Lanna Cooking Recipe on Mobile Software Design

By

Name: Kantapong Orprayoon   **ID:** 532115008

Name: Khomkrib   Singh   **ID:** 532115015

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

_____

Mr. Kittitouch Suteeca

Software Design

Cross Ref. VSE-29110	Coverage Level:	Version :
	V.3.0	V.3.0

Process Ownership	Approving Authority
CAMT	
Scope	**Approved Date**
SENIOR PROJECT	

Document History				
Version Number	**Record Date**	**Prepared/ Modified By**	**Reviewed By**	**Change Details**
V.1.0	2/07/2013	KO,KS	KO,KS	-Add introduction -Overall description -System architecture -Add User interface design
V.1.1	5/22/2013	KO,KS	KO,KS	-Add detailed design -Add data architecture -Edit User interface design
V.1.2	6/04/2013	KO,KS	KO,KS	-Edit Overall description
V.1.3	7/02/2013	KO,KS	KO,KS	-Edit PHP Class diagram pictures and description. -Edit Sequence diagrams SQ: 19-36
V.2.0	8/06/2013	KO,KS	KO,KS	-Edit web application class diagram. -Edit mobile application class diagram. -Edit web application sequence diagram. -Edit ER diagram. -Edit web UI design. - Delete **Traceability record** and move it to Traceability document.
V.2.1	9/08/2013	KO,KS	KO,KS	-Edit in **2.2 Product feature** to add more information about feature and add number of each feature. -Change name of both class Admin and AdminsController to User and UsersController in class diagram and class diagram description. -Add class ShopsController description. -Edit in class diagram to add id of each class diagram and traceability -Change name of admins tabel to users -Set float number to (10,6)

V.3.0	3/10/2013	KO,KS	KO,KS	- Edit ER diagram - Edit Seqeunce diagram - Edit Class Diagram and Description - Edit Mobile UI design

Document develop by

....

(Kantapong Orprayoon)

....

(Khomkrib Singh)

(3 October 2013)

Table of Contents

1. Introduction

1.1 Objective

Lanna Cooking Recipe on Mobile is an application run on android operation system; this application is support English language. Which is main purpose to promote Northern Thailand food by providing Lanna recipe as all food menu in our application, save time to prepare food, easy to understand with recipe information's such as food ingredients, cooking, shopping list and the location that any ingredients are sale, synchronize Facebook and Google function to work with our application. This application provides users can be share food information's, and also make interaction with users by shaking for random food recipe.

This Software Design Document is contains with a complete description of the design system for developer can develop Mobile application for Lanna Cooking Recipe. By show the scope of this project into picture which is providing the functions that helps all users of this application to get the comfortable ways when they need to interaction between customer and administrator.

1.2 Intended Audience and Reading Suggestions

This Software Requirement Specification documentation is intended to be used for anyone that relate with Lanna Cooking Recipe on Mobile system. The documentation will benefit to those who develop the system and those who use the system followed:

Development team
- The requirement specification can be use to define the scope of the application and help everyone to have the same understanding.
- Help to verify and validate whether the development team follow the requirement specification defined in this documentation or not.
- Help guiding team to right direction or the way that application should do.
- Help achieving the highest quality of the product.
- Useful to be used as a reference for discussion about the system.

Customer
- Ensure that the development team will understand requirements, achieve the requirement, and satisfy the customer needs.
- Help customer understand the quality, scope, and limitation of the system defined in the document.

1.3 Project Scope

The objective of this software requirement specification is to specify requirement to establish the application that:

- The application is Lanna Cooking Recipe on Mobile.
- Lanna Cooking Recipe on Mobile is android application.
- Lanna Cooking Recipe on Mobile supports English language.
- Lanna Cooking Recipe on Mobile provides member system.
- Lanna Cooking Recipe on Mobile provides recipe information management system.
- Lanna Cooking Recipe on Mobile provides location management system.
- Lanna Cooking Recipe on Mobile provides shopping list management system.
- Lanna Cooking Recipe on Mobile can be work with Facebook API to share menu to user account.
- Lanna Cooking Recipe on Mobile can be work with Google Map.
- Lanna Cooking Recipe on Mobile can be make interaction with users by random menu function.
- Lanna Cooking Recipe on Mobile has display recipe information on application.
- Lanna Cooking Recipe on Mobile has finding ingredient store location function.

1.4 Acronyms and Definitions

Acronyms

LCR = Lanna Cooking Recipe
PMP = Project Management Plan
SDD = Software Design Document
SRS = Software Requirment Specification
KO = Kantapong Orprayoon
KS = Khomkrib Singh
DB = Database
SQ = Sequence Diagram
UI = User Interface

2. Overall Description

2.1 Product Perspective

The Lanna Cooking Recipe is a mobile application base on android OS. So this application helps users save their time to prepare food by providing features such as shopping list feature that is contain list of needed ingredient, ingredient location finding which is feature that using Google map to find where is needed ingredient store location and random menu food system by shaking.

2.2 Product Features

The Lanna Cooking Recipe is separated into two parts. The first part is the basic feature and second part is the special features that make Lanna Cooking Recipe become unique. The following list is the description of each part in more detail.

First part: Server Features

These are 3 server features defined in this part, which are

Server Feature 1:
- **Manage recipe detail information**

  The system provides administrator can add, edit, and delete recipe detail information.
- **Manage ingredient detail information**

  The system provides administrator can add, edit, and delete ingredient detail information.
- **Manage ingredient for each menu**

  The system provides administrator can add and delete ingredient for each menu.
- **Manage cooking method information**

  The system provides administrator can add, edit, and delete cooking method information.

Server Feature 2:
- **User Register Feature**

  The system requires to registers to be a member before use this application.
- **User Authentication Feature**

  The system provides to login to application and logout from the system for user.

  The system provides to login to web service and logout from web service for administrator.

Server Feature 3:

- **Manage ingredient store location information**

The system provides administrator can add, edit, and delete ingredient store location information.

Second part: Mobile Application Features

These are 9 mobile application features defined in this part, which are

Mobile Feature 1: User interface for all frame design

The system provides usability interface to the user. For more comfortable and understanding to use the application

Mobile Feature 2: View information of recipe

The system provides information of recipes which are how to cook and list of needed ingredient.

Mobile Feature 3: Share by Facebook feature

The system provides user can share food menu on user Facebook account.

Mobile Feature 4: Ingredient information feature

The system provides user can view ingredient information and description which are name, description and picture.

Mobile Feature 5: Finding Ingredient Store Location Feature

The system provides user convenience to retrieve ingredient's store location by using Google Map API.

Mobile Feature 6: Shopping List Feature

The system provides shopping list function users can add menu to shopping list that store the list of ingredients in each menu.

Mobile Feature 7: Find direction from user to ingredient store location

The system provides can user can get direction between user and ingredient shop by using Google map API.

Mobile Feature 8: Random Menu Feature

The system provides interaction with user Random Feature that user can random by shaking a device.

2.3 User Classes and Characteristics

The system divided users into 2 groups. Information and characteristics of each group are listed below.

User

This group of person who already register for the system, they can use the application for help to save their time and get more comfortable. The user is able to:

- **User registration feature**
- **View information of recipe**
- **Authentication feature**
- **Finding ingredient store location feature**
- **Shopping list feature**
- **Random menu feature**
- **Share feature**
- **Ingredient information feature**

Administrator

This group of person who has highest privilege in the system. So they gain the management function for maintain the system. Administrator is able to:

- **Authentication feature**
- **Manage recipe detail information**
- **Manage ingredient detail information**
- **Manage ingredient for each menu**
- **Manage cooking method information**
- **Manage finding ingredient store location information**

2.4 Operation Environment

This system operates on web application and android mobile application when customers request information, android mobile application will request to web application.

Server Environment

This system supports windows server platform which capable of web-server, database-server and internet connection. The system also requires at least 10 gigabytes for a database and at least 4 gigabytes of main memory (RAM) for managing with a lot of customer's session at peak hour.

Client Environment

The system supports only android platform which capable of internet connection. Customer may need the at least version 2.2 of android in order to make the system can run properly and also require GPS for getting correctly position form Google map API. The system also requires at least 1 gigahertz for processor (CPU), at least 16 gigabytes for hard disk space and at least 2 gigabytes of main memory (RAM) for managing with a lot of customer's session at peak hour.

2.5 Design and Implementation Constraints

List of project constraints:

- The application work on only android version not less than 2.3
- The application is requires the internet connection.
- The application provide to user who is already registering in the system.
- The application has to specify available date (ingredient season) of each ingredient that has been shared

3. System architecture

Figure 1 System architecture

       Lanna Cooking Recipe on Mobile is the system that help customer for making Lanna food, and finding the location that sell ingredients. The customer can accessed from own mobile phone. Lanna Recipe and customer information had record in database system such as when customer want to see recipe information, our application will request to connect with server for receive the data from database by JSON form and display on own mobile phone. In other site, the admin can add, edit, and delete the data in the database via own PC.

4. Detailed design

4.1 Class diagram

4.1.1 PHP class diagram
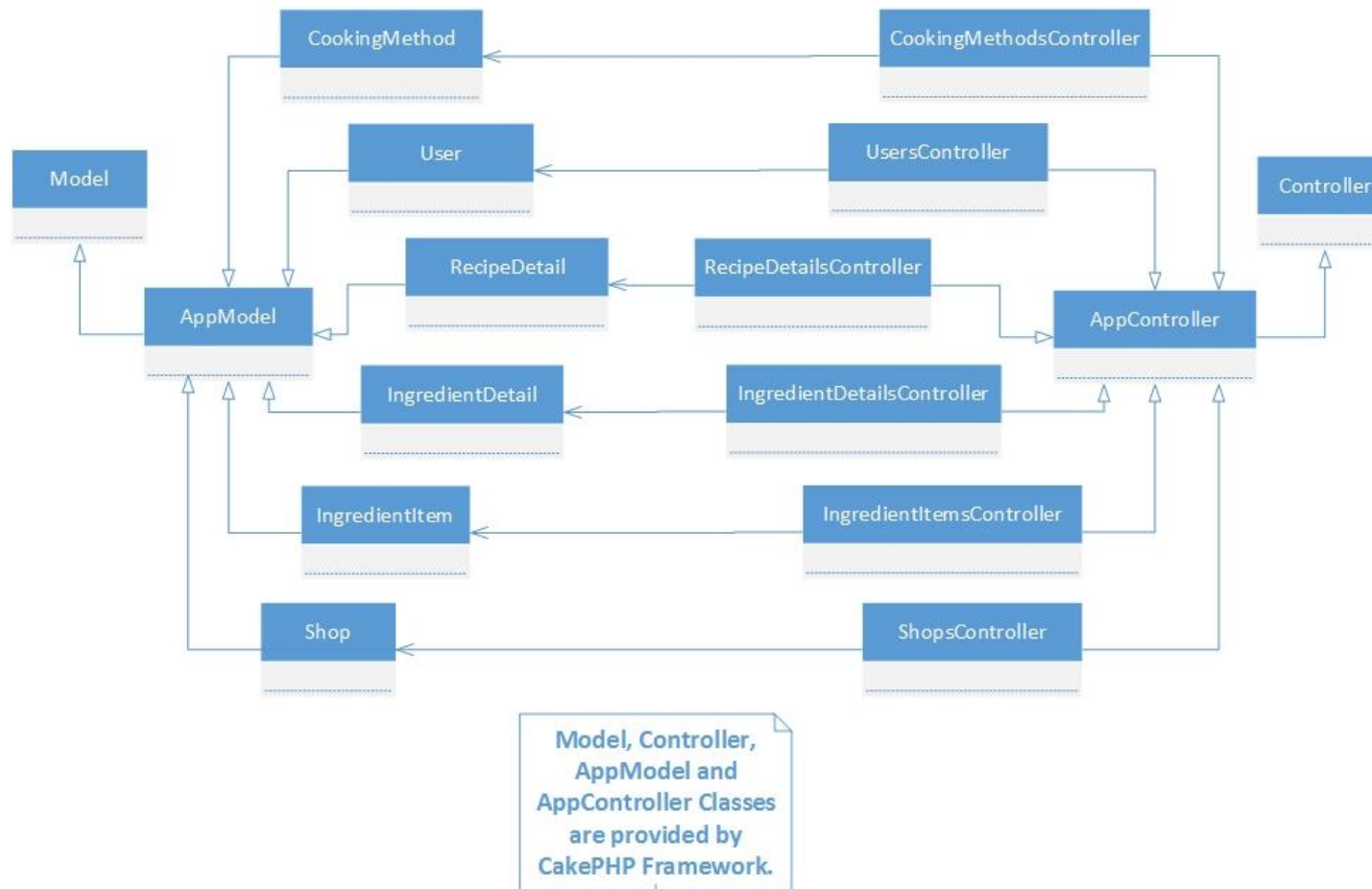
Figure 2 PHP class diagram

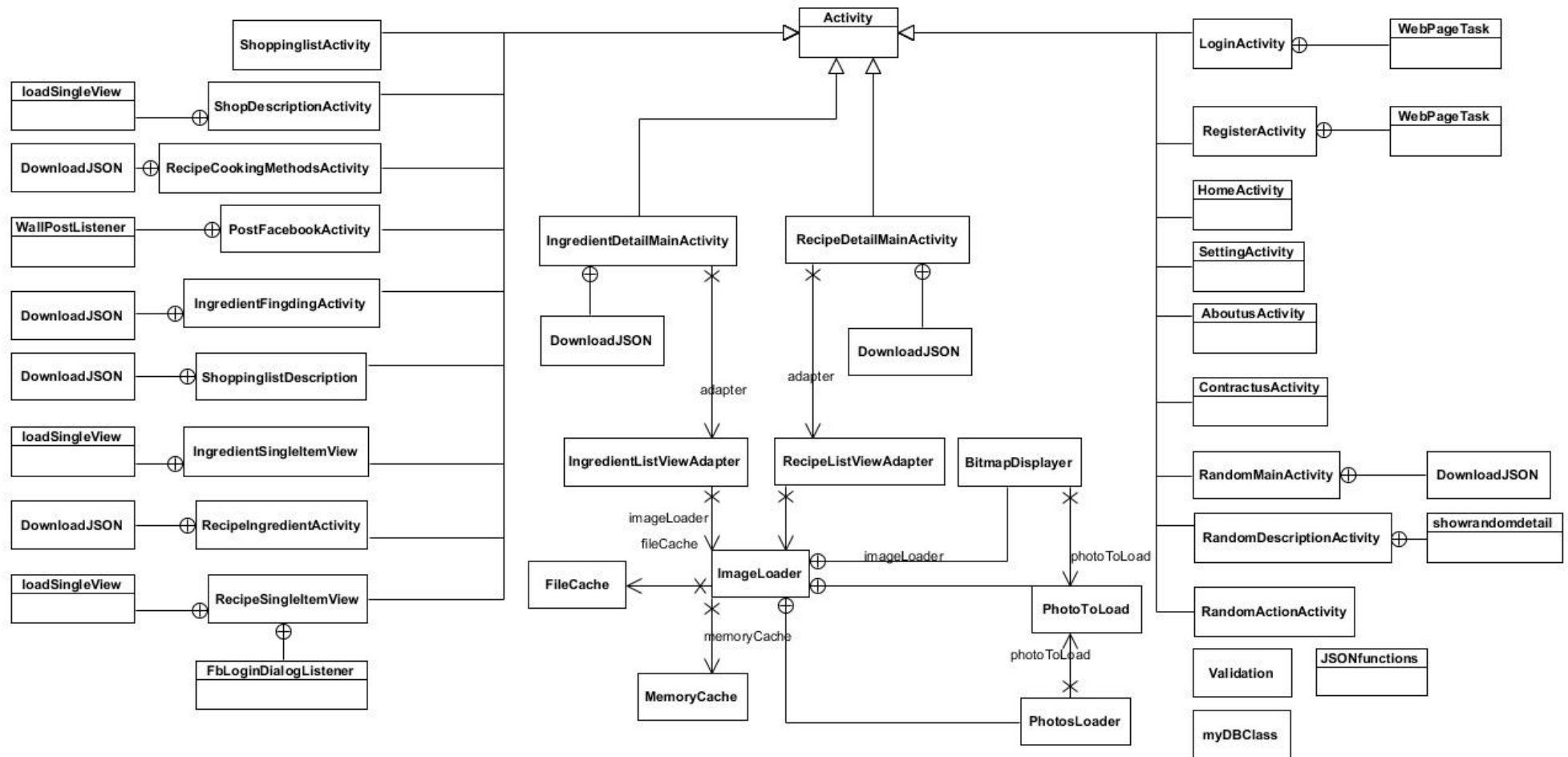| Document Name	LCR -SDD-V.3.0-docx	Owner	KS	Page	**11 / 117**
Document Type	Software Design Document	Release Date	3 October 2013	Print Date	3 October 2013

4.1.2 Android Application class diagram

Figure 3 Android Application class diagram

4.2 Class Diagram Description

4.2.1 PHP class diagram Description

Model

PHP Class Diagram ID: 1 Traceability: URS-19,20

Class Name: User
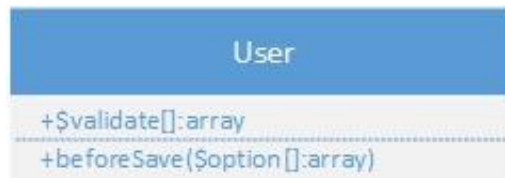
Property

ID	Name	Description	Remark
1	$validate[]	Declare variable for validate the form input data.	**Type:** array

Method

ID	Name	Description	Remark
1	beforeSave($option[]:array)	Method used to hashing the password before save to the database.	**Parameter:** $option[]

PHP Class Diagram ID: 2 Traceability: URS-21

Class Name: RecipeDetail

Property

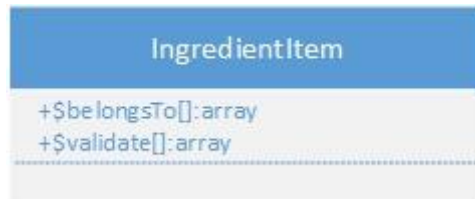| ID	Name	Description	Remark
1	$hasmany[]	Declare variable for identify the relationship between database.	**Type:** array
2	$validate[]	Declare variable for validate the form input data.	**Type:** array
3	$actsAs[]	Declare variable for validate the image before save to the database.	**Type:** array

Method

ID	Name	Description	Remark
-	-	-	-

PHP Class Diagram ID: 3 Traceability: URS-22

Class Name: IngredientDetail

Property

ID	Name	Description	Remark
1	$hasmany[]	Declare variable for identify the relationship between database.	**Type:** array
2	$validate[]	Declare variable for validate the form input data.	**Type:** array
3	$actsAs[]	Declare variable for validate the image before save to the database.	**Type:** array

Method

ID	Name	Description	Remark
-	-	-	-

PHP Class Diagram ID: 4 Traceability: URS-23

Class Name: IngredientItem

Property

ID	Name	Description	Remark
1	$belongsTo[]	Declare variable for identify the relationship between database.	**Type:** array
2	$validate[]	Declare variable for validate the form input data.	**Type:** array

Method

ID	Name	Description	Remark
-	-	-	-

PHP Class Diagram ID: 5 Traceability: URS-24

Class Name: CookingMethod

Property

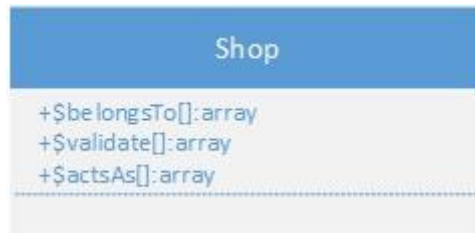| ID	Name	Description	Remark
1	$belongsTo[]	Declare variable for identify the relationship between database.	**Type:** array
2	$validate[]	Declare variable for validate the form input data.	**Type:** array
3	$actsAs[]	Declare variable for validate the image before save to the database.	**Type:** array

Method

ID	Name	Description	Remark
-	-	-	-

PHP Class Diagram ID: 6 Traceability: URS-26

Class Name: Shop

Property

ID	Name	Description	Remark
1	$belongsTo[]	Declare variable for identify the relationship between database.	**Type:** array
2	$validate[]	Declare variable for validate the form input data.	**Type:** array
3	$actsAs[]	Declare variable for validate the image of ingredient store locations.	**Type:** array

Method

ID	Name	Description	Remark
-	-	-	-

<u>**Controller**</u>

PHP Class Diagram ID: 7 Traceability: URS-19,20

Class Name: UsersController

Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	beforeFilter()	Method used to call authentication setting from AppController class.	-
2	login()	Method used to redirect to index page when admin login and check the username and password in database.	**-**
3	logout()	Method used to redirect to login page when admin clicks logout.	**-**
4	add()	Method used to redirect to add admin page for register new admin to database.	**-**
5	delete($id:int)	Method used to delete admin in database.	**Parameter:** $id

PHP Class Diagram ID: 8 Traceability: URS-21

Class Name: RecipeDetailsController

Property

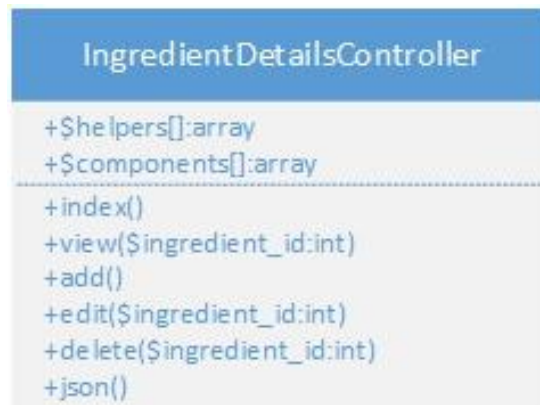| ID	Name	Description	Remark
1	$helpers[]	Declare variable for identify the HtmlHelper that used in this controller.	**Type:** array
2	$components[]	Declare variable for identify the components that used in this controller.	**Type:** array

Method

ID	Name	Description	Remark
1	index()	Method used to displays index page of recipe detail.	**-**
2	view($recipe_id:int)	Method used to redirects page to view page for shows the details of recipe detail from database.	**Parameter:** $recipe_id
3	add()	Method used to redirects page to add recipe detail page for adding recipe detail to the database.	**-**
4	edit($recipe_id:int)	Method used to redirects page to edit recipe detail page for editing recipe detail and save to database.	**Parameter:** $recipe_id
5	delete($recipe_id:int)	Method used to deletes recipe detail in database.	**Parameter:** $recipe_id
6	json()	Method used to converts JSON file from recipe detail data.	**-**

PHP Class Diagram ID: 9 Traceability: URS-22

Class Name: IngredientDetailsController

Property

ID	Name	Description	Remark
1	$helpers[]	Declare variable for identify the HtmlHelper that used in this controller.	**Type:** array
2	$components[]	Declare variable for identify the components that used in this controller.	**Type:** array

Method

ID	Name	Description	Remark
1	index()	Method used to displays index page of ingredient detail.	**-**
2	view($recipe_id:int)	Method used to redirects page to view page for shows the details of ingredient detail from database.	**Parameter:** $ingredient _id
3	add()	Method used to redirects page to add ingredient detail page for adding ingredient detail to the database.	**-**
4	edit($recipe_id:int)	Method used to redirects page to edit ingredient detail page for editing ingredient detail and save to database.	**Parameter:** $ingredient _id
5	delete($recipe_id:int)	Method used to deletes ingredient detail in database.	**Parameter:** $ingredient _id
6	json()	Method used to converts JSON file from ingredient detail data.	**-**

PHP Class Diagram ID: 10 Traceability: URS-23

Class Name: IngredientItemsController

Property

ID	Name	Description	Remark
1	$helpers[]	Declare variable for identify the HtmlHelper that used in this controller.	**Type:** array
2	$components[]	Declare variable for identify the components that used in this controller.	**Type:** array

Method

ID	Name	Description	Remark
1	index()	Method used to displays index page of ingredient item.	**-**
2	add()	Method used to redirects page to add ingredient item page for adding ingredient item to the database.	**-**
3	edit($id:int)	Method used to redirects page to edit ingredient item page for editing ingredient item and save to database.	**Parameter:** $id
4	delete($id:int)	Method used to deletes ingredient item in database.	**Parameter:** $id
5	json()	Method used to converts JSON file from ingredient item data.	**-**

PHP Class Diagram ID: 11 Traceability: URS-24

Class Name: CookingMethodsController

Property

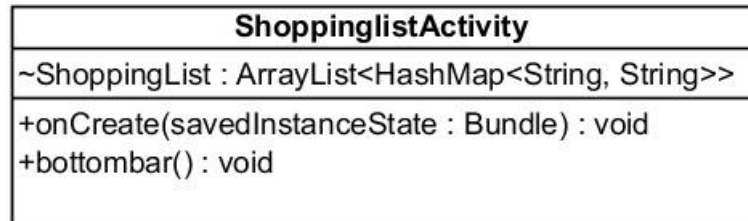| ID	Name	Description	Remark
1	$helpers[]	Declare variable for identify the HtmlHelper that used in this controller.	**Type:** array
2	$components[]	Declare variable for identify the components that used in this controller.	**Type:** array

Method

ID	Name	Description	Remark
1	index()	Method used to displays index page of cooking method.	**-**
2	view($id:int)	Method used to redirects page to view page for shows the details of cooking method from database.	**Parameter:** $id
3	add()	Method used to redirects page to add cooking method page for adding cooking method to the database.	**-**
4	edit($id:int)	Method used to redirects page to edit cooking method page for editing cooking method and save to database.	**Parameter:** $id
5	delete($id:int)	Method used to deletes cooking method in database.	**Parameter:** $id
6	json()	Method used to converts JSON file from cooking method data.	**-**

PHP Class Diagram ID: 12 Traceability: URS-26

Class Name: ShopsController

Property

ID	Name	Description	Remark
1	$helpers[]	Declare variable for identify the HtmlHelper that used in this controller.	**Type:** array
2	$components[]	Declare variable for identify the components that used in this controller.	**Type:** array

Method

ID	Name	Description	Remark
1	index()	Method used to displays index page of shop.	**-**
2	view($id:int)	Method used to redirects page to view page for shows the details of shop and show map from database.	**Parameter:** $id
3	add()	Method used to redirects page to add shop method page for adding ingredient store location to the database.	**-**
4	edit($id:int)	Method used to redirects page to edit shop page for editing ingredient store location and save to database.	**Parameter:** $id
5	delete($id:int)	Method used to deletes ingredient store location in database.	**Parameter:** $id
6	json()	Method used to converts JSON file from ingredient store location data.	**-**

4.2.2 Android Application class diagram Description

Application Class Diagram ID: 1 Traceability: URS-06

Class Name: ShoppinglistActivity

```
┌─────────────────────────────────────────────────────┐
│                 ShoppinglistActivity                 │
├─────────────────────────────────────────────────────┤
│ ~ShoppingList : ArrayList<HashMap<String, String>>   │
├─────────────────────────────────────────────────────┤
│ +onCreate(savedInstanceState : Bundle) : void        │
│ +bottombar() : void                                  │
└─────────────────────────────────────────────────────┘
```

Property

ID	Name	Description	Remark
1	ShoppingList	Object of ArrayList	**Type:** ArrayList

