COMP211H: Introduction to Software Engineering

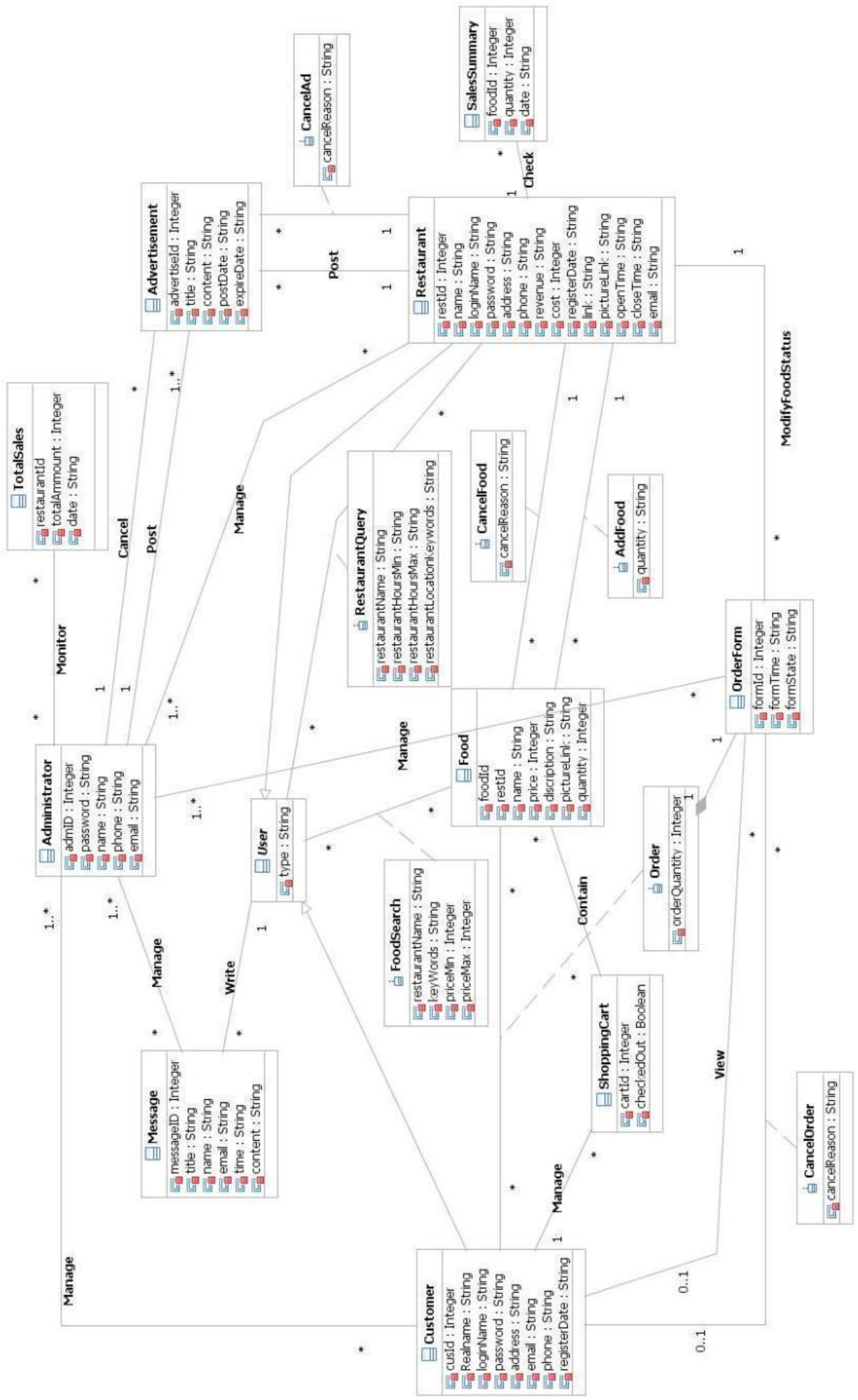
DOCUMENTATION FOR ACTIVITY 3 & 4

Group 005 Lumen

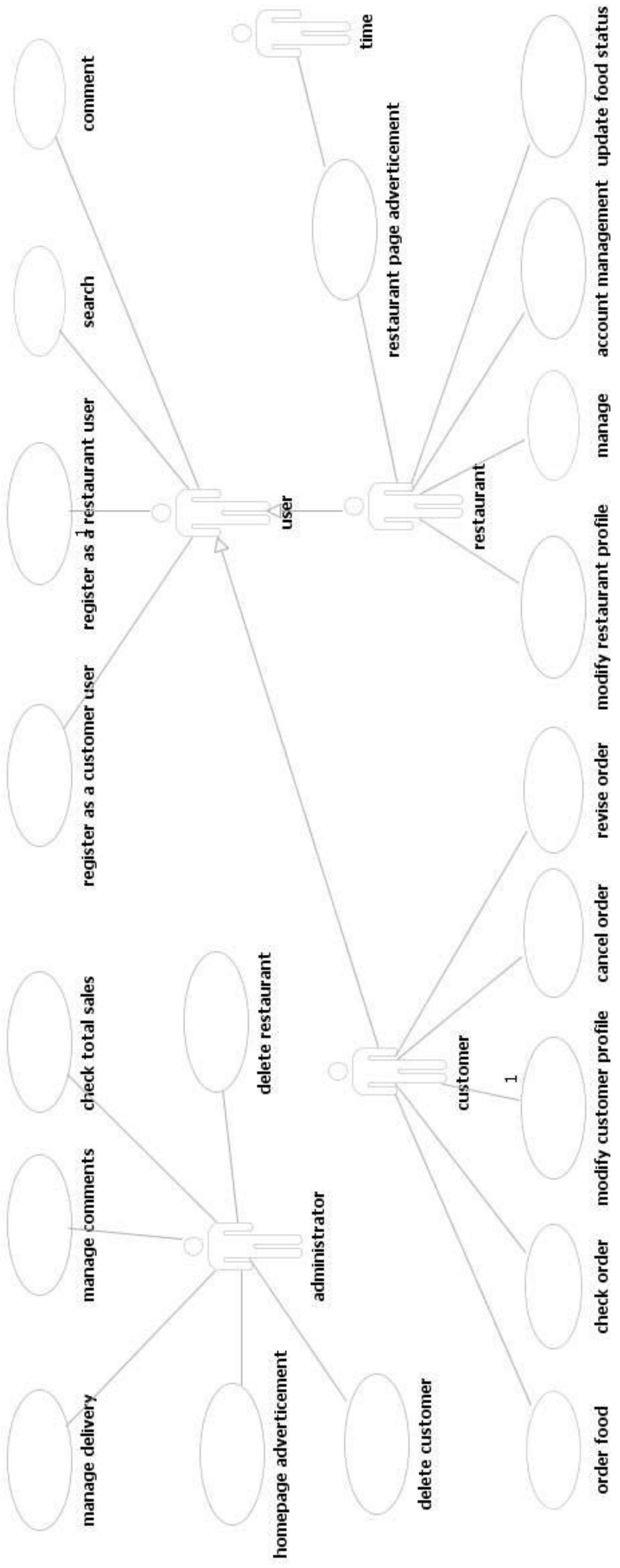
Group Members

Dong Jingming	08523163
Jiang Jiawei	08523101
Lu Jingwan	05735894
Zhao Siqi	06630716
Zhu Shucheng	05729558

PART I DOMAIN MODEL



PART II USE CASE MODEL



PART III SYSTEM ANALYSIS AND DESIGN SPECIFICATION

INTRODUCTION

Our group analyzed the project requirements and implemented the Lumen’s Restaurant which is the food ordering system. The 5 most important use cases of Lumen’s Restaurant are as following,

- Registration
- Search
- Order Food
- Sales Management
- Manage Food

Lumen’s Restaurant is supposed to be designed as an ordering system and provide convenience for users. Customers are able to browse the food of each restaurant. But if the customer wants to order through the system or the restaurant wants to run its business through the Lumen’s Restaurant system, they must do the **registration.** If the user has particular preference and wants to search for a particular item, customer can **search** for it. The core of the system is for customers to **order food**. In addition, restaurant can **manage food** through the system to provide better service. Restaurant and administrator are able to access the **sales management** to obtain an overview of the sales situation.

Registration

Registration provides the function of signing up as a customer or restaurant. Registered customer is able to order food and restaurant is able to run business through Lumen’s Restaurant system only after registration.

Search

If the customer wants to find a particular item, the system can help the customer to find the item among all the restaurants or even in a particular restaurant according to the customer’s requests.

Order Food

This is the core of Lumen’s Restaurant system. Registered customers are able to order food in this process. This process ends when the user checks out and confirms the order. After the order is placed, restaurant is able to view the items.

Sales Management

Restaurant and administrator can check the sales condition in order to help them to provide better service to customers. Delivery status is also a part of sales management.

Manage Food

Restaurant is able to add new food or modify the existing food. Customer can order the food after restaurant add it.

ANALYSIS MODEL • USE-CASE REALIZATION -- ANALYSIS

REGISTRATION

The boundary, control and entity classes are identified and list as below:

Boundary Class

RegistrationUI

This class communicates with **RegistrationMgr** and provides interface for the actors. After it retrieves the input of user, it would pass registration information to the **RegistrationMgr** to process the registration.

Control Class

RegistrationMgr

This class would obtain the registration information passed by the **RegistrationUI**, deal with the possible exceptions, and call the **RegisterCustomer** and **RegisterRestaurant** classes to store the account information.

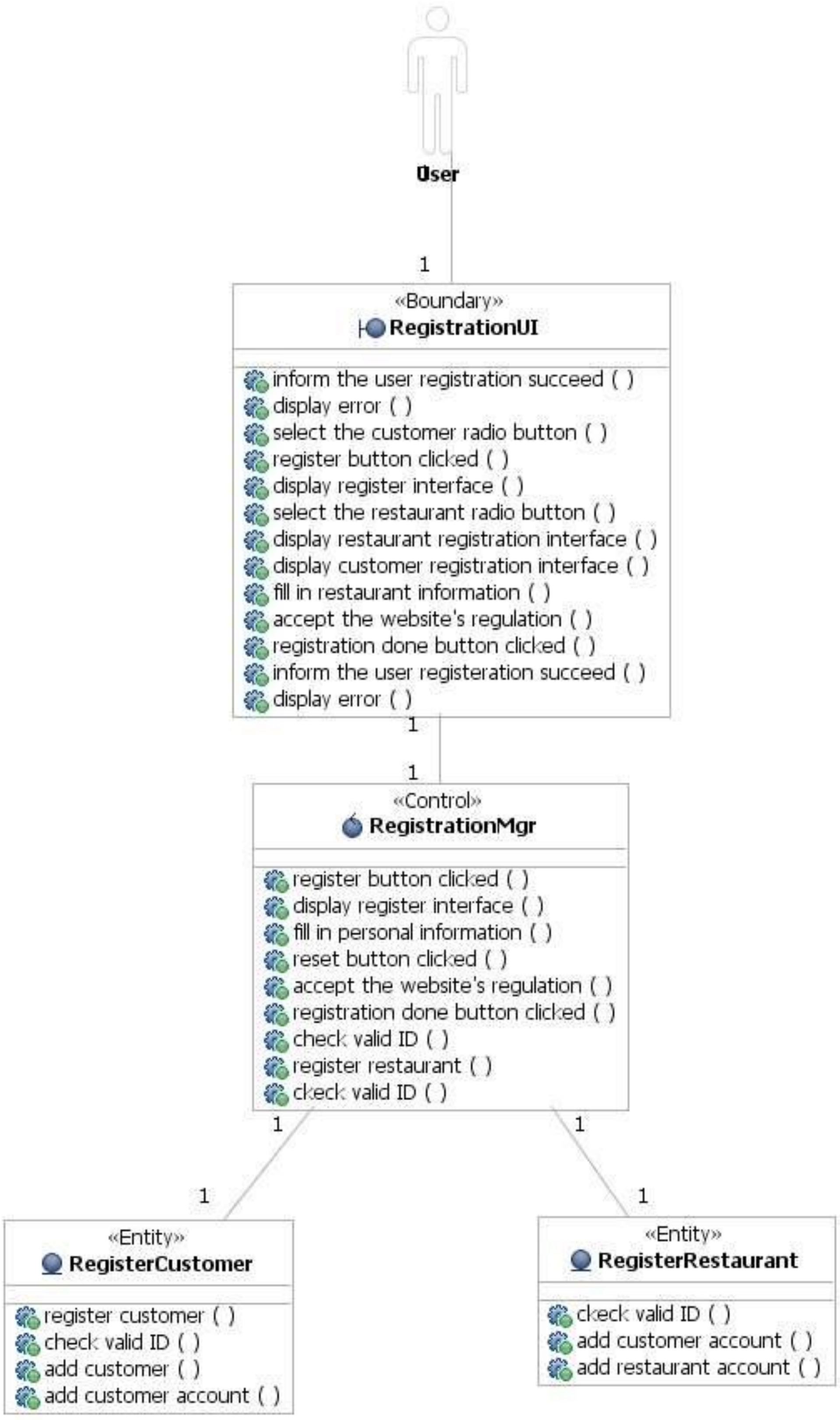
Entity Class

RegisterCustomer

After obtaining request from the **RegistrationMgr**, this class would add to the database a new record of customer information.

RegisterRestaurant

After obtaining request from the **RegistrationMgr**, this class would add to the database a new record of restaurant information.

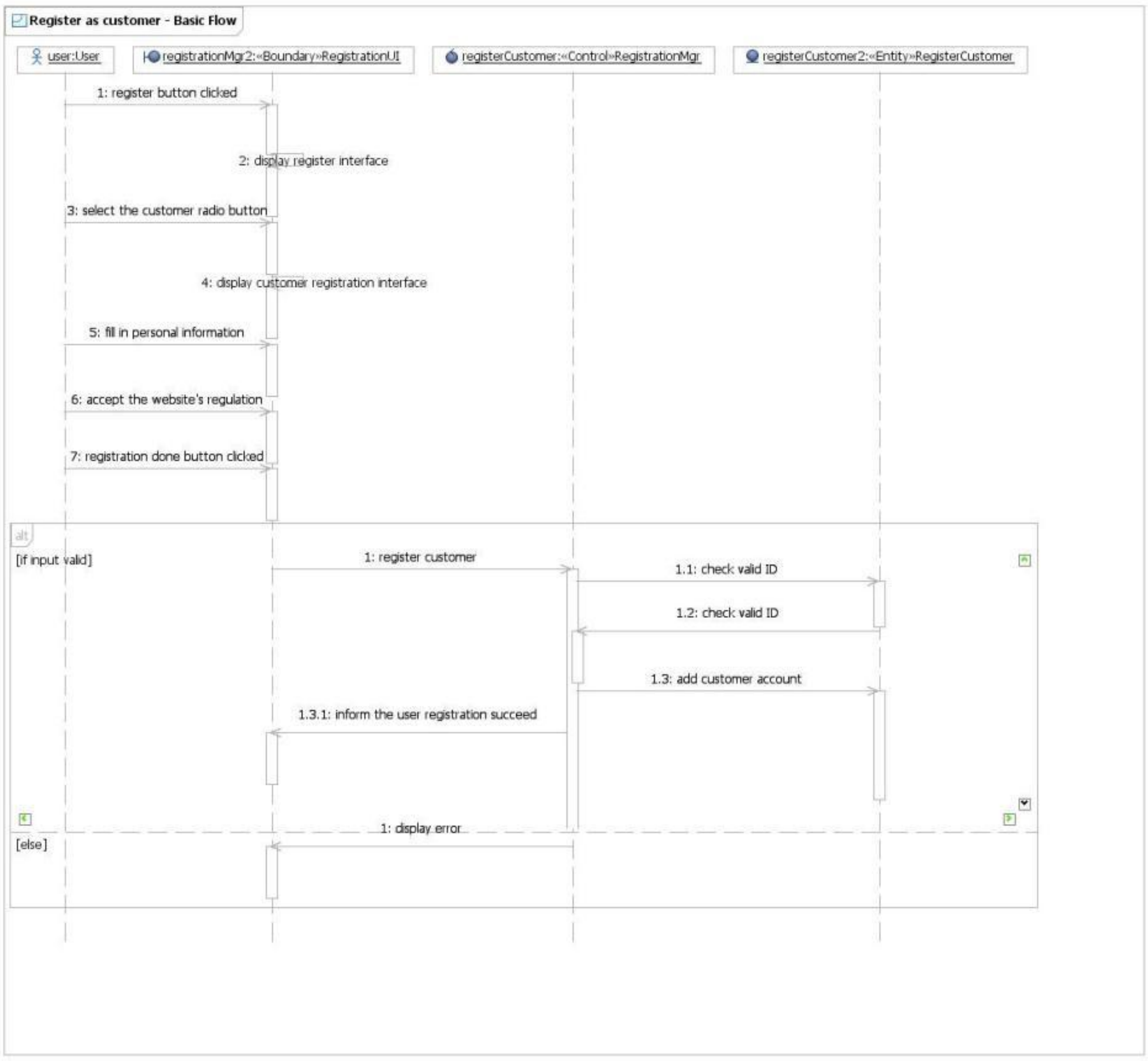

User
1
«Boundary»
RegistrationUI
inform the user registration succeed ( )
display error ( )
select the customer radio button ( )
register button clicked ( )
display register interface ( )
select the restaurant radio button ( )
display restaurant registration interface ( )
display customer registration interface ( )
fill in restaurant information ( )
accept the website's regulation ( )
registration done button clicked ( )
inform the user registeration succeed ( )
display error ( )
1
1
«Control»
RegistrationMgr
register button clicked ( )
display register interface ( )
fill in personal information ( )
reset button clicked ( )
accept the website's regulation ( )
registration done button clicked ( )
check valid ID ( )
register restaurant ( )
ckeck valid ID ( )
1
1
1
1
«Entity»
RegisterCustomer
register customer ( )
check valid ID ( )
add customer ( )
add customer account ( )
«Entity»
RegisterRestaurant
ckeck valid ID ( )
add customer account ( )
add restaurant account ( )

Register as a customer user

Flow of events

1. The use case starts when a **user** clicks on the register button or a **user** clicks on the 'log in' button with invalid username or password.

