Final Project Report

- Subject : Cafe Ajou
- Team : 학교가고싶다

김동현 amijo998@ajou.ac.kr
나용성 nayong2017@ajou.ac.kr
박민지 alswlkku@ajou.ac.kr
신채연 cheayeon33@ajou.ac.kr
전상범 beom115@ajou.ac.kr

Table of Contents

■ Figure

■ Table

❑ **Table49** Revision History

1. Vision

1.1. Introduction

Figure 1 *Cafe Usage Trend Report 2020 by Open survey*

According to the “Cafe Usage Trend Report 2020” released by Open Survey, Those in their 20s~30s had a high experience rate with 76.4% of non-face-to-face cafe orders. And People in their 20s tried to use the non-face-to-face ordering method when they wanted to think enough about menu choices by referring to detailed menu images. Currently, many Apps are in operation on the school side to provide convenience to students. However, there is a time to wait for orders and inconvenience of using the cafe between ordinary users and students studying in the cafe has not been resolved. Therefore, we would like to implement the ‘non-face-to-face cafe order’ system mentioned above. This system helps Customer reduce unnecessary time by allowing them to order drinks at any time they want, and also reduces the inconvenience of having to find another cafe because there are no seats available through seat reservation. It also helps store convenience by implementing functions that enable efficient management of the store not only from the customer but also from the store side.

1.2. Positioning

1.2.1. Business Opportunity

Customer can reduce unnecessary time by using a system that sets the time to pick up rather than a system requires them to make orders on their own feet. In addition, it can provide convenience to

Customer through a system that allows them to reserve seats together while ordering drinks by supplementing the problem of having to visit person every time to see if there are any seats available or not. This could lead to increased inflow into Customer's store and to increased store profits. Apart from these financial benefits, we thought that providing convenience to the store while providing statistics on customer sales using the app and the ability to easily add/delete menu without having to go through complicated procedures could lead to improved quality of service for the customer. Therefore, it was determined that if good results could be achieved in an efficient way, better results could be provided for both sides.

1.2.2. Problem Statement

Due to the nature of the program, which aims to reduce waiting times, the current amount of orders in the cafe(offline orders, not apps) has a significant impact. Without information about these latencies, programs can't operate efficiently. Even if minimized, offline orders would still exist and as much as possible can be linked to the Pos system to provide the expected waiting number of person. But, we thought it would be difficult to provide accurate waiting time considering the development environment and the given development time.

1.2.3. Product Position Statement

Similar programs exist outside the school at the other franchise cafes (like STARBUCKS). However, we have a competitive edge in the 'use of school members' and 'shorT distances' than other cafes outside of school. This competitiveness leads to the need for this program. Store can also reduce the burden of competition from other cafes.

1.3. Stakeholder Description

1.3.1. Stakeholder (Non-User) Summary

- **System Manager :** To maintain the System. System should know errors quickly. so Improve usability by correcting errors in a short time.
- **Pos System:** For overall off-lin sales, Manage menu and order management.
- **In-app Payment System:** To pay when the reservation is completed.

1.3.2. User Summary

- **Customer** : cafe users can check the status of their seat reservation with the system before visiting the site to reduce the hassle of having to check their seats by visiting the site in person. In addition, customers can reserve the cafe drinks in advance so they can receive their drinks as soon as they arrive at the cafe.

- **Manager** : Through this program, Manager can expect to improve customer convenience and it leads to increase the profits. Manage menu function allows managers to quickly add or delete menu in store, which also increases convenience.

1.4. Product Overview

1.4.1 Product Perspective

It is installed within the application. It will be used by the members of Ajou university who want to order a drink and also reserve a seat at the cafe in advance. I will provide services to users, and collaborate with other systems, as indicated in the form shown below.

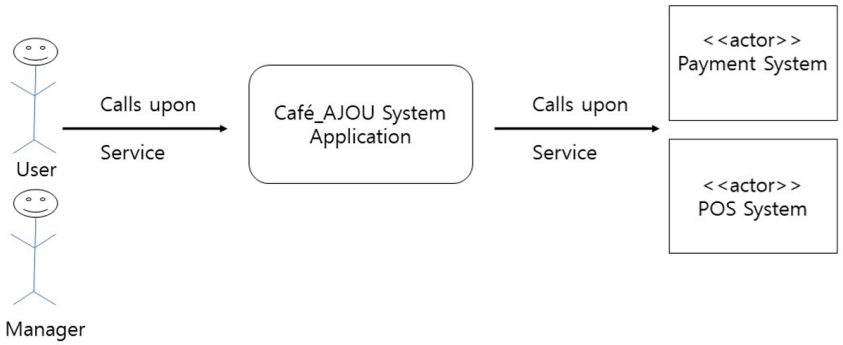


Figure2 *Product Perspective Scheme*

1.4.2 Summary of benefits

Supporting Feature	**Stakeholder benefit**
By adding or modifying or deleting menus, the system updates the changes also applied to the POS system.	Manager can manage menu information.
By entering the order/reservation information, the system guarantees a seat thatand pre-order a drink.	User can order a drink and also reserve a seat in advance at the cafe.
The system collects customer purchase data and produces statistics on monthly/weekly basis that the manager wants for each beverage.	Manager can utilize customer data to generate increased revenue and manage inventory effectively.

Table1 *Summary of benefits*

1.5. Summary of System Features

- Manager who already has authorization can manage menus including add, delete and modify menus.
- Members of Ajou university can order a drink any time they want and know how many seats are available at the cafe.
- User can pay through In-app payment if the user wants to pay in advance, not on-site payment.

2. Requirement

2.1. use case Diagram

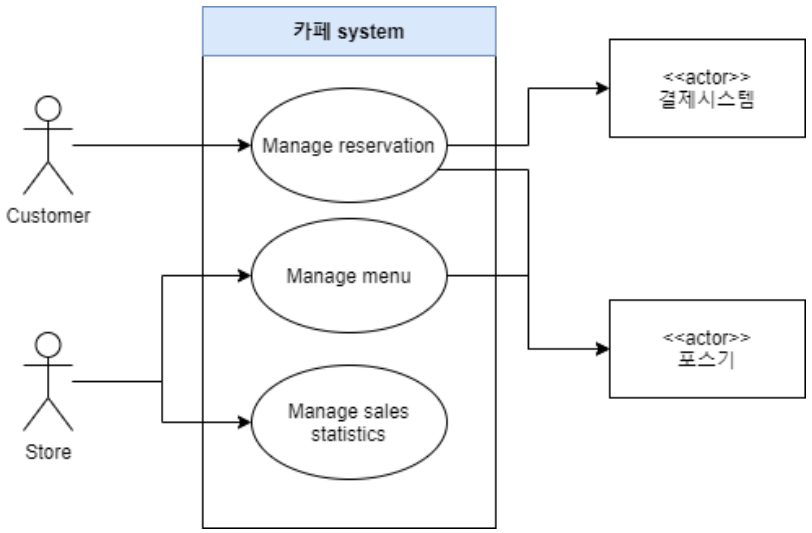


Figure 3 *Use Case Diagram*

2.2. Use Case Text

1. Manage reservation
2. Manage menu
3. Manage sales statistics

Table2 *Use Case List*

2.2.1 Functional Requirement

2.2.1.1 Manage reservation

First user selects one of three optional cafes at Ajou university and also selects takeaway or eat-in. At the next step, the user can designate time when to visit the cafe for the beverage and he can reserve a seat at the cafe with limited time. Maximum 3 times he can extend the seat with the time limit, 1h 30m(this is the course time). If the user doesn't show up at the cafe, he can't reserve a seat for 3 days as a penalty. If he pre-order a drink and a seat but changes may occur so he can modify or cancel his reservation before 30m of the reservation.

2.2.1.2 Manage menu

We need a 'Manage menu System' to make users know the menu information/list of the cafe. Using this system, manager can add new menus, delete discontinued products and modify the information of existing menus and state of menus for indicating out of order for that day.

2.2.1.3 Manage sales statistics

Based on sales of customer purchase data through the application, manager can see the statistics of sales so it helps for manager to generate increased revenue.

2.2.2 Use Case Model

Use Case 1. Manage reservation

- Level : User-goal

- Primary Actor : Customer

- **Precondition**
  - Customer는 인증이 끝난 상태로 기능에 접근한다.
  - 메뉴에는 카테고리 별 음료 이름, 가격이 저장되어 있다.
  - 시스템에 좌석 별 예약된 시간이 저장되어 있다.

- Scenario Flow :

1. Customer가 예약하기 기능을 시작한다.
2. 이용하고자 하는 매장을 선택한다.
3. 카테고리 별 음료 이름, 가격을 보고 음료를 선택한다.
   Customer repeats steps 3 until indicates done.
4. 좌석예약에 대한 여부를 선택한다.
5. 시스템이 모든 지정 예약석을 보여준다.

6. Customer가 이용하기를 원하는 시간을 입력한다.
7. 시스템은 사용자의 입력에 따라 이용 가능한 예약석을 표시한다.
8. Customer가 예약 가능한 좌석을 선택한다.
9. 주문된 정보를 시스템에게 전송한다.
10. 계산된 요금의 총액을 알려주고 결제를 요청한다.
11. 시스템이 판매를 완료하고 판매 및 결제 정보를 외부 회계 시스템으로 보낸다.
12. 시스템이 Customer에게 주문 완료 메시지를 보낸다.
13. 시스템은 store number를 통해 주문 내역을 store에게 전송한다.
14. 시스템은 주문한 내역을 저장한다.

- Extension

1a. 예약수정을 선택한 경우

1. Customer가 예약 수정 기능을 사용한다.
2. 시스템이 사용자의 예약목록을 보여준다.
3. Customer가 수정할 예약을 선택한다.

2a. 선택한 예약이 '매장에서 먹기'인 경우

1. 시스템이 예약정보를 보여준다.
2. Customer가 수정하고자 하는 정보를 선택한다.

2a. 메뉴를 수정하고자 하는 경우

1. Customer가 수정하고자 하는 메뉴를 선택한다.

1a. Customer가 수량의 수정을 원할 경우

1. Customer가 메뉴의 수량을 조정한다.

2. 시스템이 선택가능한 메뉴목록과 가격을 보여준다.
3. Customer가 원하는 다른 메뉴를 선택한다.

2b. 이용시간, 좌석 변경을 원하는 경우

1. 시스템이 예약 가능한 시간 현황을 보여준다.
2. Customer가 원하는 시간대를 선택한다.
3. 시스템이 선택한 시간대의 예약가능 좌석현황을 보여준다.
4. Customer가 원하는 좌석을 선택한다.

2b. 선택한 예약이 '테이크 아웃'인 경우

1. 시스템이 예약정보를 보여준다.
2. Customer가 수정하고자 하는 정보를 선택한다.

2a. 메뉴를 수정하고자 하는 경우

1. Customer가 수정하고자 하는 메뉴를 선택한다.

1a. Customer가 수량의 수정을 원할 경우

1. Customer가 메뉴의 수량을 조정한다.

2. 시스템이 선택가능한 메뉴목록과 가격을 보여준다.
3. Customer가 원하는 메뉴를 선택한다.

2b. 수령시간을 수정하고자 하는 경우

1. Customer가 원하는 수령시간을 선택한다.

4. 시스템이 수정된 예약의 총액을 알려주고 재결제를 요청한다.
5. 시스템이 기존의 결제를 취소하고 재결제된 정보를 외부 회계 시스템으로 보낸다.
6. 시스템은 store number를 통해 예약 수정 내역을 store에게 전송한다.
7. 시스템에 수정된 예약내용이 저장된다.

1b. 예약취소를 선택한 경우

1. Customer가 예약 취소 기능을 사용한다.
2. 시스템이 사용자의 예약목록을 보여준다.
3. Customer가 삭제하고자 하는 예약을 선택한다.
4. 시스템에서 선택된 예약의 결제를 취소하고 취소 정보를 외부 회계 시스템이 보낸다.
5. 시스템은 store number를 통해 예약 취소 내역을 store에게 전송한다.
6. 시스템이 저장된 예약내역과 좌석 정보를 삭제한다.

1c. 좌석 예약연장을 선택한 경우

1. 사용자가 예약 연장 기능을 선택한다.
2. 시스템이 이용시간을 연장해주고, 연장된 시간을 좌석 현황에 반영한다.
3. 시스템이 변경된 예약 정보를 사용자에게 보여준다.

5a. 좌석예약을 하지 않는 경우

1. 사용자가 음료를 받고자 하는 시간을 선택한다.
2. 계산 된 요금의 총액을 알려주고 결제를 요청한다.
3. 시스템이 판매를 완료하고 판매 및 결제 정보를 외부 회계 시스템으로 보낸다.
4. 시스템이 영수증과 주문 내역을 제시한다.
5. 시스템은 store number를 통해 주문 내역을 store에게 전송한다.
6. 시스템이 예약내용을 저장한다.

***Table3** [UC1] Manage reservation(fully dressed format)*

Use Case 2. Manage menu

- Level : User-goal

- Primary Actor : Manager

- **Precondition**

- 매니저는 로그인이 완료 된 상태다.
- 시스템에 기본 카테고리 리스트가 사전에 저장되어 있다.
- 시스템이 포스 시스템이 연동되어 있다.

- Scenario Flow :

1. Manager가 메뉴 추가기능을 시작한다.
2. 시스템이 선택가능한 카테고리 리스트를 보여준다.
3. Menu가 추가될 카테고리를 선택한다.
4. 추가 할 메뉴의 itemID, name, price, description을 입력한다.
5. 시스템이 입력된 메뉴의 정보를 보여준다.
6. Manager가 추가될 메뉴정보를 확인하고 등록한다.
7. 요청된 사항이 포스시스템에 등록된다.

- Extension

1a. modify Menu일경우

1. Manager가 메뉴 수정기능을 시작한다.
2. Manager는 수정할 메뉴의 ItemID을 입력한다.
3. 시스템이 선택가능한 카테고리 리스트를 보여준다.
4. Menu가 수정될 카테고리를 선택한다.
5. 메뉴의 itemID, name, price, description을 수정한다.
6. 시스템이 수정 메뉴의 정보를 보여준다.
7. Manager가 수정될 메뉴정보를 확인하고 등록한다.
8. 요청된 사항이 포스시스템에 등록된다.

1b. delete Menu일 경우

1. Manager가 메뉴 삭제기능을 시작한다.
2. 삭제할 메뉴의 ItemID을 입력한다.
3. 시스템이 삭제될 메뉴의 정보를 보여준다.
4. Manager가 수정될 메뉴정보를 확인하고 삭제요청을 한다.
5. 요청된 사항이 포스시스템에 등록된다.

1c. add Category일 경우

1. Manager가 카테고리 추가를 시작한다.
2. Manager가 추가할 카테고리의 categoryName과 categoryID를 입력한다.
3. 요청된 사항이 포스시스템에 등록된다.

***Table4** [UC2] Manage menu(fully dressed format)*

Use Case 3. Manage Sales statistics

- Level : User-goal

- Primary Actor : Manager

- **Precondition**
  - 매니저는 로그인이 완료 된 상태다.
  - 시스템에 카페 내 판매량이 사전에 저장되어 있다.

