URL : http://sonystore.shop

Sony Store

박재한 박진수 이승현

목차

CONTENT

소개

SONY

온라인 쇼핑몰 구현

- "https://store.sony.co.kr/"의 웹사이트 UI를 클론 코딩하여 SpringBoot 기반의 온라인 쇼핑몰을 제작하는 프로젝트를 진행하였습니다.
- 회원관리, 상품 카테고리 / 검색, 주문 등의 쇼핑몰 기본 기능을 구현하였습니다.
- 관리자가 매출 데이터와 사용자 활동을 한눈에 파악할 수 있는 대시보드 페이지를 React로 구현하였습니다.

웹사이트 선정 기준

- 직관적인 UI/UX와 깔끔한 디자인, 다양한 상품 카테고리와 결제 시스템 등을 갖추었다 판단하였습니다.
- 실무에 가까운 웹 어플리케이션 개발 능력을 키우고자 하였습니다.

GIT

박진수

- 회원가입 / 로그인 / 로그아웃
- 아이디/비밀번호 찾기
- 회원정보 수정/탈퇴
- 대시보드 가입자 수 그래프

jinsu4205@gmail.com

https://jinsu1.github.io/jinsubaji/

이승현

- 장바구니
- 결제
- 주문 조회
- 대시보드 총 매출 그래프

hyeon970315@gmail.com

https://iamsh.tistory.com

박재한

* 각자 구현한 UI (HTML. JS)를 대표로 업로드 하여 12,627라인이 포함되었습니다

- 제품 목록
- 제품 상세
- 대시보드 인기 상품 순위 그래프

mitshowme@naver.com

https://parkjaehan.pages.dev

PART 1

프로젝트 개요

1-1 개발환경

Front-end

HTML5

HTML5, SCSS

Javascript

React

Back-end

JAVA

Spring Framework

MySQL

MariaDB

MyBatis

Linux

Tool

Visual Studio Code

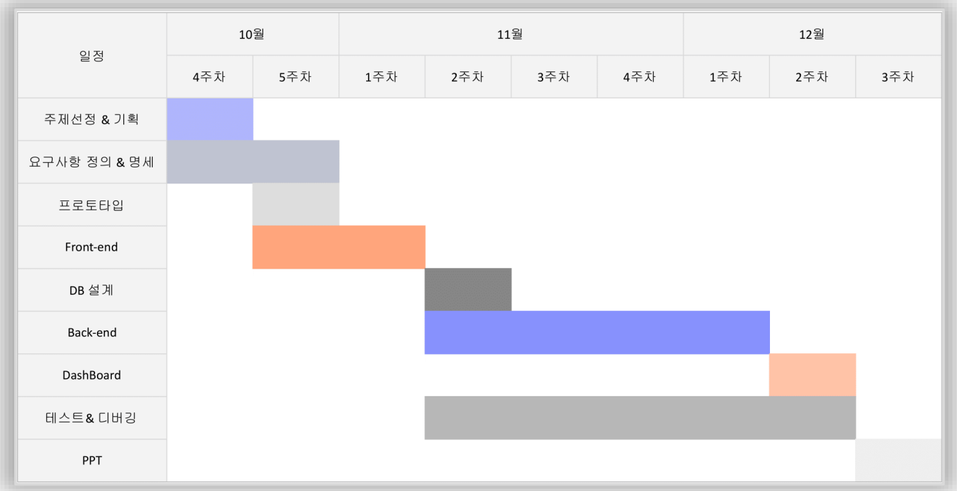
MySQL Workbench

HeidiSQL

Management

GitHub

1-2 Gantt Chart & WBS



SONY팀 퍼블리싱

Enter Project Range

Start Date	End Date
24/10/24	24/11/10

Zoom (enter 1 for Daily, 7 for Weekly)---> 1

1depth	2depth	3depth	담당자	시작일	종료일	기간	진척율 (%)	상태
제품	메인 카테고리	서브 카테고리	박재한	2024-10-25	2024-10-28	2	100 %	완료
제품 상세			이승현	2024-10-28	2024-10-30	3	100 %	완료
회원	로그인		박진수	2024-10-31	2024-10-31	1	100 %	완료
	회원 정보 찾기		박진수	2024-10-31	2024-10-31	1	100 %	완료
	회원가입		박진수	2024-10-31	2024-11-01	2	100 %	완료
	회원가입 동의		박진수	2024-11-01	2024-11-01	1	100 %	완료
	회원가입 단계		박진수	2024-11-04	2024-11-04	1	100 %	완료
마이페이지	회원정보 수정		박진수	2024-11-04	2024-11-04	1	100 %	완료
	회원탈퇴		박진수	2024-11-05	2024-11-05	1	100 %	완료
			박재한	2024-11-09	2024-11-10	0	100 %	완료
	주문/배송 조회		이승현	2024-11-07	2024-11-08	2	100 %	완료
장바구니			이승현	2024-10-31	2024-11-01	2	100 %	완료
주문	결제		이승현	2024-11-04	2024-11-05	2	100 %	완료
					2024-11-06	1	100 %	완료
					2024-11-09	2	100 %	완료
					2024-11-09	0	100 %	완료
					2024-11-01	2	100 %	완료
					2024-11-09	0	100 %	완료
					2024-11-10	0	100 %	완료
					2024-11-10	0	100 %	완료
					2024-11-04	1	100 %	완료
					2024-11-09	0	100 %	완료
					2024-11-06	1	100 %	완료
					2024-11-07	1	100 %	완료
					2024-11-08	1	100 %	완료
					2024-10-28	2	100 %	완료

SONY팀 백엔드

Enter Project Range

Start Date	End Date
24/11/20	24/12/10

Zoom (enter 1 for Daily, 7 for Weekly)---> 1

1depth	2depth	3depth	담당자	시작일	종료일	기간	진척율 (%)	상태
제품	메인 카테고리	서브 카테고리	박재한	2024-11-20	2024-11-30	8	100 %	완료
제품 상세			박재한	2024-12-01	2024-12-10	7	100 %	완료
회원	로그인		박진수	2024-11-22	2024-11-26	3	100 %	완료
	회원 정보 찾기		박진수	2024-12-01	2024-12-02	1	100 %	완료
	회원가입		박진수	2024-11-28	2024-12-08	7	100 %	완료
	회원가입 동의		박진수	2024-11-27	2024-11-28	2	100 %	완료
	회원가입 단계		박진수	2024-11-25	2024-11-25	1	100 %	완료
마이페이지	회원정보 수정		박진수	2024-11-29	2024-12-02	2	100 %	완료
	회원탈퇴		박진수	2024-12-02	2024-12-04	3	100 %	완료
			이승현	2024-12-06	2024-12-06	1	100 %	완료
	주문/배송 조회		이승현	2024-12-04	2024-12-06	3	100 %	완료
장바구니			이승현	2024-11-22	2024-11-28	5	100 %	완료
주문	결제		이승현	2024-11-28	2024-12-04	5	100 %	완료
	완료		이승현	2024-12-02	2024-12-04	3	100 %	완료
검색 결과			박진수	2024-12-08	2024-12-10	2	100 %	완료
에러페이지			박진수	2024-12-10	2024-12-10	1	100 %	완료

제작 인원 : 3명

페이지 수 : 27page

제작 기간 : 2024.10.24-12.22 (60일)

1-3 요구사항 정의


SONY
스토어 추천 제품
추천 제품
선물 제안
교육 할인
제품
카메라
비디오 카메라
렌즈
오디오
액세서리
제품상세
기획전
멤버십
등급&혜택 안내
고객서비스
구매 상담
직영점 안내
동영상 강좌
검색 결과
회원
로그인
회원가입
회원가입 동의
회원가입 단계
회원 정보 찾기
마이페이지
회원정보 수정
회원 탈퇴
주문 배송 조회
주문
결제
완료
장바구니
하단 메뉴
이용약관
사이트맵
개인정보처리방침
소비자 피해 보상보험

1-4 요구사항 명세서

1Depth	2Depth	3Depth	기능내용	
제품	카메라	전체보기		카메라 전체를 보여줌(카메라 탭을 클릭했을때 기본으로 실행)
		렌즈교환식 카메라	전체보기	렌즈 교환식카메라 전체를 보여줌, 버튼 클릭시 파란색으로 Active(렌즈 교환식카메라 버튼을 클릭했을때 기본으로 실행)
			풀프레임	풀프레임 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			APS-C	APS-C 제품만 보여줌, 버튼 클릭시 파란색으로 Active
		컴팩트 카메라		컴팩트 카메라 제품만 보여줌
	비디오카메라	전체보기		비디오 카메라 전체를 보여줌(비디오 카메라 탭을 클릭했을때 기본으로 실행)
		시네마 라인 카메라		시네마 라인 카메라 샹품만 보여줌
		캠코더	전체보기	캠코더 전체를 보여줌, 버튼 클릭시 파란색으로 Active(캠코더 버튼을 클릭했을때 기본으로 실행)
			4K 핸디캠	4K 핸디캠 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			4K 핸드헬드	4K 핸드헬드 제품만 보여줌, 버튼 클릭시 파란색으로 Active
	렌즈	전체보기		렌즈 전체를 보여줌(렌즈 탭을 클릭했을때 기본으로 실행)
		풀프레임 렌즈	전체보기	풀프레임 렌즈 전체를 보여줌, 버튼 클릭시 파란색으로 Active(풀프레임 렌즈 버튼을 클릭했을때 기본으로 실행)
			표준	표준 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			망원	망원 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			광각	광각 제품만 보여줌, 버튼 클릭시 파란색으로 Active
		APS-C 렌즈	전체보기	APS-C 렌즈 전체를 보여줌, 버튼 클릭시 파란색으로 Active(APS-C 렌즈 버튼을 클릭했을때 기본으로 실행)
			표준	표준 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			망원	망원 제품만 보여줌, 버튼 클릭시 파란색으로 Active
			광각	광각 제품만 보여줌, 버튼 클릭시 파란색으로 Active
제품 상세				이미지 슬라이더로 제품의 이미지 확인 제품의 이름, 가격, 색상종류 확인 드롭박스로 제품선택 기능 구현 선택한 제품에 따른 총 금액, 수량 확인 기능 "장바구니" 버튼, 기능 구현 "바로 구매하기" 버튼 클릭 시, "결제" 페이지로 이동 추천 제품 슬라이더로 보여주고, 제품 클릭 시, 제품의 "제품 상세" 페이지로 이동 "소니 알파 유니버스", "소니 블로그", "매뉴얼 다운로드" 클릭 시, 각 페이지로 이동 "제품 개요", "제품 상세", "배송/환불규정" 탭 클릭으로 각각의 내용 확인
검색 결과				검색 결과 키워드가 포함된 제품 목록 표시

