<무어>

<MrGreat>

설계 산출물

|  |  |
| --- | --- |
| 문서버전 |  |
| 문서 ID |  |
| 최종변경일 |  |
| 문서상태 |  |

요 약

미스터 대박 디너 서비스(Mr.Great)의 설계 산출물을 기술.

서브 시스템의 구성과 각 서브 시스템의 구조를 기술.

주요 산출물

* 아키텍처도
* 클래스도
* 교류도
* 상태도

표 1 문서 변경 기록

|  |  |  |  |
| --- | --- | --- | --- |
| 문서이름 | <무어> <MrGreat> 설계 산출물 | | |
| 문서 ID |  | | |
| 버전 | | 변경일 | 설명 |
| 1 | 0 | 2022-10-07 | 아키텍처도를 추가했다.  “Employee Client” 클래스도를 추가했다.  “Customer Client” 클래스도를 추가했다. |
| 1 | 2022-10-09 | “Server Client” 클래스도를 추가했다.  “Customer Client” 서브시스템 세부설계 설명을 추가했다.  “Staff Client” 서브시스템 세부설계 설명을 추가했다. |
| 2 | 2022-10-13 | “Server” 서브시스템 세부설계 설명을 추가했다. |
| 3 | 2022-10-15 | 유즈케이스의 교류도를 추가했다. |
| 4 | 2022-10-16 | Order 객체 상태도를 추가했다. |
| 2 | 0 |  | 릴리즈 |
| 1 |  | 오타 수정 |
| 3 | 0 |  | [피드백 반영] 서브시스템 설계 및 클래스도 수정 |
| 1 |  | [피드백 반영] 교류도, 상태도 수정 |

1 개 요

1.1 목 적

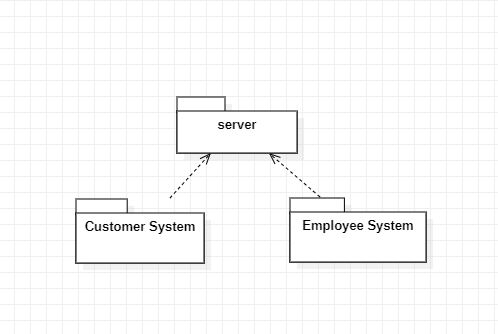
본 문서는 미스터 대박 디너 서비스(Mr.Daebak)의 산출물을 기술한다.

* 전체 시스템이 어떤 서브 시스템으로 구성되어 있는지 “아키텍처도”를 통해 보여준다.
* 서브 시스템이 어떤 클래스로 구성되어 있고 클래스들간에 어떤 관계가 있는지를 “클래스도”를 통해서 보여준다.
* 객체간에 발생하는 동적인 행위를 “교류도”를 통해서 보여준다.
* 한 객체의 상태 변화를 “상태도”를 통해서 보여준다.

1.2 참고 문헌

없음.

2 아키텍처도



전체 시스템은 아래의 서브 시스템으로 구성된다.

- Customer 시스템

- Employee 시스템

- Sever 시스템

**고객 시스템 (Customer System)**

- 서버와 통신하며 고객에게 정보를 보여주는 역할을 한다.

- Javascript를 이용하여 작성된다.

**직원 시스템(Employee System)**

- 서버와 통신하며 직원에게 정보를 보여주는 역할을 한다.

- Javascript를 이용하여 작성된다.

**Server 시스템**

- 메뉴, 고객 정보, 주문 정보를 담고 있으며 주문 저장 및 이력 관리의 역할을 한다.

- Node.js로 작성된다.

3 서브시스템 세부 설계

**3.1. “Customer System" 서브시스템 세부 설계**

Customer 시스템은 서버와 통신하며 고객에게 메뉴 화면과 주문 화면을 보여준다. 또한 서버와 통신하며 고객의 주문을 수행한다.

Customer 시스템의 주요 함수는 다음과 같다.

\* login(id, pw)

■ “Server"에 로그인 정보를 보내 로그인을 수행한다.

\* add\_new\_mem(id, pw, info)

■ 회원 정보를 입력받아 “Server”에 전송한다.

\* display\_menu()

■ 매뉴의 목록을 “Server”에 요청하고 보여준다.

\* add\_order(user\_id, menu, style, size, amount)

■ 고객이 입력한 주문을 받아서 “Server”에 전송하여 장바구니에 담는다.

\* display\_cart()

■ 현재 로그인한 회원의 장바구니를 “Server”에 요청하고 보여준다.

