Software Domain Analysis & Design

-Final Phase Document-

「Cafe inventory management system」

Contents

Tables

Figures

<Figure 5.36> Manage Inventory Combined Design Sequence Diagram
<Figure 5.37> Manage Inventory Combined Design Class Diagram

1. Vision

1.1. Revision History

Version	Date	Description	Author
Inception draft	Mar 22, 2017	First draft. To be refined elaboration step.	Team 공조
Elaboration 1	Apr 29, 2017	Elaboration 1. Refined version to the previous version.	Team 공조
Elaboration 2/Final	June 05, 2017	Final draft	Team 공조

1.2. Introduction

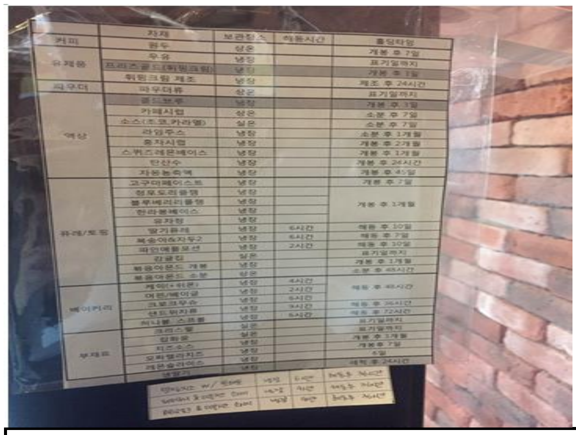
We plan to design the ‘Cafe Inventory Management System’, which manages the inventory quantities and ingredients’ expiration dates, and this app should support the various customer's business rules and interworking with external systems.

1.3. Business Opportunity

Store Managers at many of the cafes have to write the inventory quantities and ingredients’ expiration dates by hand. It takes a lot of time for them to write themselves out manually and to check out the ingredients’ expiration dates and discard them when it’s over. Also, the variety of ingredients that need to be managed is increasing as the menu gradually grows. For this reason, there is a need for a system to manage ingredients automatically.

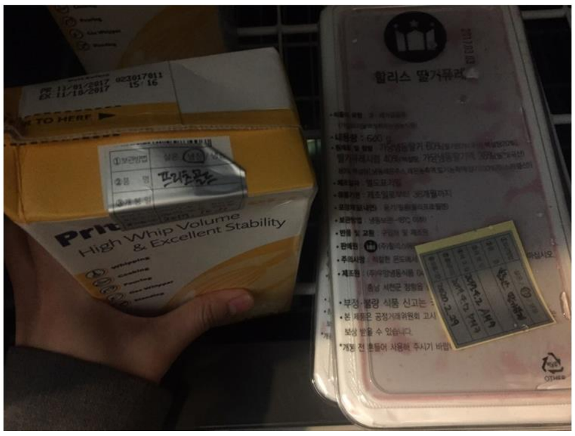
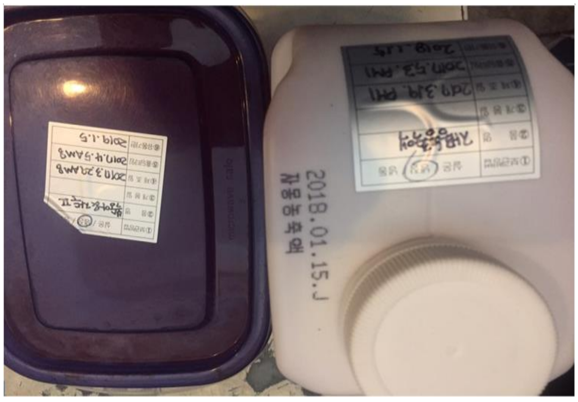
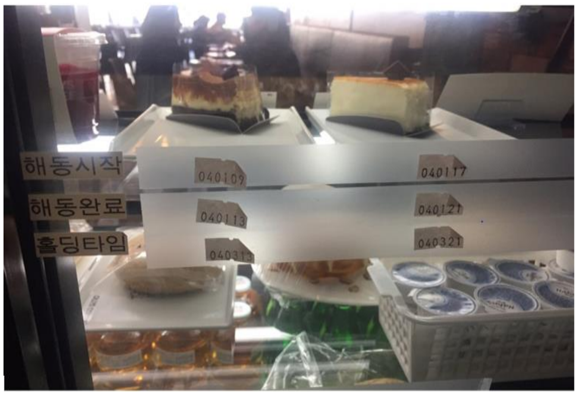
1.4. Problem Statement

Existing cafés, every time when ingredients are arrived, Store Manager writes down the ingredients’ expiration dates and holding time directly or use management book to check and manage the whole ingredients. However, if the Store Manager miss it , he may not be able to use the ingredients that has passed the expiration date, so he may throw away the worst ingredients. In addition, the Store Manager needs to check inventory quantities every day and put the order to the supplier as much as necessary, which is not only cumbersome, but also can cause the situation to be missed and the order can not be placed in time.



커피	자재	보관장소	해동시간	홀딩타임
	원두	상온		개봉 후 7일
유제품	우유	냉장		표기일까지
	프리스골드(휘핑크림)	냉장		개봉 후 3일
	휘핑크림 제조	냉장		제조 후 24시간
파우더	파우더류	상온		표기일까지
액상	콜드브루	냉장		개봉 후 3일
	카페시럽	상온		소분 후 7일
	소스(초코,카라멜)	실온		소분 후 7일
	라임주스	냉장		소분 후 1개월
	홍차시럽	냉장		개봉 후 2개월
	스퀴즈레몬베이스	냉장		개봉 후 1개월
	탄산수	냉장		개봉 후 24시간
	자몽농축액	냉장		개봉 후 45일
퓨레/토핑	고구마페이스트	냉장		개봉 후 7일
	청포도리플잼	냉장		개봉 후 1개월
	블루베리리플잼	냉장		
	한라봉베이스	냉장		
	유자청	냉장		
	딸기퓨레	냉장	6시간	해동 후 10일
	복숭아&자두2	냉장	6시간	해동 후 7일
	파인애플포션	냉장	2시간	해동 후 10일
	감귤칩	실온		표기일까지
	볶음아몬드 개봉	냉장		개봉 후 1개월
	볶음아몬드 소분	상온		소분 후 48시간
베이커리	케익(+쉬폰)	냉장	4시간	해동 후 48시간
	머핀/베이글	냉장	2시간	
	크로크무슈	냉장	6시간	
	샌드위치류	냉장	3시간	해동 후 36시간
	시나몬 스트롤	냉장	6시간	해동 후 72시간
부재료	크리스탈	실온		표기일까지
	[illegible]	실온		표기일까지
	치즈소스	냉장		개봉 후 1개월
	모짜렐라치즈	냉장		개봉후 7일
	레몬슬라이스	냉장		5일
	[illegible]	[illegible]		해동 후 24시간

[illegible] w/ 토핑	냉장	6시간	해동후 24시간
[illegible]	냉장	4시간	해동후 24시간
[illegible]	냉장	[illegible]	해동후 24시간

 Pritt High Whip Volume & Excellent Stability 프리스골드 힐리스 딸기퓨레	
	 해동시작 해동완료 홀딩타임 040109 040117 040113 040121 040321

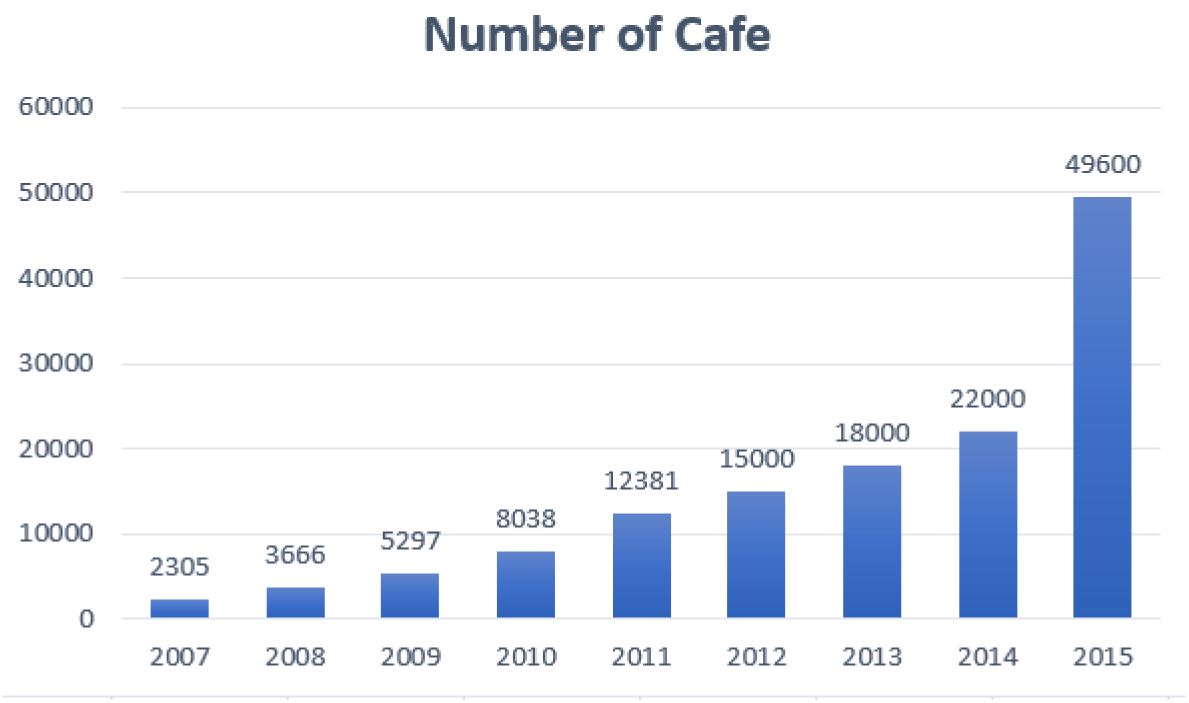
<Figure 1.1> Current cafe Inventory management System

1.5. Product Position Statement

This service will be very useful for Store Managers who want to manage inventory quantities and the ingredients' expiration dates efficiently. The user can check the expiration dates of ingredients and reduce mistakes and errors caused by inventory management.

1.6. Stakeholder Descriptions

1.6.1. Market Demographics



<Figure 1.2> Number of coffee shops according to the year

As shown above, cafes are growing exponentially every year. Additionally , many franchise cafes offer new menus every year or every season.

1.6.2. Stakeholders(non user) Summary

- **customer** : customers who visit the cafe using this system will get better services and drinks made of fresh groceries.
- **cafe brand** : cafe brands such as Starbucks can be certified as a brand of fresh groceries.

1.6.3. User Summary

- **Store Manager** : Manage the inventory quantities and expiration dates of ingredients
- **P.O.S System** : Transfer the information about the menu to be sold to the system.
- **System Administrator** : Register the store that requested the service and manage service maintenance
- **Supplier System** : Get the ordering information sent from the store.
- **Payment System** : Proceed with automatic payment for orders sent from the store.

1.6.4. Key High-Level Goals and Problems of the Stakeholders

High-Level Goal	Priority	Problems and Concerns	Current Solutions
Systematically manage the expiration dates and manage the inventory quantities easily	High	Check the inventory quantity passively and it’s inconvenient	No automatic management
		The ingredients’ expiration dates are checked passively and it’s difficult to find which one is over.	No automatic management
		Inventory quantities and ingredients’ expiration dates can be missed or mistaken.	An effort to check accurately and repeatedly
		Consumers distrust the condition of ingredient management.	Customers assume the ingredients’ condition by only complete food.
Suppliers of ingredients know the amount of deliveries conveniently and accurately	Middle	If the order is not submitted within the specified time, it will be difficult to prepare the delivery.	Direct contact with the store manager
		The ordering format is different for each store, so management is difficult	No solution

<Table 1.1> Key high-level goals and problems of stakeholders

1.6.5. User-Level Goals

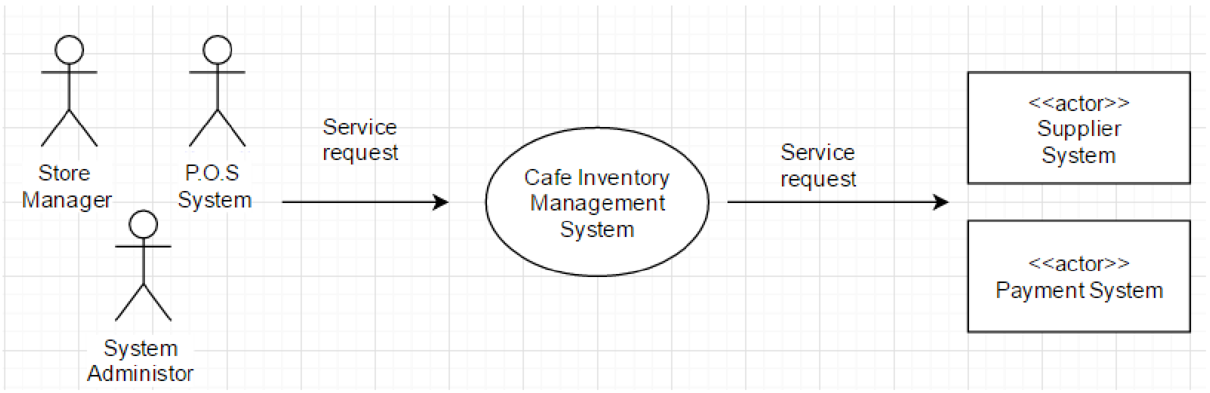
Actor	Goal
Store Manager	Inventory identification, automatic logging, and improved accuracy
	Automatically checks each ingredient expiration date and improves accuracy.
	If no modifications are made, it is automatically ordered
	It is possible to improve the sales to the consumers by forming the trust of ingredients.
P.O.S System	Share newly created menu information
System Administrator	Register or manage the store and maintain the services.

<Table 1.2> User-level goals

1.7. Product Overview

1.7.1. Product Perspective

This cafe inventory management system provides application services that allow Store Managers and ingredient suppliers to easily manage their ingredients. Store Managers can easily and accurately manage the expiration dates of ingredients. Ingredient suppliers can easily manage ingredient status by store. This system can be used not only in cafes but also in restaurants that use food ingredient.



<Figure 1.3> Context Diagram

1.7.2. Summary of Benefits

Supporting Feature	Stakeholder Benefit
Functionally, the system provides services to manage inventory, inventory supply and expiration dates of ingredients.	Automated and rapid inventory management services
Business rules that can connect to branches in various scenarios during inventory management	Building valuable and flexible business logic
Make sure that the ingredients closest to the expiration date is used first, and ingredients that passed the expiration date is not used	It can give consumers a strong sense of trust that they are using fresh food.

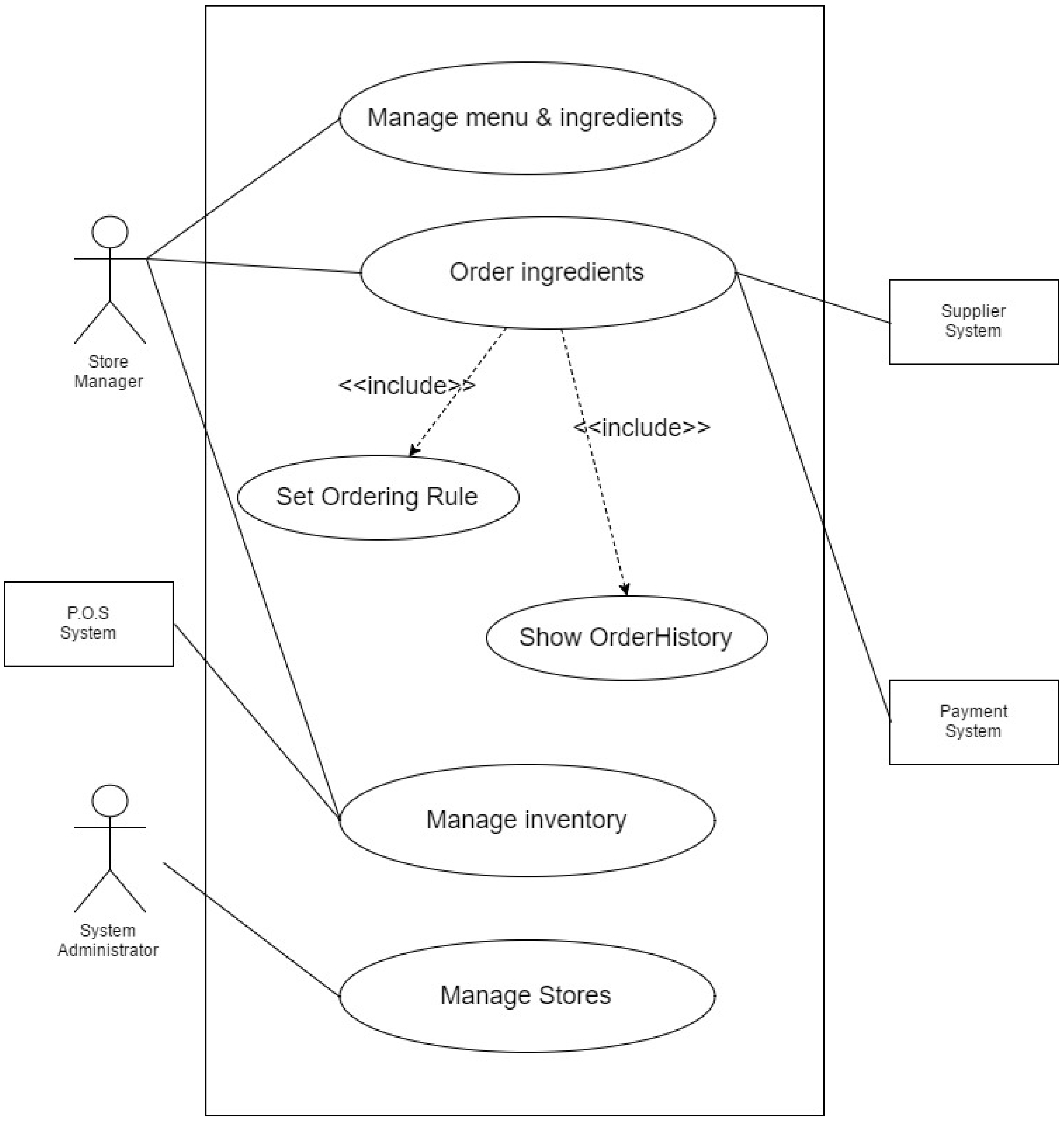
<Table 1.3> Summary of benefits

1.7.3. Summary of System Features

1. Register and modify the information of menu and ingredients
2. Manage the inventory quantity and expiration dates of ingredients
3. Manage the store where request the service.
4. Automatically order to supplier system.
5. Real-time communicates with P.O.S system
6. Alarm when ingredients’ expiration date is over.

2. Requirements

2.1. Use Case Diagram



<Figure 2.1> Use Case Diagram

2.2. Use Case Text

2.2.1. Functional Requirements

1) **Manage menu & ingredients**
The manager selects the Menu Management and the system shows a list of the menu which is currently registered. Manager enters the name of the menu items after registering new menu. The system shows the category of ingredients and manager selects the category which the ingredient is belongs to. The system shows the ingredients belonging to the selected category..The manager selects the required ingredients for the menu item and enters the required amount for the ingredients.After information is entered, the information on the new menu will be registered in the system and can be modified or deleted later.

2) **Order Ingredients**
Managers can place orders manually. When the manager selects ingredient order, the system displays the ingredient category. The manager selects the category of ingredient that requires ordering. When selected, the system displays a list of ingredient belonging to the selected category. The manager specifies the ingredient and quantity required for the order, and the manager requests the order. The system orders the supplier system and displays the completed order to the manager.

2-1) Set Ordering Rule
Managers can set the rules for overall automatic ordering. The manager selects the order management. When the manager starts to register the order day and order time, the system shows the available days and times. If manager select the day and time to order, the system will show the registered day and time of the order. When the specified time of the ordering day is reached, the system orders the supplier system for the ingredients that requires ordering. The amount to be ordered is calculated according to the order calculation rule. Managers can also choose between automatic ordering and manual ordering at any time.

2-2) Show Order History
The manager can confirm the order history at any time. The system displays a list of ordering details. When the manager selects the order history to be confirmed, the system displays information about the order selected by the manager.

3) **Manage inventory**
When the manager selects inventory management for ingredient, the system displays the ingredient category. Manager selects the category in which the manager manages the ingredients. The system displays a list of ingredients in that category and the quantity of each ingredient. When the manager selects the ingredient to be managed, the system displays the information (receiving date, quantity, expiration date) of the ingredient items corresponding to the selected ingredients. The manager adds ingredient items. When the manager inputs the information of the ingredient item (receiving date, quantity, expiration date), the system displays information of the ingredient including the added ingredient item. After the information is inputted, the information of the new ingredient item is registered in the system and it can be corrected or deleted later

4) **Manage store**
The system shows the status of registered stores on the system. The system administrator starts the store management. The system administrator selects the stores to be managed during the store on the system. When the system administrator corrects the information of the selected store, the system stores the modified store status of the store. The system shows the store status after the modification is completed.

2.2.2. Use Case Model

Use Case 1 : Manage menu & ingredients

Scope
- Cafe inventory management System

Level
- User-goal

Primary Actor
- Store Manager

Stakeholders and Interests
- **Store Manager** : Store manager wants information on the correct menus and ingredient items to be well managed..
- **POS System** : POS System wants to get information about the correct menu from the system.

Preconditions
- Store Manager is identified and authenticated.

Postconditions
- New menu is registered.
- New ingredient is registered.
- Ingredient information required to create each menu is registered.
- expiration date, holding time, and minimum holding amount from the receiving date of each ingredient item are registered.

Main Success Scenario
1. The manager selects menu management.
2. The system displays the category list of the currently registered menu.
3. The manager selects the category of the menu to be managed.
4. The system displays menus belonging to the selected category.
5. The manager starts to register the menu.
6. The manager enters the name of the menu item to be newly registered.
7. The system shows the category of the ingredient.
8. The manager selects the category which the ingredient needed for the menu is included.
9. The system shows ingredients belonging to that category.
10. The manager selects the ingredient to add to the menu item.

11. The Manager enters the required amount for the selected ingredient.

The manager repeats steps 7 through 9 for all the ingredients needed for the menu item.

12. The manager finishes inputting the ingredient information that makes the menu.
13. The system displays a list of menus with additional menus.

The manager repeats steps 3 through 12 for all necessary menus.

14. The manager ends menu registration.

Extensions (Alternative Flows)

***a.** At any time, if the system fails

1. The manager restarts the system and requests recovery to the previous state.
2. The system returns to the previous state.
   2a. There are exceptions that interfere with the recovery of the system.
      1. The system displays an error to the manager, records it, and returns to the initial state.
      2. The manager uses the system.

***b**. At any time, if the manager wants to add a new ingredient item.

1. The manager selects to add to the ingredient item.
2. The system shows the ingredient category.
3. The manager selects the category where ingredient to be created is included.
4. The system displays a list of ingredients in the category.
5. The manager enters the name of the ingredient to be added.
   5a.If the manager incorrectly entered the name of the ingredient item to be added.
      1. The manager enters the name of the correct ingredient item.
6. The manager enters the information of the ingredient items to be added (expiration date, holding time, minimum holding amount).
   6a. If the manager incorrectly entered the name of the ingredient item to be added.
      1. The manager enters the correct ingredient item information.
7. The system displays a list of ingredient items, including added ingredient items.

***c**. At any time, if the manager wants to modify the ingredients' information

1. The manager selects to modify the ingredient item.
2. The system shows the ingredient category.
3. The manager selects the category to which the ingredient to be modified belongs.
4. The system displays a list of ingredients in the category.
5. The manager selects the ingredient item to modify.
6. The system displays information on the selected ingredients (expiration date, holding time, minimum holding amount).
7. The manager enters the information of the ingredient to be modified.
8. The system displays a list of modified ingredient items for that category.

***d.** At any time, if the manager wants to delete the ingredient.

1. The manager selects to delete the ingredient.
2. The system shows the ingredient category.
3. The manager selects the category to which the ingredient to be deleted belongs.
   3a. If the ingredient item of the selected ingredient is present.
      1. The system alarms that there is a ingredient item for the selected ingredient item.
      2. The manager cancels the deletion of the ingredient item.
         2a. If the manager wants to delete ingredient items.
            1. The manager requests to delete the ingredient item.
            2. The system deletes the selected ingredient.

3. The system shows the ingredient category.

4. The system displays a list of menus belonging to the category.
5. The manager selects the ingredient item to be deleted.
6. The system displays a list of ingredient items remaining in the category.

3a. If the manager incorrectly selects a category,

1. The manager requests to show the menu category again.
2. The system returns to step 2 to display the menu category.

5a. If the manager wants to modify menu information.

1. The manager starts modifying the menu information.
2. The manager selects the menu to be modified.
3. The system shows the ingredients that make up the selected menu and the amount of each ingredient needed.
4. The manager selects the ingredient to be modified.
   4a.If the ingredient manager wants to modify does not exist in the menu information.
      1. Add ingredient items through the extension scenario * b
5. The manager modifies the required amount of the ingredient.
   5a. If the manager wants to remove the ingredient from the menu component.
      1.The manager requests to delete the selected menu ingredient.
6. The system displays a list of menus with modified menus.

5b. If the manager wants to delete menu information.

1. The manager starts deleting menu information.
2. The system displays a menu list.
3. The manager selects the menu to be deleted.
4. The system displays a list of the remaining menus.

6a. If the manager incorrectly entered the name of the new menu to be added.

1. The manager re-enters the menu name correctly.

6b. If the ingredient of the menu name entered by the manager already exists.

1. The system informs the manager that there is a menu of the same name.
2. The system displays the information of the existing menu of the same name.
3. The manager cancels the addition of ongoing menu items.
   3a. If the manager wants to create a menu item with the same name.
   1. The manager requests to create a menu item of the same name.
   2. The system adds a number to the existing menu name to create a new menu.
   3. Proceed from step 5 of the Main Success scenario.
4. The system displays a menu list.

10a. If the manager incorrectly selects the ingredient item to be added to the menu.

1. The manager selects the ingredient item correctly.

10b. If there is no ingredient item in the menu to be added.

1. Add the ingredient item through the extension scenario * b.

11a. If the manager enters the number of ingredient items incorrectly.

1. Re-enter the number of ingredient items correctly.

Special Requirements

Technology and Data Variations List

Frequency of Occurrence: Often used by users.

Open Issues

- Need to investigate remote service recovery topics.

Use Case 2 : Order Ingredients

Scope

- Cafe inventory management System

Level

- User-goal

Primary Actor

- Store Manager

Stakeholders and Interests

- Store Manager : Store manager can order the ingredients manually at any time.

Preconditions

- Store Manager is identified and authenticated.
- Manager has a list of ingredients

Postconditions

- The order day and the order time are registered.
- Ordering is completed.
- The order is approved.
- Order details are recorded.

Main Success Scenario

1. Manager selects Order Management.
2. The system shows the ingredient category.
3. The manager selects the category of ingredients that need to be ordered.
4. The system displays a list of ingredients belonging to the selected category.
5. The manager enters the ingredients and quantities that need to be ordered.

The manager repeats step 2 to 5 for all the ingredients that need to be placed.

6. The manager requests an order.
7. The system makes an order to the supplier system as the manager decided.
8. The system shows the ordering histories to the manager.

Extensions (Alternative Flows)

***a.** At any time, if the system fails

1. The manager restarts the system and requests recovery to the previous state.
2. The system returns to the previous state.
   2a. It there are exceptions that interfere with the recovery of the system.
      1. The system shows an error to the manager, records it, and returns to the initial state.
      2. The manager uses the system.

3a. 매니저가 재료의 카테고리를 잘못 선택한 경우

1. The manager requests to show the ingredient category again.
2. The system returns to step 2 to display the ingredient category.

5a. 매니저가 재료와 수량을 잘못 입력한 경우

1. The manager re-enters the ingredient and quantity correctly.

7a. If the system fails to communicate with the Supplier System.

1. The system restarts and continues the Supplier terminal and communication service.
   1a. .if the system detects that the service can not be restarted.
   1. The manager grasps the menus sold and directly calculates the ingredients that make up the menu to modify the inventory quantity of that ingredient.

Special Requirements

- When accessing remote services such as service failures in the Supplier system, they want to be restored correctly.

Technology and Data Variations List

Frequency of Occurrence: Often used by users.

Open Issues

- Need to investigate remote service recovery topics**.**

Use Case 2-1 : Set OrderingRule

Scope
- Cafe inventory management System

Level
- Subfunction

Primary Actor
- Store Manager

Stakeholders and Interests
- Store Manager : Even if the manager does not place an order, the system can calculate the amount of ingredient needed and set the day and time for the automatic ordering to proceed.

Preconditions
- Store Manager is identified and authenticated.

Postconditions
- The order day and the order time are registered.
- Ordering is completed.
- The order is approved.
- Order details are recorded.
- The order status is specified.

Main Success Scenario
1. Manager selects Order Management.
2. The manager begins registering the date of the order and ordering time on the order day.
3. The system shows the selectable days and hours of the week.
4. The manager selects the day and time of day for which the manager wants to be placed.
5. The system shows the date and time which are registered.

Extensions (Alternative Flows)
***a.** At any time, if the system fails
1. The manager restarts the system and requests recovery to the previous state.
2. The system returns to the previous state.
   2a. There are exceptions that interfere with the recovery of the system.
      1. System displays an error to the manager, records it, and returns to the initial state.
      2. The manager uses the system.

***b.** At any time, if the manager wants to toggle the automatic ordering feature on or off
1. Manager selects Order Management.
2. The manager chooses to change the automatic ordering status.
3. The system shows the existing automatic ordering status.
4. The manager changes the automatic ordering status as desired.
5. The system displays the changed automatic order status.