- Scenario Flow :

1. Manager가 판매 분석 기능을 선택한다.
2. 시스템이 카테고리를 보여준다. / 카테고리 : 성별, 시간대별
3. 원하는 카테고리를 선택한다.
4. Manager가 원하는 날짜를 입력한다. (ex. 190710 ~ 190810)
5. 시스템이 날짜에 해당하는 분석 내용을 보여준다.
   Customer and System repeats steps 4-5 until indicates done.
6. Manager가 판매 분석 기능을 종료한다.

6a. Manager가 분석 내용을 영수증으로 출력하길 원할 때

1. 시스템이 출력시스템과의 연결을 확인한다.
2. 카테고리에 따른 분석 내용을 출력시스템에 전송한다.
3. 출력 시스템이 출력한다.

***Table5** [UC3] Manage sales statistics(fully dressed format)*

2.2.3 Non-Functional Requirement

Functionality

- **Logging and Error Handling**
  - Users(Customer and Manager) authentication is required.
  - Return to the previous page in case of an error.
- **Security**
  - Users without manager authentication cannot access Store Management.

Usability

- **Human Factor**

Users should see it on their mobile phone screen because it is aimed at developing Apps.

- Accurate icon shape and menu names should be used so that customers are not confused.
- There are not only students in the school but also people who are old, So app help to adjust the size of the letters through the setting.

Reliability

- **Recoverability**
  - all data on orders and menu information should be backed up to the database once every 12 hours.

Performance

- The initial main page should be displayed within 10 seconds of access.
- The message about the user's error should be sent within 3 seconds.

Supportability

- **Adaptability**

  Order information coming in real time via APP is forwarded to the store.

- **Configurability**

  Consider UX/UI to facilitate access to information.

3. Analysis Modeling

3.1. Domain Model Diagram

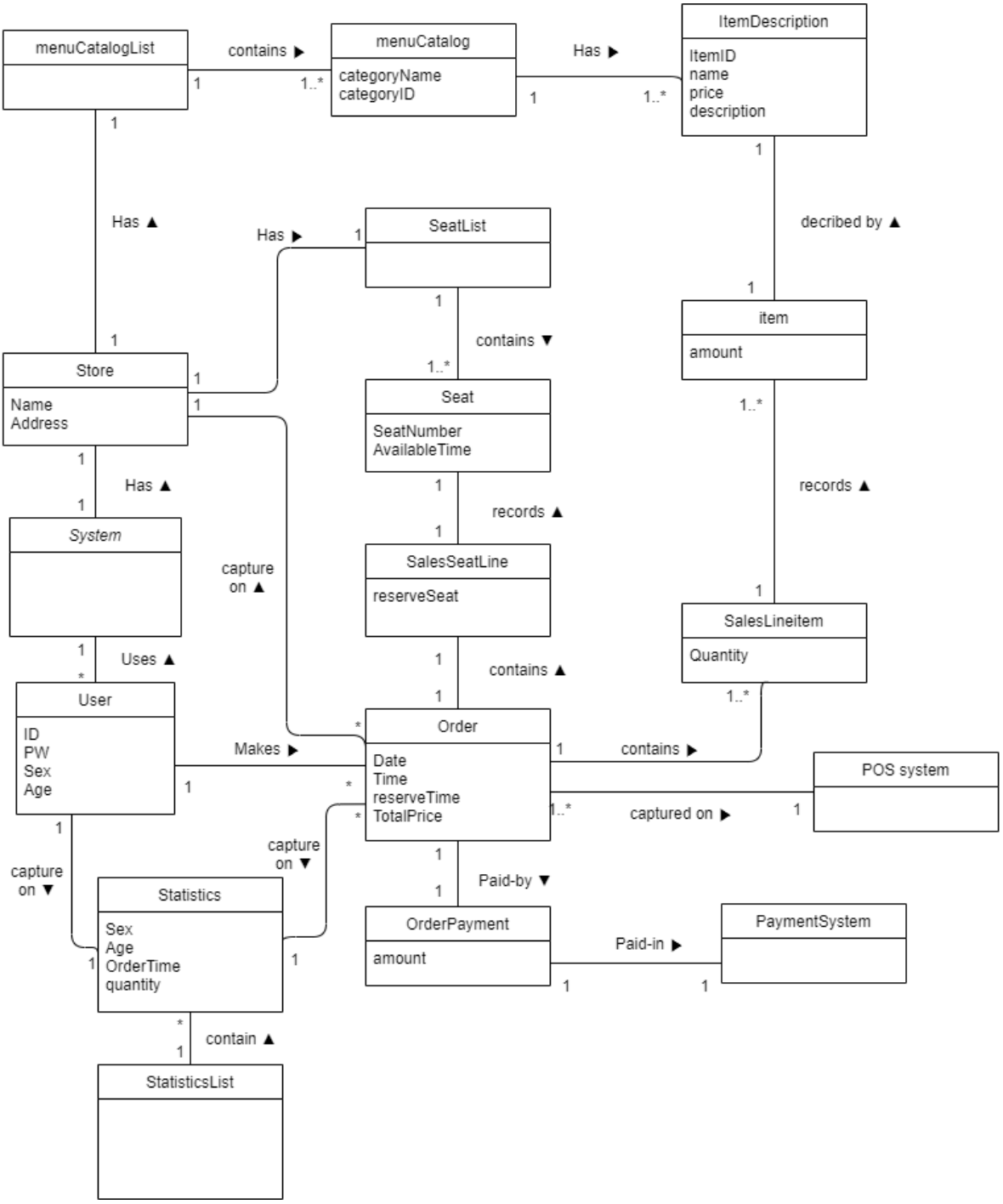


Figure 4 *Use Case 1 Domain Model Diagram*

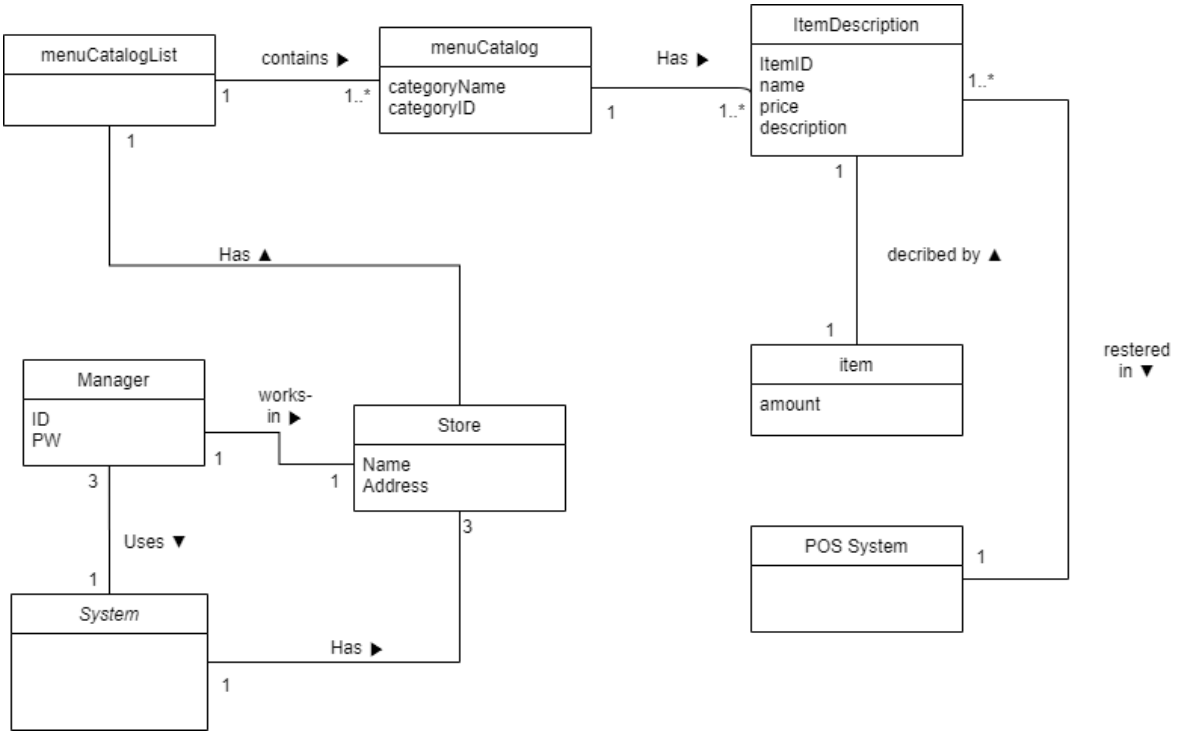


Figure 5 *Use Case 2 Domain Model Diagram*

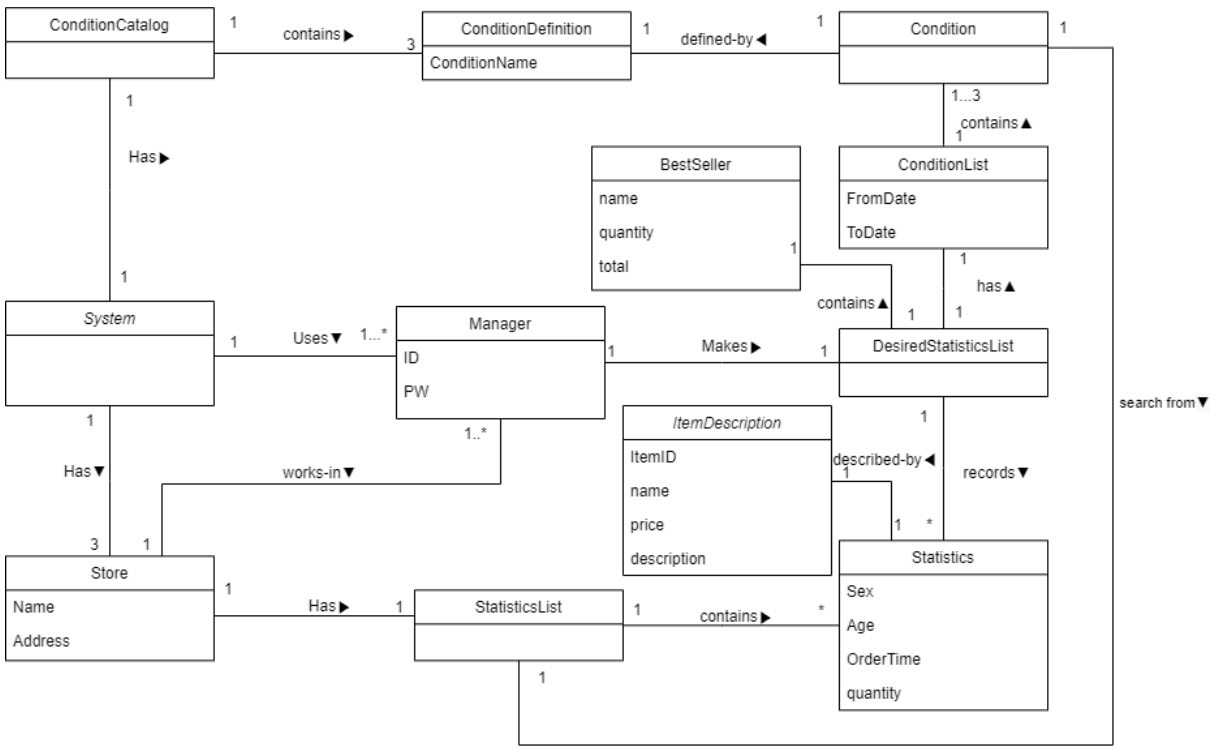


Figure 6 *Use Case 3 Domain Model Diagram*

3.2. System Sequence Diagram

3.2.1. Use Case 1 : Manage reservation

3.2.1.1. UC1 Manage reservation : System Sequence Diagram

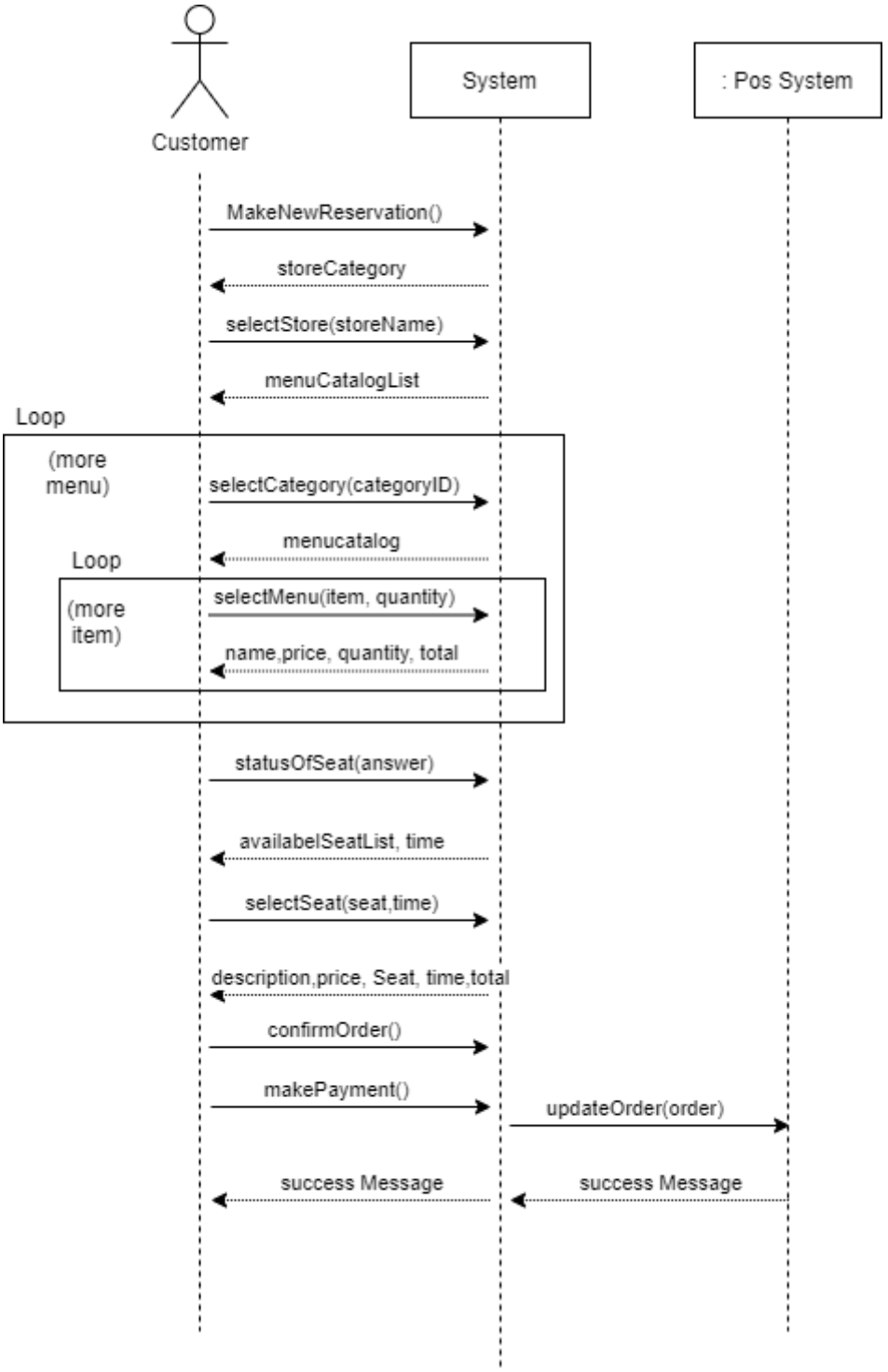


Figure 7 *UC1 Manage reservation : System Sequence Diagram*

3.2.1.2. UC1 Manage reservation : Brief description of system operation

makeNewReservation()	새로운 예약을 생성하는 요청을 시스템에 전송하고, 시스템은 예약할 Store를 선택할 수 있는 StoreCategory를 반환한다.
selectStore(storeName : String)	Customer는 예약을 원하는 store에 대한 정보를 담아 시스템에 전송한다. 그 정보를 받은 시스템은 메뉴 카타로그 리스트를 반환해준다.
selectCategory(categoryID : int)	시스템을 통해 메뉴 카타로그 리스트를 받은 Customer는 원하는 음료카테고리류를 선택한다. 시스템은 그 카테고리에 맞는 음료 메뉴들을 보여준다. 메뉴를 모두 고를 때까지 반복할 수 있다.
selectMenu(item : item, quantity : int)	Customer는 원하는 음료의와 주문한 수량을 선택한다. 원하는 메뉴의 개수만큼 반복할 수 있다.
statusOfSeat(answer : boolean)	Customer는 좌석을 예약할지에 대한 여부를 입력한다. 만약, 좌석예약을 원하지 않는다면 시스템은 픽업을 원하는 시간만 입력받고 좌석예약을 원한다면 시스템은 시간과 좌석을 입력받게 된다.
selectSeat(Seat : Seat, time : time)	Customer가 원하는 좌석과 예약할 시간을 System에게 전송한다.
confirmOrder()	주문을 확정하면 시스템은 주문내용을 이용해 통계를 DB에 저장한다.
makePayment()	price total을 바탕으로 결제를 수행한다.