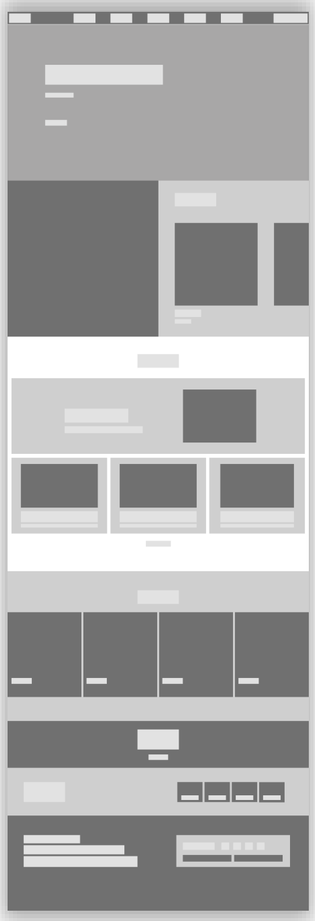
1-4 요구사항 명세서

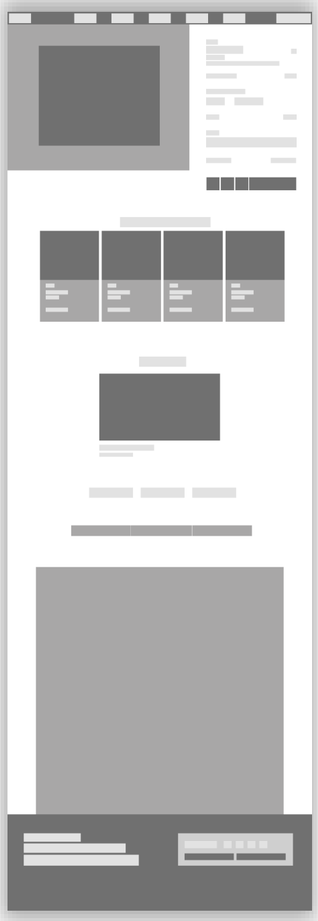
1Depth	2Depth	3Depth	기능내용	
회원	로그인		회원 로그인	이메일 아이디, 비밀번호 입력 기능 (각 항목마다 정규표현식 검사) "로그인" 버튼 클릭 시, 메인 페이지로 이동 "제품 상세" 페이지에서 "바로 구매하기"로 연결 됐을 시, 이전 페이지(제품 상세 페이지)로 이동 이메일 아이디 저장 기능 "아이디·비밀번호 찾기" 클릭 시, "회원 정보 찾기" 페이지로 이동 "회원가입" 클릭 시, "회원가입" 페이지로 이동
			로그아웃	로그아웃 (메인 페이지로 이동)
	회원 정보 찾기		아이디 찾기	이름, 휴대폰 번호 입력 기능 (각 항목마다 정규표현식 검사)
			비밀번호 찾기	이메일 아이디, 휴대폰 번호 입력 기능 (각 항목마다 정규표현식 검사)
	회원가입			"소니스토어 회원 가입" 버튼 클릭 시, "회원가입 동의" 페이지로 이동
	회원가입 동의			회원가입 약관 표시. 각 항목 체크박스 기능 필수 약관 모두 동의 시 "회원정보입력" 페이지로 이동, 비동의 시 이용약관 동의 요구 알림창 띄우기
	회원가입 단계			이메일 아이디, 비밀번호, 비밀번호 확인, 이름, 생년월일, 성별, 휴대폰 번호 입력 기능 (각 항목마다 정규표현식 검사) 이메일로 인증번호 확인 기능 "가입 완료" 버튼 클릭 시, 회원가입 완료 알림창 띄우기. "확인" 버튼 클릭 시, 메인 페이지로 이동
마이페이지	회원정보 수정			이름, 휴대폰, 이메일, 생년월일, 성별, 비밀번호, 주소, 이벤트 알림 수신여부 변경
	회원탈퇴			탈퇴사유 선택하고, 비밀번호 일치 시, 회원탈퇴 진행
				회원등급 표시. 클릭 시, "등급 & 혜택 안내" 페이지로 이동
				진행 상태와 건수 표시. "구매 내역 조회" 버튼 클릭 시, "주문/배송 조회" 페이지로 이동
	주문 조회			진행 중인 주문 표시 선택한 날짜의 기간동안의 주문 내역 조회 기능
장바구니				장바구니에 담긴 제품 내역 확인, 수량변경 기능 선택 제품 삭제 기능 결제 예정 금액, 총 수량 확인 "쇼핑 계속 하기" 버튼 클릭 시, 메인 페이지로 이동 "구매하기" 버튼 클릭 시, "주문·결제" 페이지로 이동
주문·결제	결제			주문할 제품 정보 표시 주문자 정보, 배송지 정보, 할인 정보, 결제 방법 입력 및 선택 결제 예정 금액 표시 결제 후 이메일 발송, "결제완료" 페이지로 이동
	완료			결제 완료 안내, 주문번호 표시 "계속 쇼핑하기" 클릭 시, 메인 페이지로 이동 "주문/배송 조회" 클릭 시, "주문 배송 조회" 페이지로 이동