2a. if order date and order time are already registered.

1. The system shows the date and time of the order on which the order is currently registered.
2. If the manager wants to modify the order day or order time.
   2a. Proceed from the main scenario 1.
3. If the manager does not want to modify the order of the order day or order time
   3a. Proceed from the main scenario 7.

4a. If the manager incorrectly selects the ordering date and ordering time.

1. The manager requests to show the available day and time.
2. The system shows selectable days and times.
3. The manager will re-select the desired ordering date and time.

Special Requirements

Technology and Data Variations List

Frequency of Occurrence: Often used by users

Open Issues

Use Case 2-2 : Show Order History

Scope
- Cafe inventory management System

Level
- Subfunction

Primary Actor
- Store Manager

Stakeholders and Interests
- Store Manager : You can check the order history (order date and time, order quantity for each ingredients) for all orders that were previously automatically or manually ordered.

Preconditions
- The order history for all orders previously or automatically or manually placed is stored.

Postconditions

Main Success Scenario
1. The manager selects order management.
2. The manager selects to check order history.
3. The system displays a list of ordering details.
4. The manager selects the order history to be checked out of the order history list.
5. The system displays information about the order history selected by the manager (order date, order time, list of ordered ingredients, and quantity of each ingredient).

Extensions (Alternative Flows)
3a. If the manager chooses the wrong order,
1. The manager requests the system to show the order history list again.
2. The system returns to step 2.

Special Requirements

Technology and Data Variations List

Frequency of Occurrence: Often used by users

Open Issues

Use Case 3 : Manage inventory

Scope
- Cafe inventory management System

Level
- User-goal

Primary Actor
- Store Manager

Stakeholders and Interests
- Store Manager : Store Manager wants to reduce the hassle of finding the whole product to check inventory quantity and expiration date.

Preconditions
- Store Manager is identified and authenticated.

Postconditions
- The quantity and expiration date of ingredients of the system are updated.
- The amount of inventory in the system matches the current inventory.

Main Success Scenario
1. The manager selects inventory management for the ingredient.
2. The system shows the ingredient category.
3. The manager selects the category with the ingredient to manage.
4. The system displays a list of ingredients in that category and the quantity of each ingredient.
5. The manager selects the ingredient to be managed.
6. The system shows the information (receiving date, quantity, expiration date) of the ingredient items corresponding to the selected ingredient.
7. The manager adds ingredient items.
8. The manager enters the information of the new ingredient item (receiving date, quantity, expiration date).
9. The system displays ingredient information, including added ingredient items.

The manager repeats steps 2 through 9 for all ingredients received.

10. The manager ends inventory management.

Extensions (Alternative Flows)

***a.** At any time, if the system fails
1. The manager restarts the system and requests recovery to the previous state.
2. The system returns to the previous state.
   2a. There are exceptions that interfere with the recovery of the system.
    1. The system displays an error to the manager, records it, and returns to the initial state.
    2. The manager uses the system.

***b.** At any time, if the expiration date of ingredients remains within 3 days
1. The system informs to the Store Manager that there are x days left until the expiration date of ingredients.

***c.** At anytime, if the expiration date is over
1. The system informs the Store Manager that the expiration date of the ingredient has

passed the current date.

1a. If the same ingredient is not removed after Step 1 has been carried out.

1. System Informs the manager how long the expiration date of the ingredient has passed.

1c. If the menu is sold in POS system

1. The linked system (P.O.S. System) sends the sold menu to the system.
2. The system identifies the ingredients that make up the menu that is transmitted.
3. The system modifies the stock quantity of the ingredient.

   3a. If there is less amount of ingredient that needs to be cut down than amount of ingredient that you want to modify..

   1. The system informs that there is insufficient ingredient to modify.

1c-1a. If the system fails to communicate with P.O.S System

1. The system restarts the POS register and communication services and continues.

   1a.The system detects that the service can not be restarted.

   1. The system displays an error.
   2. The manager grasps the menus sold and directly calculates the ingredients that make up the menu to modify the inventory quantity of that ingredient.

5a. If the manager incorrectly selects the ingredient

1. The manager requests the system to show a list of ingredients belonging to a previously selected category and the quantity of each ingredient
2. Start again from main scenario 4.

5b. If the ingredient item you want to select is not in the category

1. Perform Use Case 1 extension * b scenario to add new ingredient item.

7a. If the manager wants to modify ingredient item information

1. The manager selects the ingredient item to modify.
2. The manager modifies the ingredient item information to match the actual inventory.
3. The system displays information about the ingredients including the modified ingredient.

7b. If the manager wants to delete a ingredient item.

1. The manager asks for a deletion of the ingredient items.
2. The system shows the remaining ingredient items.

8a. If the manager has not entered ingredient quantity information

1. The system informs the manager that the ingredient quantity has not been entered.
2. The manager goes back to Step 8 of the main scenario.

8b.The manager has not entered a receiving date.

1. The system informs the manager that it has not entered a receiving date.
2. The system enters the current date as the receiving date.
3. The manager goes back to Step 9 of the main scenario.

Special Requirements

- If you access a remote service such as P.O.S System service failure, you want to be recovered properly.

Technology and Data Variations List

Frequency of Occurrence: Often used by users

Open Issues

- Need to investigate remote service recovery topics.

Use Case 4 : Manage store

Scope
- Cafe inventory management System

Level
- User-goal

Primary Actor
- Store Manager

Stakeholders and Interests

Preconditions
- Store Manager is identified and authenticated.

Postconditions
- Information about the store is changed.

Main Success Scenario
1. The system administrator starts the store management.
2. The system shows the status of registered stores on the system.
3. The system administrator selects the stores to be managed during the store on the system.
4. The system administrator modifies the information of the selected store.

The system administrator repeats steps 2 to 4 for all stores to be modified.

5. The system administrator asks the system to reflect the modified store status.
6. The system shows the store status after the modification is completed.

Extensions (Alternative Flows)
***a.** At any time, if the system fails
1. The manager restarts the system and requests recovery to the previous state.
2. The system returns to the previous state.
   2a. There are exceptions that interfere with the recovery of the system.
3. The system displays an error to the manager, records it, and returns to the initial state.
4. The manager uses the system.

3a. An administrator chooses the wrong store to manage
1. .The manager requests to show the registered store status again.
2. The system returns to step 2 and shows the store status.

3b. If the manager does not have the desired store
1. The administrator chooses to add to the new store.
2. The administrator enters information about the store to be added.
   2a. The administrator mistyped information about the store to add.
      1. The manager enters the information of the correct store.
3. The system shows the store status including the added stores.

4a. If the administrator wants to delete information from the selected store
1. The manager requests to delete the store.

2. The system shows the status of the remaining stores.

Special Requirements

Technology and Data Variations List

Frequency of Occurrence: Often used by users

Open Issues

2.2.3. Non-functional Requirements

1) Revision History

Version	Date	Description	Author
Inception draft	Mar 22, 2017	First draft. To be refined elaboration step.	Team 공조
Elaboration 1	Apr 29, 2017	Elaboration 1. Refined version to the previous version.	Team 공조
Elaboration 2 / Final	June 05, 2017	Final draft	Team 공조

2) Introduction
Requirements of this system which are not expressed in the use cases are treated in this document.

3) Usability Requirement

@ Help
- If user has difficulty in using this service, user can get support from the system administrator. System administrator provides users with manuals, system developer's e-mail addresses, and FAQs.
- It Provides basic sample data for menus and ingredients for initial use for your convenience.
- If the system is not running and P.O.S is used, the system will run automatically.

4) Reliability Requirement

@ Recoverability
- System will be checked 2 times a month for 30 minutes each. (02:00AM ~ 02:30AM).
- System will be backed up during system check.

@ Security
- If user tries to authenticate or get data from the database , System check the value of the data.
- Security is the capability of a system to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information. A secure system aims to protect assets and prevent unauthorized modification of information

5) Performance Requirement
@ Response Time (Latency)
- Updating inventory or ingredient information is done within 5 seconds 90% of time.
- View existing inventory or ingredient information and order history is done within 3 seconds 90% of time.

6) Supportability Requirement
@ Adaptability
- Users use many different operating systems.
- So Cafe inventory management system works well with any kinds of operating system.

 - It also works with any kinds of P.O.S system.
Therefore, the cafe inventory management system service works well on all operating systems.

7) Privacy Requirement
-Sales, menu, and order information for each store and supplier can not be viewed by the manager of the store or the manager of the supplier.

8) Portability Requirement
-Store managers can use the SIS system anywhere in the store to check inventory and order status

9) Profit Structure
 - Make a profit by receiving the monthly fee from the cafe company and the ingredient supplier who want to use this service.

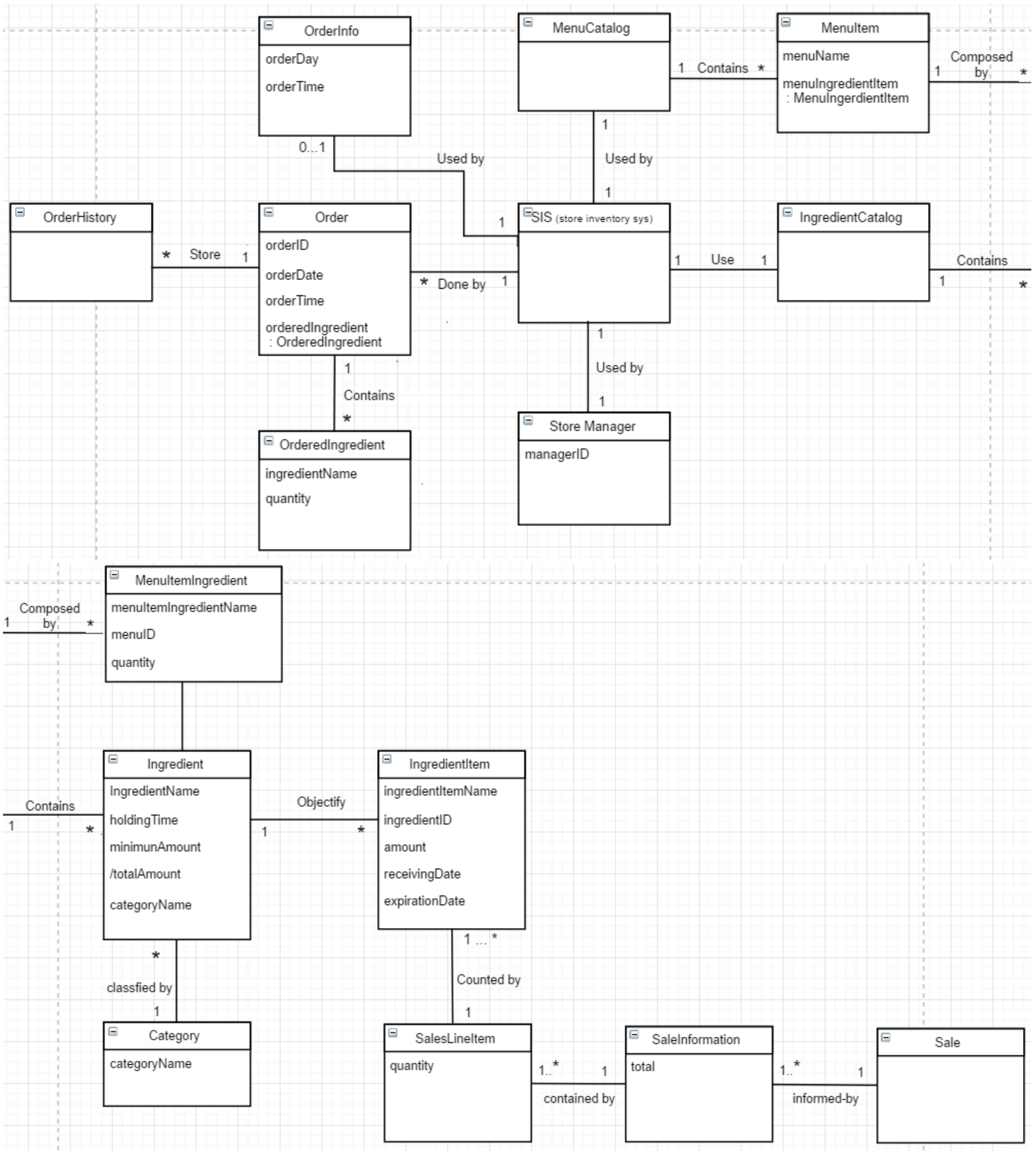
3. Domain Model

3.1. Revision History

Version	Date	Description	Author
Inception draft	Mar 22, 2017	First draft. To be refined elaboration step.	Team 공조
Elaboration 1	Apr 29, 2017	Elaboration 1. Refined version to the previous version.	Team 공조
Elaboration 2 / Final	June 05, 2017	Final draft	Team 공조

3.2. Domain Model Diagram

(Single Picture is difficult to see. So we distribute 2 parts below.)



<Figure 3.1> Domain Model Diagram

3.2.1. Domain Classes and Attributes

1) StoreManager

class Description : Manager who uses a system for managing stores in stores

Attribute	Description
managerID	manager’s ID
Association	**Description**
Store Manager(1) - SIS(1)	Store Manager uses SIS

<Table 3.1>Description of Domain class ‘StoreManager’

2) SIS (Store inventory system)

class Description : Application that Store Manager uses on its device

Attribute	Description
-	-
Association	**Description**
SIS(1) - IngredientCatalog(1)	SIS uses IngredientCatalog
SIS(1) - MenuCatalog(1)	SIS uses MenuCatalog
SIS(1) - Store Manager(1)	SIS is used by StoreManager
SIS(1) - AutoOrderRule(0...1)	AutoOrderRule is defined in SIS
SIS(1) - Order(*)	SIS proceeds Order
SIS(1) - Order(1)	SIS records OrderHistory
SIS(1) - Store(0….*)	SIS manages Store
SIS(1) - Administrator(1)	SIS is used by Administrator

<Table 3.2>Description of Domain class ‘SIS’’

3) NewIngredientItem

class Description : a new ingredient item to add

Attribute	Description
ingredientitemID	ID of ingredient item to add
ingredientID	ID or ingredient which ingredient item belongs to
ingredientItemName	name of ingredient

amount	quantity of ingredient
receivingDate	date which the ingredient entered in the store
expirationDate	date which the ingredient can be used.
Association	Description
NewIngredientItem(1) - IngredientItem(1)	NewIngredientItem becomes IngredientItem

<Table 3.3>Description of Domain class ‘NewIngredientItem’

4) IngredientItem

class Description : Ingredient items per each received date

Attribute	Description
ingredientitemID	ID of ingredient item
ingredientID	unique ID of ingredient
ingredientItemName	name of ingredient
amount	quantity of ingredient
receivingDate	date which the ingredient entered in the store
expirationDate	date which the ingredient can be used.
Association	Description
IngredientItem(1) - NewIngredientItem(1)	NewIngredientItem becomes IngredientItem
IngredientItem(*) - Ingredient(1)	Ingredient Objectifies IngredientItem
IngredientItem(1…*) - SalesLineItem	IngredientItem is counted by SalesLineItem

<Table 3.4> Description of Domain class ‘IngredientItem’

5) Ingredient

class Description : Ingredient to use in stores

Attribute	Description
ingredientID	unique ID of ingredient
ingredientName	name of ingredient
holdingTime	period for maintaining ingredients

minimumAmount	minimum quantity of the ingredient during the operating hour
totalAmount	sum of the all the ingredient items which belongs to same ingredient
category	category where the ingredients belongs.
Association	Description
Ingredient(1) - MenuItemIngredient(1)	MenuItemIngredient represents Ingredient
Ingredient(1) - IngredientItem(1...*)	Ingredient Objectifies IngredientItem
Ingredient(*) - IngredientCatalog(1)	IngredientCatalog contains Ingredient

<Table 3.5> Description of Domain class ‘Ingredient’

6) ingredientCatalog

class Description : list of ingredients

Attribute	Description
-	-
Association	Description
IngredientCatalog(1) - Ingredient(*)	IngredientCatalog contains Ingredient
IngredientCatalog(1) - SIS(1)	SIS uses IngredientCatalog.

<Table 3.6>Description of Domain class ‘ingredientCatalog’

7) MenuItemIngredient

class Description : Ingredients needed to create the corresponding menus

Attribute	Description
menuID	id of menu
ingredientID	unique id of ingredient
IngredientName	name of ingredient
quantity	quantity of the ingredient to make the menu
Association	Description
MenuItemIngredient(*) - MenuItem(1)	MenuItem is composed of MenuItem
MenuItemIngredient(1) - Ingredient(1)	MenuItemIngredient represents Ingredient

<Table 3.7> Description of Domain class ‘MenuItemIngredient’

8) NewMenuItem

class Description : a new menu item to add

Attribute	Description
menuID	id of menu
menuName	name of menu
menuCategory	Categories to which this menu belongs
menuIngredientItem	ingredients needed to create the menu
Association	**Description**
NewMenuItem(1) - MenuItem(1)	NewMenuItem becomes MenuItem

<Table 3.8> Description of Domain class ‘NewMenuItem’

9) MenuItem

class Description : the menu which is sold in the store.

Attribute	Description
menuID	ID of menu
menuName	name of menu
menuCategory	Categories to which this menu belongs
menuIngredientItem	ingredients needed to create the menu
Association	**Description**
MenuItem(1) - NewMenuItem(1)	NewMenuItem becomes MenuItem
MenuItem (*) - MenuCatalog(1)	MenuCatalog contains MenuItem.
MenuItem(1) - MenuItemIngredient(*)	Menuitem is composed of MenuItemIngredient

<Table 3.9> Description of Domain class ‘MenuItem’

10) MenuCatalog

class Description : list of menu

Attribute	Description

-	-
Association	Description
MenuCatalog(1) - SIS(1)	SIS uses MenuCatalog
MenuCatalog(1) - MenuItem(*)	MenuCatalog contains MenuItem

<Table 3.10> Description of Domain class ‘MenuCatalog’

11) AutoOrderRule

class Description : information needed to order automatically which includes ordering time and ordering day.

Attribute	Description
autoOrderingDay	the day for automatic ordering
autoOrderingTime	the time for automatic ordering
autoOrderFlag	on/off flag which determine whether to order automatically
Association	Description
AutoOrderRule(0...1) - SIS(1)	AutoOrderRule is defined in SIS

<Table 3.11> Description of Domain class ‘AutoOrderRule’

12) OrderHistory

class Description : A class containing list of the order history which has been already ordered.

Attribute	Description
-	-
Association	Description
OrderHistory(1) - SIS(1)	OrderHistory is recored by SIS.
OrderHistory(1) - Order(0…*)	OrderHistory includes Orders.

<Table 3.12>Description of Domain class ‘OrderHistory’

13) Order

class Description : information of an order which has been already ordered.

Attribute	Description
orderID	ID of an order.

orderDate	date the order is sent.
orderTime	time the order is sent.
orderedIngredient	ingredients which has been ordered.
supplierSystemAdapter	adapter used to interact with supplier system.
Association	Description
Order(1) - OrderedIngredient(*)	Order is composed of OrderedIngredient
Order(0…*) - OrderHistory(1)	OrderHistory includes Orders.

<Table 3.13> Description of Domain class ‘Order’

14) OrderedIngredient

class Description : ordered ingredients Information

Attribute	Description
ingredientName	name of ordered ingredient
quantity	Amount of ordered ingredient
Association	Description
OrderedIngredient(*) - Order(1)	Order is composed of OrderedIngredient

<Table 3.14> Description of Domain class ‘OrderIngredient’

15) SalesLineItem

class Description : Sales amount of each menu

Attribute	Description
quantity	quantity of menu which is sold.
Association	Description
SalesLineItem(1) - IngredientItem	IngredientItem is counted by SalesLineItem
SalesLineItem(1…*) - SaleInformation(1)	SaleInformation contains SalesLineitem

<Table 3.15>Description of Domain class ‘SalesLineItem’

16) SaleInformation

class Description : total quantity sold at P.O.S

Attribute	Description

total	total items which are sold.
Association	Description
SaleInformation(1) - SalesLineItem(1…*)	SaleInformation contains SalesLineitem
SaleInformation(1…*) - Sale(1)	SaleInformations are informed by Sale.

<Table 3.16>Description of Domain class ‘SaleInformation’

17) Sale

class Description : sales detail in on sale.

Attribute	Description
-	-
Association	Description
Sale(1) - SaleInformation(1…*)	SaleInformations are informed by Sale.

<Table 3.17> Description of Domain class ‘Sale’

18) Store

class Description : the store which uses Cafe Inventory Management System

Attribute	Description
storeName	name of store
storeID	unique id of the store
address	address of store
phoneNumber	phone number which can contact with the store
Association	Description
Store(0…*) - SIS (1)	Store is managed by SIS

<Table 3.18>Description of Domain class ‘Store’

19) Administrator

class Description : sales detail in on sale.

Attribute	Description
administratorID	id of administrator
Association	Description

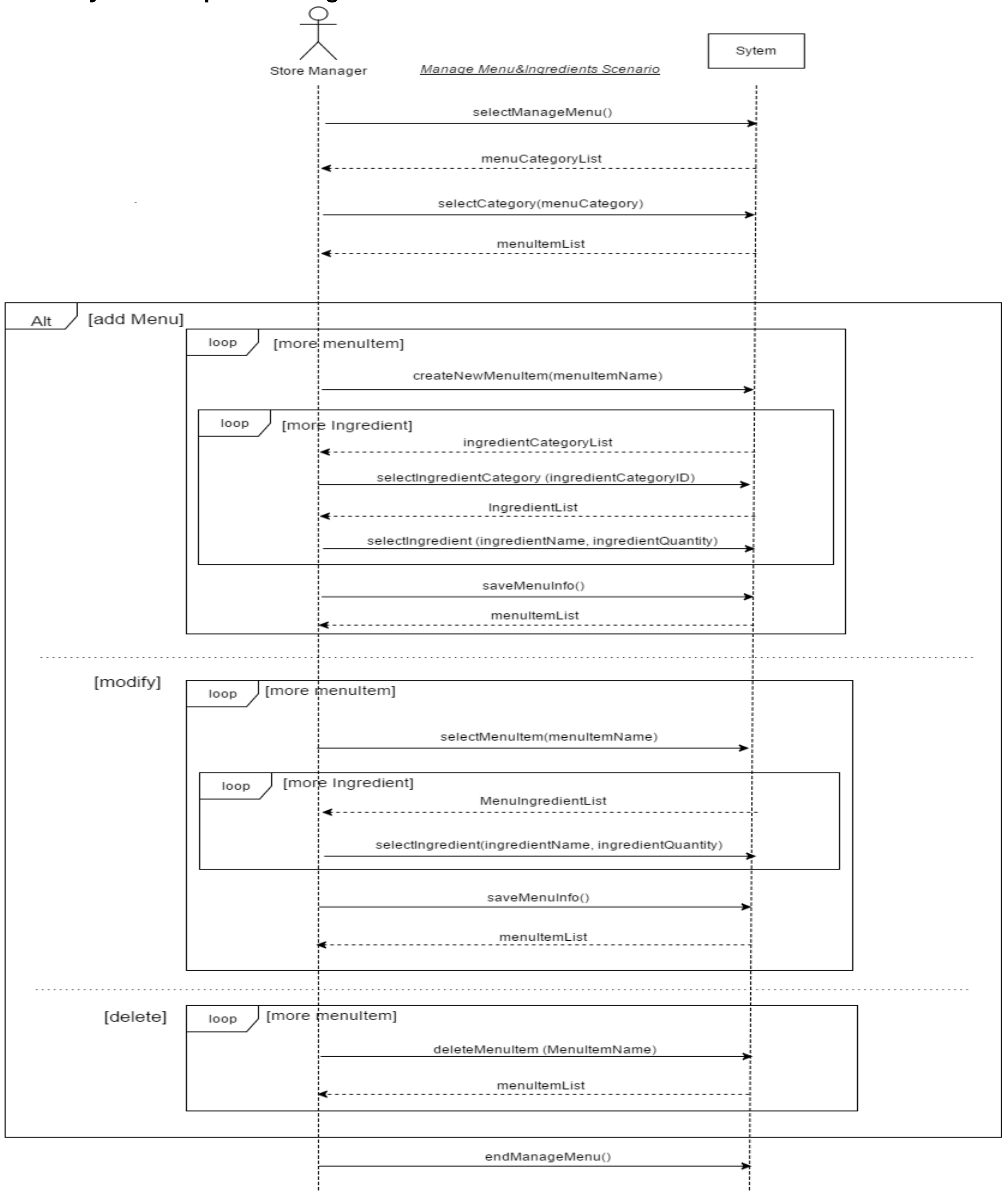
Administrator(1) - SIS(1)	Administrator uses SIS

<Table 3.19> Description of Domain class Administrator’

4. System Sequence Diagram

4.1. Manage menu & ingredients

4.1.1. System Sequence Diagram



<Figure 4.1> Manage Menu & Ingredients System Sequence Diagram

System Operation 1 : selectManageMenu()

- The store manager selects the menu management.

System Operation 2 : createNewMenuItem(menuItemName)

- The store manager adds a new menu item.
- The store manager enters the name of the menu item.

System Operation 3 : selectIngredientCategoey(category)

- The store manager selects the category of ingredient to manage when creating the menu item.

System Operation 4 : selectIngredient(ingredientName , ingredientQuantity)

- The store manager selects the ingredients to manage within the category.
- The store manager selects the quantity of the ingredient.

System Operation 5 : saveMenuInfo()

- The store manager selects the quantity of the ingredient.

System Operation 6 : endNewMenuItem()

- The store manager terminates the menu management.

System Operation 7 : selectMenuItem(menuItemName)

- The store manager selects the menu item.

System Operation 8 : deleteMenuItem(menuItemName)

- The store manager deletes the menu.

4.1.2. Operation Contracts

Operation	**selectManageMenu()**
Cross references	Manage Menu & Ingredients
Precondition	- Store Manager is identified and authenticated - Instance of menuCatalog mc is created. - mc has all information about menuitem. - Instance of menuCatalog, mc was created. - mc had all the information about all menuitems.
Postcondition	-

<Table 4.1> selectManageMenu Operation Contracts

Operation	**createNewMenuItem(menuItemName)**
Cross references	Manage Menu & Ingredients
Precondition	- Menu registration is in progress. - Instances mi of all menuItems are created. - All mi is associated with menuCatalog.
Postcondition	- Instance of newMenuItem nmi is created. - Attributes of nmi is initiated. - New menuItemID value is created and value of nmi.menuItemID is changed to newly created menuItemID. - nmi.menuName is changed to menuItemName.

<Table 4.2> createNewMenuItem Operation Contracts

Operation	**selectIngredientCategory(menuItemID)**
Cross references	Manage Menu & Ingredients
Precondition	- Menu registration of in progress. - Instances i of all ingredients are created. - Each ingredient i has information about ingredient.
Postcondition	

<Table 4.3> selectIngredientCategory Operation Contracts

Operation	**selectIngredient(ingredientName, ingredientQuantity)**
Cross references	Manage Menu & Ingredients
Precondition	- Menu registration is in progress.
Postcondition	- Instance of menuIngredientItem, mi is created. - Attribute value of mii was initialized. - nmi was associated to mii.

	- Value of mii.name was changed to ingredientName. - Value of mii.quantity was changed to ingredientQuantity.

<Table 4.4> selectIngredient Operation Contracts

Operation	**saveMenuInfo()**
Cross references	Manage Menu & Ingredients
Precondition	- Menu registration is in progress.
Postcondition	- nmi became mi.

<Table 4.5> saveMenuInfo Operation Contracts

Operation	**selectMenuItem(menuItemID)**
Cross references	Manage Menu & Ingredients
Precondition	- Instances mi of all menuItems are created.
Postcondition	

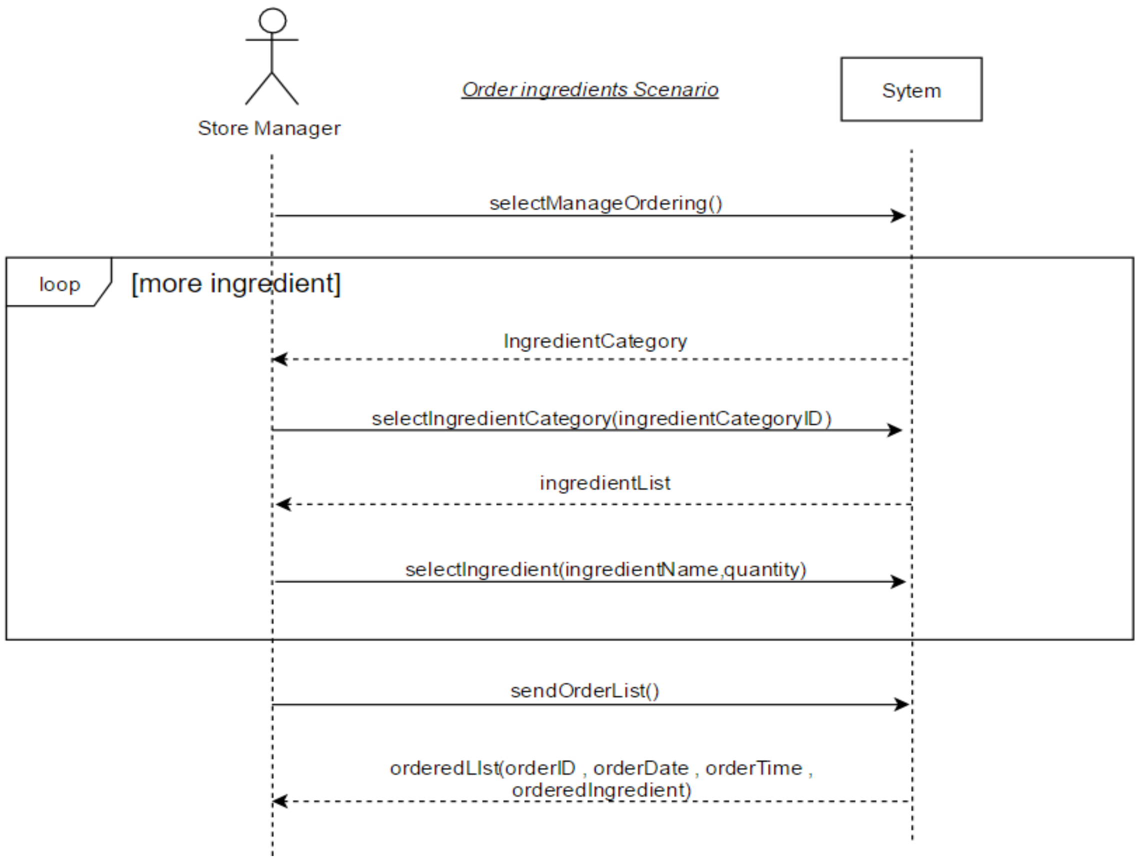
<Table 4.6> selectMenuItem Operation Contracts

Operation	**deleteMenuItem(menuItemID)**
Cross references	Manage Menu & Ingredients
Precondition	- Menu deletion is in process.
Postcondition	- Instance mi is deleted which has a value of menuItemID.