***Table6** [UC1] Manage reservation : Brief description of system operation*

3.2.1.3. UC1 Manage reservation : Operation Contracts

Contract CO1 : makeNewReservation
Operation : makeNewReservation()
Cross reference : Manage reservation
Precondition : none
Postcondition :

- A order instance o was created
- Attributes of o were initialized
- o was associated with a System.
- SalesLineItemList instance sliList was created.
- sliList was associated with o.

Table 7 *[UC1] - CO1 makeNewReservation*

Contract CO2 : selectStore

Operation : selectStore(storeName : String)

Cross reference : Manage reservation

Precondition :

- There is a Manage reservation underway.
- There is a Store object.

Postcondition :

- o was associated with a Store, based on storeName.

Table 8 *[UC1] - CO2 selectStore*

Contract CO3 : selectMenu

Operation : selectMenu(item, quantity : int)

Cross reference : Manage reservation

Precondition :

- There is a Manage reservation underway.

Postcondition :

- A SalesLineItem instance sli was created.(instance creation)
- sli was associated with o(association formed)
- sli.quantity became quantity. (attribute modification)
- sli was associated with Item.(association formed)
- sli was associated with sliList.(association formed)

Table 9 *[UC1] - CO3 selectMenu*

Contract CO4 : selectSeat

Operation : selectSeat(Seat, time)

Cross reference : Manage reservation

Precondition :

- There is a Manage reservation underway.

Postcondition :

- A SalesSeatLine instance ss was created.
- ss was associated with Seat.
- o was associated with ss.
- o.ReserveTime became time
- ss.reserveSeat became seatNumber

Table 10 *[UC1] - CO4 selectSeat*

Contract CO5 : confirmOrder

Operation : confirmOrder()

Cross reference : Manage reservation

Precondition :

- There is a Manage reservation underway.
- There is a StatisticsList instance sttList.

Postcondition :

- A Statistics instance s was created.
- s was associated with Order instance o.
- s was associated with User instance u.
- Attributes of s were initialized.
- s was associated with StatisticsList instance sttList.

Table 11 *[UC1] - CO5 confirmOrder*

Contract CO6 : makePayment

Operation : makePayment()

Cross reference : Manage reservation

Precondition :

- There is a Manage reservation underway.

Postcondition :

- A OrderPayment instance p was created. (instance creation).
- p was associated with o (association formed).
- Attributes of p were initialized.
- p was associated with POS adapter.

Table 12 *[UC1] - CO6 makePayment*

3.2.2. Use Case 2 : Manage menu

3.2.2.1. UC2 Manage menu : System Sequence Diagram

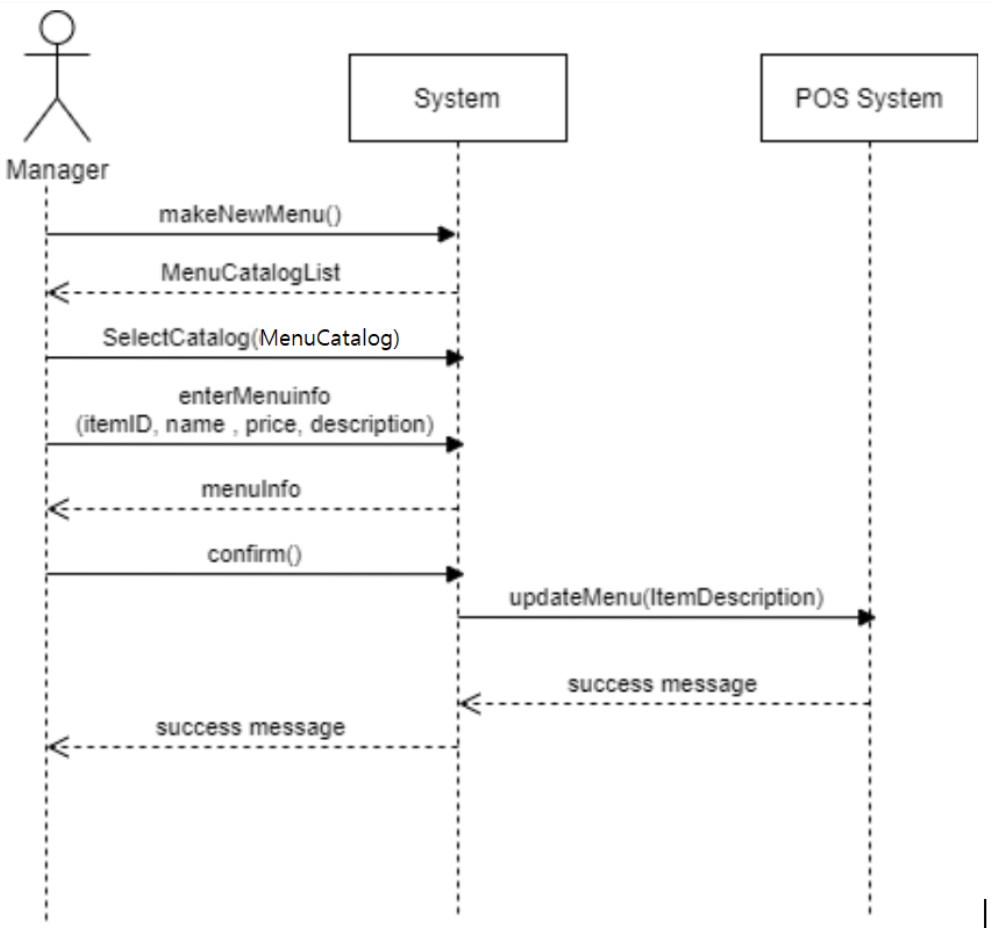


Figure 8 *UC2 Manage menu : System Sequence Diagram*

3.2.2.2. UC2 Manage menu : Brief description of system operation

makeNewMenu()	메뉴 추가 기능 요청을 시스템에 전송하고, MenuCatalogList를 반환받는다.
selectCatalog(mc : MenuCatalog)	메뉴를 추가할 메뉴카테고리(커피,주스,에이드와 같은)를 MenuCatalogList에서 선택한다. 시스템은 manager가 입력한 카탈로그에 추가될 음료의 inputable form 반환한다.
enterMenuinfo(itemID : String, name : String, price : int, description : String)	manager가 추가하고 싶은 음료의 정보와 가격을 inputable form에 맞춰 시스템에 전송한뒤 시스템은 manager에게 입력된 정보를 확인시킨다.

confirm()	manager는 입력된 정보를 확인한 후 메뉴등록을 확정한다. 시스템은 등록된 메뉴를 POS에 등록한 후 Manager에게 success message를 보낸다.

Table 13 *[UC2] Manage menu : Brief description of system operation*

3.2.2.3. UC2 Manage menu : Operation Contracts

Contract CO1 : makeNewMenu

Operation : makeNewMenu()

Cross reference : Manage menu

Precondition :

Postcondition :

- A Item instance ie was created.
- A ItemDescription instance itd was created.
- ie was associated with itd.
- Attributes of itd were initialized.

Table 14 *[UC2] - CO1 makeNewMenu*

Contract CO2 : selectCatalog

Operation : selectCatalog(mc : MenuCatalog)

Cross reference : Manage menu

Precondition :

- There is a Manage menu underway.
- There is a MenuCatalog instance mc.

Postcondition :

- itd was associated with mc.

Table 15 *[UC2] - CO2 selectCatalog*

Contract CO3 : enterMenuinfo

Operation : enterMenuinfo(itemID : int ,name : String, price : int, description : String)

Cross reference : Manage menu

Precondition :

- There is a Manage menu underway.

Postcondition :

- itd.itemID became itemID, itd.name became name, itd.price became price, itd.description became description.

***Table 16** [UC2] - CO3 enterMenuinfo*

Contract CO4 : confirm **Operation :** confirm() **Cross references :** Manage Menu **Preconditions :** - There is a make new Menu underway. **Postconditions :** - itd was associated with POS Adapter.

Table 17 [UC2] - CO4 confirm

3.2.3. Use Case 3 : Manage Statistics

3.2.3.1. UC3 Manage Statistics : System Sequence Diagram

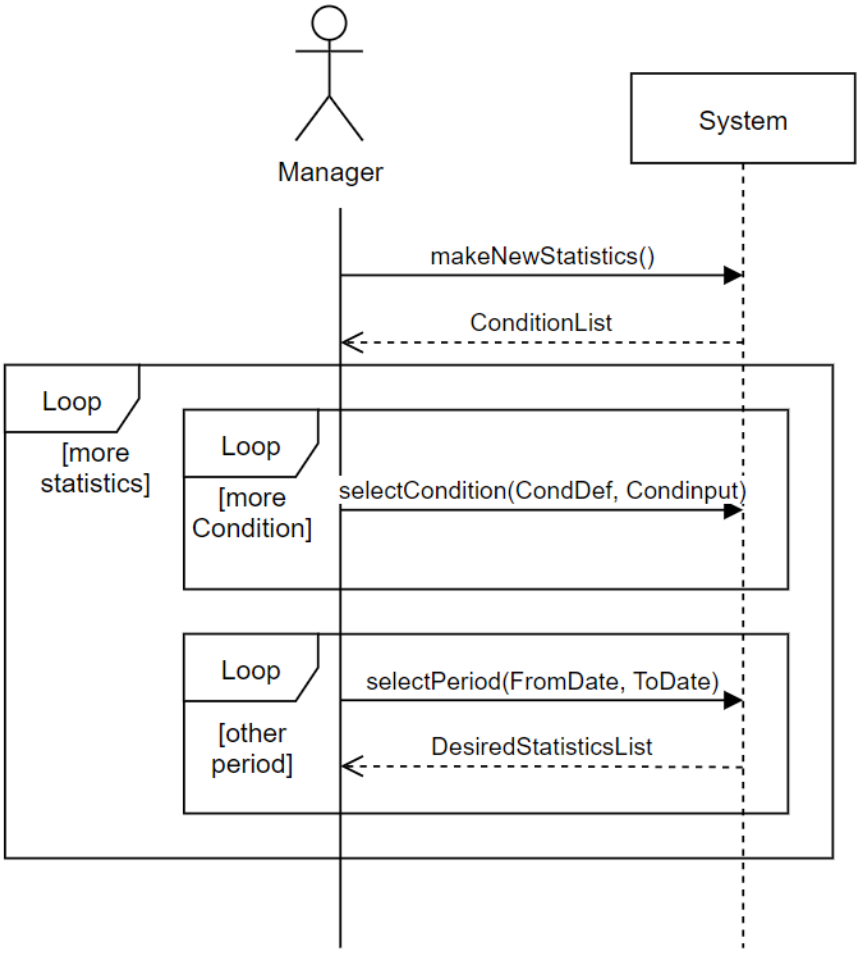


Figure 9 *UC3 Manage Sale Statistics : System Sequence Diagram*

3.2.3.2. UC2 Manage Sales Statistics : Brief description of system operation

makeNewStatistics()	판매 통계 요청을 시스템에 전송하고, ConditionList를 반환받는다.
selectCondition(CondDef, Condinput)	ConditionList에서 통계를 찾는데 사용할 조건을 입력해서 시스템에 전송한다.
selectPeriod(FromDate, ToDate)	분석을 시작하고 싶은 기간을 입력하여 시스템에 전송한다. 시스템은 매니저에게 조건과 기간에 맞는 통계를 종합해 반환한다.

Table 18 [UC3] Manage Sales Statistics : Brief description of system operation

3.2.3.3. UC3 Manage Sales Statistics : Operation Contracts

Contract CO1 : makeNewStatistics
Operation : makeNewStatistics()
Cross reference : Manage Sales Statistics
Precondition :
Postcondition :

- A DesiredStatisticsList instance dsl was created.
- dsl was associated with a System.
- A BestSeller instance bs was created.
- Attributes of bs were initialized.
- bs was associated with dsl.
- StatisticsList instance sttList was created.
- sttList was associated with dsl.
- ConditionList instance CondList was created.
- CondList was asscciated with dsl.

Table 19 [UC3] - CO1 makeNewStatistics

Contract CO2 : selectCondition
Operation : selectCondition(CondDef : ConditionDefinition, Condinput : String or int or Time)

Cross reference : Manage Statistics

Precondition :

- There is a make new Statistics underway.
- There is a ConditionDefinition instance CondDef.

Postcondition :

- Condition instance cond was created, based on CondDef
- cond was associated with CondList.

Table 20 [UC3] - CO2 selectCondition

Contract CO3 : selectPeriod

Operation : selectPeriod(FromDate, ToDate)

Cross reference : Manage Statistics

Precondition :

- There is a make new Statistics underway.
- There is a StatisticsList instance StoresttList.

Postcondition :

- CondList.FromDate became FromDate, CondList.ToDate became ToDate.
- Statistics instances in StoresttList were added to sttList, based on CondList.

Table 21 [UC3] - CO3 selectPeriod

4. Design Modeling

4.1. UC1 Manage reservation : Realization

4.1.1. Design Sequence Diagrams

상단 Operate Contract 중 selectStore(storeName : String),statusOfSeat(answer), selectCategory(categoryName : String)에 대한 Operation은 System이 User인 Customer에게 UI로 보여주고, 변수에 값을 입력하는 진행하는 단순한 단계이기 때문에 Sequence Diagram 단계에서 생략했다. 이외에는 Use case가 어떻게 구현되는지 인지가 필요한 모든 Operation은 sequence diagram을 작성했다.