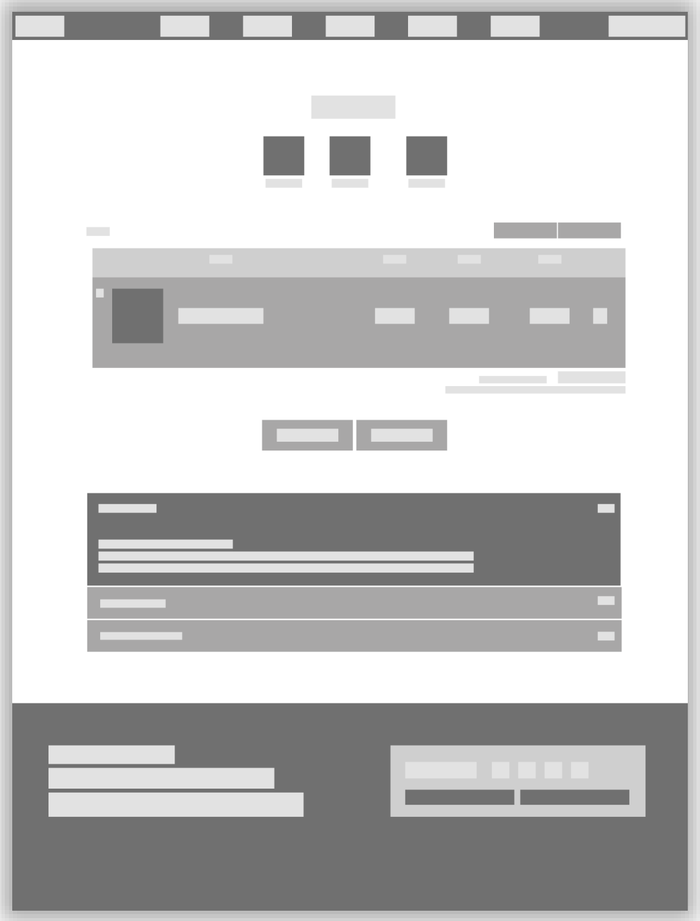
PART 2

화면구현

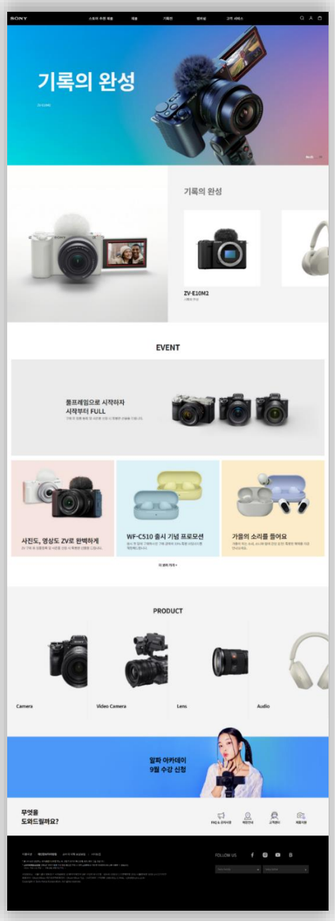
2-1 화면구현 - MockUp

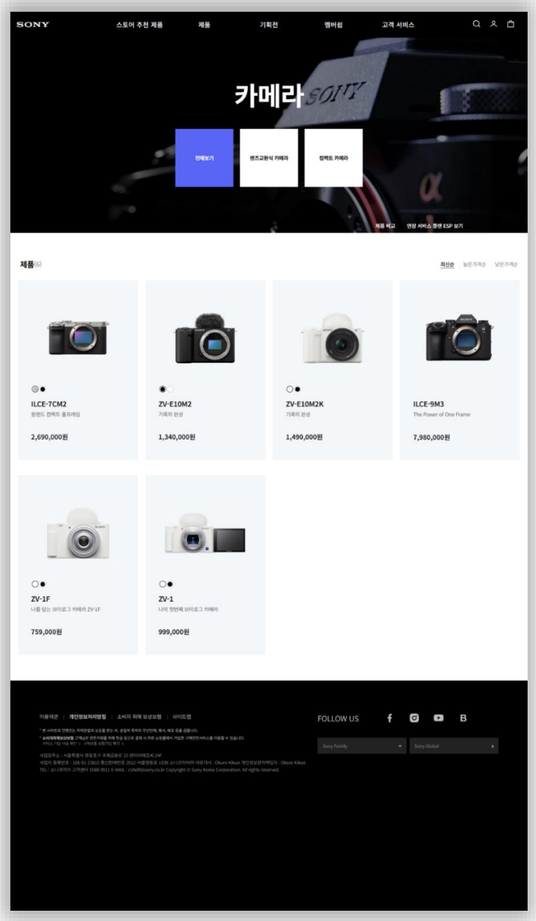


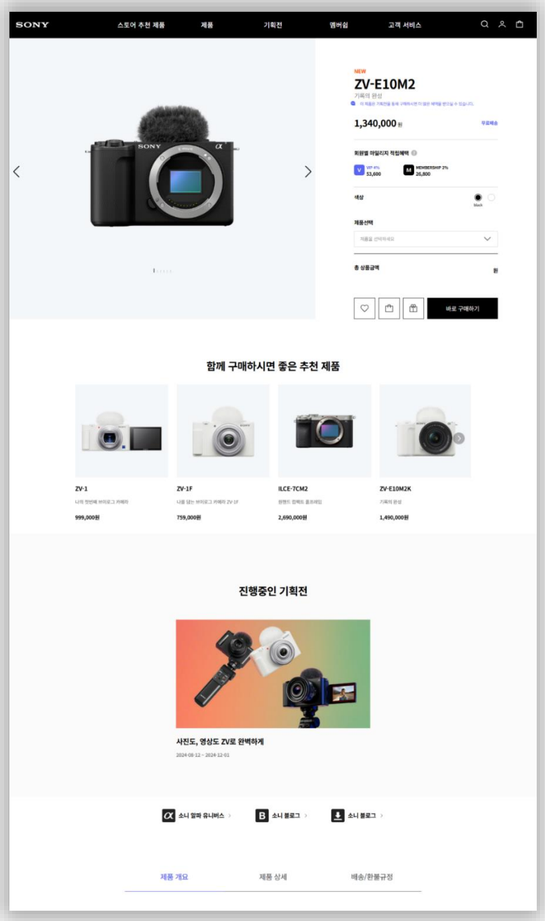


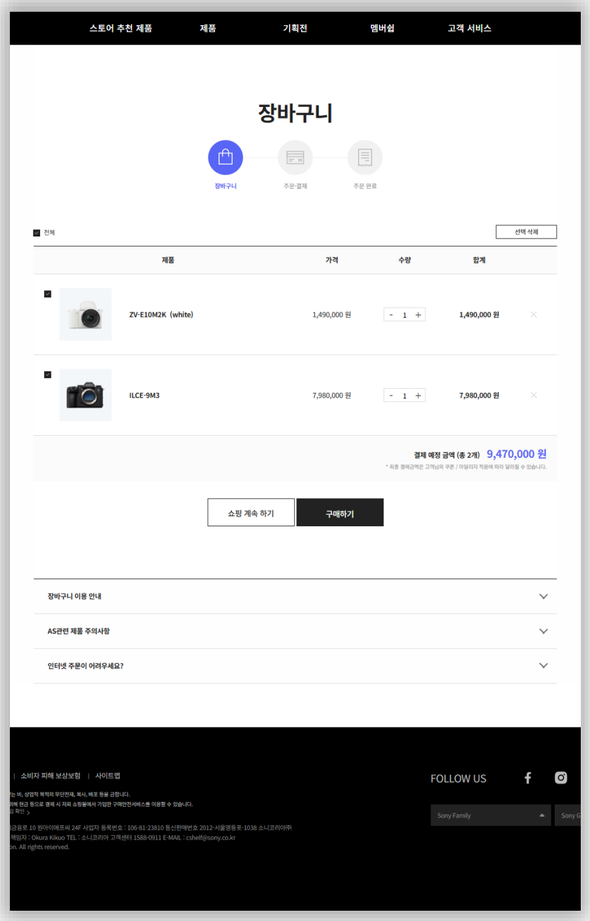


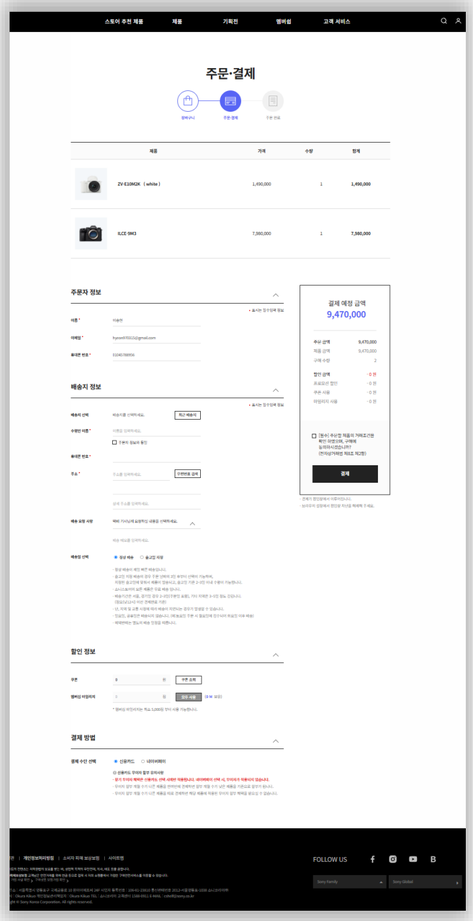
2-2 화면구현 - 사용자 인터페이스









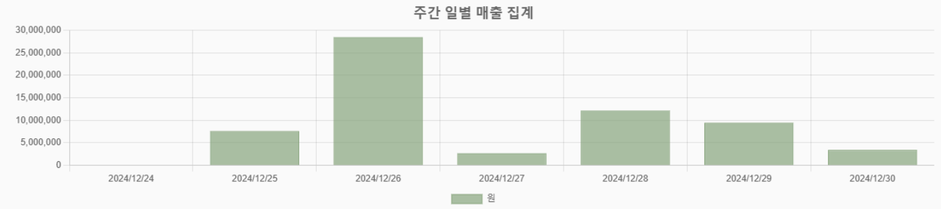


2-3 화면구현 - 대시보드

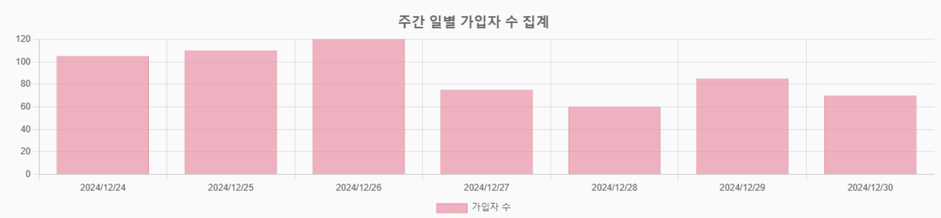
주간

DASHBOARD

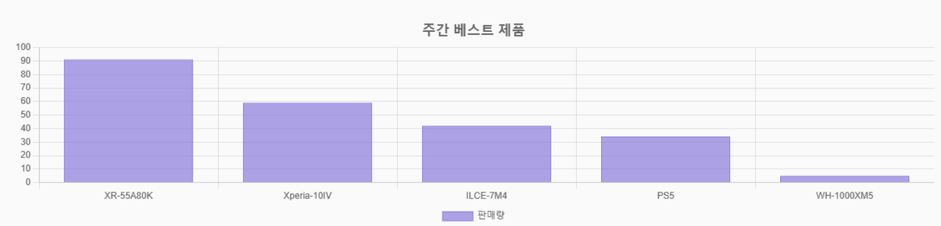
총 매출 주간



가입자 수 주간



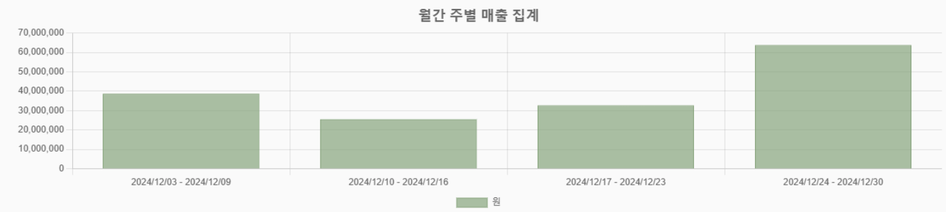
인기 제품 TOP5 주간



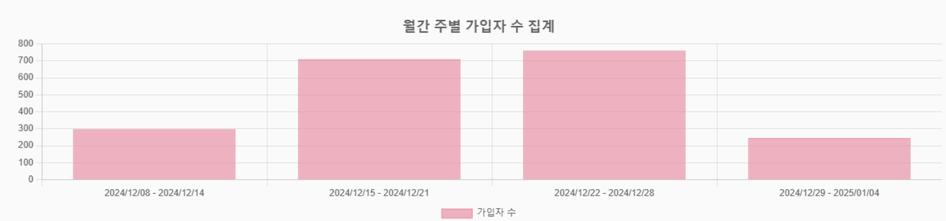
월간

DASHBOARD

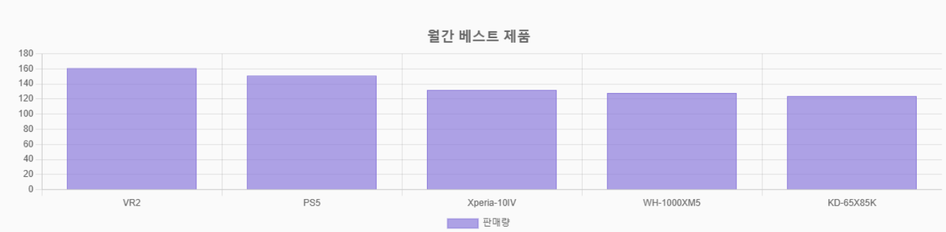
총 매출 월간



가입자 수 월간



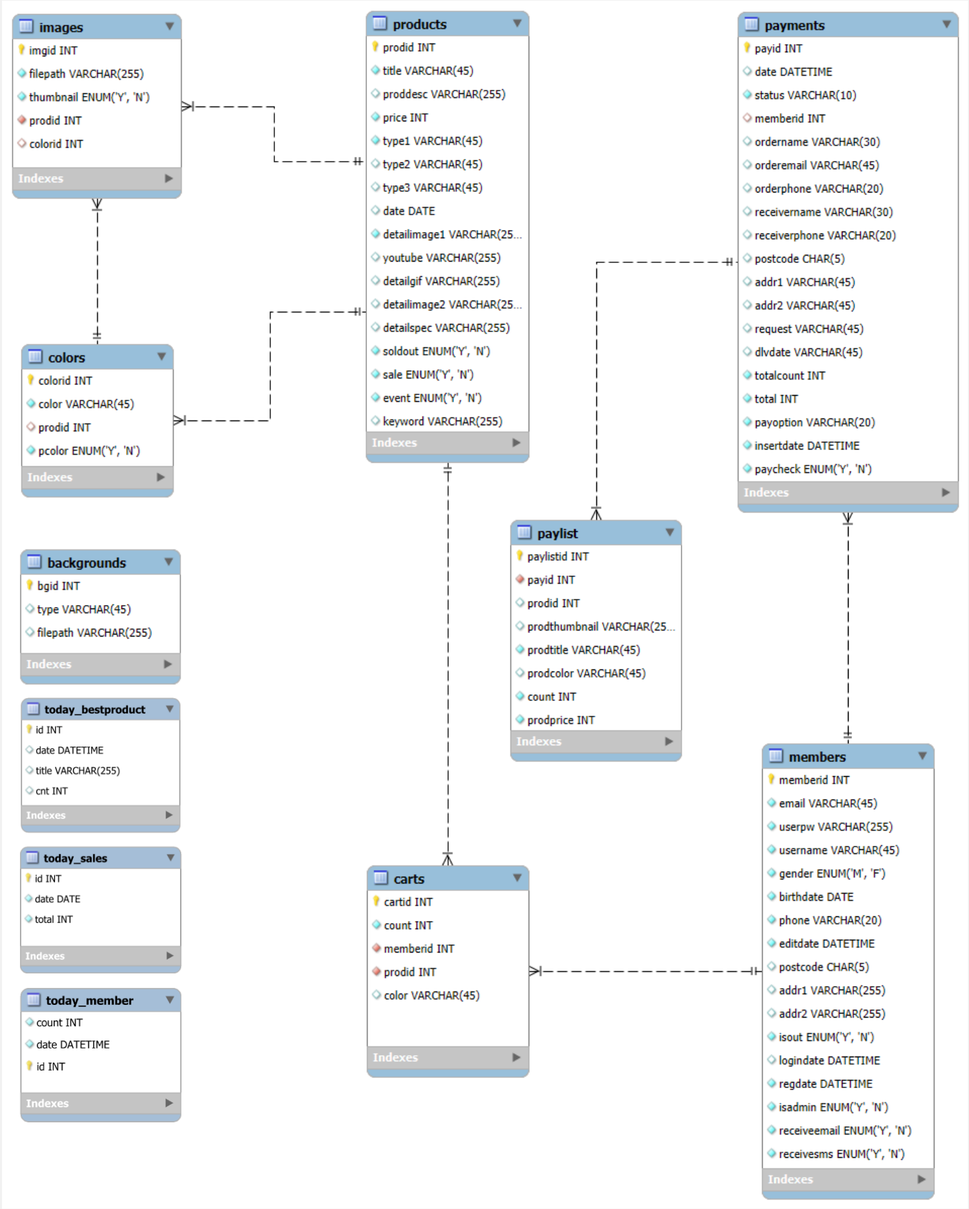
인기 제품 TOP5 월간



PART 3

데이터베이스

3-1 테이블 구성 (ERD)


images
imgid INT
filepath VARCHAR(255)
thumbnail ENUM('Y', 'N')
prodid INT
colorid INT
Indexes
products
prodid INT
title VARCHAR(45)
proddesc VARCHAR(255)
price INT
type1 VARCHAR(45)
type2 VARCHAR(45)
type3 VARCHAR(45)
date DATE
detailimage1 VARCHAR(25...
youtube VARCHAR(255)
detailgif VARCHAR(255)
detailimage2 VARCHAR(25...
detailspec VARCHAR(255)
soldout ENUM('Y', 'N')
sale ENUM('Y', 'N')
event ENUM('Y', 'N')
keyword VARCHAR(255)
Indexes
payments
payid INT
date DATETIME
status VARCHAR(10)
memberid INT
ordername VARCHAR(30)
orderemail VARCHAR(45)
orderphone VARCHAR(20)
receivername VARCHAR(30)
receiverphone VARCHAR(20)
postcode CHAR(5)
addr1 VARCHAR(45)
addr2 VARCHAR(45)
request VARCHAR(45)
dlvdate VARCHAR(45)
totalcount INT
total INT
payoption VARCHAR(20)
insertdate DATETIME
paycheck ENUM('Y', 'N')
Indexes
colors
colorid INT
color VARCHAR(45)
prodid INT
pcolor ENUM('Y', 'N')
Indexes
paylist
paylistid INT
payid INT
prodid INT
prodthumbnail VARCHAR(25...
prodtitle VARCHAR(45)
prodcolor VARCHAR(45)
count INT
prodprice INT
Indexes
backgrounds
bgid INT
type VARCHAR(45)
filepath VARCHAR(255)
Indexes
today_bestproduct
id INT
date DATETIME
title VARCHAR(255)
cnt INT
Indexes
today_sales
id INT
date DATE
total INT
Indexes
today_member
count INT
date DATETIME
id INT
Indexes
carts
cartid INT
count INT
memberid INT
prodid INT
color VARCHAR(45)
Indexes
members
memberid INT
email VARCHAR(45)
