<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. |
| 2 | 0 |  | 릴리즈 |
| 1 |  | 오타 수정 |
| 3 | 0 |  | [피드백 반영] 서브시스템 설계 및 클래스도 수정 |
| 4 | 0 |  | 서브시스템 설계 및 클래스도 수정 |

1 개 요

1.1 목 적

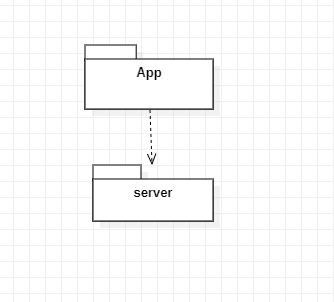
본 문서는 미스터 대박 디너 서비스(Mr.Great)의 산출물을 기술한다.

* 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.

- App Client

- Sever

**App**

- 서버와 통신하며 사용자에게 정보를 보여주는 역할을 한다.

- node.js를 이용하여 작성된다.

**Server**

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

- Node.js로 작성된다.

3 서브시스템 세부 설계

**3.1. “App" 서브시스템 세부 설계**

App은 MainPage를 호출하고 MainPage에서 기능에 해당하는 modal을 띄운다. 각 modal은 사용자에게 메뉴 화면과 주문 화면을 보여준다. 또한 user\_action 클래스를 통해 서버와 통신하며 사용자의 주문을 수행한다.

App 시스템의 주요 함수는 다음과 같다.

✓checkhandler(event)

■ 로그인한 ID의 토큰을 판별하여 사용자가 고객인지 직원인지 판별한다.

✓ onEmailHandler()

■ 입력받은 email정보를 저장한다.

✓ onPasswordHandler()

■ 입력받은 pw 정보를 저장한다.

✓ onAddressHandler()

■ 입력받은 주소 정보를 저장한다.

✓ onAmountHandler()

■ 입력받은 수량 정보를 저장한다.

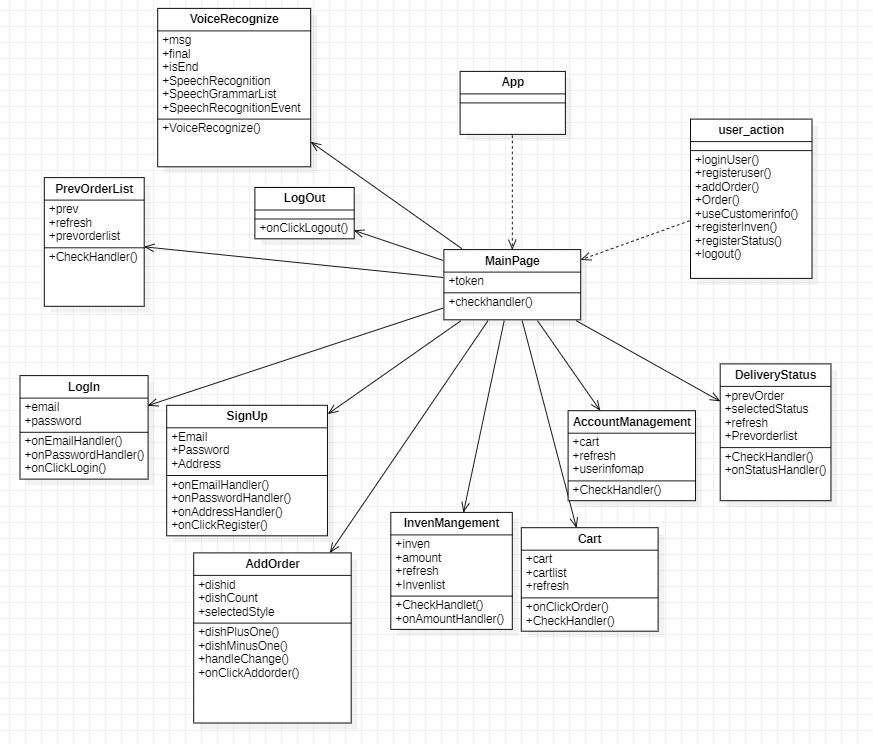
✓ onStatusHandler()

■ 입력받은 주문 상태 정보를 저장한다.

✓ onClick()

■ 버튼을 눌렀을때 해당하는 함수를 호출한다.