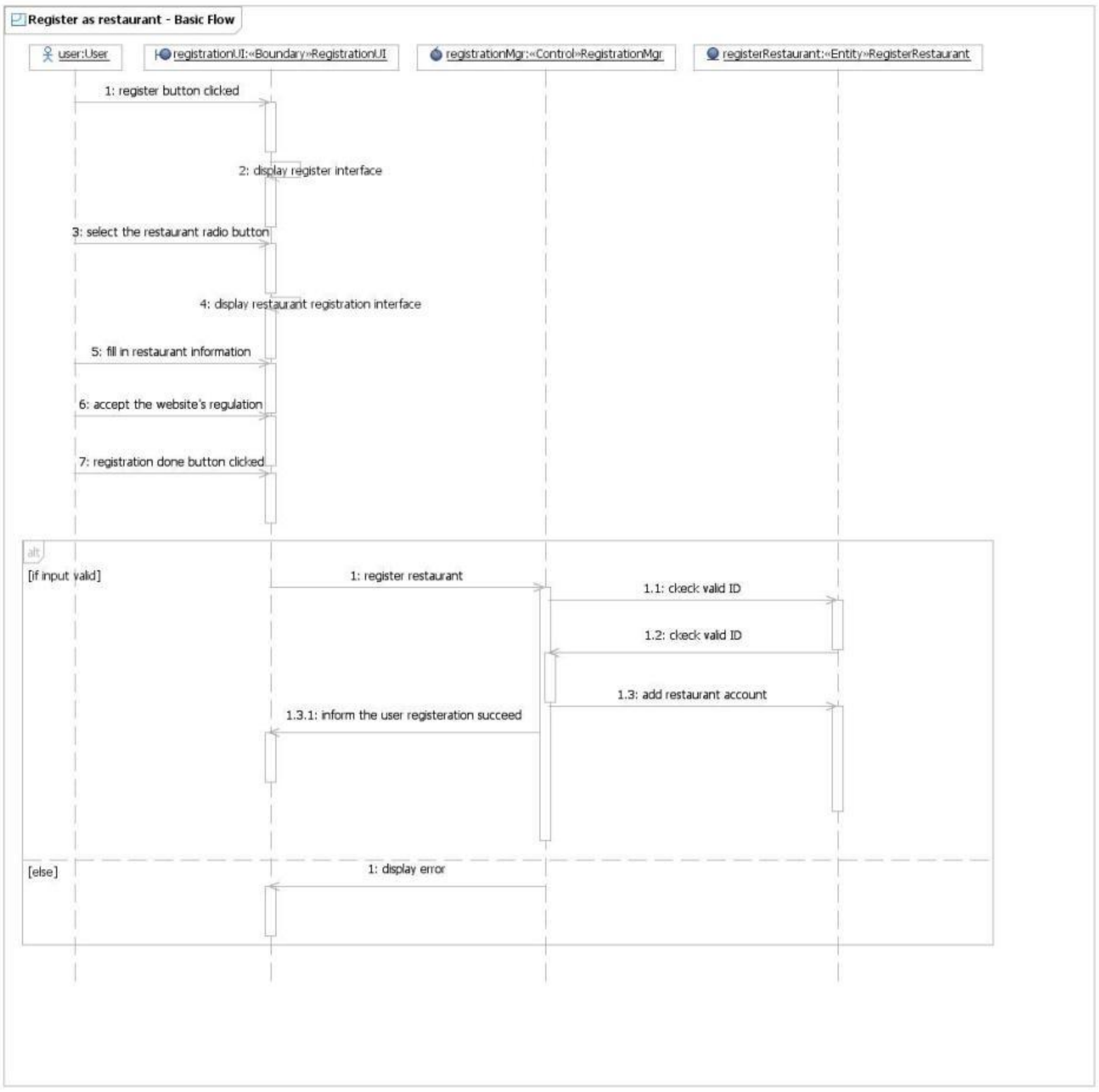
2. The **RegistrationUI** displays the interface for entering user information.
   2.1. The **user** selects the radio button ‘Customer’ on the up-left corner.
   2.2. The **user** enters the username (login name).
   2.3. The **user** enters the nickname.
   2.4. The **user** enters the password.
   2.5. The **user** confirms the password.
   2.6. The **user** enters the address.
   2.7. The **user** enters the phone number.
   2.8. The **user** enters the email.
   2.9. The **user** ticks on the ‘accept the regulation’.
3. If the ‘Reset’ button is clicked
   3.1. The **RegistrationUI** refreshes the web page and load the original one.
4. The user clicks on the ‘register’ button.
5. If any type of the input information is invalid
   5.1. The **RegistrationUI** informs the user that certain type of information is invalid
   5.2. The **RegistrationUI** refreshes the web page and load the original one.
6. The **RegistrationMgr** saves the account into the database.
7. The use case ends.


Register as customer - Basic Flow
user:User
registrationMgr2:«Boundary»RegistrationUI
registerCustomer:«Control»RegistrationMgr
registerCustomer2:«Entity»RegisterCustomer
1: register button clicked
2: display register interface
3: select the customer radio button
4: display customer registration interface
5: fill in personal information
6: accept the website's regulation
7: registration done button clicked
alt
[if input valid]
1: register customer
1.1: check valid ID
1.2: check valid ID
1.3: add customer account
1.3.1: inform the user registration succeed
[else]
1: display error

Register as a restaurant user

Flow of events

1. The use case starts when a **user** clicks on the register button or a **user** clicks on the 'log in' button with invalid username or password.
2. The **RegistrationUI** displays the interface for entering client information.
    2.1. The **user** selects the radio button 'Restaurant' on the up-right corner.
    2.2. The **user** enters the user name (login name).
    2.3. The **user** enters the restaurant name.
    2.4. The **user** enters the password.
    2.5. The **user** confirms the password.
    2.6. The **user** enters the address.
    2.7. The **user** enters the phone number.
    2.8. The **user** enters the email.
    2.9. The **user** ticks on the 'accept the regulation'.
3. If the 'Reset' button is clicked
    3.1 The **RegistrationUI** refreshes the web page and load the original one.
4. The user clicks on the 'register' button.
5. If any type of the input information is invalid
    5.1. The **RegistrationUI** informs the user that certain type of information is invalid
    5.2. The **RegistrationUI** refreshes the web page and load the original one.
6. The **RegistrationMgr** saves the account into the database.
7. The use case ends.


Register as restaurant - Basic Flow
user:User
registrationUI:«Boundary»RegistrationUI
registrationMgr:«Control»RegistrationMgr
registerRestaurant:«Entity»RegisterRestaurant
1: register button clicked
2: display register interface
3: select the restaurant radio button
4: display restaurant registration interface
5: fill in restaurant information
6: accept the website's regulation
7: registration done button clicked
alt
[if input valid]
1: register restaurant
1.1: ckeck valid ID
1.2: ckeck valid ID
1.3: add restaurant account
1.3.1: inform the user registeration succeed
[else]
1: display error

SEARCH

The boundary, control and entity classes are identified and list as below:

Boundary Class

SearchFoodUI

This class communicates with **SearchFoodMgr** and provides interface for the actors. After it retrieves the input of user, it would pass search information to the **SearchFoodMgr** to process the search.

Control Class

SearchFoodMgr

This class would obtain the search information passed by the **SearchFoodUI**, deal with the possible exceptions, and call the **SearchFood** and **SearchRestaurant** classes to complete a search.

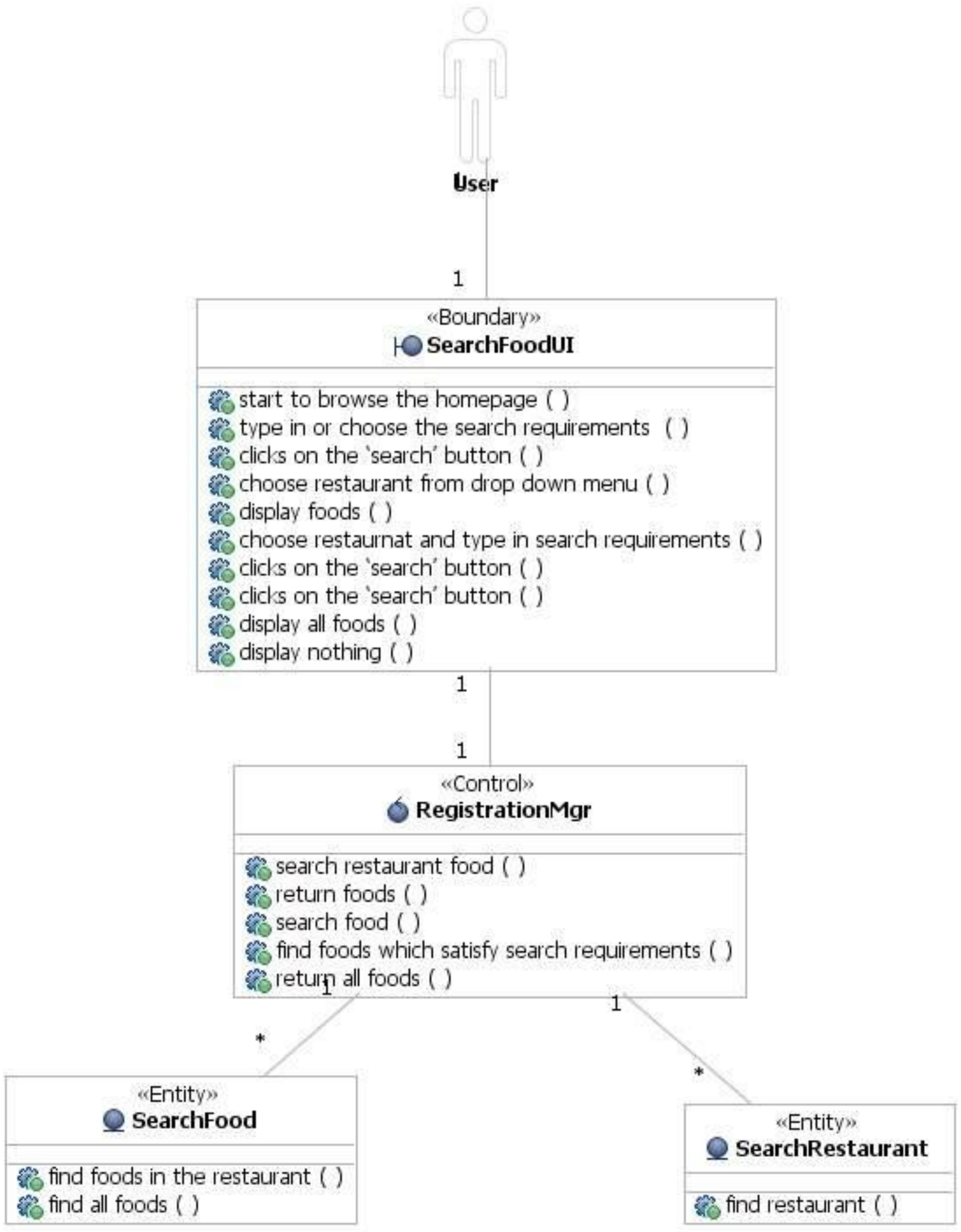
Entity Classes

SearchFood

After obtaining request from the **SearchFoodMgr**, this class would provide the details of the target food.

SearchRestaurant

After obtaining request from the **SearchFoodMgr**, this class would provide the details of the target restaurant.


User
1
«Boundary»
SearchFoodUI
start to browse the homepage ( )
type in or choose the search requirements ( )
clicks on the 'search' button ( )
choose restaurant from drop down menu ( )
display foods ( )
choose restaurnat and type in search requirements ( )
clicks on the 'search' button ( )
clicks on the 'search' button ( )
display all foods ( )
display nothing ( )
1
1
«Control»
RegistrationMgr
search restaurant food ( )
return foods ( )
search food ( )
find foods which satisfy search requirements ( )
return all foods ( )
1
*
1
*
«Entity»
SearchFood
find foods in the restaurant ( )
find all foods ( )
«Entity»
SearchRestaurant
find restaurant ( )

SEARCH FOOD

Flow of Events

1. The use case can start when the **user** logs into the system initially, or by browsing the homepage, or by entering the restaurant page.

2. The **user** can type in or choose the search requirements in three fields provided by the **SearchFoodUI**.

    2.1 The **user** can type in the price in the 'search for price' box.

    2.2 The **user** can type in the keywords in the 'search for keywords' box.

3. The user clicks on the 'search' button.

4. If no input is entered,

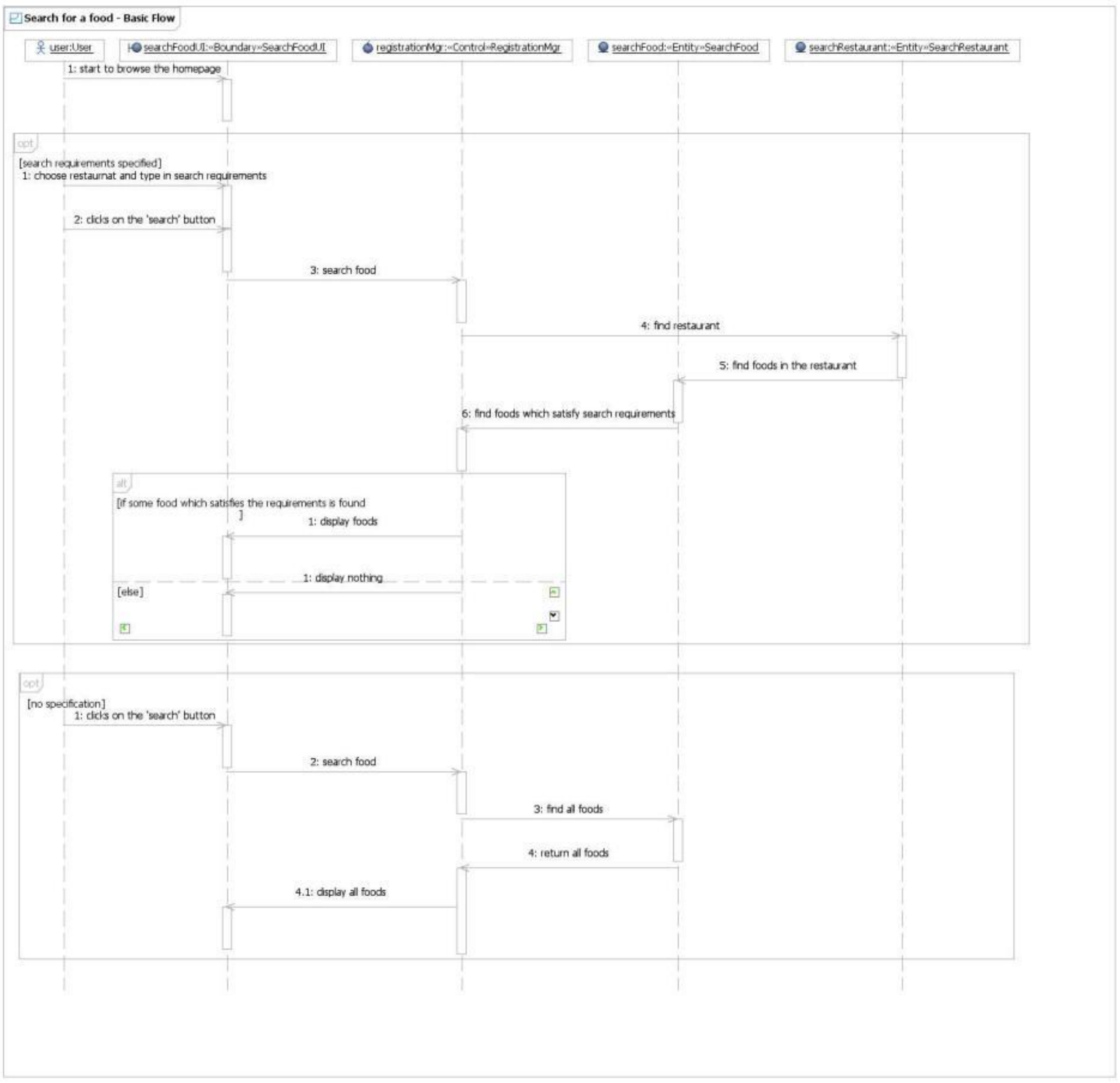
    4.1 The **SearchFoodUI** will list all the food available.