userpw VARCHAR(255)
username VARCHAR(45)
gender ENUM('M', 'F')
birthdate DATE
phone VARCHAR(20)
editdate DATETIME
postcode CHAR(5)
addr1 VARCHAR(255)
addr2 VARCHAR(255)
isout ENUM('Y', 'N')
logindate DATETIME
regdate DATETIME
isadmin ENUM('Y', 'N')
receiveemail ENUM('Y', 'N')
receivesms ENUM('Y', 'N')
Indexes

3-2 테이블 명세서

| 제품

TableName	products				
Description	제품				

No	FieldName	DataType	Null	Key	Extra	Comment
1	prodid	int	NOT NULL	PRI	auto_increment	일련번호
2	title	varchar(45)	NOT NULL			제품 이름
3	proddesc	varchar(255)	NULL			제품 설명
4	price	int	NOT NULL			제품 가격
5	type1	varchar(45)	NOT NULL			제품 카테고리1
6	type2	varchar(45)	NULL			제품 카테고리2
7	type3	varchar(45)	NULL			제품 카테고리3
8	date	date	NULL			출시일
9	detailimage1	varchar(255)	NOT NULL			제품 상세의 이미지 경로1
10	youtube	varchar(255)	NULL			제품 상세의 유튜브 경로
11	detailgif	varchar(255)	NULL			제품 상세의 gif 경로
12	detailimage2	varchar(255)	NULL			제품 상세의 이미지 경로2
13	detailspec	varchar(255)	NULL			제품 상세의 스펙
14	soldout	enum('Y','N')	NOT NULL			제품의 품절여부 (Y/N)
15	sale	enum('Y','N')	NOT NULL			제품의 할인 유무 (Y/N)
16	event	enum('Y','N')	NOT NULL			제품의 기획전 유무 (Y/N)
17	keyword	varchar(255)	NULL			검색 결과 키워드

| 색상

TableName	colors				
Description	색상				

No	FieldName	DataType	Null	Key	Extra	Comment
1	colorid	int	NOT NULL	PRI	auto_increment	일련번호
2	color	varchar(45)	NOT NULL			색상
3	prodid	int	NULL	MUL		상품의 일련번호
4	pcolor	enum('Y','N')	NOT NULL			기본색상

| 장바구니

TableName	carts				
Description	장바구니				

No	FieldName	DataType	Null	Key	Extra	Comment
1	cartid	int	NOT NULL	PRI	auto_increment	일련번호
2	count	int	NOT NULL			수량
3	memberid	int	NOT NULL	MUL		회원의 일련번호
4	prodid	int	NOT NULL	MUL		제품의 일련번호
5	color	varchar(45)	NULL			제품의 색상

| 이미지

TableName	images				
Description	제품 이미지				

No	FieldName	DataType	Null	Key	Extra	Comment
1	imgid	int	NOT NULL	PRI	auto_increment	일련번호
2	filepath	varchar(255)	NOT NULL			이미지 파일 경로
3	thumbnail	enum('Y','N')	NOT NULL			제품 대표 이미지 (Y/N)
4	prodid	int	NOT NULL	MUL		제품의 일련번호
5	colorid	int	NULL	MUL		색상의 일련번호

| 배경 이미지

TableName	backgrounds				
Description	배경이미지 테이블				

No	FieldName	DataType	Null	Key	Extra	Comment
1	bgid	int	NOT NULL	PRI	auto_increment	
2	type	varchar(45)	NULL			제품의 타입
3	filepath	varchar(255)	NULL			파일 경로