<Table 4.7> deleteMenuItem Operation Contracts

4.2. Order Ingredient

4.2.1. System Sequence Diagram


Store Manager
Order ingredients Scenario
Sytem
selectManageOrdering()
loop
[more ingredient]
IngredientCategory
selectIngredientCategory(ingredientCategoryID)
ingredientList
selectIngredient(ingredientName,quantity)
sendOrderList()
orderedLIst(orderID , orderDate , orderTime , orderedIngredient)

<Figure 4.2> Order Ingredient System Sequence Diagram

System Operation 1 : SelectManageOrdering()

- The store manager selects the order management.

System Operation 2 : selectIngredientCategory(ingredientCategoryID)

- The store manager selects the category which the ingredient to be ordered is belong to.

System Operation 3 :selectIngredient(ingredientName, quantity)

- The store manager selects the quantity and Name of the ingredient to order.

System Operation 4 :sendOrderList()

- The store manager requests the system to order.

4.2.2. Operation Contracts

Operation	**selectManageOrdering()**
Cross references	Order Ingredients
Precondition	- Store Manager is identified and authenticated. - Instance of order o is create.
Postcondition	- Ingredient ordering was started.

<Table 4.8> selctMangeOrdering Operation Contracts

Operation	**selectIngredientCategory(ingredientCategoryID)**
Cross references	Order Ingredients
Precondition	- Ingredient ordering is in process.
Postcondition	- An orderIngredient instance oi was created. - The attributes of oi have been initialized.

<Table 4.9> selectIngredientCategory Operation Contracts

Operation	**selectIngredient(ingredientName, quantity)**
Cross references	Order Ingredients
Precondition	- Ingredient ordering is in process.
Postcondition	- oi.ingredientName has become ingredientName.. - oi.quantity has become quantity.

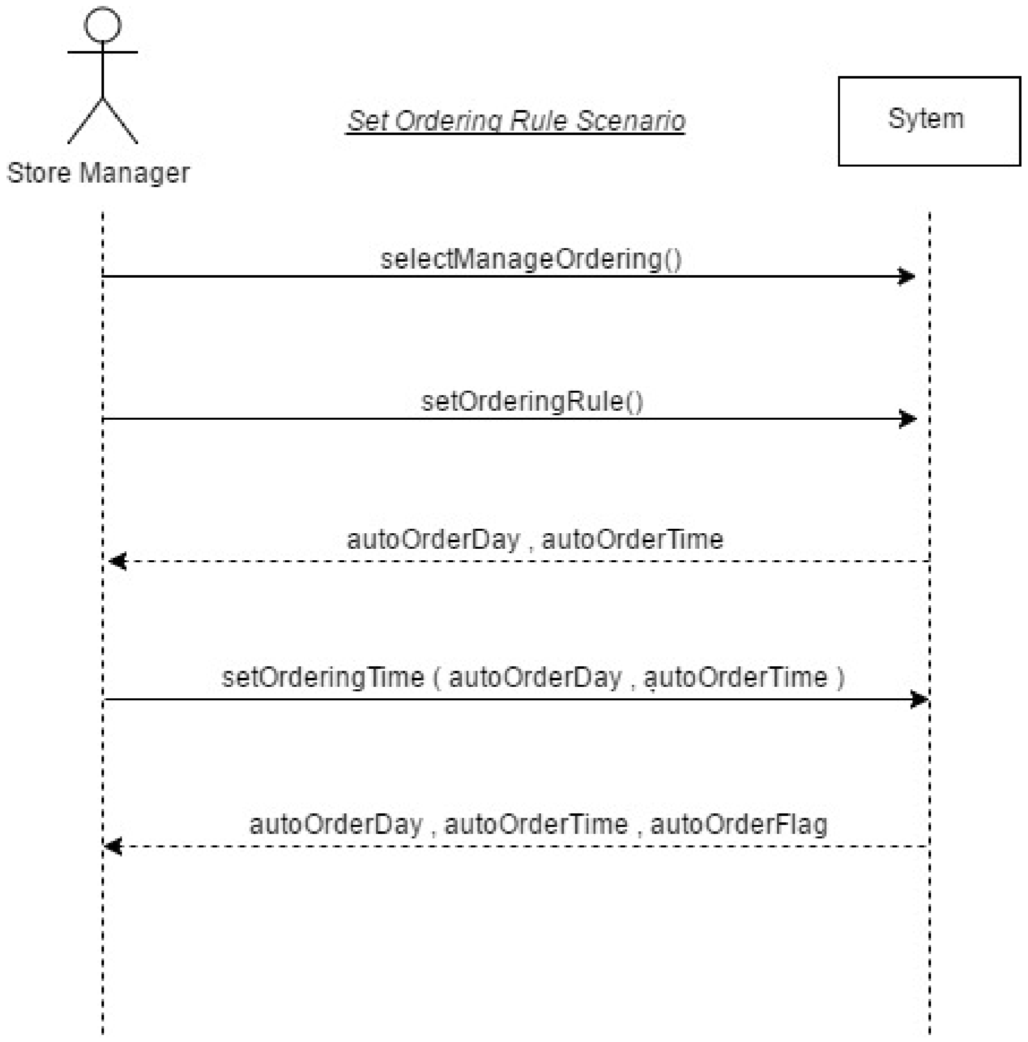
<Table 4.10> selectIngredient Operation Contracts

Operation	**sendOrderList()**
Cross references	Order Ingredients
Precondition	- Ingredient ordering is in process.
ostcondition	

<Table 4.11> sendOrderList Operation Contracts

4.3. Set Order Info

4.3.1. System Sequence Diagram


Store Manager
Set Ordering Rule Scenario
Sytem
selectManageOrdering()
setOrderingRule()
autoOrderDay , autoOrderTime
setOrderingTime (autoOrderDay , autoOrderTime)
autoOrderDay , autoOrderTime , autoOrderFlag

<Figure 4.3> Set Order Info System Sequence Diagram

System Operation 1 : SelectManageOrdering()

- The store manager selects the order management.

System Operation 2 : setOrderingRule()

- The store manager requests the system to register the ordering day and time.

System Operation 3 : setOrderingTime(orderingDay , orderingTime)

- The store manager selects the day and time of the order.

4.3.2. Operation Contracts

Operation	**selectSetOrderInfo()**
Cross references	Set OrderRule
Precondition	- Store Manager is identified and authenticated.
Postcondition	- The setting of orderInfo started.

<Table 4.12> selectSetOrderInfo Operation Contracts

Operation	**setOrderingRule()**
Cross references	Set OrderRule
Precondition	- The orderInfo setting is in progress.
Postcondition	- The orderInfo instance oi was created. - The attribute value of oi was initialized.

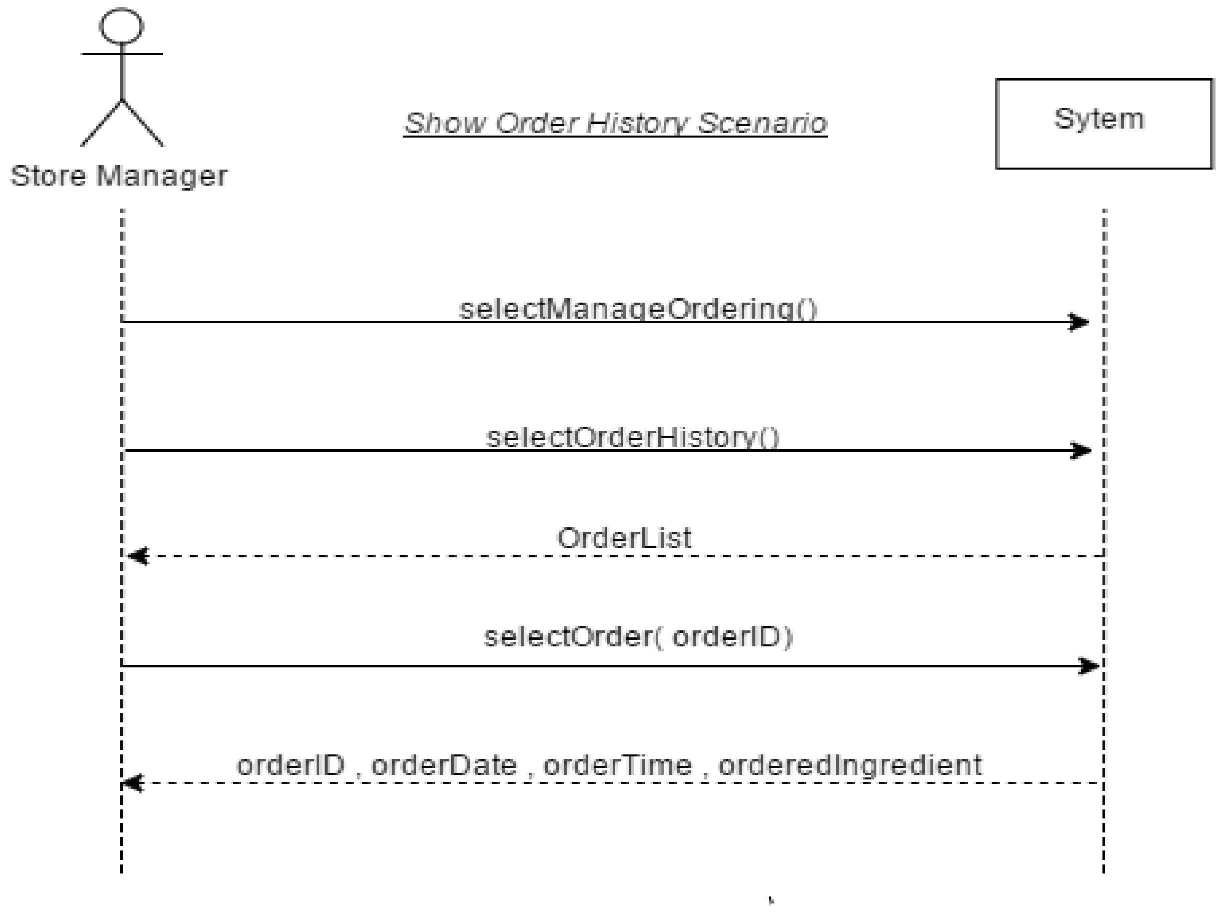
<Table 4.13> setOrderingRule Operation Contracts

Operation	**setOrderingTime(autoOrderDay, autoOrderTime)**
Cross references	Set OrderRule
Precondition	- The orderInfo setting is in progress.
Postcondition	- The oi.autoOrderDay value was changed to autoOrderDay. - oi.autoOrderTime value changed to autoOrderTime.

<Table 4.14> setOrderingTime Operation Contracts

4.4. Show Order History

4.4.1. System Sequence Diagram



<Figure 4.4> Show Order History System Sequence Diagram

System Operation 1 : selectManageOrdering()

- The store manager selects the order management.

System Operation 2 : selectOrderHistory()

- The store manager chooses to manage the order history.

System Operation 3 : selectOrder(orderID)

- The store manager selects the order history to be confirmed.

4.4.2. Operation Contracts

Operation	**selectManageOrdering()**
Cross references	Show Order History
Precondition	- Store manager is identified and authenticated.
Postcondition	-

<Table 4.15> selectManageOrdering Operation Contracts

Operation	**selectOrdeHistory()**
Cross references	Show Order History
Precondition	- The order instance o has information about all orders, including manual and automatic ordering. - Checking order history is in progress.
Postcondition	-

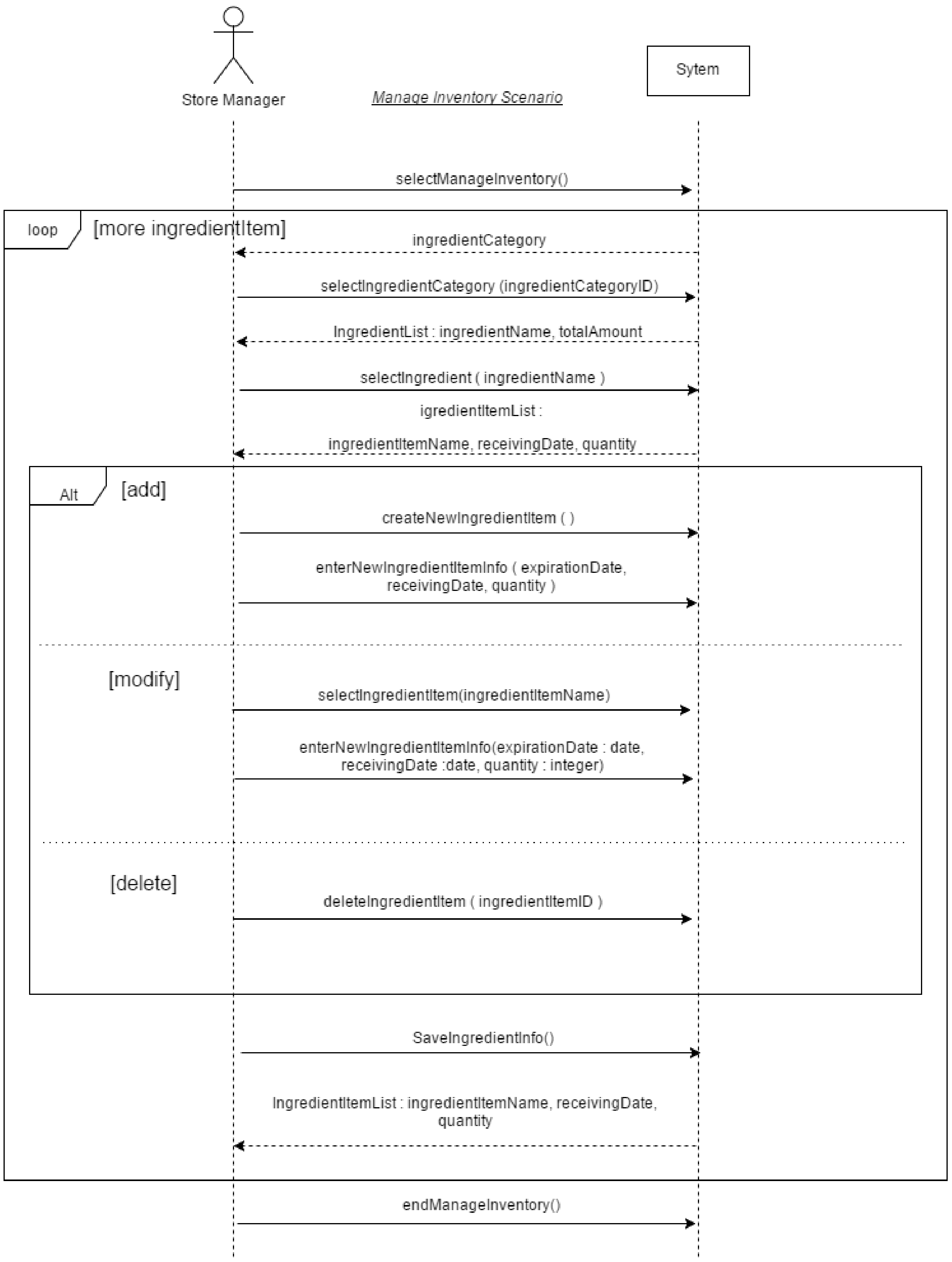
<Table 4.16> selectOrderHistory Operation Contracts

Operation	**selectOrder(orderID)**
Cross references	Show Order History
Precondition	- Checking order history is in progress.
Postcondition	-

<Table 4.17> selectOrder Operation Contracts

4.5. Manage inventory

4.5.1. System Sequence Diagram


Store Manager
Manage Inventory Scenario
Sytem
selectManageInventory()
loop
[more ingredientItem]
ingredientCategory
selectIngredientCategory (ingredientCategoryID)
IngredientList : ingredientName, totalAmount
selectIngredient (ingredientName)
igredientItemList :
ingredientItemName, receivingDate, quantity
Alt
[add]
createNewIngredientItem ( )
enterNewIngredientItemInfo (expirationDate, receivingDate, quantity)
[modify]
selectIngredientItem(ingredientItemName)
enterNewIngredientItemInfo(expirationDate : date, receivingDate :date, quantity : integer)
[delete]
deleteIngredientItem (ingredientItemID)
SaveIngredientInfo()
IngredientItemList : ingredientItemName, receivingDate, quantity
endManageInventory()

<Figure 4.5> Manage Inventory System Sequence Diagram

System Operation 1 : selectManageInventory()

- The store manager selects inventory management.

System Operation 2 : selectIngredientCategory (categoryName)

- The store manager selects the category to which the ingredient is to be managed.

System Operation 3 : selectIngredient(ingredientName)

- The store manager selects the ingredient to manage.

System Operation 4 : createNewIngredientItem()

- The store manager requests adding ingredient items to the selected ingredient.

System Operation 5 : enterNewIngredientItemInfo(expirationDate, receivingDate, quantity)

- The store manager enters the expiration date, warehousing date, and quantity of the selected ingredient.

System Operation 6 : endManageInventory()

- The store manager terminates the ingredient management.

System Operation 7 : selectIngredientItem(ingredientItemName)

- The store manager selects the ingredient item to modify.

System Operation 8 : deleteIngredientItem(ingredientItemName)

- The store manager requests deletion of the selected ingredients.

4.5.2. Operation Contracts

Operation	**selectManageInventory()**
Cross references	Manage Inventory
Precondition	- Store Manager is identified and authenticated. - The ingredientCatalog instance ic has been created. - ic has information about all ingredeints.
Postcondition	

<Table 4.18> selectmanageInventory Operation Contracts

Operation	**selectIngredientCategory (ingredientCategoryID)**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress.
Postcondition	

<Table 4.19> selectIngredientCategory Operation Contracts

Operation	**selectIngredient (ingredientName)**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress. - Instances i of all ingredients are created. - The ingredient instance has all inventory information for that ingredient. - Every ingredient instance i is associated with an ingredientCatalog instance.
Postcondition	

<Table 4.20> selectIngredient Operation Contracts

Operation	**createNewIngredientItem()**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress. - Instances ii of all ingredientItem s are created. - Every ii is associated with that ingredient instance.
Postcondition	- newIngredientItem instance nii was created. - The attribute value of nii was initialized. - A new ingredientItemID value was created and the nii.ingredientItemID value was changed to the newly created ingredientItemID value.

<Table 4.21> creatNewIngredientItem Operation Contracts

Operation	**enterNewIngredientItemInfo (expirationDate , receivingDate , quantity)**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress. - newIngredientItem instance nii was created. - The attribute value of nii has been initialized.
Postcondition	- The attribute value of nii.expirationDate changed to expirationDate. - The attribute value of nii.receivingDate changed to receivingDate. - The attribute value of nii.quantity changed to quantity.

<Table 4.22> enterNewIngredientItemInfo Operation Contracts

Operation	**saveNewIngredientItem()**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress. - An IngrdientItem instance ii has information about each existing ingredientItem.
Posondition	- newIngredientItem instance nii became ii.

<Table 4.23> saveNewIngredientItem Operation Contracts

Operation	**selectIngredientItem(ingredientItemID)**
Cross references	Manage Ingredient
Precondition	- Inventory management is in progress. - Instances ii of all ingredientItem s are created.
Postcondition	

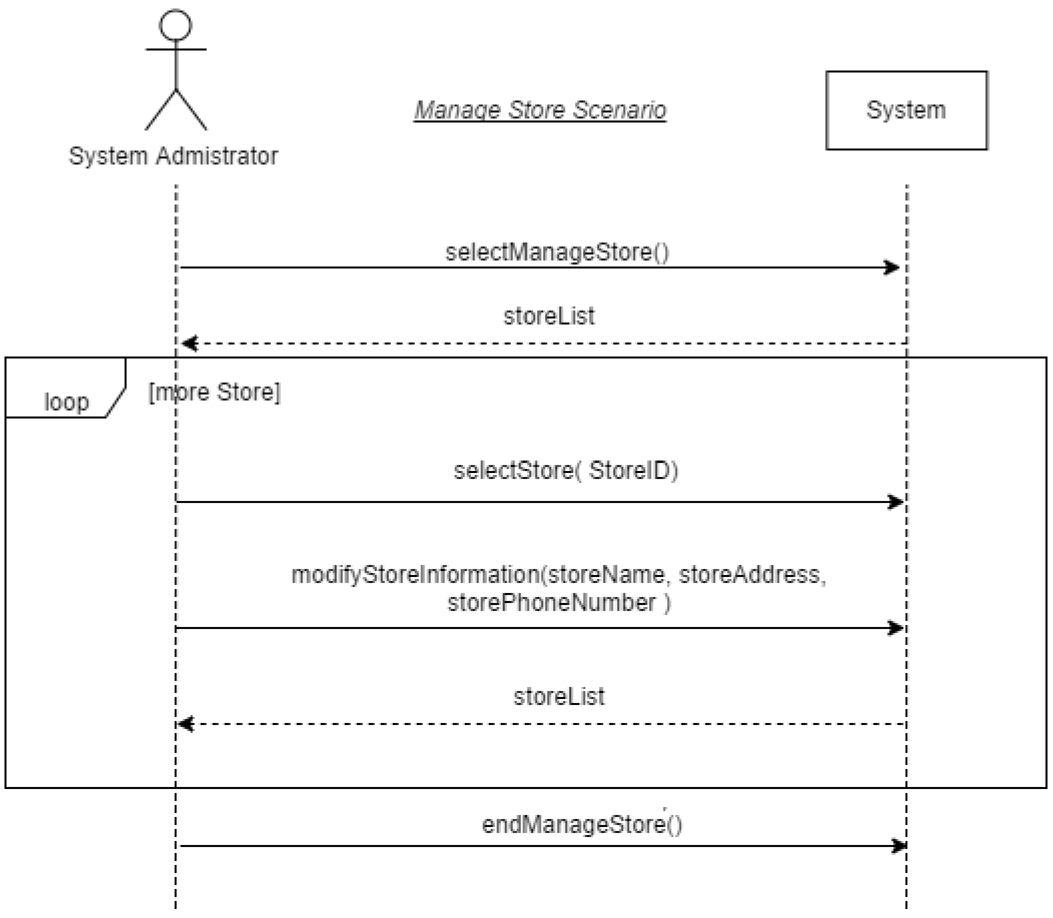
<Table 4.24> selectIngredientItem Operation Contracts

Operation	**deleteIngredientItem(ingredientItemID)**
Cross references	Manage ngredient
Precondition	- Inventory management is in progress. - An IngrdientItem instance ii has information about each existing ingredientItem.
Postcondition	- An instance ii having an ingredientItemID as an attribute value of the instance ii was deleted.

<Table 4.25> deletIngredientItem Operation Contracts

4.6. Manage Stores

4.6.1. System Sequence Diagram



<Figure 4.6> Manage Stores System Sequence Diagram

System Operation 1 : selectManageStore()

- The system administrator selects store management.

System Operation 2 : selectStore(storeID)

- The system administrator selects the stores to be managed among the stores on the system.

System Operation 3 : modifyStoreInformation(storeName,storeAddress,storePhoneNumber)

- The system administrator corrects the information of the selected store.

System Operation 4 : endManageStore()

- The system administrator ends the store management.

4.6.2. Operation Contracts

Operation	**selectManageStore()**
Cross references	Manage Stores
Precondition	- System administrator is identified and authenticated
Postcondition	- Store management was started.

<Table 4.26> selectMangeStore Operation Contracts

Operation	**selectStore(storeID)**
Cross references	Manage Stores
Precondition	- Store management is in progress.
Postcondition	- Instances ii of all stores was created. - s had all the information about the store.

<Table 4.27> selectStore Operation Contracts

Operation	**modifyStoreInformation(storeName,storeAddress,storePhoneNumber)**
Cross references	Manage Stores
Precondition	- Store management is in progress.
Postcondition	- The attribute value of s.storeName changed to storeName. - The attribute value of s.storeAddress changed to storeAddress. - The attribute value of s.storePhoneNumber changed to storePhoneNumber.

<Table 4.28> modifyStoreInformation Operation Contracts

Operation	**endManageStore()**
Cross references	Manage Stores
Precondition	- Store management is in progress.
Postcondition	

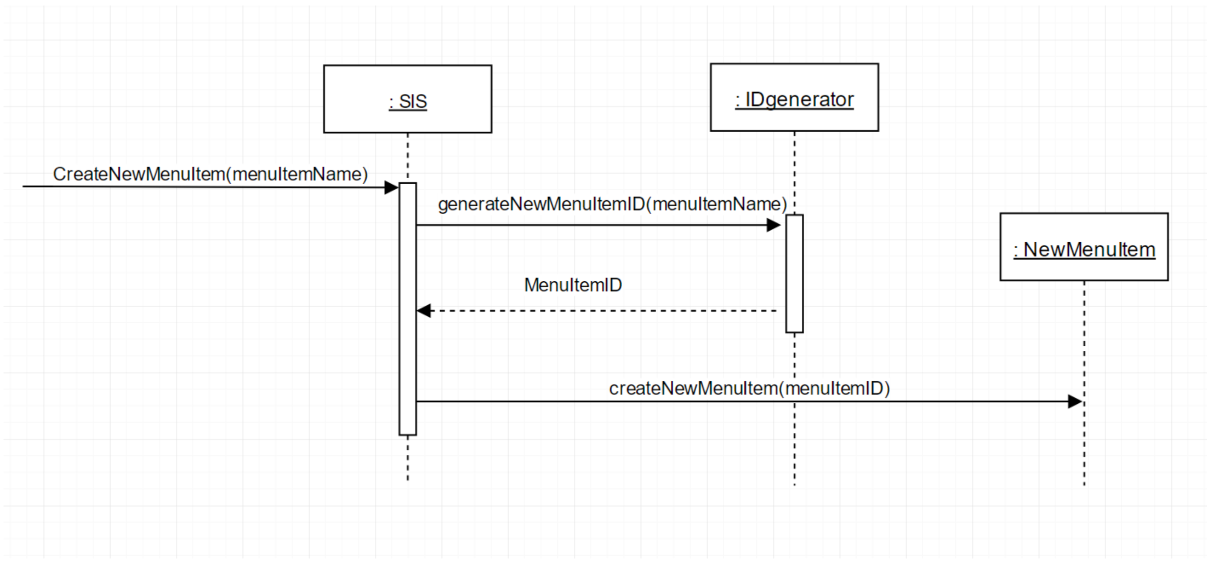
<Table 4.29> endManageStore Operation Contracts

5. Design Model

5.1. Manage menu & ingredients Realization

5.1.1. Design Sequence Diagram

1) createNewMenuItem(menuItemName)



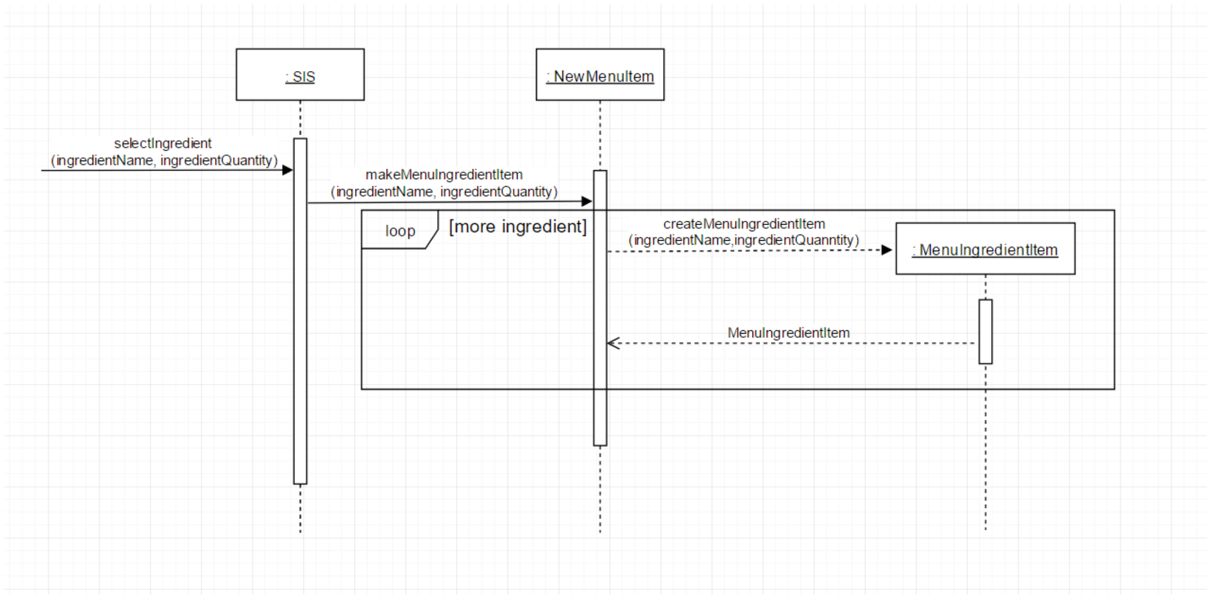
: SIS
: IDgenerator
: NewMenuItem
CreateNewMenuItem(menuItemName)
generateNewMenuItemID(menuItemName)
MenuItemID
createNewMenuItem(menuItemID)

<Figure 5.1> createNewMenuItem Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	- The menu catalog class has information about all menu items, but SIS directly creates a newMenuItem object to reduce coupling.
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.1> createManageMenu DSD GRASP Pattern

2) selectIngredient(ingredientName, ingredientQuantity)

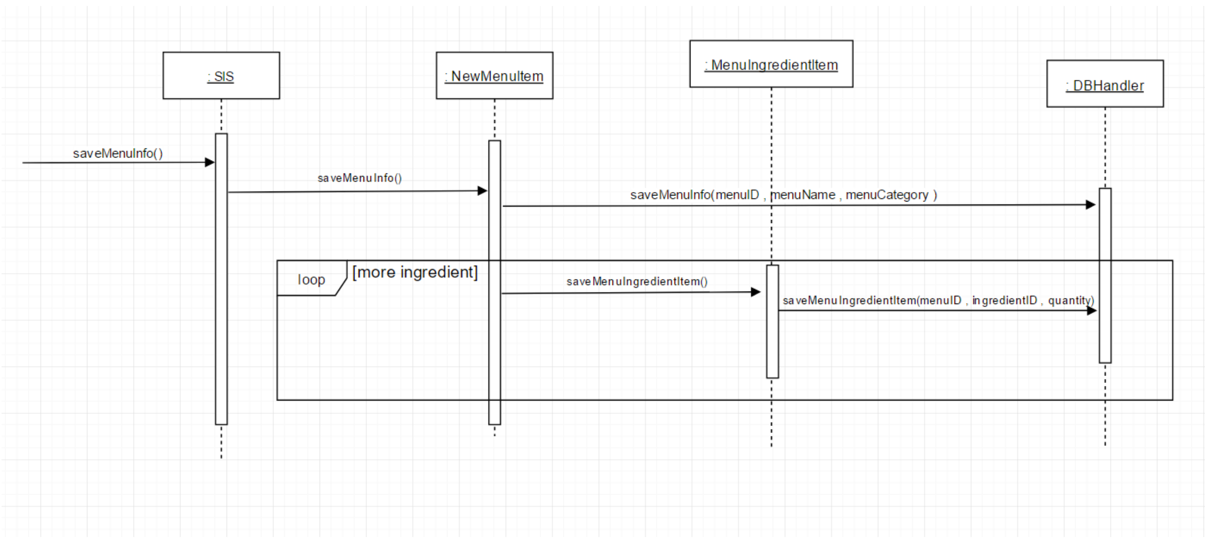

: SIS
: NewMenuItem
selectIngredient (ingredientName, ingredientQuantity)
makeMenuIngredientItem (ingredientName, ingredientQuantity)
loop
[more ingredient]
createMenuIngredientItem (ingredientName,ingredientQuanntity)
: MenuIngredientItem
MenuIngredientItem

<Figure 5.2> selectIngredient Design Sequence Diagram

GRASP Pattern	Description
Information Expert	-Since the MenuIngredientItem is the information contained in the newly created MenuItem, the NewMenuItem object creates a MenuIngredientItem object.
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system.