5. If the input requirement does not match any available

    5.1 The **SearchFoodUI** will display nothing in the search result table.

6. The **SearchFoodMgr will** retrieve the food details, then the **SearchFoodUI** displays the search results that meet the requirements.

7. The use case ends.


Search for a food - Basic Flow
user:User
searchFoodUI:«Boundary»SearchFoodUI
registrationMgr:«Control»RegistrationMgr
searchFood:«Entity»SearchFood
searchRestaurant:«Entity»SearchRestaurant
1: start to browse the homepage
opt
[search requirements specified]
1: choose restaurnat and type in search requirements
2: clicks on the 'search' button
3: search food
4: find restaurant
5: find foods in the restaurant
6: find foods which satisfy search requirements
alt
[if some food which satisfies the requirements is found]
1: display foods
1: display nothing
[else]
opt
[no specification]
1: clicks on the 'search' button
2: search food
3: find all foods
4: return all foods
4.1: display all foods

SEARCH RESTAURANT

Flow of Events

1. The use case can start when the **user** browses the homepage.
2. The **user** can choose the restaurant provided by the **SearchFoodUI**.
    2.1 The **user** can choose a restaurant from the pull down menu.
3. The **user** clicks on the 'enter' button.
4. If no input is entered,
    4.1 The **SearchFoodMgr** will direct to the default page.
6. The **SearchFoodMgr will** retrieve the restaurant information, then the **SearchFoodUI** displays the restaurant page and its food details.
7. The use case ends.

ORDER FOOD

The boundary, control and entity classes are identified and list as below:

Boundary Class

OrderFoodUI

This class communicates with **OrderFoodMgr** and provides interface for the actors. After the customer finishes his order, it would display the order page in which the customer can modify the receiver information and make changes to the order.

CheckOrderUI

This class communicates with **OrderFoodMgr** and provides interface for the actors. No changes can be made at this stage.

Control Class

OrderFoodMgr

This class would obtain the order information, deal with the possible exceptions(e.g. nothing contains in the order before the customer checks out), and call the **OrderCustomer, OrderFood**, **OrderRestaurant**, **OrderShoppingCart**, **OrderForm** classes to complete the process.

Entity Classes

OrderCustomer

After obtaining request from the **OrderFoodMgr**, this class would provide the default information of the registered customer.

OrderFood

This class would provide the information of the food to the **OrderFoodMgr.**

OrderRestaurant

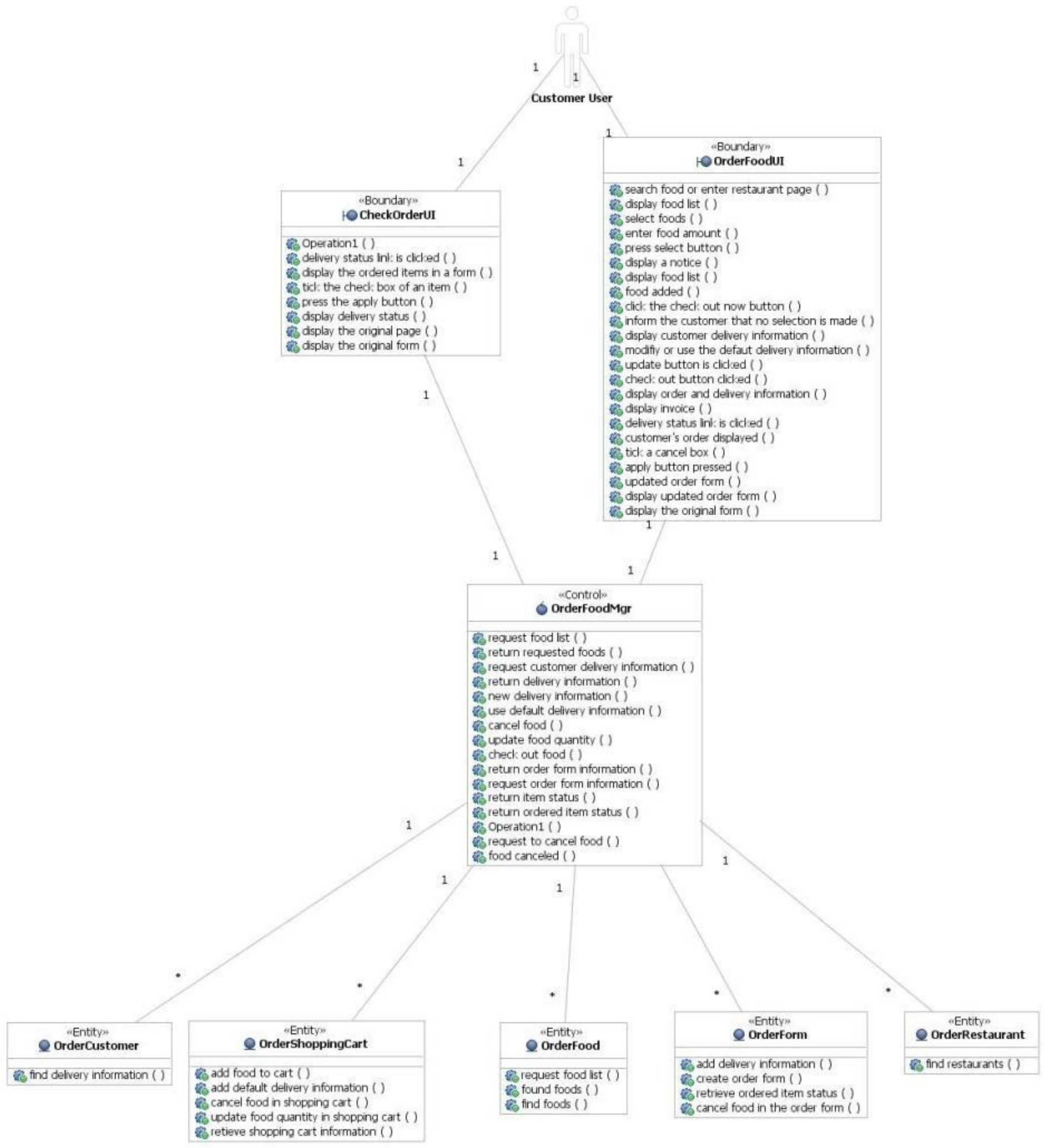
This class would provide the information of the restaurant to the **OrderFoodMgr.**

OrderShoppingCart

This class would record the order information, food ordered, and quantity of the food.

OrderForm

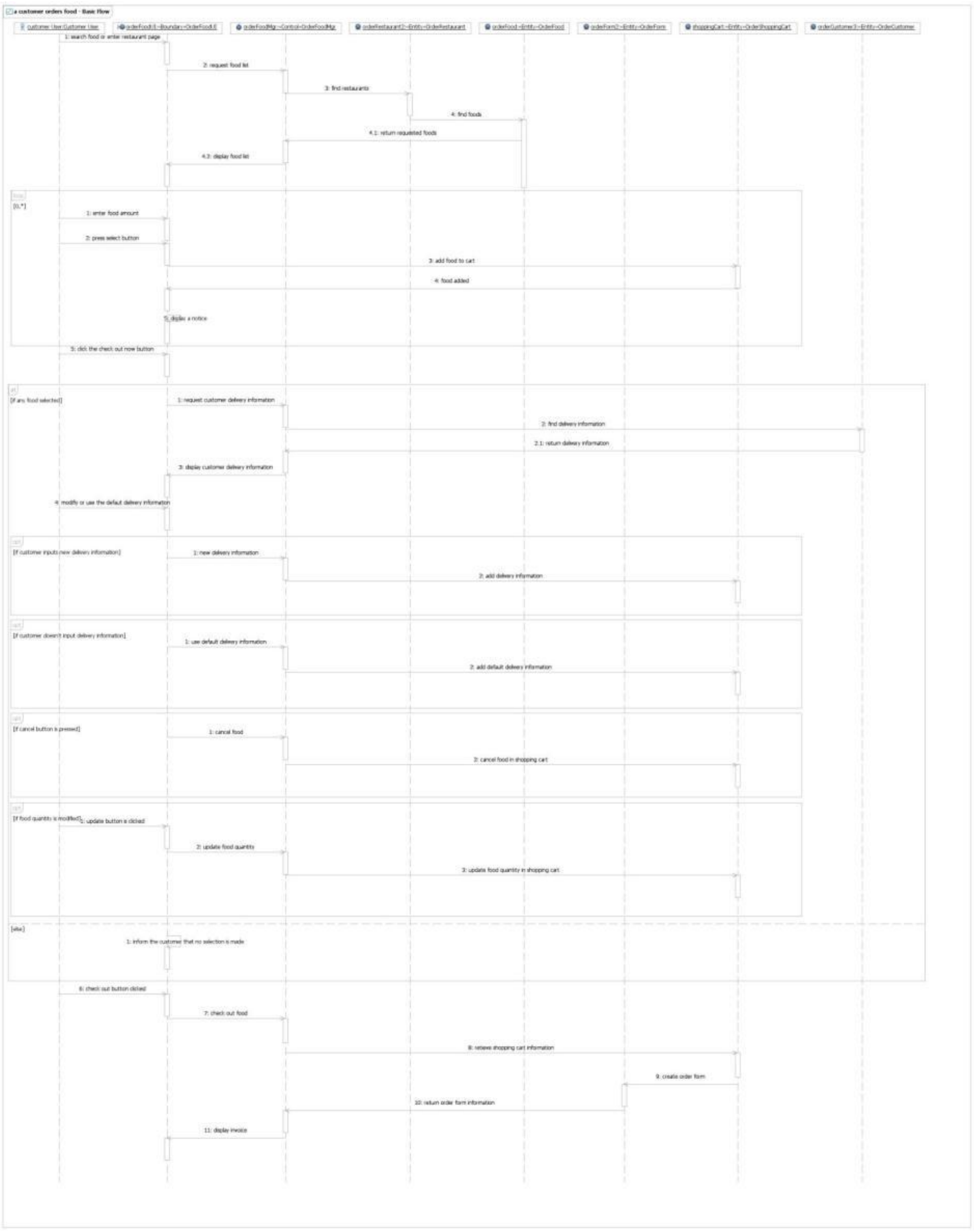
After obtaining request from the **OrderFoodMgr**, this class would provide the details of the order form displayed in the check out page.


Customer User
1
1
1
1
«Boundary»
CheckOrderUI
Operation1 ( )
delivery status link is clicked ( )
display the ordered items in a form ( )
tick the check box of an item ( )
press the apply button ( )
display delivery status ( )
display the original page ( )
display the original form ( )
«Boundary»
OrderFoodUI
search food or enter restaurant page ( )
display food list ( )
select foods ( )
enter food amount ( )
press select button ( )
display a notice ( )
display food list ( )
food added ( )
click the check out now button ( )
inform the customer that no selection is made ( )
display customer delivery information ( )
modify or use the defaut delivery information ( )
update button is clicked ( )
check out button clicked ( )
display order and delivery information ( )
display invoice ( )
delivery status link is clicked ( )
customer's order displayed ( )
tick a cancel box ( )
apply button pressed ( )
updated order form ( )
display updated order form ( )
display the original form ( )
1
1
1
1
«Control»
OrderFoodMgr
request food list ( )
return requested foods ( )
request customer delivery information ( )
return delivery information ( )
new delivery information ( )
use default delivery information ( )
cancel food ( )
update food quantity ( )
check out food ( )
return order form information ( )
request order form information ( )
return item status ( )
return ordered item status ( )
Operation1 ( )
request to cancel food ( )
food canceled ( )
1
1
1
1
*
*
*
*
*
«Entity»
OrderCustomer
find delivery information ( )
«Entity»
OrderShoppingCart
add food to cart ( )
add default delivery information ( )
cancel food in shopping cart ( )
update food quantity in shopping cart ( )
retieve shopping cart information ( )
«Entity»
OrderFood
request food list ( )
found foods ( )
find foods ( )
«Entity»
OrderForm
add delivery information ( )
create order form ( )
retrieve ordered item status ( )
cancel food in the order form ( )
«Entity»
OrderRestaurant
find restaurants ( )

ORDER FOOD

Flow of events

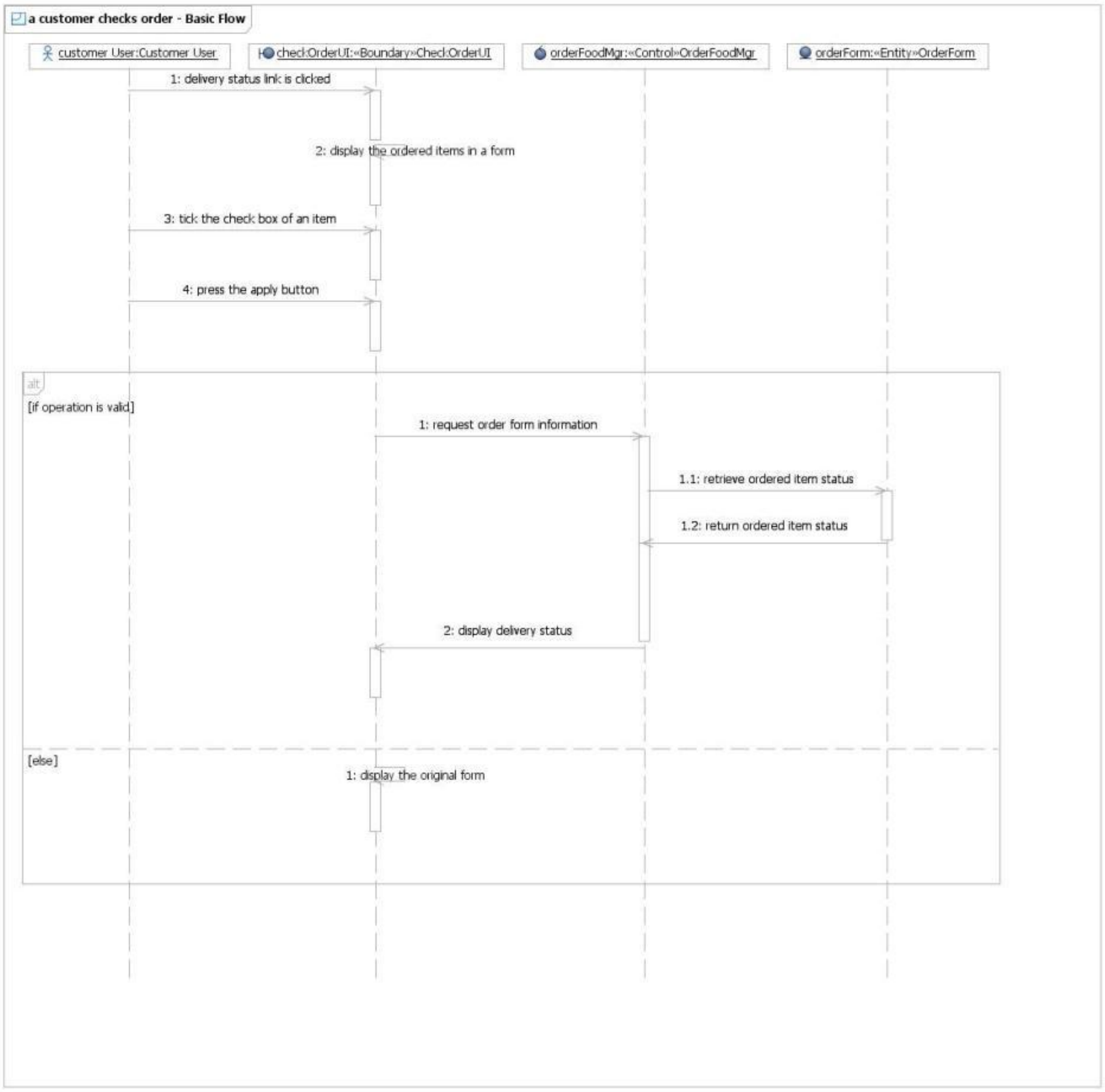
1. The use case starts when a **customer** clicks on the 'search' button or 'enter' button on the index page.
2. The **OrderFoodUI** displays the interface of food list of the search result or of the certain restaurant.
3. While the **customer** makes a selection of food
   3.1. The **customer** enters the amount of food he or she want for each food at the end of a food row.
   3.2. The **customer** presses 'select' button to confirm the amount of food selected.
   3.3. The **OrderFoodUI** displays the notice that the amount of food is selected and will be added to the shopping cart.
   3.4. By calling the **OrderShoppingCart** class through the **OrderFoodMgr,** the **OrderFoodUI** will display the updated shopping cart.
4. The **customer** press 'Check Out Now' button to activate the checkout procedure.
   4.1. If the **customer** doesn't select any food
      4.1.1. The **OrderFoodMgr** detects the exception, the **OrderFoodUI** informs the customer that no selection was made.
   4.2. If the **customer** has selection, the **OrderFoodUI** displays the delivery information including receiver name, phone number, delivery address to be confirmed or modified by customer.
5. The **customer** modifies the delivery information.
   5.1. If the **customer** inputs new delivery information
      5.1.1. The **OrderFoodMgr** will store the new input in the **OrderForm** class.
   5.2. If the **customer** doesn't input
      5.2.1. The **OrderFoodMgr** will use the default information of the customer via the **OrderCustomer** class.
   5.3. If the **customer** presses 'cancel' button on the order form
      5.3.1. The **OrderFoodMgr** will delete the corresponding food from the order and the database.
   5.4. If the **customer** modifies the quantity of the food and presses the update button
      5.4.1. The **OrderFoodUI** notifies the customer that the food quantity has been updated. The **OrderFoodMgr** will update the order form**.**
6. The **customer** presses check out button to confirm the order.
7. A new invoice page will be displayed by **OrderFoodUI** containing the order and the delivery information.
8. The use case ends.