- “App” 서브시스템 메인 클래스도



**MainPage**

* 사용자의 UI를 보여주는 클래스이다.
* token의 role을 분석하여 사용자가 고객인지 직원인지 구분하여 기능을 표시한다.
* 각 기능에 해당하는 modal을 띄운다

**Login**

* 로그인을 수행하는 클래스이다.

**Sign\_up**

* 회원가입을 수행하는 클래스이다.

**AddOrder**

* 해당메뉴를 주문하는 기능을 수행한다.
* 주문하기를 수행해서 고객이 주문하려는 주문을 장바구니에 담을 수 있다.

**LogOut**

* 로그아웃을 수행하는 클래스이다.

**Cart**

* 장바구니 기능을 수행하는 클래스이다.
* 장바구니에서 결제하는 기능을 포함한다.

**InvenManagement**

* 재고를 관리하는 클래스이다.
* 재고의 수량을 변경할 수 있다.

**AccountManagement**

* 고객정보를 관리하는 클래스이다.
* 고객정보를 변경할 수 있다.

**DeliveryStatus**

* 결제된 주문의 상태를 관리하는 클래스이다.
* 주문의 상태를 변경할 수 있다.

**PrevOrderList**

* 과거에 주문한 내역을 관리하는 클래스이다.

**user\_action**

* app의 모달에서 서버에 요청하는 api를 구현한 클래스이다.
* 각 모달에서 사용되는 api를 저장한다.

3.2 “Server” 서브시스템 세부 설계

“Server" 는 사용자에게 메뉴 관련 정보를 제공하고 주문 정보를 저장한다. 또한, App에서 받은 요청을 처리한다.

“Server" 시스템의 주요 함수는 다음과 같다.

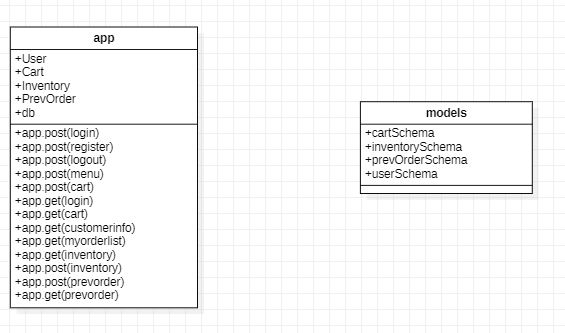
✓ app.post()

■ 데이터베이스에 정보를 전달하는 함수다.

✓ app.get()

■ 데이터베이스에서 정보를 전달받는 함수다.