<Table 5.2> selectIngredient DSD GRASP Pattern

3) saveMenuInfo()

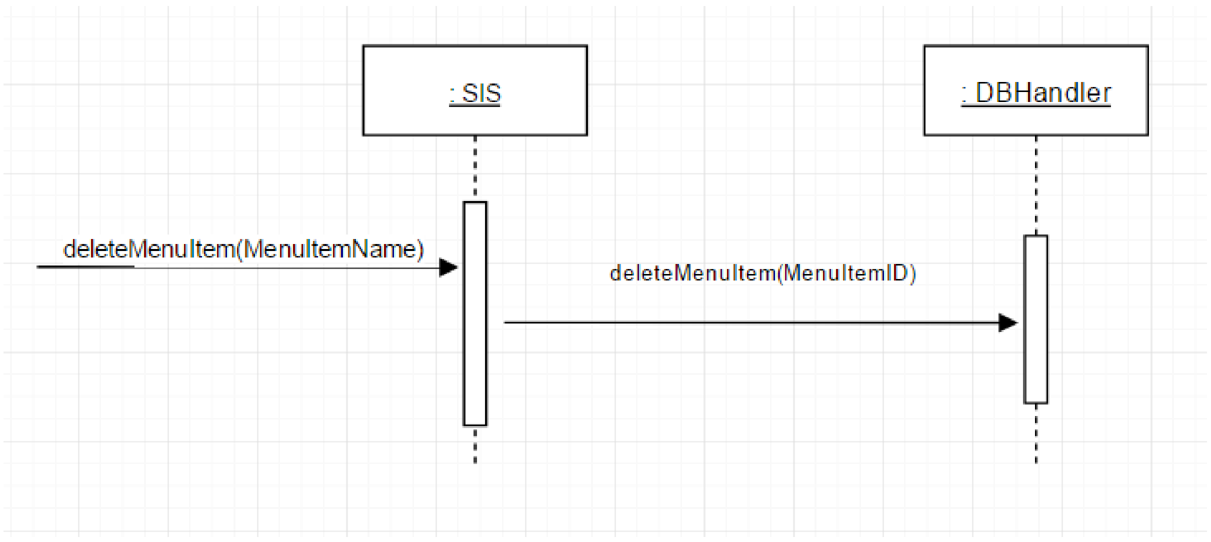


<Figure 5.3> saveMenuInfo Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	-Reduce coupling by showing MenuInfo which means ‘saved’, directly through SIS rather than through a newMenuItem object.
Controller	-SIS is a facade Controller. SIS is ‘root object’ for overall system.
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.3> saveMenuInfo DSD GRASP Pattern

4) deleteMenuItem(MenuItemName)



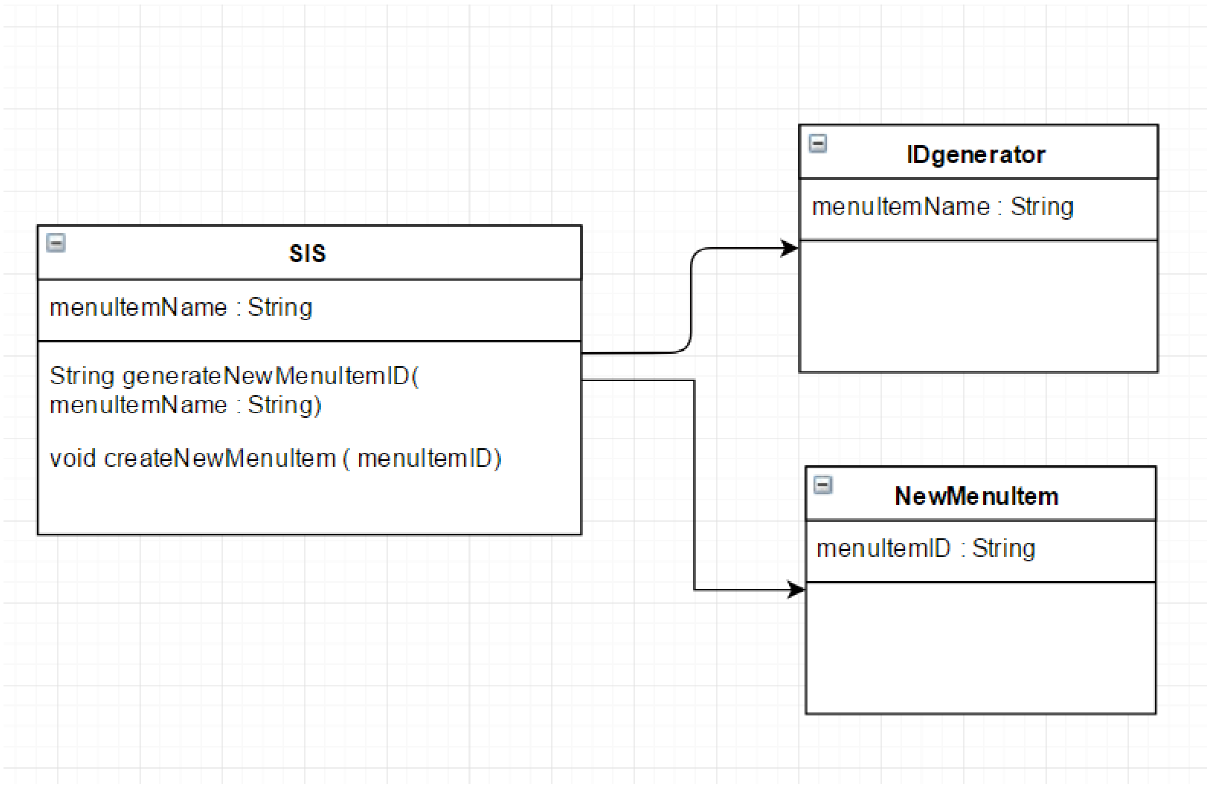
<Figure 5.4> deleteMenuItem Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	-When deleting a MenuItem, rather than associating it with a MenuItem object that matches the MenuItemId you want to delete, you simply pass the MenuItemId you want to delete to the DBHandler and delete it from the DB, so reduce coupling.
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system

<Table 5.4> deleteMenuItem DSD GRASP Pattern

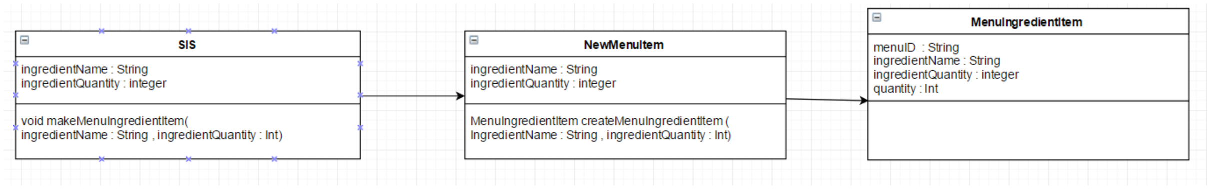
5.1.2. Design Class Diagram

1) createNewMenuItem(menuItemName)


SIS
menuItemName : String
String generateNewMenuItemID(menuItemName : String)
void createNewMenuItem (menuItemID)
IDgenerator
menuItemName : String
NewMenuItem
menuItemID : String

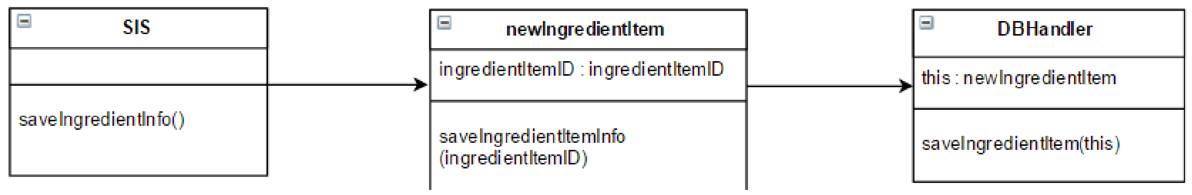
<Figure 5.5> createNewMenuItem Design Class Diagram

2) selectIngredient(ingredientName, ingredientQuantity)


SIS
ingredientName : String
ingredientQuantity : integer
void makeMenuIngredientItem(ingredientName : String , ingredientQuantity : Int)
NewMenuItem
ingredientName : String
ingredientQuantity : integer
MenuIngredientItem createMenuIngredientItem (IngredientName : String , ingredientQuantity : Int)
MenuIngredientItem
menuID : String
ingredientName : String
ingredientQuantity : integer
quantity : Int

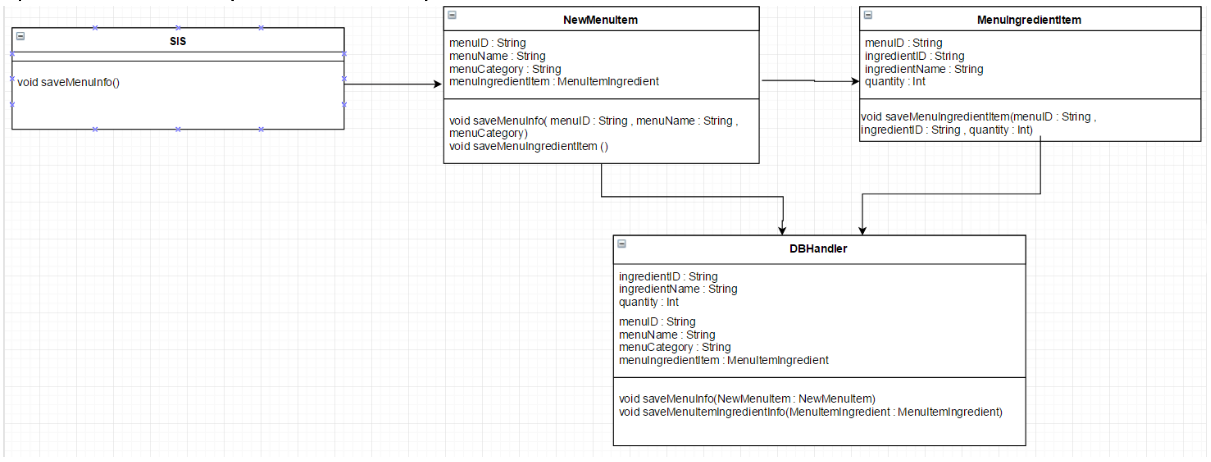
<Figure 5.6> selectIngredient Design Class Diagram

3) saveMenuInfo()



<Figure 5.7> saveMenuInfo Design Class Diagram

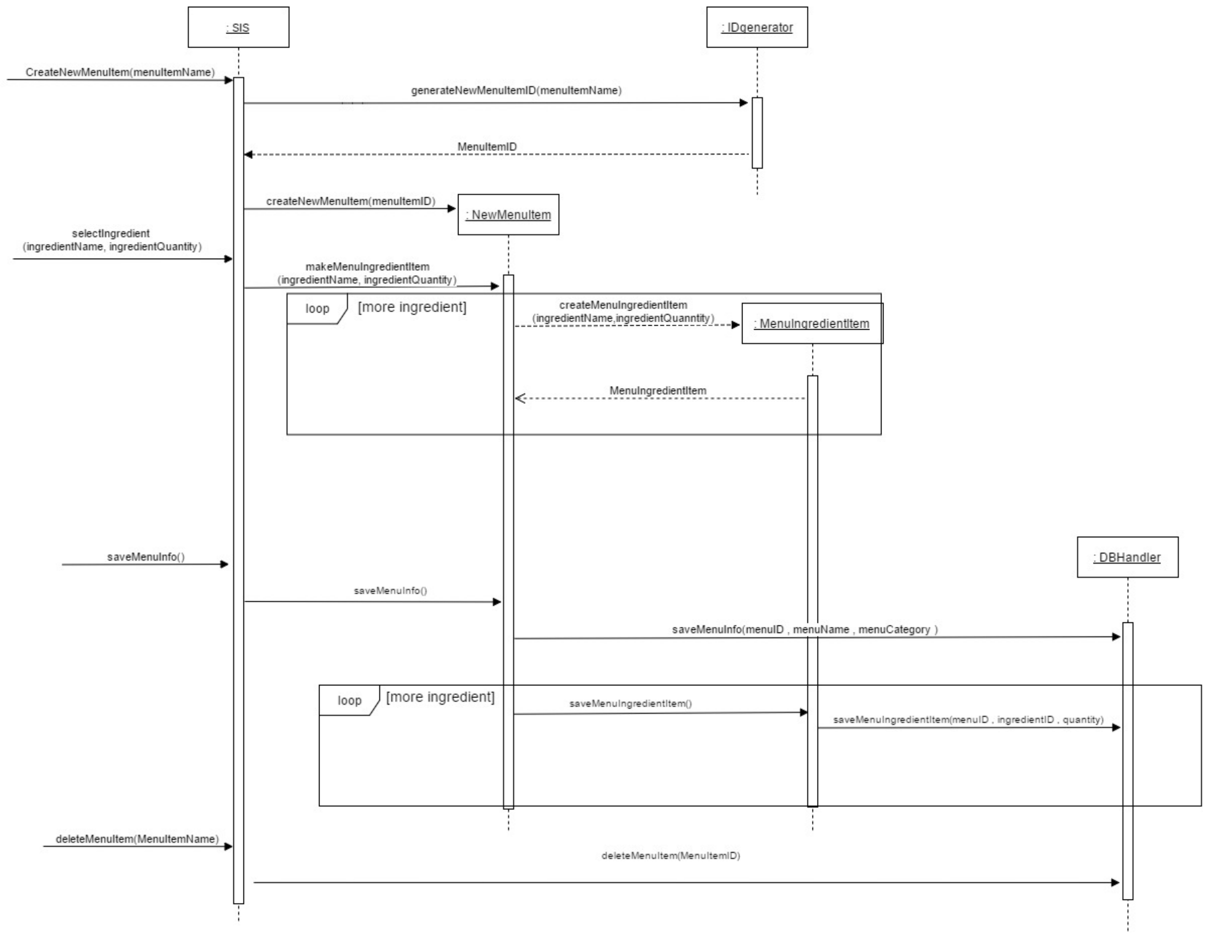
4) deleteMenuItem(MenuItemName)


SIS
void saveMenuInfo()
NewMenuItem
menuID : String
menuName : String
menuCategory : String
menuIngredientItem : MenuItemIngredient
void saveMenuInfo(menuID : String , menuName : String , menuCategory)
void saveMenuIngredientItem ()
MenuIngredientItem
menuID : String
ingredientID : String
ingredientName : String
quantity : Int
void saveMenuIngredientItem(menuID : String , ingredientID : String , quantity : Int)
DBHandler
ingredientID : String
ingredientName : String
quantity : Int
menuID : String
menuName : String
menuCategory : String
menuIngredientItem : MenuItemIngredient
void saveMenuInfo(NewMenuItem : NewMenuItem)
void saveMenuItemIngredientInfo(MenuItemIngredient : MenuItemIngredient)

<Figure 5.8> deleteMenuItem Design Class Diagram

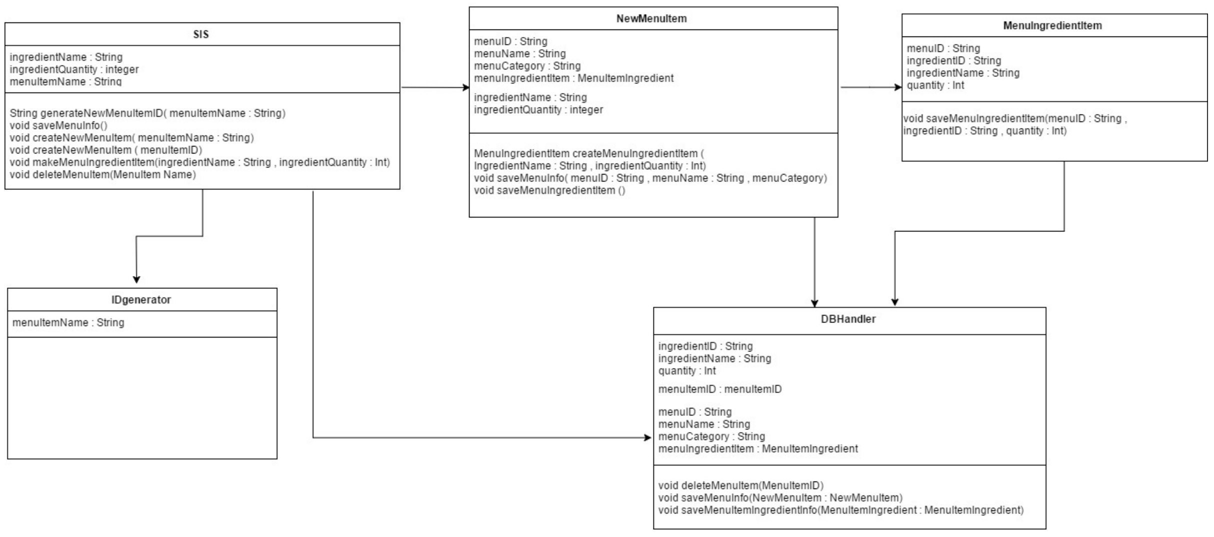
5.1.3. Combined DSD and DCD

1) combined DSD


: SIS
: IDgenerator
CreateNewMenuItem(menuItemName)
generateNewMenuItemID(menuItemName)
MenuItemID
createNewMenuItem(menuItemID)
: NewMenuItem
selectIngredient (ingredientName, ingredientQuantity)
makeMenuIngredientItem (ingredientName, ingredientQuantity)
loop
[more ingredient]
createMenuIngredientItem (ingredientName,ingredientQuanntity)
: MenuIngredientItem
MenuIngredientItem
saveMenuInfo()
saveMenuInfo()
: DBHandler
saveMenuInfo(menuID , menuName , menuCategory)
loop
[more ingredient]
saveMenuIngredientItem()
saveMenuIngredientItem(menuID , ingredientID , quantity)
deleteMenuItem(MenuItemName)
deleteMenuItem(MenuItemID)

<Figure 5.9> Manage Menu & ingredients combined Design Sequence Diagram

2) combined DCD

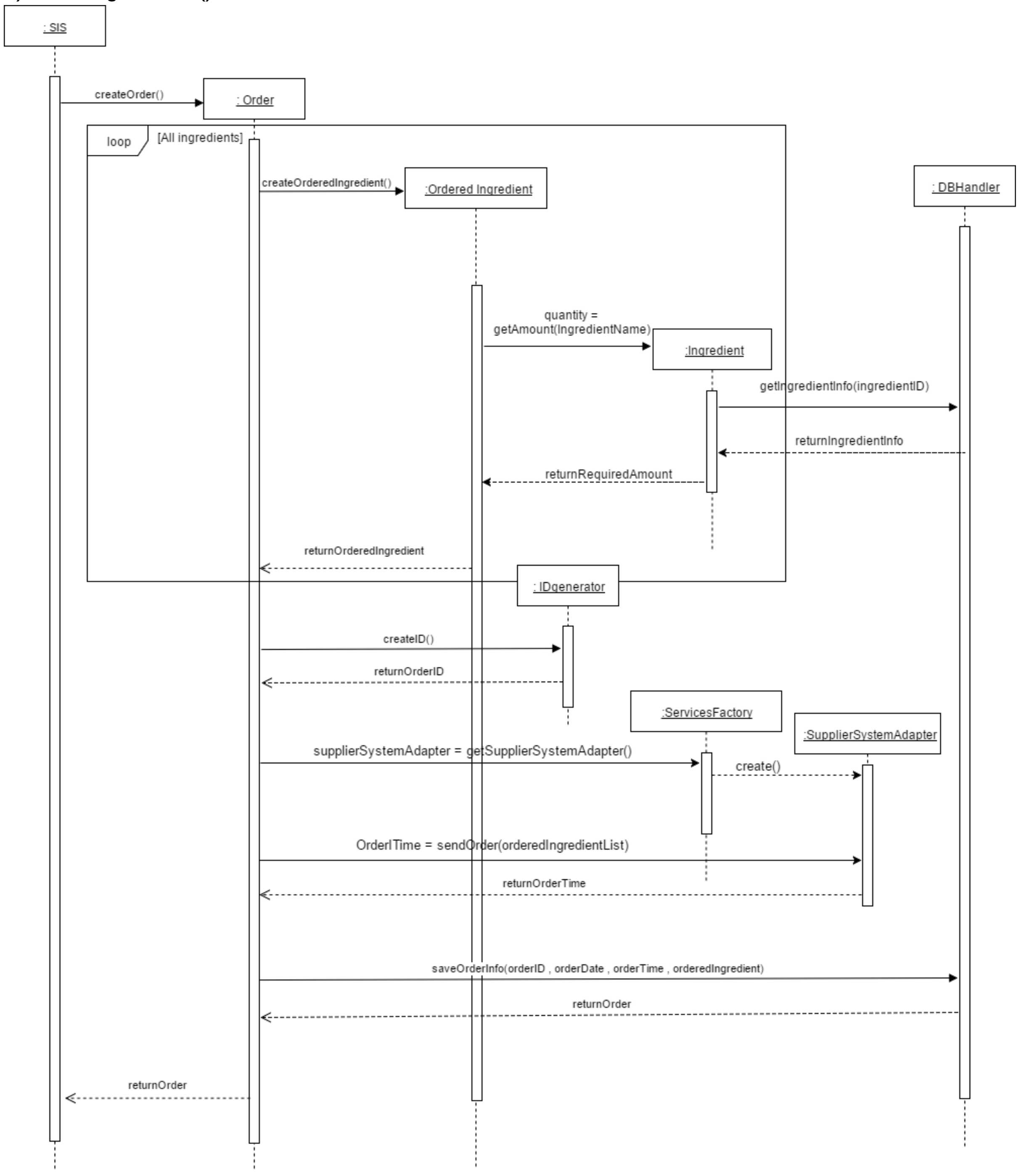

SIS
ingredientName : String
ingredientQuantity : integer
menuItemName : String
String generateNewMenuItemID(menuItemName : String)
void saveMenuInfo()
void createNewMenuItem(menuItemName : String)
void createNewMenuItem (menuItemID)
void makeMenuIngredientItem(ingredientName : String , ingredientQuantity : Int)
void deleteMenuItem(MenuItem Name)
NewMenuItem
menuID : String
menuName : String
menuCategory : String
menuIngredientItem : MenuItemIngredient
ingredientName : String
ingredientQuantity : integer
MenuIngredientItem createMenuIngredientItem (
IngredientName : String , ingredientQuantity : Int)
void saveMenuInfo(menuID : String , menuName : String , menuCategory)
void saveMenuIngredientItem ()
MenuIngredientItem
menuID : String
ingredientID : String
ingredientName : String
quantity : Int
void saveMenuIngredientItem(menuID : String ,
ingredientID : String , quantity : Int)
IDgenerator
menuItemName : String
DBHandler
ingredientID : String
ingredientName : String
quantity : Int
menuItemID : menuItemID
menuID : String
menuName : String
menuCategory : String
menuIngredientItem : MenuItemIngredient
void deleteMenuItem(MenuItemID)
void saveMenuInfo(NewMenuItem : NewMenuItem)
void saveMenuItemIngredientInfo(MenuItemIngredient : MenuItemIngredient)

<Figure 5.10> Manage Menu & ingredients combined Design Class Diagram

5.2. Order Ingredients Realization

5.2.1. Design Sequence Diagram

1) OrderIngredients()



<Figure 5.11> OrderIngredients Design Sequence Diagram

GRASP Pattern	Description
Controller	-SIS is a facade Controller. SIS is ‘root object’ for overall system
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

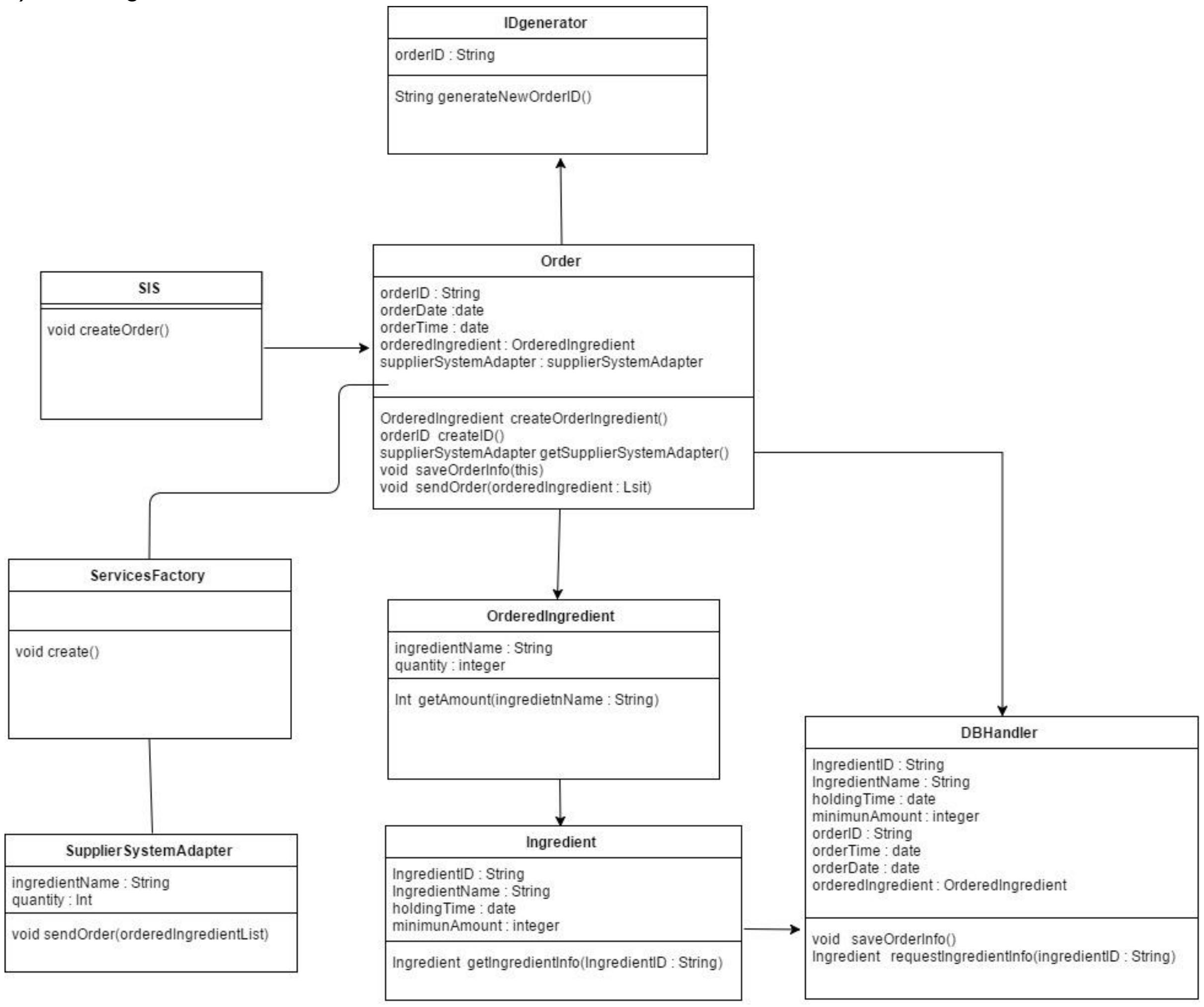
<Table 5.5> OrderIngredients DSD GRASP Pattern

GOF Pattern	Description
Adapter	- ‘SupplierSystemAdapter’ needs to support several kinds of external supplier system.
Factory	- ‘ServicesFactory’ create a Pure Fabrication object called a Factory that handles the creation.