a customer orders food - Basic Flow
customer User:Customer User
orderFoodUI:~Boundary~OrderFoodUI
orderFoodMgr:~Control~OrderFoodMgr
orderRestaurant2:~Entity~OrderRestaurant
orderFood:~Entity~OrderFood
orderForm2:~Entity~OrderForm
shoppingCart:~Entity~OrderShoppingCart
orderCustomer3:~Entity~OrderCustomer
1: search food or enter restaurant page
2: request food list
3: find restaurants
4: find foods
4.1: return requested foods
4.2: display food list
loop
[0..*]
1: enter food amount
2: press select button
3: add food to cart
4: food added
5: display a notice
5: click the check out now button
alt
[if any food selected]
1: request customer delivery information
2: find delivery information
2.1: return delivery information
3: display customer delivery information
4: modify or use the defaut delivery information
opt
[if customer inputs new delivery information]
1: new delivery information
2: add delivery information
opt
[if customer doesn't input delivery information]
1: use default delivery information
2: add default delivery information
opt
[if cancel button is pressed]
1: cancel food
2: cancel food in shopping cart
opt
[if food quantity is modified]
1: update button is clicked
2: update food quantity
3: update food quantity in shopping cart
[else]
1: inform the customer that no selection is made
6: check out button clicked
7: check out food
8: retrieve shopping cart information
9: create order form
10: return order form information
11: display invoice

CHECK ORDER

Flow of events

1. The use case starts when the **customer** presses customer link in the webpage. The use case can also be started once a **customer** finishes the check out procedure and presses 'Back to Account' button at the Invoice page.
2. The **OrderFoodUI** displays the customer account, with all orders in history listed in a form.
3. The **customer** ticked the Check box within an order and presses apply.
4. If invalid apply button (namely the apply button in another row) is pressed
    4.1. The **OrderFoodMgr** detects the exception and directs to the original check out page.
5. A detail of the order is listed by the **OrderFoodUI**, including receiver information, time and status of the order, food and prices of the order.
6. The use case ends.


a customer checks order - Basic Flow
customer User:Customer User
check:OrderUI:«Boundary»CheckOrderUI
orderFoodMgr:«Control»OrderFoodMgr
orderForm:«Entity»OrderForm
1: delivery status link is clicked
2: display the ordered items in a form
3: tick the check box of an item
4: press the apply button
alt
[if operation is valid]
1: request order form information
1.1: retrieve ordered item status
1.2: return ordered item status
2: display delivery status
[else]
1: display the original form

MANAGE FOOD

The boundary, control and entity classes are identified and listed as below:

Boundary Class

ManageFoodUI

This class communicates with **ManageFoodMgr** and provides the interface for the users. When it successfully retrieves the data from users, it would pass the food data to **ManageFoodMgr**.

Control Class

ManageFoodMgr

This class would obtain the food data from **ManageFoodUI** and verify the input data. It will handle the possible exceptions and pass the valid data to **ManageFood** and **ManageRestaurant**.

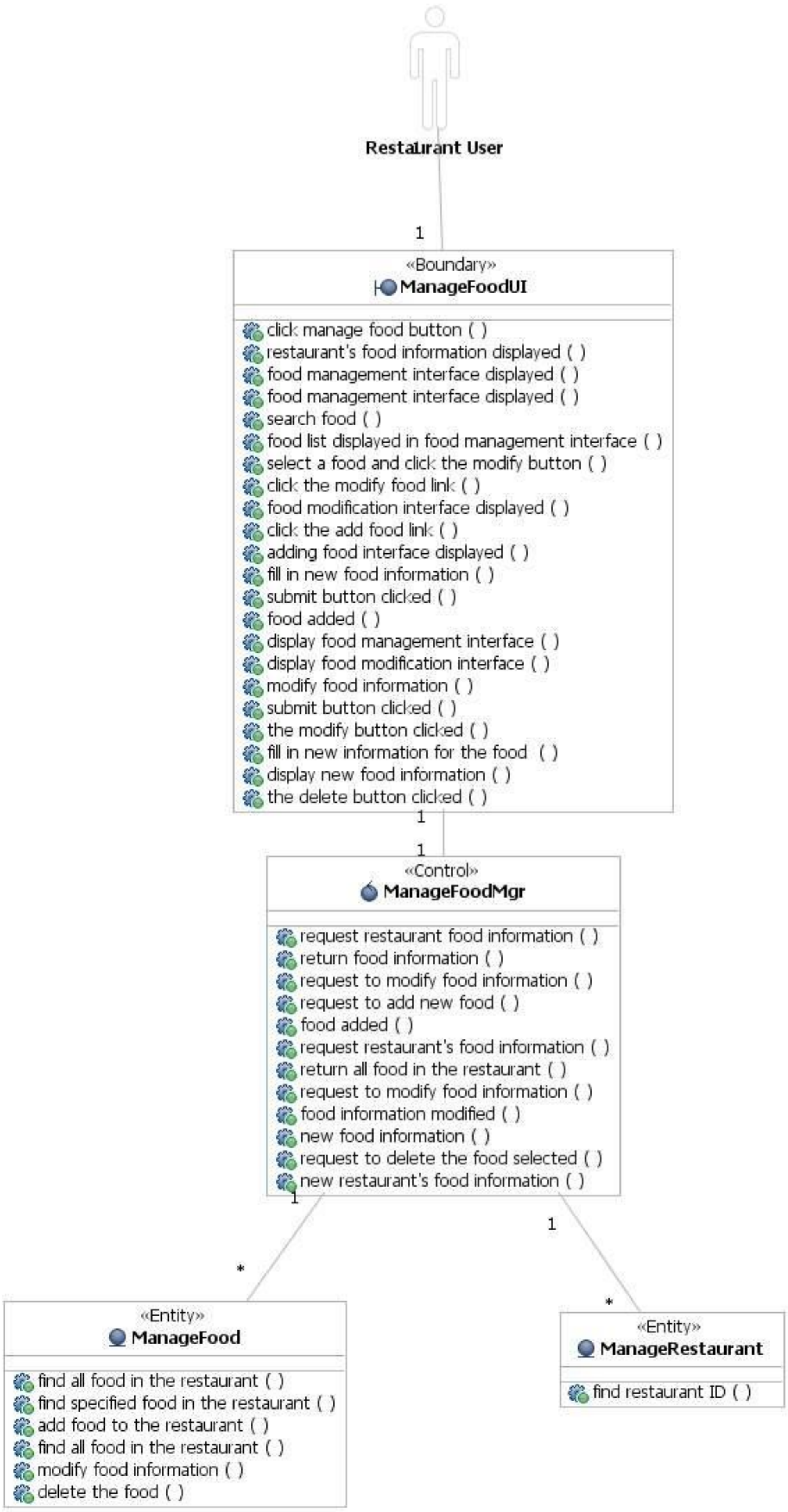
Entity Classes

ManageFood

When the valid data is retrieved and verified by the **ManageFoodUI** and **ManageFoodMgr**, it will add the data to the database.

ManageRestaurant

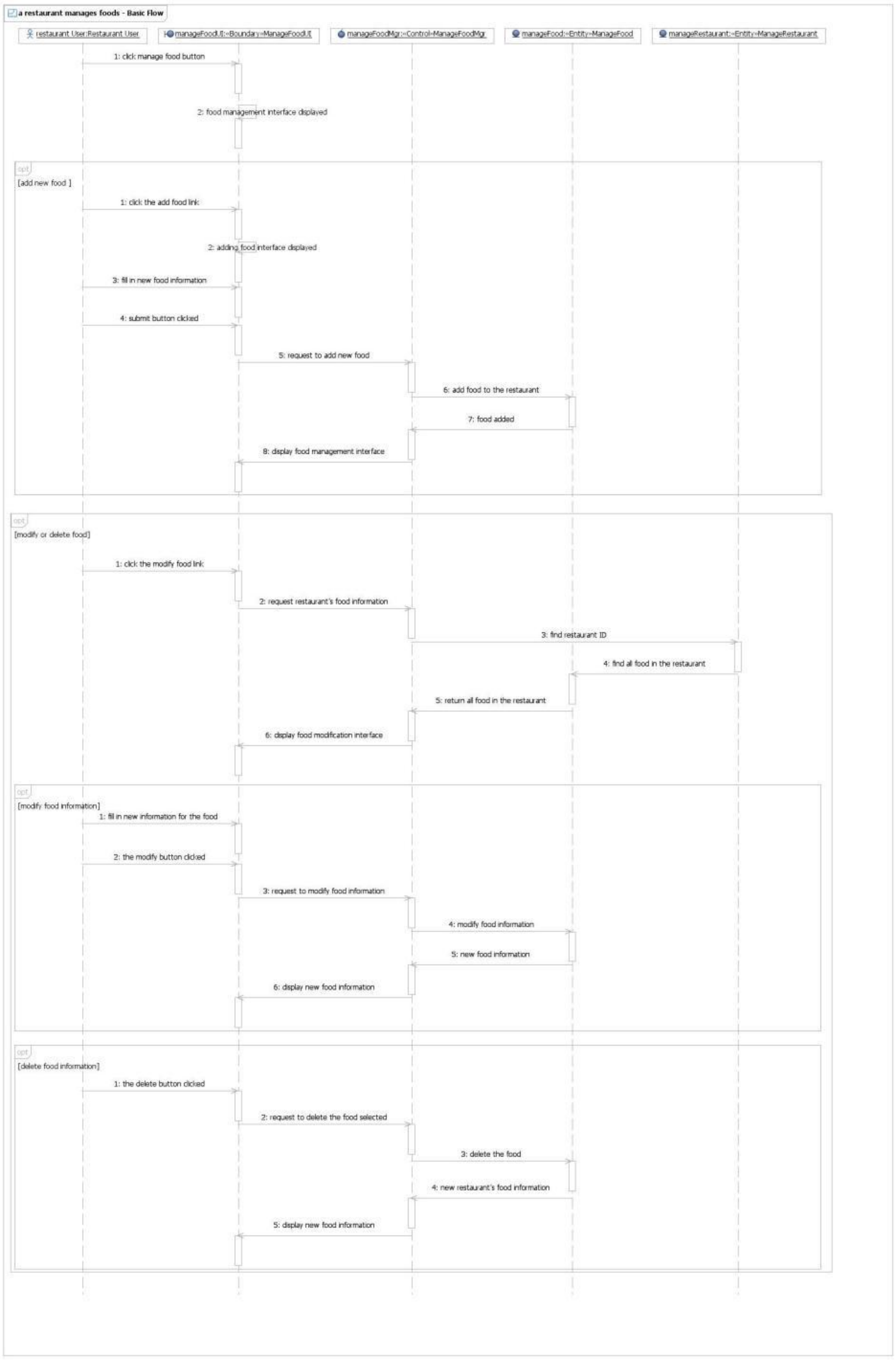
When the data is added, this class will build the connection between the restaurant and the new data in database.


Restaurant User
1
«Boundary»
ManageFoodUI
click manage food button ( )
restaurant's food information displayed ( )
food management interface displayed ( )
food management interface displayed ( )
search food ( )
food list displayed in food management interface ( )
select a food and click the modify button ( )
click the modify food link ( )
food modification interface displayed ( )
click the add food link ( )
adding food interface displayed ( )
fill in new food information ( )
submit button clicked ( )
food added ( )
display food management interface ( )
display food modification interface ( )
modify food information ( )
submit button clicked ( )
the modify button clicked ( )
fill in new information for the food ( )
display new food information ( )
the delete button clicked ( )
1
1
«Control»
ManageFoodMgr
request restaurant food information ( )
return food information ( )
request to modify food information ( )
request to add new food ( )
food added ( )
request restaurant's food information ( )
return all food in the restaurant ( )
request to modify food information ( )
food information modified ( )
new food information ( )
request to delete the food selected ( )
new restaurant's food information ( )
1
*
1
*
«Entity»
ManageFood
find all food in the restaurant ( )
find specified food in the restaurant ( )
add food to the restaurant ( )
find all food in the restaurant ( )
modify food information ( )
delete the food ( )
«Entity»
ManageRestaurant
find restaurant ID ( )

Manage Food

Flow of events

1. The use case starts when the registered **restaurant** clicks on the 'Manage Food' button after login.
2. The **ManageFoodUI** displays the interface for adding food, modifying food information and deleting food.
3. If the **restaurant** chooses to add food
    3.1. The **restaurant** enters the food name.
    3.2. The **restaurant** enters the food description.
    3.3. The **restaurant** enters the food price.
    3.4. The **restaurant** uploads the picture of the food.
    3.5. The **ManageFoodMgr** checks if the input is invalid or empty
        3.5.1. The **ManageFoodUI** will pop up a warning message indicating the problem.
        3.5.2. The flow of event resumes at {Enter food information}.
    3.6. If the 'Reset' button is clicked
        3.6.1. The flow of the event resumes at the beginning.
4. If the **restaurant** chooses to modify food information
    4.1. The **restaurant** selects the food and press 'modify' button.
    4.2. The **restaurant** then modified the information as preferred.
5. If the **restaurant** chooses to delete food
    5.1. The **restaurant** selects the food.
6. The **restaurant** clicks on the 'Submit' button.
7. The **ManageFoodUI** displays the interface.
8. The **ManageFood** and **ManageRestaurant** save the modified food information into the database.
9. The use case ends.


a restaurant manages foods - Basic Flow
restaurant User:Restaurant User
manageFoodUI:«Boundary»ManageFoodUI
manageFoodMgr:«Control»ManageFoodMgr
manageFood:«Entity»ManageFood
manageRestaurant:«Entity»ManageRestaurant
1: click manage food button
2: food management interface displayed
opt
[add new food]
1: click the add food link
2: adding food interface displayed
3: fill in new food information
4: submit button clicked
5: request to add new food
6: add food to the restaurant
7: food added
8: display food management interface
opt
[modify or delete food]
1: click the modify food link
2: request restaurant's food information
3: find restaurant ID
4: find all food in the restaurant
5: return all food in the restaurant
6: display food modification interface
opt
[modify food information]
1: fill in new information for the food
2: the modify button clicked
3: request to modify food information
4: modify food information
5: new food information
6: display new food information
opt
[delete food information]
1: the delete button clicked
2: request to delete the food selected
3: delete the food
4: new restaurant's food information
5: display new food information

SALES MANAGEMENT

The boundary, control and entity classes are identified and listed as below:

Boundary Class

RestaurantSalesUI

This class communicates with **SalesManagementMgr** and provides the interface for users. When it retrieves the data from user, it would pass the data to **SalesManagementMgr**. Then it retrieves data from database and responses to users by displaying the data.

TotalSalesUI

This class communicates with **SalesManagementMgr** and provides the interface for administrator. When it retrieves the data from administrator, it would pass the data to **SalesManagementMgr**. Then it retrieves data from database and responses to administrator by displaying the data.

DeliveryUI

This class communicates with **SalesManagementMgr** and provides the interface for administrator.