3-2 테이블 명세서

| 결제 목록

TableName	paylist				
Description	결제 제품 목록				

No	FieldName	DataType	Null	Key	Extra	Comment
1	paylistid	int	NOT NULL	PRI	auto_increment	일련번호
2	payid	int	NOT NULL	MUL		결제내역의 일련번호
3	prodid	int	NULL			제품의 일련번호
4	prodthumbnail	varchar(255)	NULL			제품의 대표 이미지 경로
5	prodtitle	varchar(45)	NOT NULL			제품의 이름
6	prodcolor	varchar(45)	NULL			구매한 제품의 색상
7	count	int	NOT NULL			제품의 수량
8	prodprice	int	NOT NULL			제품의 가격

| 결제

TableName	payments				
Description	결제				

No	FieldName	DataType	Null	Key	Extra	Comment
1	payid	int	NOT NULL	PRI	auto_increment	일련번호
2	date	datetime	NULL			결제날짜
3	status	varchar(10)	NOT NULL			처리상태
4	memberid	int	NULL	MUL		회원의 일련번호
5	ordername	varchar(30)	NULL			주문자 이름
6	orderemail	varchar(45)	NULL			
7	orderphone	varchar(20)	NULL			주문자 휴대폰번호
8	receivername	varchar(30)	NULL			수령인 이름
9	receiverphone	varchar(20)	NULL			수령인 휴대폰번호
10	postcode	char(5)	NULL			우편번호
11	addr1	varchar(45)	NULL			검색된 주소
12	addr2	varchar(45)	NULL			상세 주소
13	request	varchar(45)	NULL			배송 요청 사항
14	dlvdate	varchar(45)	NULL			배송일
15	totalcount	int	NOT NULL			전체 수량
16	total	int	NOT NULL			총 결제금액
17	payoption	varchar(20)	NOT NULL			결제방법
18	insertdate	datetime	NOT NULL			구매 희망 일시
19	paycheck	enum('Y','N')	NOT NULL			결제유무

| 회원 정보

TableName	members				
Description	회원정보				

No	FieldName	DataType	Null	Key	Extra	Comment
1	memberid	int	NOT NULL	PRI	auto_increment	일련번호
2	email	varchar(45)	NOT NULL			이메일
3	userpw	varchar(255)	NOT NULL			비밀번호
4	username	varchar(45)	NOT NULL			이름
5	gender	enum('M','F')	NOT NULL			성별 (M/F)
6	birthdate	date	NOT NULL			생년월일
7	phone	varchar(20)	NOT NULL			휴대폰번호
8	editdate	datetime	NOT NULL			변경일시
9	postcode	char(5)	NULL			우편번호
10	addr1	varchar(255)	NULL			검색된 주소
11	addr2	varchar(255)	NULL			상세 주소
12	isout	enum('Y','N')	NOT NULL			탈퇴 여부 (Y/N)
13	logindate	datetime	NULL			마지막 로그인 일시
14	regdate	datetime	NOT NULL			등록일시
15	isadmin	enum('Y','N')	NOT NULL			관리자 여부 (Y/ N)
16	receiveemail	enum('Y','N')	NOT NULL			광고성 정보 이메일 수신 여부 (Y/N)
17	receivesms	enum('Y','N')	NOT NULL			광고성 정보 문자 수신 여부 (Y/N)

| 인기제품순위

TableName	today_product				
Description	인기제품순위				

No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	일련번호
2	date	datetime	NULL			구매날짜
3	title	varchar(255)	NULL			제품명
4	cnt	int	NULL			개수

| 날짜별가입자수

TableName	today_member				
Description	날짜별 가입자 수				

No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	집계 번호
2	date	datetime	NOT NULL			가입한 날짜
3	count	int	NOT NULL			가입자 수

| 날짜별매출

TableName	today_sales				
Description	날짜별 매출				

No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	일련번호
2	date	date	NOT NULL			날짜
3	count	int	NOT NULL			매출

TEST

단위 테스트

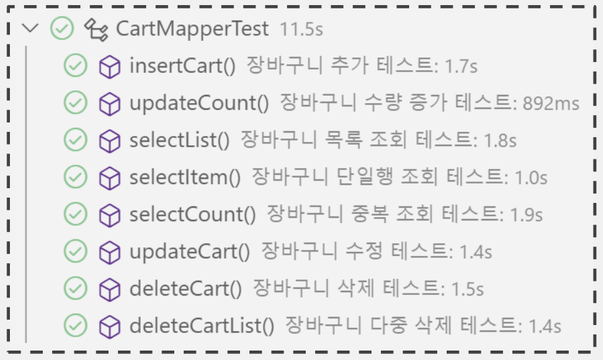
단위 테스트

Mapper 테스트 예시

```java
@Test
@DisplayName("장바구니 목록 조회 테스트")
void selectList() {
    Cart input = new Cart();
    input.setMemberid(memberid:24);

    List<Cart> output = cartMapper.selectList(input);
    for ( Cart item : output ) {
        log.debug("output: " + item.toString());
    }
}
```

Mapper 테스트 결과 예시

cartid	memberid	prodid	title	colorid	color	filepath	price	count	sum
19	24	1	ZV-E10M2K	2	black	/products/ZV-E10M2K/clr1_0.png	1490000	1	1490000
20	24	1	ZV-E10M2K	1	white	/products/ZV-E10M2K/clr0_0.png	1490000	1	1490000



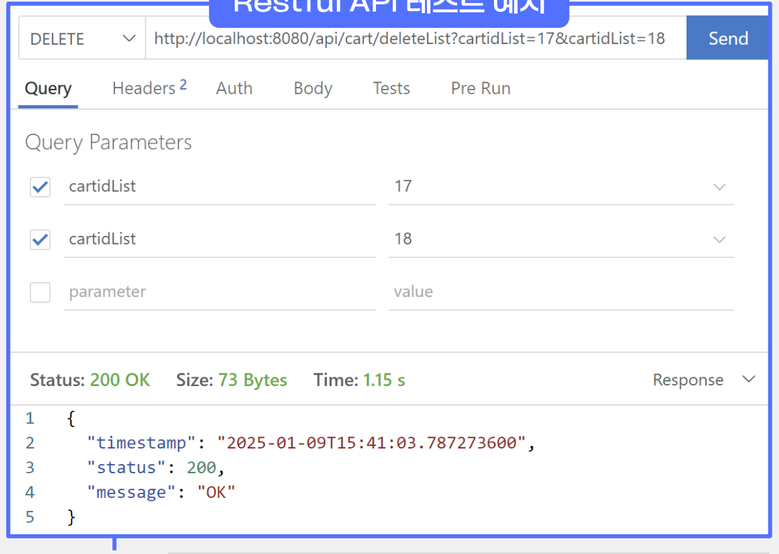
Service 테스트 예시

```java
@Test
@DisplayName("장바구니에 상품 추가 or 수량변경 테스트")
void addOrEditItem() {
    Cart input = new Cart();
    input.setCount(count:1);
    input.setMemberid(memberid:24);
    input.setProdid(prodid:1);
    input.setColor(color:"white");

    Cart output = null;

    try {
        output = cartService.addOrEditItem(input);
    } catch (Exception e) {
        log.error(msg:"Mapper 구현 에러", e);
    }

    log.debug("output: " + output);
}
```

Service 테스트 결과 예시

```
[INFO ] [CartServiceTest:56] - Started CartServiceTest in 10.819 seconds
(process running for 11.139)
[DEBUG] [selectCount:135] - ==>  Preparing: SELECT COUNT(*) FROM carts
WHERE memberid = ? AND prodid = ? AND color = ?
[DEBUG] [selectCount:135] - ==> Parameters: 24(Integer), 1(Integer), white(String)
[INFO ] [sqlonly:228] - SELECT COUNT(*) FROM carts
WHERE memberid = 24 AND prodid = 1 AND color = 'white'
[INFO ] [resultsettable:610] -
|---------|
|count(*) |
|---------|
|1        |
|---------|
[DEBUG] [selectCount:135] - <==      Total: 1
[DEBUG] [CartServiceImpl:29] - rows : 1
[DEBUG] [updateCount:135] - ==>  Preparing:
UPDATE carts SET count = count + ? WHERE memberid = ? AND prodid = ? AND color = ?
[DEBUG] updateCount:135] - ==> Parameters: 1(Integer), 24(Integer), 1(Integer), white(String)
[INFO ] [sqlonly:228] - UPDATE carts SET count = count + 1
WHERE memberid = 24 AND prodid = 1
  AND color = 'white'
[DEBUG] [updateCount:135] - <==    Updates: 1
```

Restful API 테스트 예시



VS Code Extension인
Thunder Client 를 이용한 Restful API 테스트

Restful API 테스트 결과 예시

```
[INFO ] [MyInterceptor:46] - ---------- new client connect ----------
[INFO ] [MyInterceptor:65] - [Client] 127.0.0.1, Other, Other, null, Other, null
[INFO ] [MyInterceptor:79] -
[DELETE] http://localhost:8080/api/cart/deleteList?cartidList=17&cartidList=18
[INFO ] [MyInterceptor:87] - (param) <-- cartidList = 17,18
[DEBUG] [deleteList:135] - ==>  Preparing: DELETE FROM carts WHERE cartid IN ( ? , ? )
[DEBUG] [deleteList:135] - ==> Parameters: 17(Integer), 18(Integer)
[INFO ] [sqlonly:228] - DELETE FROM carts WHERE cartid IN ( 17, 18 )
[DEBUG] [deleteList:135] - <==    Updates: 2
```

PART 4

구현기능

주요 기능

Member

회원가입 / 로그인 / 로그아웃

· 회원가입 - 이메일 아이디, 비밀번호, 비밀번호 확인, 이름, 생년월일, 휴대폰 번호를 정규표현식을 통해 유효성 검사
· 로그인 및 로그아웃 기능 구현