<Table 5.6> OrderIngredients DSD GoF Pattern

5.2.2. Design Class Diagram

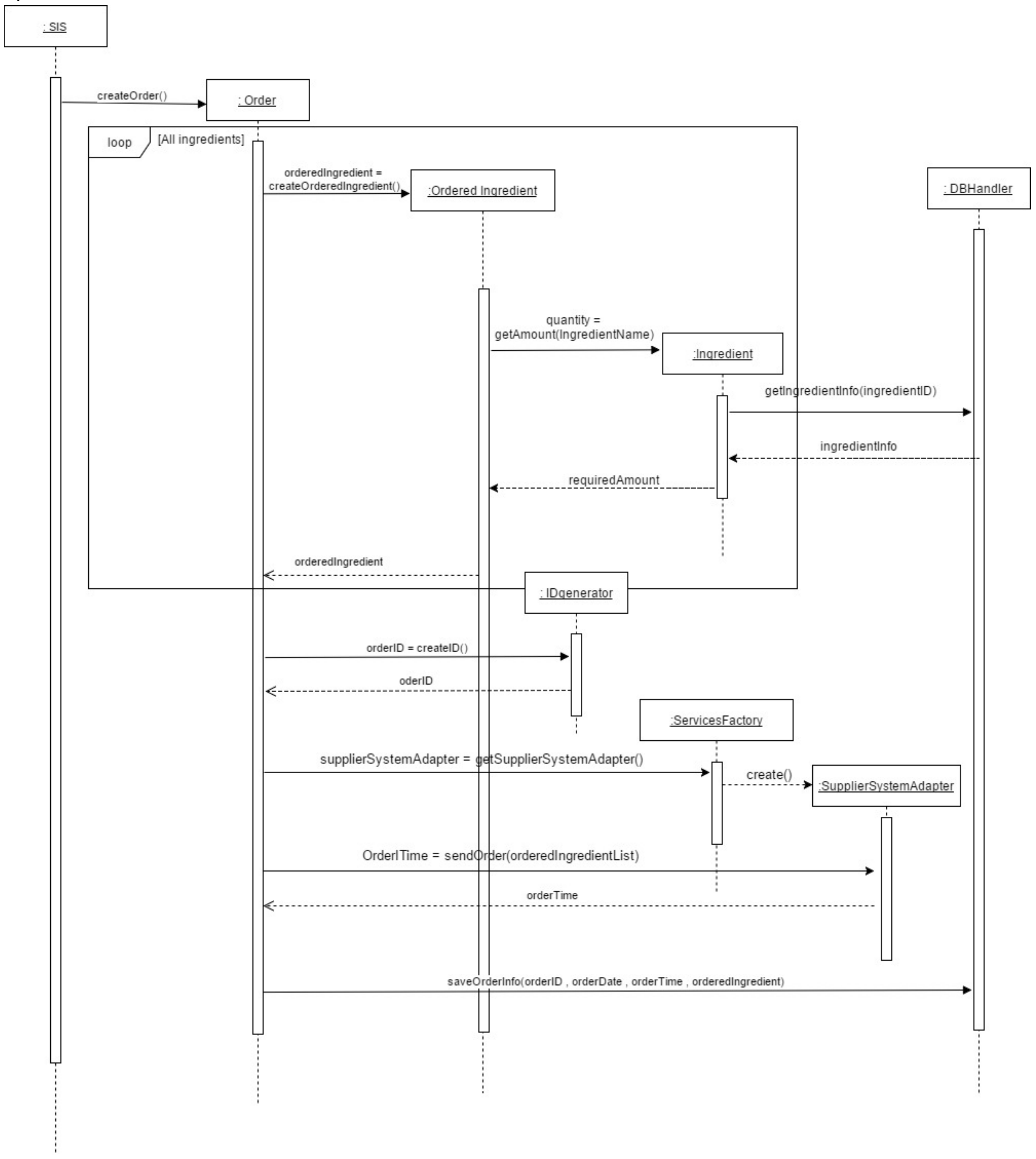
1) OrderIngredients



<Figure 5.12> OrderIngredients Design Class Diagram

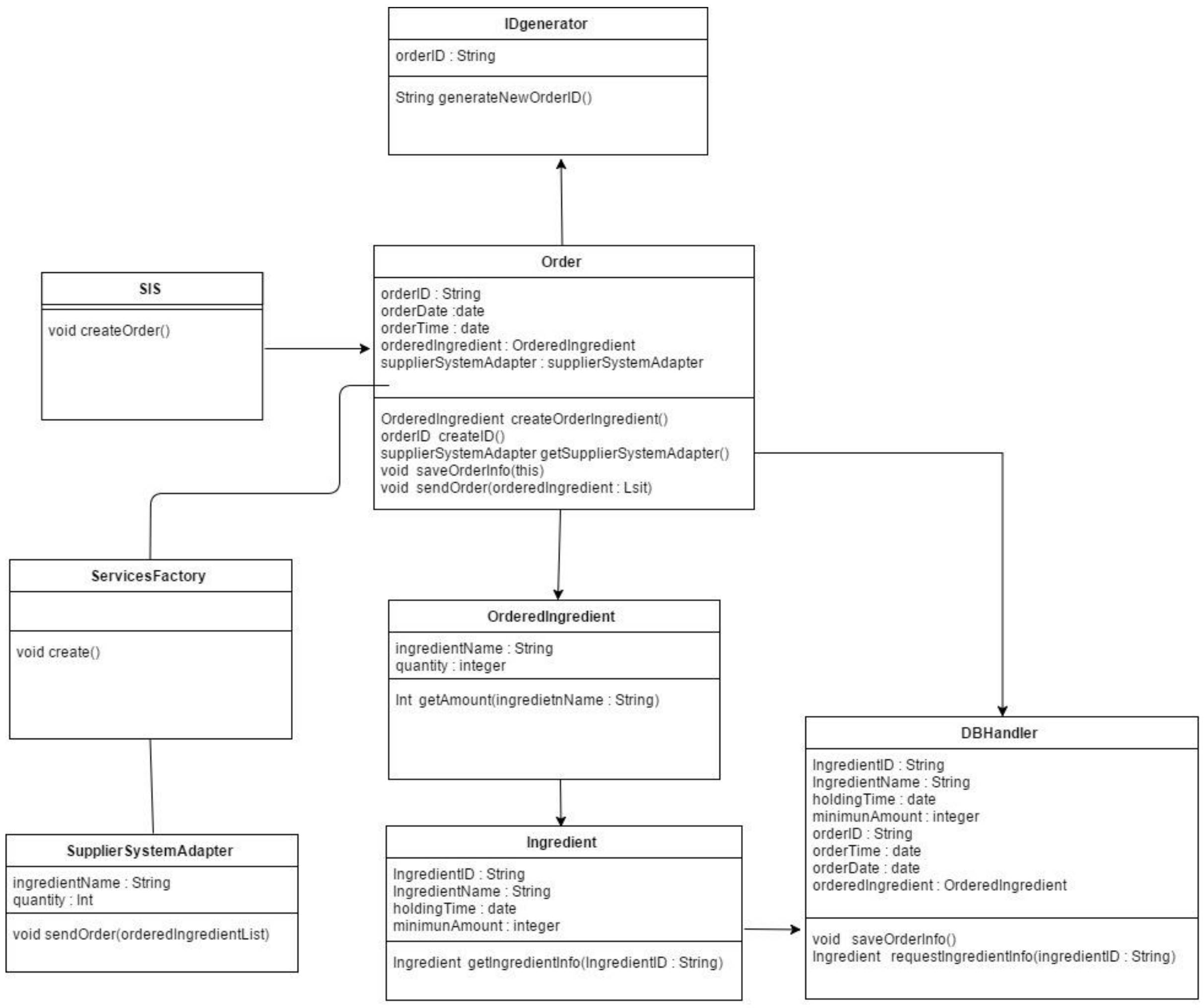
5.2.3. Combined DSD and DCD

1) combined DSD


: SIS
createOrder()
: Order
loop
[All ingredients]
orderedIngredient = createOrderedIngredient()
:Ordered Ingredient
: DBHandler
quantity = getAmount(IngredientName)
:Ingredient
getIngredientInfo(ingredientID)
ingredientInfo
requiredAmount
orderedIngredient
: IDgenerator
orderID = createID()
oderID
:ServicesFactory
supplierSystemAdapter = getSupplierSystemAdapter()
create()
:SupplierSystemAdapter
OrderlTime = sendOrder(orderedIngredientList)
orderTime
saveOrderInfo(orderID , orderDate , orderTime , orderedIngredient)

<Figure 5.13> OrderIngredients Combined Design Sequence Diagram

2) combined DCD

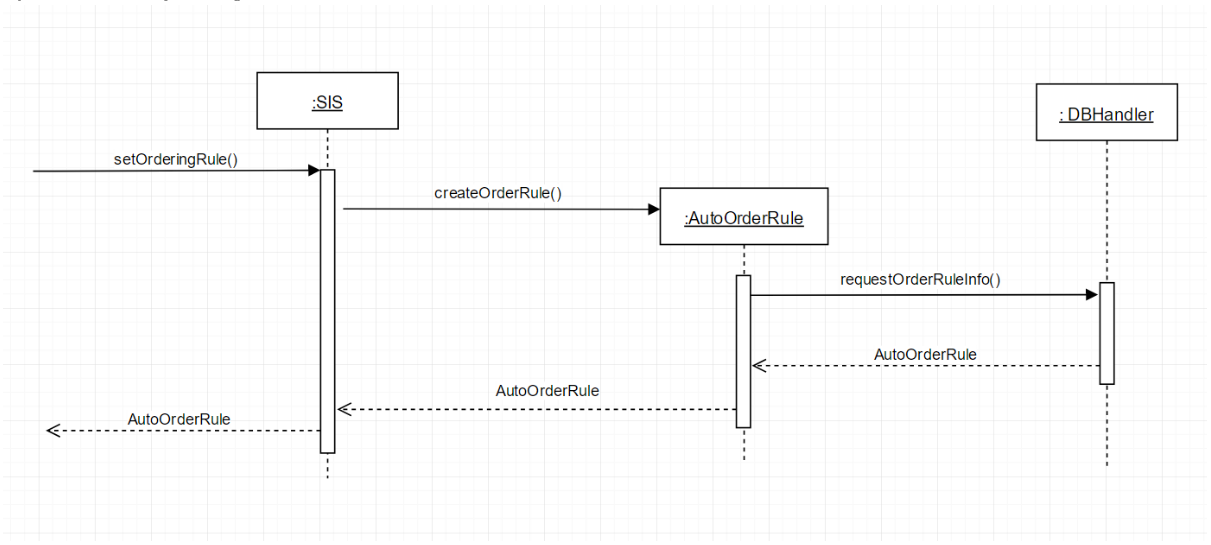


<Figure 5.14> OrderIngredients Combined Design Class Diagram

5.2-1. Set Ordering Rule

5.2-1.1. Design Sequence Diagram

1) setOrderingRule()

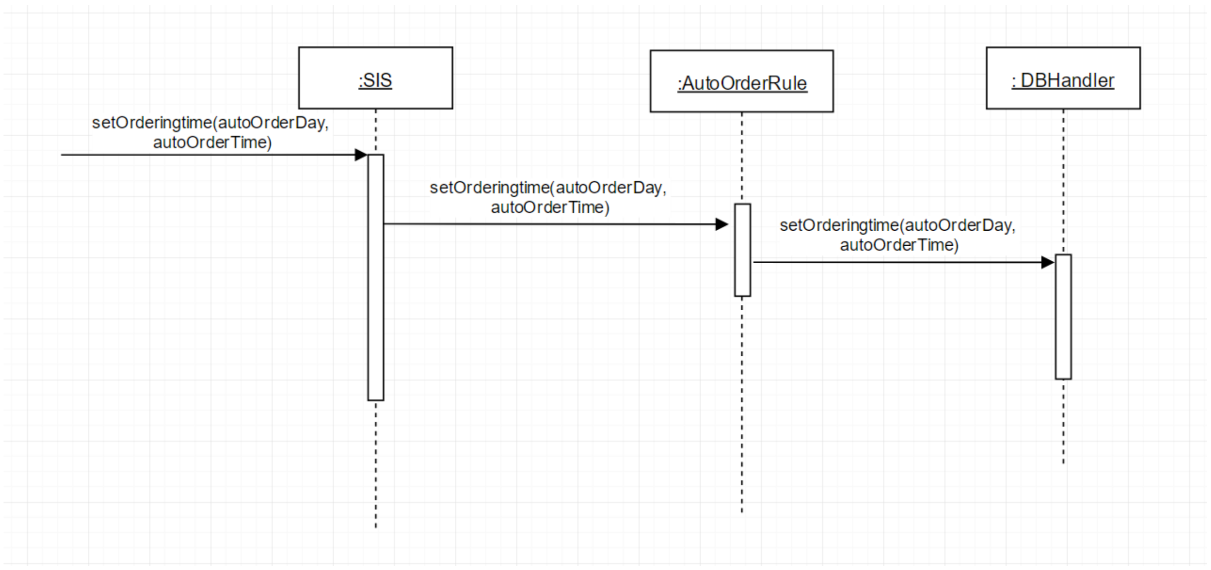


<Figure 5.15> setOrderingRule Design Sequence Diagram

GRASP Pattern	Description
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system.
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.7> setOrderingRule DSD GRASP Pattern

2) setOrderingTime(autoOrderDay, autoOrderTime)


:SIS
:AutoOrderRule
: DBHandler
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)

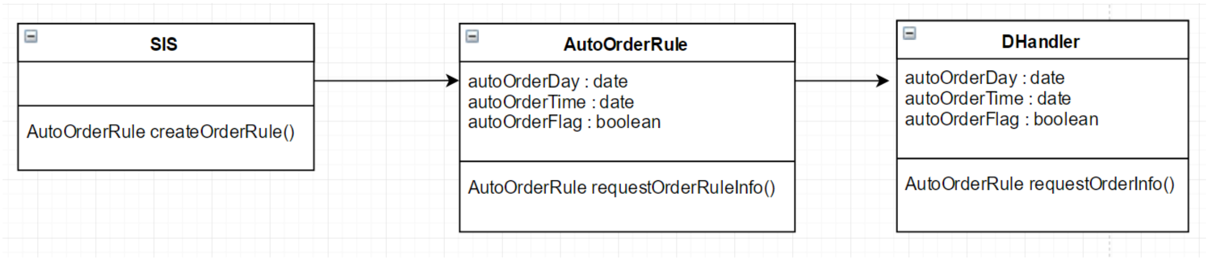
<Figure 5.16> setOrderingTime Design Sequence Diagram

GRASP Pattern	Description
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system
Pure Fabrication	- Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.8> setOrderingTime DSD GRASP Pattern

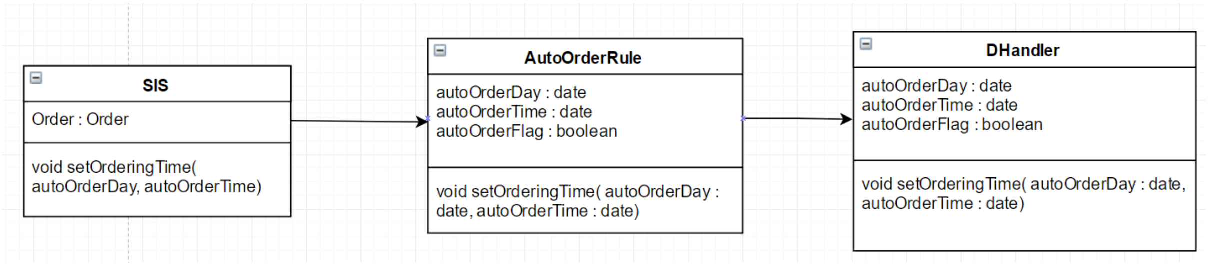
5.2-1.2. Design Class Diagram

1) setOrderingRule()



<Figure 5.17> setOrderingRule Design Class Diagram

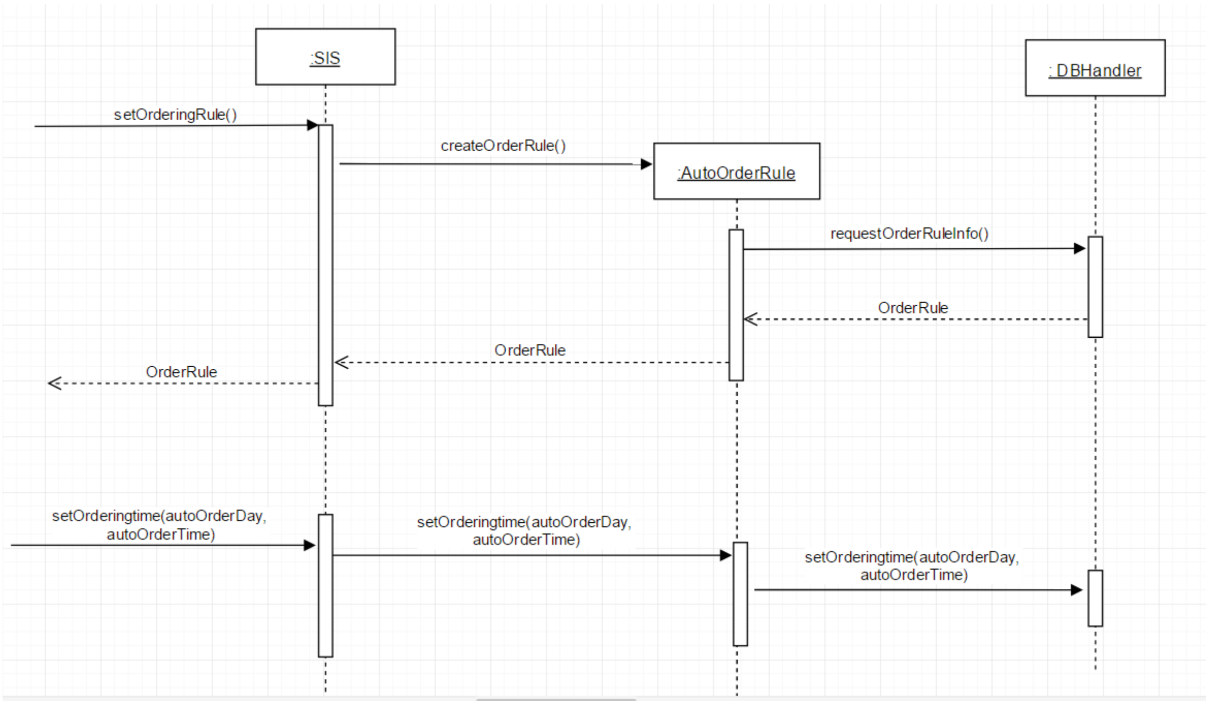
2) setOrderingTime(autoOrderDay, autoOrderTime)


SIS
Order : Order
void setOrderingTime(autoOrderDay, autoOrderTime)
AutoOrderRule
autoOrderDay : date
autoOrderTime : date
autoOrderFlag : boolean
void setOrderingTime(autoOrderDay : date, autoOrderTime : date)
DHandler
autoOrderDay : date
autoOrderTime : date
autoOrderFlag : boolean
void setOrderingTime(autoOrderDay : date, autoOrderTime : date)

<Figure 5.18> setOrderingTime Design Class Diagram

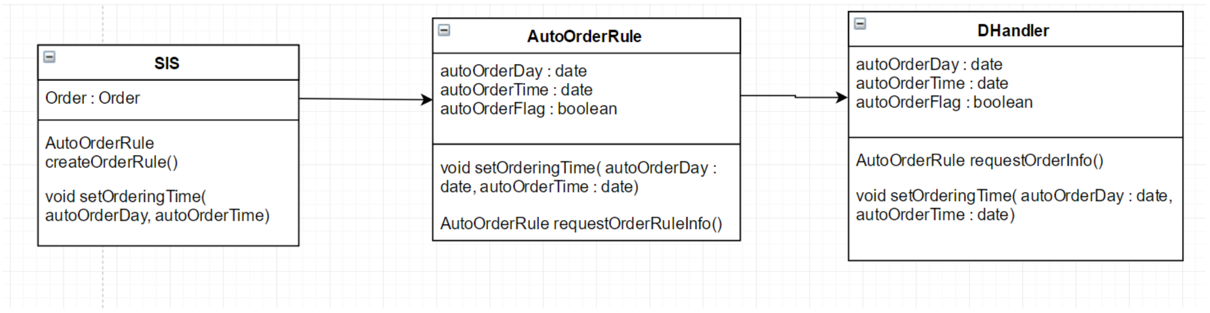
5.2-1.3. Combined DSD and DCD

1) combined DSD


:SIS
: DBHandler
setOrderingRule()
createOrderRule()
:AutoOrderRule
requestOrderRuleInfo()
OrderRule
OrderRule
OrderRule
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)

<Figure 5.19> setOrderingRule Combined Design Sequence Diagram

2) combined DCD

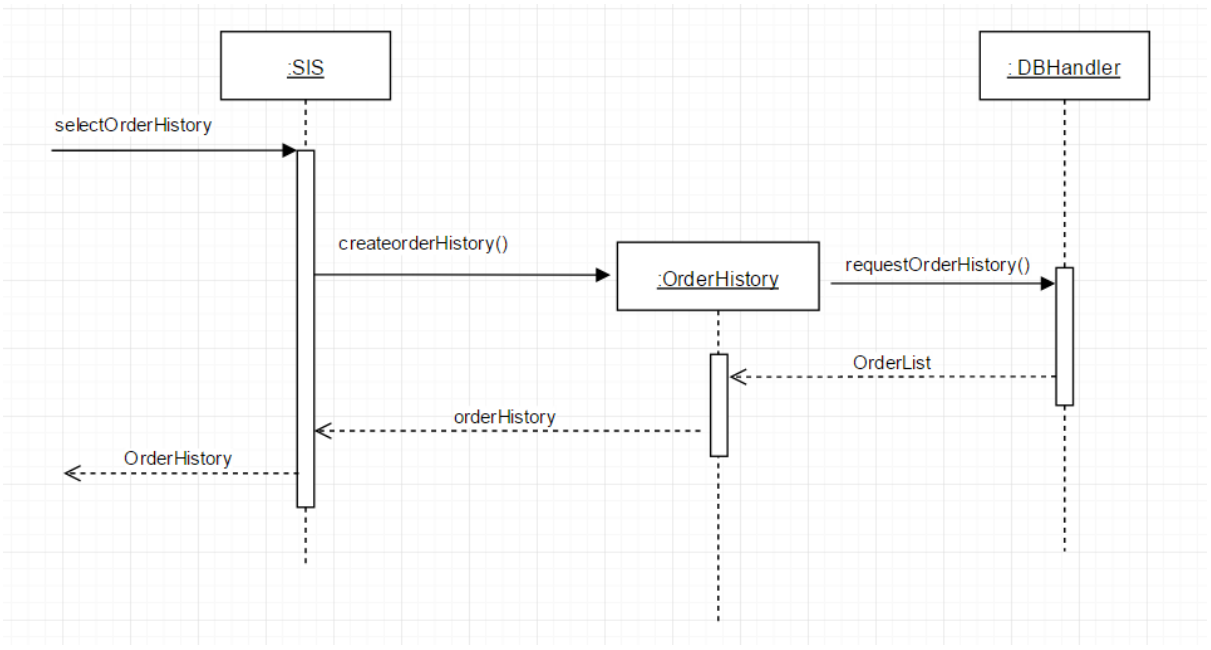

SIS
Order : Order
AutoOrderRule createOrderRule()
void setOrderingTime(autoOrderDay, autoOrderTime)
AutoOrderRule
autoOrderDay : date
autoOrderTime : date
autoOrderFlag : boolean
void setOrderingTime(autoOrderDay : date, autoOrderTime : date)
AutoOrderRule requestOrderRuleInfo()
DHandler
autoOrderDay : date
autoOrderTime : date
autoOrderFlag : boolean
AutoOrderRule requestOrderInfo()
void setOrderingTime(autoOrderDay : date, autoOrderTime : date)

<Figure 5.20> setOrderingRule Combined Design Class Diagram

5.2-2. Order History

5.2-2.1. Design Sequence Diagram

1) selectOrderHistory()

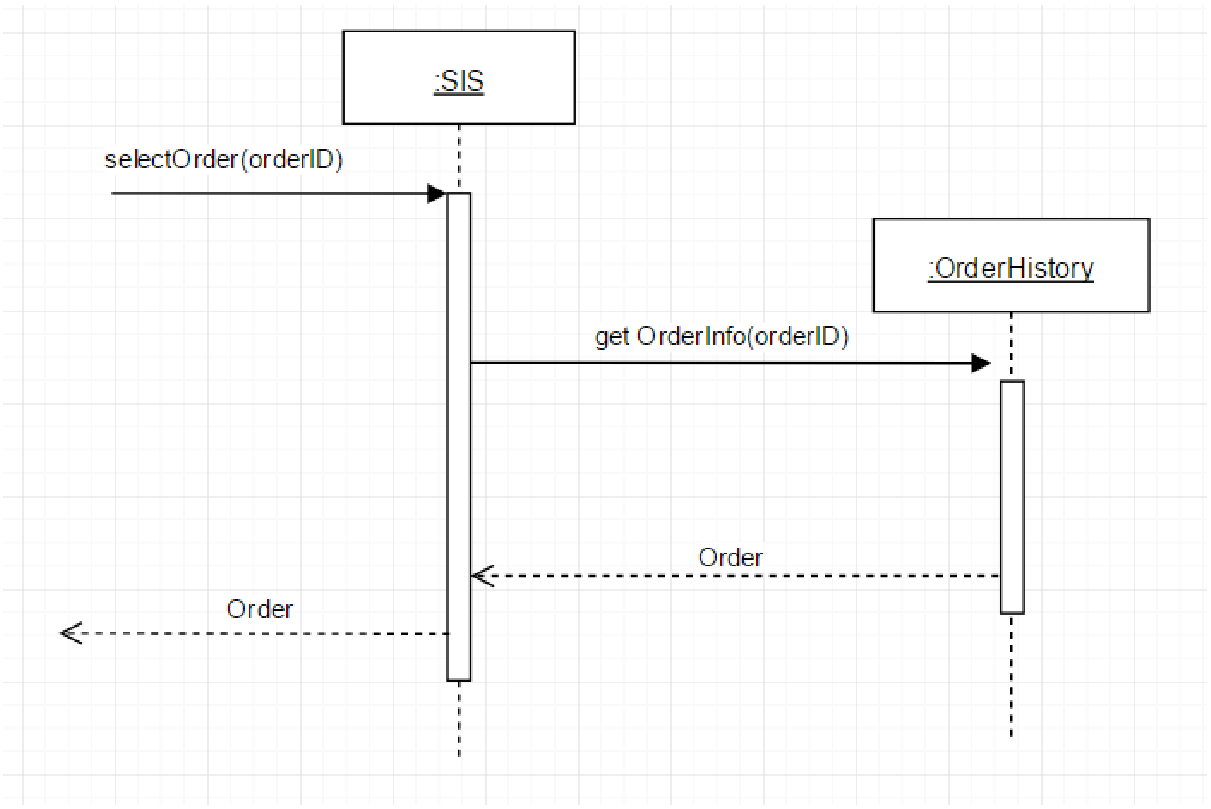


<Figure 5.21> selectOrderHistory Design Sequence Diagram

GRASP Pattern	Description
Creator	
Information Expert	
High Cohesion	
Low Coupling	
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system
Polymorphism	
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.
Indirection	
Protected Variations	

<Table 5.9> selectOrderHistory DSD GRASP Pattern

2) selectOrder(orderID)



