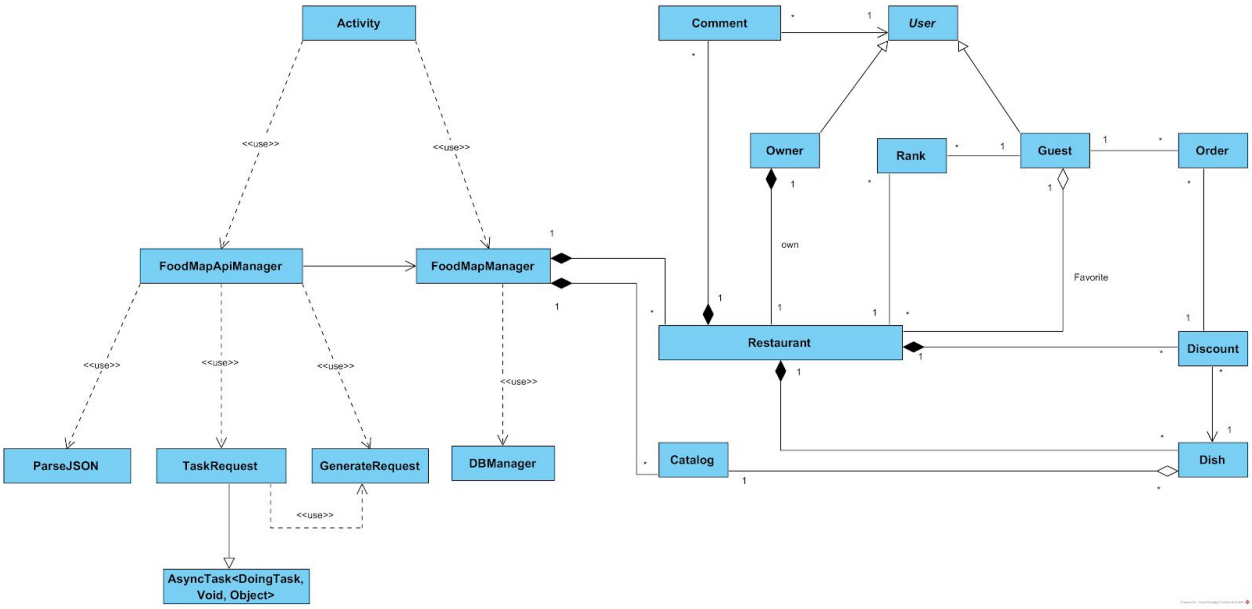
Individual Contributions Breakdown

		Luận	Ninh	Phú	Phúc	Phước
Interface Specification	Class Diagram	X	X	X	X	X
	Data Type Operation Signatures					
System Architecture and System Design	Architectural Styles	X				
	Identifying Subsystems	X				
	Mapping Subsystems to Hardware					X
	Persistent Data Storage			X		
	Network Protocol				X	X
	Global Control Flow		X	X	X	
	Hardware Requirement		X	X	X	
Algorithms and Data Structure	Algorithms					
	Data Structures					X
	Project Management		X			
	Total	3	4	4	4	4