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 2 Traceability: URS-05

Class Name: RecipeCookingMethodsActivity

```
┌─────────────────────────────────────────────────┐
│           RecipeCookingMethodsActivity           │
├─────────────────────────────────────────────────┤
│ ~rid : String                                    │
│ ~jsonobject : JSONObject                         │
│ ~jsonarray : JSONArray                           │
│ ~arraylist : ArrayList<HashMap<String, String>>  │
│ ~mProgressDialog : ProgressDialog                │
├─────────────────────────────────────────────────┤
│ +onCreate(savedInstanceState : Bundle) : void    │
│                                                  │
└─────────────────────────────────────────────────┘
```

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	jsonobject	Object of JSONObject	**Type:** JSON object
3	jsonarray	Object of JSONArray	**Type:** JSON array
4	arraylist	Set Array List	**Type:** Array List
5	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState

Application Class Diagram ID: 3 Traceability: URS-05

Class Name: DownloadJSON

```
                    DownloadJSON
        ~recipeObject : JSONObject
        ~subobject : JSONObject
        ~array : JSONArray
        ~array2 : JSONArray
        ~resultob : JSONObject

        #onPreExecute() : void
        #doInBackground(params : Void ...) : Void
        #onPostExecute(args : Void) : void
```

Property

ID	Name	Description	Remark
1	recipeObject	Object of JSONObject	**Type:** JSON object
2	subobject	Object of JSONObject	**Type:** JSON object
3	array	Object of JSONArray	**Type:** JSON array
4	array2	Object of JSONArray	**Type:** JSON array
5	resultob	Object of JSONObject	**Type:** JSON object

Method

ID	Name	Description	Remark
1	onPreExecute() : void	invoked on the UI thread before the task is executed	**Return Type:** void
2	doInBackground (params : Void ...) : Void	invoked on the background thread immediately after **onPostExecute()** finishes executing	**Return Type:** void **Parameter:** params : Void …
3	onPostExecute(args : Void) : void	invoked on the UI thread after the background computation finishes	**Return Type:** void **Parameter:** args

Application Class Diagram ID: 4 Traceability: URS-11

Class Name: PostFacebookActivity

```
              PostFacebookActivity
-mFacebook : Facebook
-mFacebookCb : CheckBox
-mProgress : ProgressDialog
-mRunOnUi : Handler = new Handler()
-APP_ID : String
~rimg : String

#onCreate(savedInstanceState : Bundle) : void
-postToFacebook(review : String) : void
```
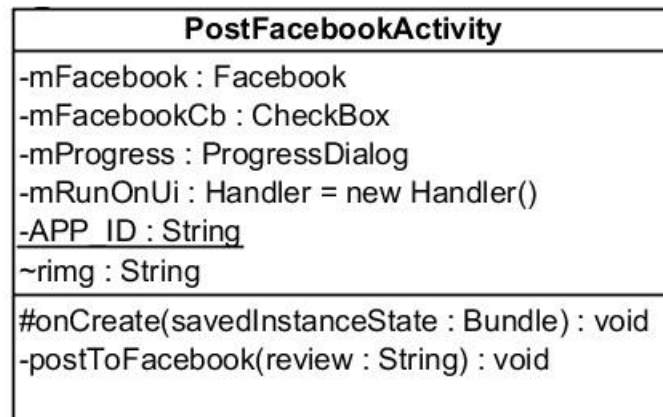
Property

ID	Name	Description	Remark
1	mFacebook	Object of Facebook	**Type:** Facebook object
2	mFacebookCb	Object of Checkbox	**Type:** Checkbox object
3	mProgress	Object of ProgressDialog	**Type:** ProgressDialog object
4	mRunOnUi	Object of Handler	**Type:** Handler object
5	APP_ID	String of Facebook API ID	**Type:** String
6	rimg	Set recipe image	**Type:** String

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	postToFaceBook(review : String): void	The system calls this method to post status on Facebook.	**Return Type:** void **Parameter:** review

Application Class Diagram ID: 5 Traceability: URS-12

Class Name: IngredientFingdingActivity

```
IngredientFingdingActivity
~rid : String
~jsonobject : JSONObject
~jsonarray : JSONArray
~mMap : GoogleMap
~arraylist : ArrayList<HashMap<String, String>>
~mMarker : Marker
~lm : LocationManager
~lat : double
~lng : double
~mProgressDialog : ProgressDialog
~listener : LocationListener = new LocationListener()

+onCreate(savedInstanceState : Bundle) : void
+onResume() : void
+onPause() : void
-addMarkersToMap() : void
+bottombar() : void
+getInfoContents(marker : Marker) : View
+getInfoWindow(arg0 : Marker) : View
```

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	jsonobject	Object of JSONObject	**Type:** JSON object
3	jsonarray	Object of JSONArray	**Type:** JSON array
4	mMap	Object of Google Map	**Type:** GoogleMap object
5	arraylist	Set Array List	**Type:** Array List
6	mMarker	Object of marker from google API	**Type:** Marker object
7	lm	Object of LocationManager	**Type:** LocationManager
8	lat	Set Latitude	**Type:** double
9	lng	Set Longitude	**Type:** double
10	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog
11	listener	Object of LocationListener	**Type:** LocationListener object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	onResume() : void	Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().	**Return Type:** void
3	onPause() : void	Called when the system is about to start resuming a previous activity.	**Return Type:** void
4	addMarkersToMap() : void	To add marker to Google map	**Return Type:** void
5	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
6	getInfoContents(marker : Marker) : View	Provides custom contents for the default info window frame of a marker	**Return Type:** View
7	getInfoWindow (arg0 : Marker) : View	Provides a custom info window for a marker.	**Return Type:** View

Application Class Diagram ID: 6 Traceability: URS-09

Class Name: ShoppinglistDescription

```
ShoppinglistDescription
~rid : String
~jsonobject : JSONObject
~jsonarray : JSONArray
~contenttxt : TextView
~arraylist : ArrayList<HashMap<String, String>>
~mProgressDialog : ProgressDialog

+onCreate(savedInstanceState : Bundle) : void
+bottombar() : void
```

Property

ID	Name	Description	Remark
1	rid	Set id recipe ID	**Type:** String
2	jsonobject	Object of JSONObject	**Type:** JSON object
3	jsonarray	Object of JSONArray	**Type:** JSON array
4	Contexttxt	Object of TextView	**Type:** TextView object
5	arraylist	Set Array List	**Type:** Array List
6	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 7 Traceability: URS-16

Class Name: IngredientSingleItemView

```
          IngredientSingleItemView
~rid : String
~rname : String
~rdes : String
~rimg : String
~rcategory : String
~mProgressDialog : ProgressDialog
~bmImg : Bitmap
────────────────────────────────────
+onCreate(savedInstanceState : Bundle) : void
+bottombar() : void
```

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	rname	Object of JSONObject	**Type:** JSON object
3	rdes	Set recipe description	**Type:** String
4	rimg	Set recipe image	**Type:** String
5	rcategory	Set category	**Type:** String
6	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog object
7	bmImg	Set bitmap from recipe image	**Type:** Bitmap object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 8 Traceability: URS-05

Class Name: RecipeIngredientActivity

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	jsonobject	Object of JSONObject	**Type:** JSON object
3	jsonarray	Object of JSONArray	**Type:** JSON array
4	arraylist	Set Array List	**Type:** Array List
5	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState

Application Class Diagram ID: 9 Traceability: URS-05, URS-11

Class Name: RecipeSingleItemView

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	rname	Object of JSONObject	**Type:** JSON object
3	rdes	Set recipe description	**Type:** String
4	rimg	Set recipe image	**Type:** String
5	rcategory	Set category	**Type:** String
6	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog object
7	Bmlmg	Set bitmap from recipe image	**Type:** Bitmap object
8	mFacebook	Object of Facebook	**Type:** Facebook object
9	mFacebookBtn	Object of Checkbox	**Type:** Checkbox object
10	PERMISSIONS	Set PERMISSIONS for connecting with Facebook	
11	APP_ID	Set Facebook API ID	**Type:** String
12	mFbHandler	Handler of successful message when connecting with Facebook	**Type:** Handler object
13	mHandler	Handler of unsuccessful message when connecting with Facebook	**Type:** Handler object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
3	facebook():void	The system calls this method to connect with Facebook account.	**Return Type:** void
4	onFacebookClick():void	The system calls this method to connect or leave from Facebook account.	**Return Type:** void
5	getFbName():void	The system calls this method to get Facebook account name.	**Return Type:** void
6	fbLogout():void	The system calls this method to logout from Facebook account.	**Return Type:** void

Application Class Diagram ID: 10 Traceability: URS-11

Class Name: FbLoginDialogListener


```
                FbLoginDialogListener
+onComplete(values : Bundle) : void
+onFacebookError(error : FacebookError) : void
+onError(error : DialogError) : void
+onCancel() : void
```
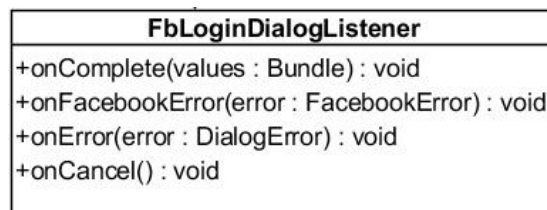
Property

ID	Name	Description	Remark
-	-	-	-

Method

ID	Name	Description	Remark
1	onComplete(values:Bundle):void	The system calls this method when successful to connecting with Facebook.	**Return Type:** void **Parameter:** values
2	onFacebookError(error: FacebookError):void	The system calls this method when unsuccessful to connecting with Facebook.	**Return Type:** void **Parameter:** error
3	onError(error:DialogError):void	The system calls this method to show error dialog when unsuccessful to connecting with Facebook.	**Return Type:** void **Parameter:** error
4	onCancel():void	The system calls this method to show error dialog when unsuccessful to cancel to connect with Facebook	**Return Type:** void

Application Class Diagram ID: 11 Traceability: URS-05

Class Name: loadSingleView

```
            loadSingleView
#onPreExecute() : void
#doInBackground(args : String ...) : String
#onPostExecute(args : String) : void
```

Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	onPreExecute() : void	invoked on the UI thread before the task is executed	**Return Type:** void
2	doInBackground (params : Void ...) : Void	invoked on the background thread immediately after **onPostExecute()** finishes executing	**Return Type:** void **Parameter:** params : Void …
3	onPostExecute(args : Void) : void	invoked on the UI thread after the background computation finishes	**Return Type:** void **Parameter:** args

Application Class Diagram ID: 12 Traceability: URS-15

Class Name: IngredientDetailMainActivity

```
                    IngredientDetailMainActivity
~jsonobject : JSONObject
~jsonarray : JSONArray
~listview : ListView
~mProgressDialog : ProgressDialog
~arraylist : ArrayList<HashMap<String, String>>
~IID : String
~INAME : String
~IDES : String
~IIMG : String
~ICATEGORY : String
~imgurl : String
~adapter : IngredientListViewAdapter
────────────────────────────────────────────────
+onCreate(savedInstanceState : Bundle) : void
+bottombar() : void
-isInternetConnected() : boolean
-showAlertNoNet() : void
```

Property

ID	Name	Description	Remark
1	jsonobject	Object of JSONObject	**Type:** JSON object
2	jsonarray	Object of JSONArray	**Type:** JSON array
3	listview	Object of ListView	**Tpye:** ListView object
4	arraylist	Set Array List	**Type:** Array List
5	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog
6	IID	Json node name for ingredient ID	**Type:** String
7	INAME	Json node name for ingredient name	**Type:** String
8	IDES	Json node name for ingredient description	**Type:** String
9	IIMG	Json node name for ingredient image	**Type:** String
10	ICATEGORY	Json node name for ingredient category	**Type:** String
11	imgurl	Set image Url	**Type:** String
12	adapter	Object of IngredientListViewAdapter	**Type:**RecipeListViewAdaper object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
3	isInternetConnected():boolean	The system calls this method for checking internet connection.	**Return Type:** boolean
4	showAlertNoNet(): void	The system calls this method to show alert message when no internet connection.	**Return Type:** void

Application Class Diagram ID: 13 Traceability: URS-15

Class Name: IngredientListViewAdapter

```
                    IngredientListViewAdapter
─────────────────────────────────────────────────────────────────
~context : Context
~inflater : LayoutInflater
~data : ArrayList<HashMap<String, String>>
~imageLoader : ImageLoader
─────────────────────────────────────────────────────────────────
+IngredientListViewAdapter(context : Context, arraylist : ArrayList<HashMap<String, String>>)
+getCount() : int
+getItem(position : int) : Object
+getItemId(position : int) : long
+getView(position : int, convertView : View, parent : ViewGroup) : View
```

Property

ID	Name	Description	Remark
1	Context	Object of Context	**Type:** Context object
2	Inflater	Object of LayoutInflater	**Type:** LayoutInflater object
3	Data	Arraylist of data	**Type:** Array
4	imageLoader	Object of ImageLoarder class	**Type:** ImageLoarder object

Method

ID	Name	Description	Remark
1	IngredientListViewAdapter(context:context, arraylist : ArrayList<HashMap<String,String>>)	Constructor of IngredientListViewAdapter	**Return Type:** - **Parameter:** Context,Arraylist
2	getCount():int	How many items are in the data set represented by this Adapter	**Return Type:** int **Parameter:**
3	getItem(position : int) : Object	Get position of on click item	**Return Type:** Object **Parameter:** int
4	getItemId(position : int) : long	Get position of on click item	**Return Type:** long **Parameter:** int
5	getView(position : int,convertView : View , parent: ViewGroup) : View	Get a View that displays the data at the specified position in the data set.	**Return Type:** View **Parameter:** int, View, ViewGroup

Application Class Diagram ID: 14 Traceability: URS-04

Class Name: RecipeDetailMainActivity

```
                    RecipeDetailMainActivity
~jsonobject : JSONObject
~jsonarray : JSONArray
~listview : ListView
~mProgressDialog : ProgressDialog
~arraylist : ArrayList<HashMap<String, String>>
~RID : String
~RNAME : String
~RDES : String
~RIMG : String
~RCATEGORY : String
~imgurl : String
~adapter : RecipeListViewAdapter
─────────────────────────────────────────────────────
+onCreate(savedInstanceState : Bundle) : void
+bottombar() : void
-isInternetConnected() : boolean
-showAlertNoNet() : void
```

Property

ID	Name	Description	Remark
1	jsonobject	Object of JSONObject	**Type:** JSON object
2	jsonarray	Object of JSONArray	**Type:** JSON array
3	Listview	Object of ListView	**Tpye:** ListView object
4	arraylist	Set Array List	**Type:** Array List
5	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog
6	RID	Json node name for recipe ID	**Type:** String
7	RNAME	Json node name for recipe name	**Type:** String
8	RDES	Json node name for recipe description	**Type:** String
9	RIMG	Json node name for recipe image	**Type:** String
10	RCATEGORY	Json node name for recipe category	**Type:** String
11	Imgurl	Set image Url	**Type:** String
12	adapter	Object of RecipeListViewAdapter	**Type:** RecipeListViewAdaper object

Method

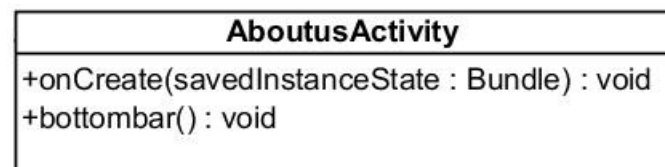| ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
3	isInternetConnected():boolean	The system calls this method for checking internet connection.	**Return Type:** boolean
4	showAlertNoNet(): void	The system calls this method to show alert message when no internet connection.	**Return Type:** void

Application Class Diagram ID: 15 Traceability: URS-04

Class Name: RecipeListViewAdapter

```
                        RecipeListViewAdapter
~context : Context
~inflater : LayoutInflater
~data : ArrayList<HashMap<String, String>>
~imageLoader : ImageLoader

+RecipeListViewAdapter(context : Context, arraylist : ArrayList<HashMap<String, String>>)
+getCount() : int
+getItem(position : int) : Object
+getItemId(position : int) : long
+getView(position : int, convertView : View, parent : ViewGroup) : View
```

Property

ID	Name	Description	Remark
1	Context	Object of Context	**Type:** Context object
2	Inflater	Object of LayoutInflater	**Type:** LayoutInflater object
3	Data	Arraylist of data	**Type:** Array
4	imageLoader	Object of ImageLoarder class	**Type:** ImageLoarder object

Method

ID	Name	Description	Remark
1	RecipeListViewAdapter(context:context, arraylist : ArrayList<HashMap<String,String>>)	Constructor of RecipeListViewAdaptor	**Return Type:** - **Parameter:** Context,Arraylist
2	getCount():int	How many items are in the data set represented by this Adapter	**Return Type:** int **Parameter:**
3	getItem(position : int) : Object	Get position of on click item	**Return Type:** Object **Parameter:** int
4	getItemId(position : int) : long	Get position of on click item	**Return Type:** long **Parameter:** int
5	getView(position : int,convertView : View , parent: ViewGroup) : View	Get a View that displays the data at the specified position in the data set.	**Return Type:** View **Parameter:** int, View, ViewGroup

Application Class Diagram ID: 16 Traceability: URS-05

Class Name: FileCache

```
              FileCache
-cacheDir : File
+FileCache(context : Context)
+getFile(url : String) : File
+clear() : void
```
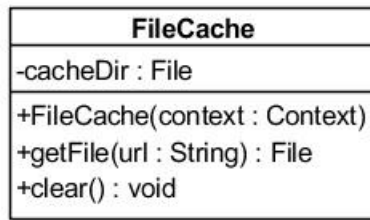
Property

ID	Name	Description	Remark
1	cacheDir	Set cache directory	**Type:** File

Method

ID	Name	Description	Remark
1	FileCache(context : Context)	Construct of FileCache class	**Return Type:** - **Parameter:** Context
2	getFile(url : String) : File	Get file by url	**Return Type:** File **Parameter:** String
3	clear() : void	Clear file	**Return Type:** void

Application Class Diagram ID: 17 Traceability: URS-04,05

Class Name: ImageLoader

```
                        ImageLoader
-imageViews : Map<ImageView, String> = Collections
        .synchronizedMap(new WeakHashMap<ImageView, String>())
~executorService : ExecutorService
~handler : Handler = new Handler()
~stub_id : int = R.drawable.ic_launcher
~memoryCache : MemoryCache = new MemoryCache()
~fileCache : FileCache
─────────────────────────────────────────────────────────
+ImageLoader(context : Context)
+DisplayImage(url : String, imageView : ImageView) : void
-queuePhoto(url : String, imageView : ImageView) : void
-getBitmap(url : String) : Bitmap
-decodeFile(f : File) : Bitmap
~imageViewReused(photoToLoad : PhotoToLoad) : boolean
+clearCache() : void
```

Property

ID	Name	Description	Remark
1	imageViews	Set cache directory	**Type:** Map
2	ececutorService	Object of ExecutorService	**Type:** ExecutorService object
3	handler	Object of handler	**Type:** handler object
4	stub_id	Set ID of default image	**Type:** int
5	MemoryCache	Object of MemoryCache	**Type:** MemoryCache object
6	fileCache	Object of FileCache	**Type:** FileCache object

Method

ID	Name	Description	Remark
1	imageLoader(context : android.content.Context)	Constructor of imageLoader class	**Return Type:** - **Parameter:** Context
2	DisplayImage(url : String, imageView : ImageView) : void	Display image	**Return Type:** void **Parameter:** String,ImageView
3	queuePhoto(url : String, imageView : ImageView) : void	Queue image	**Return Type:** void **Parameter:** String,ImageView
4	getBitmap(url : String) : Bitmap	Get Bitmap image from url	**Return Type:** Bitmap **Parameter:** String
5	decodeFile(f : File) :Bitmap	Decode image into Bitmap	**Return Type:** Bitmap **Parameter:** File
6	imageViewReused(photoToLoad : PhotoToLoad) : boolean	Reused image that already load	**Return Type:** boolean **Parameter:** PhotoToLoad
7	clearCache() : void	Clear cache file	**Return Type:** void

Application Class Diagram ID: 18 Traceability: URS-04,05

Class Name: MemoryCache

```
                        MemoryCache
-TAG : String
-cache : Map<String, Bitmap> = Collections
         .synchronizedMap(new LinkedHashMap<String, Bitmap>)
-size : long
<<Property>> -limit : long
────────────────────────────────────────────────────────
+MemoryCache()
+get(id : String) : Bitmap
+put(id : String, bitmap : Bitmap) : void
-checkSize() : void
+clear() : void
~getSizeInBytes(bitmap : Bitmap) : long
```

Property

ID	Name	Description	Remark
1	TAG	Set TAG as String for debugging	**Type:** String
2	cache	Collection of cache file	**Type:** Map
3	Size	Size of cache file	**Type:** long
4	Limit	Limit of cache file	**Type:** long

Method

ID	Name	Description	Remark
1	MemoryCache()	Constructor of MemoryCache class	**Return Type:** -
2	get(id : String): Bitmap	Get cache file from string	**Return Type:** Bitmap **Parameter:** String
3	Put(id : String, bitmap : Bitmap) : void	Put cache file by Bitmap	**Return Type:** void **Parameter:** Bitmap
4	checkSize() : void	Check size of cache file	**Return Type:** void
5	clear() : void	Clear cache file	**Return Type:** void
6	getSizeInBytes(bitmap : Bitmap): long	Get size of cache file (in bytes) by Bitmap	**Return Type:** long **Parameter:** Bitmap

Application Class Diagram ID: 19 Traceability: URS-02

Class Name: LoginActivity

```
                    LoginActivity
+url : String
+result : String
──────────────────────────────────────────────
#onCreate(savedInstanceState : Bundle) : void
+login() : String
+postData(url : String, nameValuePairs : List<NameValuePair>) : String
-isInternetConnected() : boolean
-showAlertNoNet() : void
```

Property

ID	Name	Description	Remark
1	url	Set URL Link for access to web server	**Type:** String
2	result	Using in WebPageTask() class	**Type:** String

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Login()	Get username and password into ArrayList and sent to server by postData() method	**Return Type:** String **Parameter:** -
3	postData(url:String , nameValuePairs: List<NameValuePair>):String	Sent data to server following URL	**Return Type:** String **Parameter:** -
4	isInternetConnected():boolean	The system calls this method for checking internet connection.	**Return Type:** boolean
5	showAlertNoNet(): void	The system calls this method to show alert message when no internet connection.	**Return Type:** void

Application Class Diagram ID: 20 Traceability: URS-01

Class Name: RegisterActivity

```
                    RegisterActivity
-editext_username : EditText
-editext_password : EditText
-editext_display : EditText
+urlHost : String
+result : String
#onCreate(savedInstanceState : Bundle) : void
-registerViews() : void
+signUp() : String
+postData(url : String, nameValuePairs : List<NameValuePair>) : String
-checkValidation() : boolean
```

Property

ID	Name	Description	Remark
1	edittext_username	Input text for username	**Type:** EditText
2	edittext_password	Input text for password	**Type:** EditText
3	edittext_display	Input text for display name	**Type:** EditText
4	urlHost	Set URL Link for access to web server	**Type:** String
5	result	Using in WebPageTask() class	**Type:** String

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	registerViews()	Get data from EditText to checkValidation	**Return Type:** void **Parameter:** -
3	signup():String	Get username , password and display name into ArrayList and sent to server by postData() method	**Return Type:** String **Parameter:** -
4	postData(url:String , nameValuePairs: List<NameValuePair>):String	Sent data to server following URL	**Return Type:** String **Parameter:** -
5	checkValidation():boolean	Method for check validation of input information which are username ,password ,display name	**Return Type:** boolean **Parameter:** -

Application Class Diagram ID: 21 Traceability: -

Class Name: HomeActivity

HomeActivity
+onCreate(savedInstanceState : Bundle) : void

Property

ID	Name	Description	Remark
-	-	-	-

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState

Application Class Diagram ID: 22 Traceability: -

Class Name: SettingActivity

SettingActivity
+onCreate(savedInstanceState : Bundle) : void

Property

ID	Name	Description	Remark
-	-	-	-

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState

Application Class Diagram ID: 23 Traceability: -

Class Name: AboutusActivity

Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 24 Traceability: -

Class Name: ContractusActivity

Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 25 Traceability: URS-10

Class Name: RandomActivity

```
                    RandomMainActivity
    ~jsonobject : JSONObject
    ~jsonarray : JSONArray
    ~listview : ListView
    ~mProgressDialog : ProgressDialog
    ~arraylist : ArrayList<HashMap<String, String>>
    ~RID : String
    ~RNAME : String
    ~RDES : String
    ~RIMG : String
    ~RCATEGORY : String
    ~imgurl : String

    +onCreate(savedInstanceState : Bundle) : void
    +bottombar() : void
    -isInternetConnected() : boolean
    -showAlertNoNet() : void
```

Property

ID	Name	Description	Remark
1	jsonobject	Object of JSONObject	**Type:** JSON object
2	jsonarray	Object of JSONArray	**Type:** JSON array
3	listview	Object of ListView	**Tpye:** ListView object
4	arraylist	Set Array List	**Type:** Array List
5	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog
6	RID	Json node name for recipe ID	**Type:** String
7	RNAME	Json node name for recipe name	**Type:** String
8	RDES	Json node name for recipe description	**Type:** String
9	RIMG	Json node name for recipe image	**Type:** String
10	RCATEGORY	Json node name for recipe category	**Type:** String
11	Imgurl	Set image Url	**Type:** String

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
3	isInternetConnected():boolean	The system calls this method for checking internet connection.	**Return Type:** boolean
4	showAlertNoNet(): void	The system calls this method to show alert message when no internet connection.	**Return Type:** void

Application Class Diagram ID: 26 Traceability: URS-10

Class Name: RandomDescriptionActivity

```
┌─────────────────────────────────────────────┐
│           RandomDescriptionActivity           │
├─────────────────────────────────────────────┤
│ ~rid : String                                 │
│ ~rname : String                               │
│ ~rdes : String                                │
│ ~rimg : String                                │
│ ~rcategory : String                           │
│ ~mProgressDialog : ProgressDialog             │
│ ~bmImg : Bitmap = null                        │
├─────────────────────────────────────────────┤
│ +onCreate(savedInstanceState : Bundle) : void │
│ +bottombar() : void                           │
│                                               │
└─────────────────────────────────────────────┘
```

Property

ID	Name	Description	Remark
1	rid	Set id recipe	**Type:** String
2	rname	Object of JSONObject	**Type:** JSON object
3	rdes	Set recipe description	**Type:** String
4	rimg	Set recipe image	**Type:** String
5	rcategory	Set category	**Type:** String
6	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog object
7	bmImg	Set bitmap from recipe image	**Type:** Bitmap object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void

Application Class Diagram ID: 27 Traceability: URS-10

Class Name: RandomActionActivity

```
                     RandomActionActivity
  ~sensorManager : SensorManager
  ~sensor : Sensor
  ~mProgressDialog : ProgressDialog
  ~rid : String
  ~rname : String
  ~rdes : String
  ~rimg : String
  ~rcate : String
  ~accelListener : SensorEventListener = new SensorEventListener()
  ─────────────────────────────────────────────────────────────
  +onCreate(savedInstanceState : Bundle) : void
  +onResume() : void
  +onStop() : void
  +bottombar() : void
```