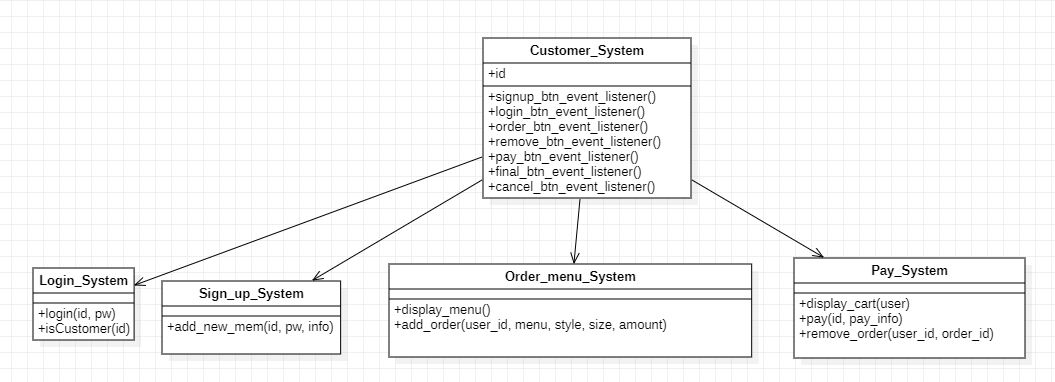
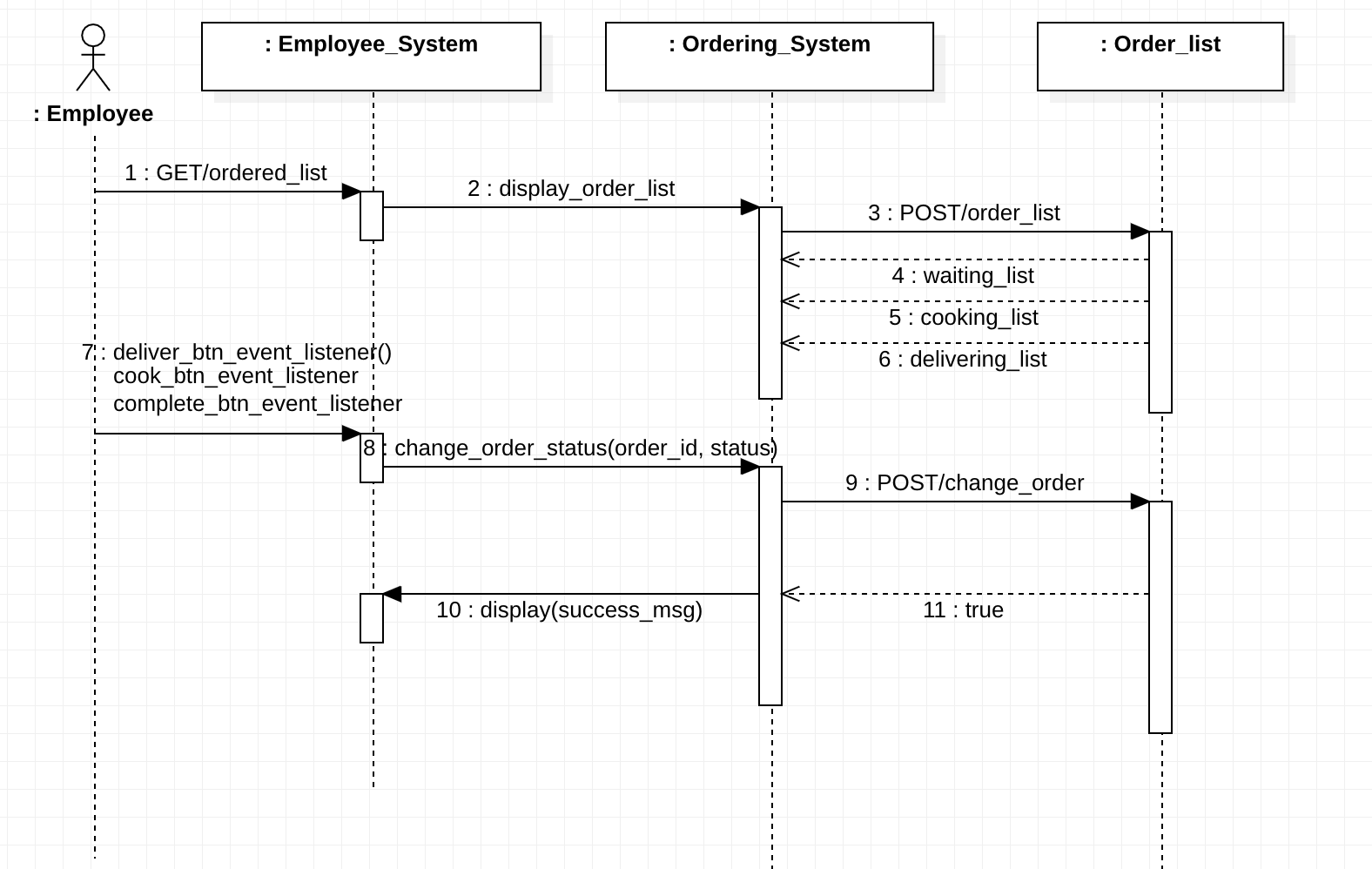
\* pay(id, pay\_info)

■ 결제 방법을 선택하고 주문 정보를 “Server”에 보낸다.

\* remove\_order(uesr\_id, order\_id)

■ 장바구니에서 제외할 주문을 “Server”에 보내서 장바구니에서 제외한다.

- “Customer Client” 서브시스템 메인 클래스도



**Customer\_System**

* 고객이 사용하는 주문 시스템의 UI를 보여주는 클래스이다.
* 고객이 회원가입과 로그인을 진행할 수 있다.
* 매뉴와 장바구니를 보여주고 장바구니에 담기와 결제를 진행할 수 있다.
* 장바구니에 있는 주문 중에서 삭제할 주문을 장바구니에서 제외할 수 있다.

**Login\_System**

* 로그인을 수행하는 클래스이다.

**Sign\_up\_System**

* 회원가입을 수행하는 클래스이다.

**Order\_menu\_System**

* 매뉴 목록을 화면에 보여주는 클래스이다.
* 주문하기를 수행해서 고객이 주문하려는 주문을 장바구니에 담을 수 있다.

**Pay\_System**

* 서버에 장바구니 목록을 요청하고 화면에 보여주는 클래스이다.
* 삭제할 주문 목록을 선택하여 장바구니에서 제외할 수 있다.
* 결제 방법을 선택하여 최종적인 결제를 수행할 수 있다.

3.2 “Employee System” 서브시스템 세부 설계

Staff 시스템은 서버와 통신하며 직원에게 현재 처리해야하는 주문 목록을 보여준다. 또한 현재까지 완료된 주문의 history를 보여준다.

Employee Client 시스템의 주요 함수는 다음과 같다.

\* login(id, pw)

■ “Server"에 id, pw를 보내 로그인을 수행한다.

\* display\_order\_lists()

■ “Server"에서 주문 목록을 받아 보여준다.

\* change\_order\_status(order, status)

■ 주문 목록 중에서 상태를 바꿀 주문을 설정한다.

\* cook\_btn\_event\_listener(), delivery\_btn\_event\_listener(), complete\_btn\_event\_listener()

■ 주문의 상태에 맞추어 각 버튼에 맞게 change\_order\_status를 호출한다.

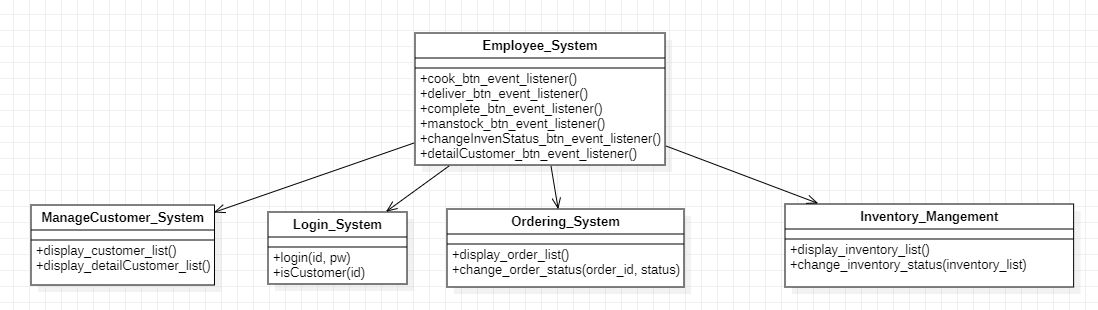
\* display\_inventory\_list

■ 현재 재고 목록을 화면에 보여준다.