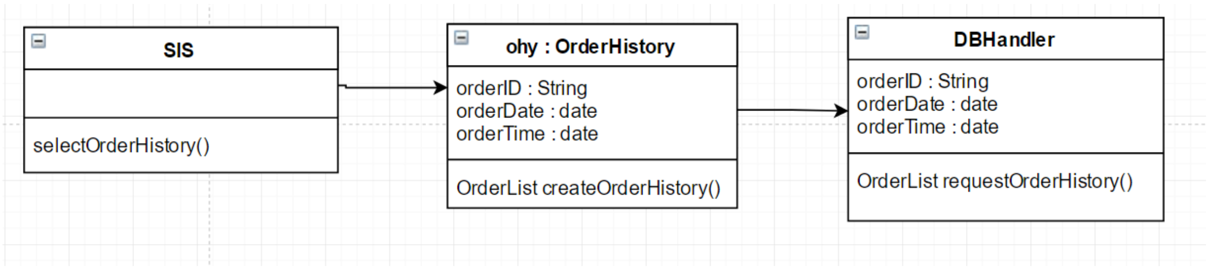
<Figure 5.22> selectOrder Design Sequence Diagram

GRASP Pattern	Description
Creator	
Information Expert	
High Cohesion	
Low Coupling	
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system.
Polymorphism	
Pure Fabrication	
Indirection	
Protected Variations	

<Table 5.10.> selectOrder DSD GRASP Pattern

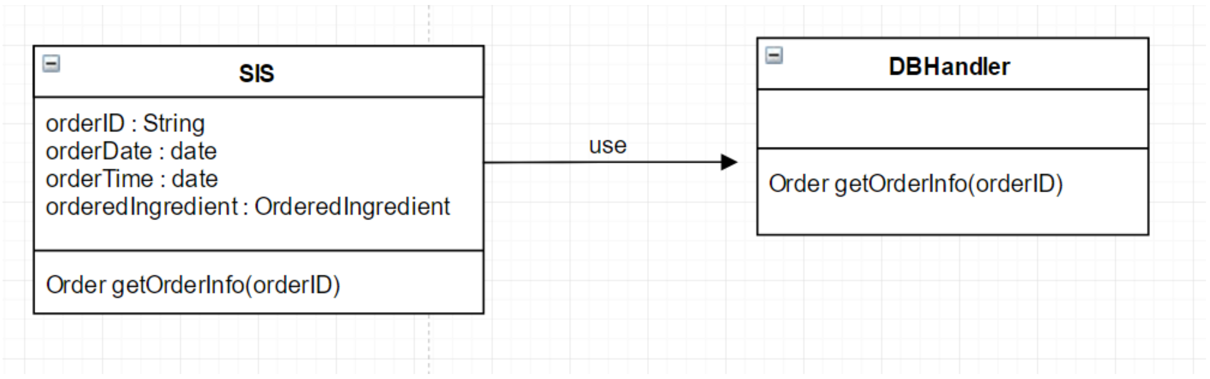
5.2-2.2. Design Class Diagram

1) selectOrderHistory()



<Figure 5.23> selectOrderHistory Design Class Diagram

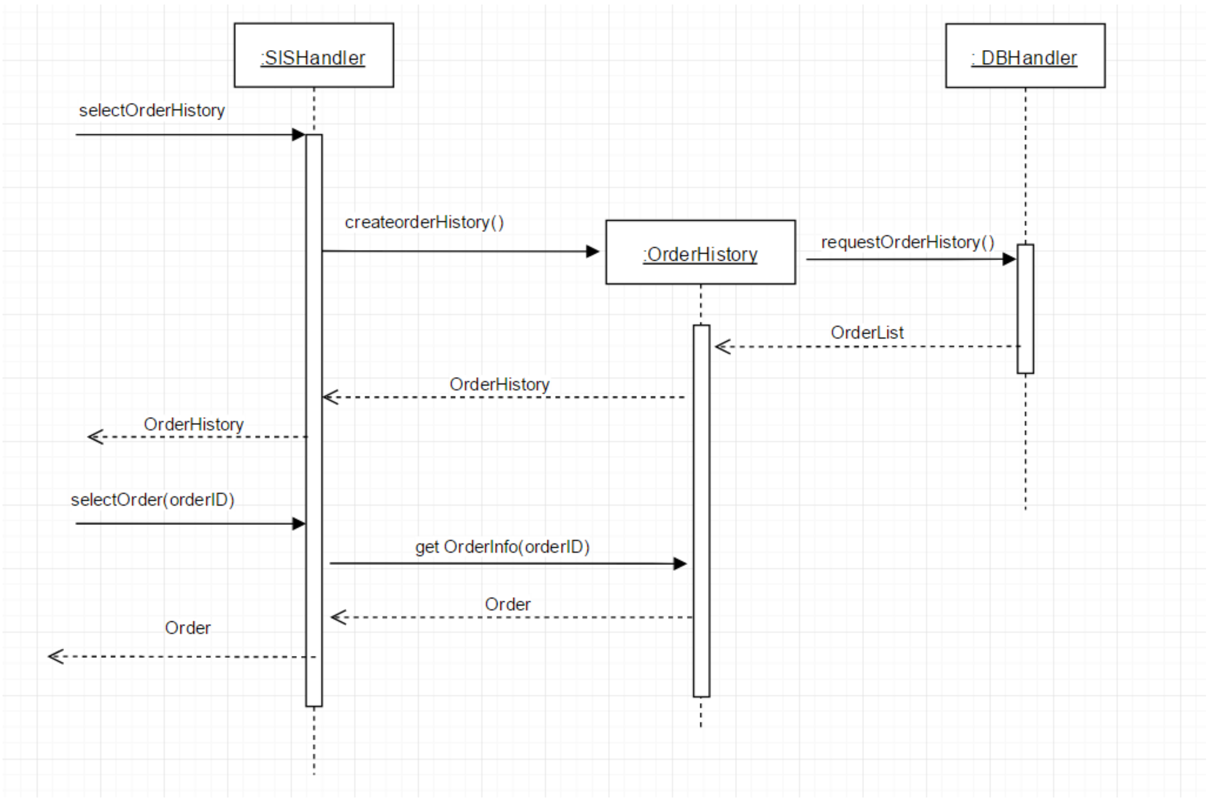
2) selectOrder(orderID)


SIS
orderID : String
orderDate : date
orderTime : date
orderedIngredient : OrderedIngredient
Order getOrderInfo(orderID)
use
DBHandler
Order getOrderInfo(orderID)

<Figure 5.24> selectOrder Design Class Diagram

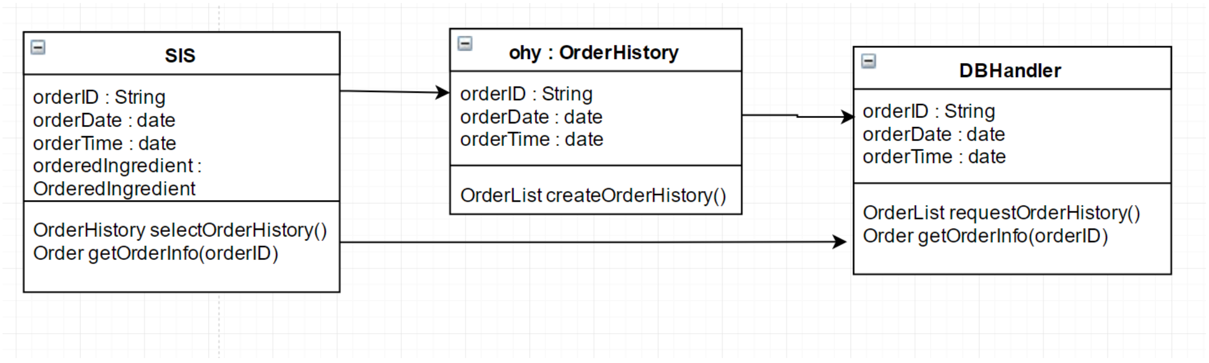
5.2-2.3. Combined DSD and DCD

1) combined DSD



<Figure 5.25> Order History Combined Design Sequence Diagram

2) combined DCD

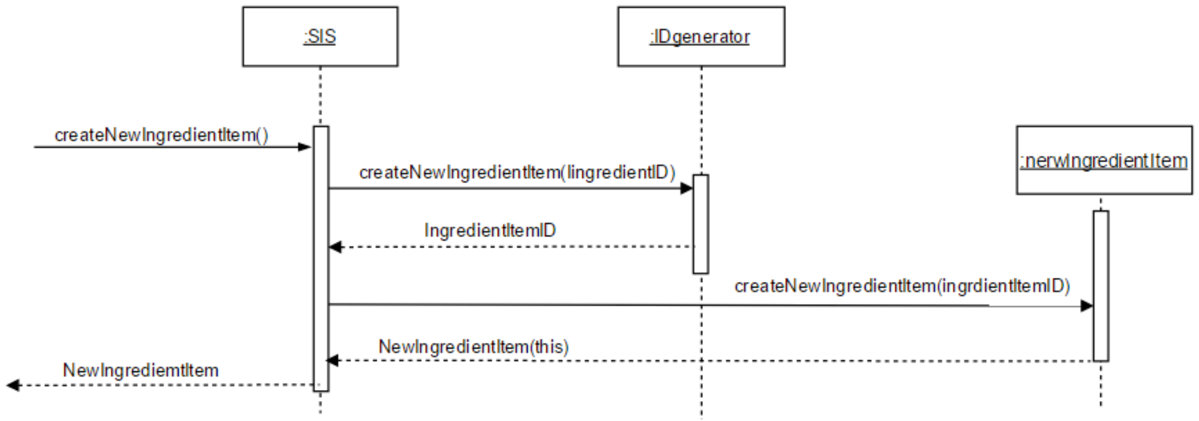

SIS
orderID : String
orderDate : date
orderTime : date
orderedIngredient : OrderedIngredient
OrderHistory selectOrderHistory()
Order getOrderInfo(orderID)
ohy : OrderHistory
orderID : String
orderDate : date
orderTime : date
OrderList createOrderHistory()
DBHandler
orderID : String
orderDate : date
orderTime : date
OrderList requestOrderHistory()
Order getOrderInfo(orderID)

<Figure 5.26> Order History Combined Design Class Diagram

5.3. Manage inventory Realization

5.3.1. Design Sequence Diagram

1) createNewIngredientItem()



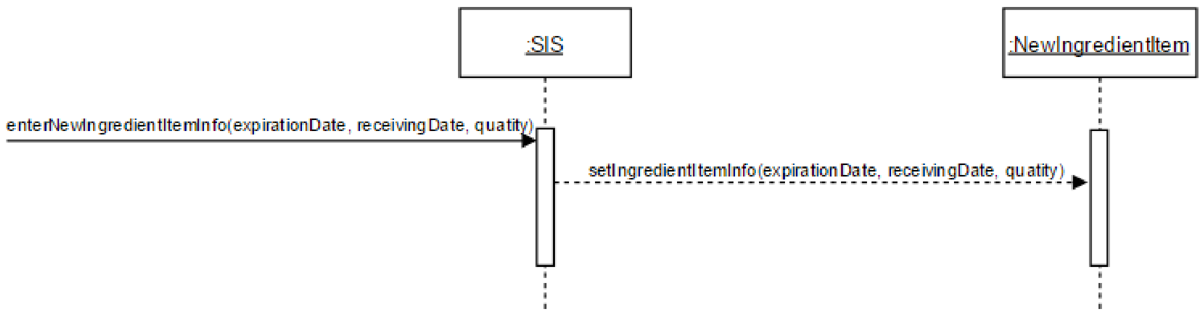
:SIS
:IDgenerator
:nerwIngredientItem
createNewIngredientItem()
createNewIngredientItem(IingredientID)
IngredientItemID
createNewIngredientItem(ingrdientItemID)
NewIngredientItem(this)
NewIngrediemtItem

<Figure 5.27> createNewIngredientItem Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	- The ingredient catalog class has information about all ingredients, but SIS directly creates a nerwIngredientItem object to reduce coupling.
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system.
Protected Variations	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.11>createNewIngredientItem DSD GRASP Pattern

2) enterNewIngredientItemInfo(expirationDate, receivingDate, quatity)

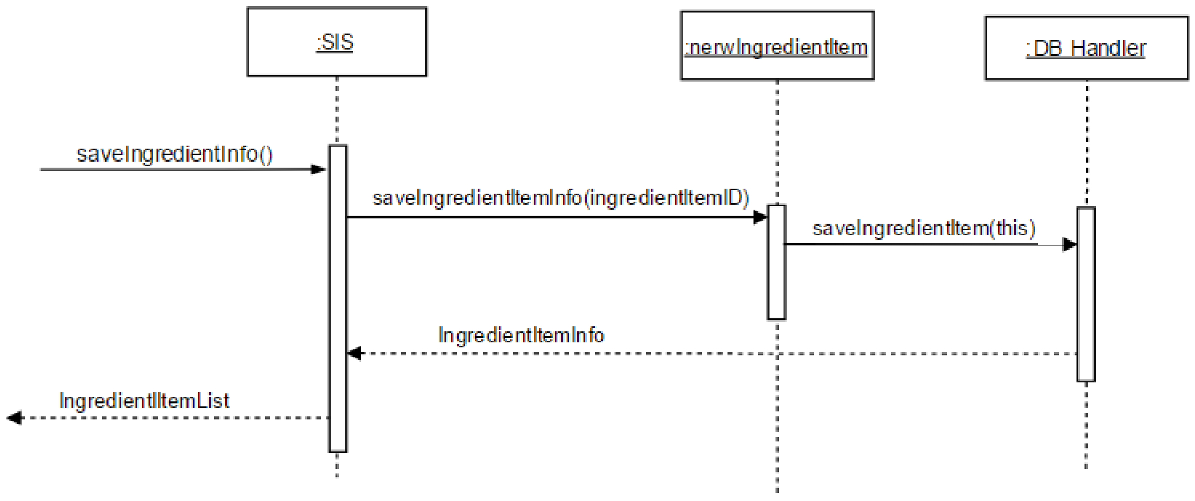

:SIS
:NewIngredientItem
enterNewIngredientItemInfo(expirationDate, receivingDate, quatity)
setIngredientItemInfo(expirationDate, receivingDate, quatity)

<Figure 5.28>enterNewIngredientItemInfo Design Sequence Diagram

GRASP Pattern	Description
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.12>enterNewIngredientItemInfo DSD GRASP Pattern

3) saveIngredientInfo()

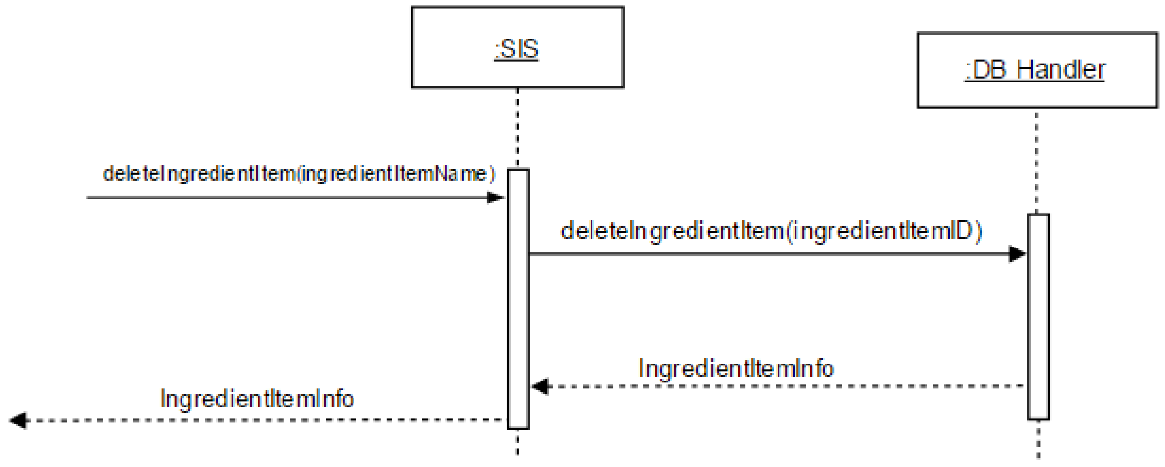


<Figure 5.29>saveIngredientInfo Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	-Reduce coupling by showing IngredientItemInfo which means 'saved', directly through SIS rather than through a newIngredientItem object.
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

<Table 5.13>saveIngredientInfo DSD GRASP Pattern

4) deleteIngredientItem(ingredientitemName)

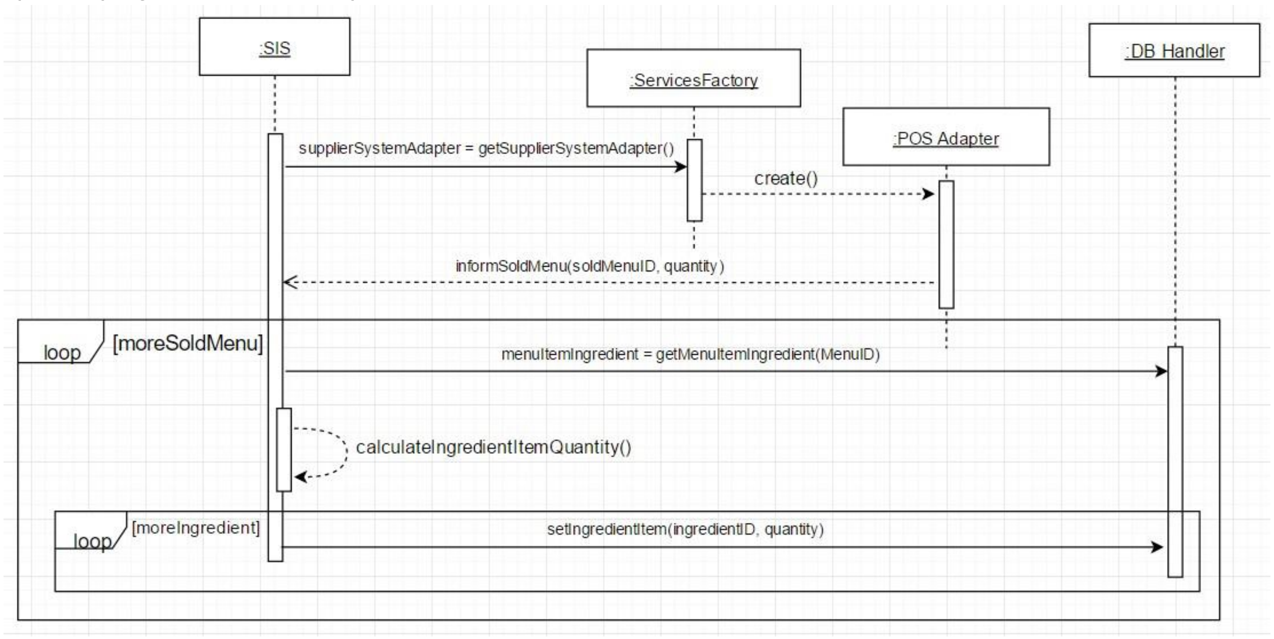

:SIS
:DB Handler
deleteIngredientItem(ingredientItemName)
deleteIngredientItem(ingredientItemID)
IngredientItemInfo
IngredientItemInfo

<Figure 5.30>deleteIngredientItem Design Sequence Diagram

GRASP Pattern	Description
Low Coupling	-When deleting a IngredientItem, rather than associating it with a IngredientItem object that matches the IngredientItemID you want to delete, you simply pass the IngredientItemID you want to delete to the DBHandler and delete it from the DB, so reduce coupling.
Controller	-SIS is a facade Controller. SIS is ‘root object’ for overall system.

<Table 5.14>deleteIngredientItem DSD GRASP Pattern

5) modifyIngredientItemInfo by POS


:SIS
:ServicesFactory
:POS Adapter
:DB Handler
supplierSystemAdapter = getSupplierSystemAdapter()
create()
informSoldMenu(soldMenuID, quantity)
loop
[moreSoldMenu]
menuItemIngredient = getMenuItemIngredient(MenuID)
calculateIngredientItemQuantity()
loop
[moreIngredient]
setIngredientItem(ingredientID, quantity)

<Figure 5.31>modifyIngredientItemInfo by POS Design Sequence Diagram

GRASP Pattern	Description
Controller	-SIS is a facade Controller. SIS is 'root object' for overall system.
Pure Fabrication	-Having a pure fabrication object named DB Handler for high cohesion and low coupling. Handler acts as an intermediator to DB, which is external service.

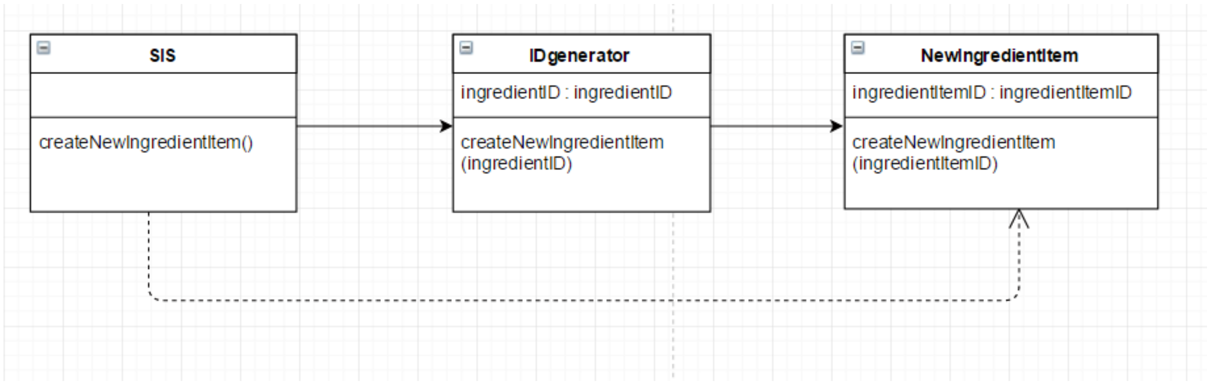
<Table 5.15>modifyIngredientItemInfo by POS DSD GRASP Pattern

GOF Pattern	Description
Adapter	- 'POSAdapter' needs to support several kinds of external POS system.
Factory	- 'ServicesFactory' create a Pure Fabrication object called a Factory that handles the creation.

<Table 5.16>modifyIngredientItemInfo by POS DSD GOF Pattern

5.3.2. Design Class Diagram

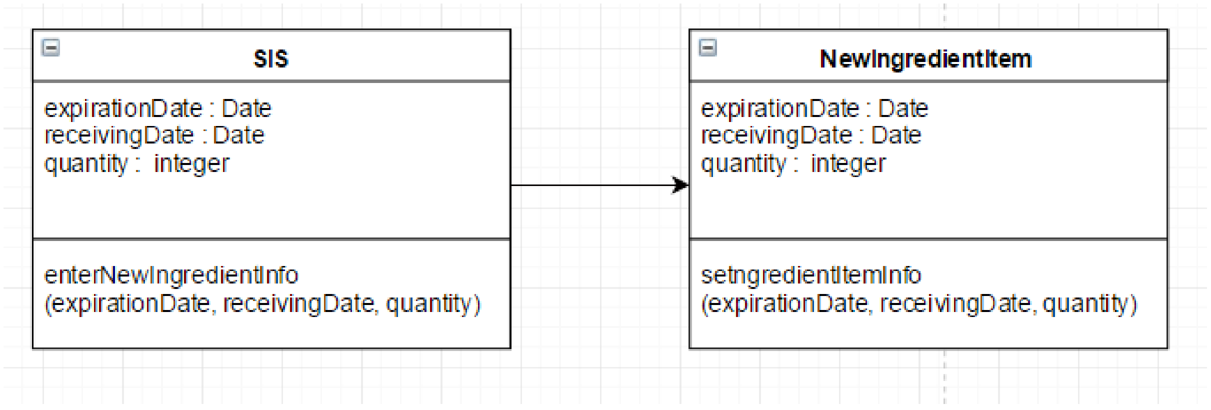
1) createNewIngredientItem()



SIS
createNewIngredientItem()
IDgenerator
ingredientID : ingredientID
createNewIngredientItem (ingredientID)
NewIngredientItem
ingredientItemID : ingredientItemID
createNewIngredientItem (ingredientItemID)

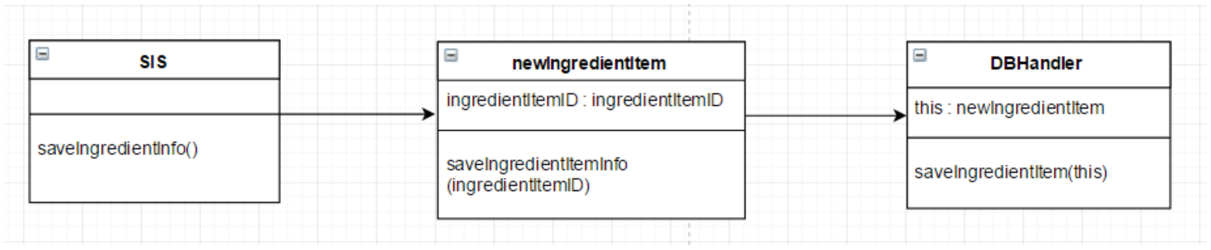
<Figure 5.32> createNewIngredientItem Design Class Diagram

2) enterNewIngredientInfo(expirationDate , receivingDate , quantity)



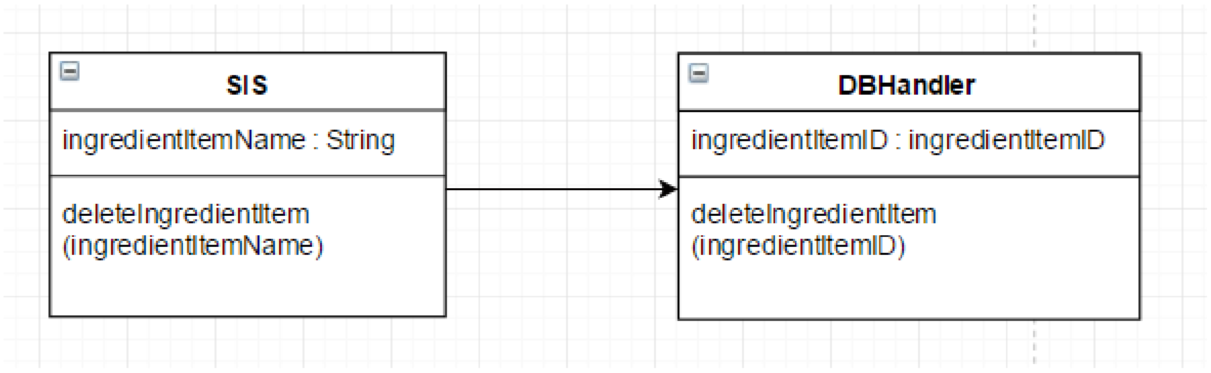
<Figure 5.33> enterNewIngredientInfo Design Class Diagram

3) saveIngredientInfo()



<Figure 5.34> saveIngredientInfo Design Class Diagram

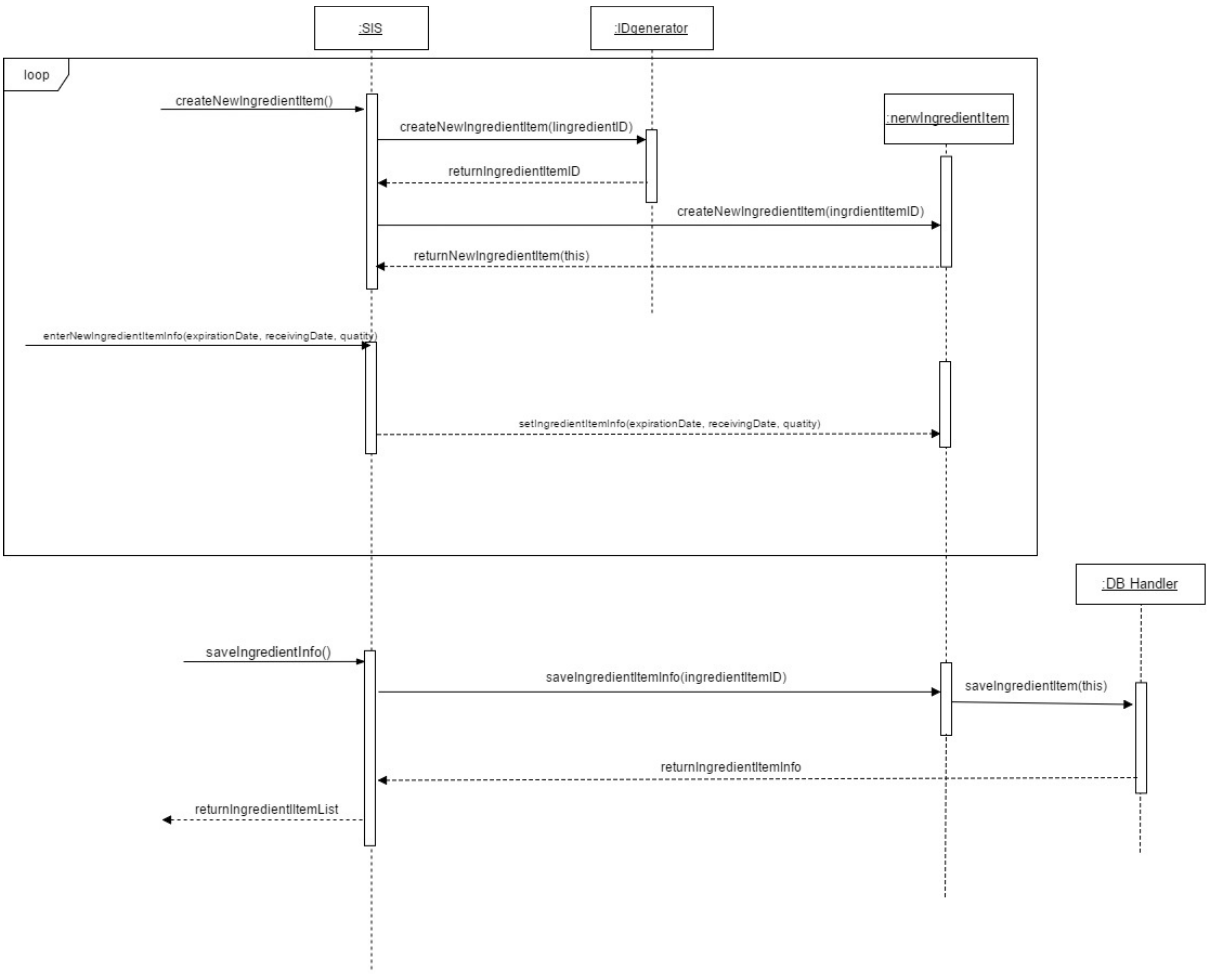
4) deleteIngredientItem(ingredientName)