아이디/비밀번호 찾기

· 이름과 휴대폰 번호를 조건으로 검색을 통한 아이디 찾기
· 이메일 아이디와 휴대폰 번호를 조건으로 회원정보 검색 후 랜덤 비밀번호 부여 후 해당 이메일 아이디로 새로 설정된 비밀번호가 포함된 메일 전송

회원정보 수정 / 회원탈퇴

· 이름 변경 - 휴대폰 번호로 회원정보 업데이트
(휴대폰 번호는 중복 검사 후 회원가입 가능)
· 비밀번호 변경 - 로그인 후 세션에 저장된 회원정보를 입력받은 비밀번호와 일치하는지 확인. 유효성 검사 후 새 비밀번호로 변경
· 주소 및 광고성 이메일 수신, 문자 수신 여부를 수정 가능
· 회원 탈퇴 - 회원을 탈퇴하면 회원정보의 탈퇴 회원 여부의 값을 "Y"로 변경
→ 스케줄러를 통해 정해진 시간마다 해당 값이 "Y"인 컬럼의 회원 정보 delete 실행

Product

제품 목록

· 제품의 종류 별로 DB에서 Select한 결과를 화면에 표시
· 제품의 type 별로 분류되는 버튼 및 탭 메뉴 구현
· 최신순, 높은 가격순, 낮은 가격순 버튼 클릭 시 DB에 저장된 값으로 정렬
· 색상이 없는 제품을 제외한 모든 제품에 DB에 저장된 해당 색상 버튼 출력과 마우스 Over 시 해당 색상의 사진으로 변경
· 해당 제품을 클릭 시 제품 상세 화면으로 이동

제품 상세

· 제품의 이미지를 DB에서 Select해 swiper 및 pagination 으로 슬라이드 구현
· DB에 저장된 색상값으로 색상이 없는 제품을 제외한 모든 제품에 해당 색상 버튼 출력과 마우스 클릭 시 해당 색상의 사진으로 변경
· 드롭박스로 해당 제품의 색상과 제품 선택 기능 구현· 제품의 가격을 개수에 맞게 계산
· 제품의 상세 이미지와 해당 제품의 YOUTUBE 영상 출력

검색

· header의 아이콘 메뉴를 통해 접근 가능하며 키워드를 입력하면 키워드가 포함된 문자열을 검색 후 검색결과 페이지에 제품목록을 렌더링

Order

장바구니

· 제품상세페이지에서 제품의 정보를 장바구니 테이블에 삽입
· 삽입된 장바구니 테이블의 데이터들을 제품, 색상, 이미지 테이블과 조인하여 조회
· 수량변경 버튼 클릭 시, 장바구니 테이블의 수량값 업데이트
· 선택한 제품을 장바구니 테이블에서 단일삭제 / 다중삭제

주문·결제

· '장바구니 데이터' 또는 제품 상세 페이지의 '제품 데이터'를 결제 테이블에 삽입 (주문정보 제외)
· 회원의 결제테이블에서 최근 배송지 5개까지 조회
· 다음 주소찾기 API 사용하여 주소 입력
· 주문·배송 정보 입력폼 유효성 검사
· 결제 버튼 클릭 시, 결제 페이지에 입력된 주문·배송 정보로 결제 테이블 업데이트 / 결제 완료 메일 발송
· 결제 완료 시, 장바구니 테이블의 데이터 삭제 (장바구니에서 구매했을 시)
· 결제하지 않고 페이지 이동 시, 스케줄러로 매일 오전 4시에 체크하여 결제 테이블의 paycheck 컬럼이 "N" 인 데이터를 조회하여 삭제

주문 조회

· 결제 테이블의 status 컬럼이 "결제완료"인 데이터를 카운트하여 뷰에 전달
· 사용자가 입력한 기간동안의 주문 내역을 결제 테이블에서 조회하여 뷰에 데이터 전달