\* change\_inventory\_status(inventory\_list)

■ 현재 재고 상태를 재고 목록에 입력한다.

- “Employee Client” 서브시스템 메인 클래스도



Employee\_System

* 직원이 사용하는 주문 관리 시스템의 UI를 보여주는 클래스이다.
* 주문 내역을 보여준다.
* 주문 내역 중에서 주문의 상태를 전환할 수 있다.
* 현재 재고를 보여주고 재고의 상태를 전환할 수 있다.

Login\_System

* 서버에 id, pw를 보내서 로그인을 수행하는 클래스이다.

Ordering\_System

* 서버에 주문 목록을 요청하고 그 목록을 화면에 보여주는 클래스이다.

또한 주문의 상태 전환을 처리한다.

Inventory\_System

* 현재 재고 상태를 화면에 보여준다.
* 재고목록의 상태를 바꿔준다.

3.3 “Server” 서브시스템 세부 설계

“Server" 시스템은 고객에게 메뉴 관련 정보를 제공하고 주문 정보를 저장한다. 또한, Customer 시스템과 Employee 시스템에서 받은 요청을 처리한다.

“Server" 시스템의 주요 함수는 다음과 같다.

\* signup(id, pw, info)

■ “Customer System"에서 새로운 회원 정보를 입력받아 저장한다.

✓ login(id, pw)

■ "Customer System" 혹은 “Employee System"에서 로그인 정보를 입력 받는다.

\* ordered\_menu\_list

■ 로그인한 고객에 대해 주문 내역을 속성으로 한다.

\* add\_order(customer\_id, menu, style, size, amount, additional\_info)

■ "Customer System"으로부터 받은 정보로 주문을 생성하고 장바구니에 추가한다.

\* pay(pay\_info)

■ 결제 시스템에게 결제 정보를 넘기며 결제를 요청한다.

\* modify\_inven\_list(inventory\_list)

■ 재고 목록을 수정한다.

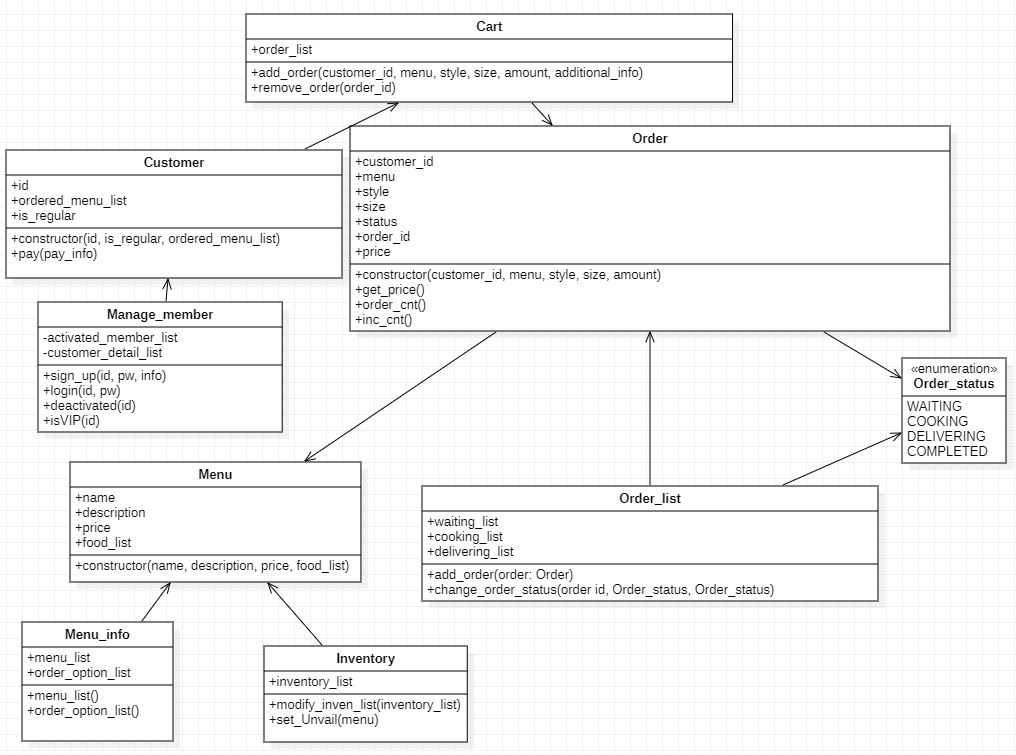
\* set\_Unvail(menu)

■ 재고가 없는 메뉴를 비활성화한다.

\* change\_order\_status(order\_id, from, to)

■ 지정한 order의 상태를 바꾼다.

- “Server” 서브시스템 메인 클래스도



Menu\_info

* 서비스의 기본 정보를 담고 있는 클래스이다.
* "Server" 시스템은 Menu\_info를 통해 "Customer System" 에게 메뉴 정보, 메뉴에 따른 주문 옵션 정보, 결제 정보를 넘겨준다.

Menu

* 메뉴 정보를 저장하는 type을 나타내는 클래스이다.
* 메뉴에 대한 설명이 저장되어 있다.

Manage\_Member

* 회원가입과 로그인을 관리하는 클래스이다.
* 회원가입과 로그인을 수행할 수 있으며, 현재 시스템에 로그인된 고객의 리스트를 갖는다.

Customer

* 고객 클래스이다.
* 고객 Member 객체, 장바구니를 속성으로 갖는다.
* pay 함수를 통해 장바구니에 담긴 주문에 대해 최종 결제를 할 수 있다.

Cart

* 장바구니 type 클래스이다.
* add\_order를 통해 장바구니에 주문 정보를 추가할 수 있다.

Order

* 주문 정보를 저장하는 type을 나타내는 클래스이다.
* 메뉴, 스타일, 메뉴 크기, 메뉴 수량, 추가 주문 정보를 저장한다.

Order\_list

* 주문 목록을 관리하는 클래스이다.
* change\_order\_status 함수로 각 주문의 상태를 전환 가능하다.

