<뚝배기>
<MrGreat>
설계 산출물

문서버전	3.1
문서 ID	SE-2020-002
최종변경일	2020-11-15
문서상태	릴리즈

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	<뚝배기> <MrGreat> 설계 산출물		
문서 ID	SE-2020-002		
버전		변경일	설명
1	0	2020-11-04	아키텍처도를 추가했다. “Staff Client” 클래스도를 추가했다. “Server” 클래스도를 추가했다.
	1	2020-11-04	“Customer Client” 클래스도를 추가했다. “Customer Client” 서브시스템 세부설계 설명을 추가했다. “Staff Client” 서브시스템 세부설계 설명을 추가했다. “Server” 서브시스템 세부설계 설명을 추가했다.
	2	2020-11-05	교류도를 추가했다. 상태도를 추가했다.
2	0	2020-11-05	릴리즈
	1	2020-11-06	오타 수정
3	0	2020-11-14	[피드백 반영] 서브시스템 설계 및 클래스도 수정
	1	2020-11-15	[피드백 반영] 교류도, 상태도 수정

1 개 요

1.1 목 적

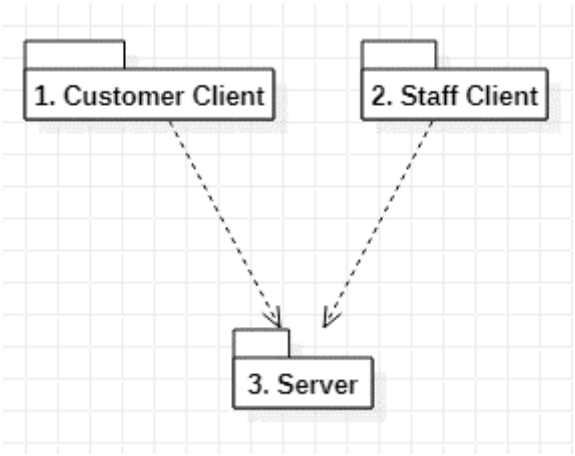
본 문서는 미스터 대박 디너 서비스(Mr.Great)의 산출물을 기술한다.

- ✓ 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 Client 시스템
- Staff Client 시스템
- Sever 시스템

고객 Client 시스템

- 서버와 통신하며 고객에게 정보를 보여주는 역할을 한다.
- Javascript를 이용하여 작성된다.

직원 Client 시스템

- 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 Client" 서브시스템 세부 설계

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

“Customer Client" 시스템의 주요 함수는 다음과 같다.

✓ login(id, pw)

■ “Server"에 id, pw를 보내 로그인을 수행한다.

✓ add_new_member(id, pw)

■ 새로운 회원 정보를 입력받아 “Server”에 전송한다.

✓ display_menu()

■ 매뉴의 목록과 매뉴에 대한 설명, 그리고 최근에 고객이 주문한 목록을 “Server”에 요청하고 보여준다.

✓ add_order(menu, order_options)

■ 고객이 입력한 주문 정보를 받아 “Server”에 전송하여 장바구니에 담는다.

✓ display_basket()

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

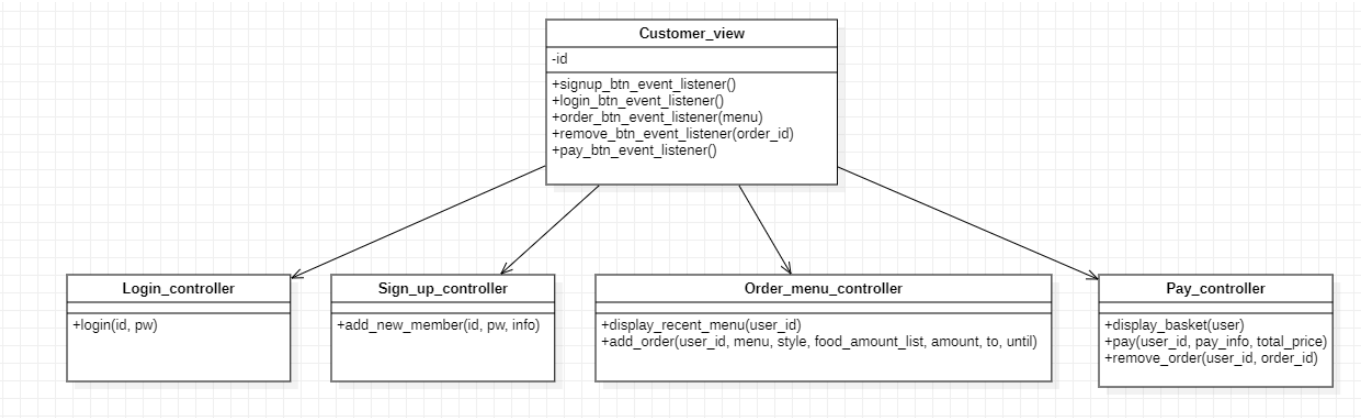
✓ pay(id, pay_info)

■ 결제 방법을 선택하고 주문 정보를 “Server”에 보낸다.

✓ remove_order(order_id)

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

- “Customer Client” 서브시스템 메인 클래스도



Customer_view

- ✓ 고객이 사용하는 주문 시스템의 UI 를 보여주는 클래스이다.
- ✓ 고객이 회원가입과 로그인을 진행할 수 있다.
- ✓ 매뉴와 장바구니를 보여주고 장바구니에 담기와 결제를 진행할 수 있다.
- ✓ 장바구니에 있는 주문 중에서 제외할 주문을 장바구니에서 제외할 수 있다.

Login_controller

- ✓ 서버에 id, pw 를 보내서 로그인을 수행하는 클래스이다.

Sign_up_controller

- ✓ 서버에 id, pw 를 보내서 회원가입을 수행하는 클래스이다.

Order_menu_controller

- ✓ 서버에 매뉴 목록을 요청하고 그 목록을 화면에 보여주는 클래스이다.
- ✓ 주문하기를 수행해서 고객이 주문하려는 주문을 장바구니에 담을 수 있다.

Pay_controller

- ✓ 서버에 장바구니 목록을 요청하고 그 목록을 화면에 보여주는 클래스이다.
- ✓ 제외할 주문 목록을 선택하여 장바구니에서 제외할 수 있다.
- ✓ 결제 방법을 선택하여 최종적인 결제를 수행할 수 있다.