4.1.1.1. Operation 1: makeNewReservation()

4.1.1.1.1. Sequence Diagram

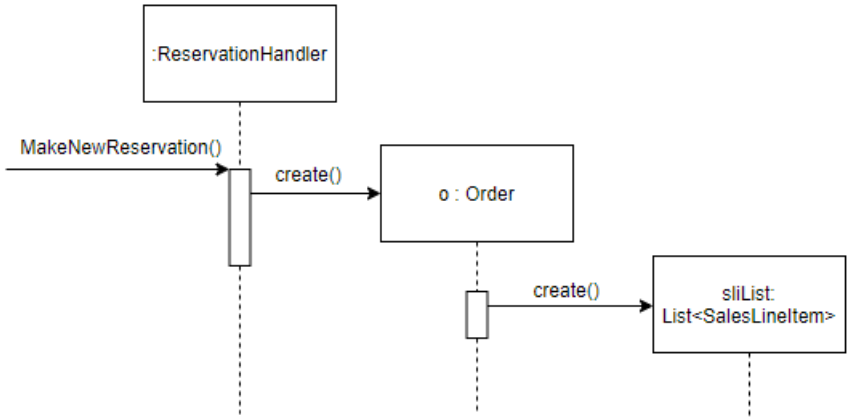


Figure 10 *[UC1] Operation 1 - sequence diagram : makeNewReservation()*

4.1.1.1.2. GRASP Pattern

Controller	**- ReservationHandler** is the first object beyond the UI layer receives and coordinates a system operation.
Creator	**- ReservationHandler** creates an **Order instance.** **- Order** creates an **SalesLineItemList.**
Information expert	NA
Low Coupling	NA
High Cohesion	NA

Table 22 *[UC1] - Operation 1 : GRASP Pattern*

4.1.1.1.3. GoF Pattern

Facade	**ReservationHandler** is Facade Controller

Table 23 *[UC1] - Operation 1 : GoF Pattern*

4.1.1.2. Operation 3 : selectMenu(item, quantity)

4.1.1.2.1. Sequence Diagram

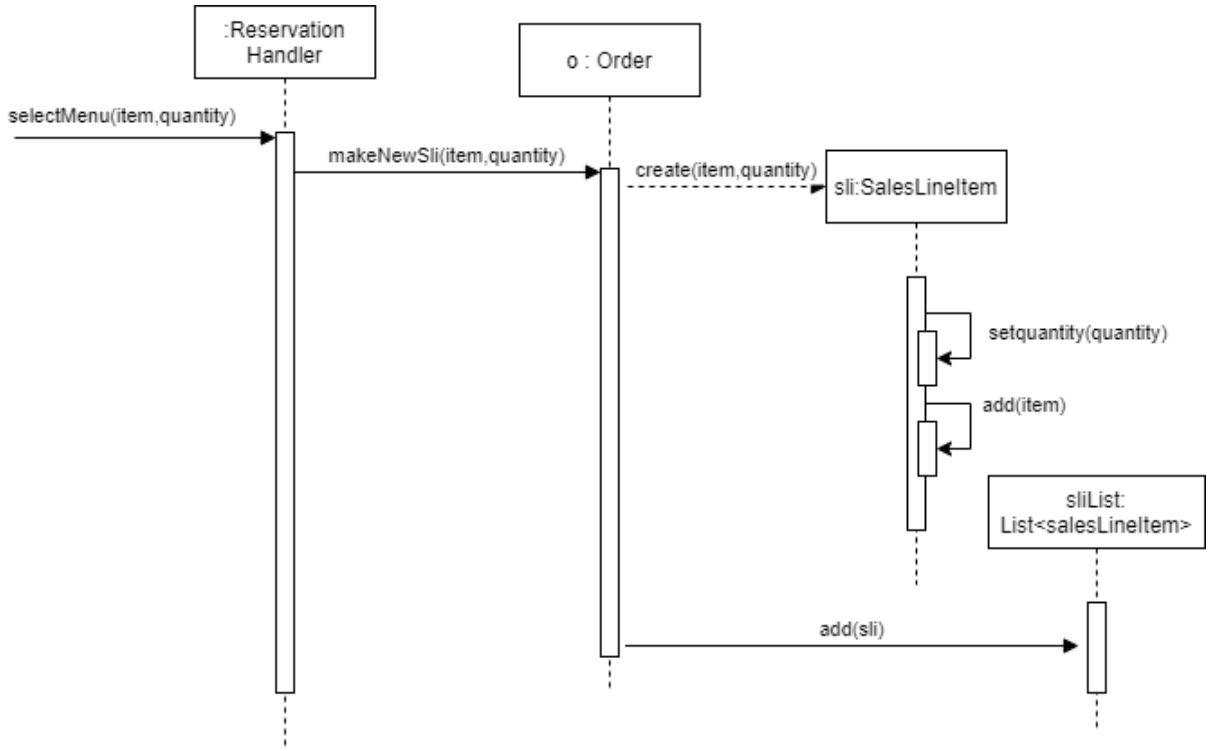


Figure 11 *[UC1] Operation 3 - Sequence Diagram : selectMenu(item, quantity)*

4.1.1.2.2. GRASP Pattern

Controller	**- ReservationHandler** is the first object beyond the UI layer receives and coordinates a system operation.
Creator	**- Order** creates **SalesLineItem.**
Information expert	NA
Low Coupling	NA
High Cohesion	NA

Table 24 *[UC1] - Operation 3 : GRASP Pattern*

4.1.1.2.3. GoF Pattern

Facade	**ReservationHandler** is Facade Controller

Table 25 *[UC1] - Operation 3 : GoF Pattern*

4.1.1.3. Operation 4 : selectSeat(Seat, time)

4.1.1.3.1. Sequence Diagram

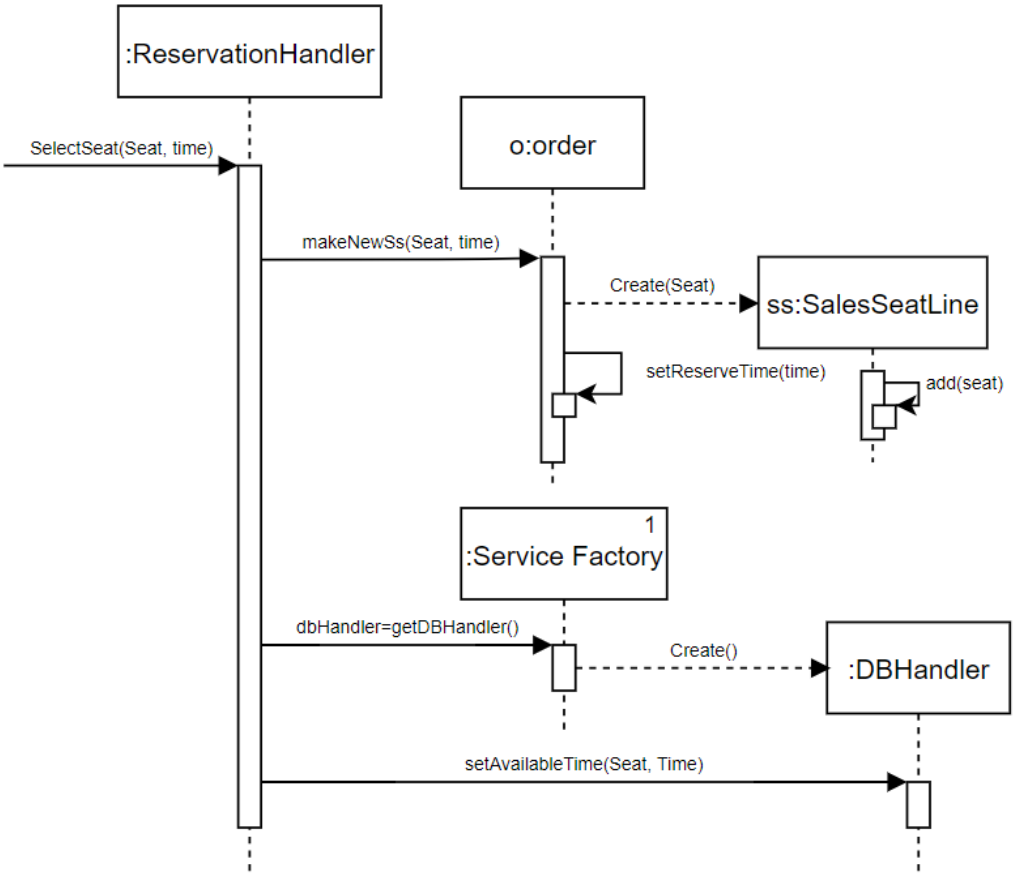


Figure 12 *[UC1] Operation 4 - Sequence diagram : selectSeat(seat, time)*

4.1.1.3.2. GRASP Pattern

Creator	**- Order** creates a **SalesSeatLine instance.** **- Service Factory** creates **DBHandler.**
Controller	**- ReservationHandler** is first object beyond the UI layer receives and coordinates a system operation.
Information expert	NA

Low Coupling	**- ReservationHandler** delegates the setAvailableTime reponsibility to the **DBHandler.**
High Cohesion	NA

Table 26 *[UC1] - Operation 4 : GRASP Pattern*

4.1.1.3.2. Gof Pattern

Facade	**- Reservation Handler** is Facade Controller
Factory	- **Service Factory** creates **DBHandler.**
Singleton	- **Service Factory** creates the factory itself.

Table 27 *[UC1] - Operation 4 : GoF Pattern*

4.1.1.4. Operation 5 : confirmOrder()

4.1.1.4.1. Sequence Diagram

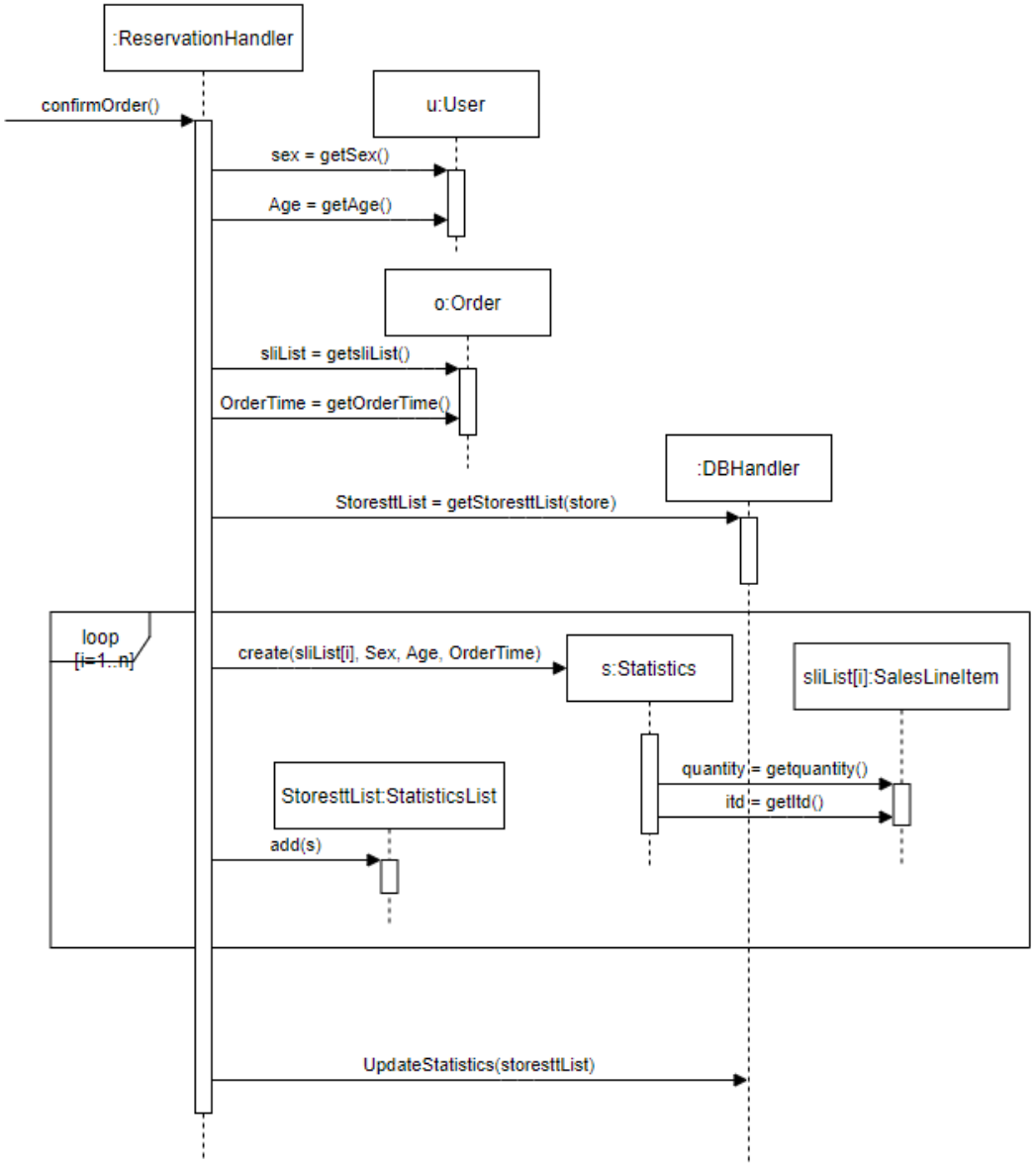


Figure 13 *[UC1] Operation 5 - Sequence Diagram : confirmOrder()*

4.1.1.4.2. GRASP Pattern

Creator	**- ReservationHandler** creates a **Statistics instance**.
Controller	**- ReservationHandler** is the first object beyond the UI layer receives and coordinates a system operation.