Control Class

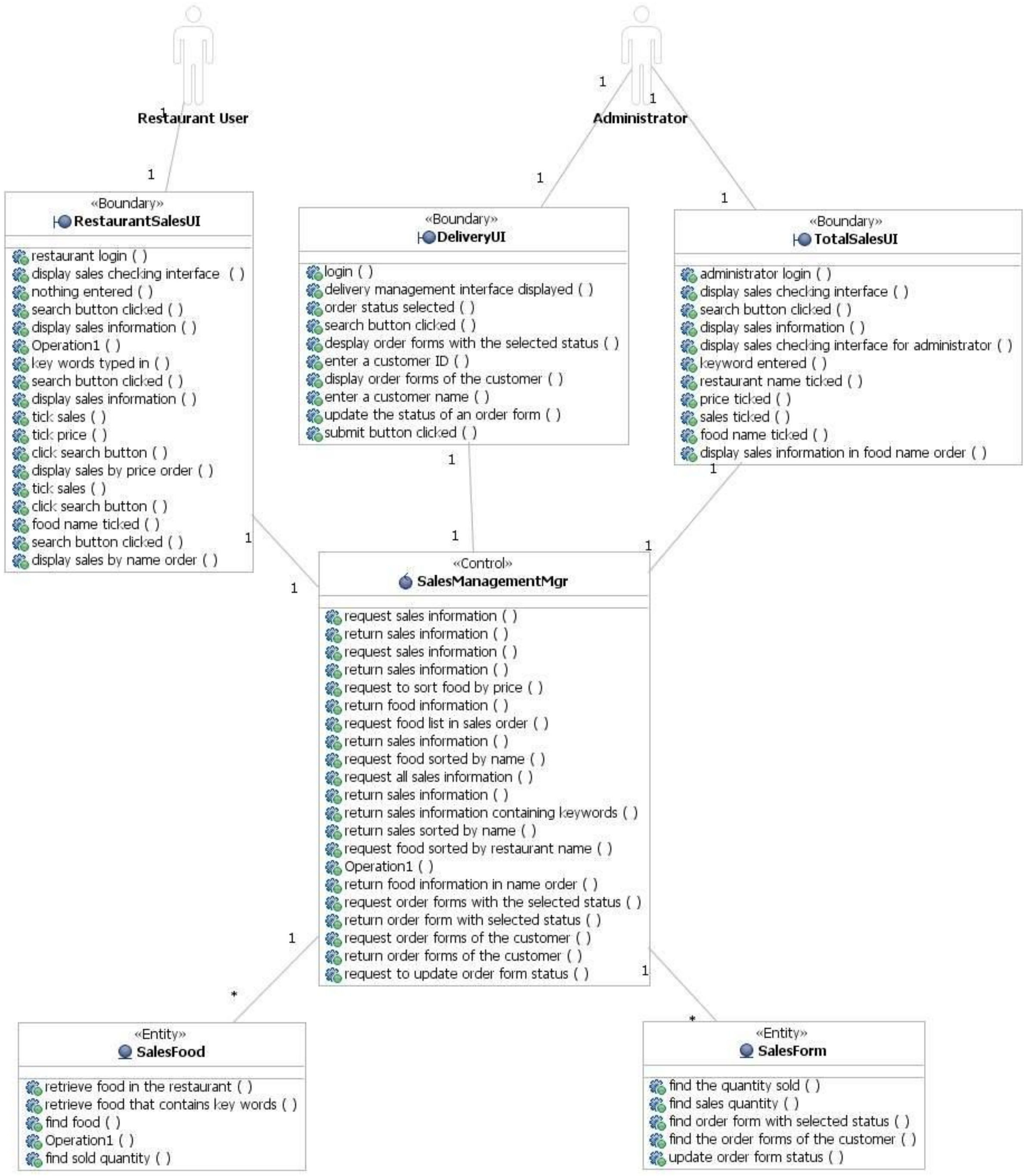
SalesManagementMgr

This class will obtain the data from boundary classes, then response the requests of user or pass the data to entity class.

Entity Classes

Sales

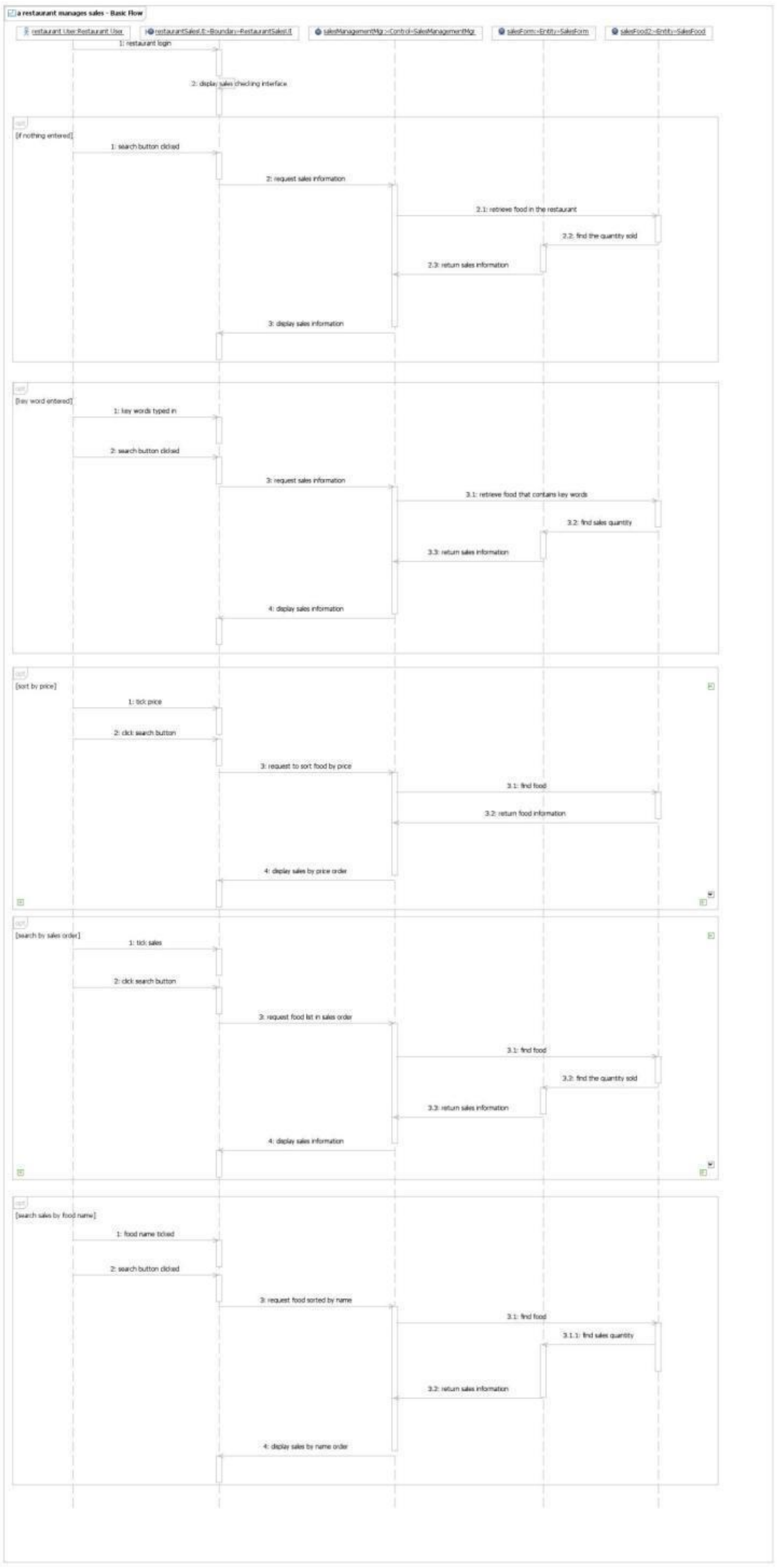
It receives the requests passed by **SalesManagementMgr** and performs the computation according to the requests.


Restaurant User
Administrator
1
1
1
1
1
1
«Boundary»
RestaurantSalesUI
restaurant login ( )
display sales checking interface ( )
nothing entered ( )
search button clicked ( )
display sales information ( )
Operation1 ( )
key words typed in ( )
search button clicked ( )
display sales information ( )
tick sales ( )
tick price ( )
click search button ( )
display sales by price order ( )
tick sales ( )
click search button ( )
food name ticked ( )
search button clicked ( )
display sales by name order ( )
«Boundary»
DeliveryUI
login ( )
delivery management interface displayed ( )
order status selected ( )
search button clicked ( )
desplay order forms with the selected status ( )
enter a customer ID ( )
display order forms of the customer ( )
enter a customer name ( )
update the status of an order form ( )
submit button clicked ( )
«Boundary»
TotalSalesUI
administrator login ( )
display sales checking interface ( )
search button clicked ( )
display sales information ( )
display sales checking interface for administrator ( )
keyword entered ( )
restaurant name ticked ( )
price ticked ( )
sales ticked ( )
food name ticked ( )
display sales information in food name order ( )
1
1
1
1
1
1
«Control»
SalesManagementMgr
request sales information ( )
return sales information ( )
request sales information ( )
return sales information ( )
request to sort food by price ( )
return food information ( )
request food list in sales order ( )
return sales information ( )
request food sorted by name ( )
request all sales information ( )
return sales information ( )
return sales information containing keywords ( )
return sales sorted by name ( )
request food sorted by restaurant name ( )
Operation1 ( )
return food information in name order ( )
request order forms with the selected status ( )
return order form with selected status ( )
request order forms of the customer ( )
return order forms of the customer ( )
request to update order form status ( )
1
*
1
*
«Entity»
SalesFood
retrieve food in the restaurant ( )
retrieve food that contains key words ( )
find food ( )
Operation1 ( )
find sold quantity ( )
«Entity»
SalesForm
find the quantity sold ( )
find sales quantity ( )
find order form with selected status ( )
find the order forms of the customer ( )
update order form status ( )

RESTAURANT SALES MANAGEMENT

Flow of events

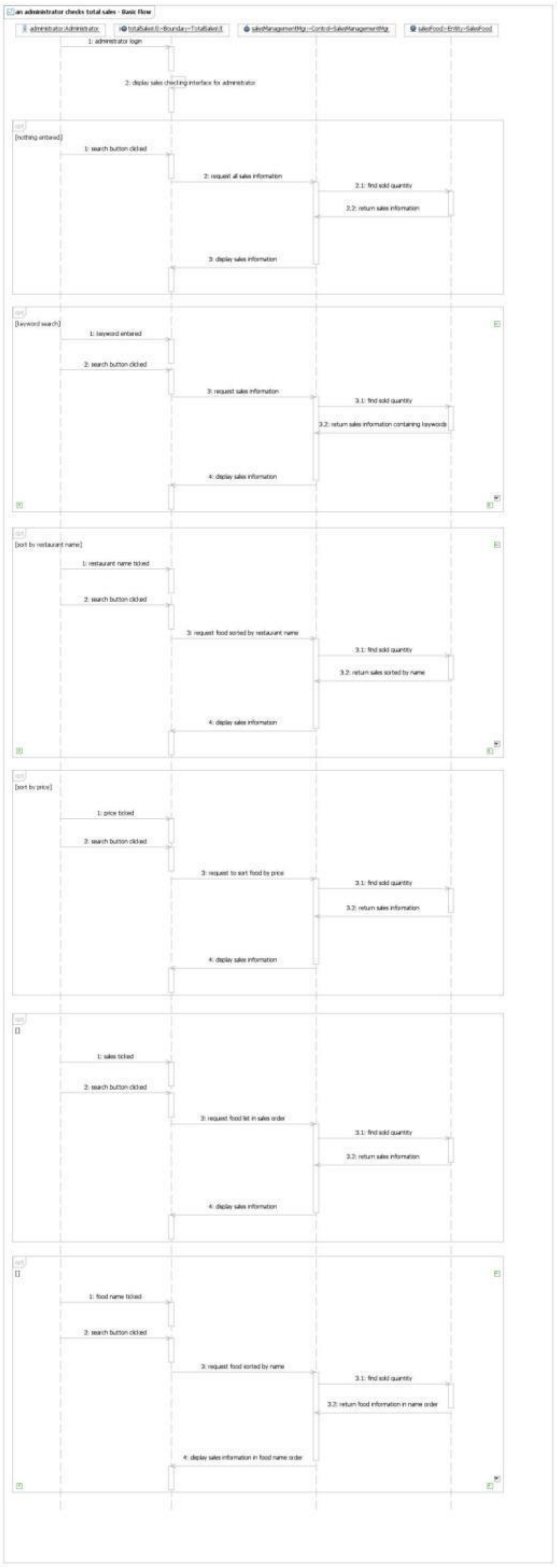
1. The use case starts when a registered restaurant logs in the system.
2. The **RestaurantSalesUI** displays the interface for checking the profit and the interface for checking the sales.
3. The **restaurant** performs searching in five means.
    - 3.1 The **restaurant** enters nothing, the **SalesManagementMgr** performs the verification.
        - 3.1.1 The **restaurant** clicks on the ‘search’ button.
        - 3.1.2 All kinds of food will be displayed in the table by **RestaurantSalesUI**.
    - 3.2 The **restaurant** enters a key word in the text box, the **SalesManagementMgr** Performs the verification and computation.
        - 3.2.1 The **restaurant** clicks on the ‘search’ button.
        - 3.2.2 All kinds of food containing the key word will be listed by **RestaurantSalesUI**.
    - 3.3 The **restaurant** ticks price, the **SalesManagementMgr** performs the verification and computation.
        - 3.3.1 The **restaurant** clicks on the ‘search’ button.
        - 3.3.2 All kinds of food will be sorted by the price in an ascending order by **RestaurantSalesUI**.
    - 3.4 The **restaurant** ticks sales, the **SalesManagementMgr** performs computation.
        - 3.4.1 The **restaurant** clicks on the ‘search’ button.
        - 3.4.2 All kinds of food will be sorted by the sales quantity by the **RestaurantSalesUI**.
    - 3.5 The **restaurant** ticks food name, the **SalesManagementMgr** performs computation.
        - 3.5.1 The **restaurant** clicks on the ‘search’ button.
        - 3.5.2 All kinds of food will be sorted by the food name in alphabetic order by the **RestaurantSalesUI**.
4. The use case ends.


a restaurant manages sales - Basic Flow
restaurant User:Restaurant User
restaurantSalesUI:<<Boundary>>RestaurantSalesUI
salesManagementMgr:<<Control>>SalesManagementMgr
salesForm:<<Entity>>SalesForm
salesFood2:<<Entity>>SalesFood
1: restaurant login
2: display sales checking interface
opt
[if nothing entered]
1: search button clicked
2: request sales information
2.1: retrieve food in the restaurant
2.2: find the quantity sold
2.3: return sales information
3: display sales information
opt
[key word entered]
1: key words typed in
2: search button clicked
3: request sales information
3.1: retrieve food that contains key words
3.2: find sales quantity
3.3: return sales information
4: display sales information
opt
[sort by price]
1: tick price
2: click search button
3: request to sort food by price
3.1: find food
3.2: return food information
4: display sales by price order
opt
[search by sales order]
1: tick sales
2: click search button
3: request food list in sales order
3.1: find food
3.2: find the quantity sold
3.3: return sales information
4: display sales information
opt
[search sales by food name]
1: food name ticked
2: search button clicked
3: request food sorted by name
3.1: find food
3.1.1: find sales quantity
3.2: return sales information
4: display sales by name order