3.2 “Staff Client” 서브시스템 세부 설계

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

“Staff Client" 시스템의 주요 함수는 다음과 같다.

✓ login(id, pw)

■ “Server"에 id, pw를 보내 로그인을 수행한다.

✓ display_order_lists()

■ “Server"에서 현재 주문 목록을 받아 보여준다.

✓ display_order_history()

■ “Server"에서 과거 주문 목록을 받아 보여준다.

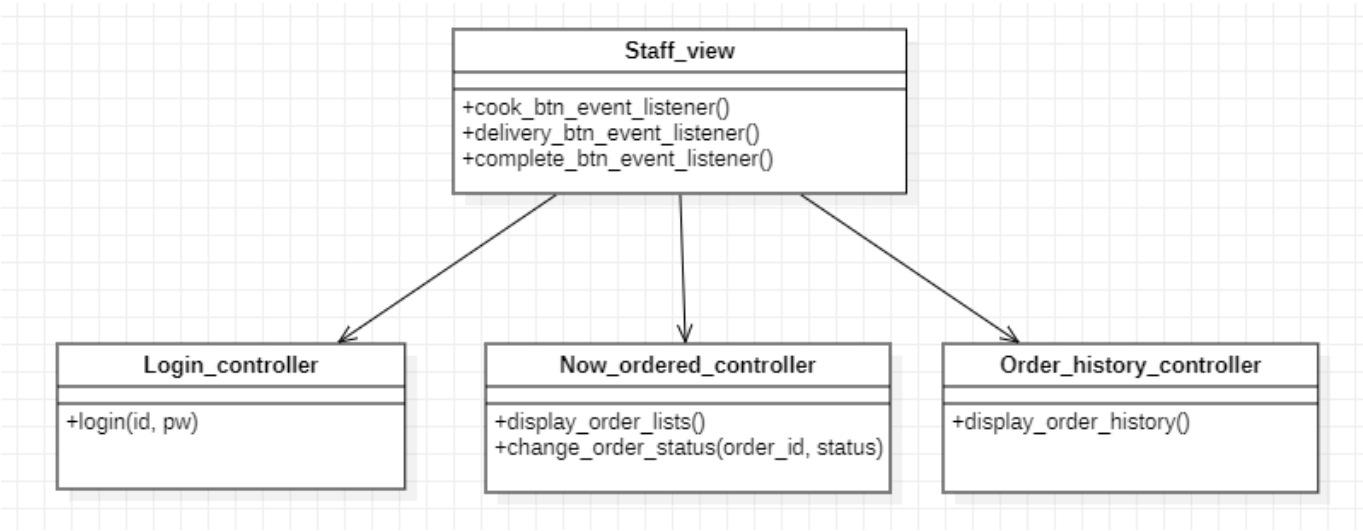
✓ change_order_status(oreder, status)

■ 현재 주문 목록 중에서 상태를 전환할 주문을 “Server"에 보내서 상태를 바꾼다.

✓ cook_btn_event_listener(), delivery_btn_event_listener(), complete_btn_event_listener()

■ 선택된 현재 주문 목록에 대해서 각 버튼에 맞게 change_order_status를 호출한다.

- “Staff Client” 서브시스템 메인 클래스도



Staff_view

- ✓ 직원이 사용하는 주문 관리 시스템의 UI를 보여주는 클래스이다.
- ✓ 현재 주문 내역과 과거 주문 내역을 보여준다.
- ✓ 현재 주문 내역 중에서 주문의 상태를 전환할 수 있다.

Login_controller

- ✓ 서버에 id, pw를 보내서 로그인을 수행하는 클래스이다.

Now_ordered_controller

- ✓ 서버에 현재 주문 목록을 요청하고 그 목록을 화면에 보여주는 클래스이다.
- ✓ 또한 현재 주문의 상태 전환을 처리한다.

Order_history_controller

- ✓ 서버에 과거 주문 목록을 요청하고 그 목록을 화면에 보여주는 클래스이다.