Property

ID	Name	Description	Remark
1	sensorManager	Object of sensor manager for using sensor service	**Type:** SensorManager Object
2	sensor	Object of sensor	**Type:** Sensor Object
3	rid	Set id recipe	**Type:** String
4	rname	Object of JSONObject	**Type:** JSON object
5	rdes	Set recipe description	**Type:** String
6	rimg	Set recipe image	**Type:** String
7	rcate	Set category	**Type:** String
8	mProgressDialog	Object of ProgressDialog	**Type:** ProgressDialog object
9	accelListener	Set listener for accelerometer event	**Type:** SensorEventListener Object

Method

ID	Name	Description	Remark
1	onCreate(savedInstanceState : Bundle) : void	The system calls this when creating the activity	**Return Type:** void **Parameter:** saveInstanceState
2	Bottombar():void	The system calls this method for setting command botton on screen bottom.	**Return Type:** void
3	onResume() : void	Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().	**Return Type:** void
4	onPause() : void	Called when the system is about to start resuming a previous activity.	**Return Type:** void

Application Class Diagram ID: 28 Traceability: URS-01,02

Class Name: WebPageTask

```
                    WebPageTask
#doInBackground(param : Void ...) : String
#onPostExecute(result : String) : void
```
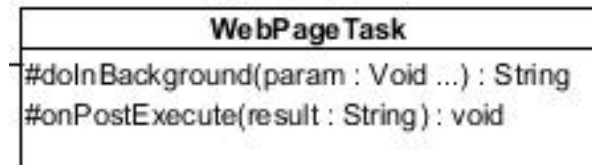
Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	doInBackground(param : Void …) : String	Override this method to perform a computation on a background thread	**Return Type:** String **Parameter:** param
2	onPostExecute(result:String): void	Runs on the UI thread after doInBackground(param : Void …)	**Return Type:** void **Parameter:** result

Application Class Diagram ID: 29 Traceability: URS-04,05,06,07,08,09,10,11

Class Name: JSONfunctions

```
                           JSONfunctions
+getJSONfromURL(url : String, params : List<NameValuePair>) : JSONObject
+getJSONfromURL(url : String) : JSONObject
```
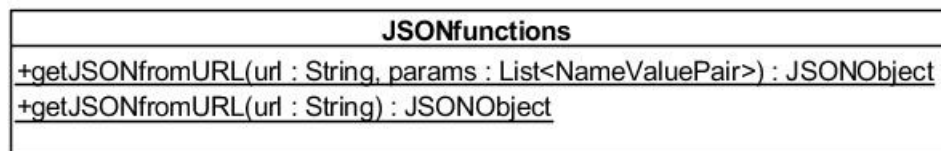
Property

ID	Name	Description	Remark
-	-	-	**-**

Method

ID	Name	Description	Remark
1	getJSONfromURL(url : String, params : List<NameValuePair>)	Get JSON form URL by parameters (url,params)	**Return Type:** JSONObject **Parameter:** url,params
2	getJSONfromURL(url : String)	Get JSON form URL by parameters (url)	**Return Type:** void **Parameter:**

Application Class Diagram ID: 30 Traceability: URS-01,02

Class Name: Validation

```
                            Validation
-EMAIL_REGEX : String
-REQUIRED_MSG : String
-EMAIL_MSG : String
+isEmailAddress(editText : EditText, required : boolean) : boolean
+isValid(editText : EditText, regex : String, errMsg : String, required : boolean) : boolean
+hasText(editText : EditText) : boolean
```

Property

ID	Name	Description	Remark
1	EMAIL_REGEX	String for checking valid input email data	**Type:** String
2	REQUIRED_MSG	Required text that show when input invalid data	**Type:** String
3	EMAIL_MSG	Required text that show when input invalid email	**Type:** String

Method

ID	Name	Description	Remark
1	isEmailAddress(editText:EditText,required:boolean): boolean	Check email validation	**Return Type:** boolean **Parameter:** editText, required
2	isValid(editText:EditText, regex:String ,errMsg : String , required :boolean):boolean	Check input field is valid	**Return Type:** boolean **Parameter:** editText,regex,errMsg, required
3	hasText(editText:EditText): boolean	Check input field has any text of not	**Return Type:** boolean **Parameter:** required

Application Class Diagram ID: 31 Traceability: URS-06,07,08,09

Class Name: myDBClass

```
                        myDBClass
-DATABASE_VERSION : int
-DATABASE_NAME : String
-TABLE_NAME : String
+myDBClass(context : Context)
+onCreate(db : SQLiteDatabase) : void
+InsertData(Username : String, RecipeID : String) : long
+SelectData(Username : String) : String []
+SelectAllData(Username : String) : ArrayList<HashMap<String, String>>
+DeleteData(RecipeID : String, Username : String) : void
```

Property

ID	Name	Description	Remark
1	DATABASE_VERSION	Set database version	**Type:** : int
2	DATABASE_NAME	Set database name	**Type:** String
3	TABLE_NAME	Set table name	**Type:** String

Method

ID	Name	Description	Remark
1	myDBClass(context : android.content.Context)	Constructor of myDBClass	**Return Type:** - **Parameter:** Context
2	onCreate(db : SQLiteDatabase) : void	Create SQLite Database	**Return Type:** void **Parameter:** db
3	InsertData(Username : String, RecipeID : String) : long	InsertData by using Username and RecipeID	**Return Type:** long **Parameter:** Username, RecipeID
4	SelectData(Username : String) : String []	Select data from username	**Return Type:** String [] **Parameter:** Username
5	SelectAllData(Username : String) :ArrayList<java.util.HashMap <String, String>>	Select all data from username	**Return Type:** String [] **Parameter:** ArrayList
6	DeleteData(RecipeID : String, Username : String) : void	Delete data from username and recipe id	**Return Type:** void **Parameter:** Username , RecipeID

3.1 Sequence diagram

SQ01: User Register
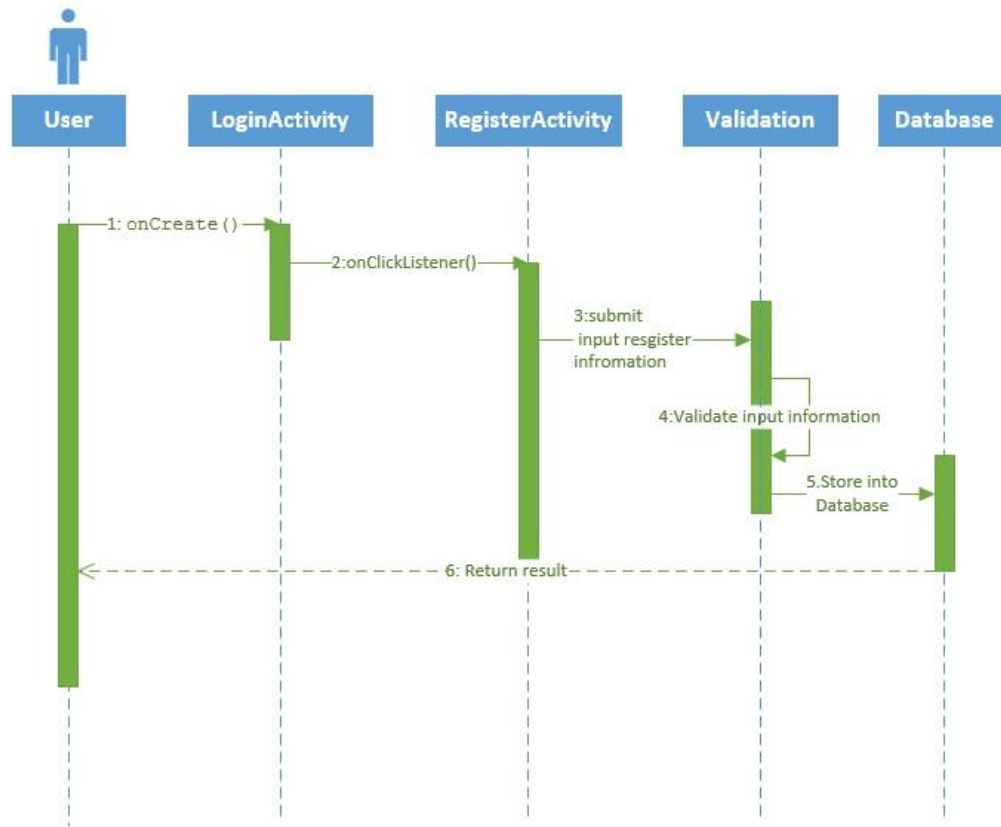
Figure 4 SQ01: User Register
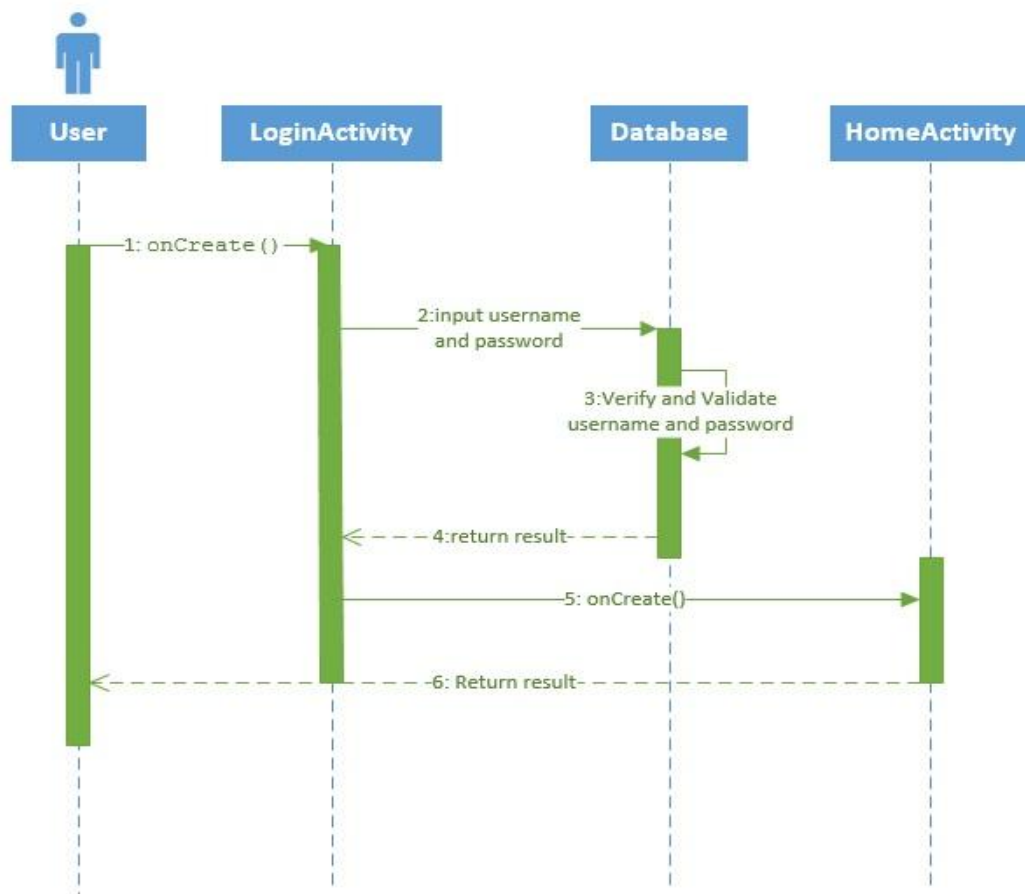
SQ02: User Login

Figure 5 SQ02: User Login

SQ03: User Logout

SQ03:User Logout
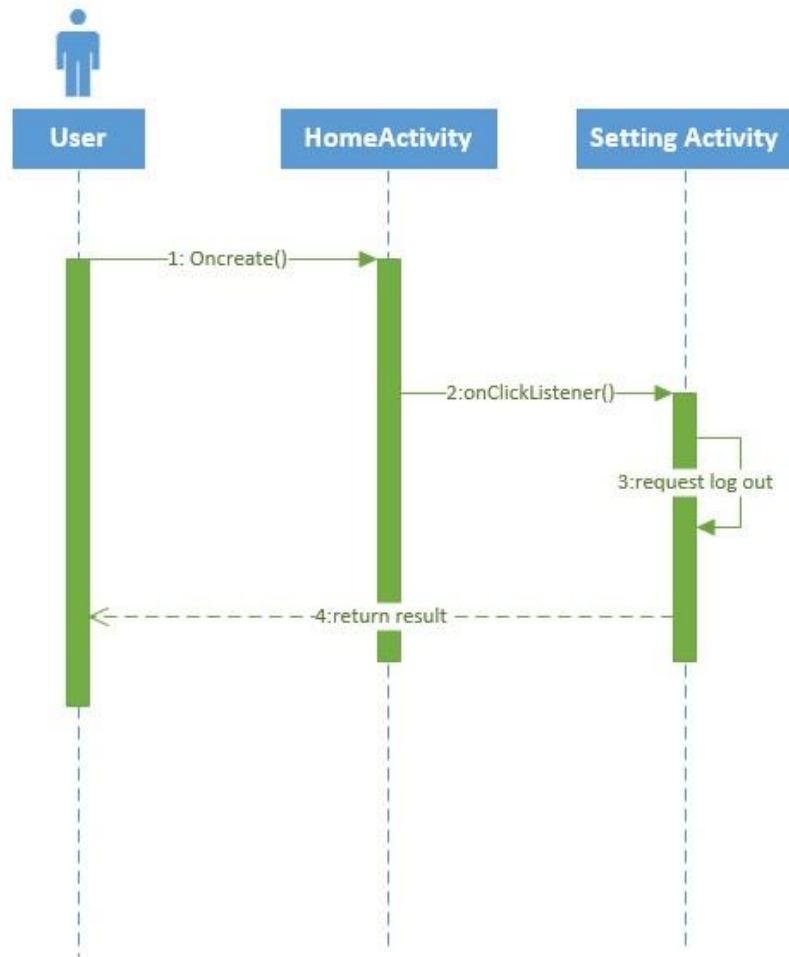
Figure 6 SQ03: User Logout

SQ04: View list of recipe

SQ04:View list of recipe
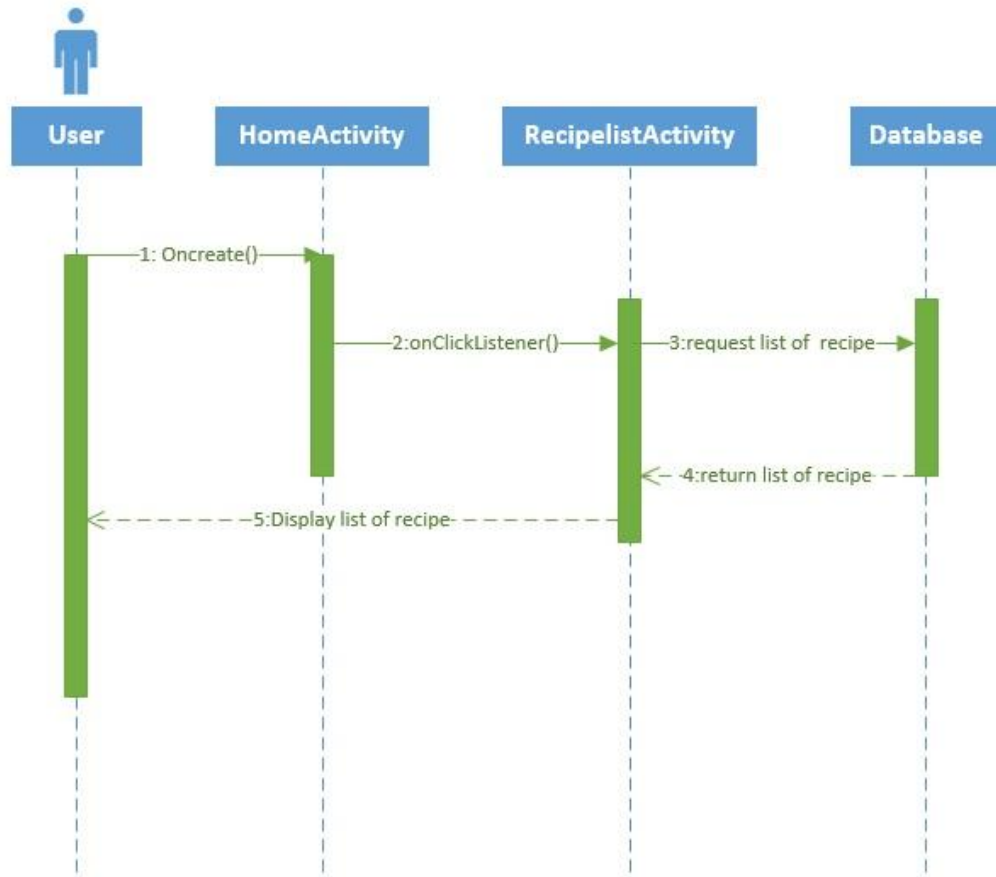
Figure 7 SQ04: View list of recipe

SQ05: View recipe information
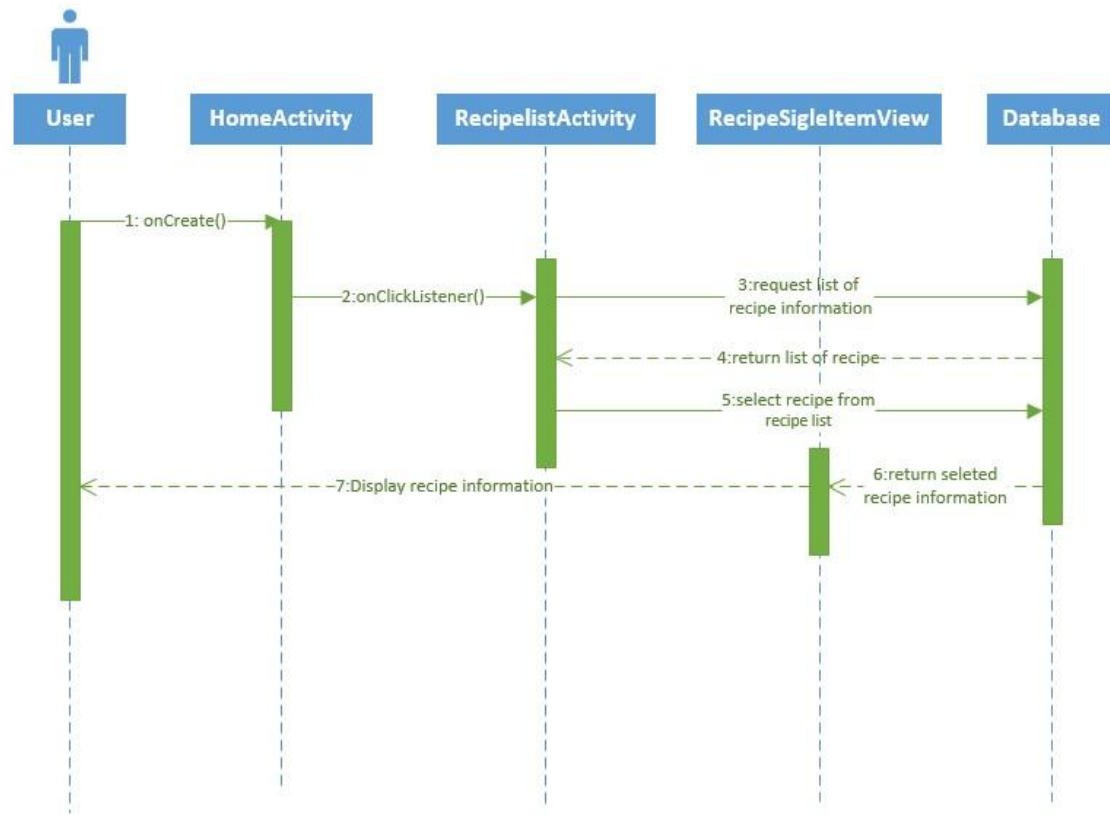
Figure 8 SQ05: View recipe information

SQ06: Add menu to shopping list
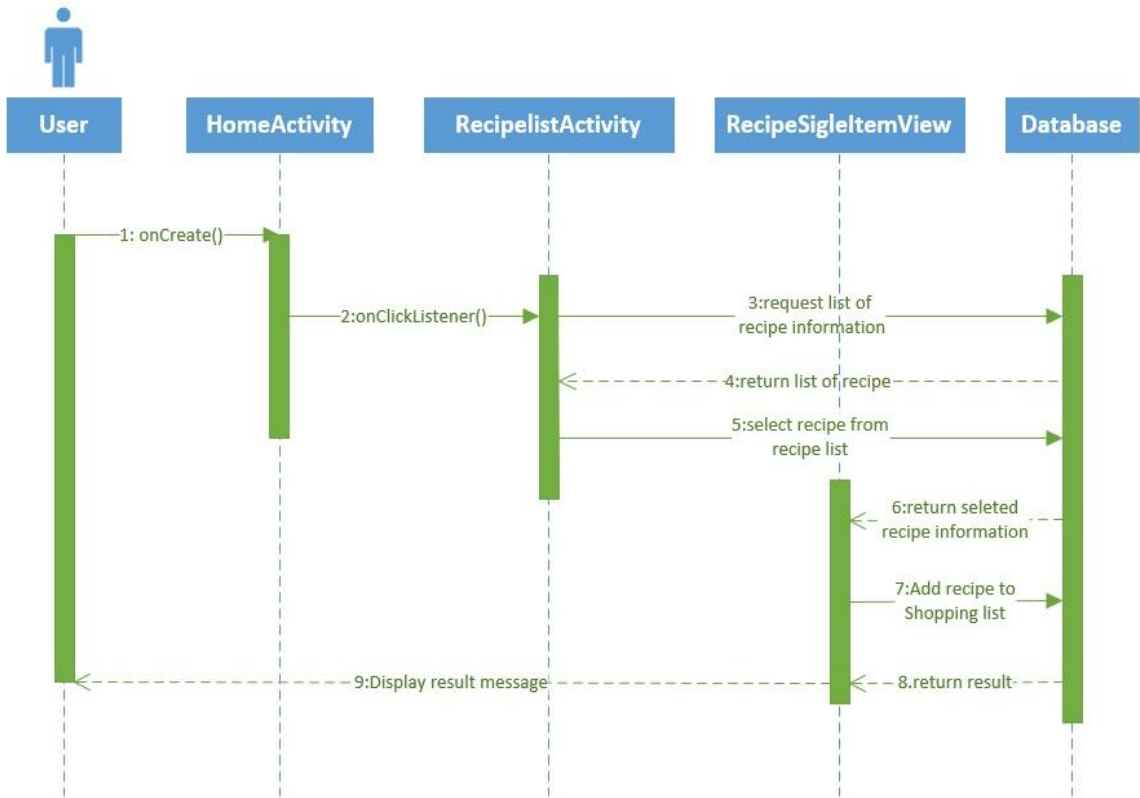
Figure 9 SQ06: Add menu to shopping list

SQ07: Delete menu from shopping list

SQ07:Delete menu from shopping list
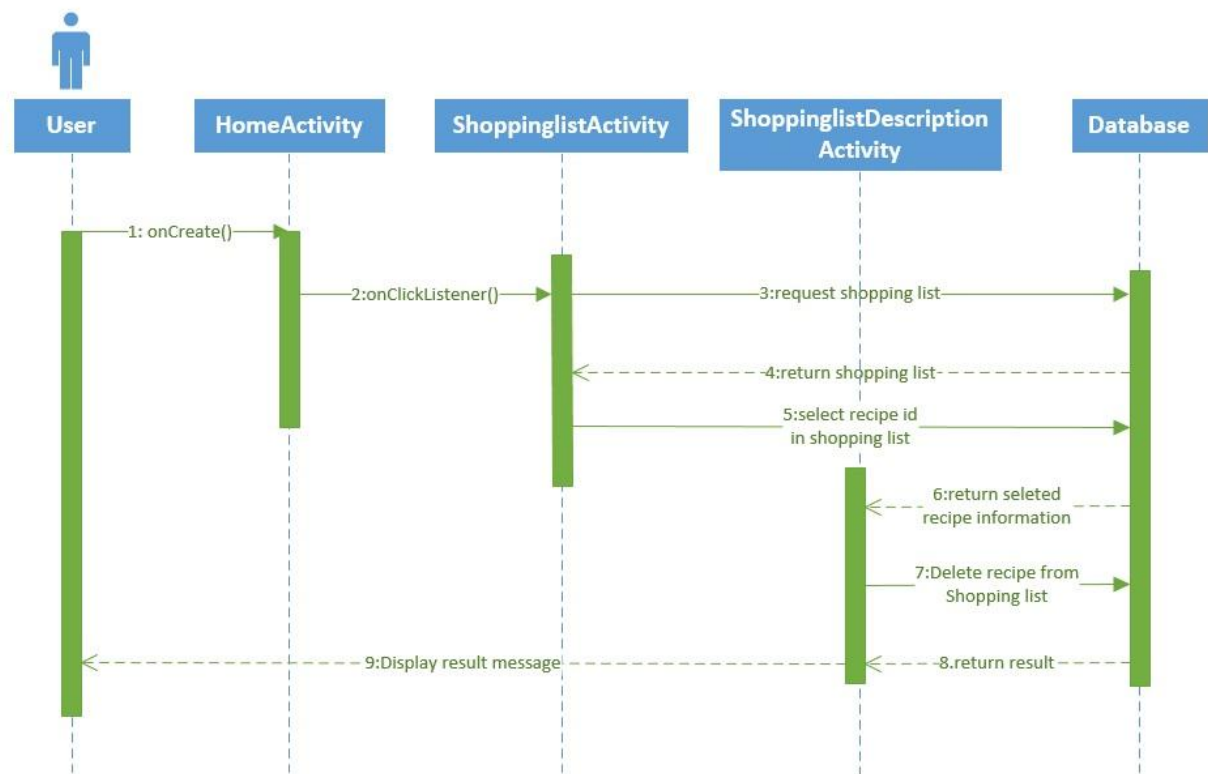
Figure 10 SQ07: Delete menu from shopping list

SQ08: View list of menu in shopping list
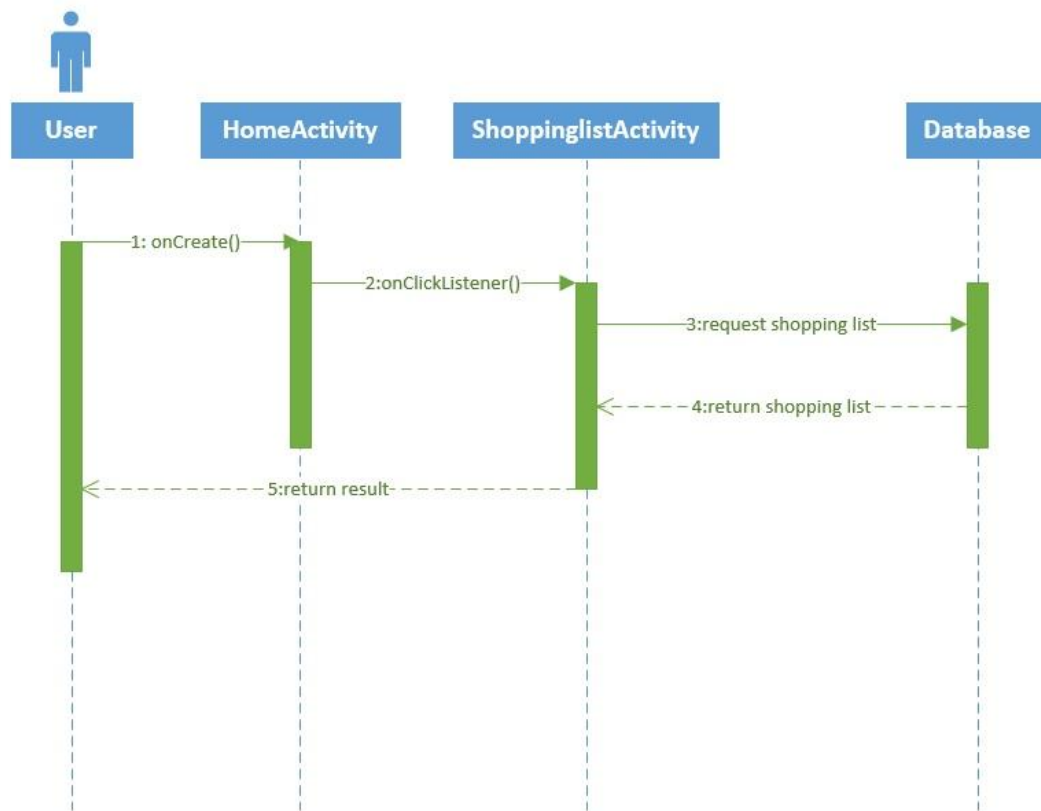
Figure 11 SQ08: View list of menu in shopping list