Information expert	**- Order** knows **SalesLineItemList, OrderTime.** **- User** knows **Sex, Age.** **- SalesLineItem** knows **quantity, itemDescription.**
Low Coupling	**- ReservationHandler** delegates the UpdateStatistics responsibility to the **DBHandler.** **- ReservationHandler** does not need to know how to get **StoresttList.**
High Cohesion	NA

***Table 28** [UC1] - Operation 5 : GRASP Pattern*

4.1.1.4.3. Gof Pattern

Facade	**- Reservation Handler** is Facade Controller

***Table 29** [UC1] - Operation 5 : GoF Pattern*

4.1.1.4. Operation 6 : makePayment()

4.1.1.4.1. Sequence Diagram

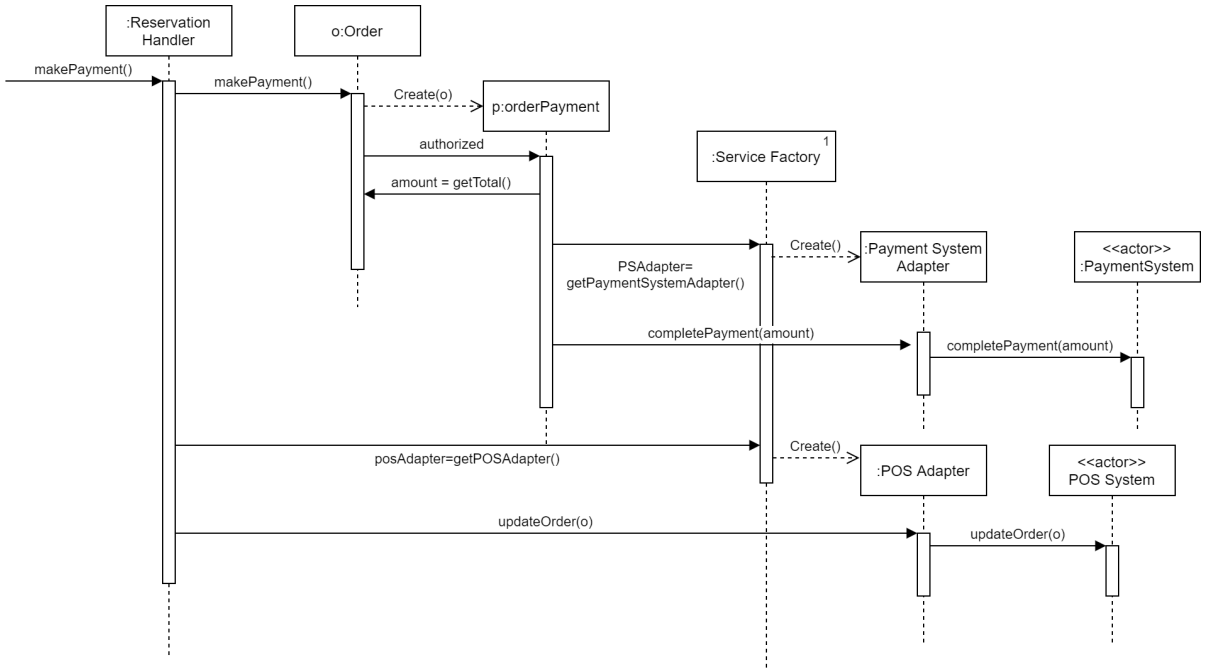


***Figure 14** [UC1] Operation 6 - Sequence Diagram : makePayment()*

4.1.1.4.2. GRASP Pattern

Creator	**- Order** creates **orderPayment.**
Controller	**- ReservationHandler** is first object beyond the UI layer receives and

	coordinates a system operation.
Information expert	**- Order** knows amount**.** **- ReservationHandler** knows **Order.**
Low Coupling	**- ReservationHandler** delegates the create orderPayment responsibility to the **Order.**
High Cohesion	NA

Table 30 *[UC1] - Operation 6 : GRASP Pattern*

4.1.1.4.2. GoF Pattern

Adapter	**Payment system Adapter** calls adaptee to perform the payment. **POS Adapter** calls adaptee to carry out the request.
Factory	**- Service Factory** create **Payment system Adapter** **- Service Factory** create **POS Adapter**
Singleton	Only one **Service Factory** needs to be created.
Facade	**ReservationHandler** is Facade Controller

Table 31 *[UC1] - Operation 6 : GoF Pattern*

4.1.2. Combined DCD

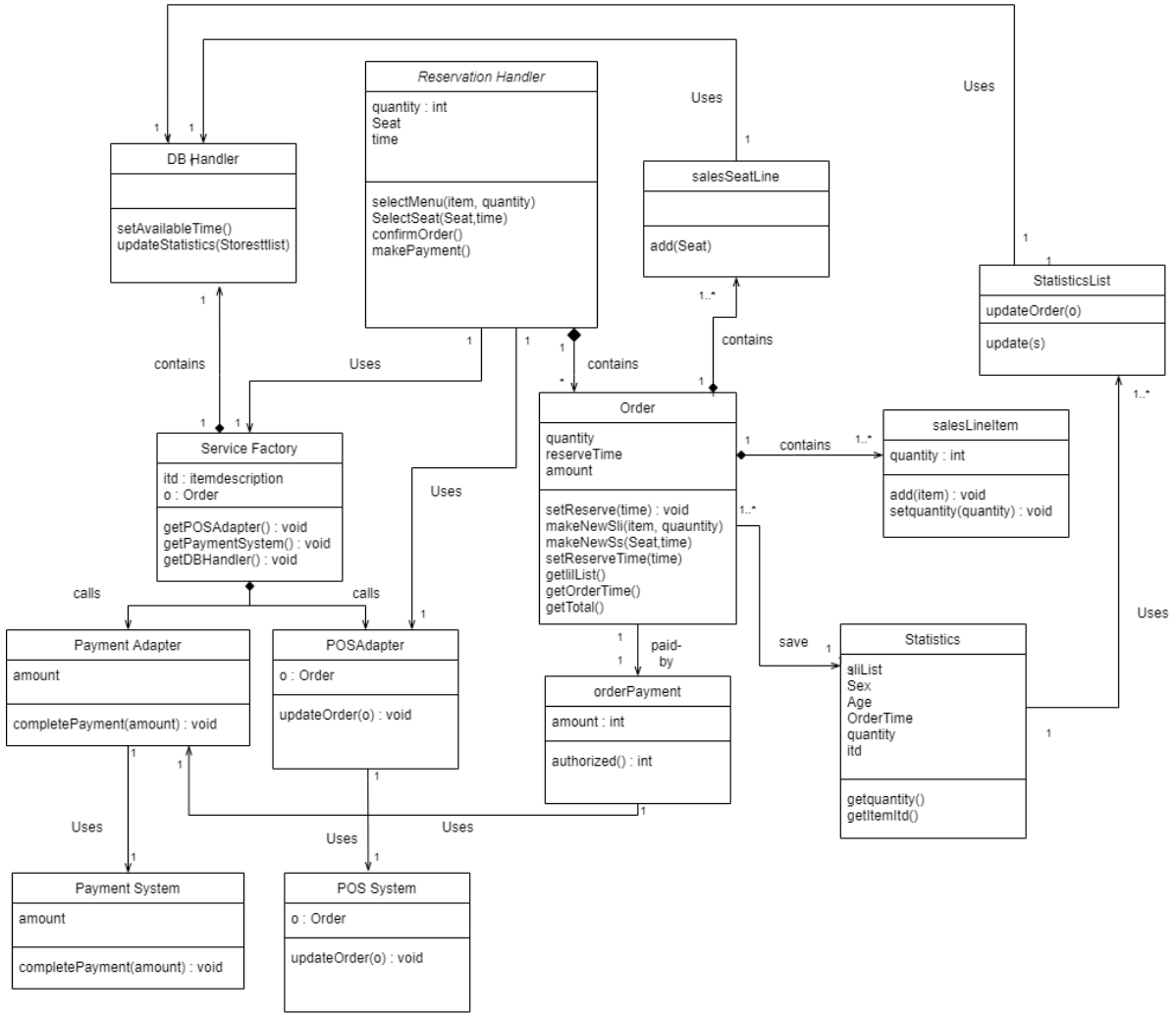


Figure 15 *Use Case 1 : Combined Design Class Diagram*

4.2. UC2 Manage menu ：Realization

4.2.1. Design Sequence Diagrams

UC2인 Manage menu 에서는 모든 Operation에 대해 Use case가 어떻게 구현되는지 인지가 필요하기 때문에 sequence Diagram을 작성했다.

4.2.1.1. Operation 1 : makeNewMenu()

4.2.1.1.1. Sequence Diagram

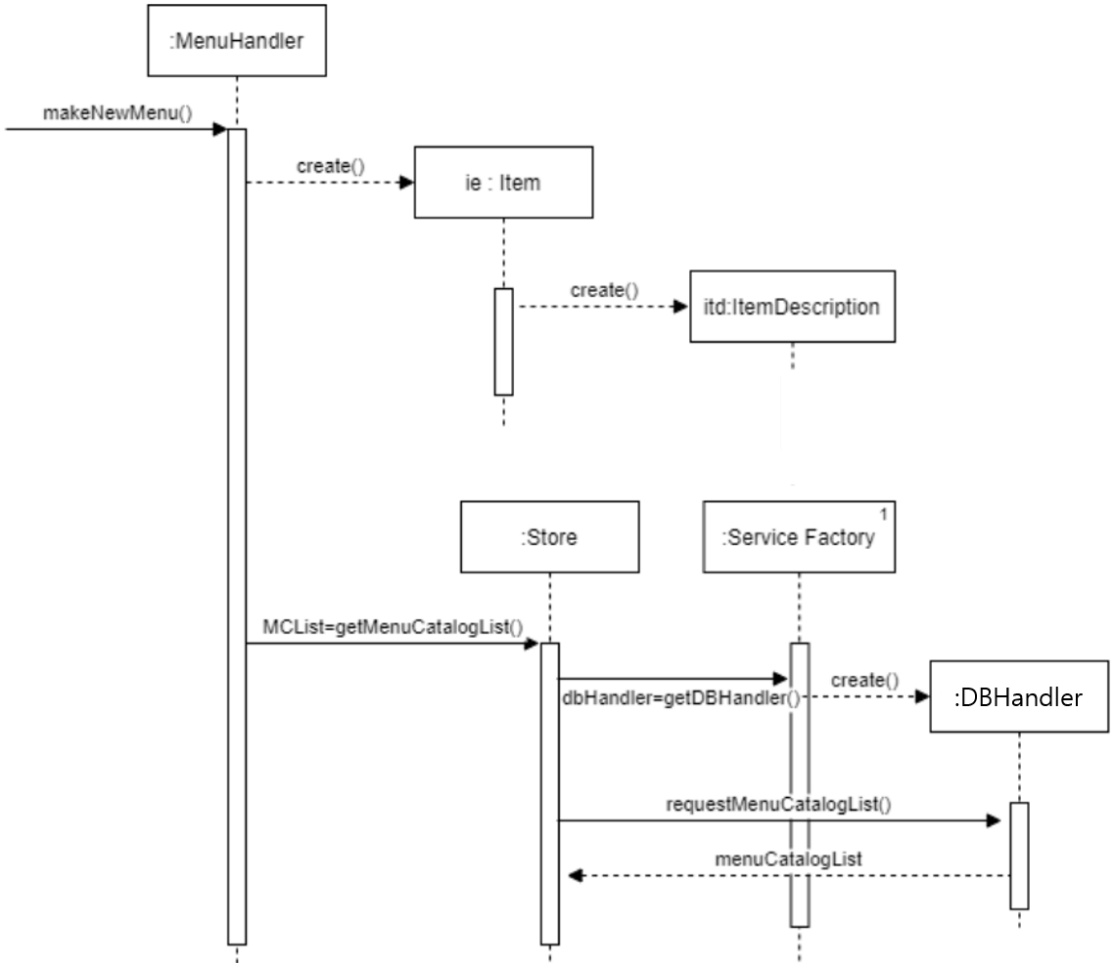


Figure 16 *[UC 2] Operation 1 - Sequence Diagram ：makeNewMenu()*

4.2.1.1.2. GRASP Pattern

Creator	**- MenuHandler** creates **Item.**

	- Item creates **ItemDescription.** **- Service Factory** creates **DBHandler.**
Controller	**- MenuHandler** is first object beyond the UI layer receives and coordinates a system operation.
Information expert	**- Store** knows **MenuCatalogList.**
Low Coupling	**- MenuHandler** delegates the requesMenuCatalogList responsibility to the **Store.**
High Cohesion	NA

***Table 32** [UC2] - Operation 1 : GRASP Pattern*

4.2.1.1.2. GoF Pattern

Facade	**MenuHandler** is Facade Controller
Factory	**Service Factory** creates **DBHandler.**
Singleton	Only one **Service Factory** needs to be created.

***Table 33** [UC2] - Operation 1 : GoF Pattern*

4.2.1.2. Operation 2 : selectCatalog(mc)

4.2.1.2.1. Sequence Diagram

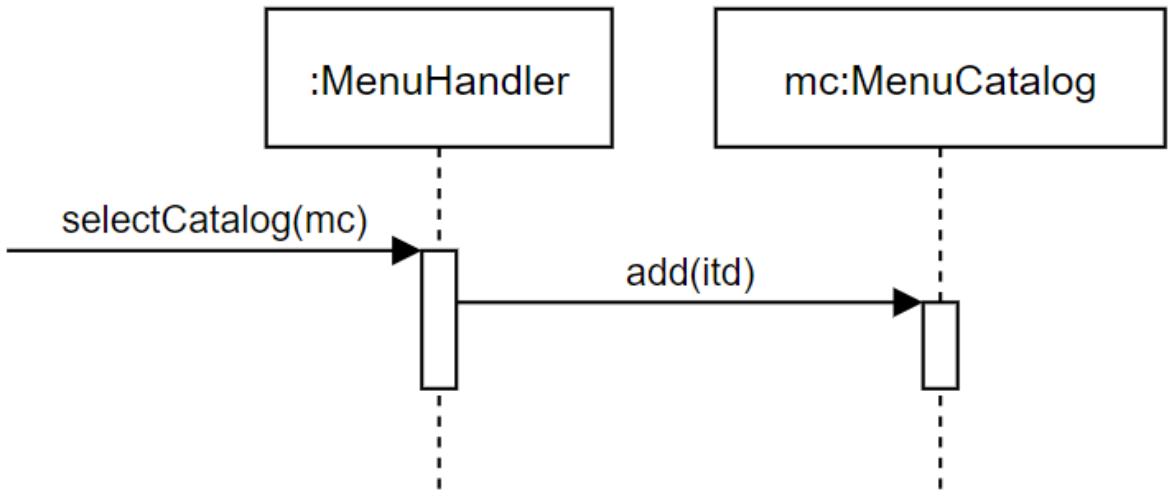


***Figure 17** [UC 2] Operation 2 -Sequence Diagram : selectCatalog(mc)*

4.2.1.2.2. GRASP Pattern

Creator	NA
Controller	**- MenuHandler** is first object beyond the UI layer receives and coordinates a system operation
Information expert	**- MenuHandler** knows **itd.**
Low Coupling	NA
High Cohesion	NA

Table 34 *[UC2] - Operation 2 : GRASP Pattern*

4.2.1.4.3 GoF Pattern

Facade	**MenuHandler** is Facade Controller

Table 35 *[UC2] - Operation 2 : GoF Pattern : selectCatalog*

4.2.1.3. Operation 3 : enterMenuinfo(itemID, name, price, description)

4.2.1.3.1. Sequence Diagram

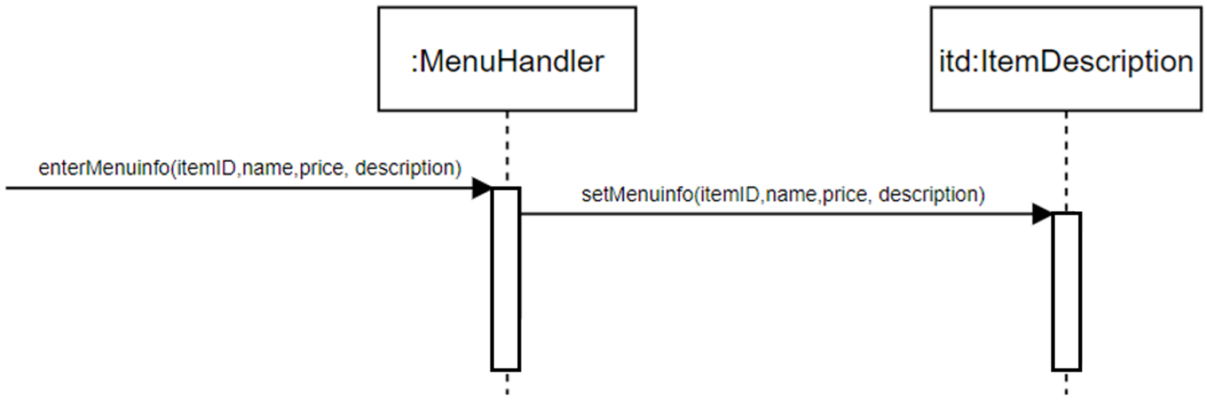


Figure 18 *[UC 2] Operation 3 - Sequence Diagram : enterMenuinfo(itemID, name, price, description)*