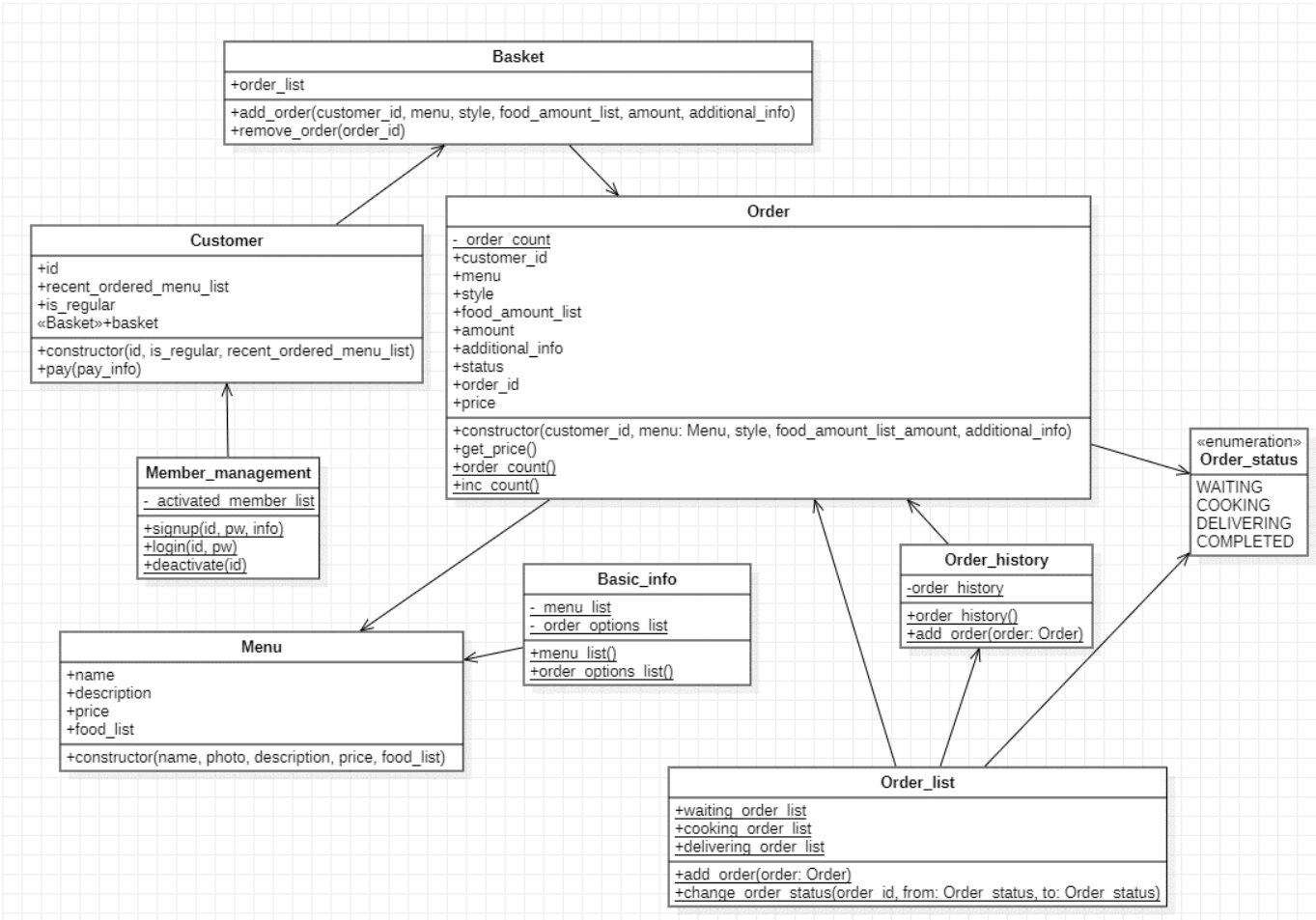
3.3 “Server” 서브시스템 세부 설계

“Server" 시스템은 고객에게 메뉴 관련 정보를 제공하고 주문 정보를 저장한다. 또한, "Customer Client”와 “Staff Client”에게 받은 요청을 처리하는 역할을 한다.

“Server" 시스템의 주요 함수는 다음과 같다.

- ✓ signup(id, pw, info)
  - ■ “Customer Client"에서 새로운 회원 정보를 입력받아 저장한다.
- ✓ login(id, pw)
  - ■ "Customer Client" 혹은 “Staff Client"에서 로그인 정보를 입력받아 사용자를 식별한다.
- ✓ recent_ordered_menu_list
  - ■ 로그인한 고객에 대해 최근 주문 내역을 반환한다.
- ✓ add_order(customer_id, menu, style, food_amount_list, amount, additional_info)
  - ■ "Customer Client" 시스템으로부터 전송받은 정보로 주문 정보를 생성하고 장바구니에 추가한다.
- ✓ pay(pay_info)
  - ■ 결제 시스템에게 결제 정보를 넘기며 결제를 요청한다.
- ✓ order_history()
  - ■ 최근 열 개의 주문 내역을 반환한다.
- ✓ change_order_status(order_id, from, to)
  - ■ 지정한 order 의 상태를 from 에서 to 로 바꾼다.

- “Server” 서브시스템 메인 클래스도



Basic_info

✓ 미스터 대박 서비스의 기본적인 정보를 담고 있는 클래스이다.

✓ "Server" 시스템은 Basic_info를 통해 "Customer Client" 시스템에게 메뉴 정보, 메뉴에 따른 주문 옵션 정보, 결제 정보를 넘겨준다.

Menu

✓ 메뉴 정보를 저장하는 type을 나타내는 클래스이다.

✓ 예시 사진과 메뉴 설명에 대한 정보를 담고 있다.

Member_management

✓ 회원가입과 로그인을 관리하는 클래스이다.

✓ 회원가입과 로그인을 수행할 수 있으며, 현재 시스템에 로그인된 고객의 리스트를 속성으로 갖는다.

Customer

✓ 고객을 나타내는 클래스이다.

✓ 고객을 식별하는 Member 객체, 장바구니를 속성으로 갖는다.

✓ pay 함수를 통해 장바구니에 담긴 주문에 대해 최종 결제를 할 수 있다.

Basket

✓ 장바구니 type 클래스이다.

✓ 장바구니는 주문 정보의 집합이며 add_order를 통해 장바구니에 주문 정보를 추가할 수 있다.

Order

✓ 주문 정보를 저장하는 type을 나타내는 클래스이다.

✓ 메뉴, 스타일, 빨 음식, 수량 정보, 추가 주문 정보를 저장하고 있으며 생성자로 주문 정보를 설정할 수 있다.

Order_list

✓ 아직 완료되지 않은 주문을 관리하는 클래스이다.

✓ 상태별로 클래스를 리스트로 저장하고 있으며, change_order_status 함수로 각 주문의 상태를 전환 가능하다.

Order_history

✓ 주문 내역을 관리하는 클래스이다.

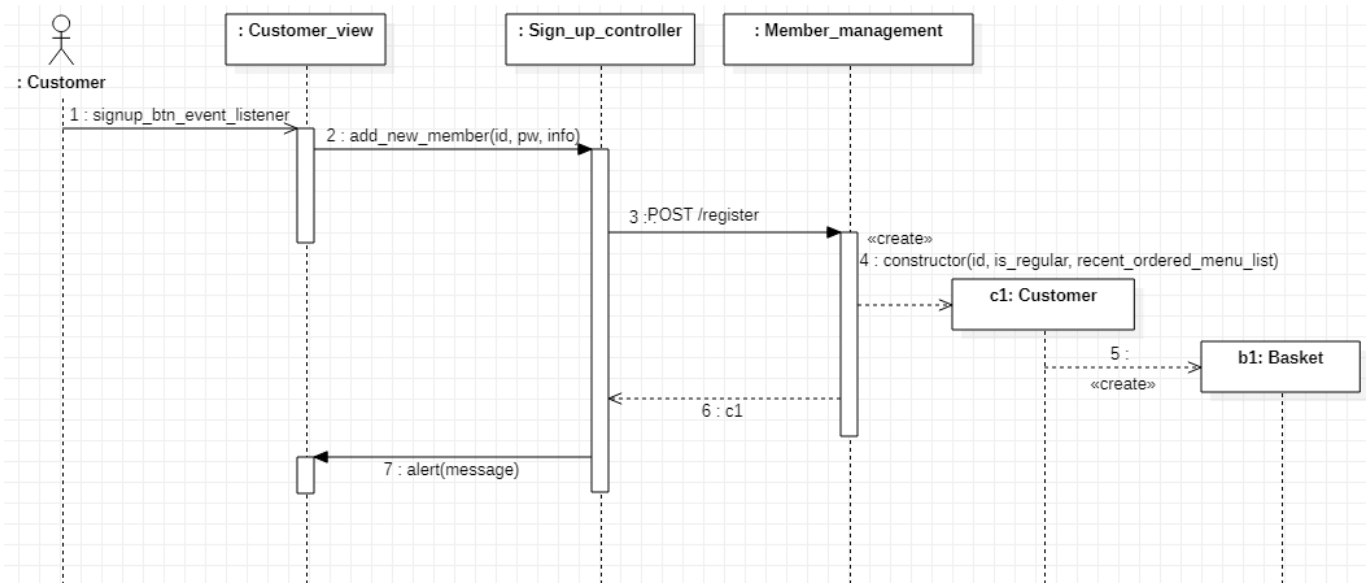
✓ 주문 내역을 저장하고 있으며, 주문이 완료될 시 order_history에 저장하는 역할을 한다.