Table of Contents

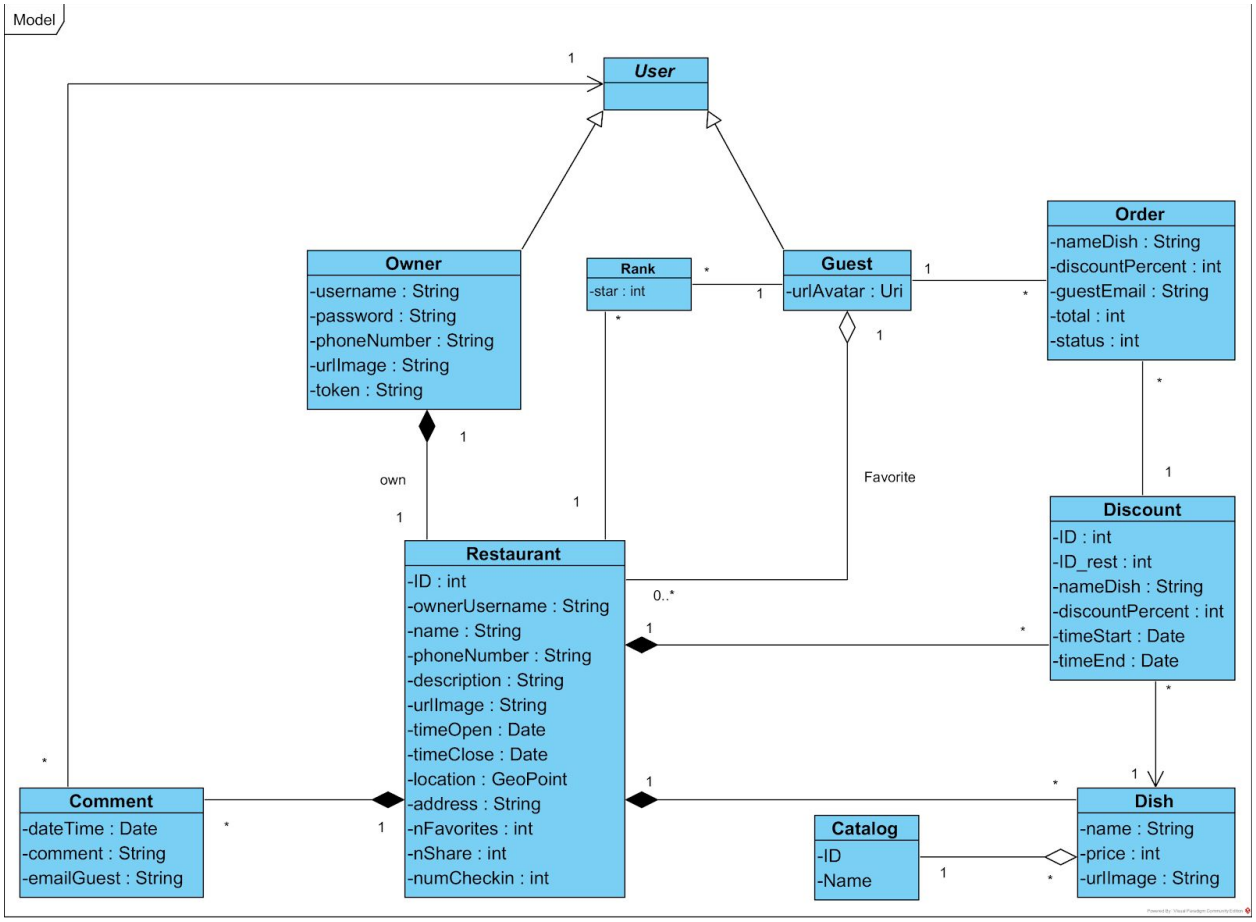
Class Diagram and Interface Specification