ADMINISTRATOR CHECKS THE TOTAL SALES OF THE SYSTEM

Flow of events

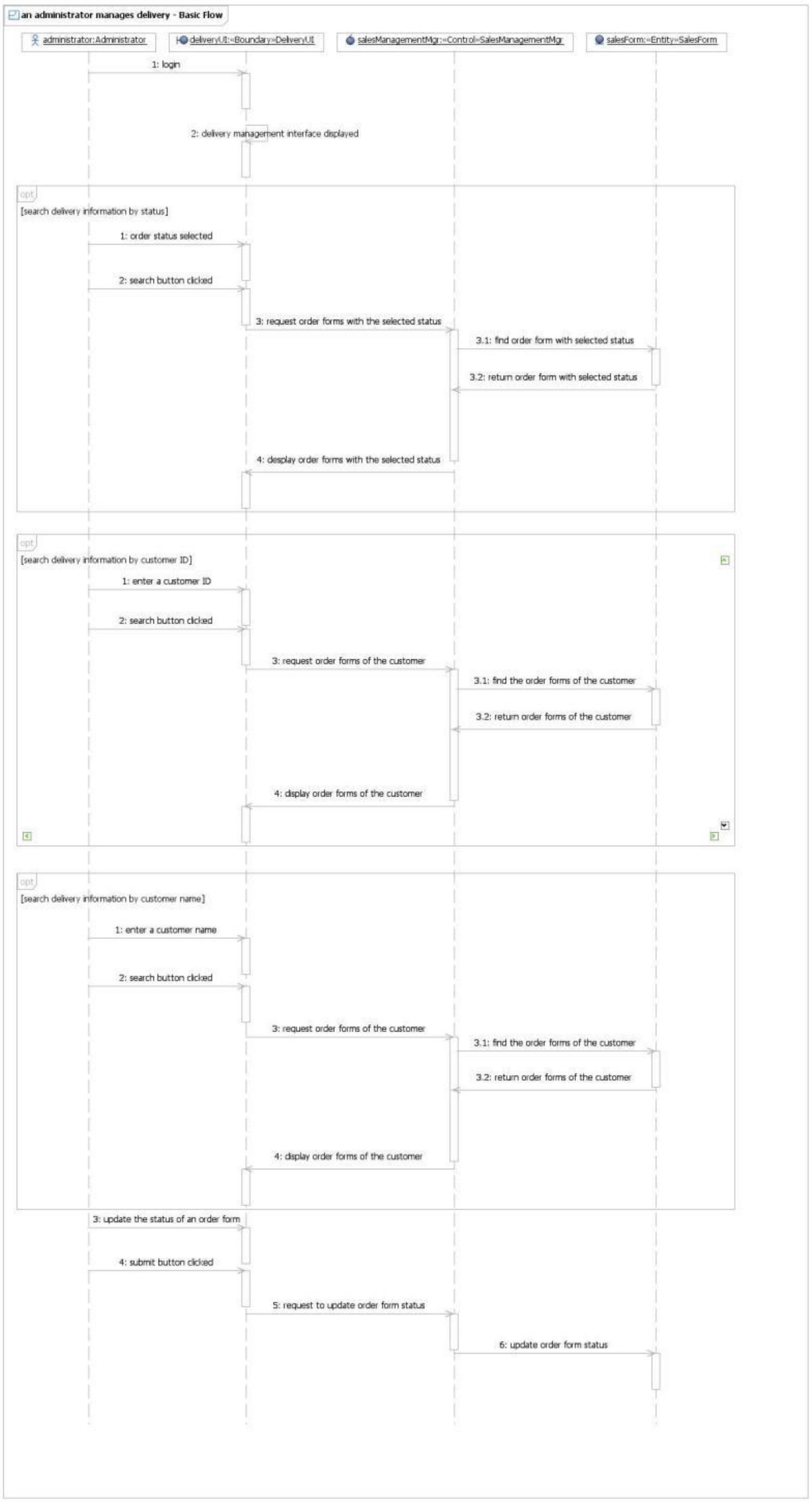
1. The use case starts when an **administrator** logs in the system.
2. The **TotalSalesUI** displays the interface of sales information.
3. The **administrator** performs checking in six means.
    - 3.1. The **administrator** enters nothing, the **SalesManagementMgr** performs verification.
        - 3.1.1. The **administrator** clicks on the 'search' button.
        - 3.1.2. The sales information of all food will be displayed in the table by the **TotalSalesUI**.
    - 3.2. The **administrator** enters a key word in the text box; the **SalesManagementMgr** performs verification and the computation.
        - 3.2.1. The **administrator** clicks on the 'search' button.
        - 3.2.2. The sales information of all food containing the key word will be listed in the table by the **TotalSalesUI**.
    - 3.3. The **administrator** ticks 'Restaurant name'. The **SalesManagementMgr** performs verification and the computation.
        - 3.3.1. The **administrator** clicks on the 'search' button.
        - 3.3.2. The sales information of all food will be listed in group of each restaurant by the **TotalSalesUI**.
    - 3.4. The **administrator** ticks 'price'. The **SalesManagementMgr** performs verification and the computation.
        - 3.4.1. The **administrator** clicks on the 'search' button.
        - 3.4.2. The sales information of all food will be listed in an ascend order of the price by the **TotalSalesUI**.
    - 3.5. The **administrator** ticks 'sales'. The **SalesManagementMgr** performs verification and the computation.
        - 3.5.1. The **administrator** clicks on the 'search' button.
        - 3.5.2. The sales information of all food will be listed in a descend order of the selling quantity by the **TotalSalesUI**.
    - 3.6. The **administrator** ticks 'Food Name'. The **SalesManagementMgr** performs verification and the computation.
        - 3.6.1. The **administrator** clicks on the 'search' button.
        - 3.6.2. The sales information of all food will be listed alphabetically by the **TotalSalesUI**.
4. The use case ends.


an administrator checks total sales - Basic Flow
administrator:Administrator
totalSalesUI:«Boundary»TotalSalesUI
salesManagementMgr:«Control»SalesManagementMgr
salesFood:«Entity»SalesFood
1: administrator login
2: display sales checking interface for administrator
opt
[nothing entered]
1: search button clicked
2: request all sales information
2.1: find sold quantity
2.2: return sales information
3: display sales information
opt
[keyword search]
1: keyword entered
2: search button clicked
3: request sales information
3.1: find sold quantity
3.2: return sales information containing keywords
4: display sales information
opt
[sort by restaurant name]
1: restaurant name ticked
2: search button clicked
3: request food sorted by restaurant name
3.1: find sold quantity
3.2: return sales sorted by name
4: display sales information
opt
[sort by price]
1: price ticked
2: search button clicked
3: request to sort food by price
3.1: find sold quantity
3.2: return sales information
4: display sales information
opt
[]
1: sales ticked
2: search button clicked
3: request food list in sales order
3.1: find sold quantity
3.2: return sales information
4: display sales information
opt
[]
1: food name ticked
2: search button clicked
3: request food sorted by name
3.1: find sold quantity
3.2: return food information in name order
4: display sales information in food name order

ADMINISTRATOR MANAGE THE DELIVERY

Flow of events

1. The use case starts when an **administrator** logs in the system.
2. The **DeliveryUI** displays the interface of delivery management.
3. The **administrator** performs checking and modification in four means.
    - 3.1. The **administrator** enters nothing.
        - 3.1.1. The **administrator** clicks on the ‘search’ button.
        - 3.1.2. All the order form information will be displayed in the table.
    - 3.2. The **administrator** selects an order status in the pull down menu.
        - 3.2.1. The **administrator** clicks on the ‘search’ button.
        - 3.2.2. All the order forms with the selected status will be listed.
    - 3.3. The **administrator** enters a customer ID.
        - 3.3.1. The **administrator** clicks on the ‘search’ button.
        - 3.3.2. All the order forms of the customer will be listed.
    - 3.4. The **administrator** enters a customer name.
        - 3.4.1. The **administrator** clicks on the ‘search’ button.
        - 3.4.2. All the order forms of the customer will be listed.
4. The **administrator** updates the status of the order form from the pull down menu.
5. The **administrator** clicks on ‘submit’ button at the bottom of this table to confirm the action.
6. The use case ends.


an administrator manages delivery - Basic Flow
administrator:Administrator
deliveryUI:«Boundary»DeliveryUI
salesManagementMgr:«Control»SalesManagementMgr
salesForm:«Entity»SalesForm
1: login
2: delivery management interface displayed
opt
[search delivery information by status]
1: order status selected
2: search button clicked
3: request order forms with the selected status
3.1: find order form with selected status
3.2: return order form with selected status
4: desplay order forms with the selected status
opt
[search delivery information by customer ID]
1: enter a customer ID
2: search button clicked
3: request order forms of the customer
3.1: find the order forms of the customer
3.2: return order forms of the customer
4: display order forms of the customer
opt
[search delivery information by customer name]
1: enter a customer name
2: search button clicked
3: request order forms of the customer
3.1: find the order forms of the customer
3.2: return order forms of the customer
4: display order forms of the customer
3: update the status of an order form
4: submit button clicked
5: request to update order form status
6: update order form status

REGISTRATION

BOUNDARY CLASSES

RegistrationUI		
Responsibility	**Collaborators**	
Register a customer user to the system.	RegistrationMgr	
Register a restaurant user to the system.	RegistrationMgr	
Attributes of class		
Name	*Description*	*Data Type*

CONTROL CLASSES

RegistrationMgr		
Responsibility	**Collaborators**	
Start registration of a user. Generate user ID. Generate registration time.	RegistrationUI Customer, Restaurant Customer	
Add user to the database.	Customer, Restaurant	
Attributes of class		
Name	*Description*	*Data Type*

ENTITY CLASSES

RegisterCustomer		
Responsibility	**Collaborators**	
The customer account information.		
Attributes of class		
Name	*Description*	*Data Type*
Customer ID	The unique key to identify a customer.	Integer
Login name	The name of the customer account.	Char
Real name	The real name of the customer.	Char
Password	The password of the account.	Char
Address	Address of the customer.	Char
Phone	The contact phone number of the customer.	Char
Email	The email address of the customer.	Char
Customer registration date	The date when the customer registered.	Datetime

RegisterRestaurant		
Responsibility	**Collaborators**	
The restaurant account information.		
Attributes of class		
Restaurant ID	The unique key to identify a restaurant.	Integer
Login name	The log-in name of the restaurant account.	Char
Real name	The real name of the restaurant.	Char

Password	The password of the account.	Char
Address	Address of the restaurant.	Char
Phone	The contact phone number of the restaurant.	Char
Revenue	The revenue of the restaurant on our system.	Decimal
Cost	The cost of the restaurant on our system.	Decimal
Link	The web page link of the restaurant.	Varchar
Picture Link	The restaurant’s picture link address on server.	Varchar
Open Time	The restaurant’s open time.	Varchar
Close Time	The restaurant’s close time.	Varchar
Rate	The rating of restaurant.	Decimal
Email	The email address of the restaurant.	Char
Customer registration date	The date when the restaurant registered.	Datetime

SEARCH FOOD

BOUNDARY CLASSES

SearchFoodUI		
Responsibility	**Collaborators**	
Display the food matched.	SearchMgr	
Display the food details.	SearchMgr	
Attributes of class		
Name	*Description*	*Data Type*
Restaurant name	*The name of restaurant to be selected.*	*String*
Price	*The preferred price of food.*	*Integer*
Key word	*The key word of the food.*	*String*

SearchRestaurantUI		
Responsibility	**Collaborators**	
Display the food matched.	SearchMgr	
Display the food details.	SearchMgr	
Display the restaurant information.	SearchMgr	
Attributes of class		
Name	*Description*	*Data Type*
Restaurant name	*The name of restaurant to be selected.*	*String*

CONTROL CLASSES

SearchMgr		
Responsibility	**Collaborators**	
Check the verification of the input.	Food, Restaurant	
Get the restaurant page.	Restaurant	
Get the food details.	Food, Restaurant	
Attributes of class		
Name	*Description*	*Data Type*

ENTITY CLASSES

SearchFood		
Responsibility	**Collaborators**	
The food information.		
Attributes of class		
Name	*Description*	*Data Type*
Food ID	The unique key to identify a food.	Integer
Restaurant ID	ID of the restaurant offering the food.	Integer
Food name	The name of the food.	Char
Food price	The price of the food.	Decimal
Food description	The description of the food.	Varchar
Food picture_link	The picture link address of the food in server.	Varchar
Food quantity	The quantity of food offered by the restaurant.	Integer
Food cost	The cost to produce the food (for statistic use).	Decimal

SearchRestaurant		
Responsibility	**Collaborators**	
The restaurant link page.		
Attributes of class		
Name	*Description*	*Data Type*
Restaurant ID	The unique key to identify a restaurant.	Integer
Real name	The real name of the restaurant.	Char
Address	Address of the restaurant.	Char
Phone	The contact phone number of the restaurant.	Char
Link	The web page link of the restaurant.	Varchar
Open Time	The restaurant's open time.	Varchar
Close Time	The restaurant's close time.	Varchar
Email	The email address of the restaurant.	Char

ORDER FOOD

BOUNDARY CLASSES

OrderFoodUI		
Responsibility	**Collaborators**	
Display the food ordered.	OrderFoodMgr	
Display the food details.	OrderFoodMgr	
Display the order page.	OrderFoodMgr	
Display the invoice page.	OrderFoodMgr	
Attributes of class		
Name	*Description*	*Data Type*

CheckOrderUI	
Responsibility	**Collaborators**

Display the order form of the customer.	OrderFoodMgr	
Display the order form detailss.	OrderFoodMgr	
Attributes of class		
Name	*Description*	*Data Type*

CONTROL CLASSES

OrderFoodMgr		
Responsibility	**Collaborators**	
Retrieve the order information.	OrderFoodUI, OrderCustomer, OrderFood, OrderRestaurant	
Add the order to the system.	OrderFoodUI, OrderRestaurant, OrderFood, OrderRestaurant	
Retrieve the order form information from the database.	OrderFoodUI, OrderForm	
Attributes of class		
Name	*Description*	*Data Type*

ENTITY CLASSES

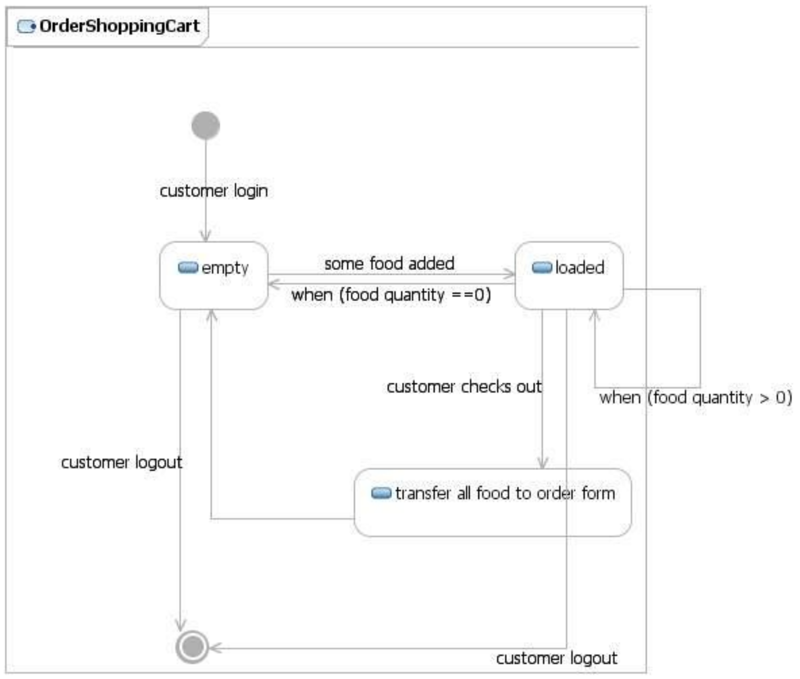
OrderCustomer		
Responsibility	**Collaborators**	
Saves the customer information of the order.		
Attributes of class		
Name	*Description*	*Data Type*
Receiver name.	The receiver name of the food order.	String
Receiver address.	The delivery address of the food order.	String
Receciver cellphone.	The contact information of the receiver.	Integer

OrderFood		
Responsibility	**Collaborators**	
Saves the food information of the order.		
Attributes of class		
Name	*Description*	*Data Type*
Food name.	The food name in the order.	String
Food quantity.	The quantity of the food ordered.	Integer
Food price.	The price of the food ordered.	Decimal

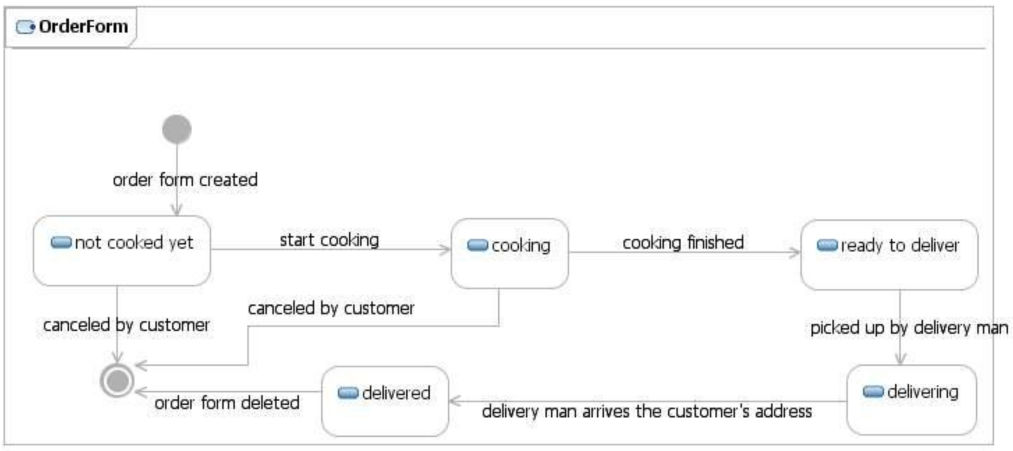
OrderRestaurant		
Responsibility	**Collaborators**	
Records the restaurant of the ordered food.		
Attributes of class		
Name	*Description*	*Data Type*
Restaurant name.	The name of the restaurant.	String

Restaurant ID.	The unique ID to identify the restaurant.	Integer

OrderShoppingCart		
Responsibility	**Collaborators**	
The shopping cart information of a customer.		
Attributes of class		
Name	*Description*	*Data Type*
Restaurant name.	The name of the restaurant.	String
Restaurant ID.	The unique ID to identify the restaurant.	Integer



OrderForm		
Responsibility	**Collaborators**	
The order form information of the customer.		
Attributes of class		
Name	*Description*	*Data Type*
Form ID	The unique ID to identify the order form.	Integer
Customer ID	The ID of the customer who ordered the form.	Integer
Form time	The time when the order form is generated.	Datetime
Form state	The state of the order form.	Enumeration
Receiver name	The receiver name of the food order.	String
Receiver address	The delivery address of the food order.	String
Receciver cellphone	The contact information of the receiver.	Integer
Food ID	The ID of food in the order form.	Integer
Food quantity	The quantity of the food in the order form.	Integer



SALES MANAGEMENT

BOUNDARY CLASSES

RestaurantSalesUI		
Responsibility	**Collaborators**	
Display the food sale of the restaurant.	SalesManagementMgr	
Attributes of class		
Name	*Description*	*Data Type*
Price	The price range.	Decimal
Sales	The total sale.	Integer
Food name	The name of the food.	String

TotalSalesUI		
Responsibility	**Collaborators**	
Display the food sale of the system.	SalesManagementMgr	
Attributes of class		
Name	*Description*	*Data Type*
Restaurant name	*The name of the restaurant*	*String*
Price	The price range.	Decimal
Sales	The total sale.	Integer
Food name	The name of the food.	String