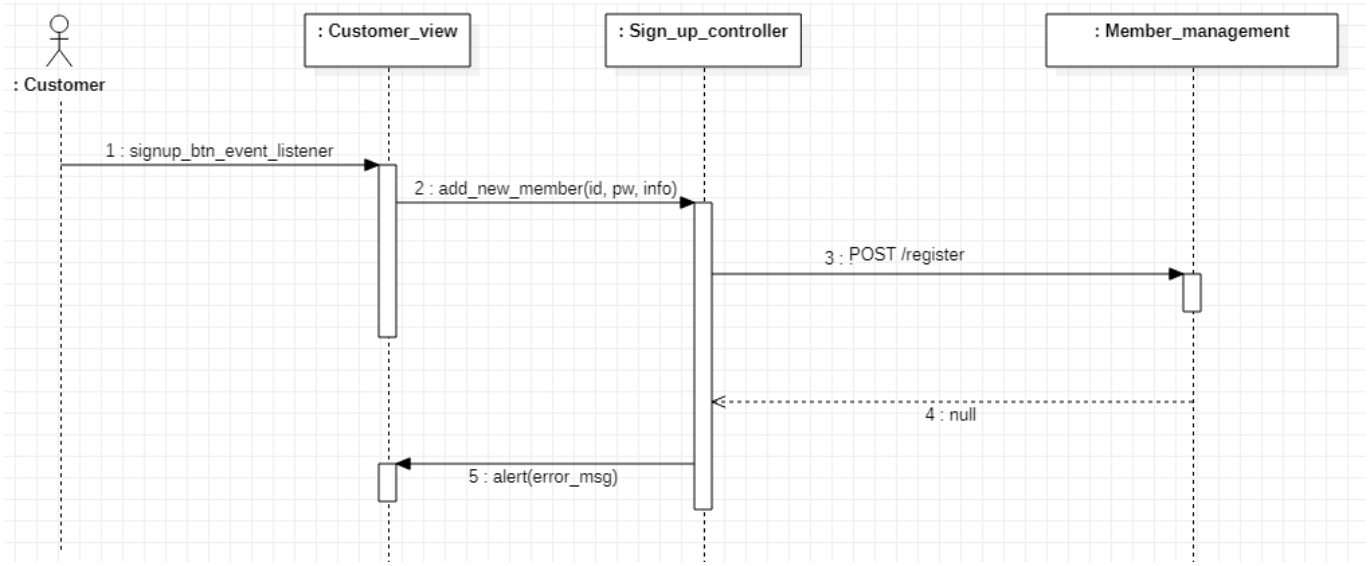
Order_status

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

4 교류도

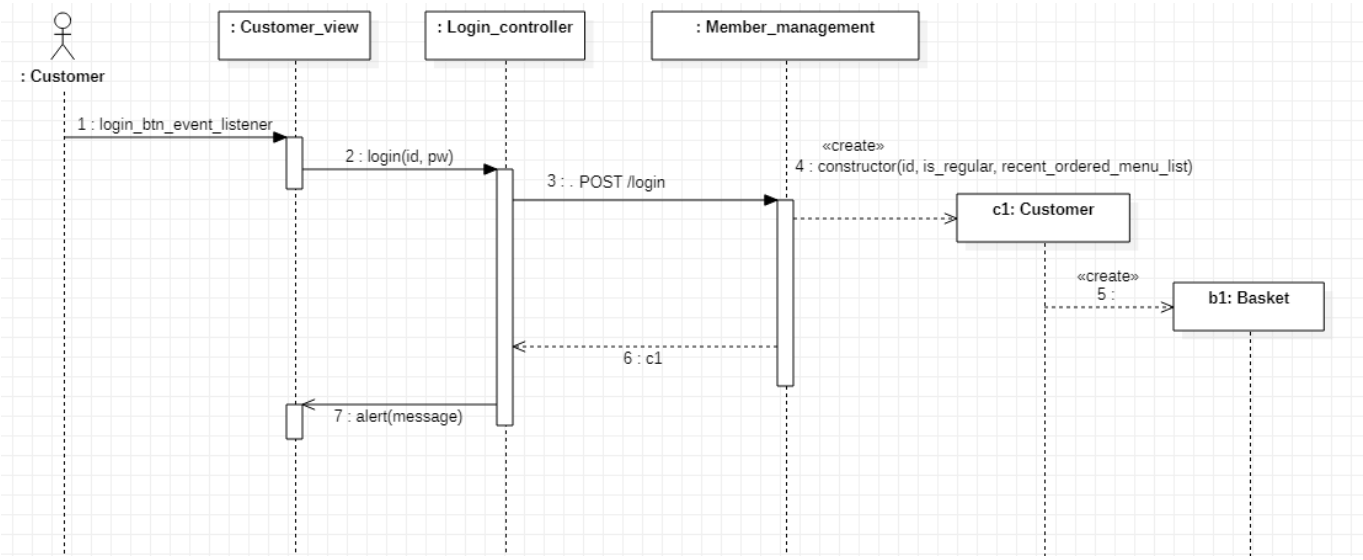
4.1 회원가입 (Sign Up)