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

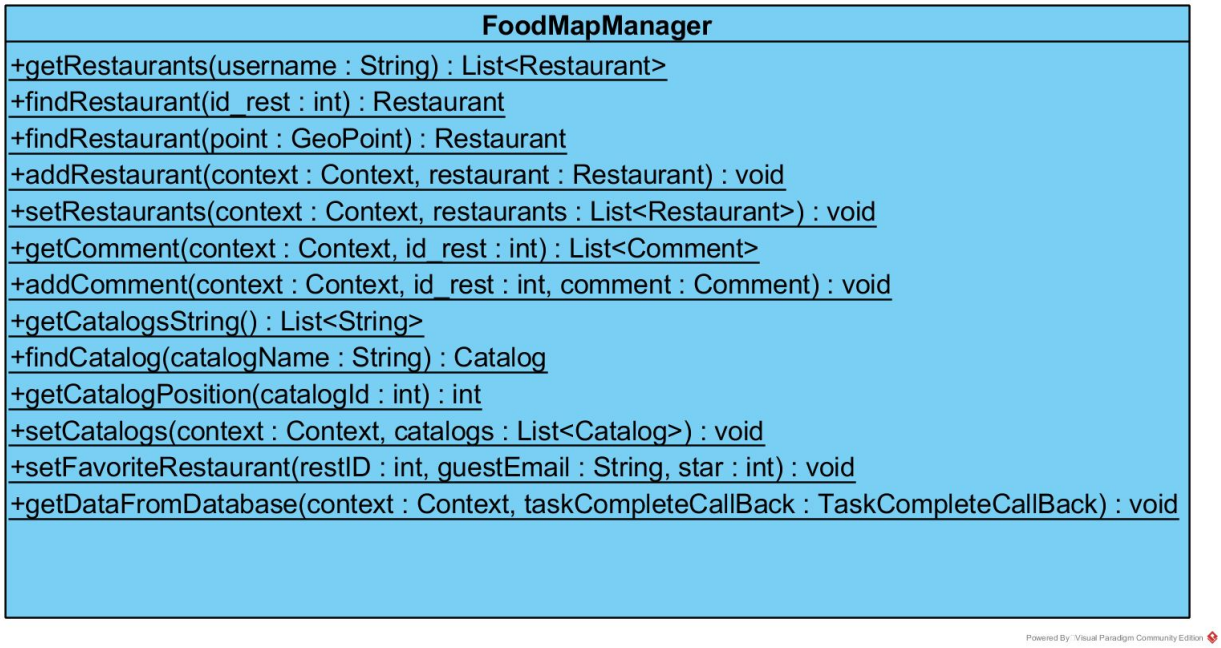
Lớp ParseJSON dùng để lấy các đối tượng có trong chuỗi JSON.

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:

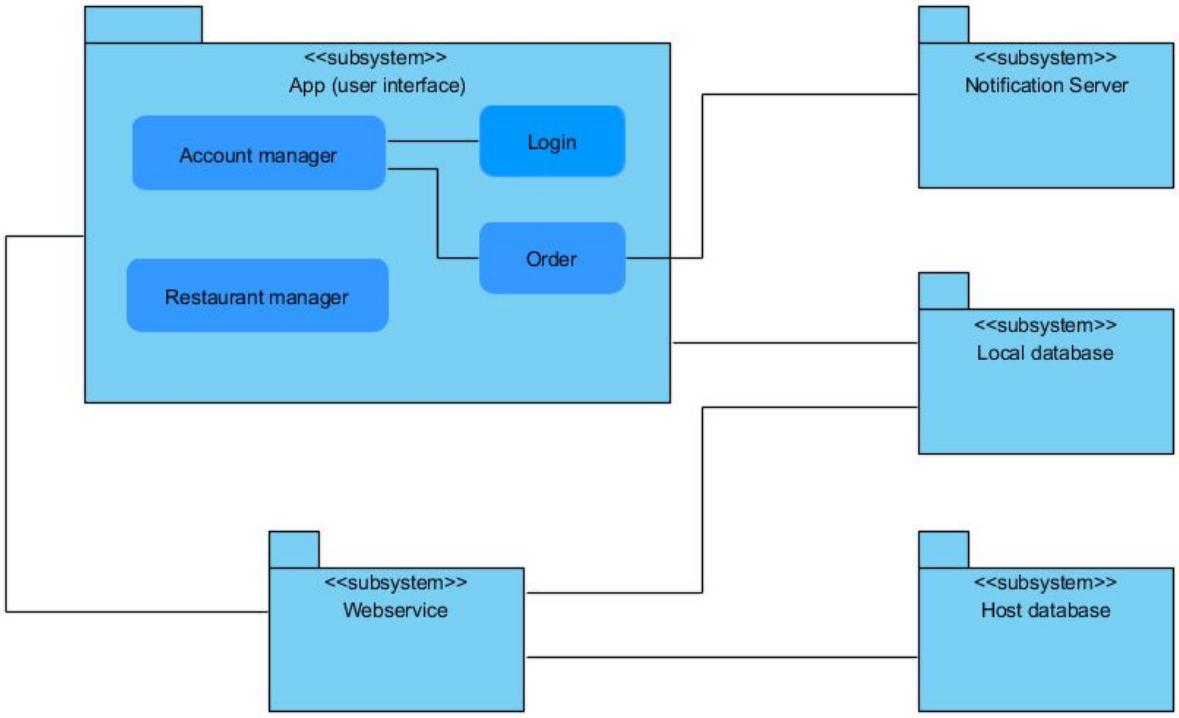
Gồm các tầng:

- 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

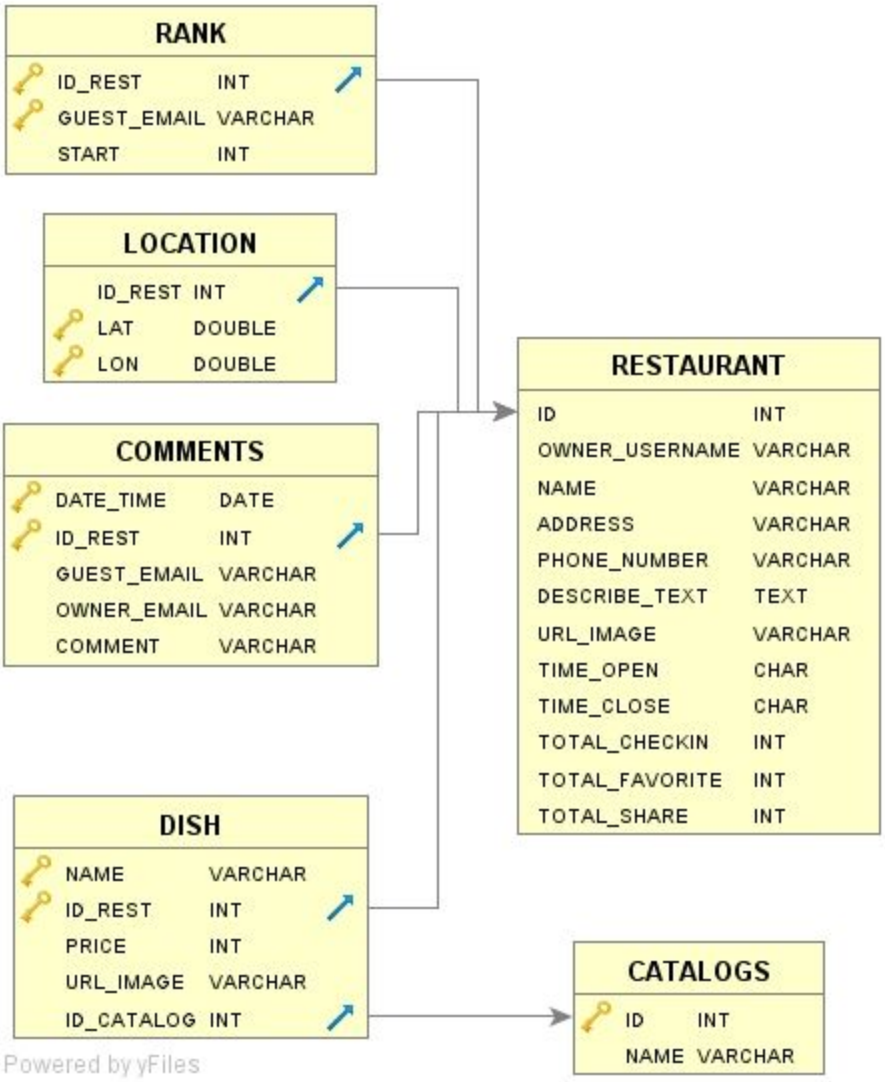
- Webservice sử dụng cơ sở dữ liệu mysql được chạy ở hosting 000webhost.com
- Nodejs server chạy trên host herokuapp được sử dụng để thông báo khi có khách đặt hàng.

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.

7. Hardware Requirements

Yêu cầu chung:

- Kết nối Internet (3G/4G/Wifi) để nạp dữ liệu lần đầu.
- GPS
- Cấp quyền đọc/ghi trên bộ nhớ.

Cấu hình điện thoại tối thiểu

Hệ điều hành	Android 4.4.2
RAM	512 MB
ROM	
CPU	Snapdragon 400

Cấu hình điện thoại đề nghị

Hệ điều hành	Android 7
RAM	2 GB
ROM	Không quan trọng
CPU	Snapdragon 425

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.