SIS
ingredientItemName : String
deleteIngredientItem (ingredientItemName)
DBHandler
ingredientItemID : ingredientItemID
deleteIngredientItem (ingredientItemID)

<Figure 5.35> deleteIngredientItem Design Class Diagram

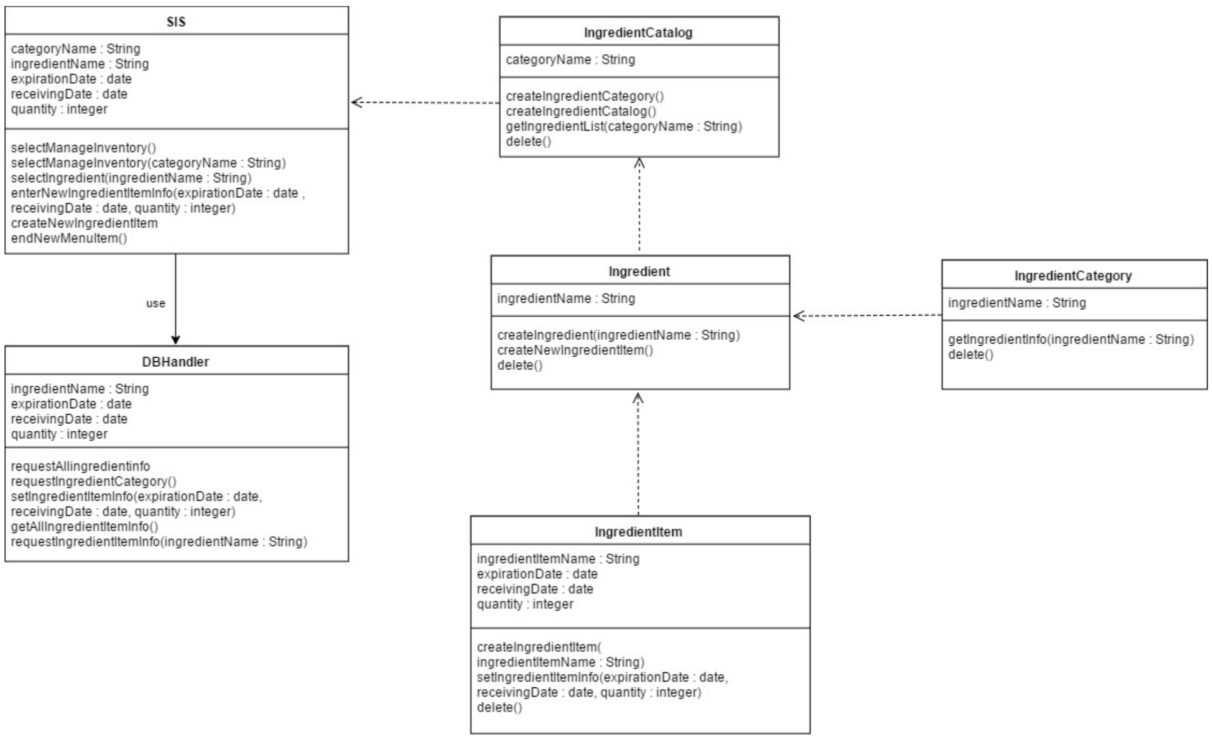
5.3.3. Combined DSD and DCD

1) Combined DSD


:SIS
:IDgenerator
loop
createNewIngredientItem()
:nerwIngredientItem
createNewIngredientItem(iingredientID)
returnIngredientItemID
createNewIngredientItem(ingrdientItemID)
returnNewIngredientItem(this)
enterNewIngredientItemInfo(expirationDate, receivingDate, quatity)
setIngredientItemInfo(expirationDate, receivingDate, quatity)
:DB Handler
saveIngredientInfo()
saveIngredientItemInfo(ingredientItemID)
saveIngredientItem(this)
returnIngredientItemInfo
returnIngredientItemList

<Figure 5.36> Manage Inventory Combined Design Sequence Diagram

2) Combined DCD


SIS
categoryName : String
ingredientName : String
expirationDate : date
receivingDate : date
quantity : integer
selectManageInventory()
selectManageInventory(categoryName : String)
selectIngredient(ingredientName : String)
enterNewIngredientItemInfo(expirationDate : date , receivingDate : date, quantity : integer)
createNewIngredientItem
endNewMenuItem()
use
DBHandler
ingredientName : String
expirationDate : date
receivingDate : date
quantity : integer
requestAllingredientinfo
requestIngredientCategory()
setIngredientItemInfo(expirationDate : date, receivingDate : date, quantity : integer)
getAllIngredientItemInfo()
requestIngredientItemInfo(ingredientName : String)
IngredientCatalog
categoryName : String
createIngredientCategory()
createIngredientCatalog()
getIngredientList(categoryName : String)
delete()
Ingredient
ingredientName : String
createIngredient(ingredientName : String)
createNewIngredientItem()
delete()
IngredientCategory
ingredientName : String
getIngredientInfo(ingredientName : String)
delete()
IngredientItem
ingredientItemName : String
expirationDate : date
receivingDate : date
quantity : integer
createIngredientItem(ingredientItemName : String)
setIngredientItemInfo(expirationDate : date, receivingDate : date, quantity : integer)
delete()

<Figure 5.37> Manage Inventory Combined Design Class Diagram

6. Design Class Diagram of the system

7. Architecture

7.1. Introduction.

7.1.1. Logical View

Describe ‘Cafe Inventory Management System’ in view point of layer (UI, Domain, Technical services), package, framework, subsystem, interface. Show UC1, UC3’s implementation scenario looks like interaction diagram.

7.1.2. Process View

It shows main ‘Cafe Inventory Management System’ execute process. Grouping thread to process and each threads have their executive factor.

7.1.3. Deployment View

It shows relationships about server and database.

7.1.4. Use Case View

It shows use-case view about UC1,UC2-1,UC2-2,UC3 which was implemented based on each scenario.

7.1.5. Data View

It shows data flows about UC1,UC2-1,UC2-2,UC3. These are show relation between data flow and database.

7.1.6. Implementation View

It shows application (implementation scene) and source code about ‘Cafe Inventory Management System’.

7.2. Architectural Factors

Factor	Measures and quality scenarios	Variability	Impact of factor	Priority for success	Difficulty or Risk
Reliability - Recoverability					
Perform periodic checks.	The system is checked once a month for one hour (from 2:00 am to 3:00 am).	Current flexibility: - as described by factor Evolution : - none	Influence on design is small.	H	M
Repair external service and connection failure.	If the connection with the external service fails, reconnect within 3 minutes and make it available.	Current flexibility: - SME recommends that system is not working at the moment and connect later. Evolution : - The number of users who use the service within three years increases. An overload of external service is caused by the user.	It has a big impact on large-scale design.	H	M
Reliability - Security					
Protect the internal database.	Validates the input when the user tries to authenticate or retrieve the value of the database.	Current flexibility: - as described by factor Evolution : - none	Influence on design is small.	H	H
Performance - Response Time					
응답시간을 줄인다.	Updating inventory or ingredient information is done within 5 seconds 90% of time. View existing	Current flexibility: - as described by factor. Evolution : - Updates within 3 years are updated in 95% or more in 3 seconds.	It has a big impact on large-scale design.	M	L

	inventory or ingredient information and order history is done within 3 seconds 90% of time.	Displays information within three years of 95% or more in one second.			
Performance - Throughput					
Increase the amount that can be processed for a certain amount of time.	Quickly handle events that users generate.	Current flexibility: - Processes more than 95% of user's main events within 5 seconds. Evolution : - Within three years, it processes more than 95% of user's major events within 3 seconds.	It has a big impact on large-scale design.	M	L
Supportability - Adaptability					
Support various environme nts.	Users who want to use Cafe Inventory System in other environments will be supported within 15 days.	Current flexibility: - as described by factor Evolution : - none	It has a significant impact on design in terms of transformations protected from many factors. For example, the operating system and the UI are different.	M	H

<Table 7.1> Architectural Factors

7.3. Architectural Decisions (Technical memos)

7.3.1. Technical Memo 1

Issue : Reliability - Repair external service and connection failure.

Solution Summary : To recover from errors with external services and connection failures.

Factor :

- Recovery from external services connection failures.

Solution :

- Provide a local service server. Prevent unexpected system failures. Provide a local implementation of the database. And the database of information about user information, history information, and system state before failure will be a small cache recently reserved. Therefore, if a failure occurs with an external system, use cache storage to recover.

Motivation :

- Users don’t want to have problem using Cafe Inventory Management System. Therefore, if we offer this level of reliability, it will be a very attractive service. And, we want to provide our system eternally without server failure.

Unresolved Issues :

- None.

Alternatives Considered :

- None.

7.3.2. Technical Memo 2

Issue : Supportability - Support various environments.

Solution Summary : Protected Variation using interfaces and Devices.

Factor :

- Support other environment(mobile phone, tablet, …).

Solution :

- Provide common interface for different Operation System.(Android, IOS, …). For example, you can use swift and android. With various environments, users can use our Cafe Inventory Management System service comfortably in all device.
-

Motivation :

- These days, there are a lot of devices with different Operation System. Users want to get consistent services for various device they use.

Unresolved Issues :

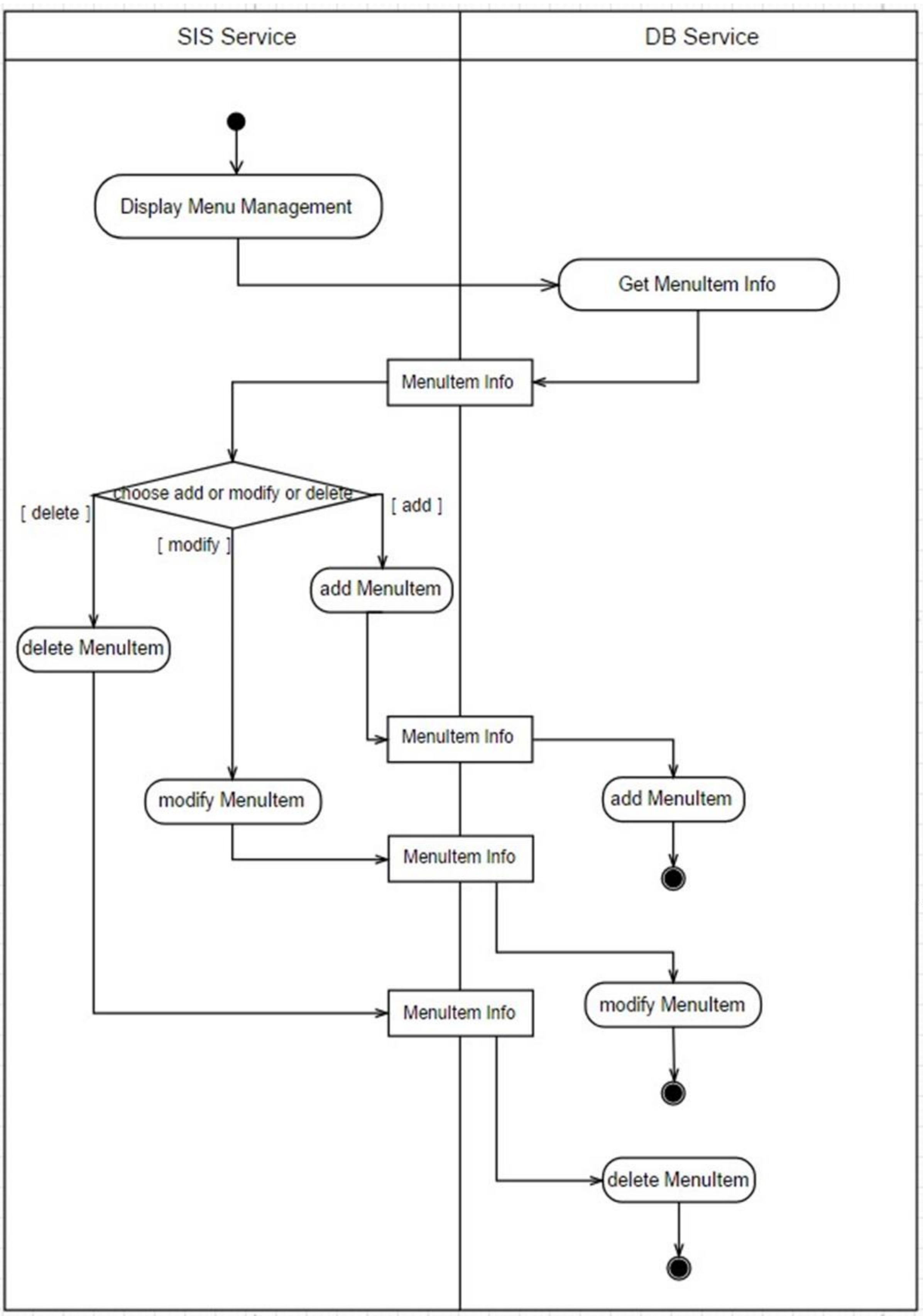
- None.

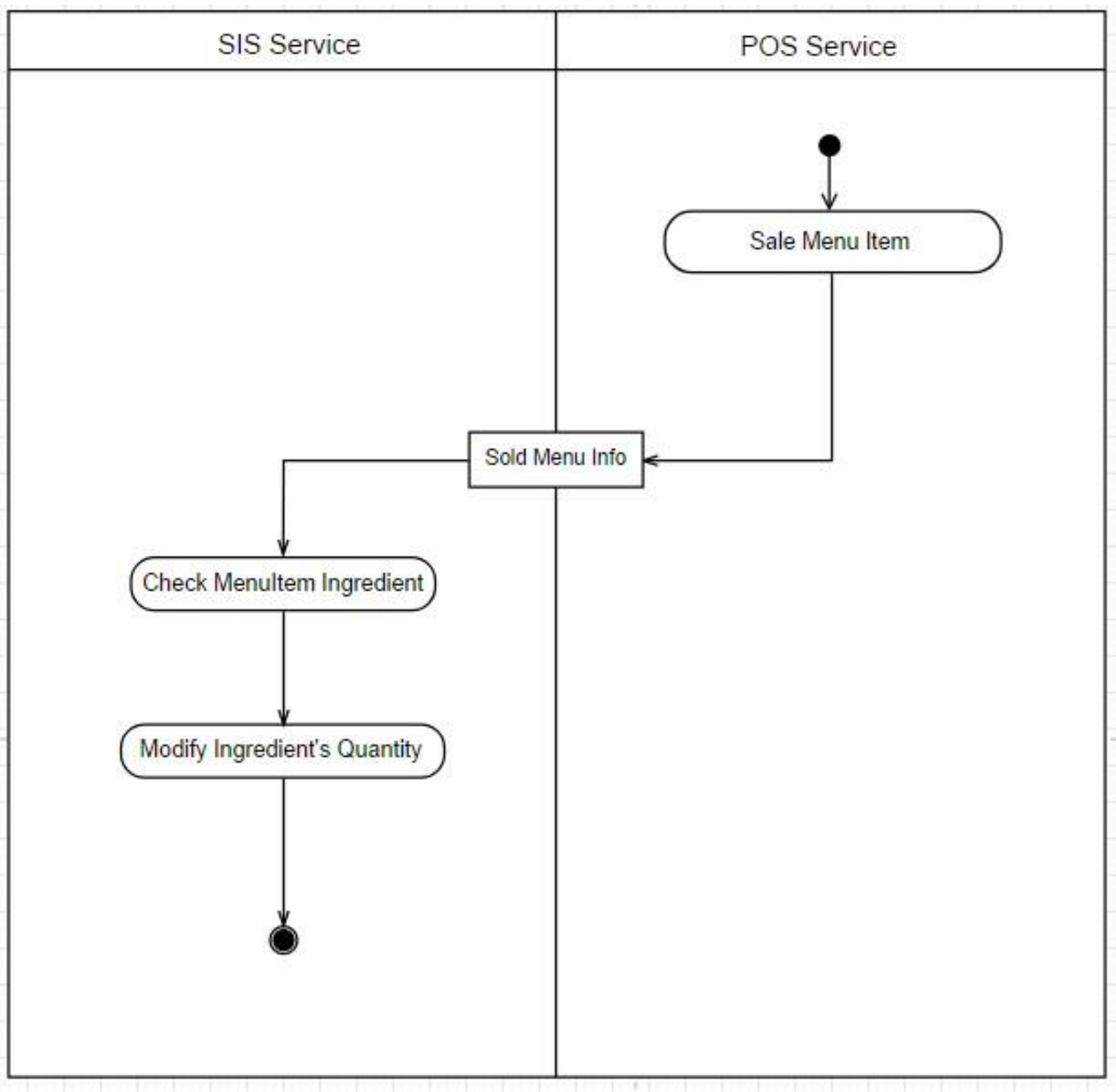
Alternatives Considered

- None.

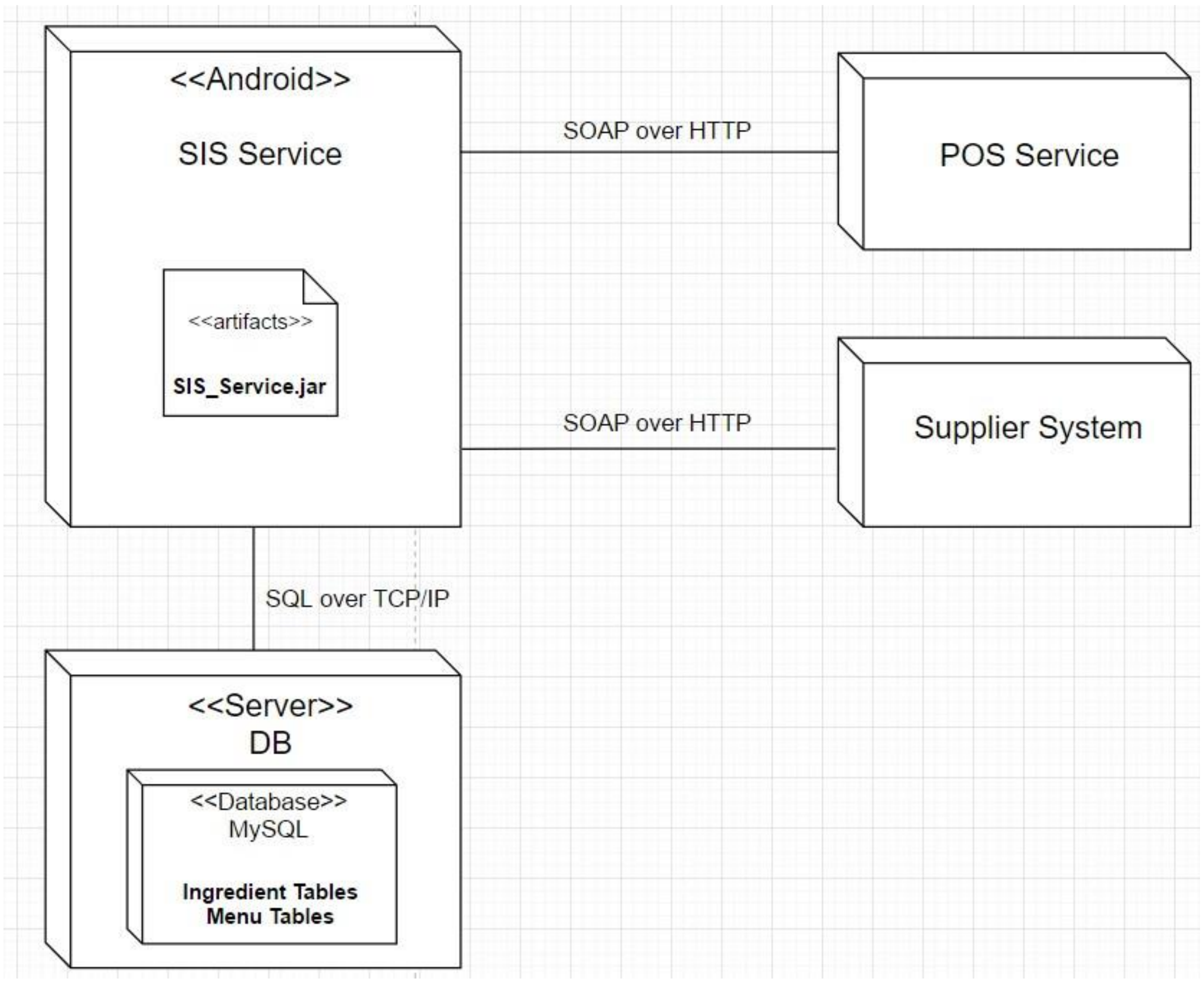
7.4. Logical View

7.5. Process View



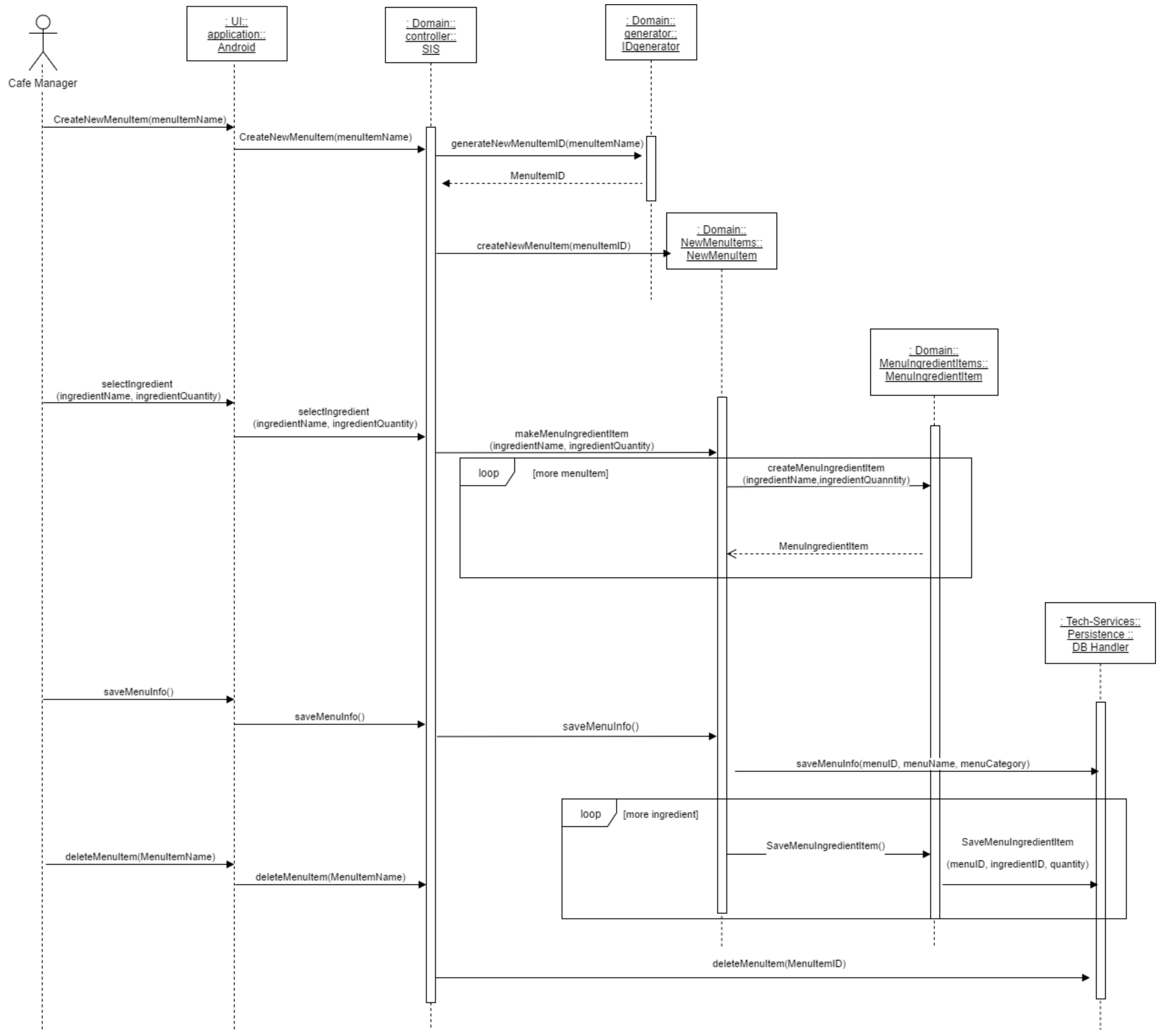

SIS Service
POS Service
Sale Menu Item
Sold Menu Info
Check MenuItem Ingredient
Modify Ingredient's Quantity

7.6. Deployment View



7.7. Use Case View

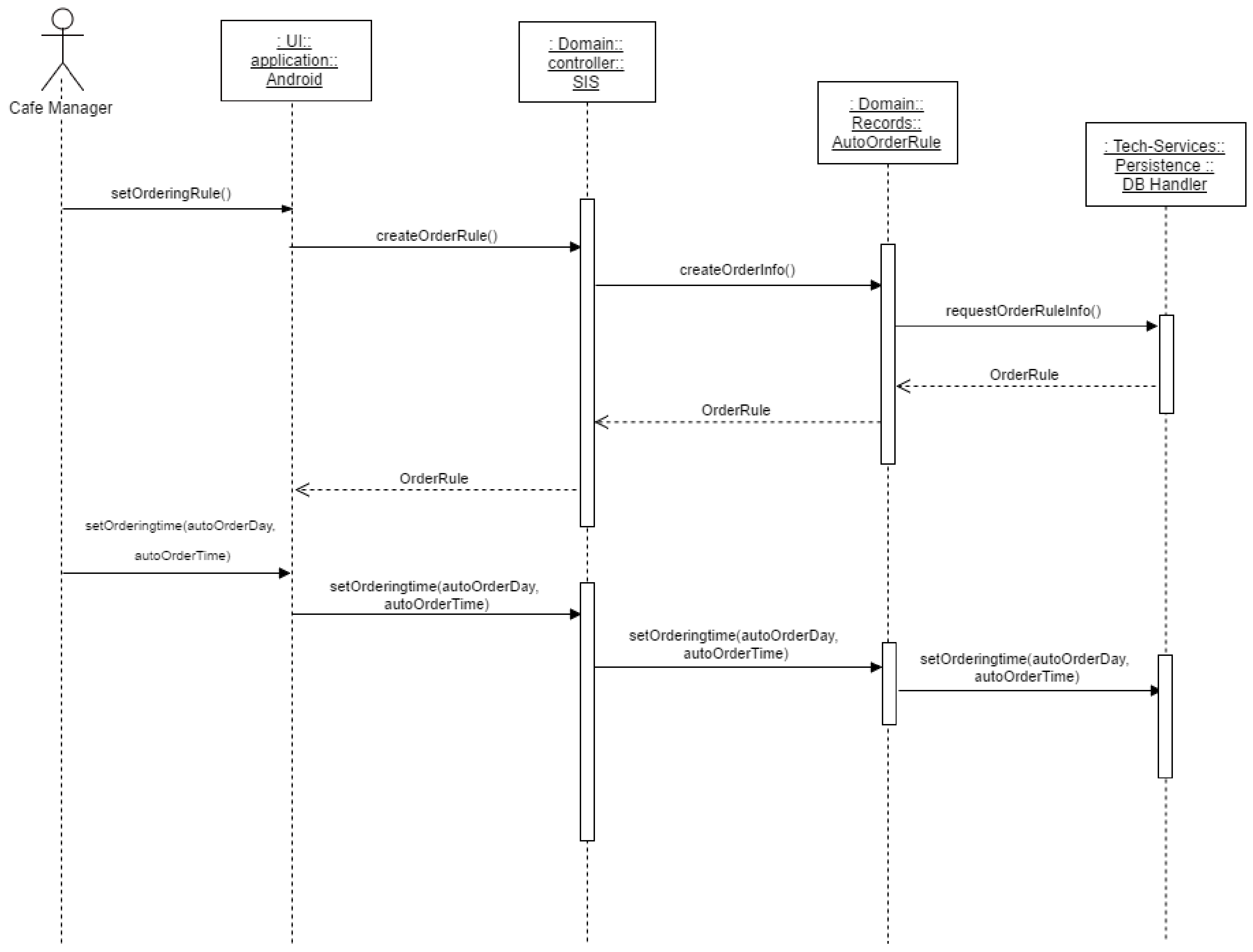
7.7.1. Manage menu & ingredients



Cafe Manager
: UI:: application:: Android
: Domain:: controller:: SIS
: Domain:: generator:: IDgenerator
: Domain:: NewMenuItems:: NewMenuItem
: Domain:: MenuIngredientItems:: MenuIngredientItem
: Tech-Services:: Persistence :: DB Handler
CreateNewMenuItem(menuItemName)
CreateNewMenuItem(menuItemName)
generateNewMenuItemID(menuItemName)
MenuItemID
createNewMenuItem(menuItemID)
selectIngredient (ingredientName, ingredientQuantity)
selectIngredient (ingredientName, ingredientQuantity)
makeMenuIngredientItem (ingredientName, ingredientQuantity)
loop
[more menuItem]
createMenuIngredientItem (ingredientName,ingredientQuanntity)
MenuIngredientItem
saveMenuInfo()
saveMenuInfo()
saveMenuInfo()
saveMenuInfo(menuID, menuName, menuCategory)
loop
[more ingredient]
SaveMenuIngredientItem()
SaveMenuIngredientItem
(menuID, ingredientID, quantity)
deleteMenuItem(MenuItemName)
deleteMenuItem(MenuItemName)
deleteMenuItem(MenuItemID)