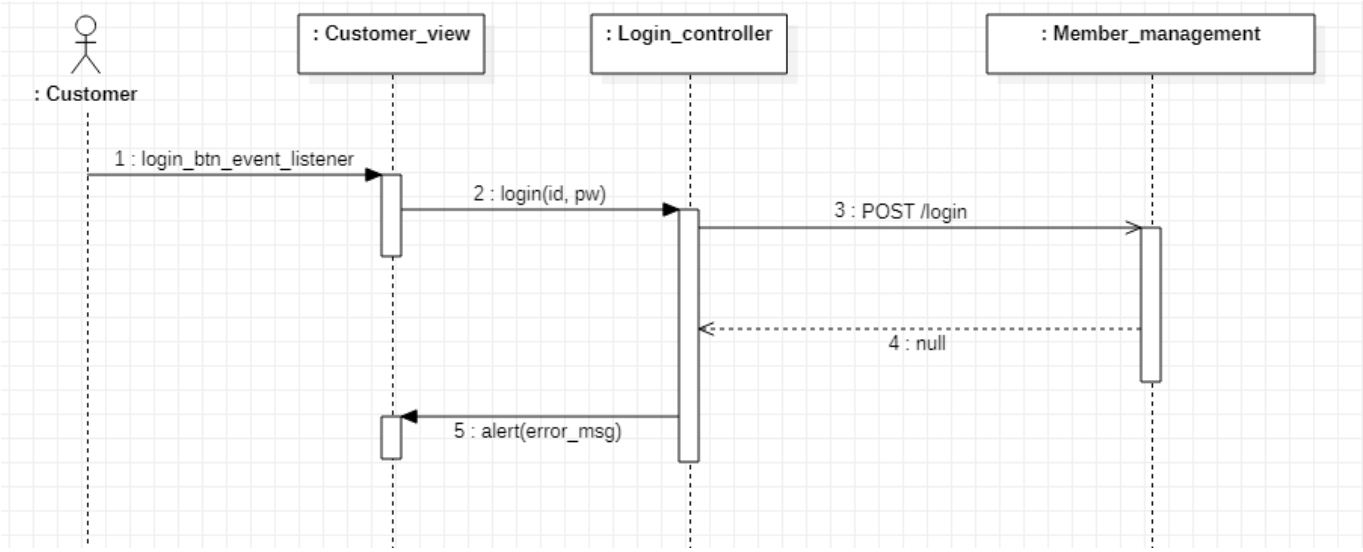




1. 고객이 '회원가입'을 선택하면 sign_up_btn_event_listener()가 호출된다.
2. Customer_view 에서는 Sign_up_controller 의 add_new_member(id, pw, info)를 통해서 입력한 정보를 전송한다.
3. Sign_up_controller 에서는 POST /register 를 통해서 Member_management 에 회원가입 요청을 한다.
4. 회원가입에 성공하면 Member_management 에서는 고객 객체를 생성하고 그 객체를 return 한다. Sign_up_controller 에서는 그 객체를 확인하고 회원가입 완료 메시지를 Customer view 에 보낸다.
5. 만약 이미 있는 아이디를 입력하여 회원가입을 실패하면 null 을 return 한다. Sign_up_controller 에서는 null 을 확인하고 회원가입 실패 메시지를 띄운다.

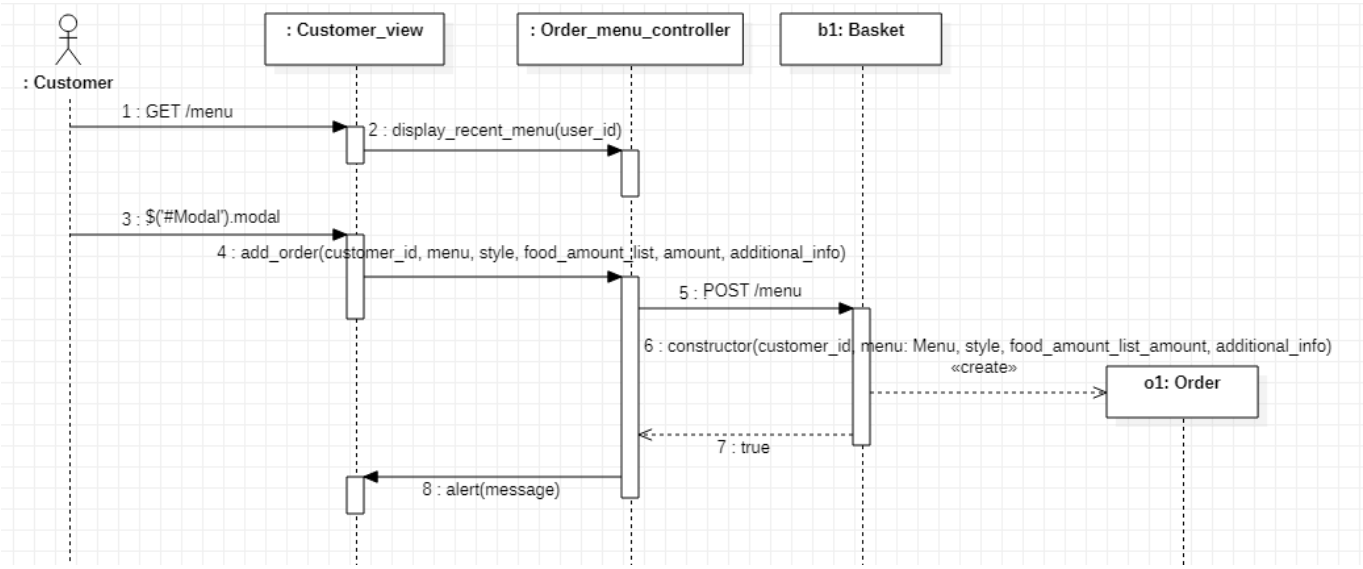
4.2 로그인 (Login)