SQ09: View information of recipe in shopping list
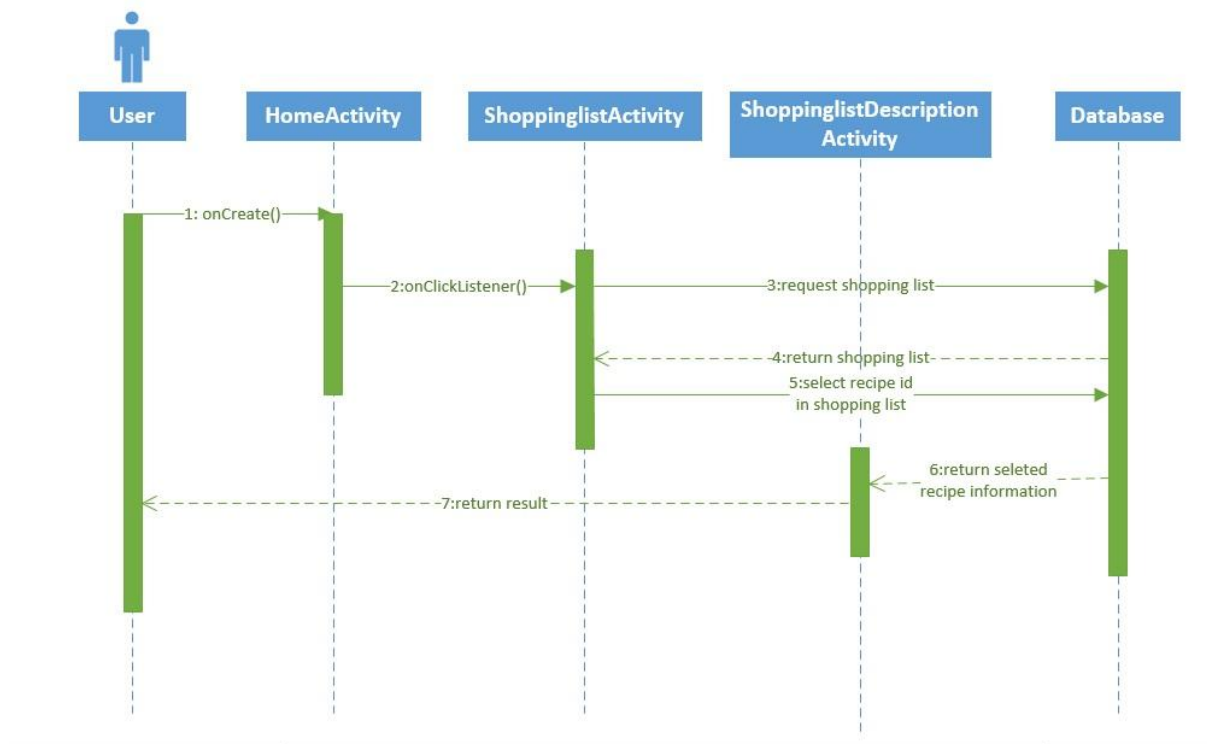
Figure 12 SQ09: View information of recipe in shopping list

SQ10: Random food menu

SQ10:Random food menu
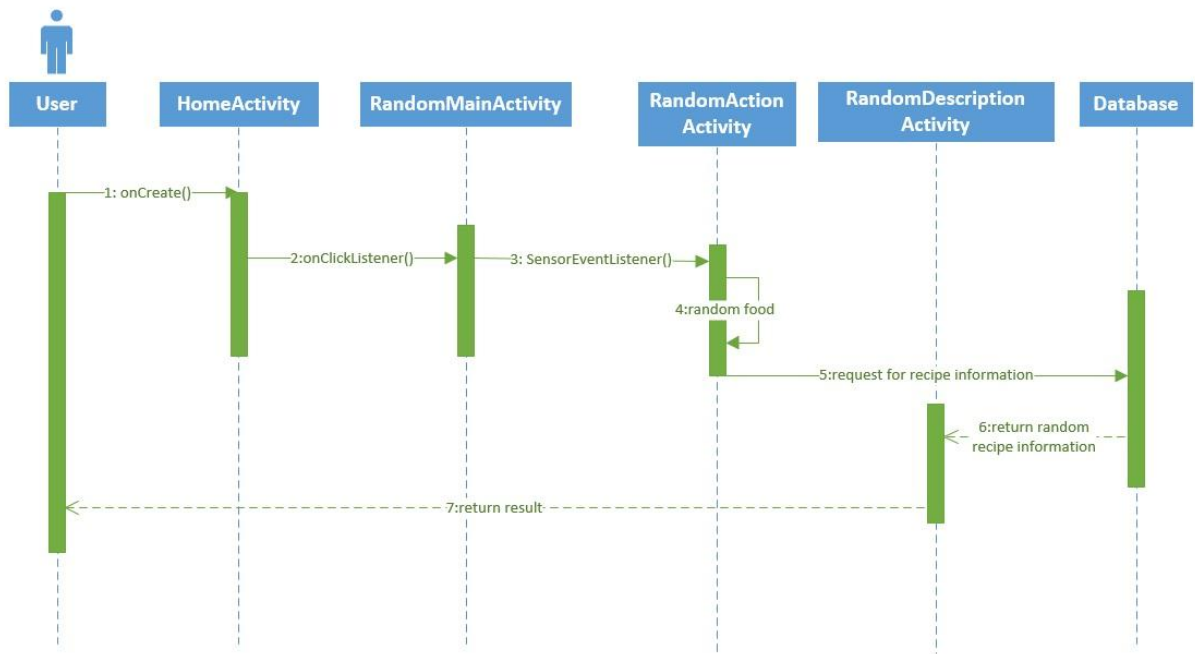
Figure 13 SQ10: Random food menu

SQ11: Sharing menu on Facebook
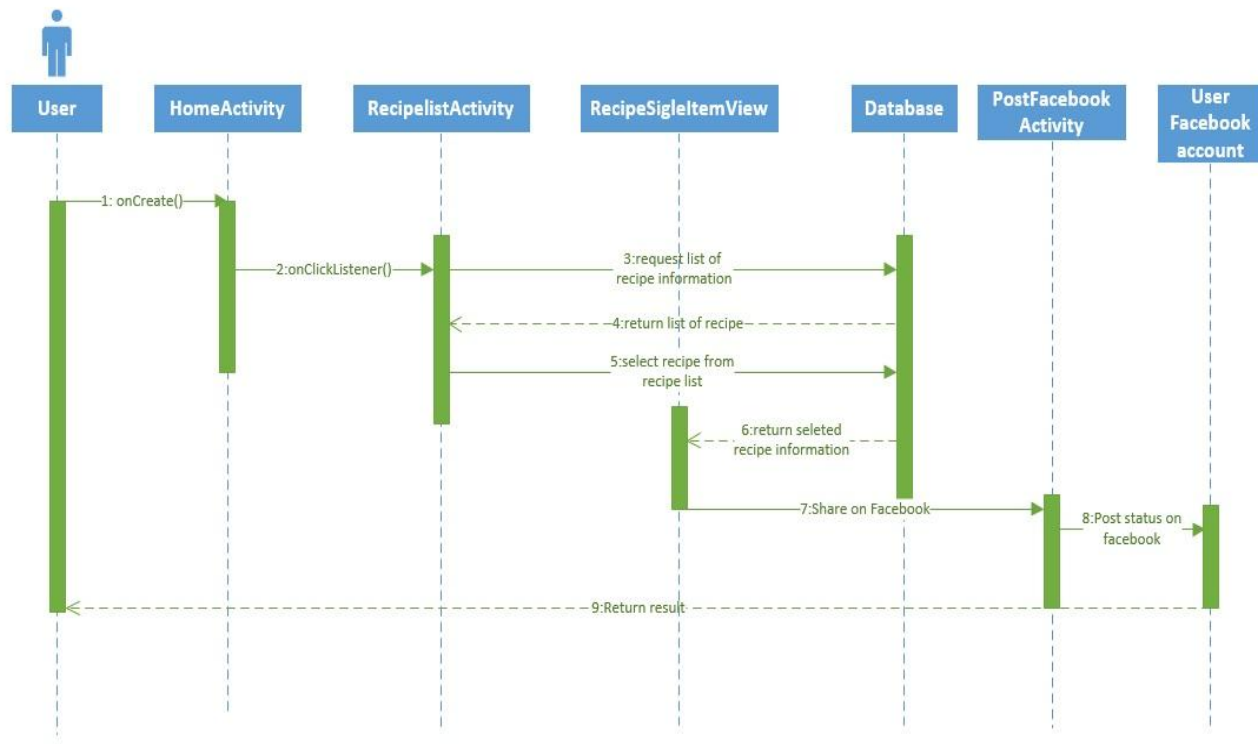
Figure 14 SQ11: Sharing menu on Facebook

SQ12: View ingredient store location around user location on Google map
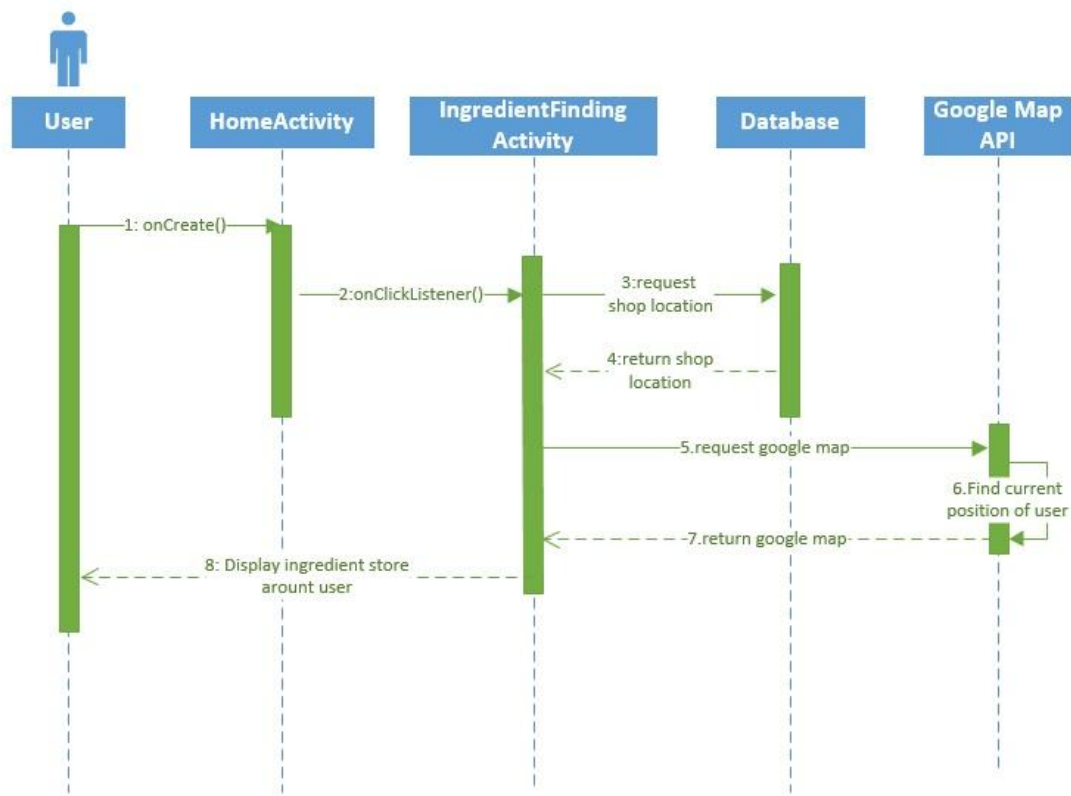
Figure 15 SQ12: View ingredient store location

around user location on Google map

SQ13: View ingredient store location information

SQ13:View ingredient store location information
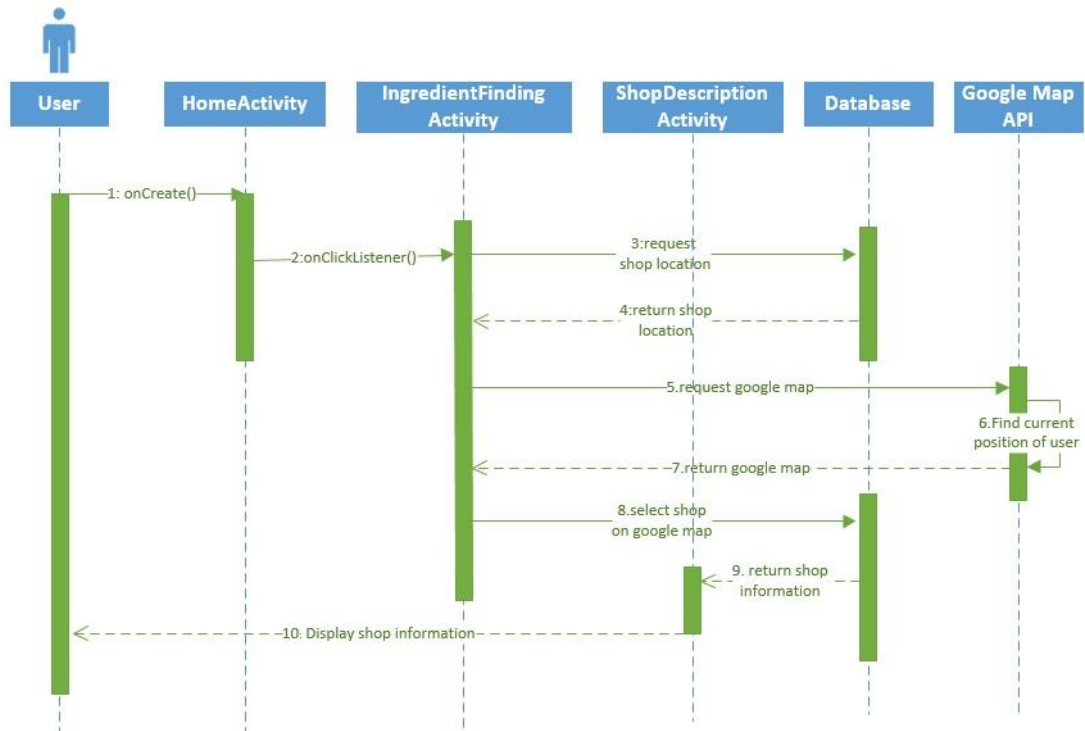
Figure 16 SQ13: View ingredient store location information

SQ14: Get direction between user and ingredient store location

SQ14:Get direction between user and ingredient store location
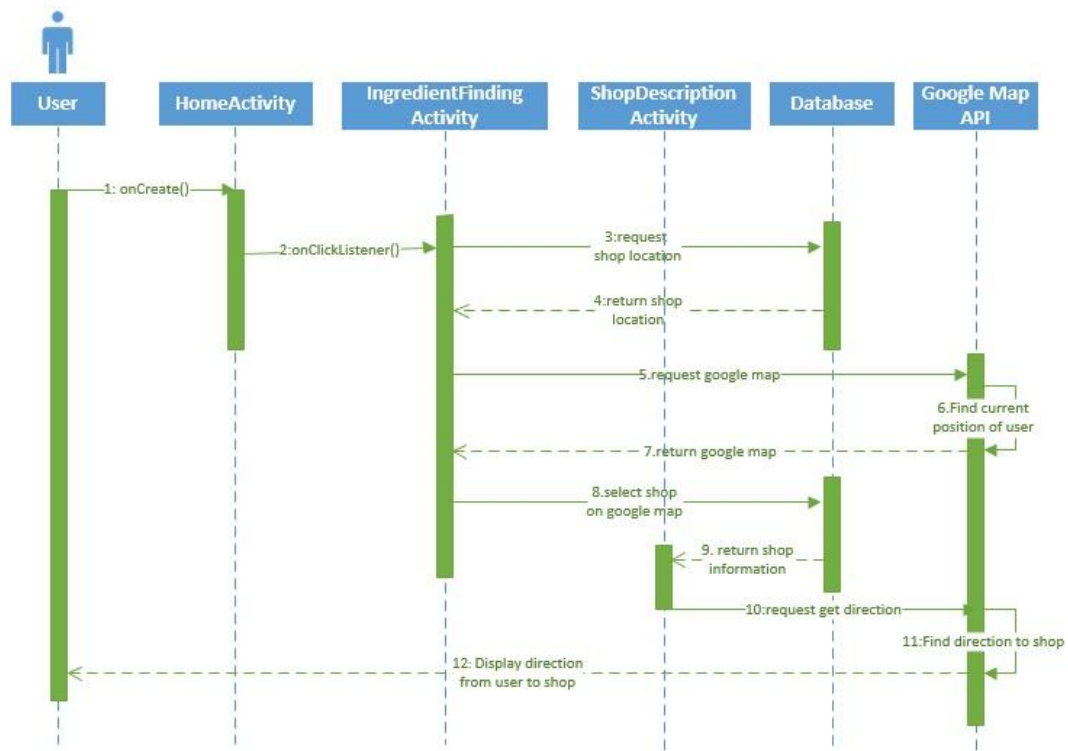
Figure 17 SQ14: Get direction between user and ingredient store location

SQ15: View list of ingredient

SQ15:View list of ingredient
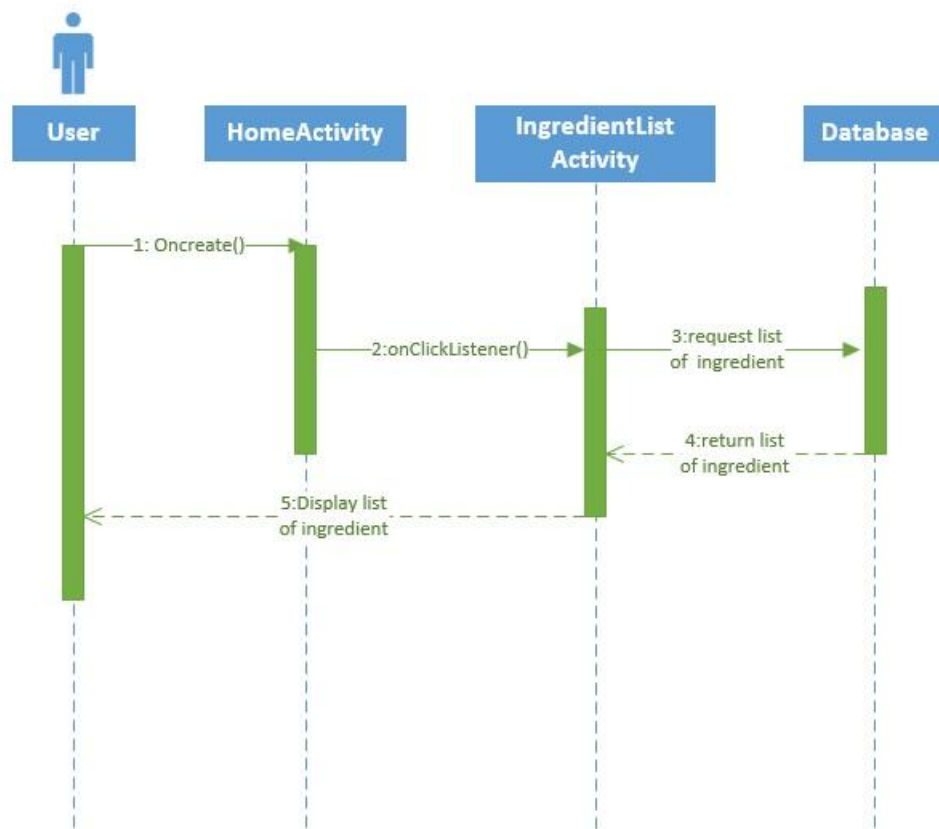
Figure 18 SQ15: View list of ingredient

SQ16: View ingredient information
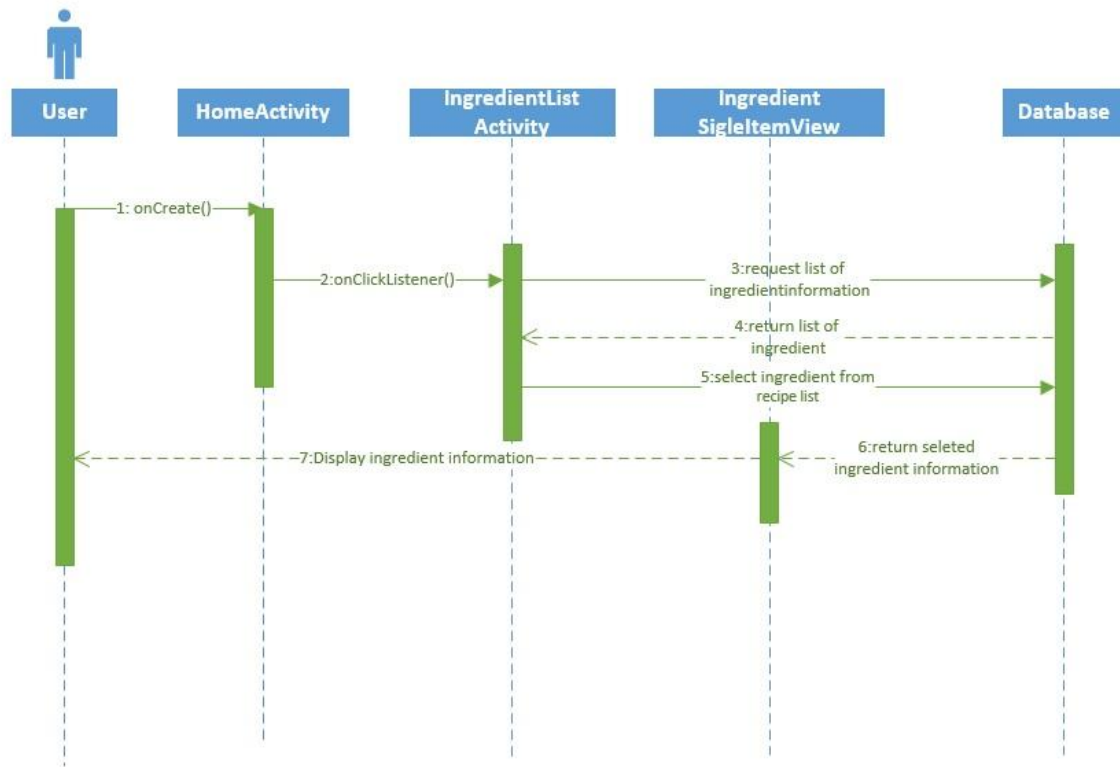
Figure 19 SQ16: View ingredient information
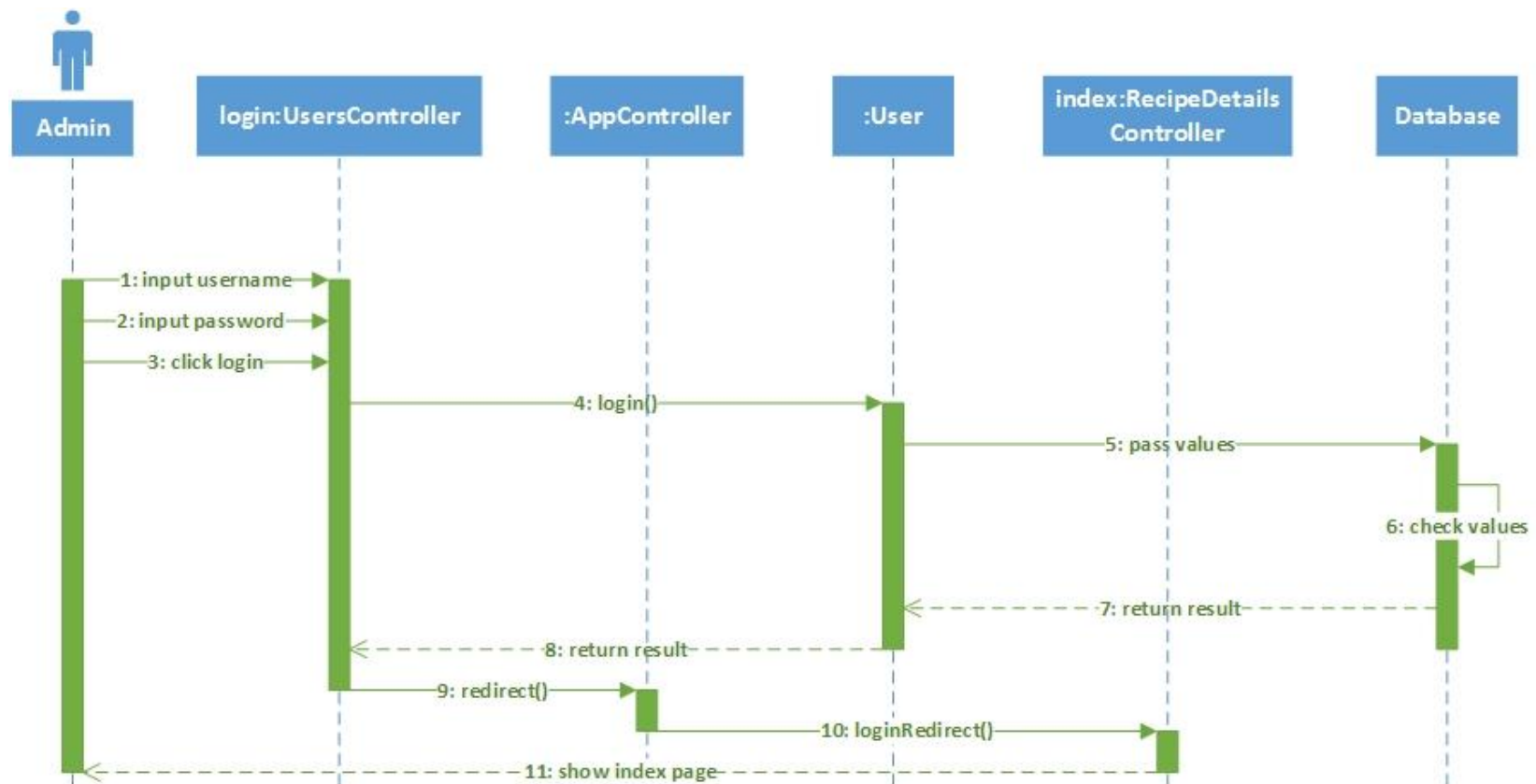
SQ17: Admin logins