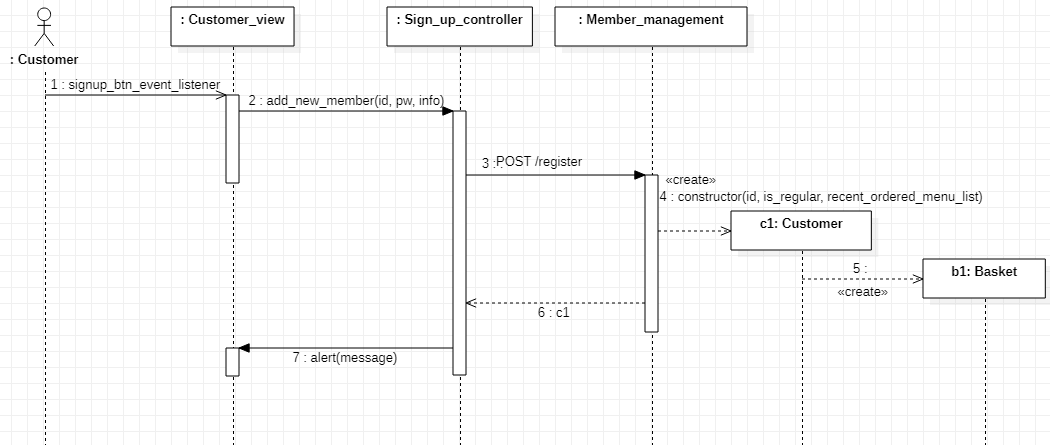
- “Server” 서브시스템 메인 클래스도

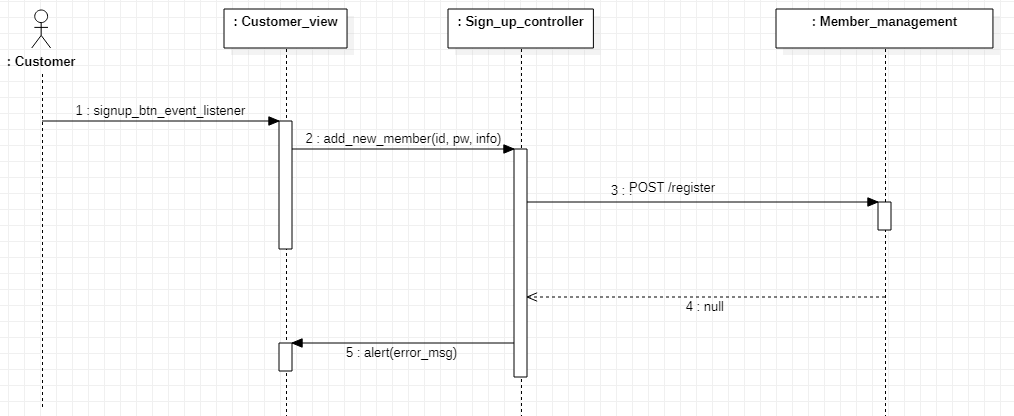


--------------------교류도는 수정안함-------------------------------

4 교류도

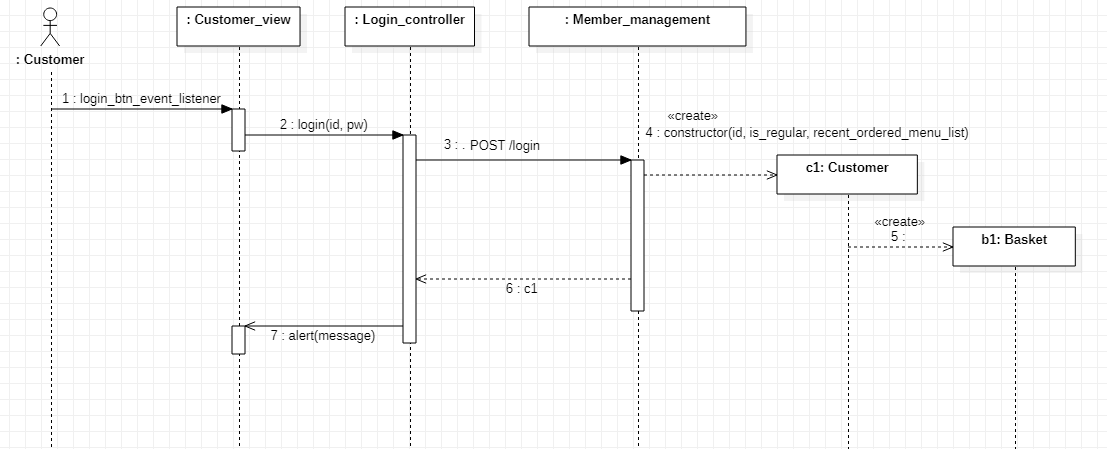
4.1 회원가입 (Sign Up)

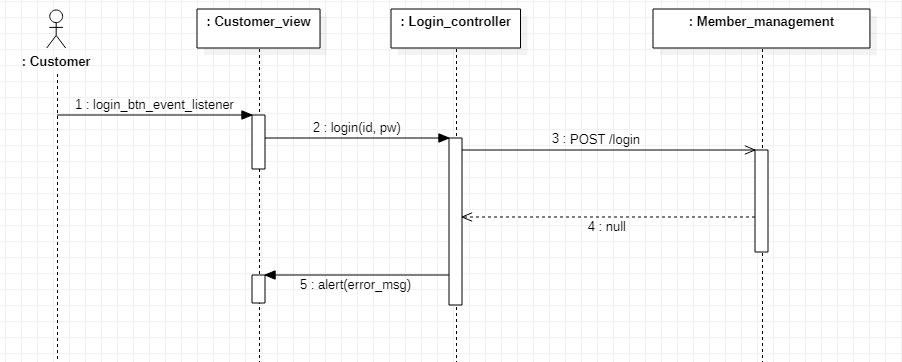




1. 고객이 ‘회원가입’을 선택하면 sign\_up\_btn\_event\_listener()가 호출된다.
2. Customer\_view에서는 Sign\_up\_controller의 add\_new\_member(id, pw, info)를 통해서 입력한 정보를 전송한다.
3. Sign\_up\_controller에서는 POST /register를 통해서 Member\_management에 회원가입 요청을 한다.
4. 회원가입에 성공하면 Member\_management에서는 고객 객체를 생성하고 그 객체를 return한다. Sign\_up\_controller에서는 그 객체를 확인하고 회원가입 완료 메시지를 Customer view에 보낸다.
5. 만약 이미 있는 아이디를 입력하여 회원가입을 실패하면 null을 return한다. Sign\_up\_controller에서는 null을 확인하고 회원가입 실패 메시지를 띄운다.