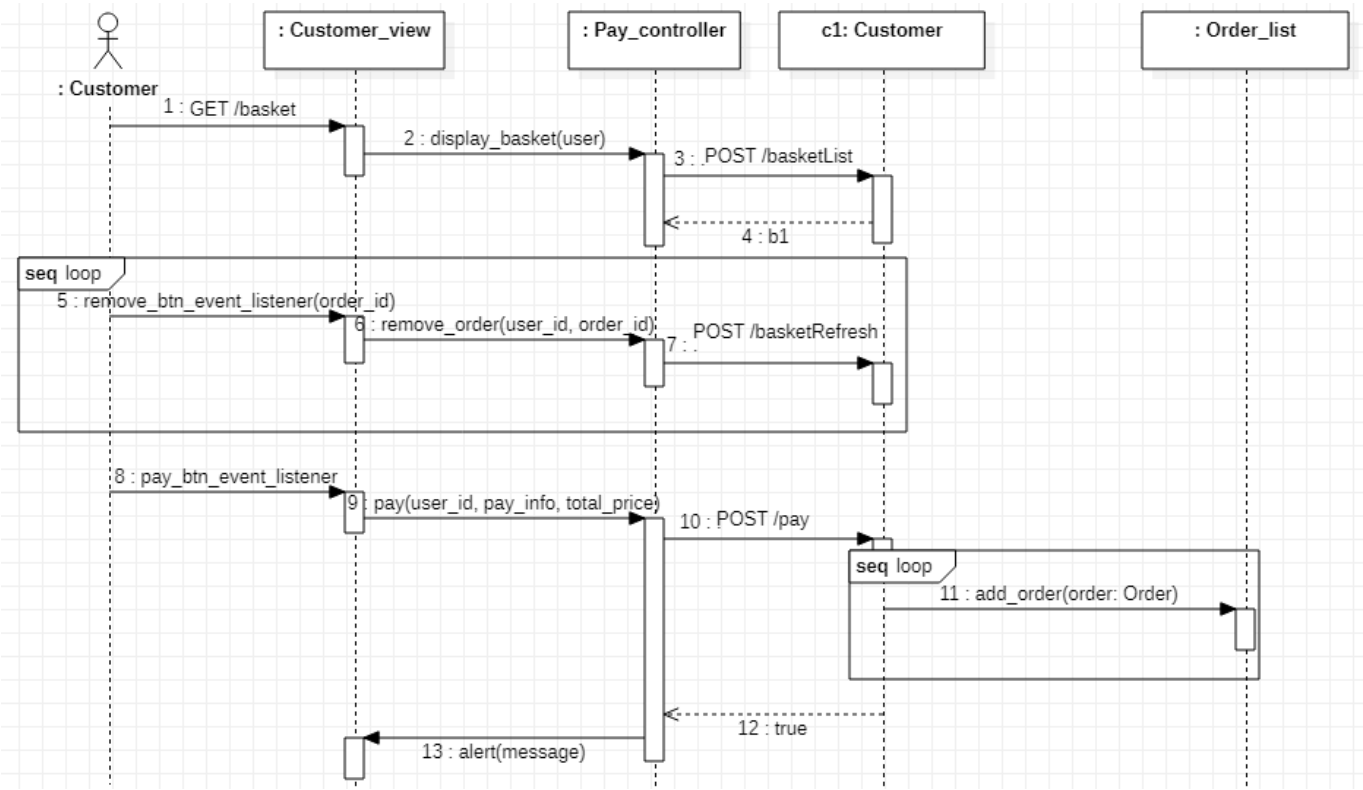
1. 고객이 아이디, 비밀번호를 입력하고 “로그인” 버튼을 누르면 login_btn_event_listener()가 호출되고 Login_controller 의 login(id, pw)가 호출된다.
2. POST /login 을 통해서 id, pw 를 Member_management 에 보내면 일치하는 id, pw 가 있는지 확인한다.
3. 만약 일치하는 id, pw 가 있다면 active customer 에 해당 고객을 추가하고 고객 객체를 return 한다.
4. Login_controller 에서 그 객체를 받으면 로그인 성공 메시지를 띄운다.
5. 만약 일치하는 id, pw 가 없다면 null 을 return 하고 Login_controller 에서는 로그인 실패 메시지를 띄운다.

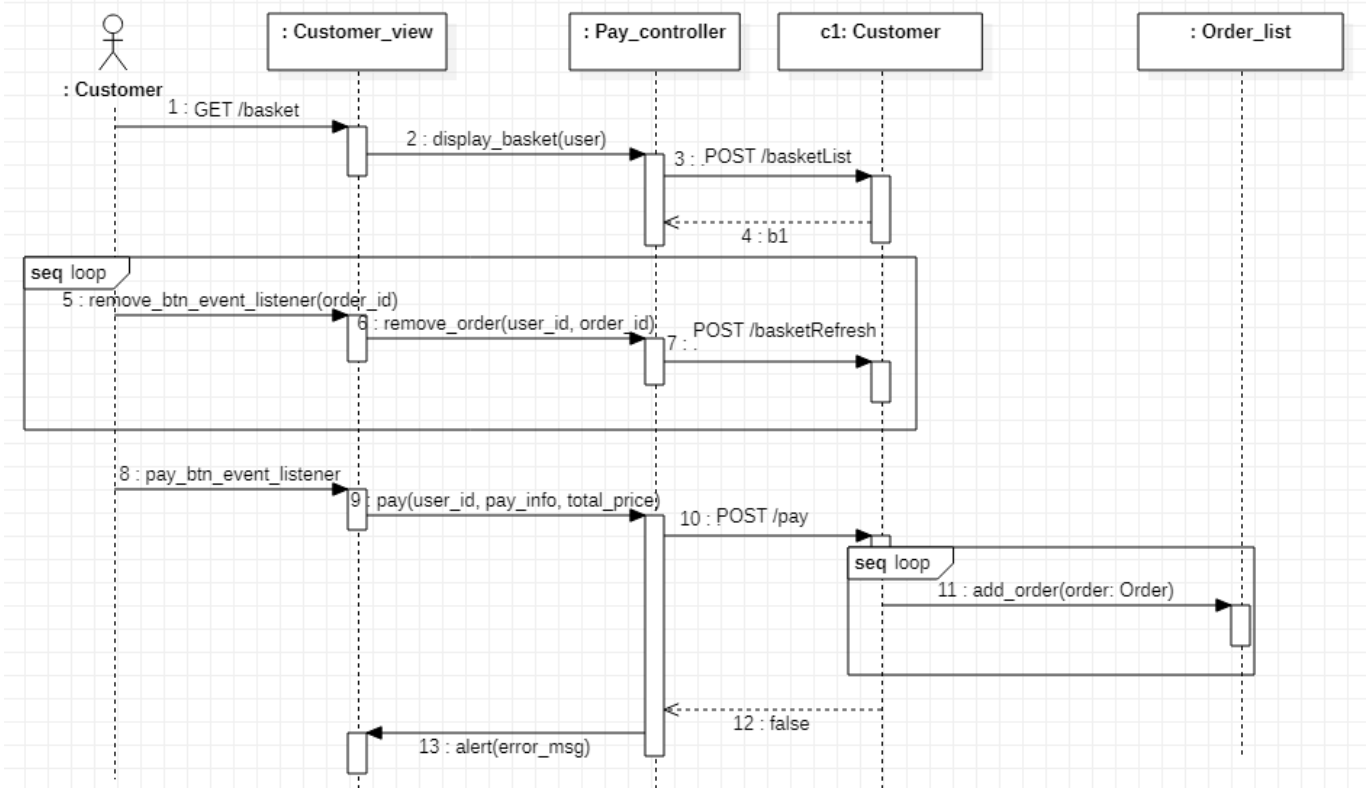
4.3 주문 (Order menu)