Figure 20 SQ17: Admin logins
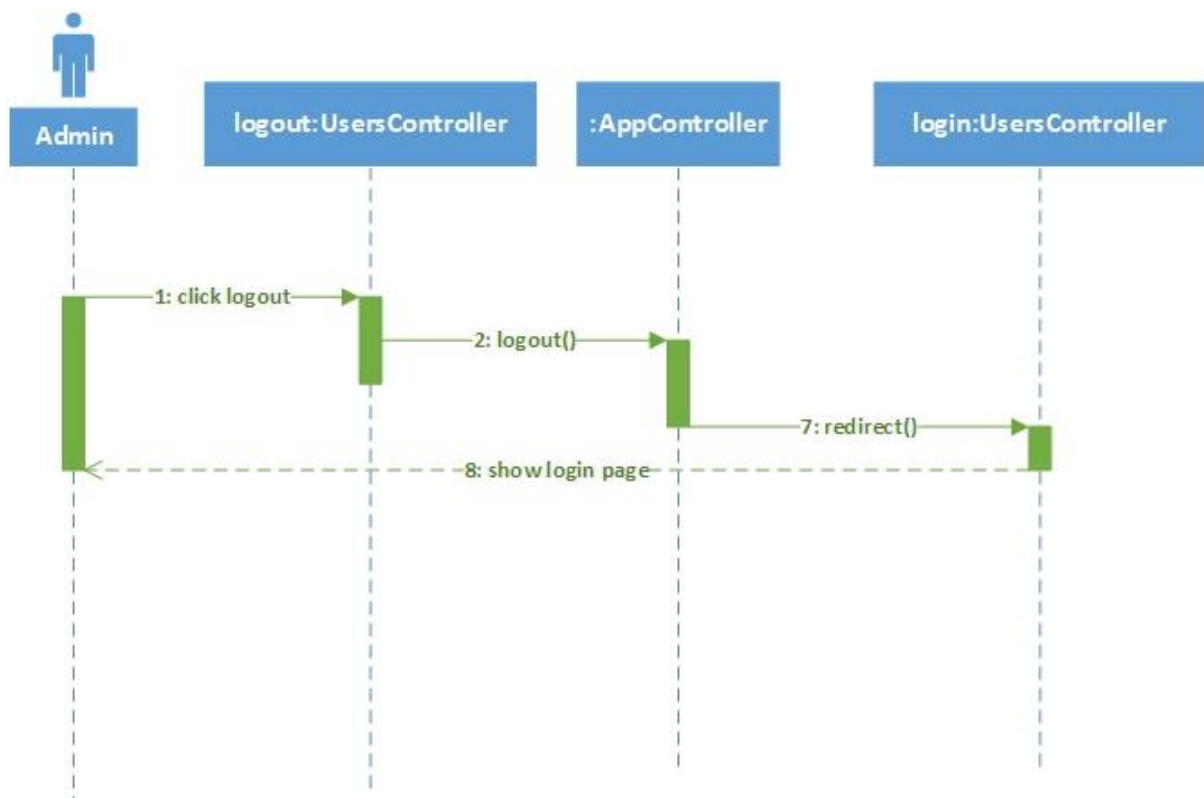
SQ18: Admin logouts

Figure 21 SQ18: Admin logouts

SQ19: Admin views recipe detail
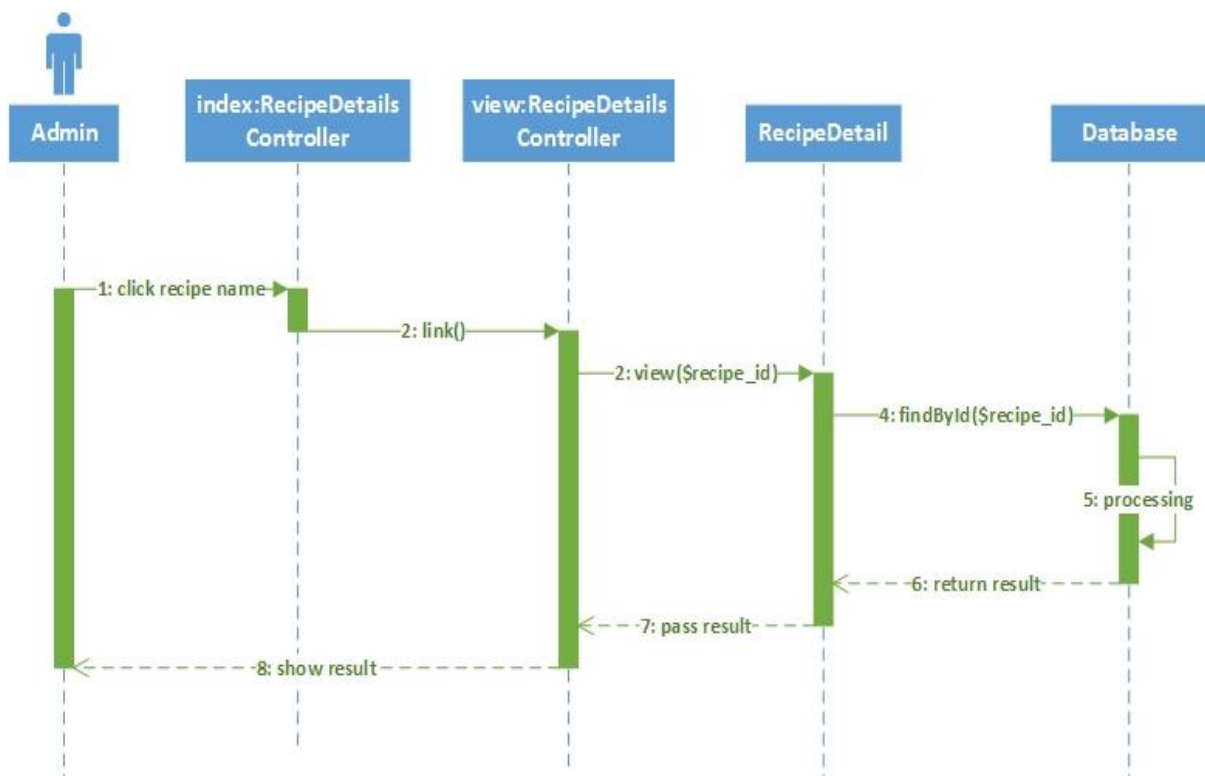
Figure 22 SQ19: Admin views recipe detail

SQ20: Admin adds recipe detail
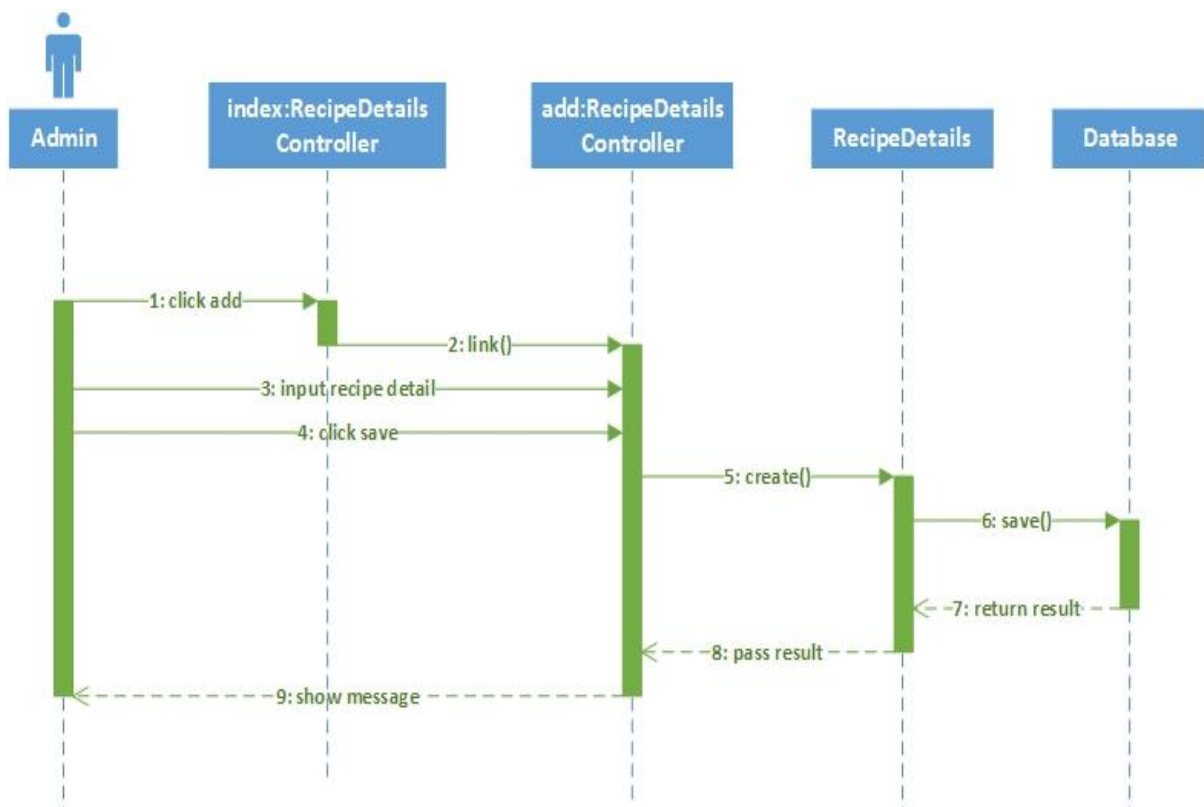
Figure 23 SQ20: Admin adds recipe detail

SQ21: Admin edits recipe detail
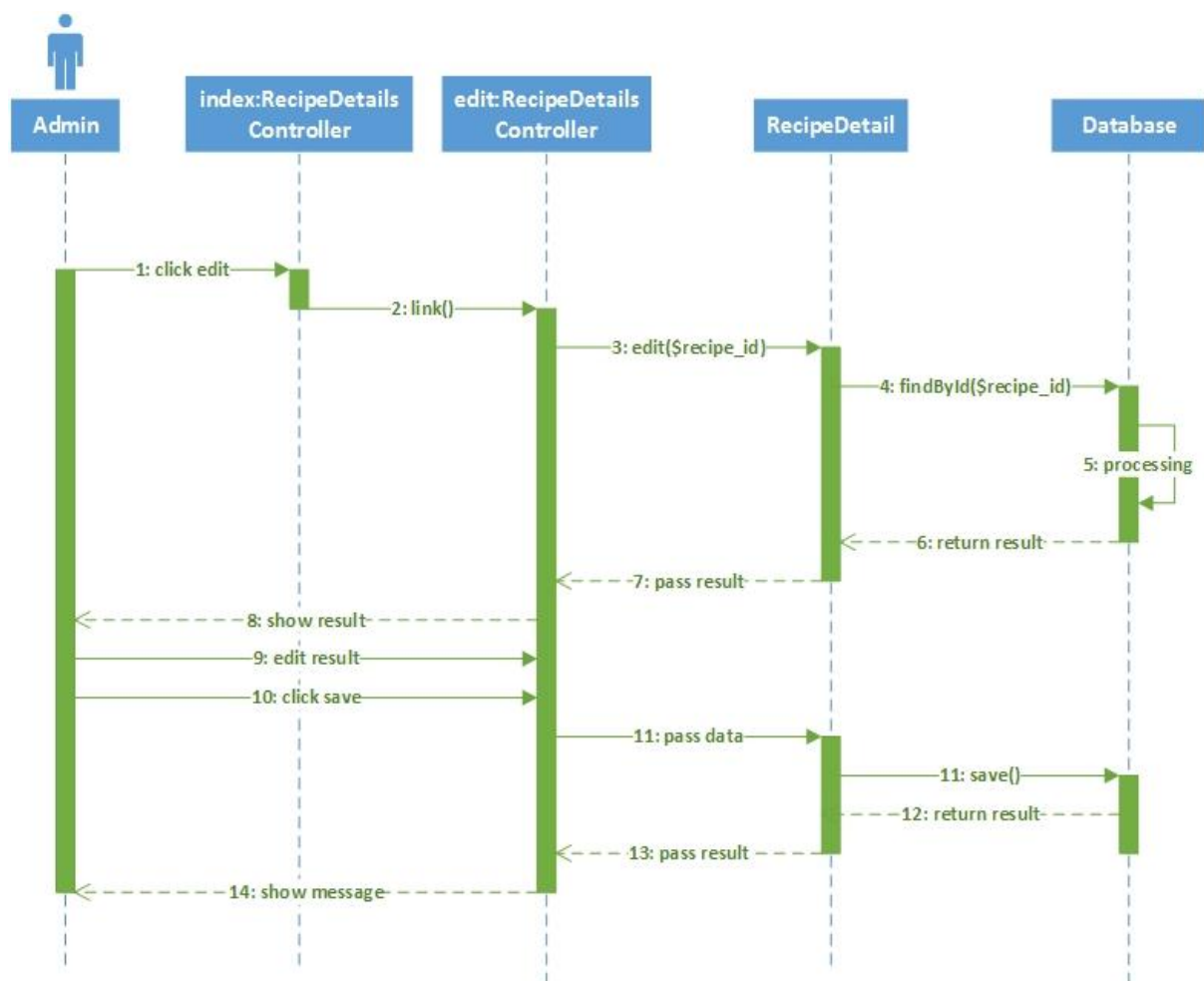
Figure 24 SQ21: Admin edits recipe detail

SQ22: Admin deletes recipe detail
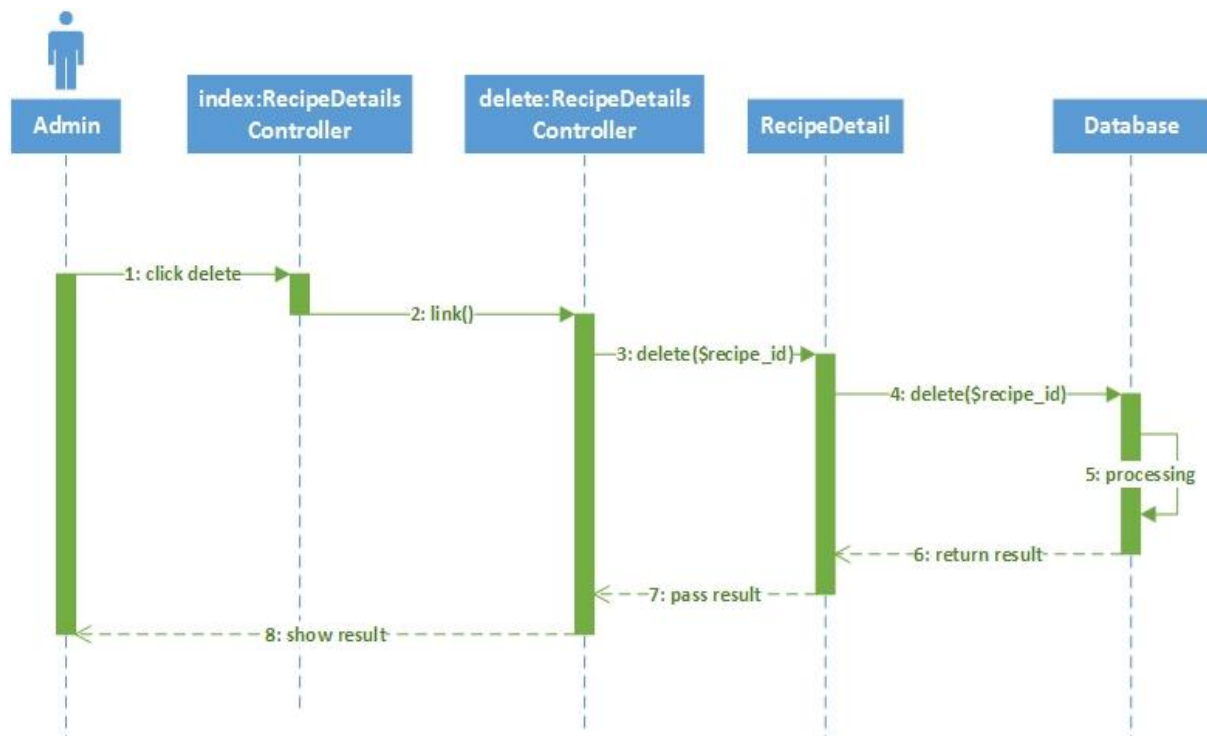
Figure 25 SQ22: Admin delete recipe detail

SQ23: Admin views ingredient detail
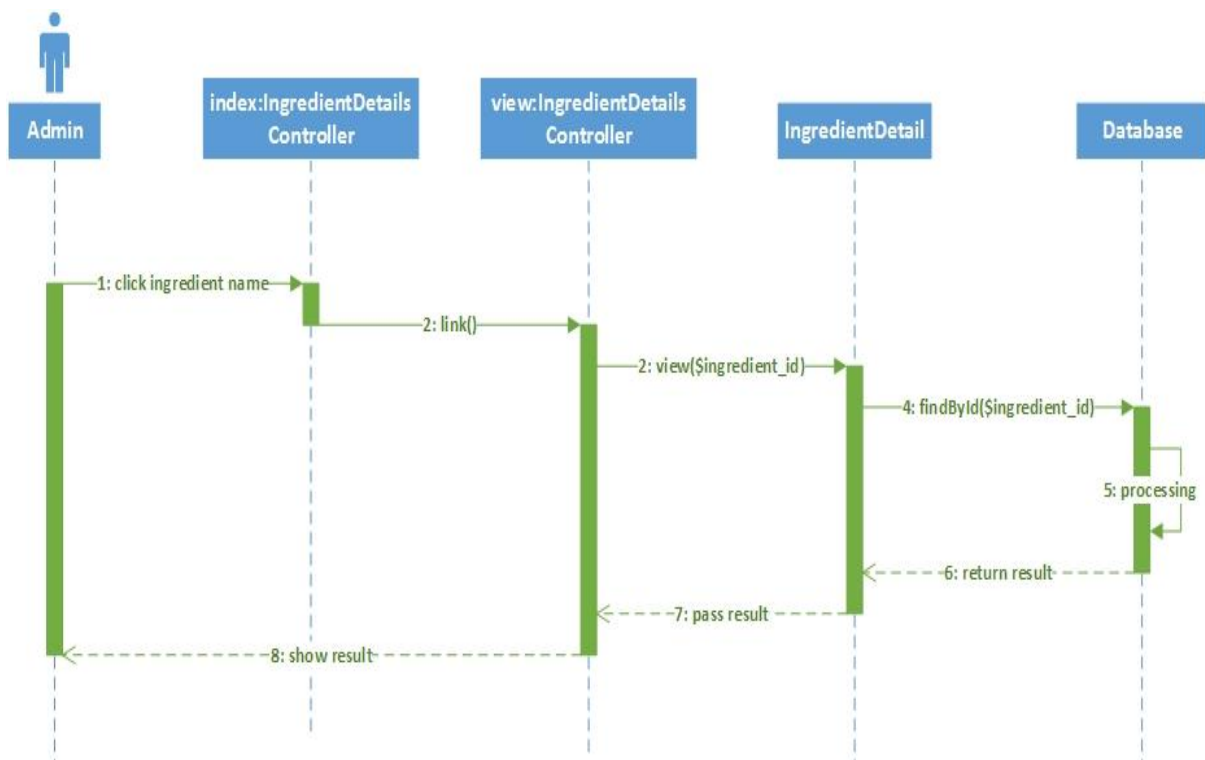
Figure 26 SQ23: Admin views ingredient detail

SQ24: Admin adds ingredient detail
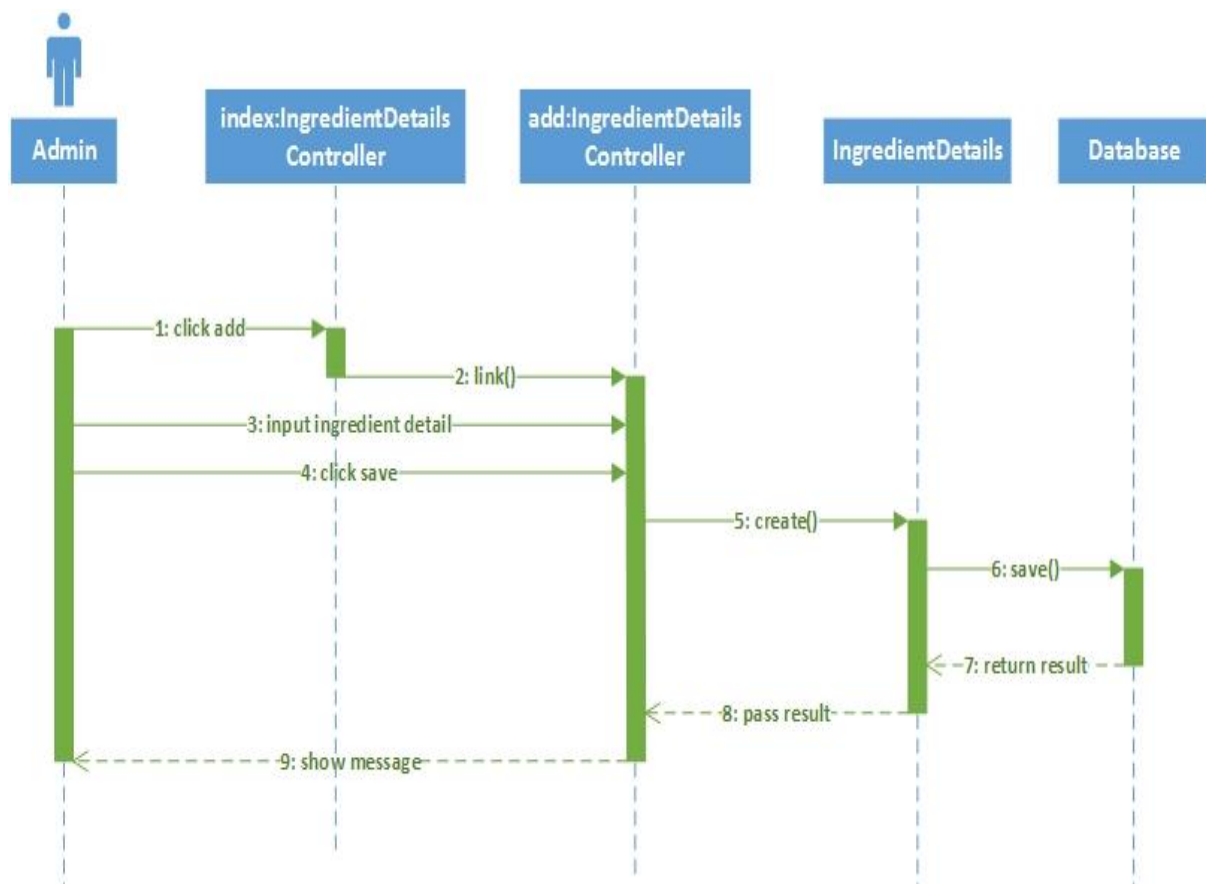
Figure 27 SQ24: Admin adds ingredient detail

SQ25: Admin edits ingredient detail
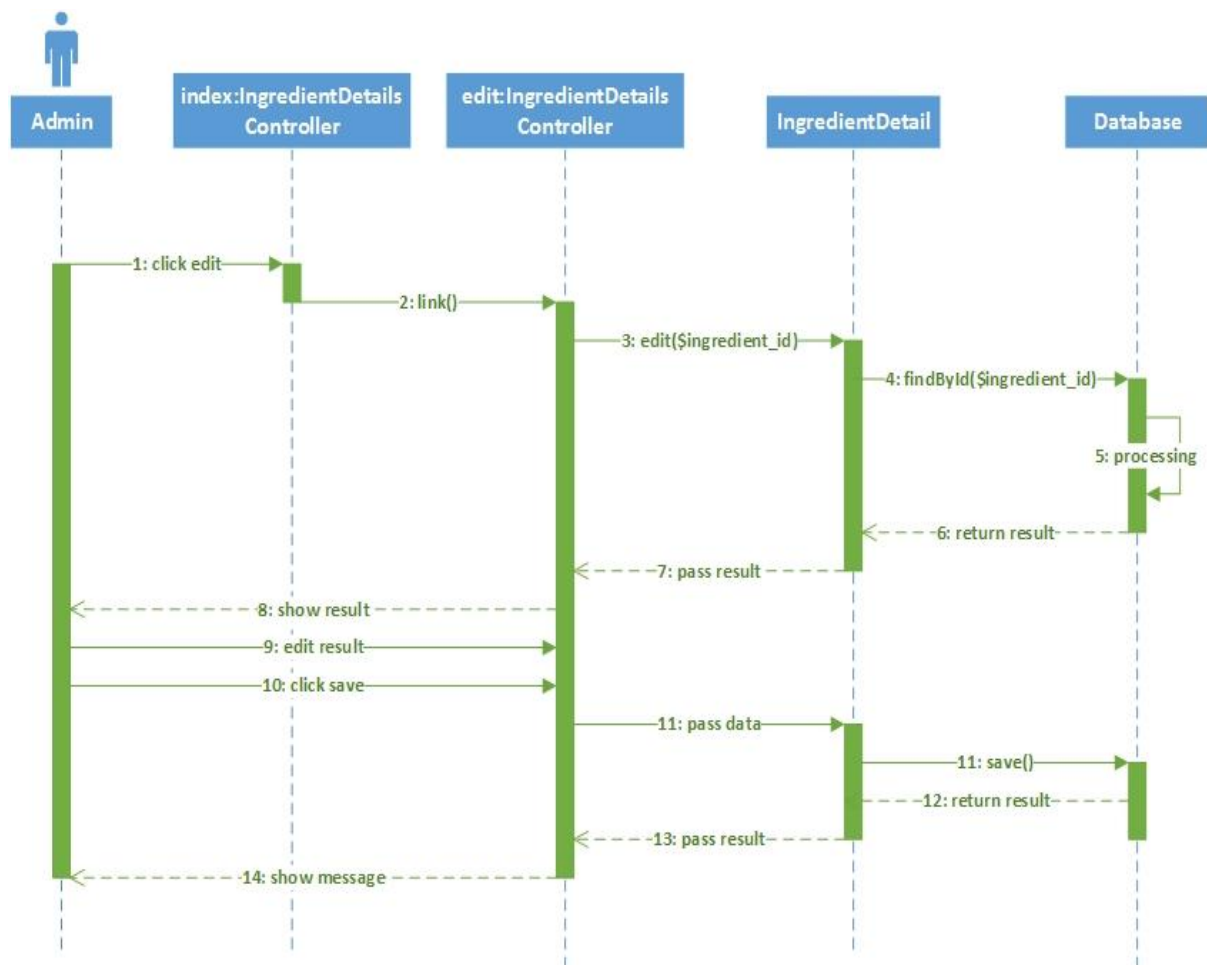
Figure 28 SQ25: Admin edits ingredient detail