4.2.1.3.2. GRASP Pattern

Creator	NA
Controller	**-MenuHandler** is first object beyond the UI layer receives and coordinates a system operation
Information	NA

expert	
Low Coupling	NA
High Cohesion	NA

Table 36 *[UC2] - Operation 3 : GRASP Pattern*

4.2.1.4.3 GoF Pattern

Facade	**MenuHandler** is Facade Controller

Table 37 *[UC2] - Operation 3 : GoF Pattern*

4.2.1.4. Operation 4 : confirm()

4.2.1.4.1. Sequence Diagram

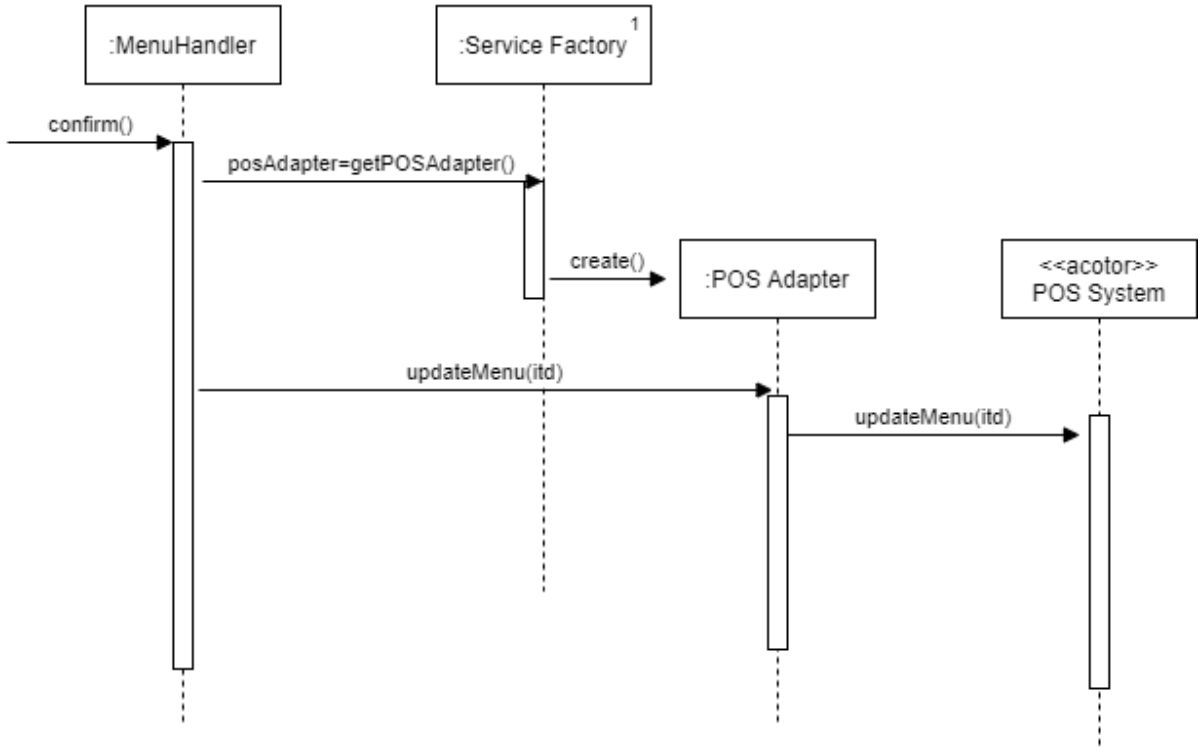


Figure 19 *[UC 2] Operation 4 -Sequence Diagram : confirm()*

4.2.1.4.2. GRASP Pattern

Creator	NA
Controller	**- MenuHandler** is first object beyond the UI layer receives and coordinates a

	system operation
Information expert	**- MenuHandler** knows **ItemDescription.**
Low Coupling	NA
High Cohesion	NA

***Table 38** [UC2] - Operation 4 : GRASP Pattern*

4.2.1.4.3 GoF Pattern

Adapter	**POS Adapter** calls adaptee to carry out the request.
Factory	**Service Factory** create **POS Adapter**
Singleton	Only one **Service Factory** needs to be created.
Facade	**MenuHandler** is Facade Controller

***Table 39** [UC2] - Operation 4 : GoF Pattern*

4.2.2. Combined DCD

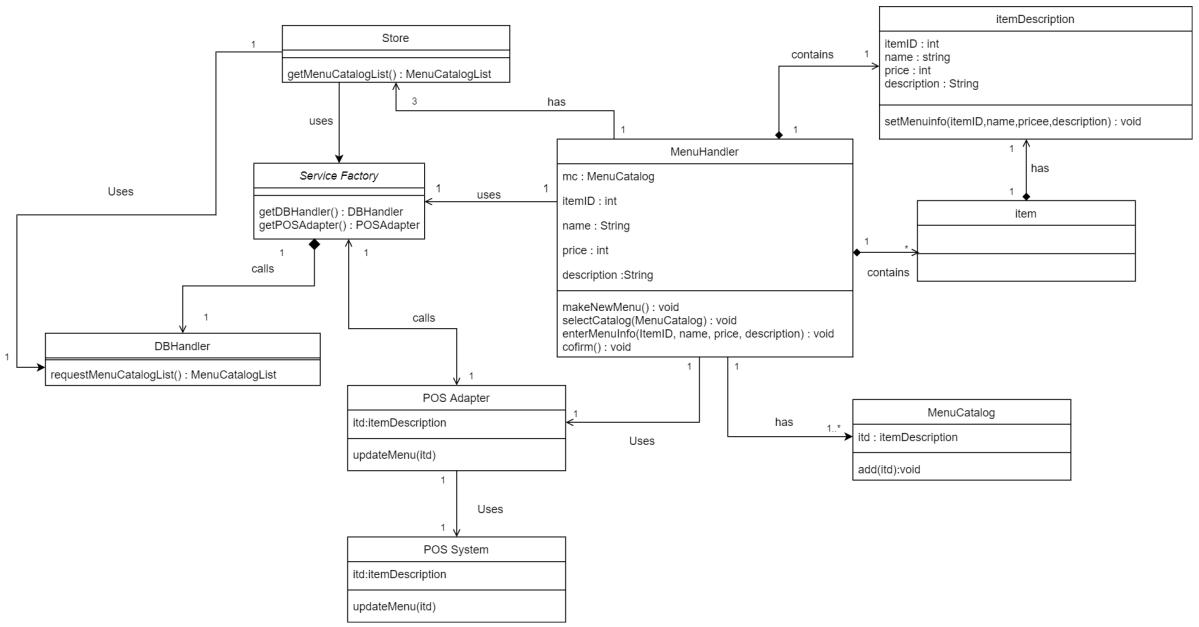


***Figure 20** Use Case 2 : Combined Design Class Diagram*

4.3. UC3 Manage Statistics : Realization

4.3.1. Design Sequence Diagrams

4.3.1.1. Operation 1 : makeNewStatistics()

4.3.1.1.1. Sequence Diagram

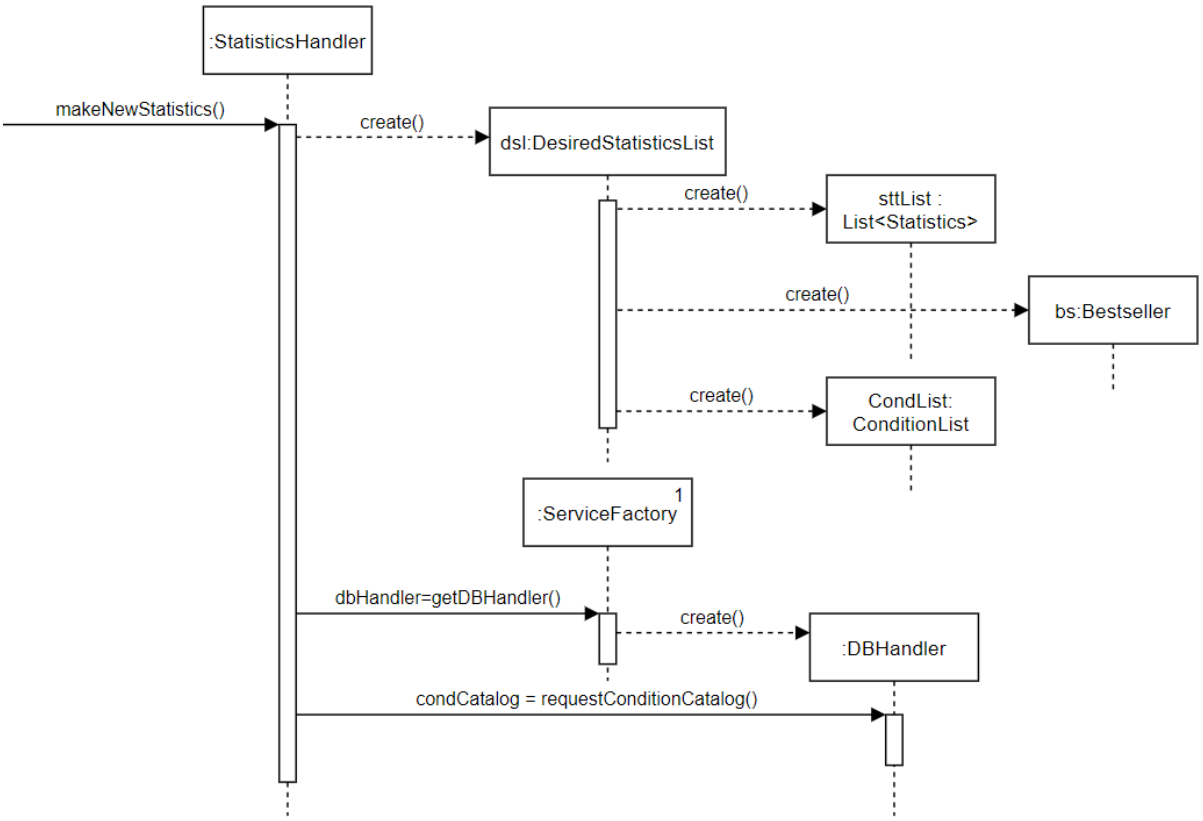


Figure 21 *[UC 3] Operation 1 -Sequence Diagram : makeNewStatistics()*

4.3.1.1.2. GRASP Pattern

Creator	**- StatisticsHandler** creates **DesiredStatisticsList.** **- DesiredStatisticsList** creates **BestSeller.** **- DesiredStatisticsList** creates **StatisticsList.**
Controller	**- StatisticsHandler** is first object beyond the UI layer receives and coordinates a system operation.
Information expert	NA

Low Coupling	NA
High Cohesion	NA

***Table 40** [UC3] - Operation 1 : GRASP Pattern*

4.3.1.1.3. GoF Pattern

Facade	**- StatisticsHandler** is a **Facade Controller.**
Factory	- **Service Factory** creates **DBHandler.**
Singleton	- **Service Factory** creates the factory itself.

***Table 41** [UC3] - Operation 1 : GoF Pattern*

4.3.1.2. Operation 2 : selectCondition(CondDef, Condinput)

4.3.1.2.1. Sequence Diagram

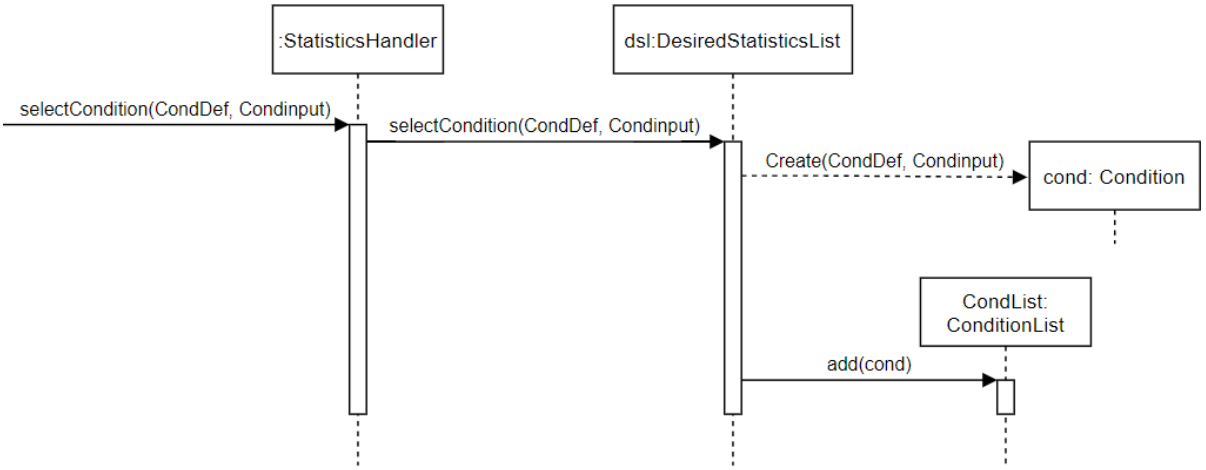


***Figure 22** [UC 3] Operation 2 - Sequence Diagram : selectCondition(CondDef, Condinput)*

4.3.1.2.2. GRASP Pattern

Creator	**- DesiredStatisticsList** creates a **Condition instance**.
Controller	**- StatisticsHandler** is first object beyond the UI layer receives and coordinates a system operation.
Information expert	- **DesiredStatisticsList** knows **Condition.**
Low Coupling	**- System** delegates the create Condition responsibility to the **DesiredStatisticsList.**