Inventory

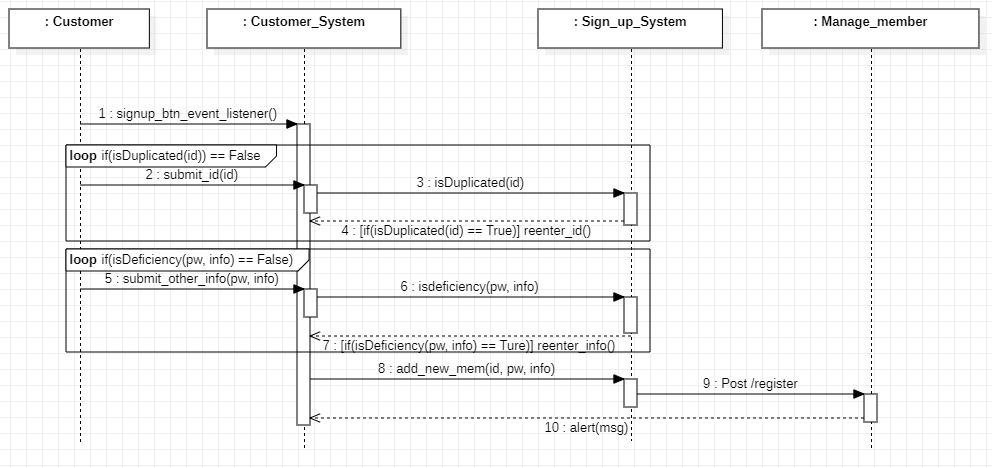
* 재고를 관리하는 클래스이다.
* 재고 목록을 수정할 수 있고, 재고의 상태에 따라 메뉴를 비활성화 시킬 수 있다.

Order\_status

* 주문의 상태를 나타내는 enumeration으로 주문의 네 가지 상태를 나타내는 데 사용된다.

4 교류도

4.1 회원 가입 (Sign Up)



1. 회원 가입 단추를 누르면 회원 가입 창인 signupform.html을 받아온다.
2. 사용할 아이디를 입력하면 자동으로 isDuplicated()를 실행하여 중복 검사를 한다.
3. 중복된 아이디이면 2로 돌아간다.
4. 중복되지 않은 아이디이면 비밀 번호를 비롯한 추가 정보를 입력한다.
5. 추가로 입력한 정보를 isDeficiency()로 검증한다.
6. 부족한 내용이 있으면 4로 돌아간다..
7. 부족한 내용이 없으면 Post /register를 호출하여 회원 정보를 저장하고, 가입 완료를 표시한다.

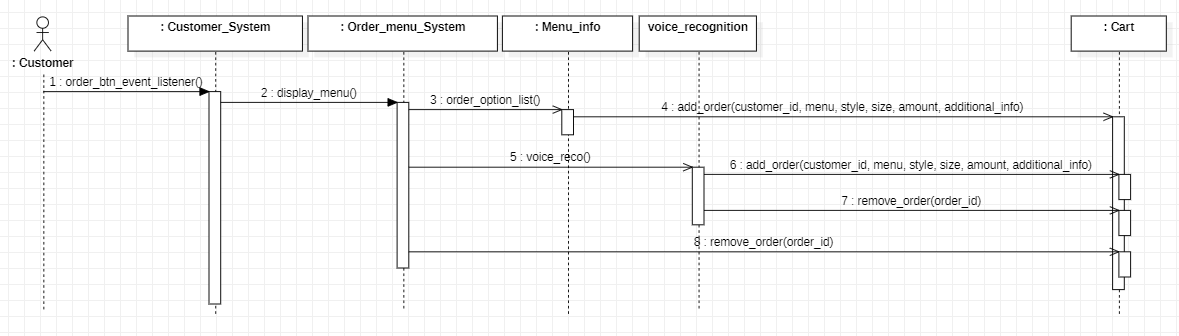
4.2 로그인 (Login)

테이블이(가) 표시된 사진

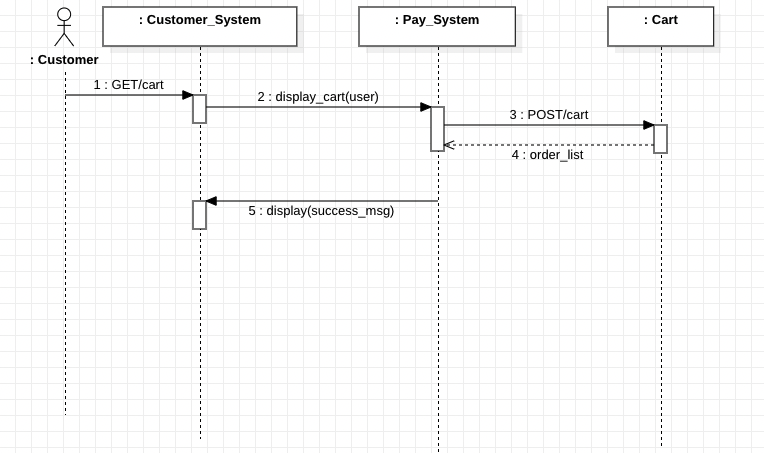
자동 생성된 설명

1. 로그인 단추를 눌러 로그인 창인 login.html을 가져온다.
2. 로그인 창에 아이디와 비밀번호를 입력한다.
3. 아이디나 비밀번호 가운데 입력하지 않은 것이 있으면 입력하라고 안내한다.
4. 둘 다 입력했으면 login()를 호출하여 로그인을 시도한다.
5. 없는 계정이라고 반환되면 2로 돌아간다.
6. 계정이 있으면 아이디로 고객인지 직원인지 판단한다.
7. 고객이면 고객 창을 띄우고, 직원이면 직원 창을 띄운다.

4.3 주문 (Order)



1. 고객이 주문 단추를 눌러 order.html를 받아온다.
2. Display\_menu()를 호출하여 메뉴를 불러온다.
3. 고객이 메뉴를 선택하면 order\_option\_list()를 호출하여 세부 메뉴를 표시한다.
4. 고객이 모든 선택을 마치면 add\_order()를 호출하여 장바구니에 고객이 선택한 메뉴를 추가한다.
5. 고객이 음성 인식 단추를 누르면 voice\_reco()를 호출하여 음성 인식 기능을 활성화한다.
6. 음성 인식 중에는 음성 인식이 자동으로 고객의 주문을 파악해 add\_order()로 장바구니에 넣는다.
7. 음성 인식으로 장바구니 내용을 지울 수 있다.
8. 음성 인식이 아닌 장바구니 단추로도 장바구니 내용을 지울 수 있다.