SQ26: Admin deletes ingredient detail
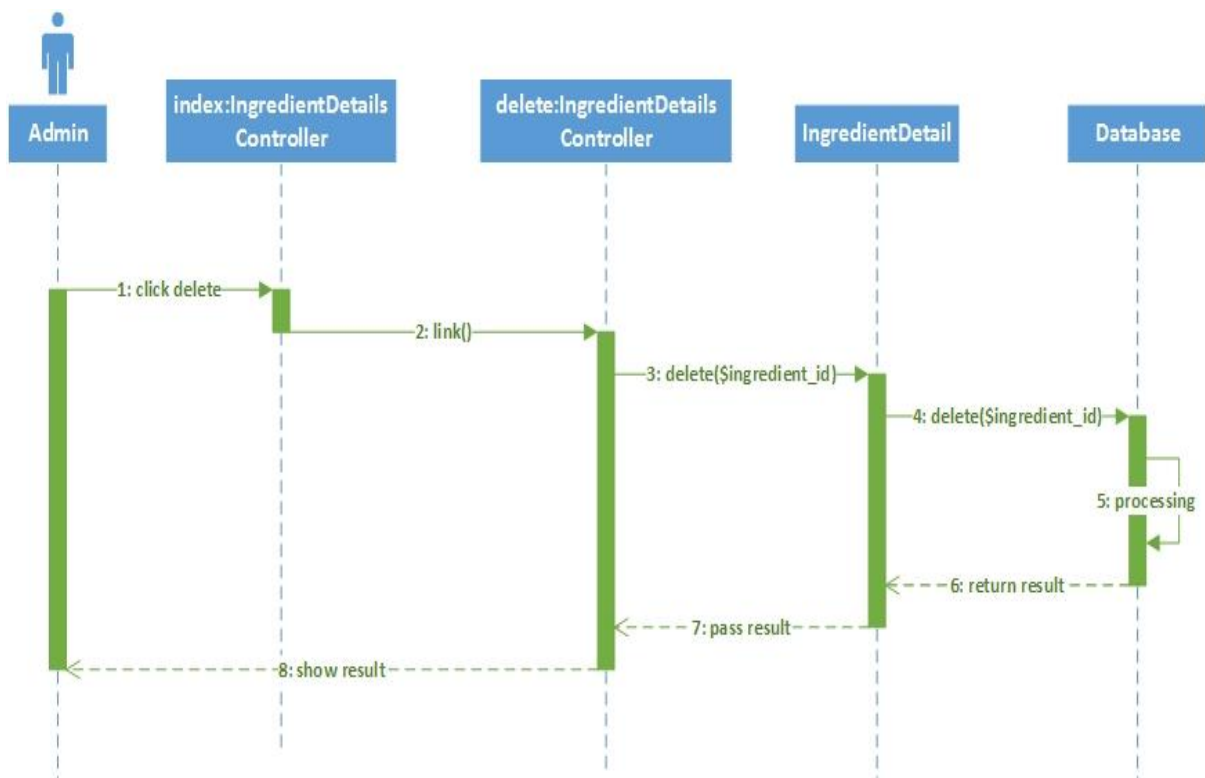
Figure 29 SQ26: Admin deletes ingredient detail

SQ27: Admin adds ingredient items
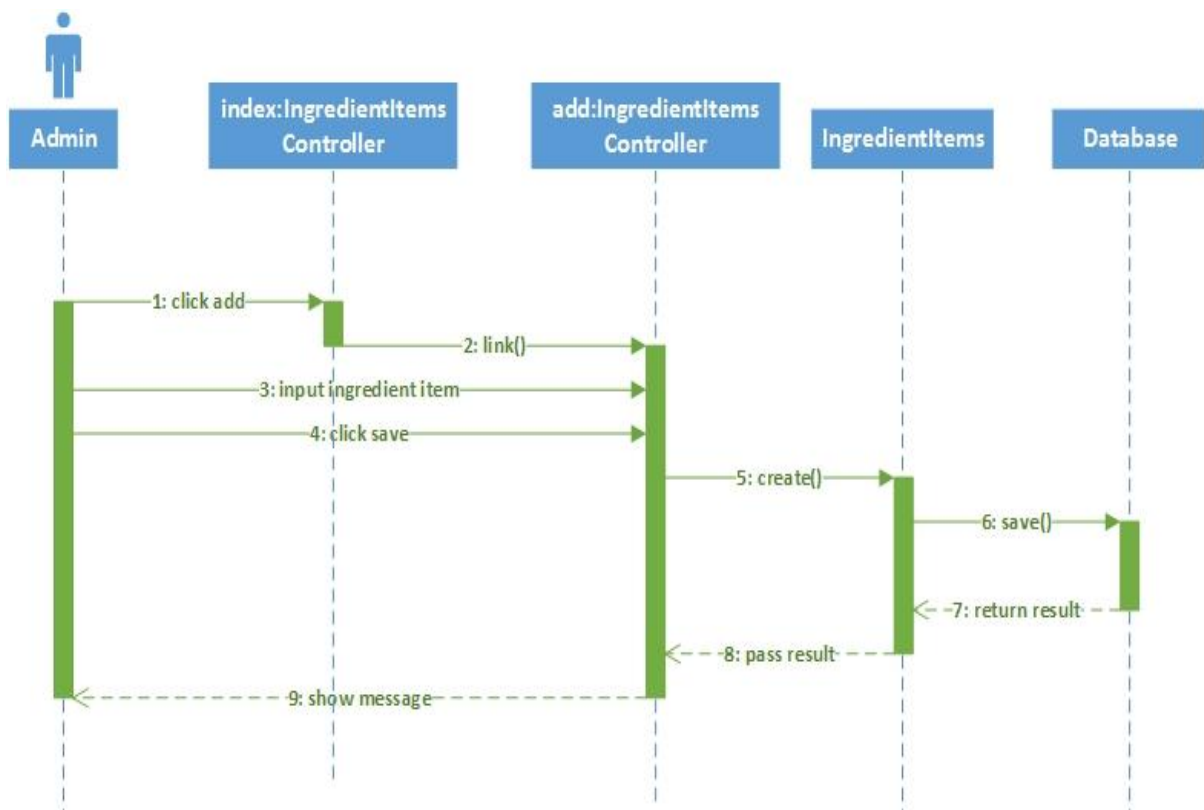
Figure 30 SQ27: Admin adds ingredient items.

SQ28: Admin edits ingredient items
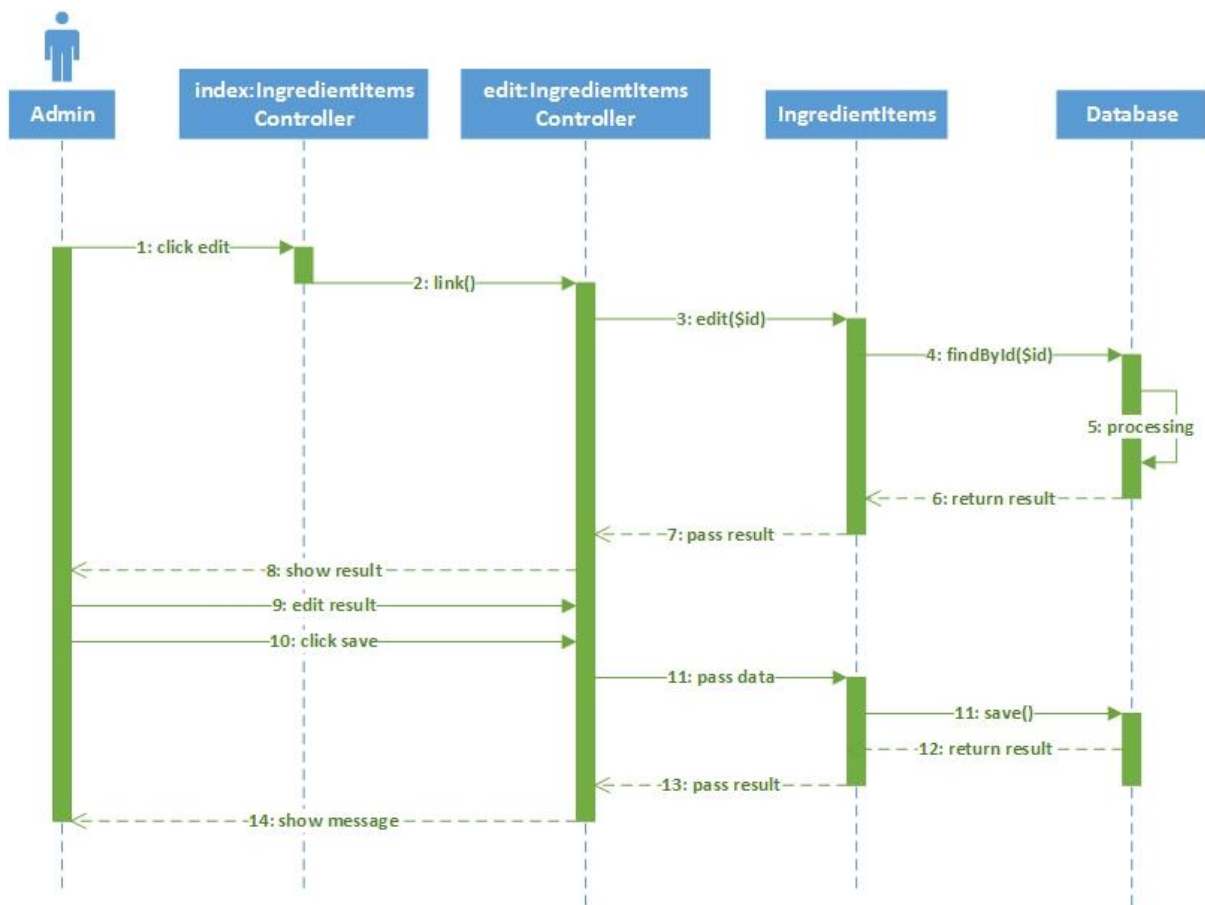
Figure 31 SQ28: Admin edits ingredient items.

SQ29: Admin deletes ingredient items
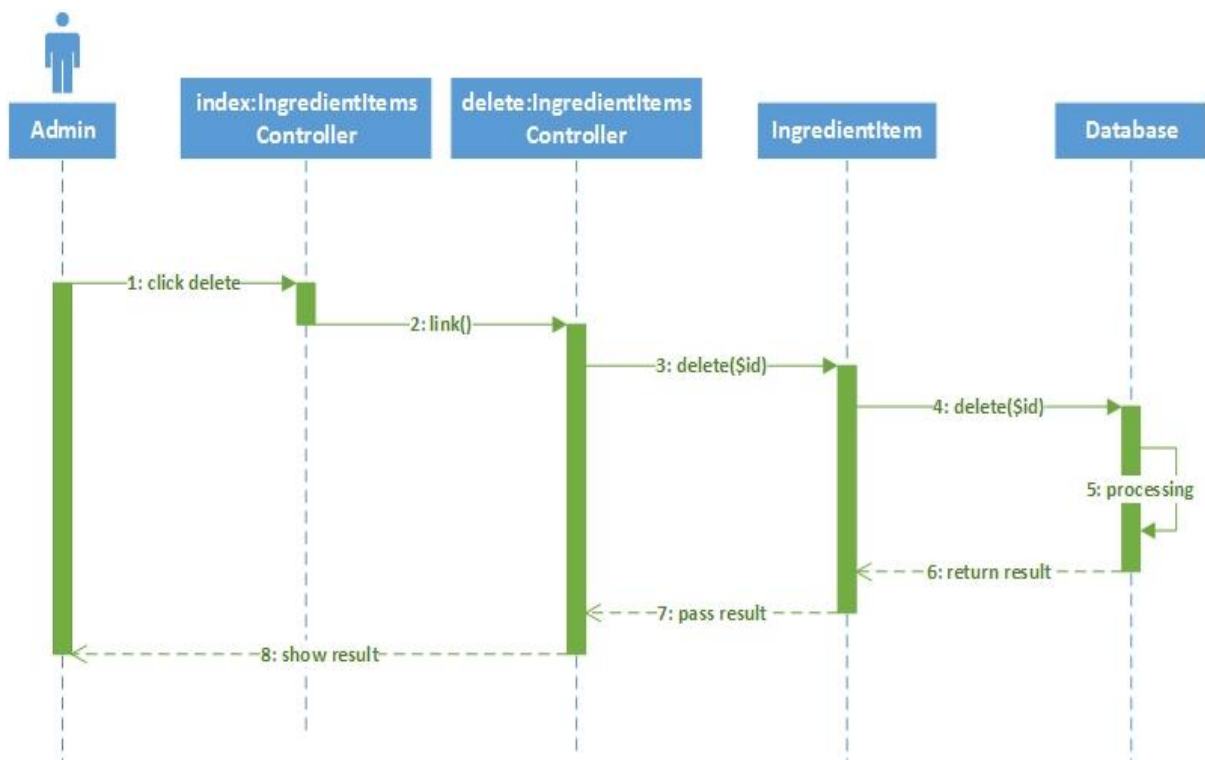
Figure 32 SQ29: Admin deletes ingredient items.

SQ30: Admin views cooking method
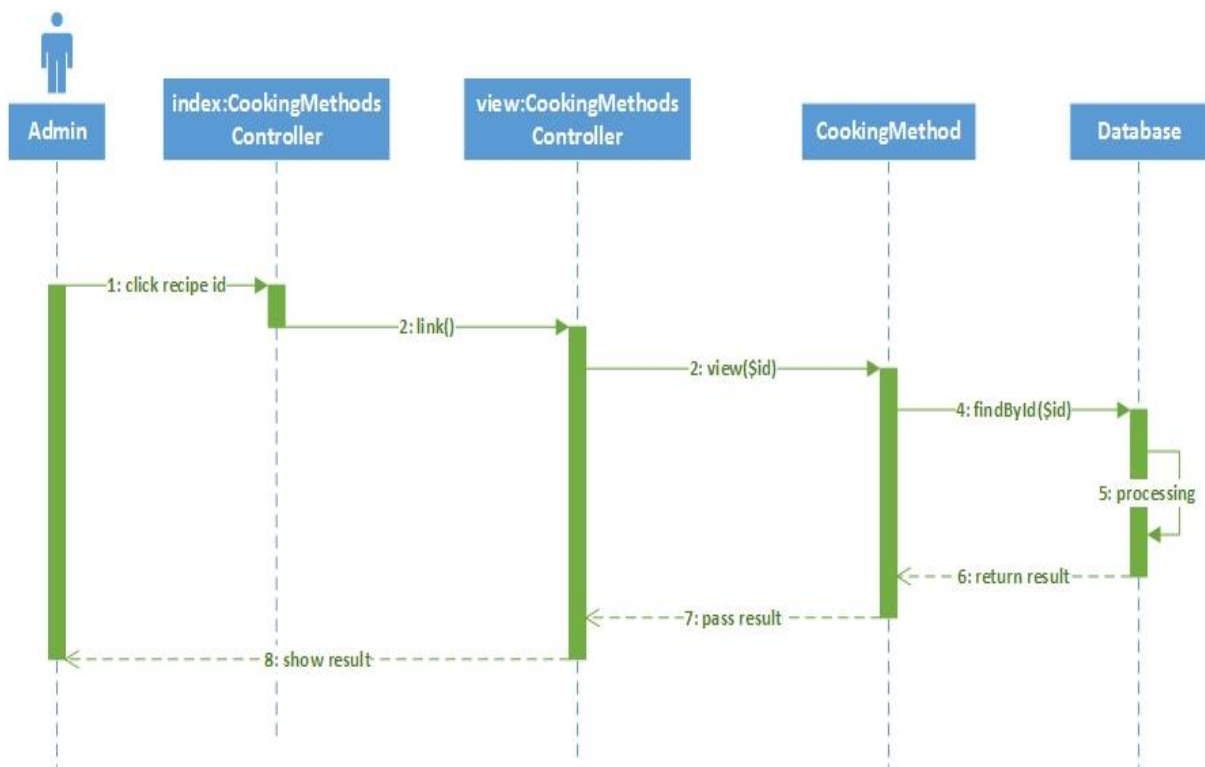
Figure 33 SQ30: Admin views cooking method

SQ31: Admin adds cooking method
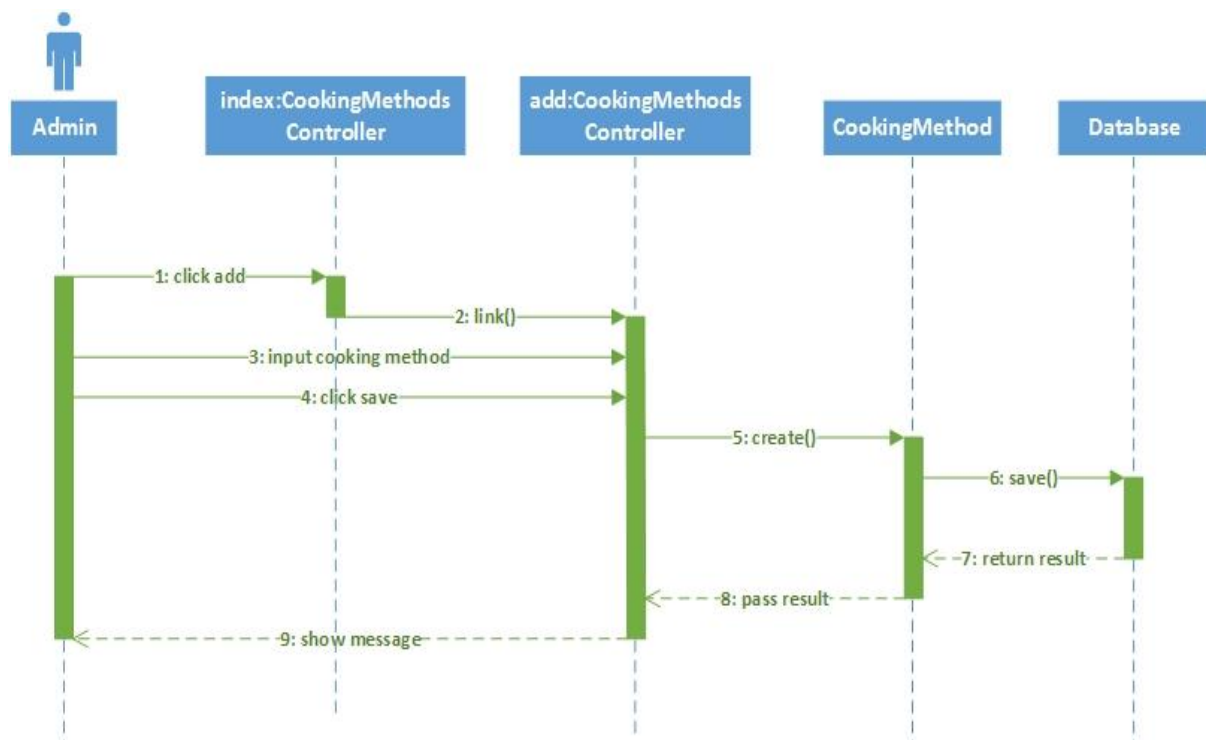
Figure 34 SQ31: Admin adds cooking method

SQ32: Admin edits cooking method
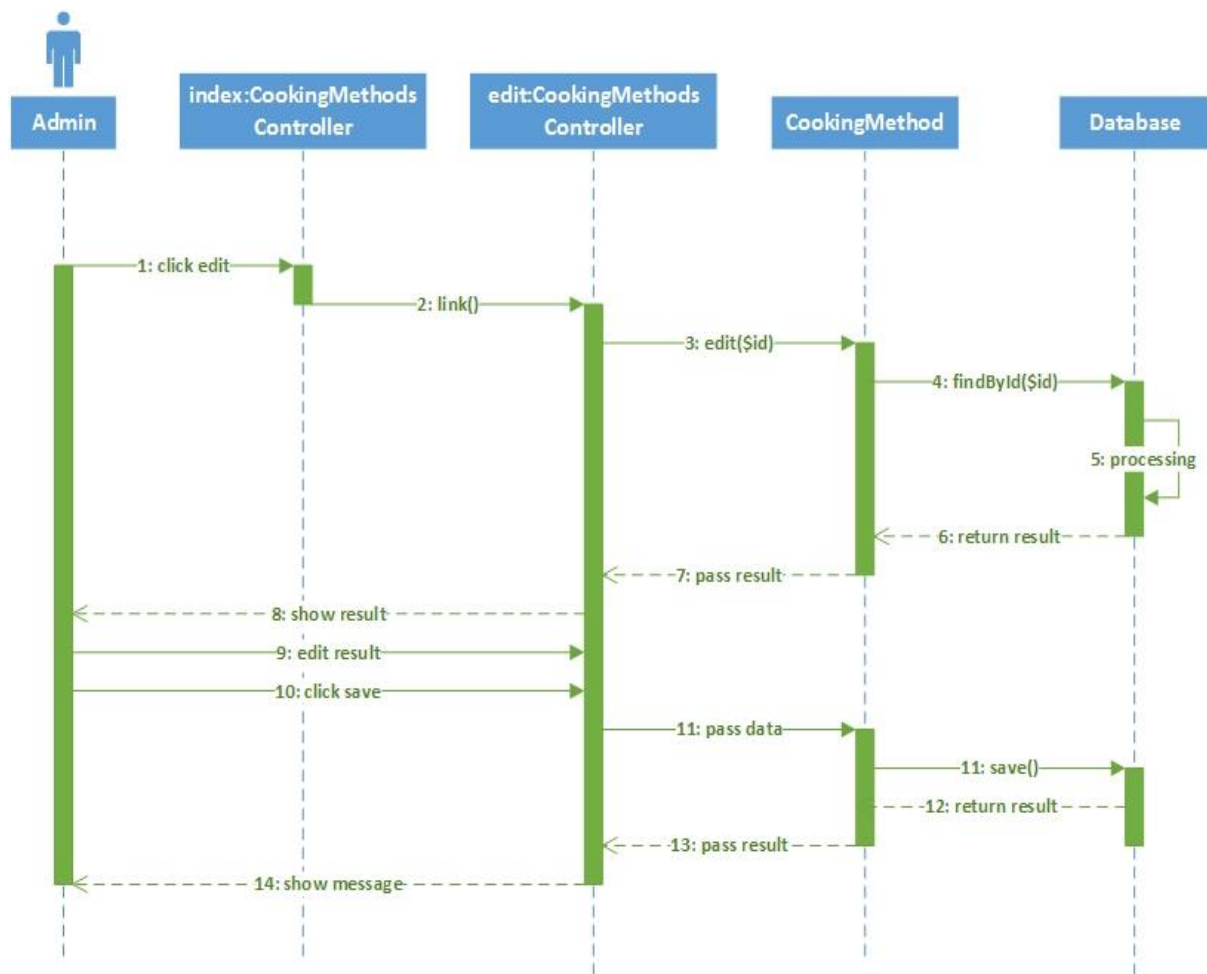
Figure 35 SQ32: Admin edits cooking method

SQ33: Admin deletes cooking method
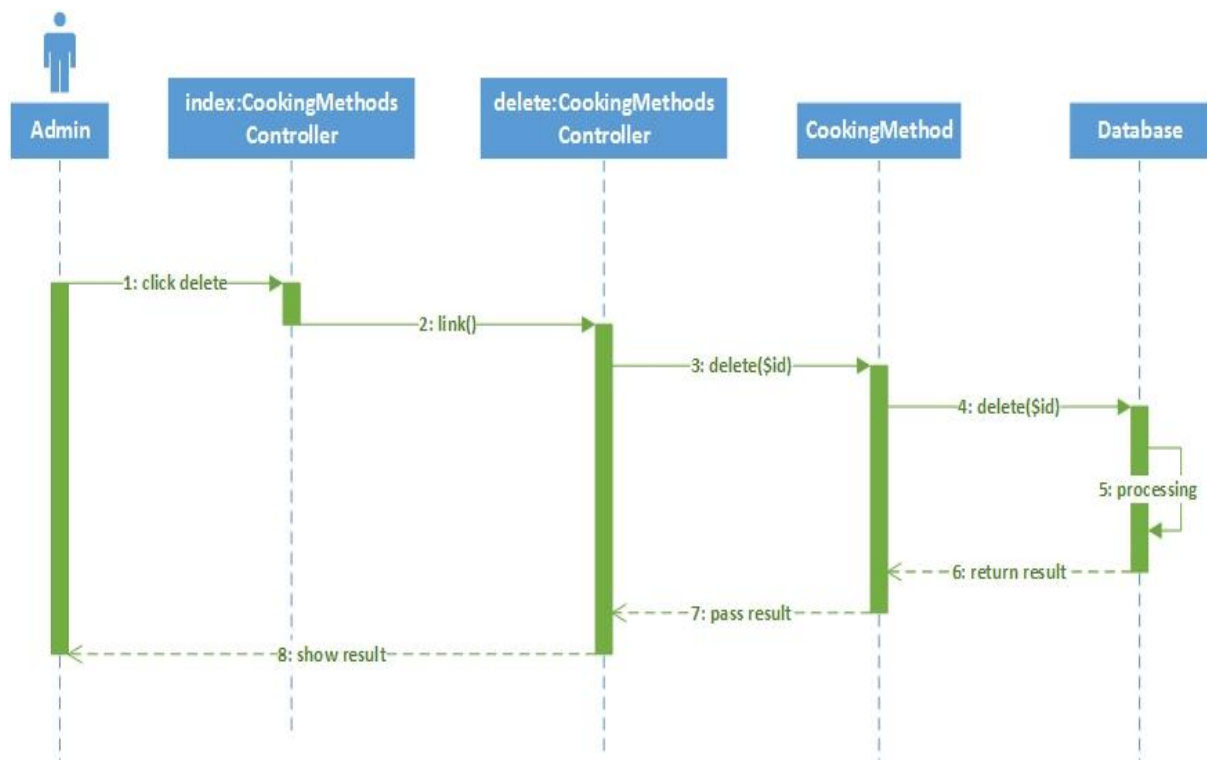
Figure 36 SQ33: Admin deletes cooking method

SQ34: Admin views ingredient store location
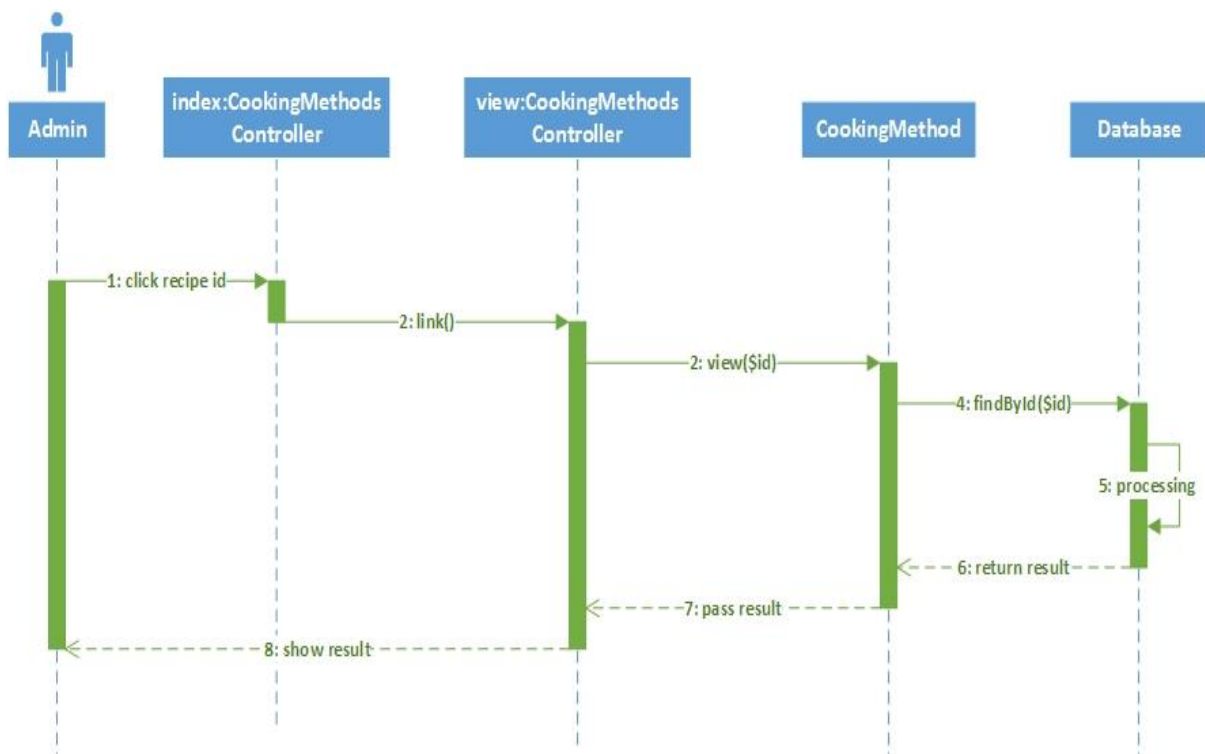
Figure 37 SQ34: Admin views ingredient store location