Dashboard

대시보드

· 매출 데이터와 사용자 활동을 파악하는 대시보드 기능 구현 (주간 /월간)
· 총 매출 그래프 / 신규 회원 추이 / 인기 상품 순위

4-1 클래스 다이어그램

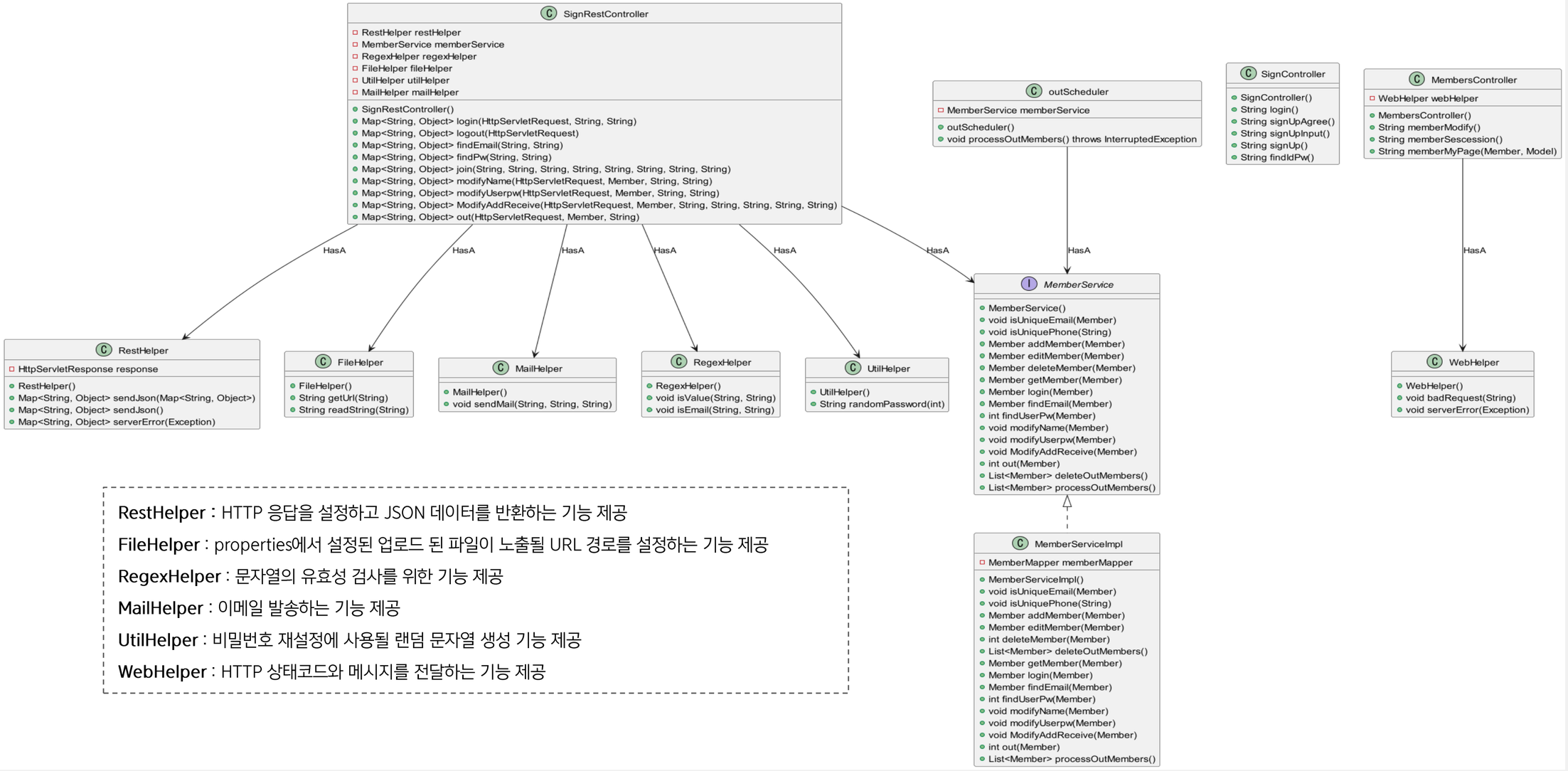
Member

SignRestController
회원 관련 정보를 처리하고 JSON형태로 객체 데이터를 반환

OutScheduler
회원 탈퇴한 회원들을 정해진 시간에 주기적으로 데이터를 지우는 기능

SignController, MemberController
View를 반환

MemberService
회원 정보를 추가, 수정, 조회, 삭제하는 기능



RestHelper : HTTP 응답을 설정하고 JSON 데이터를 반환하는 기능 제공

FileHelper : properties에서 설정된 업로드 된 파일이 노출될 URL 경로를 설정하는 기능 제공

RegexHelper : 문자열의 유효성 검사를 위한 기능 제공

MailHelper : 이메일 발송하는 기능 제공

UtilHelper : 비밀번호 재설정에 사용될 랜덤 문자열 생성 기능 제공

WebHelper : HTTP 상태코드와 메시지를 전달하는 기능 제공

4-1 클래스 다이어그램

Product

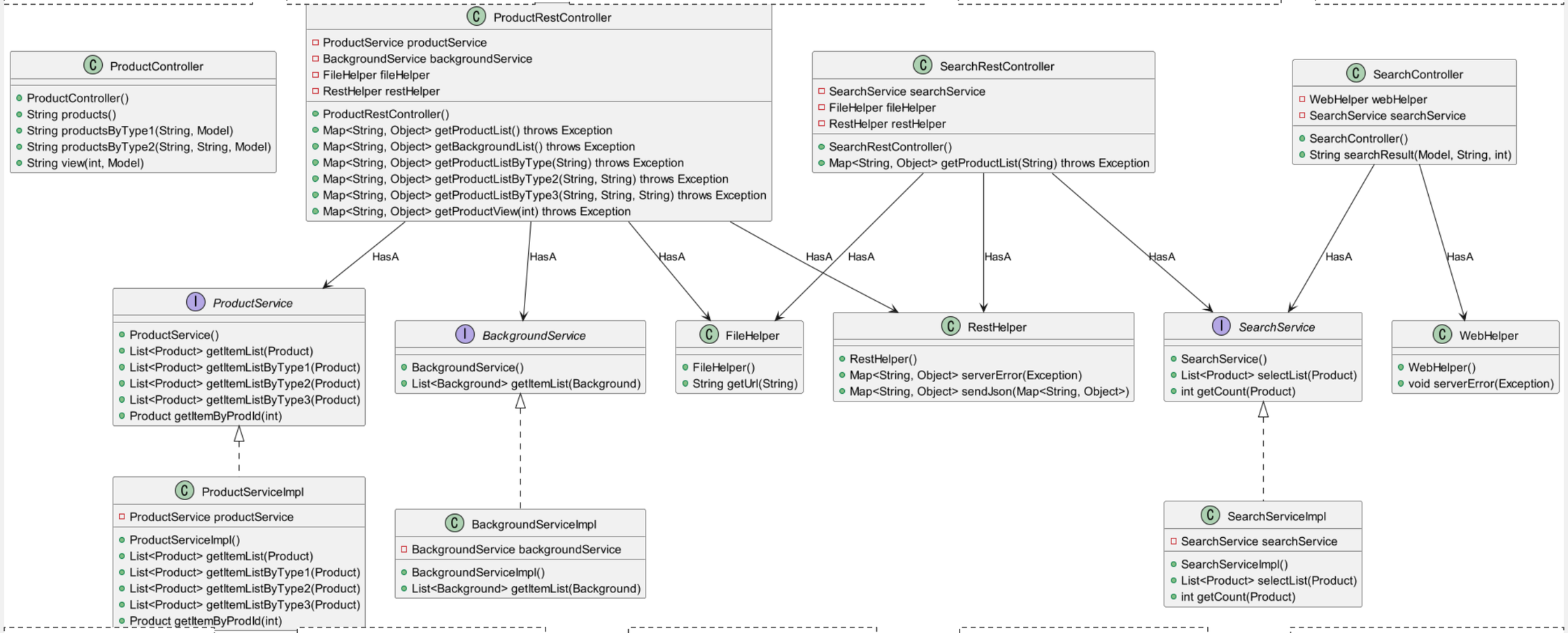
ProductController
제품의 유형(Type)별 페이지와 상세 페이지를 반환

ProductRestController
제품 관련 RESTful API 제공

FileHelper
properties에서 설정된 업로드 된 파일이 노출될 URL 경로를 설정하는 기능 제공

SearchRestController
Keyword를 기반으로 제품 목록을 조회하여 반환하는 RESTful API 제공

SearchController
Keyword를 기반으로 검색 결과 페이지 반환



ProductService
제품의 Type1,2,3 에 따라 제품 정보 조회

BackgroundService
제품의 배경 이미지 조회

RestHelper
HTTP 응답을 설정하고 JSON 데이터를 반환하는 기능 제공

SearchService
제품의 검색 결과에 대한 개수 조회 및 목록 조회

WebHelper
HTTP 상태코드와 메시지를 전달하는 기능 제공

4-1 클래스 다이어그램

Order

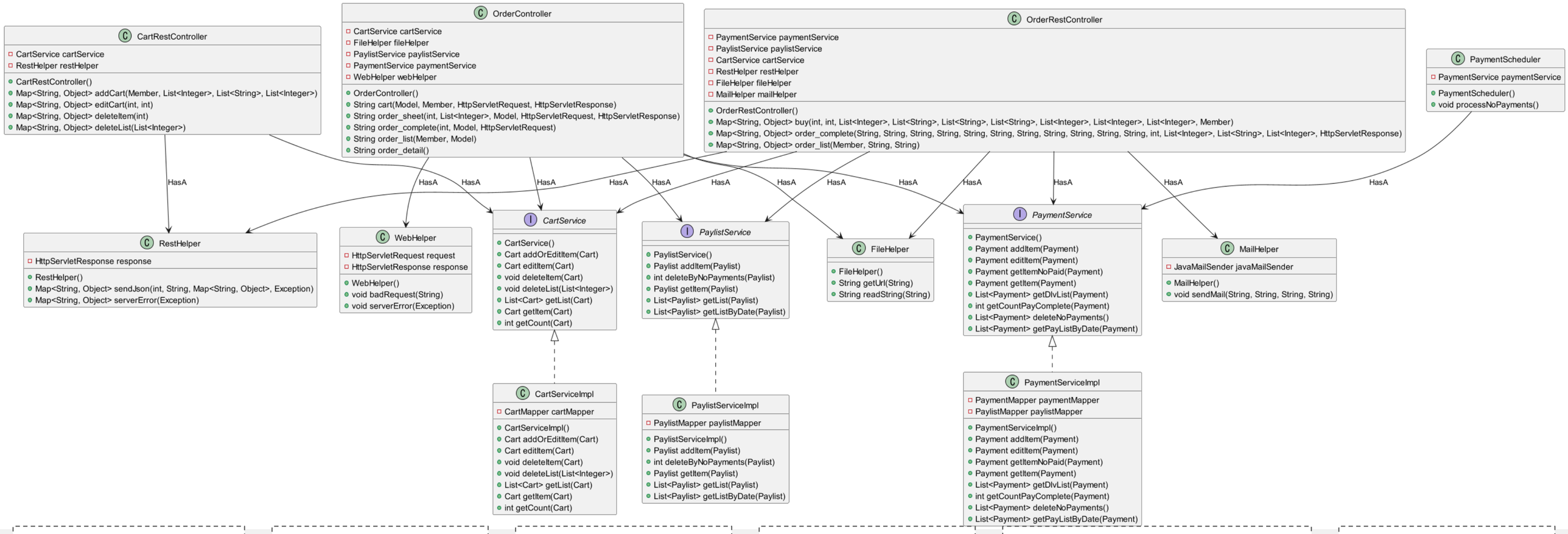
CartRestController
장바구니에 상품을 추가, 수정, 삭제하는 RESTful API 제공

OrderController
장바구니, 주문서, 주문 완료, 주문 목록 페이지 반환

OrderRestController
상품 구매, 주문 완료, 주문 목록 조회 등의 RESTful API 제공

MailHelper
이메일 발송하는 기능 제공

PaymentScheduler
미결제 상품을 조회하고 삭제하는 기능



RestHelper
HTTP 응답을 설정하고 JSON 데이터를 반환하는 기능 제공

WebHelper
HTTP 상태코드와 메시지를 전달하는 기능 제공

CartService
장바구니에 상품을 추가, 수정, 삭제, 조회

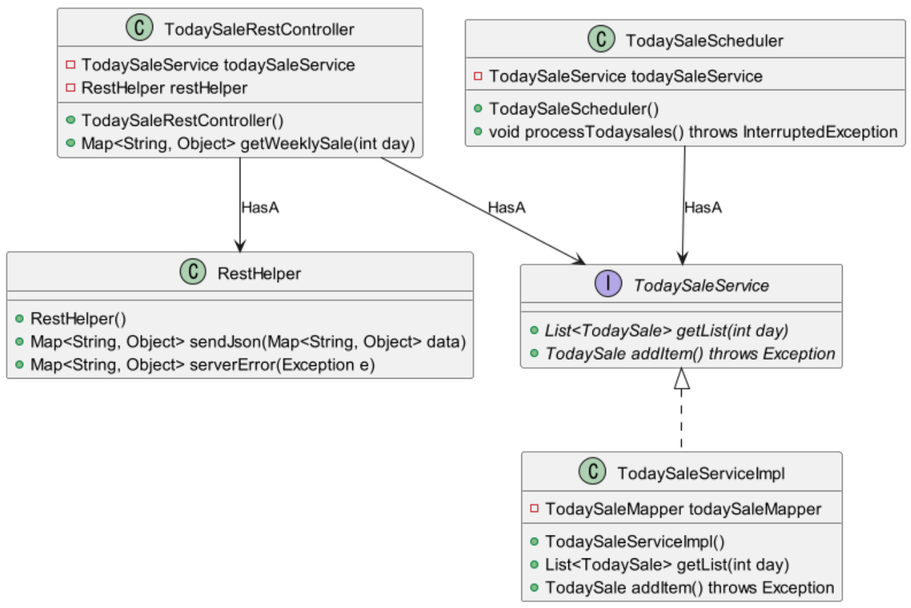
PaylistService
결제 목록에 항목을 추가, 조회, 삭제

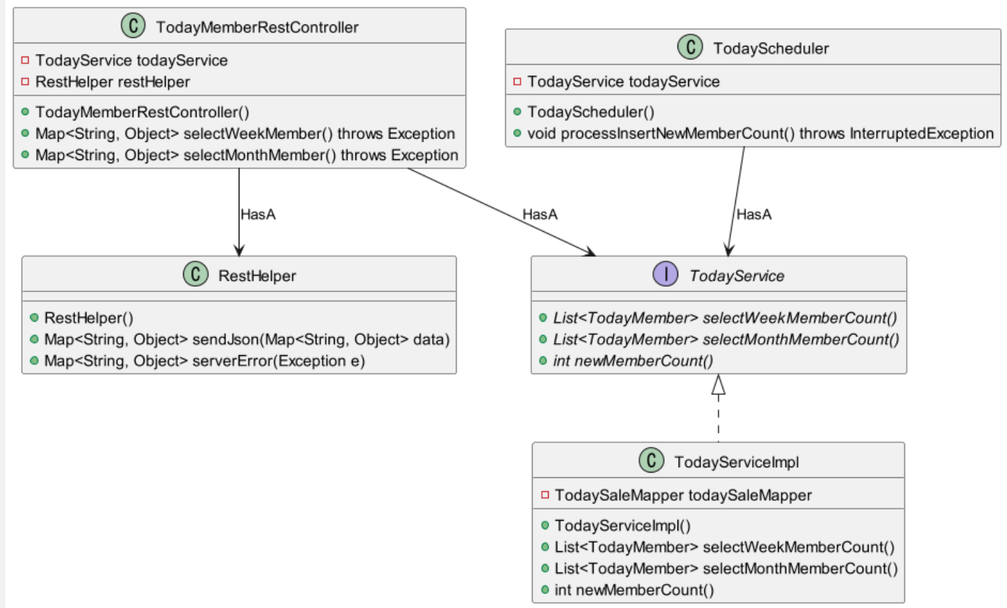
FileHelper
properties에서 설정된 업로드 된 파일이 노출될 URL 경로를 설정하는 기능 제공