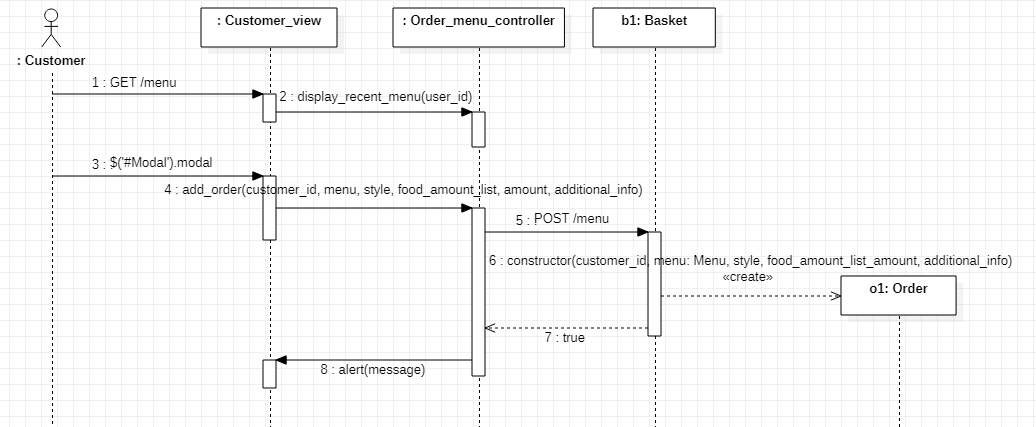
4.2 로그인 (Login)





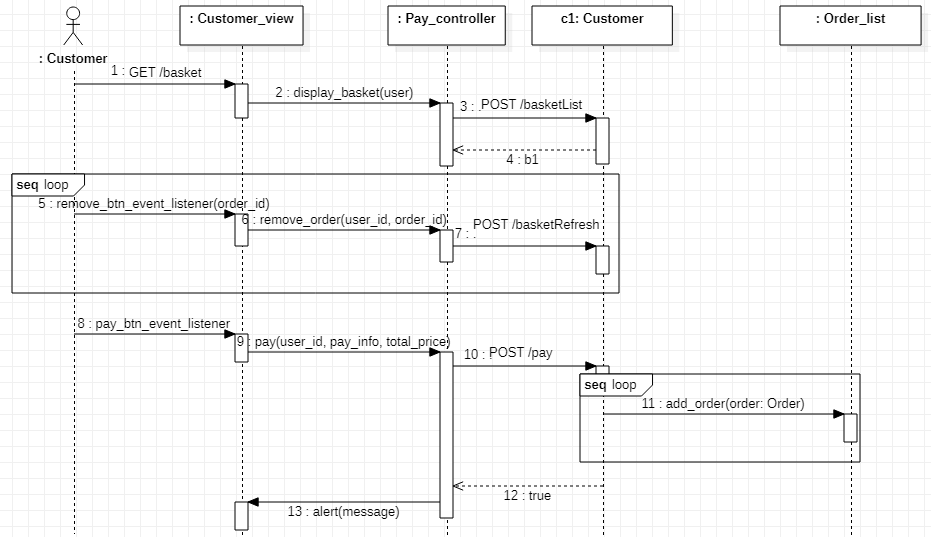
1. 고객이 아이디, 비밀번호를 입력하고 “로그인” 버튼을 누르면 login\_btn\_event\_listener()가 호출되고 Login\_controller의 login(id, pw)가 호출된다.
2. POST /login을 통해서 id, pw를 Member\_management에 보내면 일치하는 id, pw가 있는지 확인한다.
3. 만약 일치하는 id, pw가 있다면 active customer에 해당 고객을 추가하고 고객 객체를 return한다.
4. Login\_controller에서 그 객체를 받으면 로그인 성공 메시지를 띄운다.
5. 만약 일치하는 id, pw가 없다면 null을 return하고 Login\_controller에서는 로그인 실패 메시지를 띄운다.