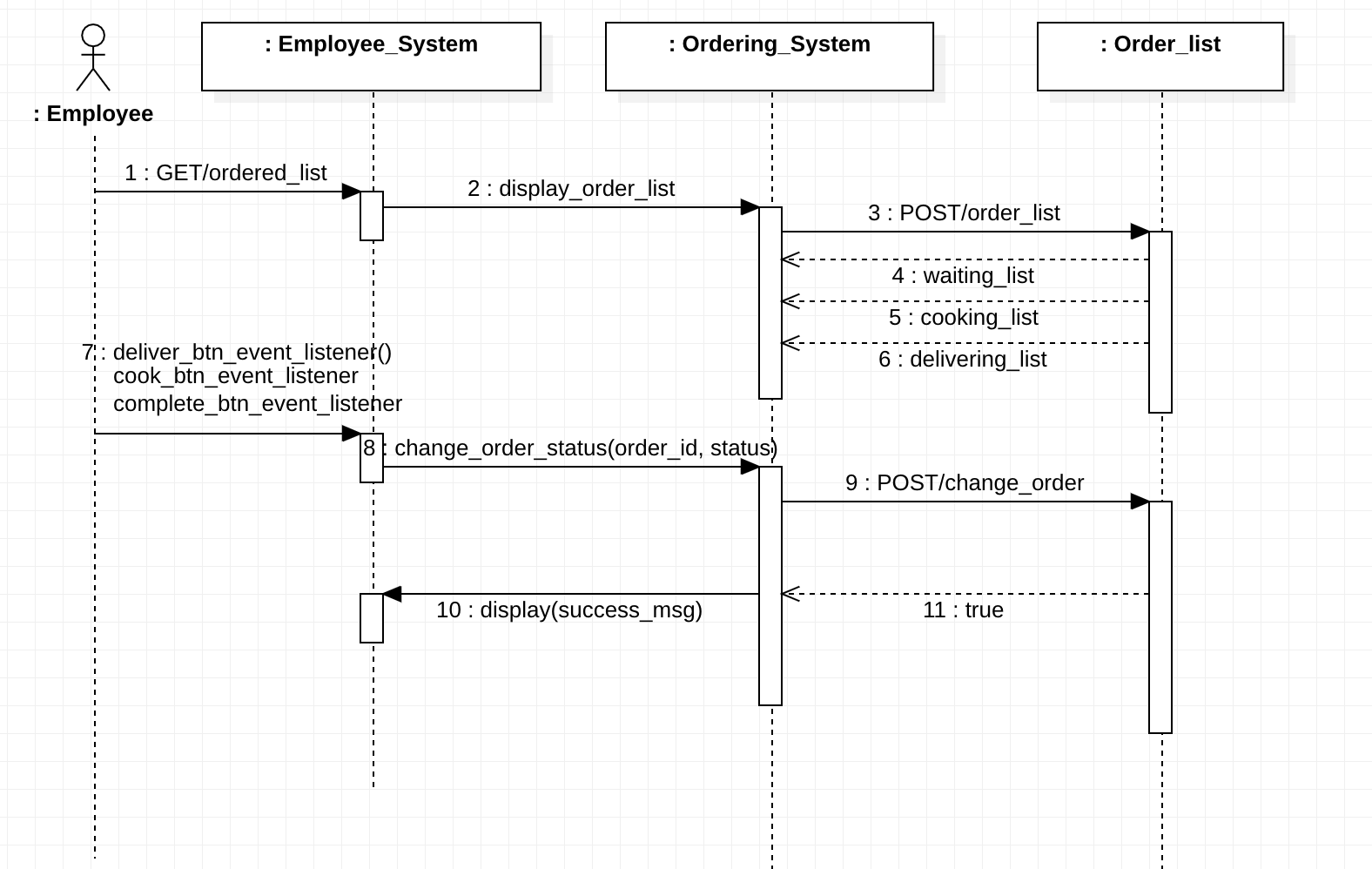
4.4 내 주문목록 조회 (Check My order)

1. 고객이 주문목록 조회 버튼을 눌러 cart.html를 받아온다.
2. display\_cart()를 호출하여 주문 목록을 불러온다.
3. Pay\_System에서는 POST /cart를 통해서 장바구니를 불러오는 주문을 서버에 전달하고

order\_list를 받는다.

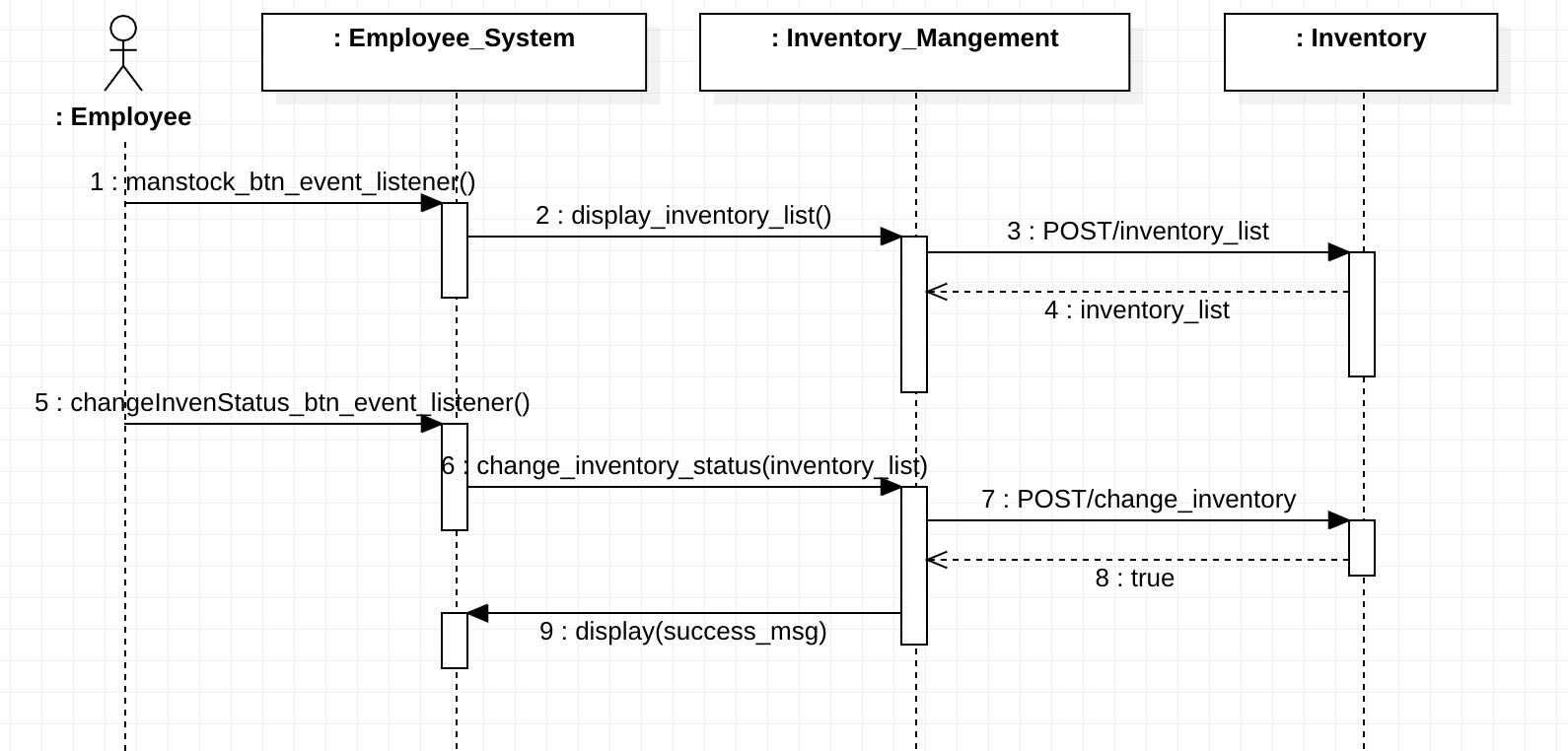
4 . Customer view에 성공적으로 완료되었다는 창이 뜬다.