Project Management and Plan of Work

1. Merging the Contributions from Individual Team Members

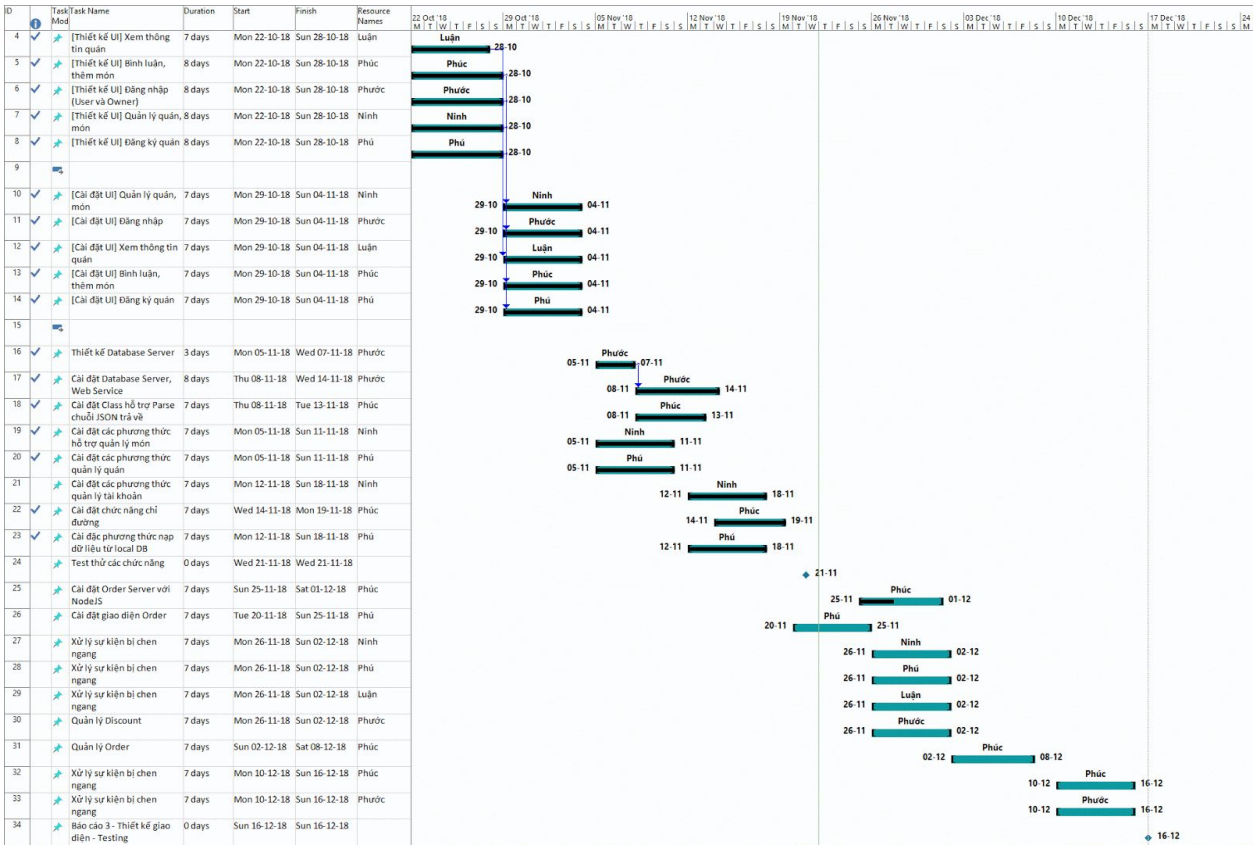
Các đóng góp	Luận	Ninh	Phú	Phúc	Phước
Tạo Webservice					X
Tạo Database trên server					X
Tạo dữ liệu giả cho DB Server		X			
Viết các phương thức hỗ trợ ParseJSON				X	
Nhúng bản đồ vào app					
Tạo database dưới local			X		
Thiết kế logo, giao diện loading	X				
Thiết kế giao diện App	X	X	X	X	X
Front-end	X	X			
Back-end			X	X	X

2. Project Coordination and Progress Report

TOTAL	11	4
Use case	**Đã cài đặt**	**Đang tiến hành**
Quản lý đánh giá		X
Quản lý menu món	X	
Quản lý thông tin quán	X	
Quản lý Discount		X
Đăng ký quán	X	
Đăng nhập	X	
Đăng ký tài khoản	X	
Feedback cải thiện app	X	
Xem các quán xung quanh khu vực	X	
Thay đổi bán kính hiển thị		X
Tìm đường đi đến quán ăn	X	
Xem món ăn	X	
Xem quán ăn	X	
Đánh giá quán ăn	X	
Đặt món		X

3. Plan of Work

.