SQ35: Admin adds ingredient store location
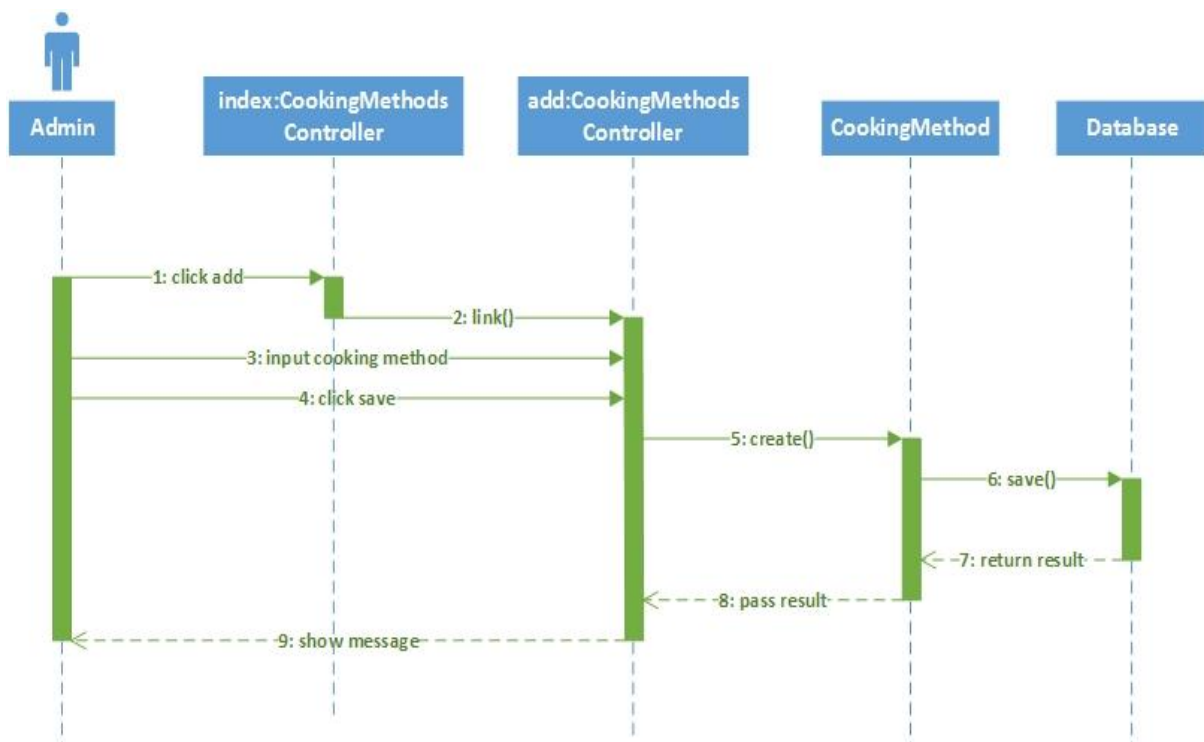
Figure 38 SQ35: Admin adds ingredient store location

SQ36: Admin edits ingredient store location
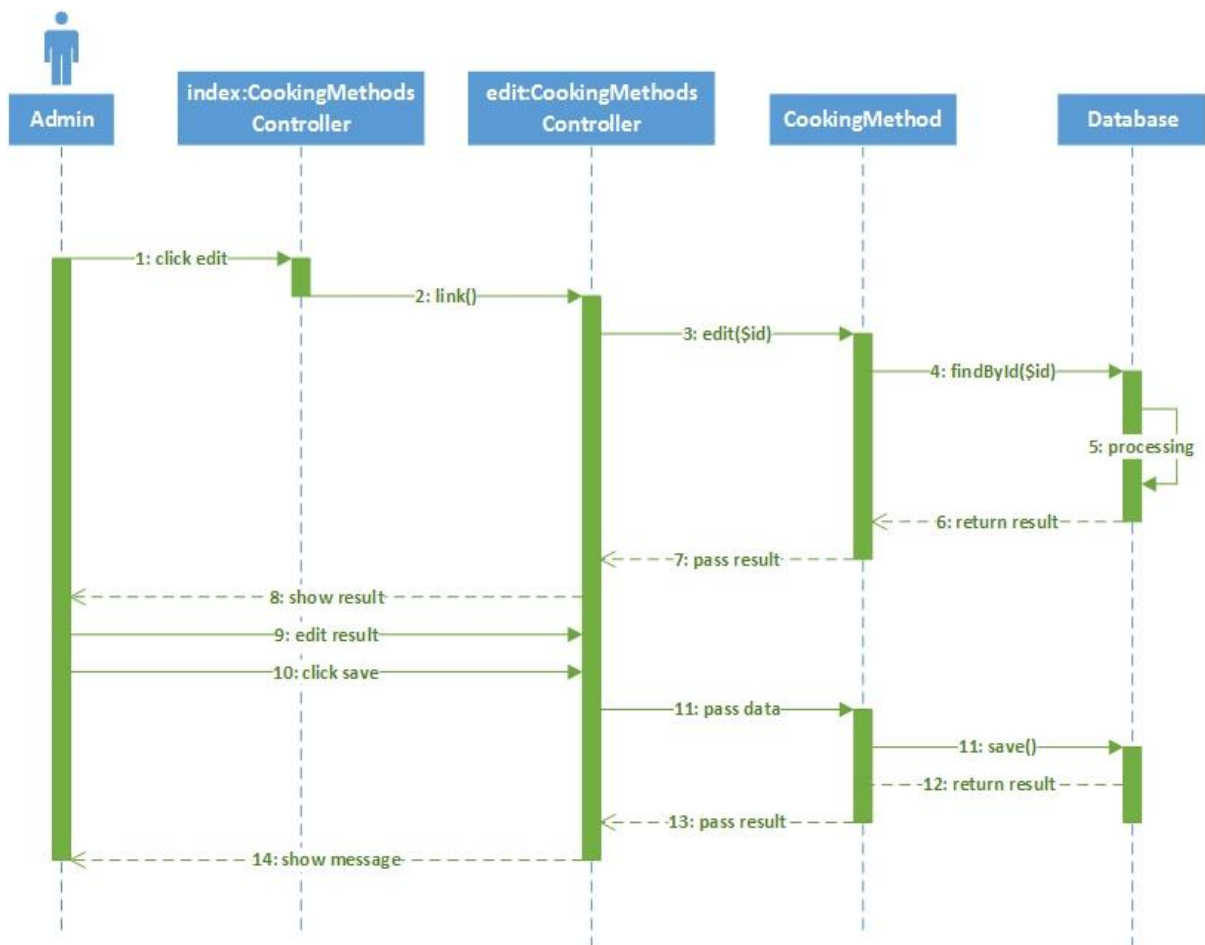
Figure 39 SQ36: Admin edits ingredient store location

SQ37: Admin deletes ingredient store location
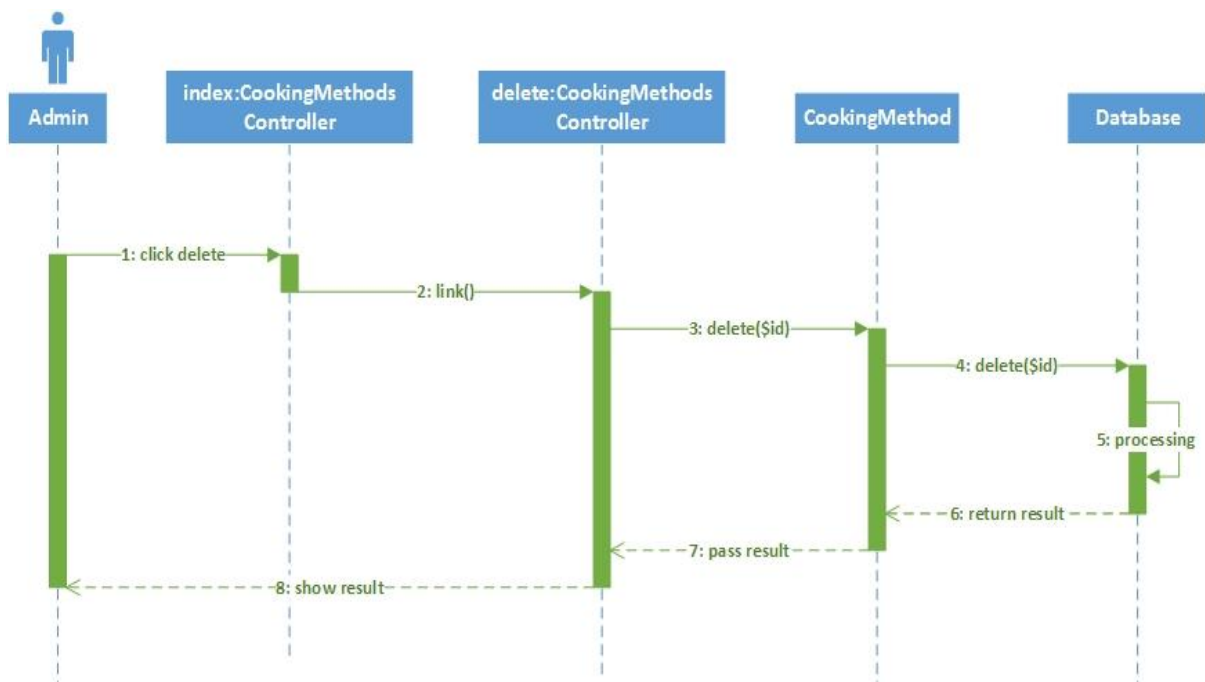
Figure 40 SQ37: Admin deletes ingredient store location

4. Data architecture

4.1 ER diagram
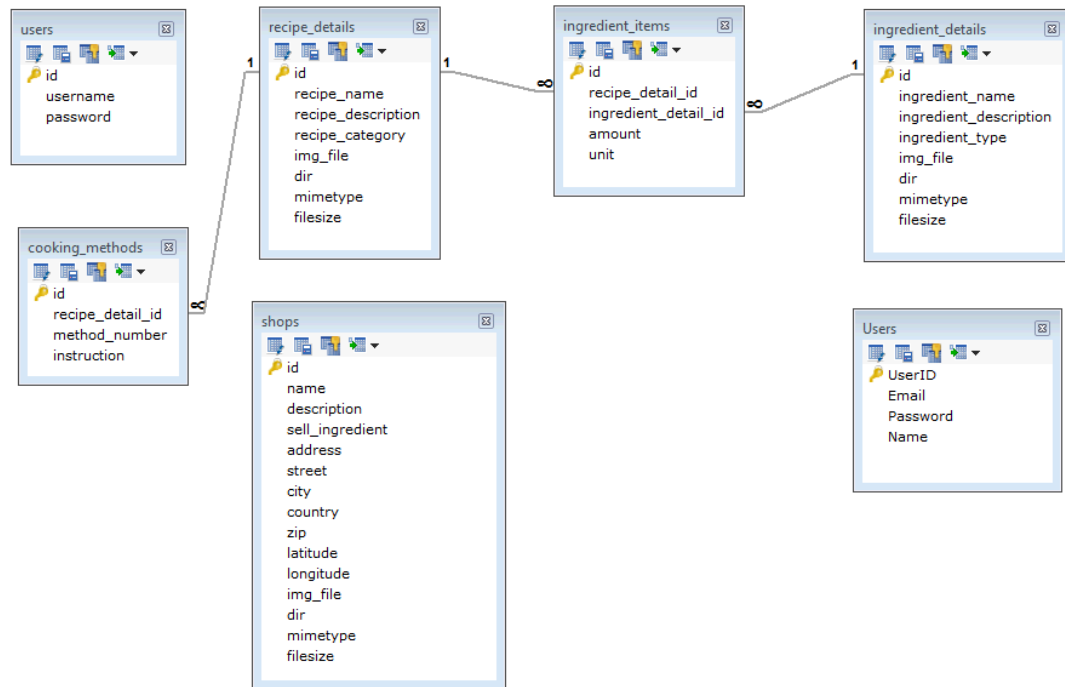
Figure 46 ER diagram

4.2 Database design

DB-01 users

Field	Type	Extra
id**(PK)**	int(10)	Auto increment
username	varchar(20)	
password	varchar(20)	

DB-02 recipe_details

Field	Type	Extra
id**(PK)**	int(10)	Auto increment
recipe_name	varchar(20)	
recipe_description	text	
recipe_category	varchar(20)	
img_file	varchar(255)	
dir	varchar(255)	
mimetype	varchar(255)	
filesize	int(11)	

DB-03 ingredient_details

Field	Type	Extra
id**(PK)**	int(10)	Auto increment
ingredient_name	varchar(50)	
ingredient_description	text	
ingredient_type	varchar(20)	
img_file	varchar(255)	
dir	varchar(255)	
mimetype	varchar(255)	
filesize	int(11)	

DB-04 ingredient_items

Field	Type	Extra
id**(PK)**	int(11)	**Auto increment**
recipe_detail_id**(FK)**	int(11)	
ingredient_detail_id**(FK)**	int (11)	
amount	varchar(20)	
unit	varchar(10)	

DB-05 cooking_methods

Field	Type	Extra
id(**PK**)	int(11)	Auto increment
recipe_detail_id(**FK**)	Int(10)	
method_number	int(11)	
instruction	text	

DB-06 shops

Field	Type	Extra
id**(PK)**	int(5)	Auto increment
name	varchar(5)	
description	text	
sell_ingredient	text	
address	text	
street	varchar(50)	
city	varchar(50)	
state	varchar(50)	
zip	varchar(10)	
latitude	float	(10,6)
longitude	float	(10,6)
img_file	varchar(255)	
dir	varchar(255)	
mimetype	varchar(255)	
filesize	int(11)	

DB-07 Users

Field	Type	Extra
UserID**(PK)**	int(5)	Auto increment
Email	varchar(50)	
Password	varchar(20)	
Name	varchar(10)	

5. User interface design

5.1 Web Service

UI-01 login.ctp

Figure 47 UI-01 login.ctp

This is a first page when admin open a web application. It will login form.

UI-02 index.php

Figure 48 UI-02 index.ctp

This is a second page after the admin logging into a web application. It will display index.ctp and shows list of recipe name from database. This page will shows a list of menu buttons, menu name, username, logout button, table from database and provides the links for add, edit, delete functions. When click on menu button the structure of this page will same, but difference on data that used from database. (e.g. click on "Ingredient Detail" button, it will show table data from ingredient detail table.)

UI-03 view.ctp

Figure 49 UI-03 view.ctp

This page will show the details of data when admin click on the link in a table. (e.g. click on recipe name, It will show all of details from database.)

UI-04 add.ctp

Figure 50 UI-04 add.ctp

This is a add detail page. It will display when admin click on link "Add" as any index page. It will show add detail form that show follow by field name in database.

UI-05 edit.ctp

Figure 51 UI-05 edit.ctp

When admin click edit icon on action column as any table. This UI will display. It will show detail from database that clicked.

UI-06 shops/index.ctp

Figure 52 UI-06 shops/index.ctp

This page is difference from other index page as it will show Google Map, without Google Map it show the same thing with any index page.

5.2 Mobile Application

UI-07 Login
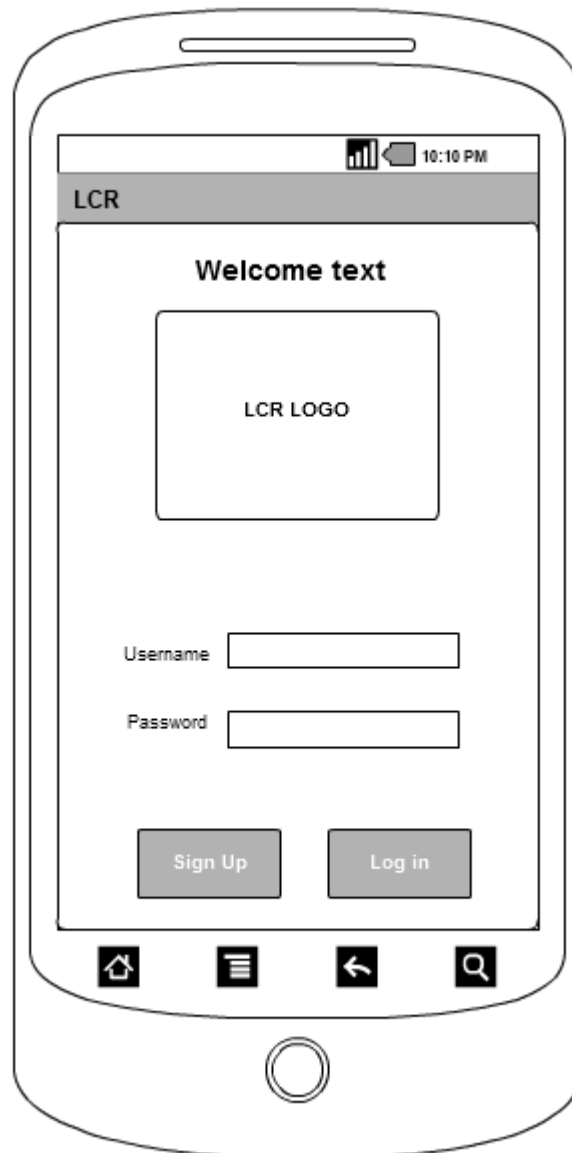
Figure 53 UI-07 Login

This UI is a UI that the customer can login before use our application. It will show a textbox that customer input E-mail and Password.

Document Name	LCR -SDD-V.3.0-docx	Owner	KS	Page	**100 / 117**
Document Type	Software Design Document	Release Date	3 October 2013	Print Date	3 October 2013

UI-08 Register
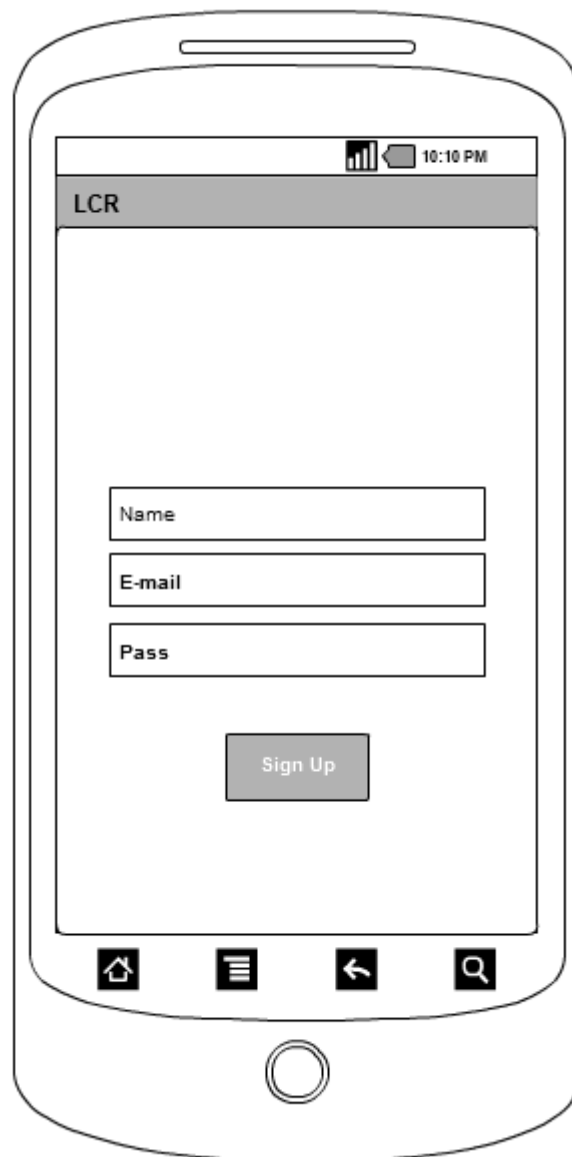
Figure 54 UI-08 Register

   This UI is a UI that the customer can register for using our application. It will show information that used.
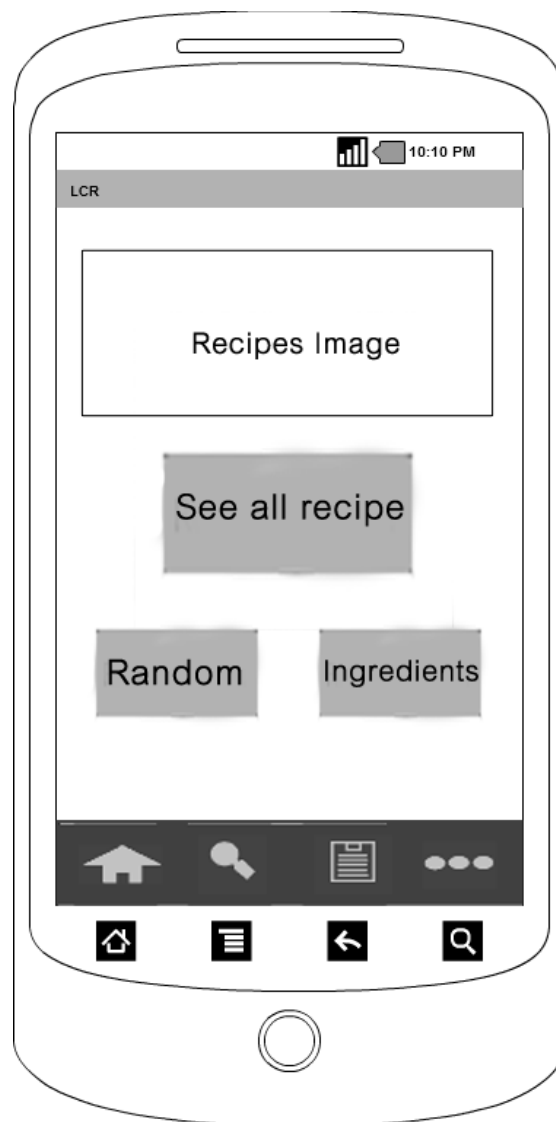
UI-09 Home

Figure 55 UI-09 Home

This UI will show after the customer access to our application. It will show Recommended Menu, Category, Random function and See all button. See all button is a button that show all list of our recipe. all of page has a menu bar. In menu bar have Home, Search, shopping List, and More button.
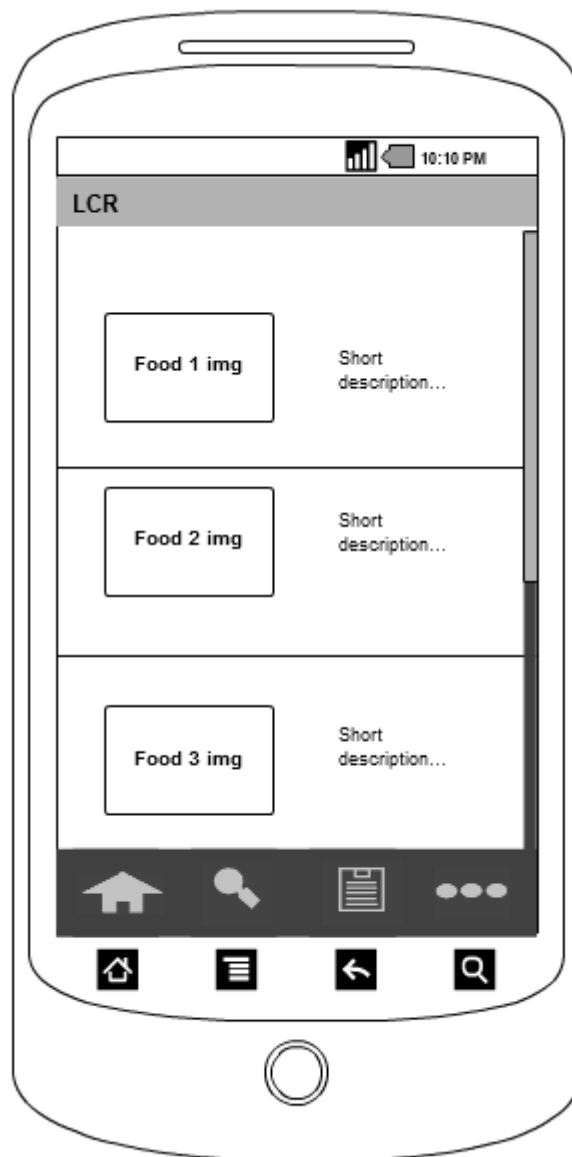
UI-10 Recipe List

Figure 56 UI-10 Recipe List

This UI will show list of recipe that we have. It will show after the customer select See all button. The customer can see all recipe each of category.
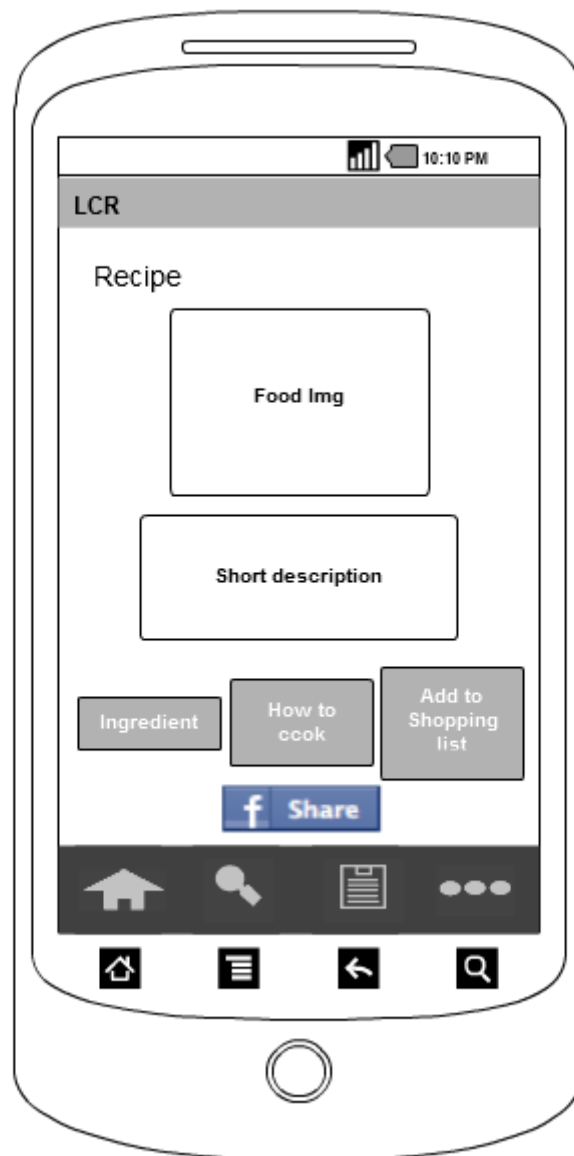
UI-11 Recipe Detail

Figure 57 UI-11 Recipe Detail

This UI will show after the customer select some recipe. It will show food name, ingredient, how to cook, food image, and food description. This page will provide customer to add food to shopping list by select Add to Shopping List button and share this food by select Share button.
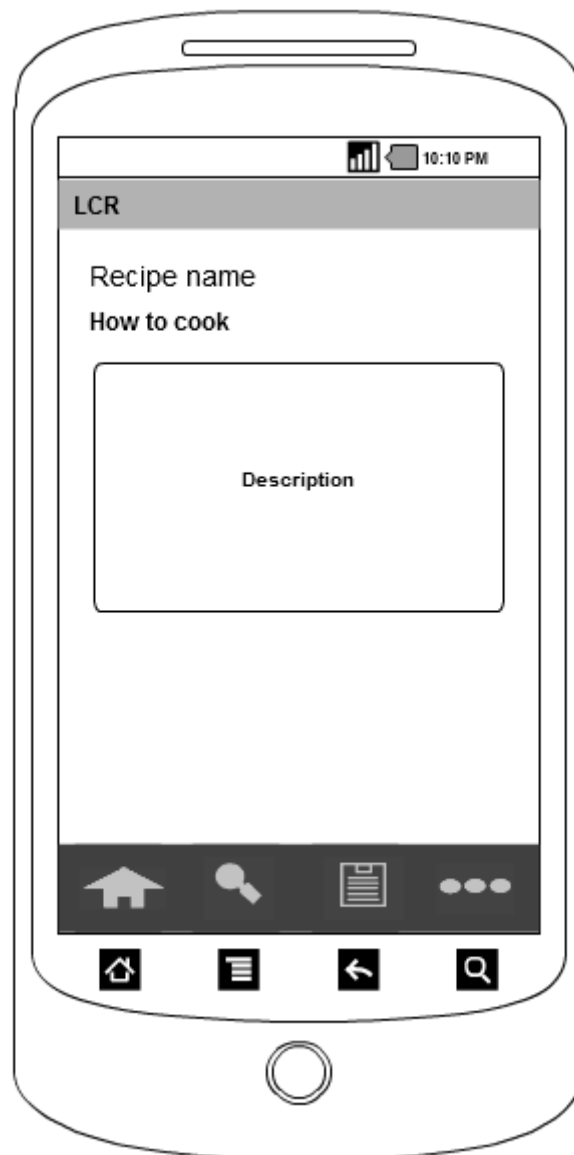
UI-12 How to cook

Figure 58 UI-12 How to cook

This UI will show after the customer select How to cook button from recipe detail page. It will show how to cook description.