4.5 주문 관리 (Check order)



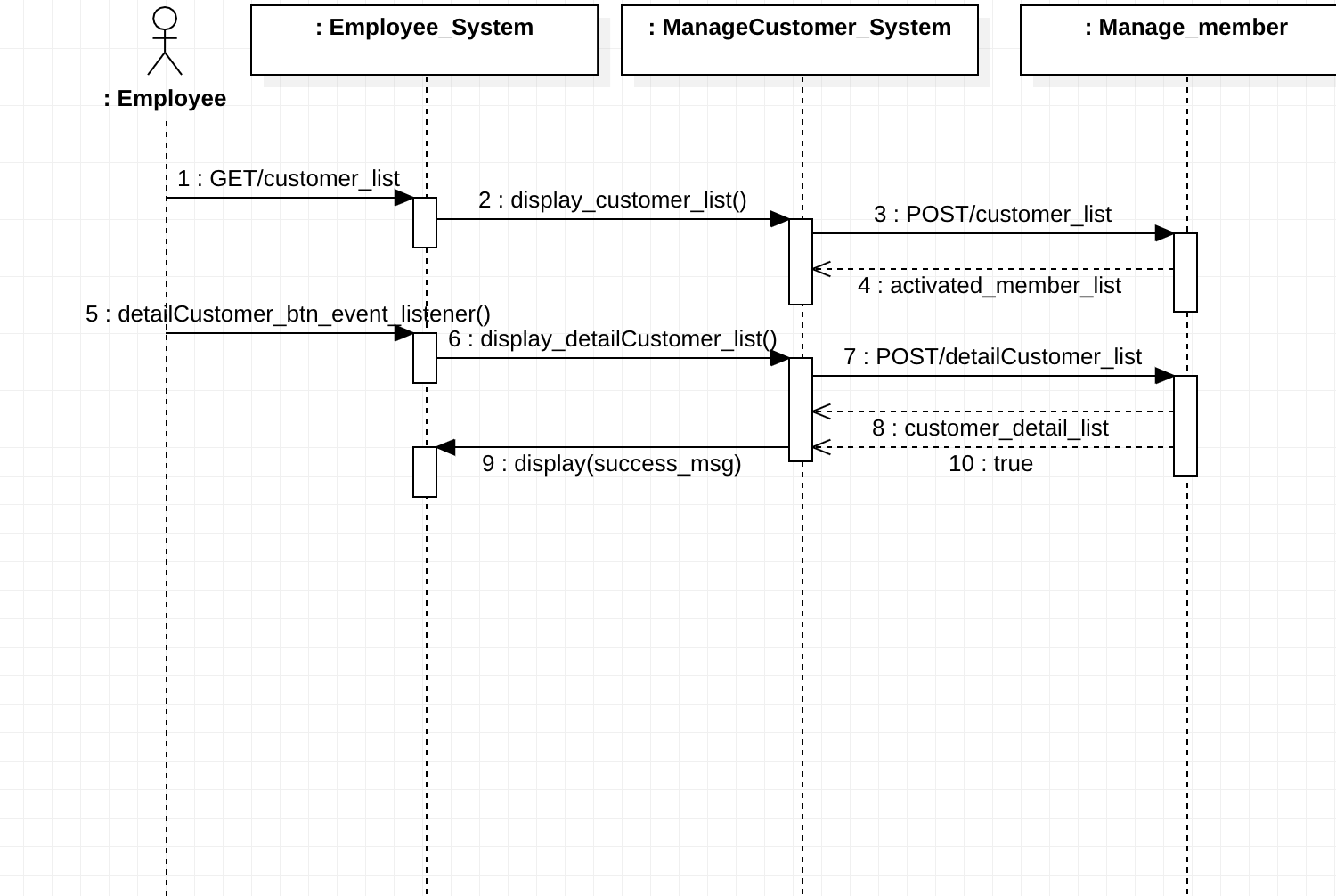
1. “Ordered\_list” 버튼을 클릭하면 GET/ordered\_list를 호출하고 ordered\_list.html을 받아서 완료된 주문 내역 페이지를 출력한다.
2. 또한 display\_order\_list를 호출하여 현재 주문 목록을 받아 출력한다. 현재 주문 목록을 받을 때에는 서버에 POST/order\_list를 통해서 받는다.
3. 현재 주문 중에서 상태를 전환할 주문들을 선택하고 각각 변경하고 싶은 상태(cooking, delivering, completed)에 맞춰 cook\_btn\_event\_listener 또는 deliver\_btn\_event\_listener 또는 complete\_btn\_event\_listener를 호출하면 Ordering\_System의 change\_order\_status(order\_id, status)가 호출된다.
4. Ordering\_System에서는 POST /change\_order를 통해서 상태를 전환할 주문을 서버에 전달한다.
5. 정상적으로 주문의 상태 전환이 완료되면 true를 return한다.
6. Employee view에 성공적으로 완료되었다는 창이 뜬다.