1. “매뉴”를 선택하면 GET /menu 가 호출되고 menu.html 을 받아서 매뉴 목록을 띄운다. 또한 display_recent_menu(user_id)를 통해서 해당 유저의 최근 주문 목록을 출력한다.
2. 어떤 매뉴의 “주문하기” 버튼을 누르면 부트스트랩 내장 함수인 $(#Modal).modal 을 통해서 매뉴의 상세 옵션이 보이고 “장바구니에 담기”를 누르면 Order_menu_controller 에 add_order()가 호출되어 현재 입력한 옵션에 맞는 주문을 전달한다.
3. Order_menu_controller 에서는 POST /menu 를 통해서 고객의 Basket 객체에 주문 객체 정보를 담는다.
4. Basket 에서는 true 를 return 하고 Order_menu_controller 에서는 주문 성공 메시지를 띄운다.

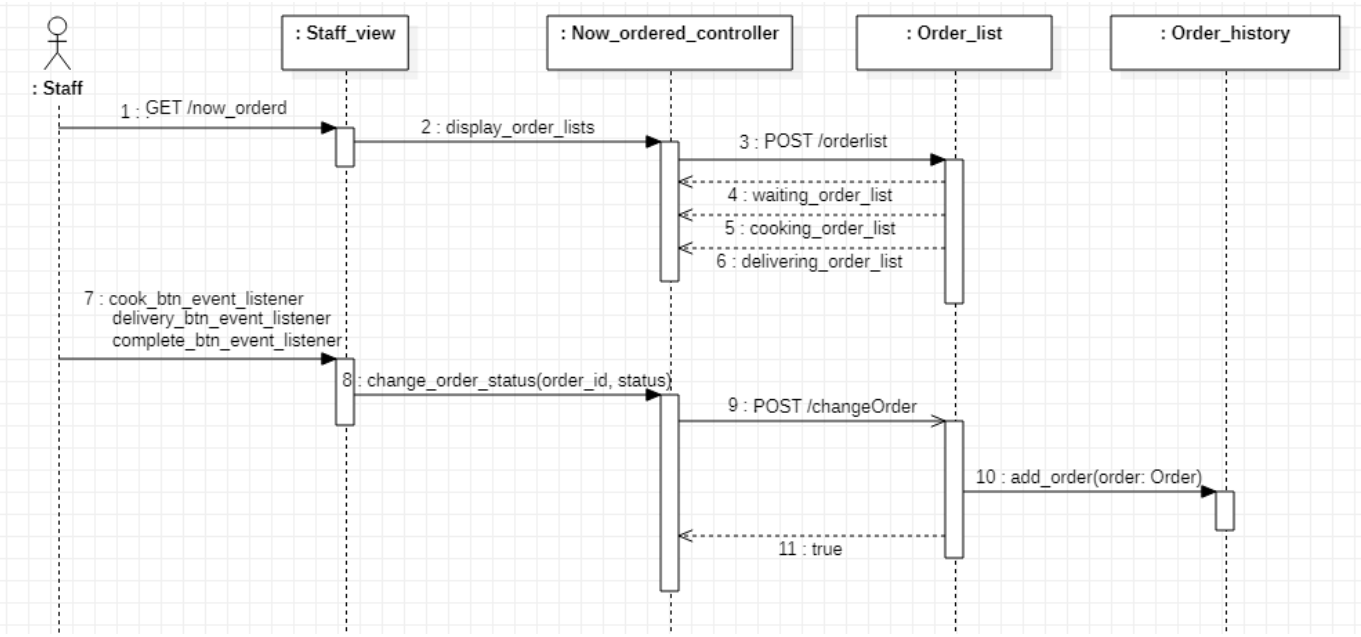
4.4 결제 (Pay)





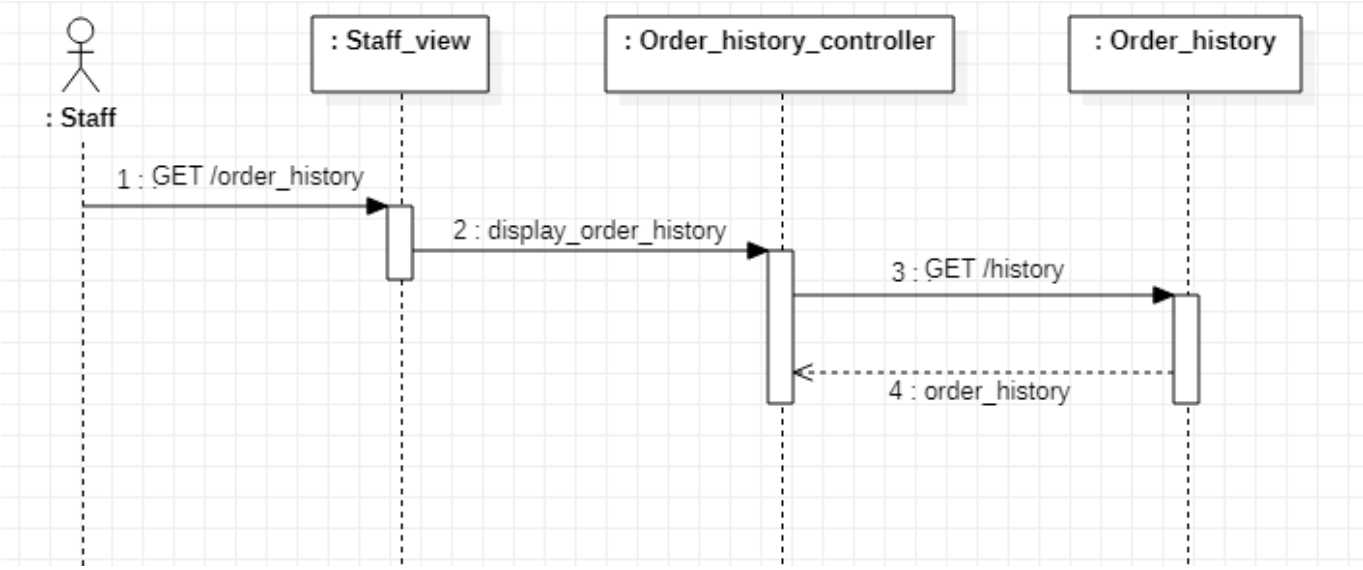
1. 고객이 “장바구니” 버튼을 누르면 GET /basket 이 호출되고 basket.html 을 받아 장바구니 화면을 출력한다.
2. 또한 display_basket(user)를 Pay_controller 에 호출하여 고객의 장바구니를 화면에 나열한다. Pay_controller 에서 고객의 장바구니를 얻기 위해서는 POST /basketList 를 통해서 서버에 있는 Customer 객체의 장바구니 목록을 받는다.
3. 삭제할 주문이 있는 경우 삭제할 주문의 “제외하기” 버튼을 누르면 remove_btn_event_listener(order_id)를 통해서 Pay_controller 에 유저 id 와 주문 id 를 넘겨준다.
4. Pay_controller 에서는 POST /basketRefresh 를 통해서 선택된 주문을 서버에서 제외한다.
5. “결제” 버튼을 누르면 pay_btn_event_listener 를 호출하고 Pay_controller 의 pay()를 실행한다.
6. Pay_controller 에서는 POST /pay 를 통해서 서버에서 고객 객체가 주문을 수행하도록 한다.
7. Customer 객체에서는 장바구니에 있는 모든 주문에 대하여 add_order 를 호출하여 현재 주문 목록인 Order_list 에 주문을 추가한다.
8. 만약 결제를 성공하면 true 를 return 하고 Pay_controller 에서는 결제 성공 메시지를 띄운다.
9. 만약 결제를 실패하면 false 를 return 하고 Pay_controller 에서는 결제 실패 메시지를 띄운다.