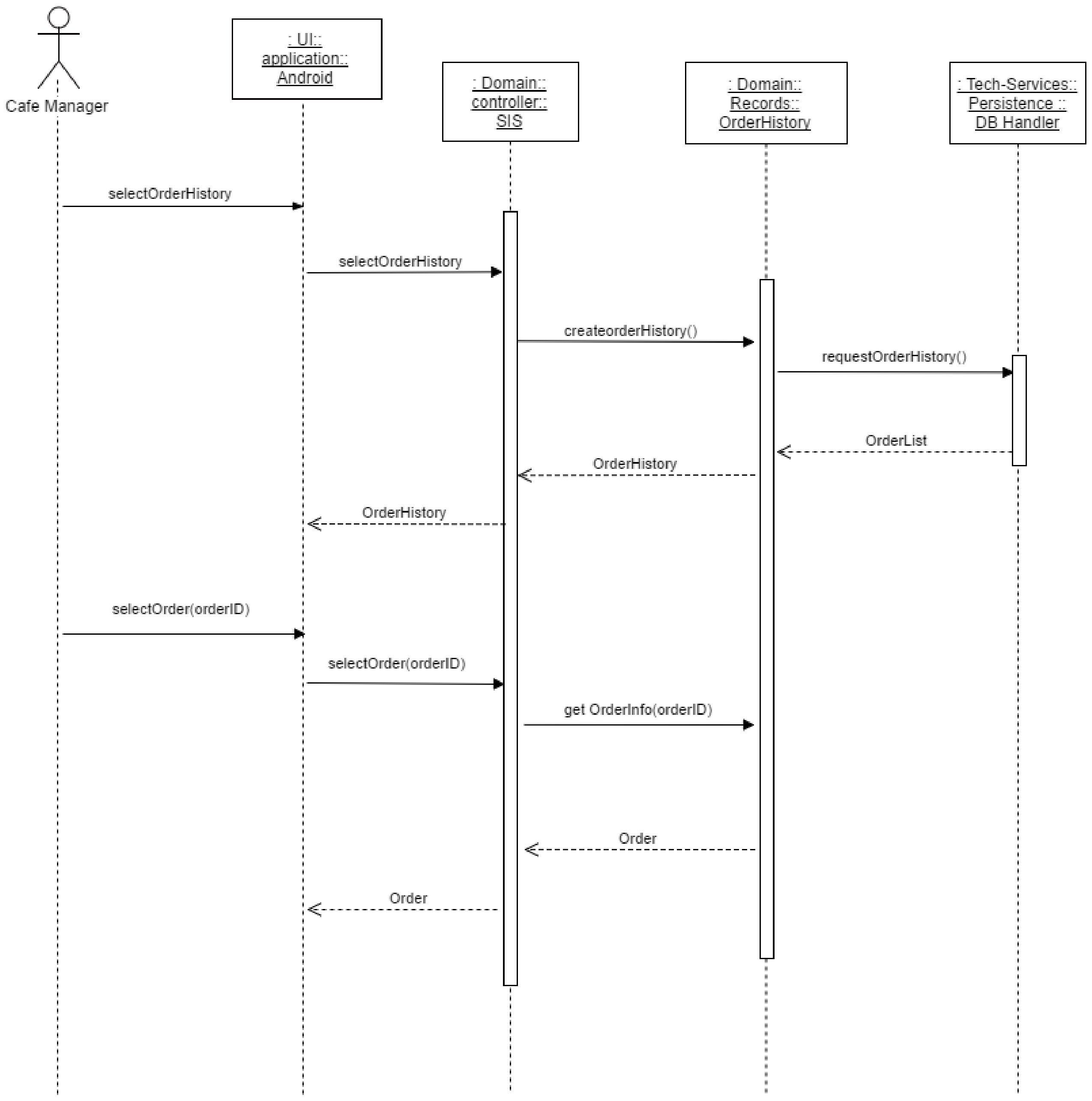
<Figure> Manage menu & ingredients Use Case View

7.7.2. Set ordering rule


Cafe Manager
: UI:: application:: Android
: Domain:: controller:: SIS
: Domain:: Records:: AutoOrderRule
: Tech-Services:: Persistence :: DB Handler
setOrderingRule()
createOrderRule()
createOrderInfo()
requestOrderRuleInfo()
OrderRule
OrderRule
OrderRule
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)
setOrderingtime(autoOrderDay, autoOrderTime)

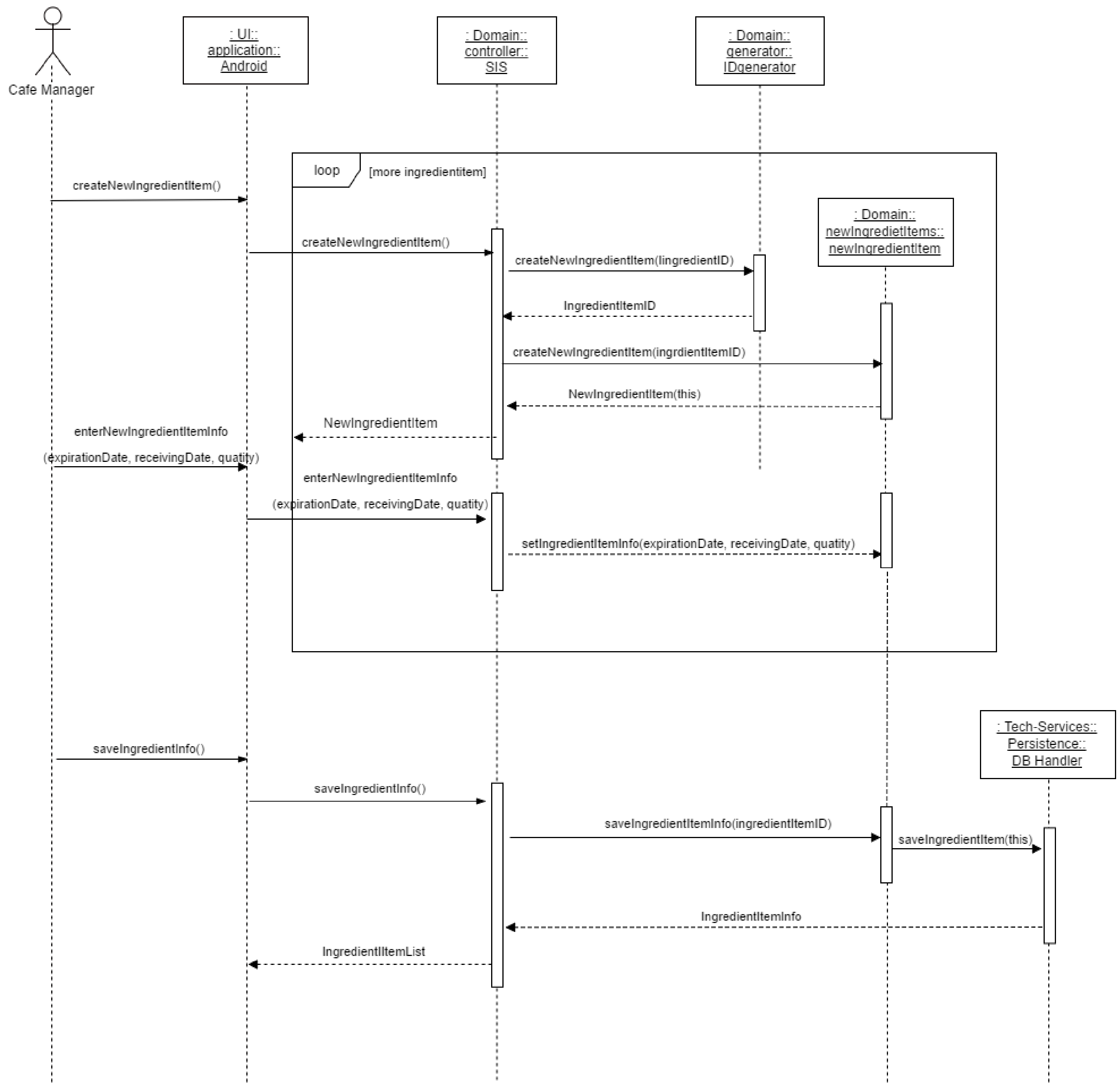
<Figure> Set ordering rule Use Case View

7.7.3. Show order history



<Figure> Show order history Use Case View

7.7.4. Manage inventory

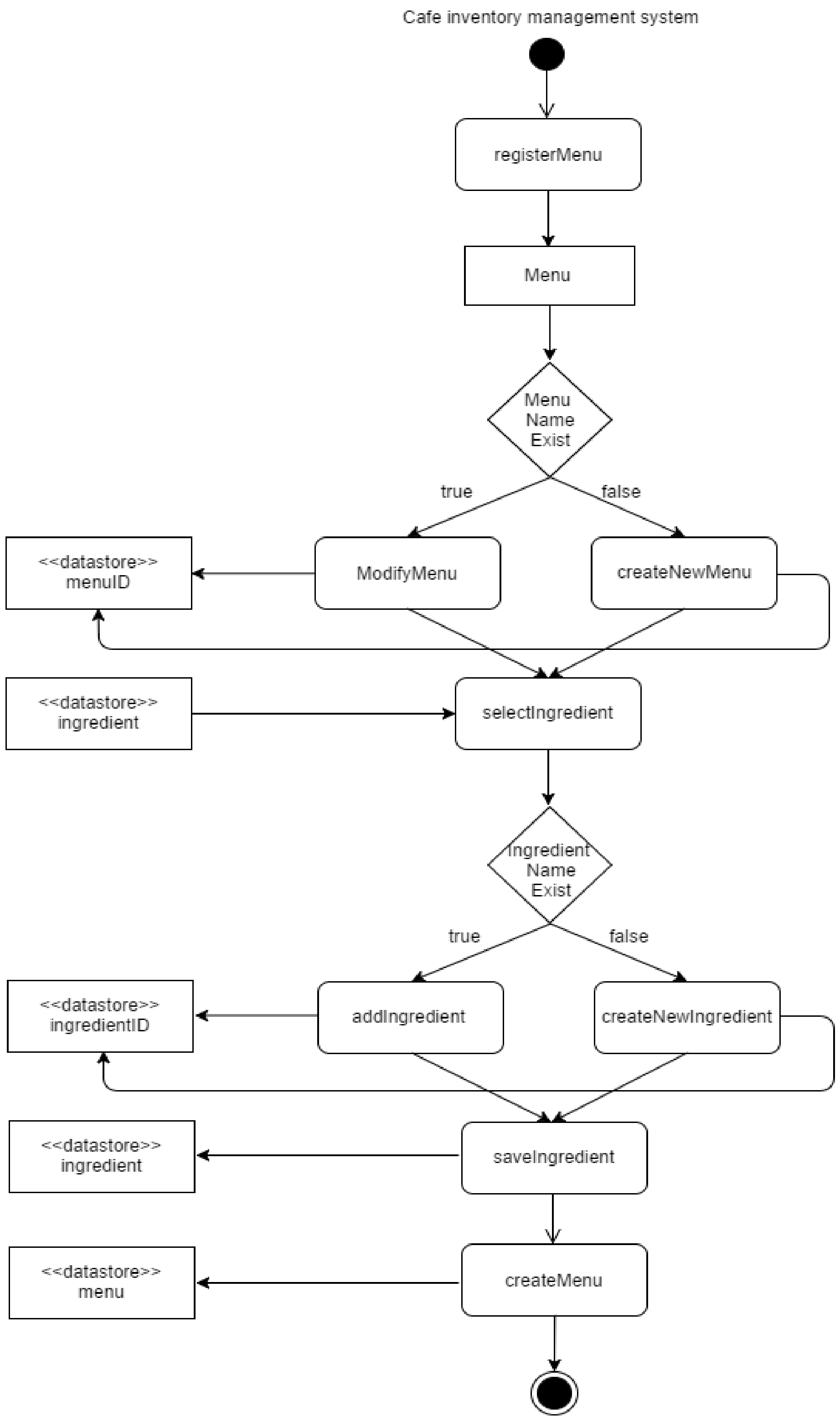


Cafe Manager
: UI:: application:: Android
: Domain:: controller:: SIS
: Domain:: generator:: IDgenerator
: Domain:: newIngredietItems:: newIngredientItem
: Tech-Services:: Persistence:: DB Handler
loop
[more ingredientitem]
createNewIngredientItem()
createNewIngredientItem()
createNewIngredientItem(IingredientID)
IngredientItemID
createNewIngredientItem(ingrdientItemID)
NewIngredientItem(this)
NewIngredientItem
enterNewIngredientItemInfo (expirationDate, receivingDate, quatity)
enterNewIngredientItemInfo (expirationDate, receivingDate, quatity)
setIngredientItemInfo(expirationDate, receivingDate, quatity)
saveIngredientInfo()
saveIngredientInfo()
saveIngredientItemInfo(ingredientItemID)
saveIngredientItem(this)
IngredientItemInfo
IngredientIItemList

<Figure> Manage inventory Use Case View

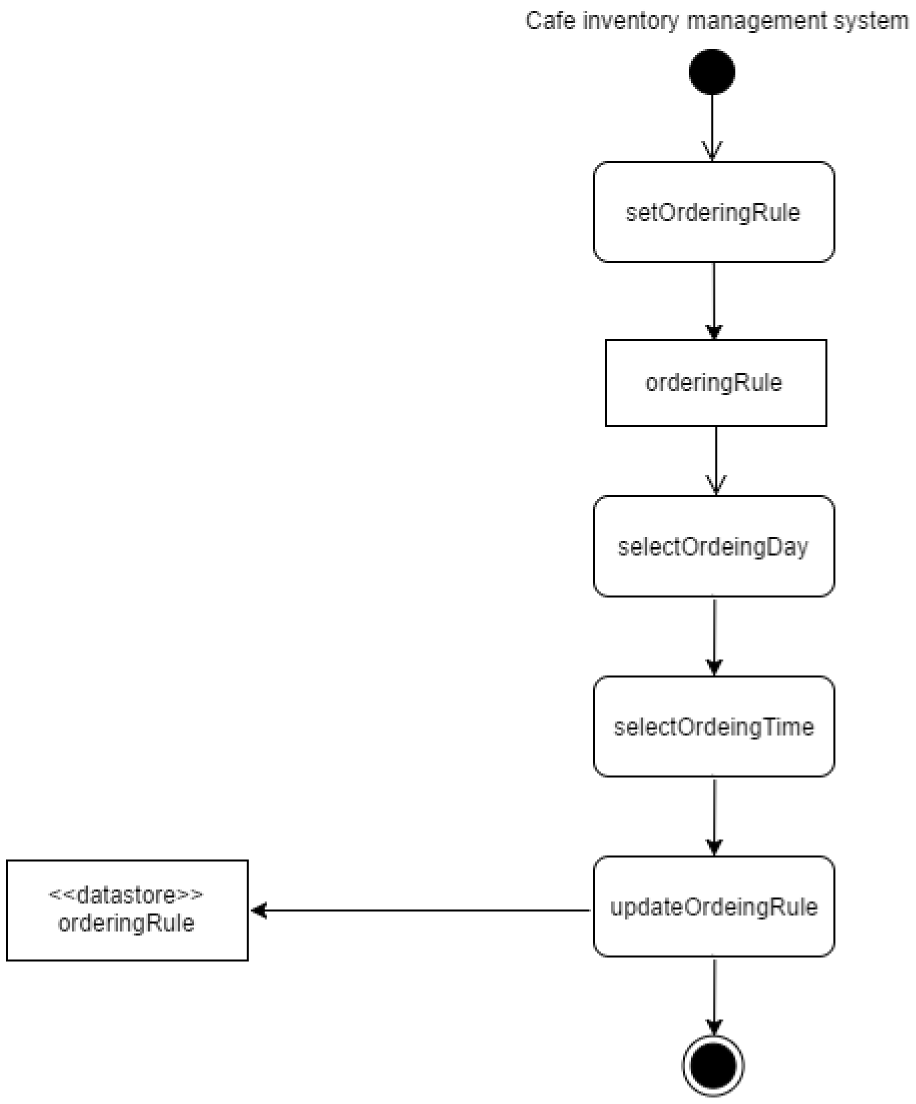
7.8. Data View

7.8.1. Manage menu & ingredients



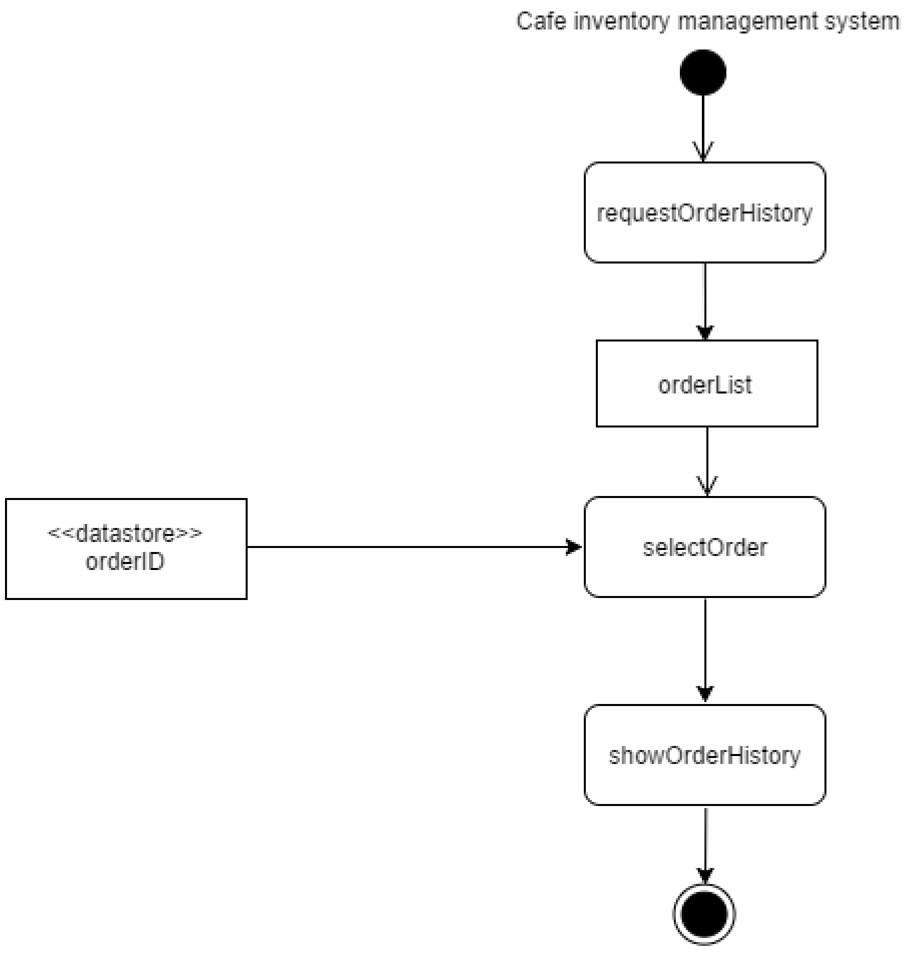
<Figure> Manage menu & ingredients Data View

7.8.2. Set ordering rule



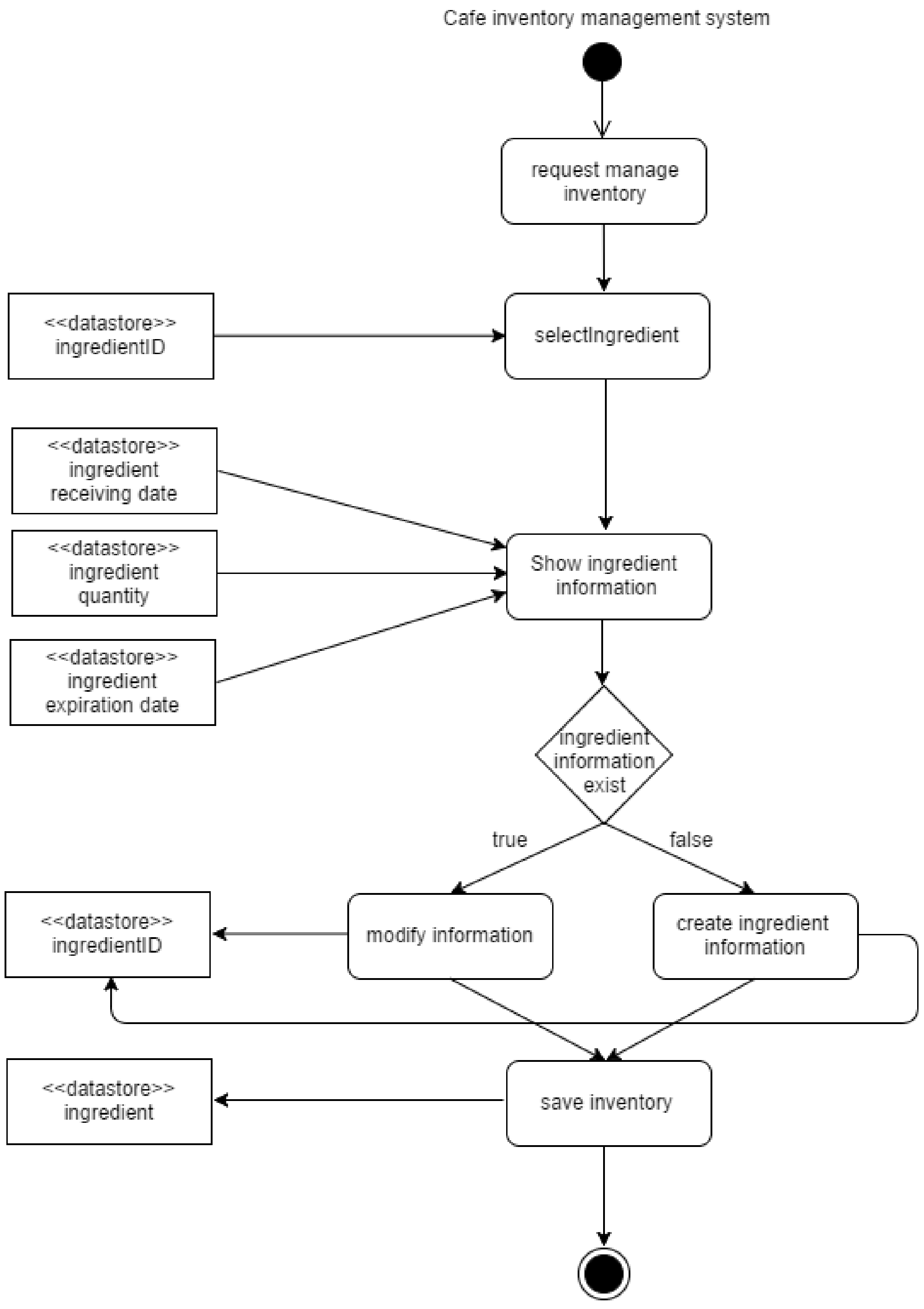
<Figure> Set ordering rule Data View

7.8.3. Show order history

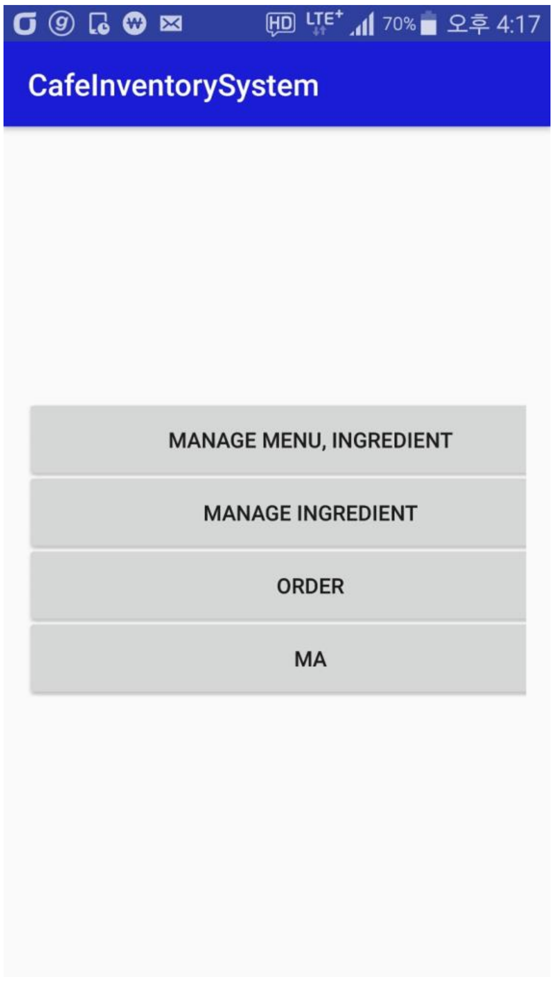


<Figure> Show order history Data View

7.8.4. Manage inventory


Cafe inventory management system
request manage inventory
<<datastore>> ingredientID
selectIngredient
<<datastore>> ingredient receiving date
<<datastore>> ingredient quantity
<<datastore>> ingredient expiration date
Show ingredient information
ingredient information exist
true
false
<<datastore>> ingredientID
modify information
create ingredient information
<<datastore>> ingredient
save inventory

<Figure> Manage inventory Data View



7.9. Implementation View

▶This is main page. You can manage menus and materials, and manage inventory.

8. Conclusion

8.1. Objectives

- Developed a cafe inventory management system application service that helps store manager manage food inventory and expiration date easily and accurately, and enables food suppliers to easily manage deliveries by store. This service can be extended to such restaurants as well as a cafe using a food ingredient.

8.2. Current State of Design & Implementation

1) Design

- Design is 80% done.
  Because our design doesn't include Security/Implementation/Development view.
  - UC1. Manage menu & ingredients
  - UC2. Order Ingredients
  - UC2-1. Set ordering Rule
  - UC2-2. Show orderHistory
  - UC3. Manage Inventory

2) Implementation

- Implementation is 40% done. We implemented following use case instances.
  - UC1. Manage menu & ingredients
  - UC2. Order Ingredients
  - UC2-1. Set ordering Rule
  - UC2-2. Show orderHistory
  - UC3. Manage Inventory
- Implementation in detail :

  @ Manage menu & ingredients

  1) 유저가 메뉴와 재료를 관리하고싶다면 “Manage menu & ingredients”를 선택한다.

  2) 시스템은 메뉴 추가, 수정, 삭제와 재료 추가, 수정 삭제 메뉴를 보여준다.

  3)

  @ Order Ingredients

  1) 유저가 발주를 하고싶다면 “Order Ingredient”를 선택한다.

  2) 이 메뉴에서 발주 룰을 정하고 싶다면 “Set Ordering Rule”을 선택 할 수 있고, 발주된 내역을 보고싶다면 “Show OrderHistory”를 선택할 수 있다.

  3)

  @ Manage Inventory

  1)

  2)

  3)

8.3. Addition Work(Refine the design)

- None.

8.4. Lesson learned through this project

Through this class, we learned objecanalysis t oriented analysis and design. First, we discussed about the idea. And then, we started to analyze and design step by step. Because it was our first experience of object oriented and design, we made a lot of mistakes from the beginning. So we had meeting whenever we have time. We continuously modified what we were doing. Thanks to all members effort, we could complete our project. Of course, the result is not perfect, but we learned a lot from it. We understood the procedure of object oriented analysis & design. We learned how to work in team. We learned a lot by trial and error. So, we believe that we can do better next time.

9. References

- 『Applying UML And Patterns, 3rd Edition』, Craig Larman, Addison Wesley Professional
- 통계청 *Statistics Korea*

10. Appendix

A. Glossary

<Revision History>

Version	Date	Description	Author
Inception draft	Mar 22, 2017	First draft. To be refined elaboration step.	Team 공조
Elaboration 1	Apr 29, 2017	Elaboration 1. Refined version to the previous version.	Team 공조
Elaboration 2/Final	June 05, 2017	Final draft	Team 공조

<Definitions>

Term	Definition and Information	Format	Validation Rules	Aliases
Store Manager	person who manages inventory in the store.	-	-	-
Supplier	person who supplies inventory to the Store Manager.	-	-	-
Administrator	person who manages this system.	-	-	-
SIS	Cafe Management Inventory System	-	-	-
Holding Time	The holding time is the period that can be kept from the date of receipt of the ingredient	-	-	-
Order calculation rule	the rule to calculate how many ingredients are needed to oreder (minimum quantity - total quantity)			

<Table 10> Glossary.

<Business Rule>

1. When the specified autoOrderRule, autoOrderTime, the system automatically orders the amount of material calculated according to the defined order calculation rule. 정해진 autoOrderRule . autoOrderTime 이 되면 시스템은 정해진 order calculation rule 에 따라 계산된 양만큼의 재료를 자동으로 발주한다.
2. When registering autoOrderRule and autoOrderTime, autoOrderFlag is automatically turned on.
3. In the case of automatic ordering, the supplier system automatically makes an order through the automatic payment system upon ordering. 자동 발주의 경우 supplier system 으로 발주와 동시에 자동 결제 시스템을 통해 자동으로 결제한다.
4. P.O.S System shares database with Café Inventory Management System and receives information about menus and materials. P.O.S System 은 Cafe Inventory Management System 과 데이터베이스를 공유하여 메뉴 및 재료에 대한 정보를 제공받는다.
5. The supplier system covers all suppliers so that what suppliers supply what materials does not affect the system. Supplier system 은 모든 supplier 를 커버함으로써 어떠한 supplier 가 어떠한 재료를 공급하는가는 시스템에 어떠한 영향도 주지않는다.
6.