**4.6 재고 관리 (Manage Inventory)**



1. “재고 관리” 버튼을 클릭하면 manstock\_btn\_event\_listener를 호출하고 inventory\_list.html을 받아서 재고 관리 페이지를 출력한다.
2. 또한 display\_inventory\_list를 호출하여 현재 재고 관리 목록을 받아 출력한다. 현재 재고 관리 목록을 받을 때에는 서버에 POST/inventory\_list를 통해서 받는다.
3. 현재 재고 중에서 상태를 전환할 재고들을 선택하고 changeInvenStatus\_btn\_event\_listener를 호출하면 Inventory\_Management의 change\_inventory\_status(inventory\_list)가 호출된다.
4. Inventory\_Management에서는 POST /change\_inventory를 통해서 상태를 전환할 주문을 서버에 전달한다.
5. 정상적으로 재고 관리 목록의 상태 전환이 완료되면 true를 return한다.
6. Employee view에 성공적으로 완료되었다는 창이 뜬다.

**4.7 고객 관리 (Manage Customer)**



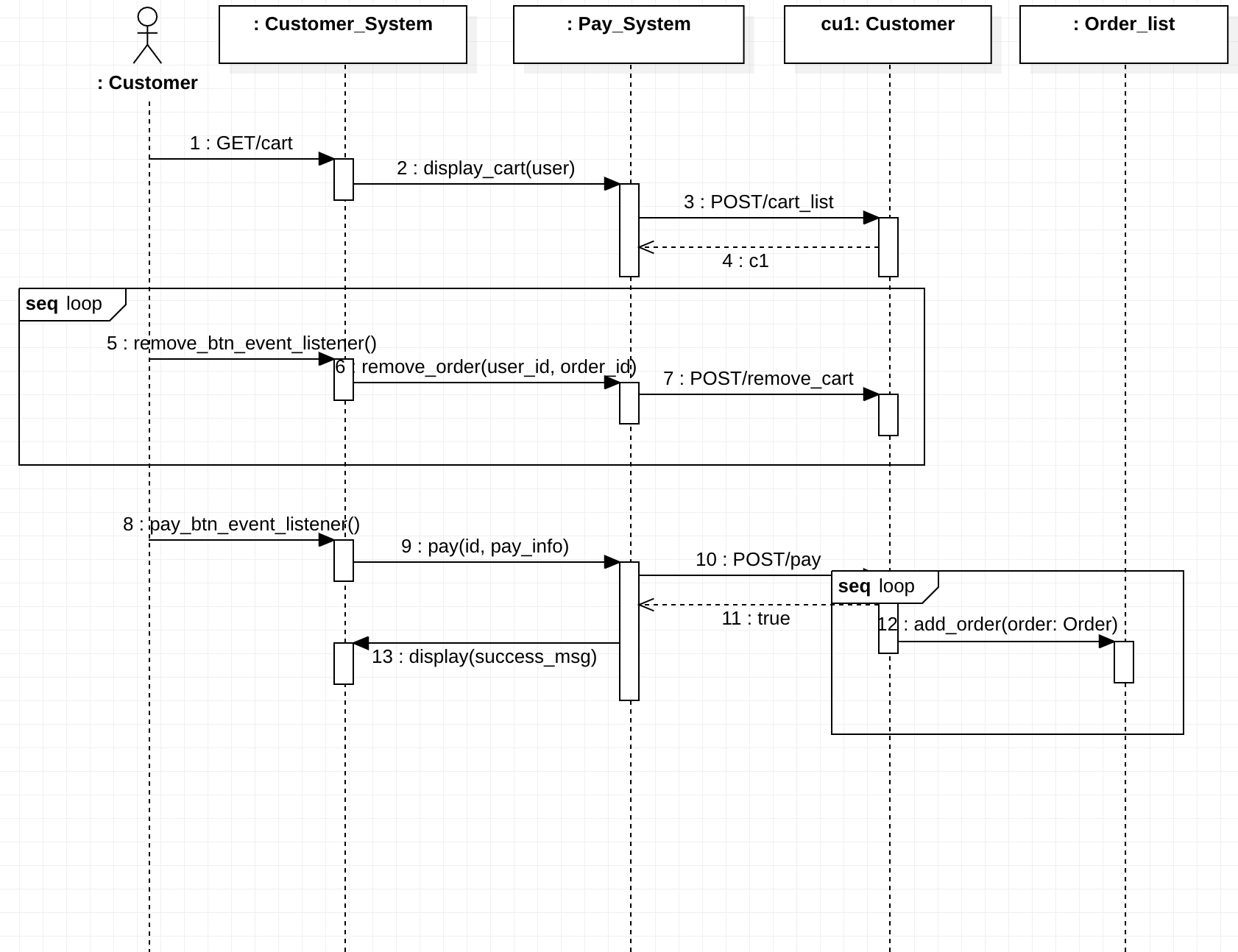
1. “고객 관리” 버튼을 클릭하면 GET/customer\_list를 호출하고 customer\_list.html을 받아서 고객 관리 페이지를 출력한다.
2. 또한 display\_customer\_list를 호출하여 현재 고객 관리 목록을 받아 출력한다. 현재 고객 관리 목록을 받을 때에는 서버에 POST/customer\_list를 통해서 받는다.
3. 현재 고객 중에서 상세 정보를 확인하고 싶은 고객을 선택하고

detailCustomer\_btn\_event\_listener를 호출하면 ManageCustomer\_System 의 display\_detailCustomer\_list가 호출된다.

1. ManageCustomer\_System 에서는 POST /detailCustomer\_list를 통해서 상세 회원 정보를 서버에 전달한다.
2. 정상적으로 상세 회원 정보 리스트 출력이 완료되면 true를 return한다.
3. Employee view에 성공적으로 완료되었다는 창이 뜬다.