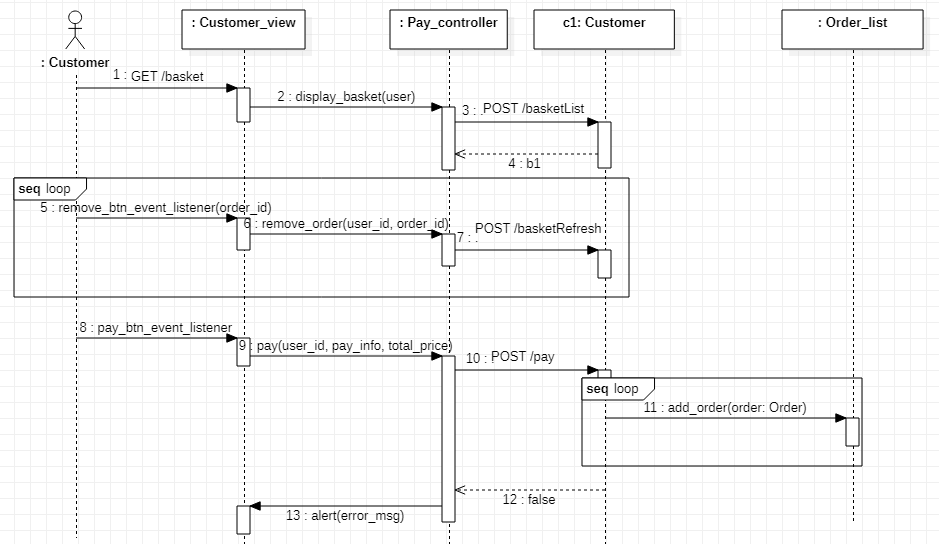
**4.3 주문 (Order menu)**



1. “매뉴”를 선택하면 GET /menu가 호출되고 menu.html을 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).modal을 통해서 매뉴의 상세 옵션이 보이고 “장바구니에 담기”를 누르면 Order\_menu\_controller에 add\_order()가 호출되어 현재 입력한 옵션에 맞는 주문을 전달한다.
3. Order\_menu\_controller에서는 POST /menu를 통해서 고객의 Basket객체에 주문 객체 정보를 담는다.
4. Basket에서는 true를 return하고 Order\_menu\_controller에서는 주문 성공 메시지를 띄운다.