DeliveryUI		
Responsibility	**Collaborators**	
Display the order forms.	SalesManagementMgr	
Attributes of class		
Name	*Description*	*Data Type*

CONTROL CLASSES

SalesManagementMg	
Responsibility	**Collaborators**

Retrieves the sales information.	Restaurant sales UI, SalesForm	
Retrieves the delivery information.	Delivery UI, SalesForm	
Attributes of class		
Name	*Description*	*Data Type*

ENTITY CLASSES

SalesForm		
Responsibility	**Collaborators**	
Saves the order form information.		
Attributes of class		
Name	*Description*	*Data Type*
Form ID	The unique ID to identify the order form.	Integer
Form time	The time when the order form is generated.	Datetime
Form state	The state of the order form.	Enumeration
Customer ID	The ID of the customer who ordered the form.	Integer
Customer name	The customer name in the order form.	String
Restaurant name	The name of the restaurant.	String

SalesFood		
Responsibility	**Collaborators**	
Saves the information of food.		
Attributes of class		
Name	*Description*	*Data Type*
Food name	The name of the food.	String
Food ID	The ID of food in the order form.	Integer
Food quantity	The quantity of the food in the order form.	Integer
Price	The price range of food.	Decimal
Sales	The total sale of the food..	Integer

MANAGE FOOD

BOUNDARY CLASSES

ManageFoodUI		
Responsibility	**Collaborators**	
Display the food management.	ManageFoodMgr	
Attributes of class		
Name	*Description*	*Data Type*

CONTROL CLASSES

ManageFoodMgr	
Responsibility	**Collaborators**
Retrieves the food information.	**ManageFoodUI, ManageFood, ManageRestaurant**

Upload the food picture.	**ManageFood, ManageRestaurant**	
Generate the food ID.	**ManageFood**	
Attributes of class		
Name	*Description*	*Data Type*

ENTITY CLASSES

ManageFood		
Responsibility	**Collaborators**	
The information of the newly added food.		
Attributes of class		
Name	*Description*	*Data Type*
Food ID	The unique key to identify a food.	Integer
Restaurant ID	ID of the restaurant offering the food.	Integer
Food name	The name of the food.	Char
Food price	The price of the food.	Decimal
Food description	The description of the food.	Varchar
Food picture_link	The picture link address of the food in server.	Varchar
Food quantity	The quantity of food offered by the restaurant.	Integer

ManageRestaurant		
Responsibility	**Collaborators**	
The restaurant information.		
Attributes of class		
Name	*Description*	*Data Type*
Restaurant ID	ID of the restaurant offering the food.	Integer
Restaurant name	The name of the restaurant where food is added.	String

DESIGN MODEL • DATABASE DESIGN

Tables contained in the database file are customer, restaurant, food, order_form, contain and message.

CUSTOMER

Key

cus_ID

Data Specification

cus_name: The real name of the customer.

cus_login_name: The login name of the customer to the system.

cus_ID is automatically generated by the database.

Other Content Data

cus_passcode, cus_address, cus_email, cus_phone, cus_register_date

Derived Variable
Abandoned Variable

RESTURANT
Key
res_ID
Data Specification
res_name: the real name of the restaurant, mcdonald e.g.
res_login_name: the login name of the restaurant to the system.
rest_revenue: the revenue of the restaurant on the system.
rest_cost: the cost of the all the food sold in the system.
Other Content Data
rest_passcode, rest_address, rest_phone, rest_register_date, rest_email, rest_open_time, rest_close_time, rest_pic_link
Derived Variable
rest_link
Abandoned Variable
rest_rate

FOOD
Key
food_ID
Data Specification
food_name: The name of food, different restaurant can have the same food name, but with different food ID.
rest_id: To identify the restaurant offering the food.
Other Content Data
food_price, food_description, food_picture_link, food_quantity
Derived Variable
Abandoned Variable
food_cost

ORDER_FORM
Key
form_ID
Data Specification
The form_ID is automatically generated by the database.
cus_id: The ID of the customer who orders the form.
form_state: The state of the order, 'not cooked yet', 'cooking', 'ready to deliver', 'delivering', 'done'.
Other Content Data
receiver_name, receiver_address, receiver_phone.
Derived Variable
form_time

Abandoned Variable

CONTAIN

Key

contain_id

Data Specification

The contain ID is automatically generated by the database.

form_id: The ID of the form to which the contain element belongs.

Other Content Data

food_id

Derived Variable

order_quantity

Abandoned Variable

MESSAGE

Key

message_ID

Data Specification

The message_ID is automatically generated by the database.

Other Content Data

Message_title, message_name, message_mail, message_time, message_content

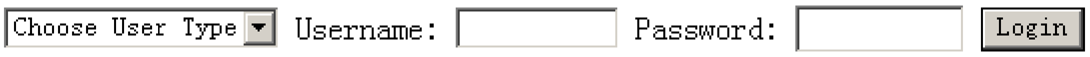
Derived Variable

Abandoned Variable

DESIGN MODEL • USE-CASE REALIZATION -- DESIGN

Registration

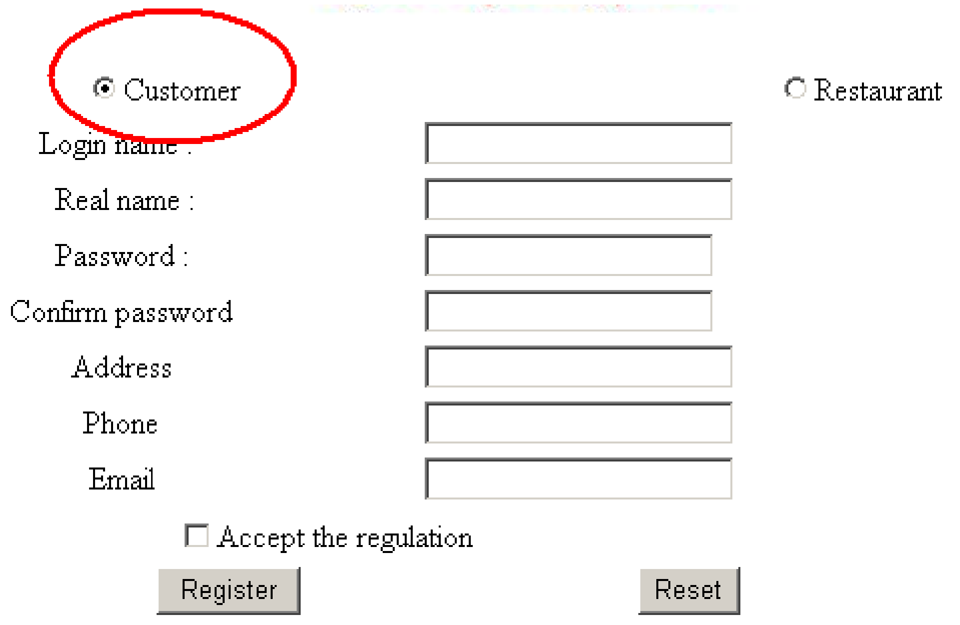
Boundary Class



Input of	Format
Choose User Type	Pull down menu
Username	Char(20)
Password	Char(10)

Customer registration UI

This is the user interface that deals with the customer registration.



[✓ Customer] Restaurant

Login name :	
Real name :	
Password :	
Confirm password	
Address	
Phone	
Email	

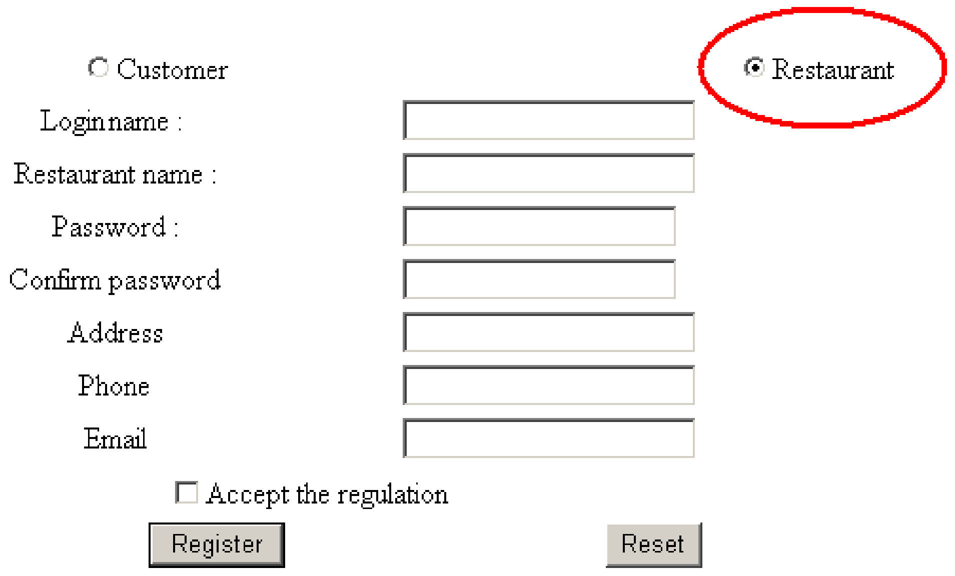
☐ Accept the regulation

Register Reset

Input of	Format	Description
Select the user type	Radio Button	Customer user
Login name	Char (20)	The name of the customer account
Real name	Char (30)	The real name of the customer
Password	Char (10)	The password of the account
Confirm password	Char (10)	The password confirmed by the user
Address	Char (50)	Address of the customer
Phone	Char (8)	The contact phone number of the customer
Email	Char (20)	The email address of the customer
Accept the regulation	Check Box	Acceptance of the Lumen regulation for public users

Restaurant registration UI

This is the user interface that deals with the restaurant registration.



Customer [✓ Restaurant]

Login name :	
Restaurant name :	
Password :	
Confirm password	
Address	
Phone	
Email	

☐ Accept the regulation

Register Reset

Input of	Format	Description
Select the user type	Radio Button	Restaurant user
Login name	Char (20)	The name of the restaurant account
Restaurant name	Char (20)	The name of the restaurant
Password	Char (10)	The password of the account
Confirm password	Char (10)	The password confirmed by the user
Address	Char (50)	Address of the restaurant
Phone	Int (8)	The contact phone number of the restaurant
Email	Char (20)	The email address of the restaurant
Accept the regulation	Check Box	Acceptance of the Lumen regulation for public users

Control Class

RegistrationMgr	
Responsibilities	**Collaborators**
Start registration of a user	Registration UI
Generate user ID	Customer, Restaurant
Generate registration time	Customer
Add user to the database	Customer, Restaurant

Entity Classes

RegisterCustomer:

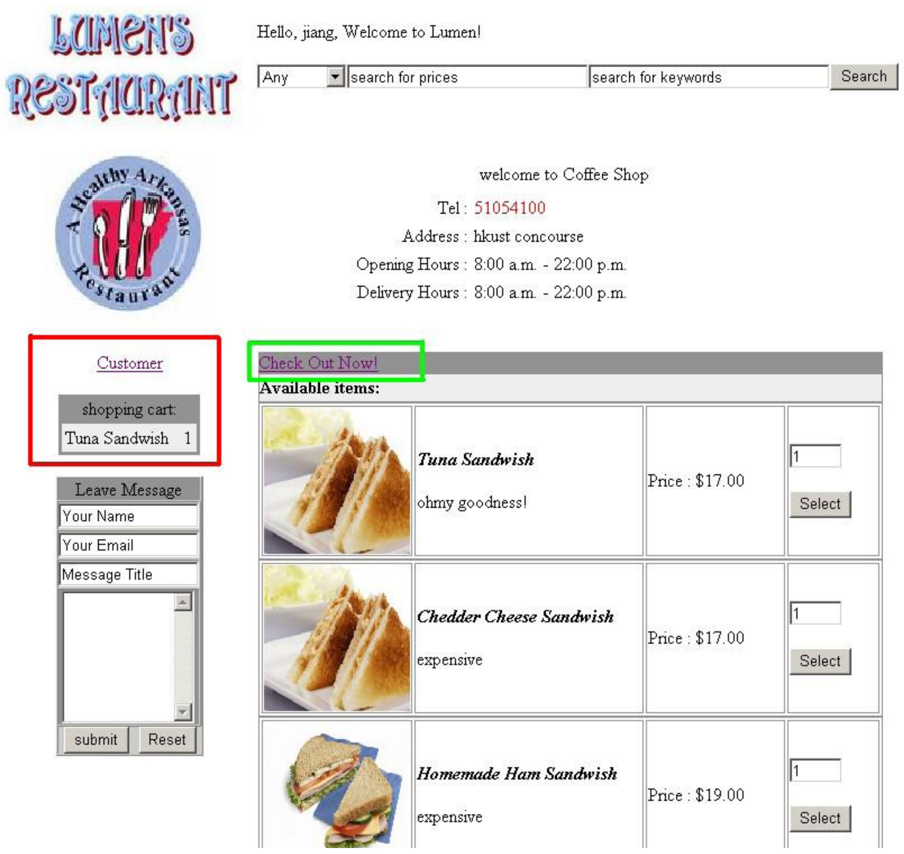
This class is used to verify the validation of the input. And, if the input of the required customer account is valid it adds the customer account to the database.

RegisterRestaurant:

This class is used to verify the validation of the input. And, if the input of the required restaurant account is valid it adds the customer account to the database.

Order food

Boundary Class


Hello, jiang, Welcome to Lumen!

welcome to Coffee Shop

Tel : 51054100

Address : hkust concourse

Opening Hours : 8:00 a.m. - 22:00 p.m.

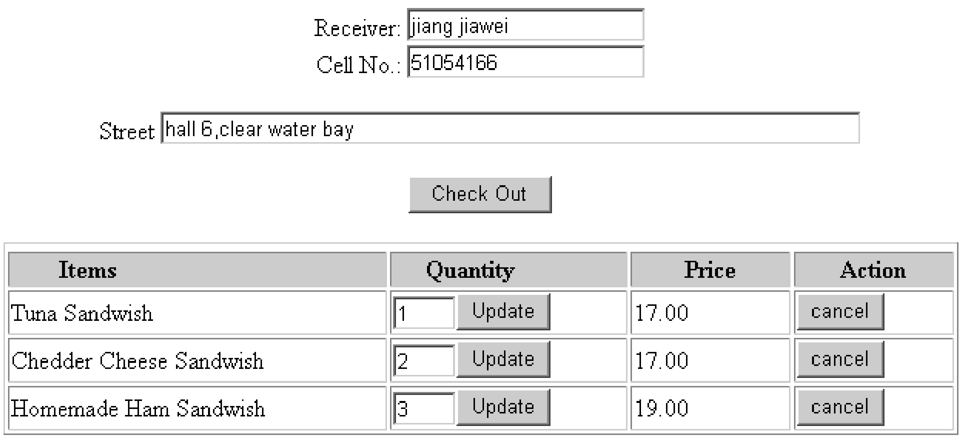
Delivery Hours : 8:00 a.m. - 22:00 p.m.

	Tuna Sandwish ohmy goodness!	Price : $17.00	1 Select
	Chedder Cheese Sandwish expensive	Price : $17.00	1 Select
	Homemade Ham Sandwish expensive	Price : $19.00	1 Select

Input of	Format	Description
Quantity of the item	Int	The quantity of the ordered items

Check out UI

This is the user interface that deals with the check out process.