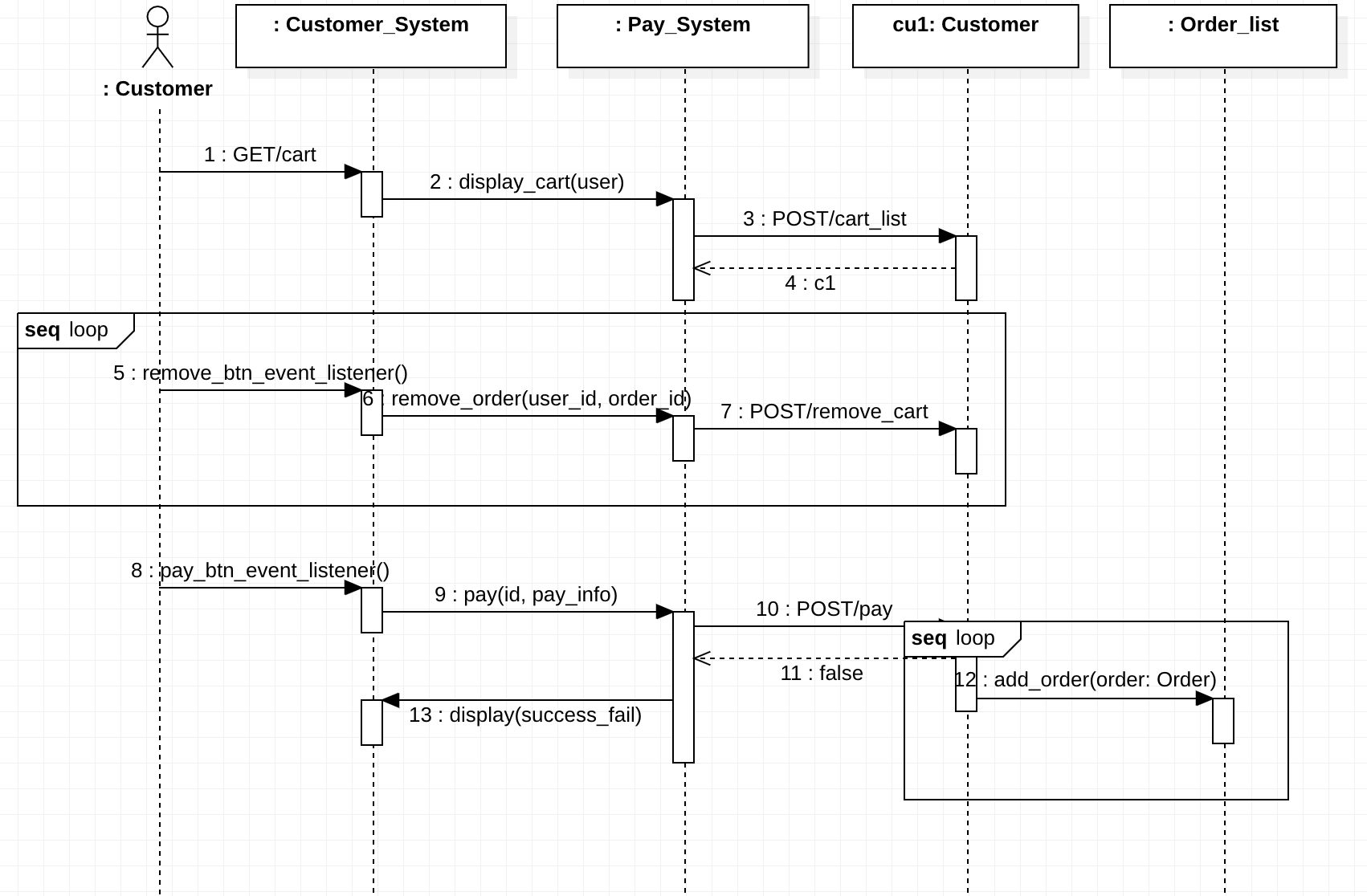
**4.8 결제 (Pay)**

**주흐름**

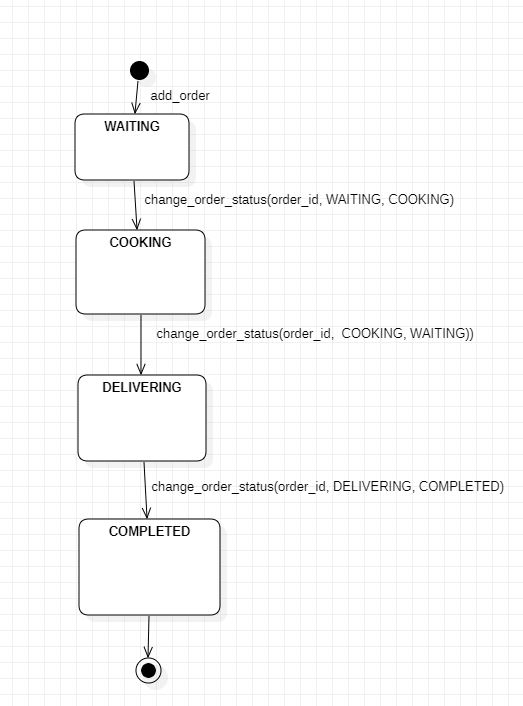
****

1. 고객이 “장바구니” 버튼을 누르면 GET /cart이 호출되고 cart.html을 받아 장바구니 화면을 출력한다.
2. 또한 display\_cart(user)를 Pay\_System에 호출하여 고객의 장바구니를 화면에 나열한다. Pay\_System에서 고객의 장바구니를 얻기 위해서는 POST /cart\_list를 통해서 서버에 있는 Customer객체의 장바구니 목록을 받는다.
3. 삭제할 주문이 있는 경우 삭제할 주문의 “취소하기” 버튼을 누르면 remove\_btn\_event\_listener()를 통해서 Pay\_System 에 유저 id와 주문 id를 넘겨준다.
4. Pay\_System에서는 POST /remove\_cart를 통해서 선택된 주문을 서버에서 제외한다.
5. “결제” 버튼을 누르면 pay\_btn\_event\_listener를 호출하고 Pay\_System의 pay()를 실행한다.
6. Pay\_System에서는 POST /pay를 통해서 서버에서 고객 객체가 주문을 수행하도록 한다.
7. Customer객체에서는 장바구니에 있는 모든 주문에 대하여 add\_order를 호출하여 현재 주문 목록인 Order\_list에 주문을 추가한다.
8. 만약 결제를 성공하면 true를 return하고 Pay\_System에서는 결제 성공 메시지를 띄운다.
9. 만약 결제를 실패하면 false를 return하고 Pay\_System에서는 결제 실패 메시지를 띄운다.

부흐름



5 Order 객체 상태도



Order 클래스는 주문을 나타내는 type 클래스이다. Order 클래스는 다음과 같은 상태 변화를 갖는다.

\* WAITING

■ Order 객체가 생성되면 WAITING 상태가 된다.

■ add\_order 메소드를 통해 Order 객체가 생성되고 WAITING이 된다.

\* COOKING

■ 조리중인 상태를 나타낸다.

■ change\_order\_status 메소드를 통해 COOKING 상태로 전환된다.

\* DELIVERING

■ 배달 중인 상태를 나타낸다.

■ Order 객체는 change\_order\_status 메소드를 통해 DELIVERING 상태로 전환된다.

\* COMPLETED

■ 배달이 완료된 주문을 나타내는 상태이다.

■ Order 객체는 change\_order\_status 메소드를 통해 COMPLETED 상태로 전환된다.