High Cohesion	NA

Table 42 *[UC3] - Operation 2 : GRASP Pattern*

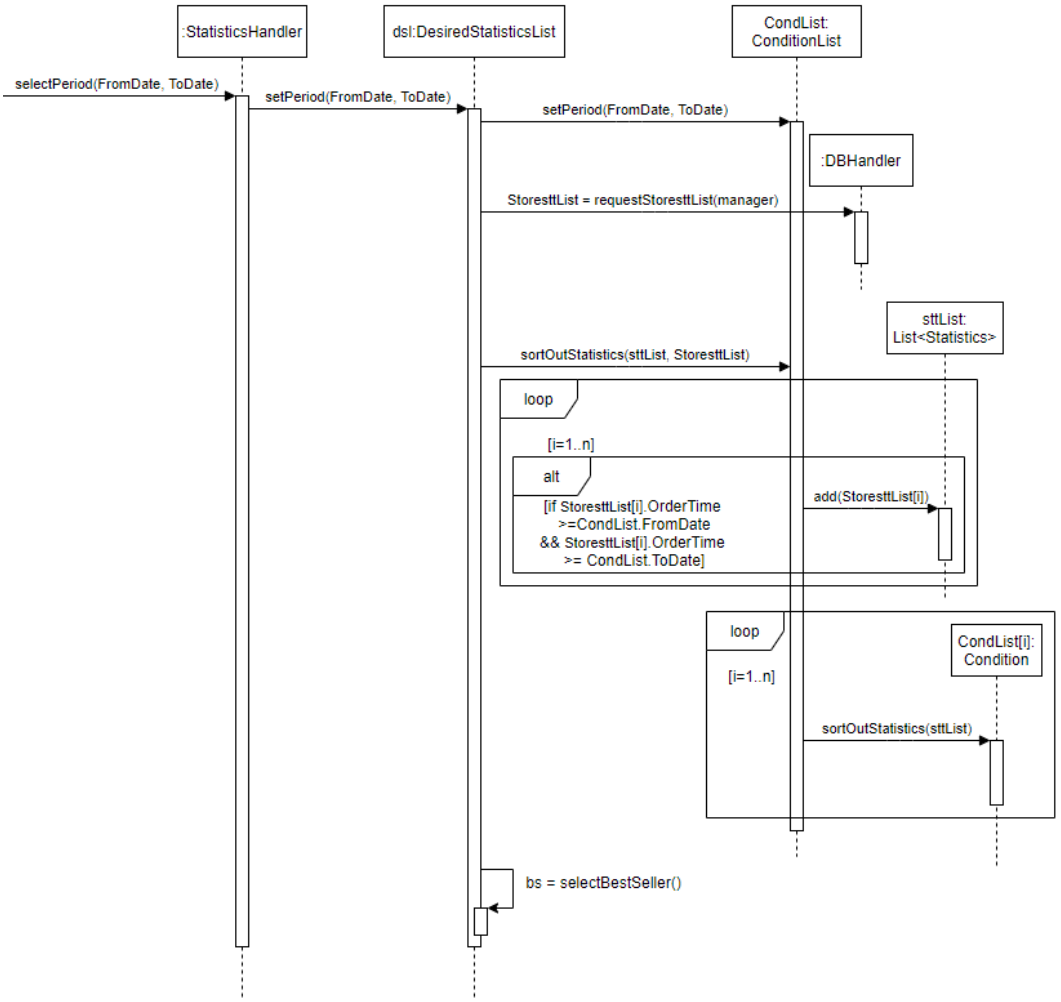
4.3.1.2.3. GoF Pattern

Facade	**- StatisticsHandler** is a **Facade Controller.**

Table 43 *[UC3] - Operation 2 : GoF Pattern*

4.3.1.3. Operation 3 : selectPeriod(FromDate, ToDate)

4.3.1.3.1. Sequence Diagram



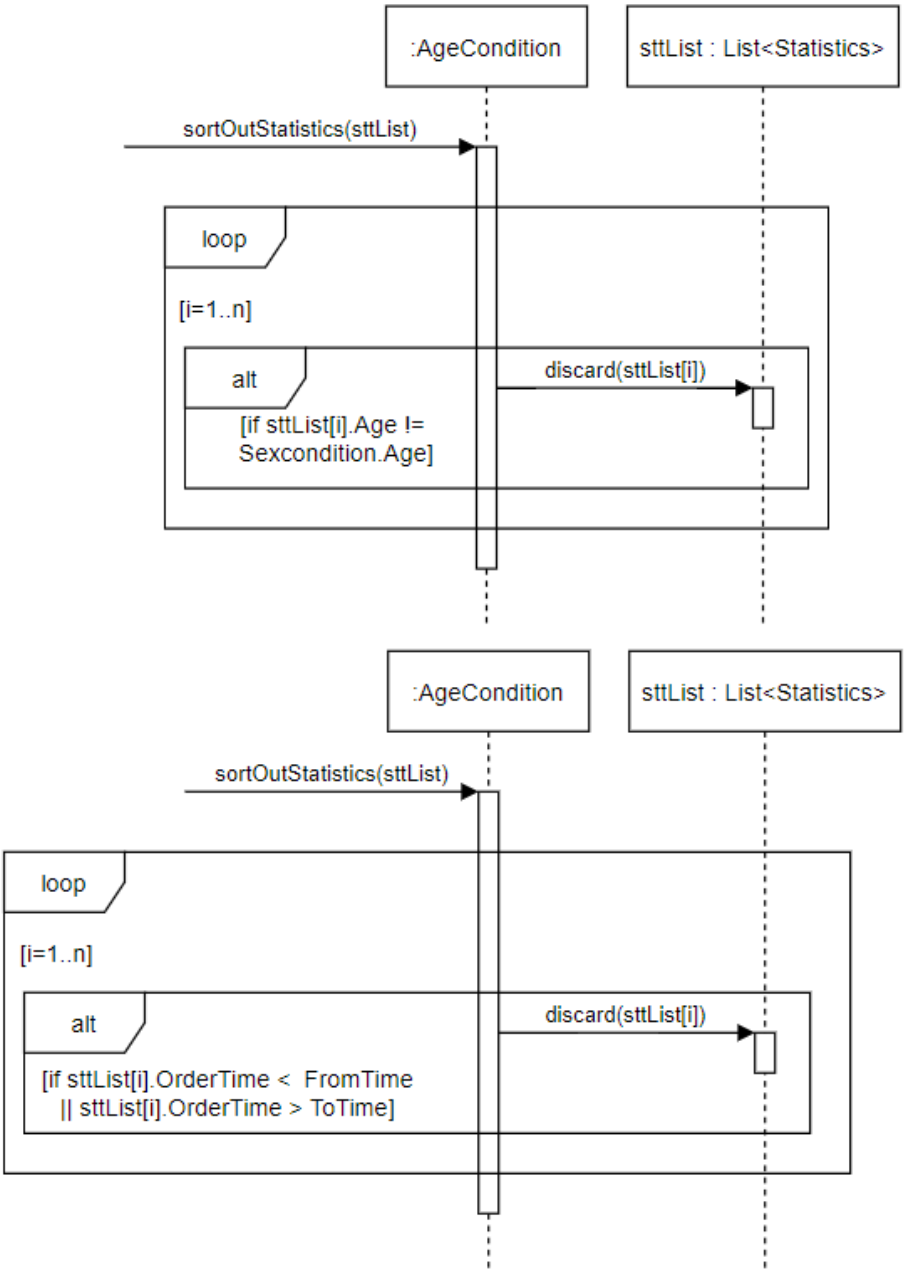


Figure 23 *[UC 3] Operation 3 - Sequence Diagram : selectPeriod(FromDate, ToDate)*

4.3.1.3.2. GRASP Pattern

Creator	NA
Controller	**- StatisticsHandler** is first object beyond the UI layer receives and coordinates a system operation.
Information expert	- **ConditionList** knows FromDate and ToDate. - **DesiredStatisticsList** knows **sttList.**
Low Coupling	- **DesiredStatisticsList** does not need to know how to get **StatisticsList.**
High Cohesion	- **DesiredStatisticsList** select its own **BestSeller.**
Polymorphism	**- Condition** is a base class for different modes of sortOutStatistics.

Table 44 *[UC3] - Operation 3 : GRASP Pattern*

4.3.1.3.3. GoF Pattern

Facade	**- StatisticsHandler** is a **Facade Controller.**

Table 45 *[UC3] - Operation 3 : GoF Pattern*

4.2.2. Combined DCD

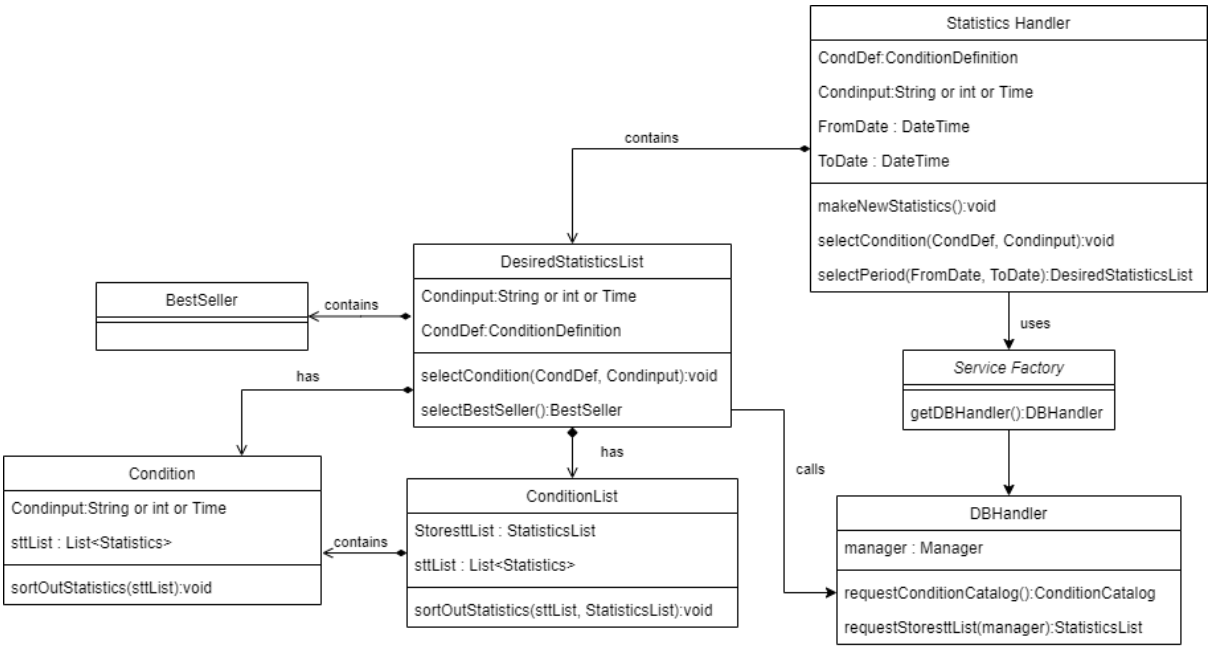


Figure 24 *Use Case 3 : Combined Design Class Diagram*

5. Software Architecture

5.1 Introduction

we will talk about SAD. SAD stands for Software Architect Document.This will be looked at from various perspectives, such as Logical, data, deployment, process and use case view, and the priority of risk will be known. Through this we can make this project useful.

- **Logical**

  Conceptual organization of the software in terms of the most important layers, subsystems, packages, frameworks, class, classes, and interfaces. Also summarizes the functionality of the major software elements, such as each subsystems

- **Process**

  Processes and threads. Their responsibilities, collaboration, and the allocation of logical elements to them.

- **Deployment**

  Physical deployment of processes and components to processing nodes, and the physical network configuration between nodes.

- **Data**

  Overview of the data flows, persistent data schema, the schema mapping from objects to persistent data (usually in a relational database), the mechanism of mapping from objects to a database, database stored procedures and triggers

- **Use case**

Summary of the most architecturally significant use cases and their non-functional requirements. That is, those use cases that, by their implementation, illustrate significant architectural coverage or that exercise many architectural elements.

5.2 Architectural Factors

Factor	Measures and Quality scenarios	Variability(Current flexibility and future evolution)	Impact of factor(and its variability) on stakeholders, architecture and other factors	Priority for success	Difficulty or Risk
Reliability-Recoverability					
recovery from server Failure	The server can be recovered although the server is shut down.	· **Current Flexibility :** All data must be stored in db. · **Evolution :** none	During the menu update phase, the server must be accessed and the user must be satisfied with the recovery from the server failure.	H	H
recovery connection with DB	If the connection to the DB is lost, request the connection again within 40 seconds.	· **Current Flexibility :** Store the latest data before disconnecting. · **Evolution :** To do recovery, let you have your own DB server.	High impact on the large-scale design. Errors in the DB result in loss of credibility in statistics. Therefore, the value of the application decreases.	H	H
Functionality- security					
Access to Manager	Users who do not have previous manager certification do not have access to the categories	· **Current Flexibility :** Existing users can gain manager access only by registering the store's business license number.	This application is easy for businesses with stores to use this application.	H	L

	that the manager can access.				