Receiver: jiang jiawei

Cell No.: 51054166

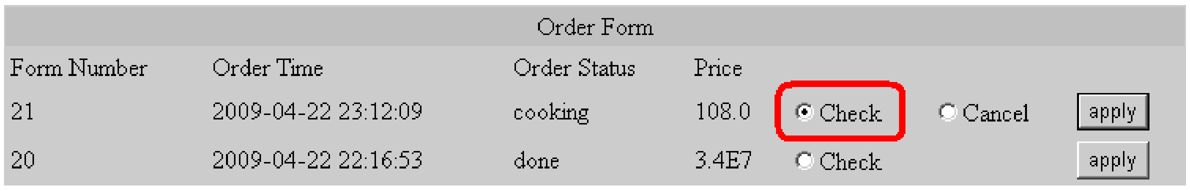
Street hall 6,clear water bay

Items	Quantity	Price	Action
Tuna Sandwish	1 Update	17.00	cancel
Chedder Cheese Sandwish	2 Update	17.00	cancel
Homemade Ham Sandwish	3 Update	19.00	cancel

Input of	Format	Description
Receiver	Char (30)	The default name is the user name of current account
Cell Number	Char (8)	The default cell number is the number of current account
Street	Char (50)	The default address is the address of current user
Quantity of items	int	Users are able to update the quantity

Order form UI

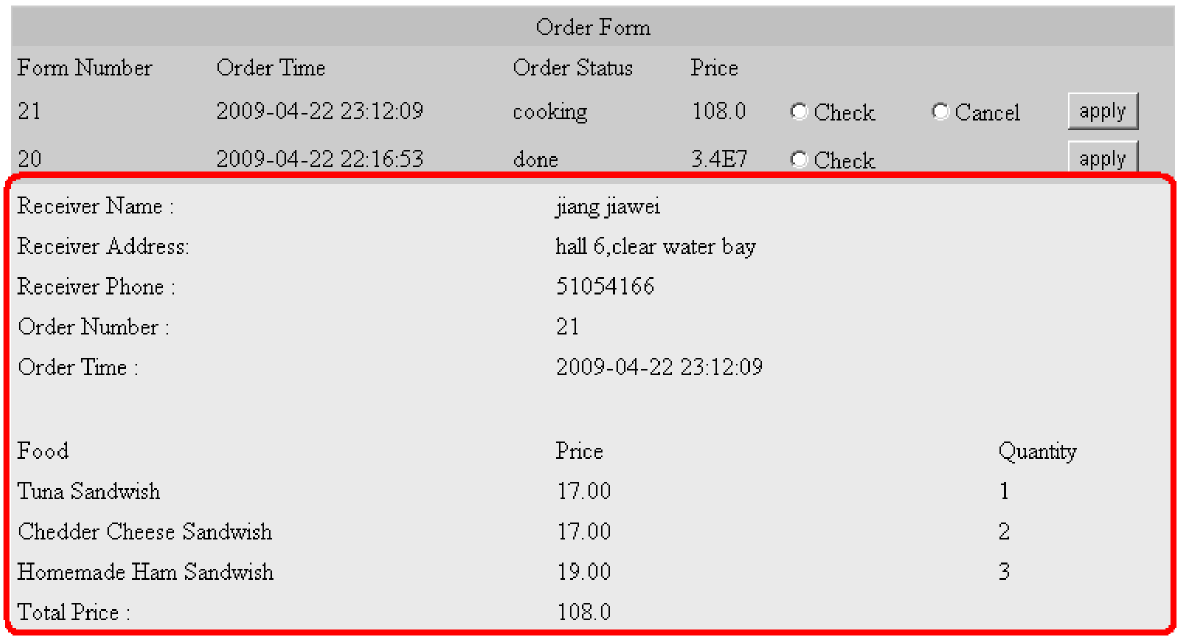
This is the user interface which displays the order form for the user during the check out process. This user interface also provides the function of order modification.



Order Form						
Form Number	Order Time	Order Status	Price			
21	2009-04-22 23:12:09	cooking	108.0	[✓ Check]	Cancel	apply
20	2009-04-22 22:16:53	done	3.4E7	Check		apply

Detailed Order Form UI

This is the user interface of the detailed order form.



Order Form						
Form Number	Order Time	Order Status	Price			
21	2009-04-22 23:12:09	cooking	108.0	Check	Cancel	apply
20	2009-04-22 22:16:53	done	3.4E7	Check		apply

Receiver Name :	jiang jiawei	
Receiver Address:	hall 6,clear water bay	
Receiver Phone :	51054166	
Order Number :	21	
Order Time :	2009-04-22 23:12:09	
Food	Price	Quantity
Tuna Sandwish	17.00	1
Chedder Cheese Sandwish	17.00	2
Homemade Ham Sandwish	19.00	3
Total Price :	108.0	

Input of	Format	Description
Check	Radio Box	Check the details of the order
Cancel	Radio Box	Cancel the order if the Order Status is not 'done'

Control Class

OrderFoodMgr	
Responsibilities	**Collaborators**
Retrieve the order information	OrderFoodUI, OrderCustomer, OrderFood, OrderRestaurant
Add the order to the system	OrderFoodUI, OrderRestaurant, OrderFood, OrderRestaurant
Retrieve the order form information from the database	OrderFoodUI, OrderForm

Entity Classes

OrderFood:

This class is used to retrieve the ordered food and add the information to the order form.

OrderRestaurant:

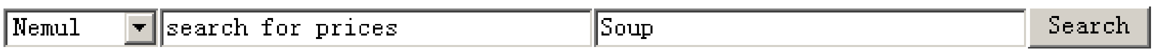
This class is used to retrieve the restaurant information of the ordered food and add it to the order form.

OrderCustomer:

This class is used to retrieve the customer who orders the food and add it to the order form.

Search Food

Boundary Class



Input of	Format	Description
Select restaurant	Pull down menu	
Search for price range	Text	The range of price of the food in the restaurant
Search for keywords	Text	The keywords in the food name

Control Class

SearchMgr	
Responsibilities	**Collaborators**
Check the and verify the input	Food, Restaurant
Get the restaurant page	Restaurant
Get the food details	Food, Restaurant

Entity Classes

SearchFood:

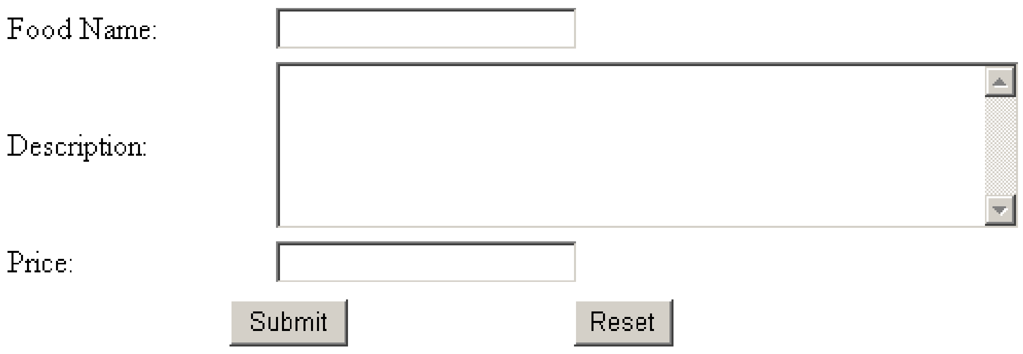
This class handles the requests from users and displays the food search results.

SearchRestaurant:

This class handles the requests from users and displays the search results which belong to the same restaurant.

Manage Food

Boundary Class



Input of	Format	Description
Food Name	Char (50)	The name of the new food
Description	Char (100)	The description of the new food
Price	Decimal (4,2)	The price of the food

Control Class

ManageFoodMgr	
Responsibilities	**Collaborators**
Retrieves the food information	ManageFoodUI, ManageFood, ManageRestaurant
Upload the food picture	ManageFood, ManageRestaurant
Generate the food ID	ManageFood

Entity Classes

ManageFood:

This class is used to manage the food information of the system and the attribute in database.

ManageFoodUI:

This class is used to provide the user interface to restaurant such that the food information can be managed by the restaurant.

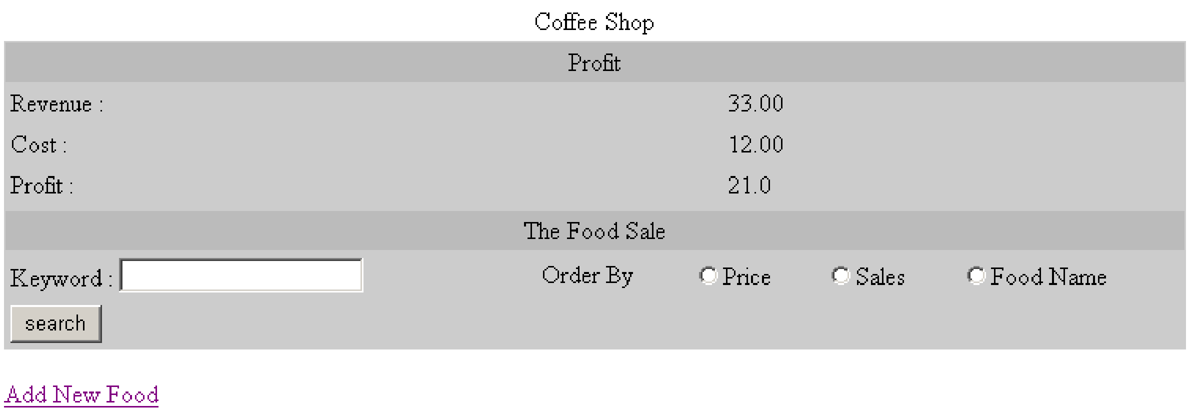
ManageRestaurant:

This class is used to update the restaurant information.

Sales Management

Boundary Class

Restaurant UI



Coffee Shop

Profit	
Revenue :	33.00
Cost :	12.00
Profit :	21.0

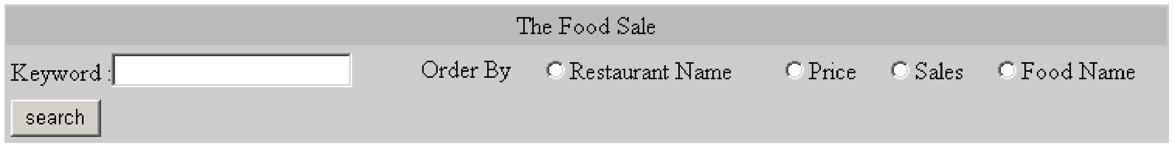
The Food Sale

Keyword : Order By ○ Price ○ Sales ○ Food Name

Input of	Format	Description
Keyword	Text	Keyword in the food name
Order by price	Radio Box	The sales information will be sorted by the food's price in a non-descending order
Order by Sales	Radio Box	The sales information will be sorted by the sales quantity of the food in a non-descending order
Order by food name	Radio Box	The sales information will be sorted by the food name in an alphabetical order

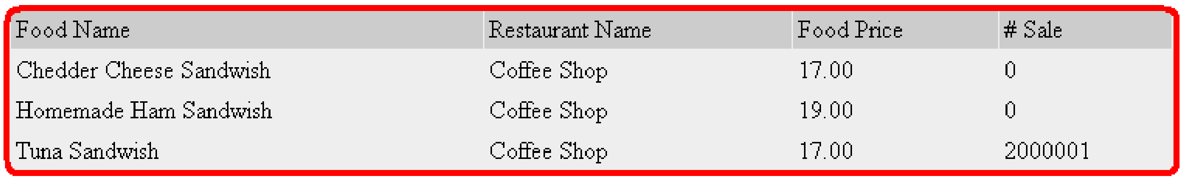
Administrator UI

This is the user interface from the administrator's point of view. Administrator is able to browse, reply and delete by login as an administrator.



The Food Sale

Keyword : Order By ○ Restaurant Name ○ Price ○ Sales ○ Food Name



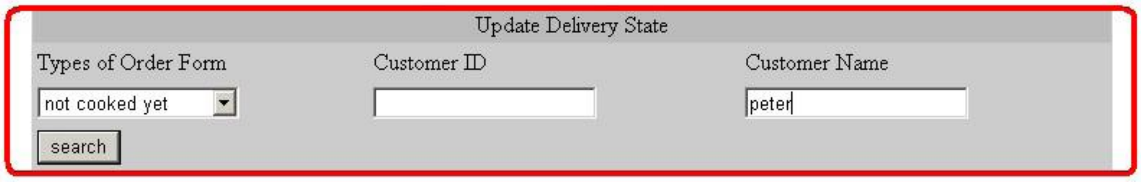
Food Name	Restaurant Name	Food Price	# Sale
Chedder Cheese Sandwish	Coffee Shop	17.00	0
Homemade Ham Sandwish	Coffee Shop	19.00	0
Tuna Sandwish	Coffee Shop	17.00	2000001

Input of	Format	Description
Keyword	Text	The sales information which contains the keyword will be displayed
Order by Restaurant Name	Radio Box	The sales information of the restaurant will be displayed
Order by Price	Radio Box	The sales information will be sorted by the price
Order by Food Name	Radio Box	The sales information of the food will be displayed

Delivery management UI

This is the user interface that manages the delivery status.

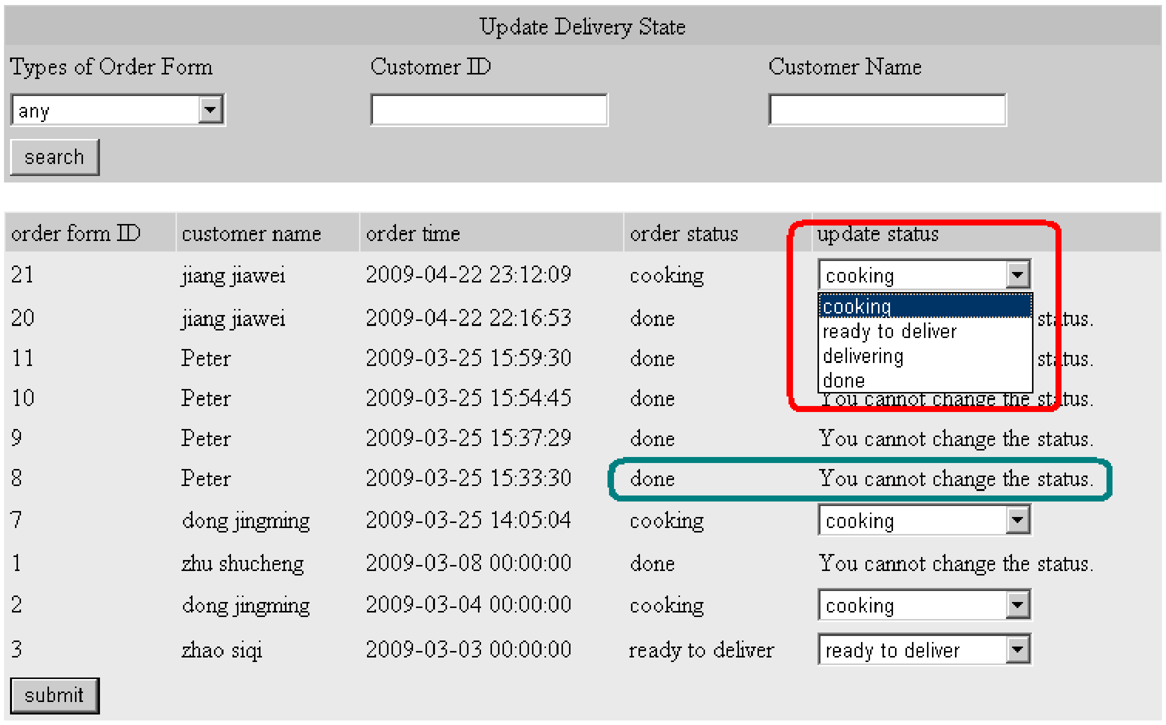
LUMEN'S RESTAURANT



Update Delivery State

Types of Order Form	Customer ID	Customer Name
not cooked yet		peter

Input of	Format	Description
Types of order form	Pull down menu	Administrator can select a particular cooking status
Customer ID	Number	The order content of the customer will be displayed
Customer Name	Text	The order content of the customer will be displayed



Update Delivery State

Types of Order Form	Customer ID	Customer Name
any		

order form ID	customer name	order time	order status	update status
21	jiang jiawei	2009-04-22 23:12:09	cooking	cooking
20	jiang jiawei	2009-04-22 22:16:53	done	status.
11	Peter	2009-03-25 15:59:30	done	status.
10	Peter	2009-03-25 15:54:45	done	You cannot change the status.
9	Peter	2009-03-25 15:37:29	done	You cannot change the status.
8	Peter	2009-03-25 15:33:30	done	You cannot change the status.
7	dong jingming	2009-03-25 14:05:04	cooking	cooking
1	zhu shucheng	2009-03-08 00:00:00	done	You cannot change the status.
2	dong jingming	2009-03-04 00:00:00	cooking	cooking
3	zhao siqi	2009-03-03 00:00:00	ready to deliver	ready to deliver

Input of	Format	Description
Update Status	Pull down menu	Update the cooking status by selecting the desire option if the status is not 'done'

Control Class

SalesManagementMgr	
Responsibilities	**Collaborators**
Retrieve the sales information	Restaurant sales UI, SalesForm
Retrieves the delivery information	DeliveryUI, SalesForm

Entity Classes

Sales:

This class is used to retrieve the sales information and delivery status. Then display the results according to the user's requests.

PART IV GROUP ORGANIZATION

Activity 3

In activity 3, we need to analyze the 5 most important user cases.

We selected the 5 most important use cases and assigned them to group members Jiang Jiawei, Zhao Siqi and Zhu Shucheng. There are totally four parts in System Analysis and Design Specification.

System Analysis and Design Specification	
Tasks	**Group members**
Analysis Model: Use-Case Realization – Analysis	Zhu Shucheng, Jiang Jiawei
Analysis Model: Class Analysis	Zhu Shucheng, Jiang Jiawei
Design Model: Database Design	Jiang Jiawei
Design Model: Use-case Realization – Design	Zhao Siqi

Group members Dong Jingming and Lu Jingwan discussed and provided their opinions to the other 3 group members.

Activity 4

In activity 4, we need to complete and finalize the system.

We tested the previous system and suggested the improvement that we can perform. Then Dong Jingming and Lu Jingwan took charge of implementation. They fixed the previous bugs. They improved the user interface and added the new functionalities such as ranking and advertisement. Some functionality was improved such as viewing the financial situation.

We all tested the system and provided their suggestions to the 2 group members.

PART V INDIVIDUAL CONTRIBUTION

The following table shows the individual contribution for activity 3 and activity 4, given by Zhu Shucheng, the group leader.

Group member name	% of effort (Activity 3)	% of effort (Activity 4)
Dong Jingming	100	100
Jiang Jiawei	100	100
Lu Jingwan	100	100
Zhao Siqi	100	100
Zhu Shucheng	100	100
Total	500	500