ID	Task Mod	Task Name	Duration	Start	Finish	Resource Names
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 days	Wed 14-11-18	Mon 19-11-18	Phúc
23		Cài đặc phương thức nạp dữ liệu từ local DB	7 days	Mon 12-11-18	Sun 18-11-18	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	Sun 25-11-18	Sat 01-12-18	Phúc
26		Cài đặt giao diện Order	7 days	Tue 20-11-18	Sun 25-11-18	Phú
27		Xử lý sự kiện bị chen ngang	7 days	Mon 26-11-18	Sun 02-12-18	Ninh
28		Xử lý sự kiện bị chen ngang	7 days	Mon 26-11-18	Sun 02-12-18	Phú
29		Xử lý sự kiện bị chen ngang	7 days	Mon 26-11-18	Sun 02-12-18	Luận
30		Quản lý Discount	7 days	Mon 26-11-18	Sun 02-12-18	Phước
31		Quản lý Order	7 days	Sun 02-12-18	Sat 08-12-18	Phúc
32		Xử lý sự kiện bị chen ngang	7 days	Mon 10-12-18	Sun 16-12-18	Phúc
33		Xử lý sự kiện bị chen ngang	7 days	Mon 10-12-18	Sun 16-12-18	Phước
34		Báo cáo 3 - Thiết kế giao diện - Testing	0 days	Sun 16-12-18	Sun 16-12-18	

Hình P1. Kế hoạch hiện tại và sắp tới trong tương lai

4. Breakdown of Responsibilities

TOTAL	17	14	10	11	12
	Phước	Phúc	Phú	Ninh	Luận
AddDishActivity	X				
CheckInActivity					X
ChooseLocationActivity	X				X
EditDishActivity		X	X	X	
EditRestaurantActivity			X	X	
FavoriteRestaurantActivity			X		X
LoadingActivity					X
LoginGuestActivity	X	X			
LoginOwnerActivity	X	X			
MainActivity	X		X		X
MapActivity					X
OrderListActivity		X	X		
RegisterRestaunrantActivity	X		X		
RegisterOwnerActivity	X				
RestaurantInfoActivity				X	X
RestaurantMangeActivity	X			X	
CommentActivity	X	X			
ManageAccountActivity				X	
DBManager			X		
FoodMapManager	X	X	X	X	X
FoodMapApiManager::Login()	X				
FoodMapApiManager::deleteAccount()			X		
FoodMapApiManager::createRestaurant()	X				
FoodMapApiManager::updateAccount()		X			
FoodMapApiManager::createAccount()				X	

FoodMapApiManager::addGuest()	X				
FoodMapApiManager::deleteDish()				X	
FoodMapApiManager::addFavorite()					X
FoodMapApiManager::addComment()	X				
FoodMapApiManager::updateDish()		X			
FoodMapApiManager::deleteFavorite()					X
FoodMapApiManager::getFavorite()					X
FoodMapApiManager::deleteRestaurant()				X	
FoodMapApiManager::updateRestaurant()				X	
FoodMapApiManager::uploadImage()	X				
FoodMapApiManager::deleteImage()				X	
FoodMapApiManager::getOffer()		X			
FoodMapApiManager::addDish()	X				
FoodMapApiManager::addRank()		X			X
FoodMapApiManager::addOffer()			X		
FoodMapApiManager::getCatalog()		X			
OrderService		X			
WebService	X				
ParseJSON		X			
TaskRequest		X			

References

Cách viết đặc tả cho lớp, phương thức:
https://courses.cs.washington.edu/courses/cse331/11wi/conceptual-info/specifications.html