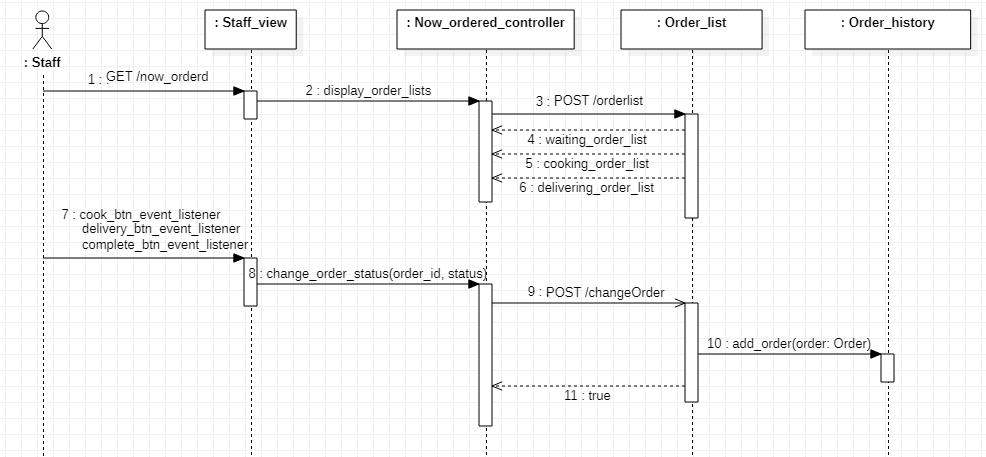
**4.4 결제 (Pay)**





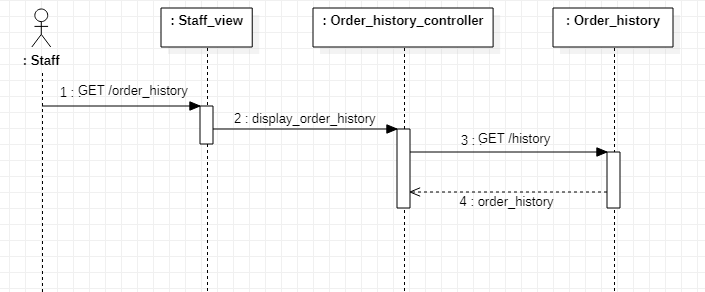
1. 고객이 “장바구니” 버튼을 누르면 GET /basket이 호출되고 basket.html을 받아 장바구니 화면을 출력한다.
2. 또한 display\_basket(user)를 Pay\_controller에 호출하여 고객의 장바구니를 화면에 나열한다. Pay\_controller에서 고객의 장바구니를 얻기 위해서는 POST /basketList를 통해서 서버에 있는 Customer객체의 장바구니 목록을 받는다.
3. 삭제할 주문이 있는 경우 삭제할 주문의 “제외하기” 버튼을 누르면 remove\_btn\_event\_listener(order\_id)를 통해서 Pay\_controller에 유저 id와 주문 id를 넘겨준다.
4. Pay\_controller에서는 POST /basketRefresh를 통해서 선택된 주문을 서버에서 제외한다.
5. “결제” 버튼을 누르면 pay\_btn\_event\_listener를 호출하고 Pay\_controller의 pay()를 실행한다.
6. Pay\_controller에서는 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\_controller에서는 결제 성공 메시지를 띄운다.
9. 만약 결제를 실패하면 false를 return하고 Pay\_controller에서는 결제 실패 메시지를 띄운다.

4.5 주문 처리 (Process Order)



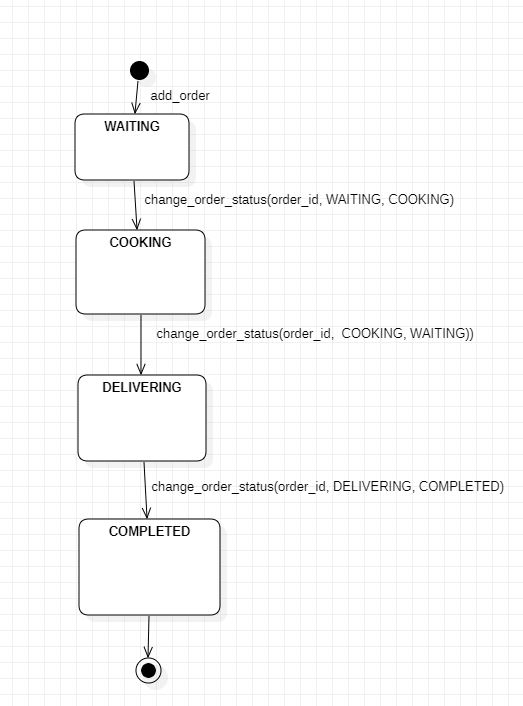
1. 직원이 ‘Now Ordered’를 선택하면 GET /now\_ordered가 호출되고 now\_ordered.html을 받아서 출력한다.
2. 또한 display\_order\_list를 호출하여 현재 주문 목록을 받아 출력한다. 현재 주문 목록을 받을 때에는 서버에 POST /orderlist를 통해서 받는다.
3. 현재 주문 중에서 상태를 전환할 주문들을 선택하고 cook\_btn\_event\_listener 또는 delivery\_btn\_event\_listener 또는 complete\_btn\_event\_listener를 호출하면 Now\_ordered\_controller의 change\_order\_status(order\_id, status)가 호출된다.
4. Now\_ordered\_controller에서는 POST /changeOrder를 통해서 상태를 전환할 주문을 서버에 전달한다.
5. 정상적으로 주문의 상태 전환이 완료되면 true를 return한다.

4.8 주문 내역 확인 (Check order)



1. “Order History”를 선택하면 GET /order\_history를 호출하고 order\_history.html을 받아서 완료된 주문 내역 페이지를 출력한다.
2. 또한 display\_order\_history를 통해서 현재까지 완료된 주문 목록을 받아서 출력한다.
3. 완료된 주문 목록을 받을 때에는GET /history를 통해서 서버에 있는 Order\_history에서 받는다.

.5 Order 객체 상태도



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

✓ WAITIN.

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

✓ COOKING

■조리중인 상태를 나타낸다..

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

✓ DELIVERING

■ 배달 중인상태를 나타낸다. .

■ change\_order\_status 메소드를 통 COOKING에서해 DELIVERING 상태로 전환된다.

✓ COMPLETED

■ 배달이 완료된상태를 나타낸다. .

■ change\_order\_status 메소드를 통 DELIVERING에서 COMPLETED 상태로 전환된다.