Table 46 *Architectural Factors*

5.3 Logical view

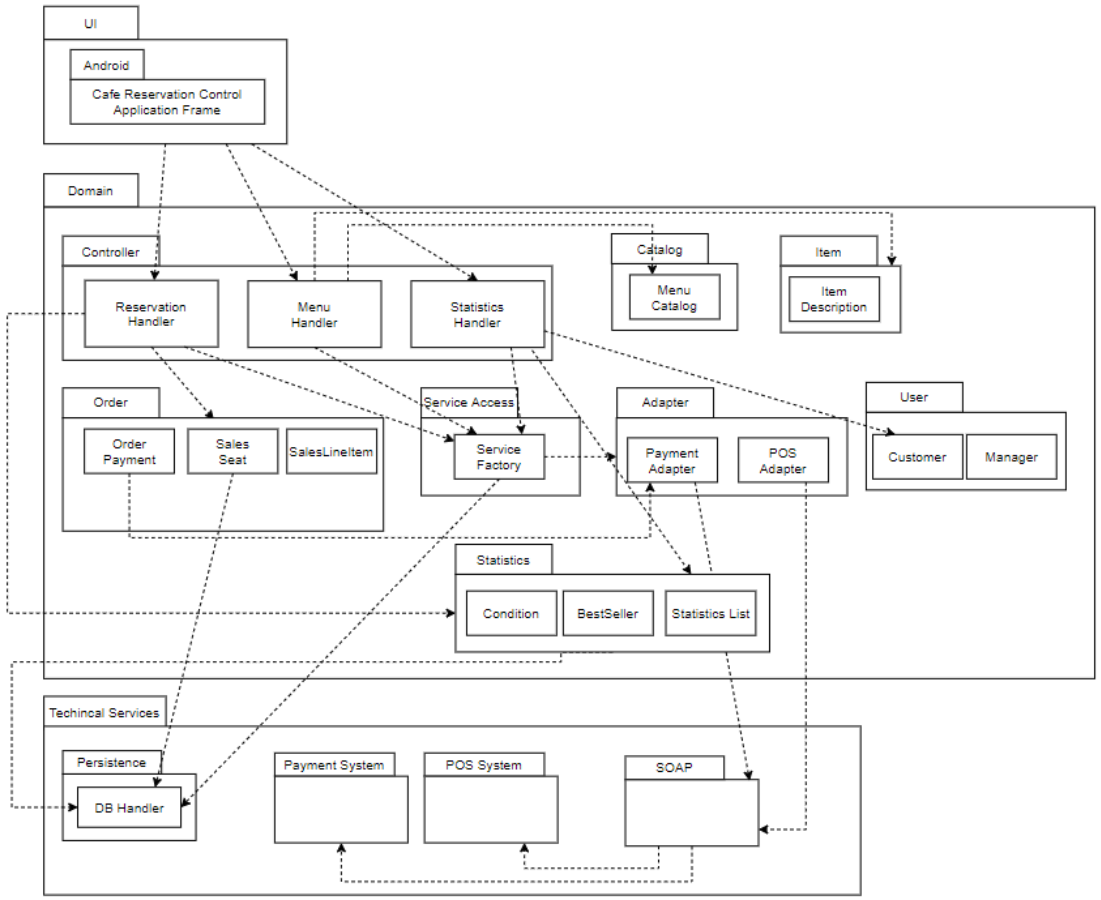


Figure 25 *logical view*

5.4 Process View

Not designed

5.5 Deployment view

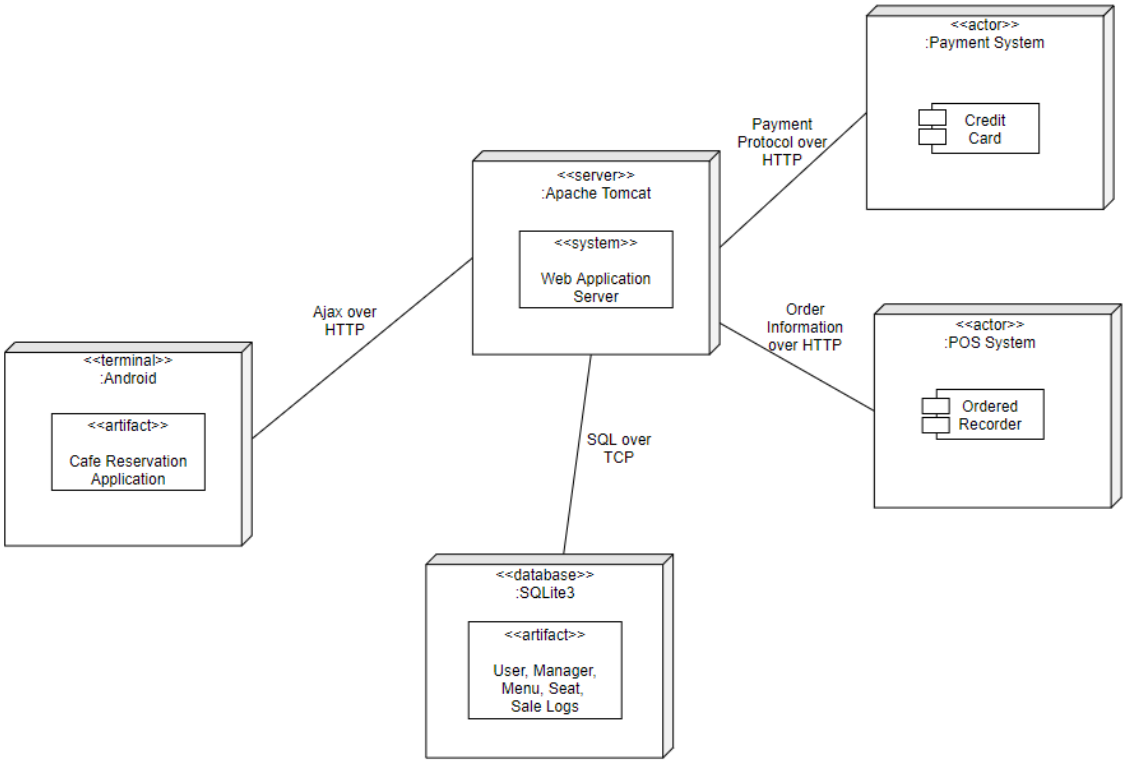


Figure 26 *Deployment View*

5.6 Use Case view

5.6.1 UC1 Use Case view

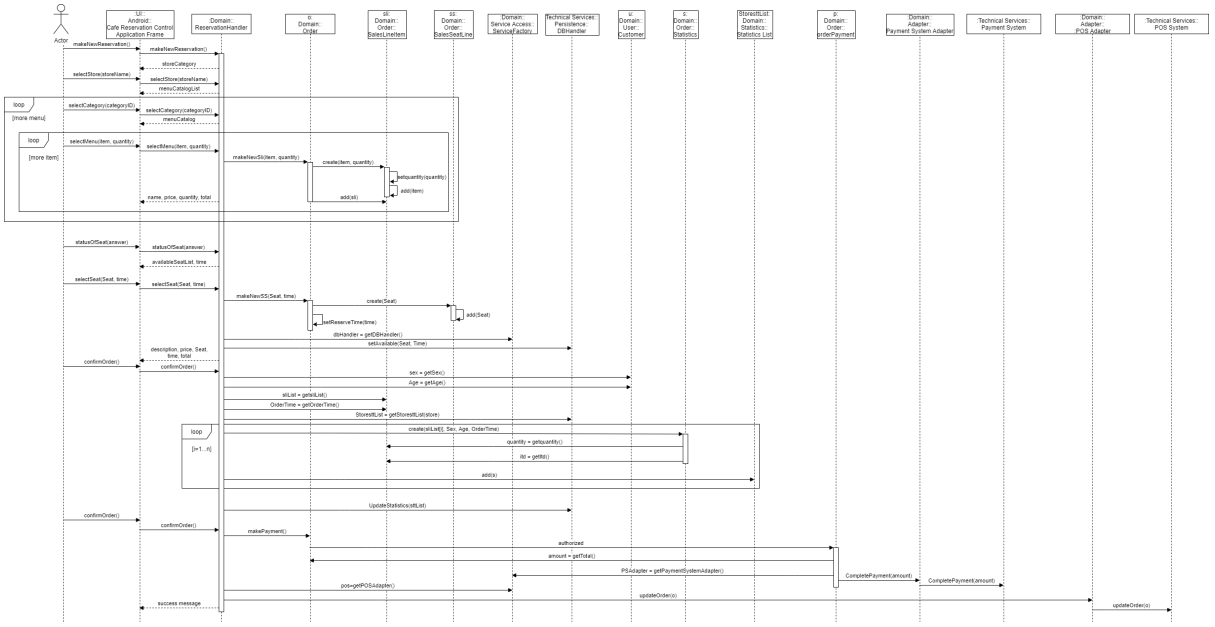


Figure 27 *UC1 Use case View*

5.7 Data view

5.7.1 UC1

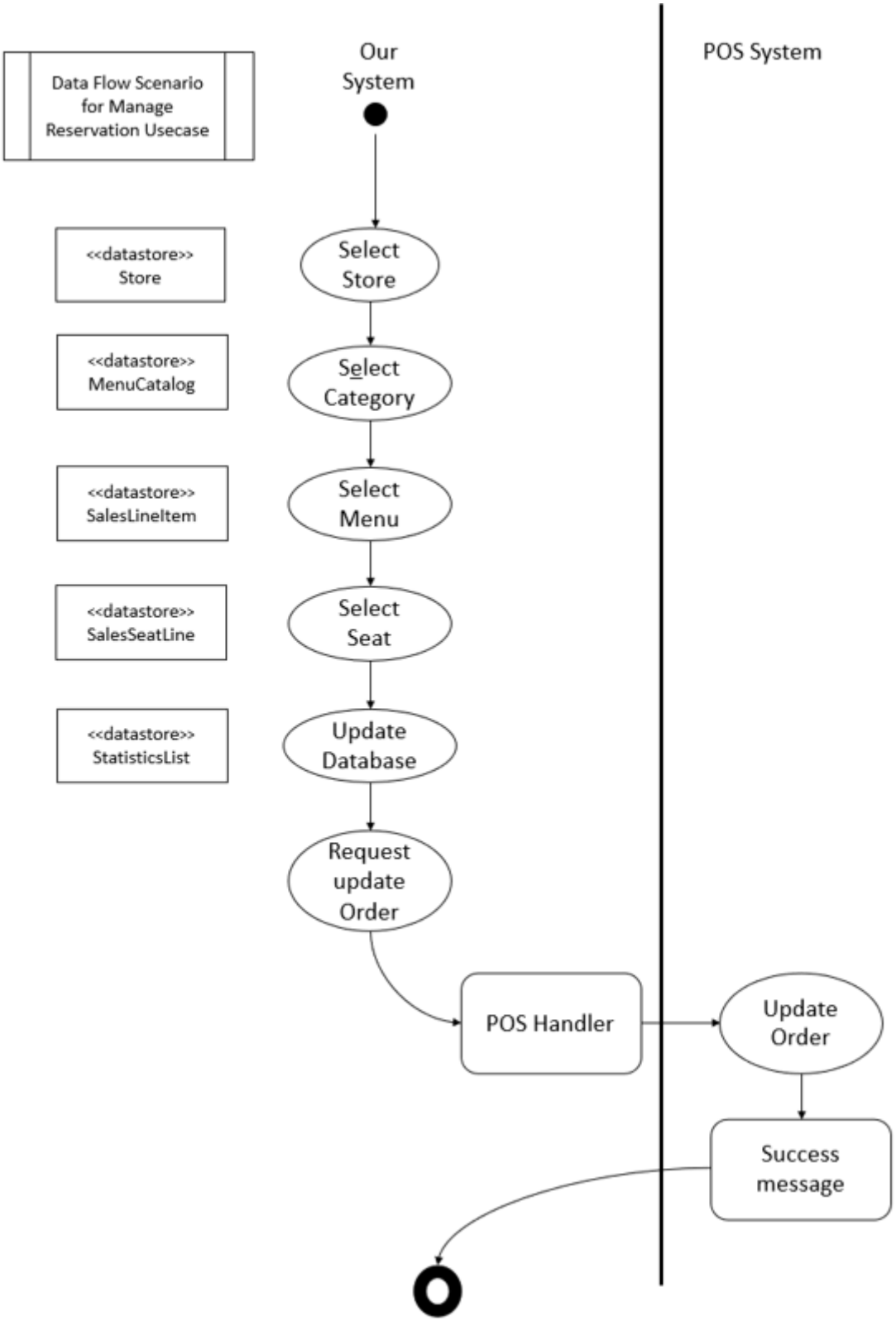


Figure 28 *UC1 Data View*

5.7.2 UC2

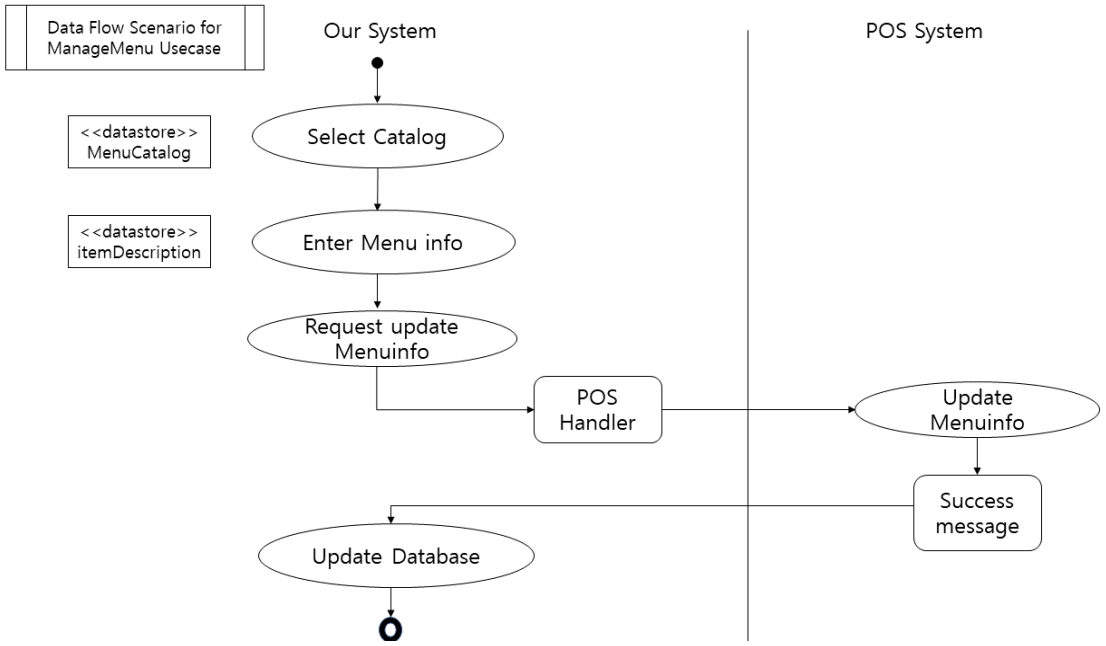


Figure 29 *UC2 Data View*

5.8 Technical Memo

Issue :
Performance - Managers should be able to use this application regardless of time.
(The existing customer can be accessed outside of the cafe's business hours, but not all services are available.)

Solution Summary : Select a trusted server hosting

Factor : Managers must have 24-hour access to add menus or use statistical functions outside of business hours.

Solution : If the server goes down, select a trusted server hosting company that can recover quickly.

Motivation : Modifying menus during business hours can confuse customers. Therefore, managers should be able to modify what's about the cafe at a convenient time outside of business hours. If it is not available 24 hours a day, managers will not use the system because it is not competitive.

Unresolved Issue : the question of whether our program can pay the price

Alternative Considered : Price comparisons are made under the same service delivery conditions.

Table 47 *Technical Memo*

6. Conclusion

We completed 80% design modeling for 3 use cases. About code implementation, we were struggling to implement codes with java but with python gui, it's more convenient so used python gui. We completed the success scenario of use case1: manage reservation up to 80%.

In the use case3: Manage sales statistics, we have to compile statistics of sales results so we implemented a database. Manager can select conditions like time zone, age and sex by date when you want to analyze data. We were about to select one condition but selecting more than 2 conditions is more effective therefore we changed it.

Except those, cancellation and modification parts in use case1 are classified as extension scenarios so we didn't implement them. But in terms of the level of completion, it would be worthwhile to refine the use case1 including giving penalty in order to prevent no-shows and 3times maximum extension of seat rule that we stated at the part of functional requirements.

Use case2: Manage menu is nearly the same as use case1 like selecting category and adding the name, price and description of new item that manager wants to add. when dealing with a success scenario, add menu. It is a little bit hard for us to implement code about displaying a menu item list after the manager adds new menu items. So we decided to implement use case 1 and 3. But of course this part, it would be helpful to understand this application and meaningful because this part is quite a core function of this application thus it should be done.

We've got so many lessons from this course like teamwork, domain knowledge and methods of designing diagrams etc. Even though It's not that easy to design a complex program we learned that in depth. We can use some concepts and patterns that we've learned through this course when we implement other programs and also anticipate that we could improve the implementing process by using UML diagrams, notations and OOAD principles. In fact the real application is more complex but there was a limitation of time so it's a shame that we couldn't handle it all.

7. References

Figure 1 - **https://contents.opensurvey.co.kr/form_coffee_2020**

8. Appendix

8.1 Terms and definitions

Term	Definition and Information	Note
시스템 운영자	시스템의 유지보수와 에러수정, 편의성 개선등을 하는 사람	
Manager	카페의 운영을 담당하며 시스템의 메뉴관리를 수행할 수 있는 사람	
Customer	예약 시스템을 이용해 음료를 예약하고자 하는 카페 손님	
Manage reservation	시스템 상에서 원하는 음료, 시간, 자리를 선택해서 예약 시스템에 저장하고 예약한 내용을 수정하거나 취소할 수 있는 기능	
Sales Statistics	매니저가 매출의 통계 등을 볼 수 있는 기능	
Manage menu	카페 매니저가 시스템에 메뉴를 추가하거나 제거할 수 있는 기능	
예약 시스템	카페 이용객이 입력한 예약 내용이 저장되는 시스템	
인앱결제 시스템	예약한 음료에 대한 값을 미리 지불하고자 하는 카페 이용객을 위한 인앱 결제를 지원하는 시스템	
메뉴관리 시스템	카페 매니저가 추가한 메뉴가 저장되는 공간으로 카페 이용객이 예약할 때 저장된 메뉴를 보여줌	

***Table 48** terms and definition*

9. Revision History

Version	Date	Description	Author
Inception Phase 1	2020.09.24	First version of the inception phase report	학교가고싶다
Domain modeling (UC1 Manage reservation)	2020.9.28	Drawing Domain modeling diagram for the complex use case	학교가고싶다
SSD and OC (UC1 Manage reservation)	2020.10.27	Drawing System sequence diagram and writing operate contract for the most complex use case	학교가고싶다
Elaboration Phase 1.1 (UC1 Manage reservation & UC2 Manage menu)	2020.10.28	Modify Domain model diagram, System sequence diagram and Refresh Use case text	학교가고싶다
Elaboration Phase 1.2	2020.10.30	Add interaction diagrams per certain operation contracts and draw GRASP Pattern	학교가고싶다
Elaboration Phase 1.3	2020.11.02	Draw combined DCD	학교가고싶다

Elaboration Phase 2.1	2020.12.01	Modify Domain model diagram, System sequence diagram and Refresh Use case text	학교가고싶다
Elaboration Phase 2.2	2020.12.03	The scenario was clearly written and the domain model diagram was added. (for UC3)	학교가고싶다
Elaboration Phase 2.3	2020.12.05	The sequence diagram was modified, considering the GoF design.	학교가고싶다
Elaboration Phase 2.3	2020.12.06	70 % to complete the report of the implementation, and GUI	학교가고싶다

Table 49 *Revision history*