4.5 주문 처리 (Process Order)



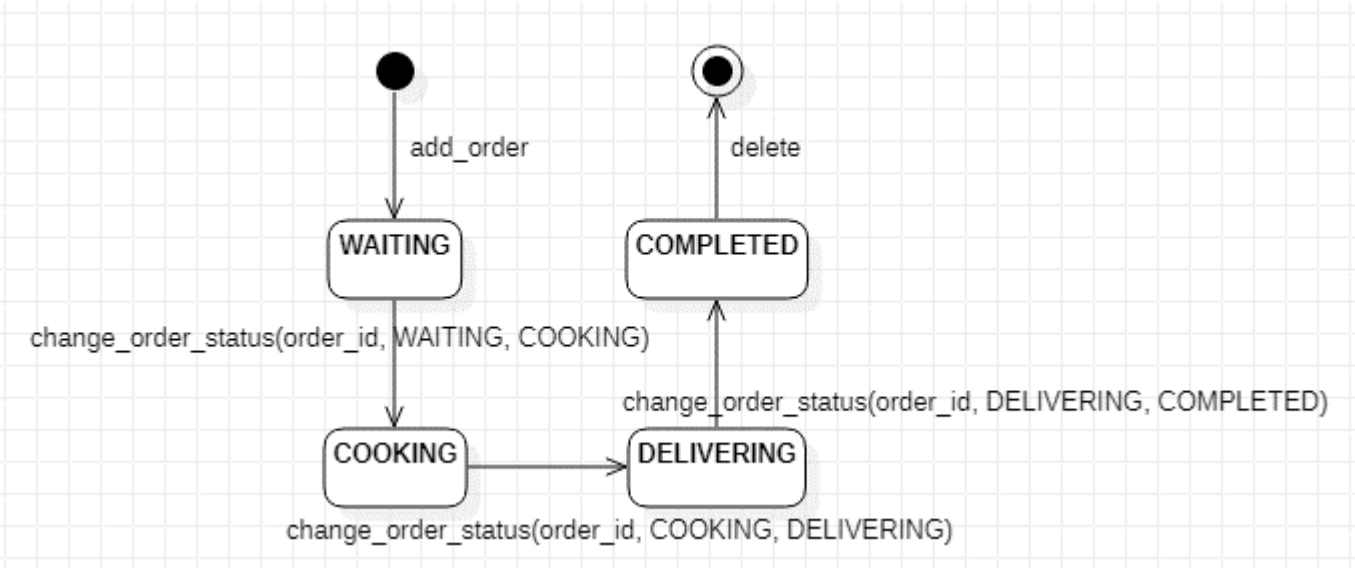
1. 직원이 'Now Ordered'를 선택하면 GET /now_ordered 가 호출되고 now_ordered.html 을 받아서 출력한다.
2. 또한 display_order_list 를 호출하여 현재 주문 목록을 받아 출력한다. 현재 주문 목록을 받을 때에는 서버에 POST /orderlist 를 통해서 받는다.
3. 현재 주문 중에서 상태를 전환할 주문들을 선택하고 cook_btn_event_listener 또는 delivery_btn_event_listener 또는 complete_btn_event_listener 를 호출하면 Now_ordered_controller 의 change_order_status(order_id, status)가 호출된다.
4. Now_ordered_controller 에서는 POST /changeOrder 를 통해서 상태를 전환할 주문을 서버에 전달한다.
5. 정상적으로 주문의 상태 전환이 완료되면 true 를 return 한다.

4.8 주문 내역 확인 (Check order)



1. “Order History”를 선택하면 GET /order_history 를 호출하고 order_history.html 을 받아서 완료된 주문 내역 페이지를 출력한다.
2. 또한 display_order_history 를 통해서 현재까지 완료된 주문 목록을 받아서 출력한다.
3. 완료된 주문 목록을 받을 때에는 GET /history 를 통해서 서버에 있는 Order_history 에서 받는다.

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, 초기 상태는 WAITING이 된다.

✓ COOKING

■ 조리가 들어간 주문을 나타내는 상태이다.

■ change_order_status 메소드를 통해 COOKING 상태로 전환된다.

✓ DELIVERING

■ 배달 중인 주문을 나타내는 상태이다.

■ Order 객체는 change_order_status 메소드를 통해 DELIVERING 상태로 전환된다.

✓ COMPLETED

■ 배달이 완료되어 완료된 주문을 나타내는 상태이다.

■ Order 객체는 change_order_status 메소드를 통해 COMPLETED 상태로 전환된다.

■ 이후 Order_history 객체에 저장되었다가 해당 주문 이후 10개의 기록이 더 발생되면 삭제된다.