UI-13 Ingredient List View
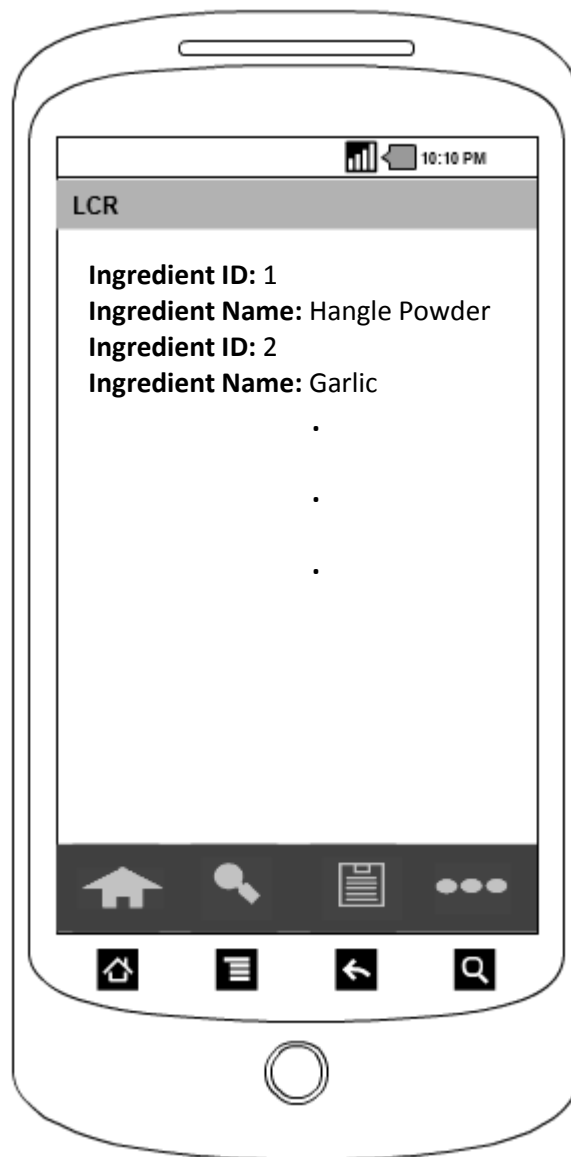
Figure 59 UI-13 Ingredient List View

This UI will show after the customer select Ingredient button from home page. It will show list of ingredient in database. It will show ingredient id and name.

UI-14 Menu Ingredient List
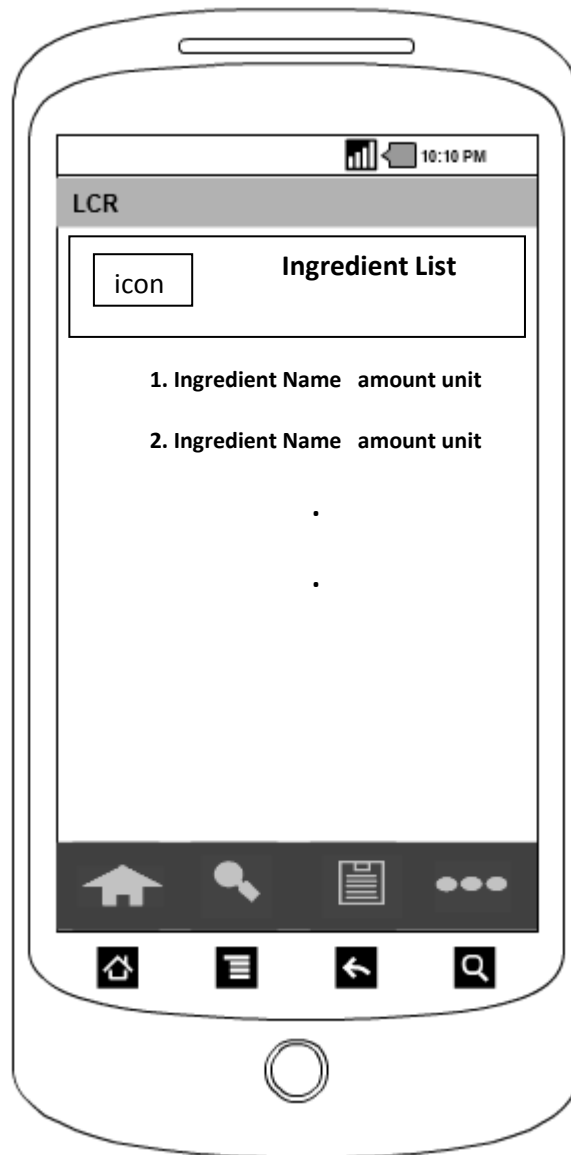
Figure 60 UI-14 Menu Ingredient List

This UI will show after the customer select Ingredient button from recipe detail page. It will show list of needed ingredient.

UI-15 Ingredient Detail
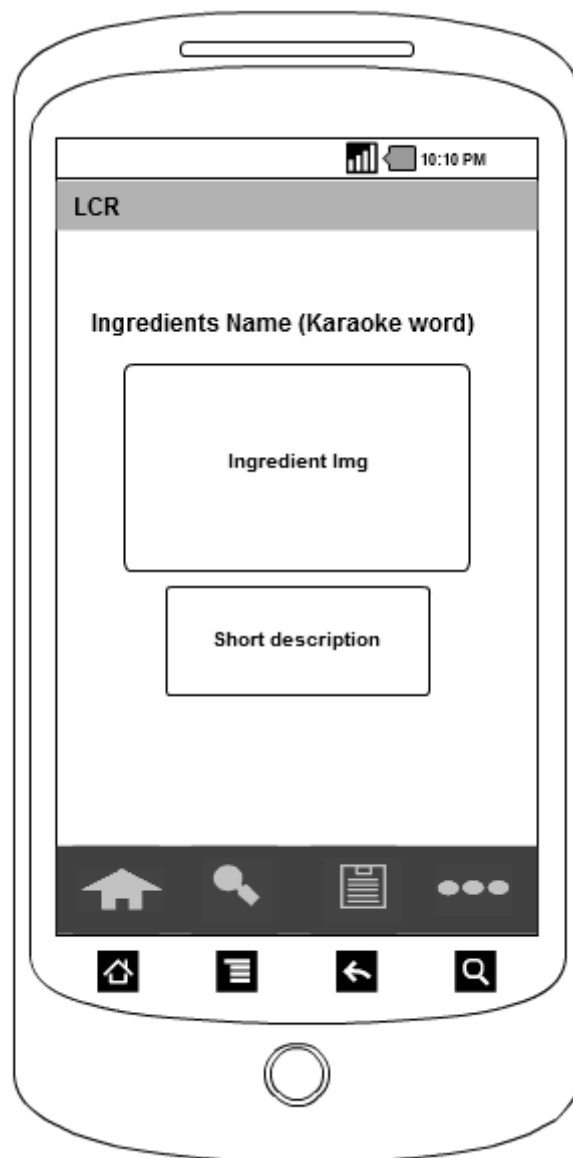
Figure 61 UI-15 Ingredient Detail

This UI will show after the customer select some ingredient from ingredient list in Ingredient list page. It will show name of ingredient, ingredient image and short description.

UI-16 Finds Ingredient Google Map
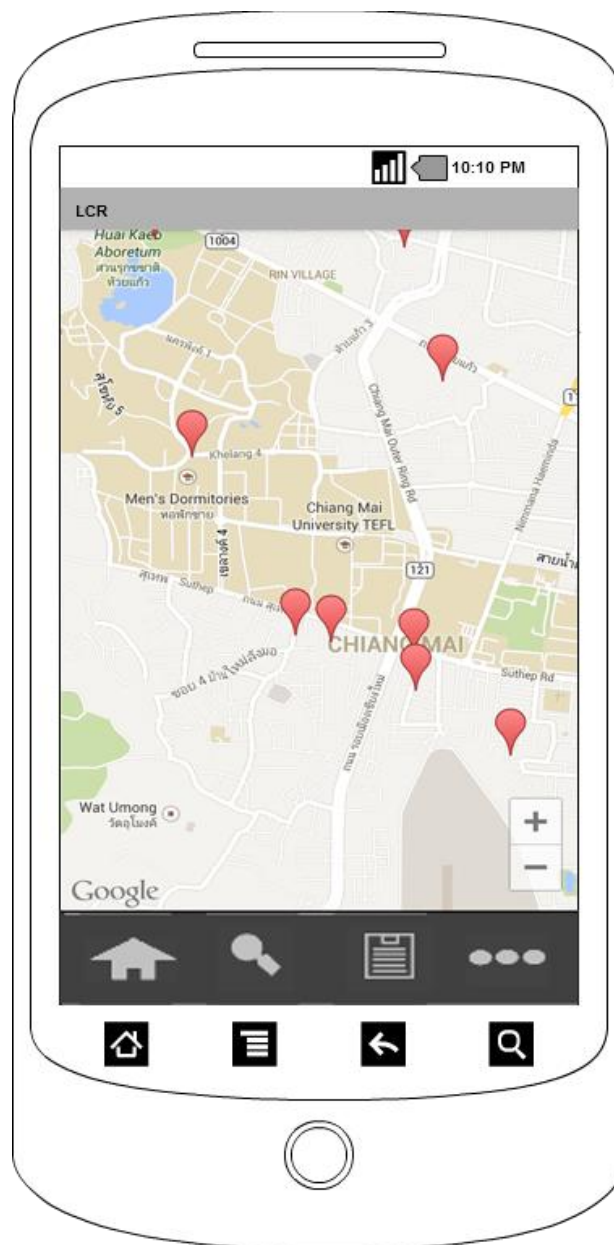
Figure 62 UI-16 Find Ingredient Google Map

This UI will show after the customer select Searching Location button. It will show interface of Google Map. The customer can search ingredient store location. create their own store location via this page.

UI-17 Ingredient Shop Description
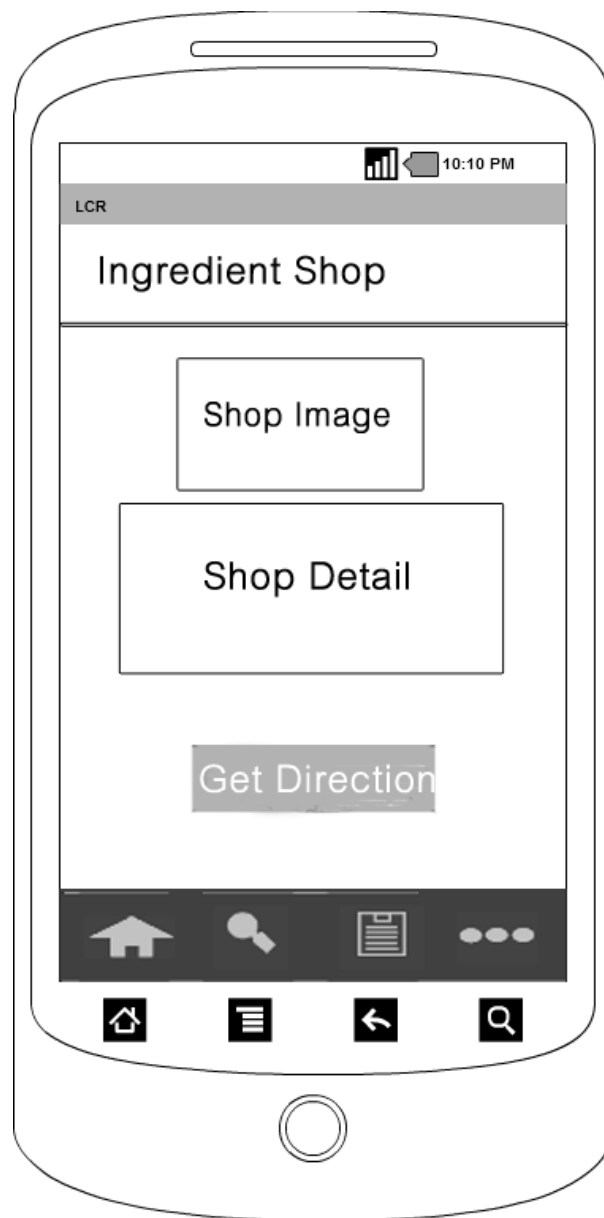
Figure 63 UI-17 Ingredient Shop Description

This UI will show after the customer select any shop marker in google map. It will show the shop name, image, short descrtiption, and the customer can get direction to the shop by select get direction button.
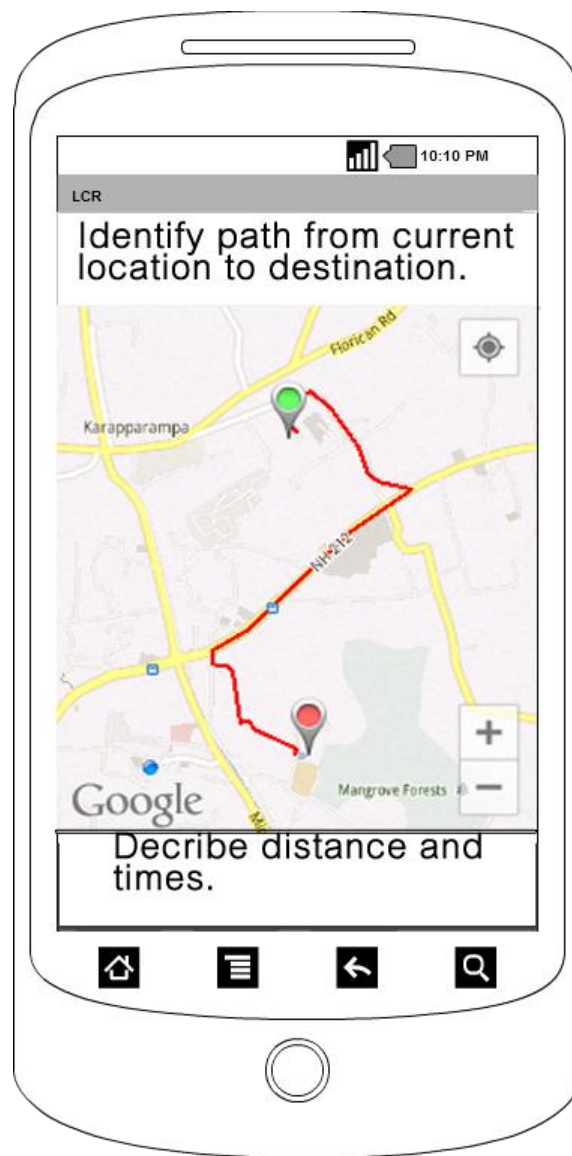
UI-18 Shop path

Figure 64 UI-18 Shop Path

This UI will show after the customer selects get direction button. It will show interface of Google Map that show the path between shop and current location of customer, and it can describe the times and distance of the shop destination.

UI-19 Random Food Function

Figure 65 UI-19 Random Food Function

This UI will show after the customer select random function. The customer can selects random button for random menu.

UI-20 Random Food Shaking
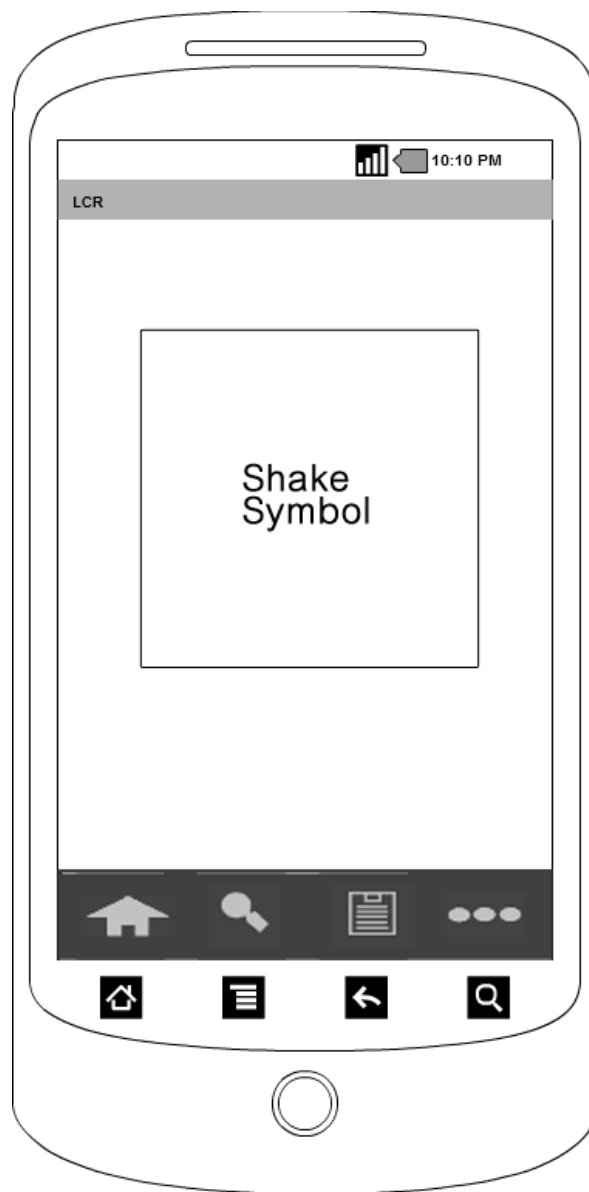
Figure 66 UI-20 Random Food Shaking

This UI will show after the customer selects random button. It will show the symbol for provides customer can shakes their phone for receive a random menu.

UI-21 Random Food Result
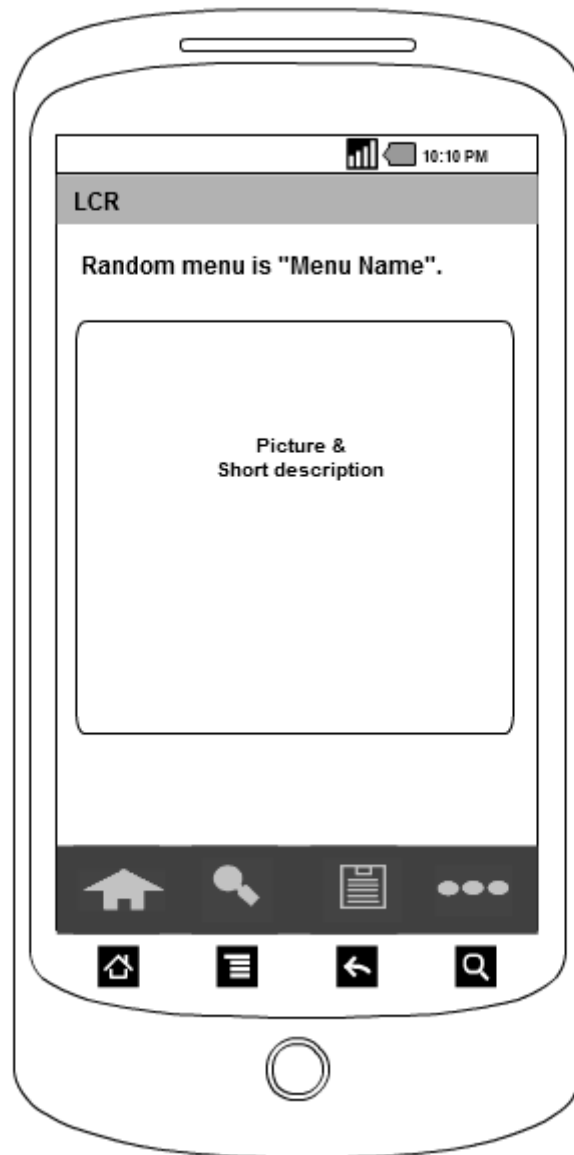
Figure 67 UI-21 Random Food Result

This UI will show after the customer shake their phone. It will show the food menu that the customer received.

UI-22 Shopping List

Figure 68 UI-22 Shopping List

This UI will show after the customer open Shopping List menu from menu bar. It will show list of menu that customer added via recipe detail page.

UI-23 Shopping List detail
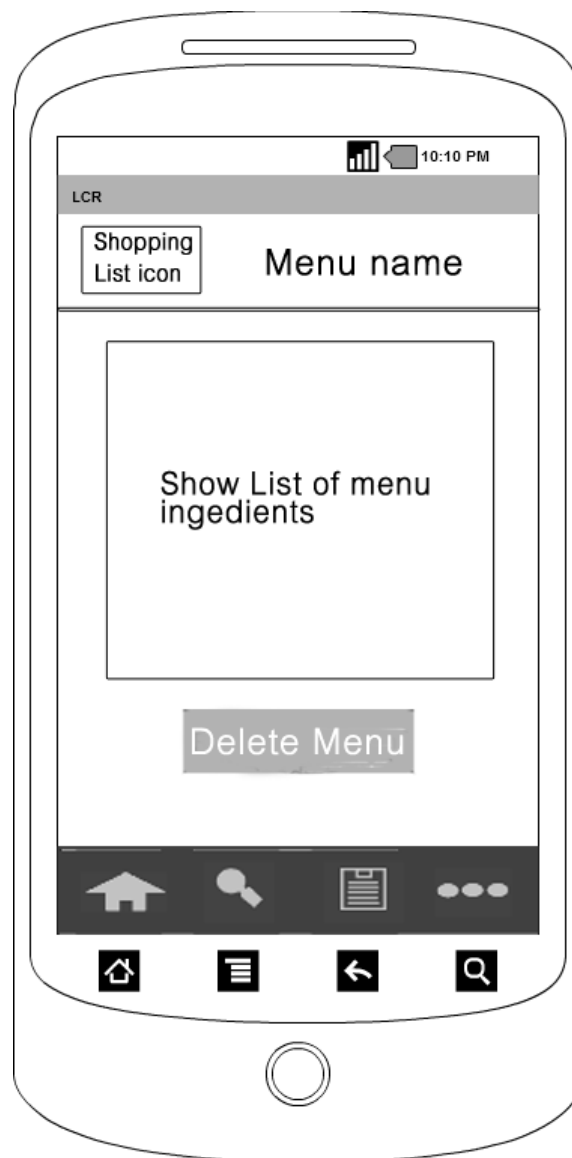
Figure 69 UI-23 Shopping List Summary

This UI will show after the customer select any menu from shopping list page. It will show all of ingredients that have in selected menu and the customer can delete their menu by selects delete menu button.

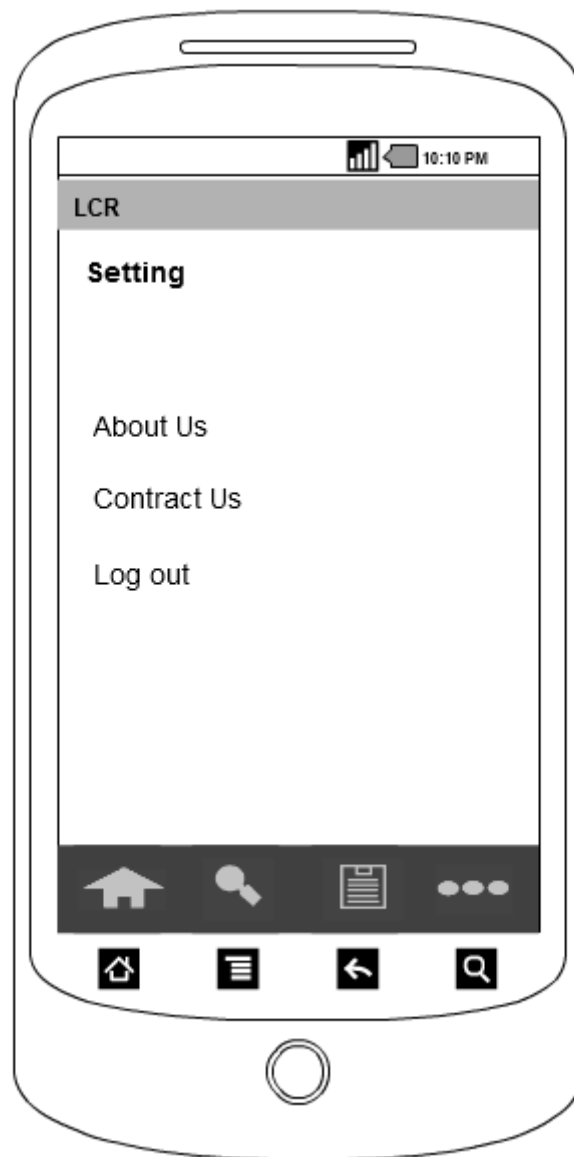**UI-24 Setting function**

Figure 70 UI-24 Setting Function

This UI will show after the customer select setting function from menu bar. It will show a list of links are About Us, Contract US, and Log out.