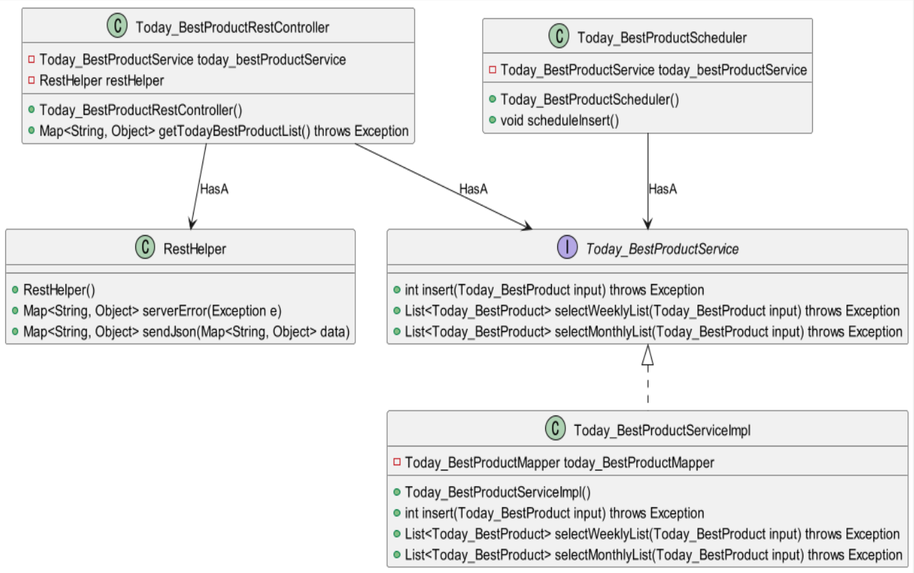
PaymentService
결제 항목을 추가, 수정, 조회, 삭제하는 기능

4-1 클래스 다이어그램

Dashboard







TodaySaleRestController : 총 매출을 조회하는 RESTful API 제공

TodaySaleScheduler : 매일 새벽 1시에 전날 매출 데이터를 집계하는 스케줄러

TodaySaleService : 전날 매출을 추가하고, 지정 기간 동안의 매출 데이터 조회

TodayMemberRestController : 주간 및 월간 신규 회원을 조회하는 RESTful API 제공

TodayScheduler : 매일 새벽 1시에 전날 신규 회원 데이터를 집계하는 스케줄러

TodayService : 신규 회원 데이터 추가 및 조회

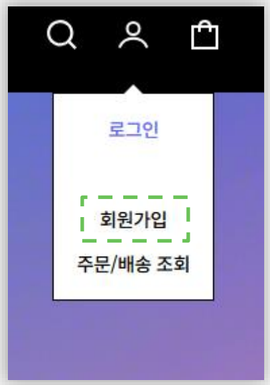
TodayBestProductRestController : 주간 및 월간 상품 판매량을 조회하는 RESTful API 제공

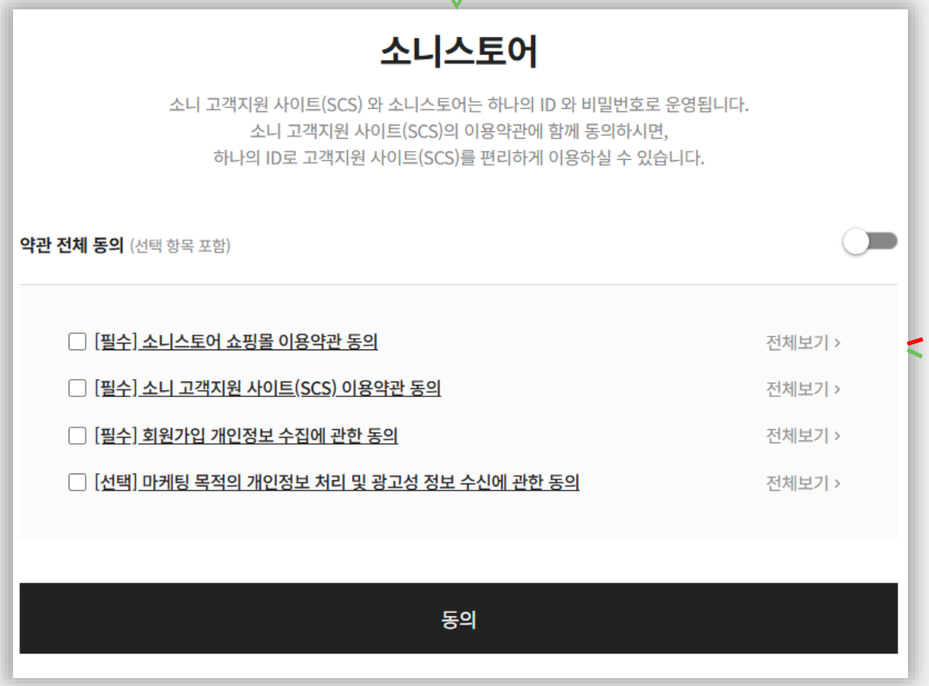
TodayBestProductScheduler : 매일 새벽 1시에 전날 판매량을 집계하는 스케줄러

TodayBestProductService : 전날 판매량을 추가하고, 주별, 월별 판매량 데이터 조회

RestHelper : HTTP 응답을 설정하고 JSON 데이터를 반환하는 기능 제공

4-2 기능설명 - 회원가입 (1)



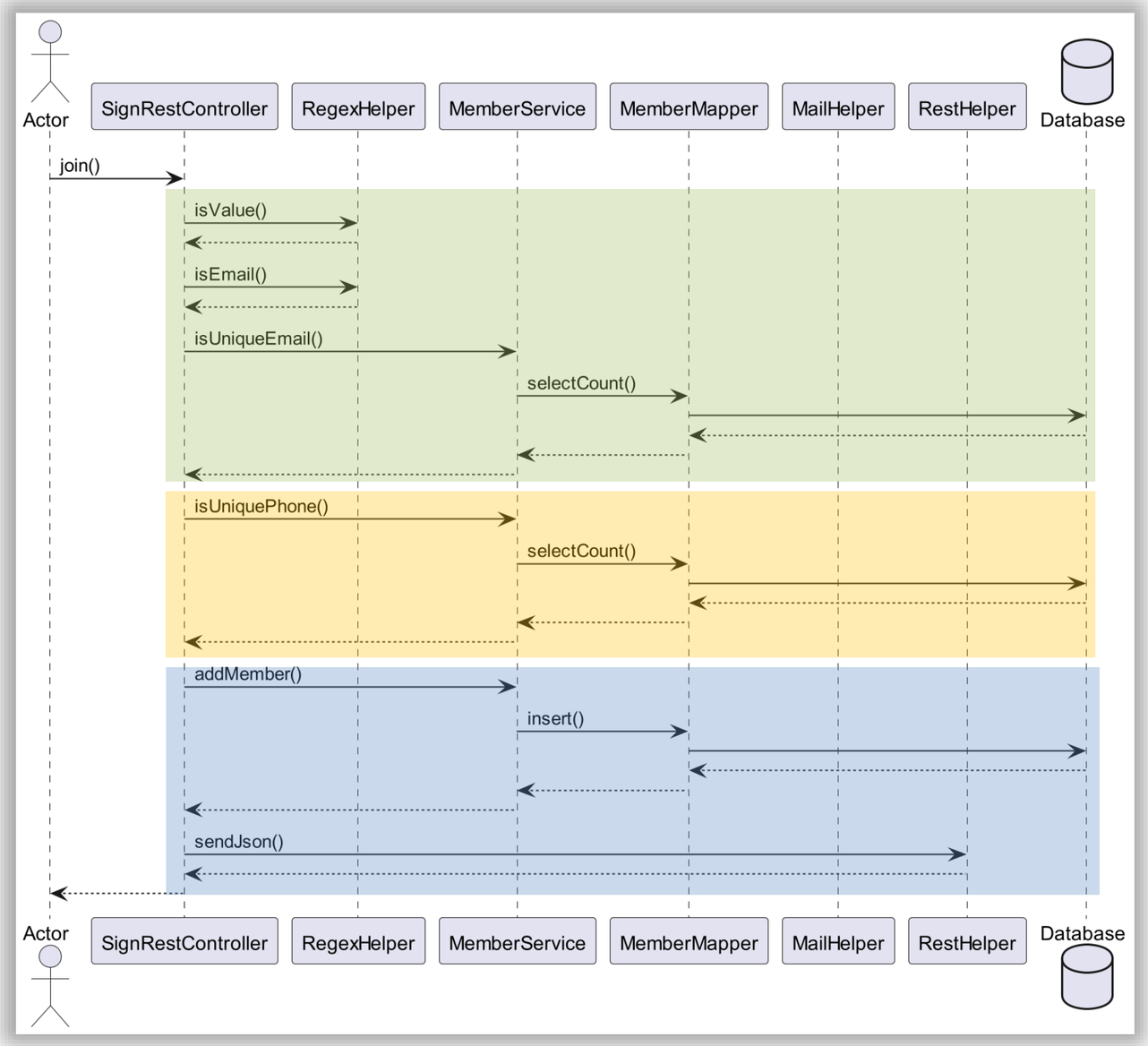


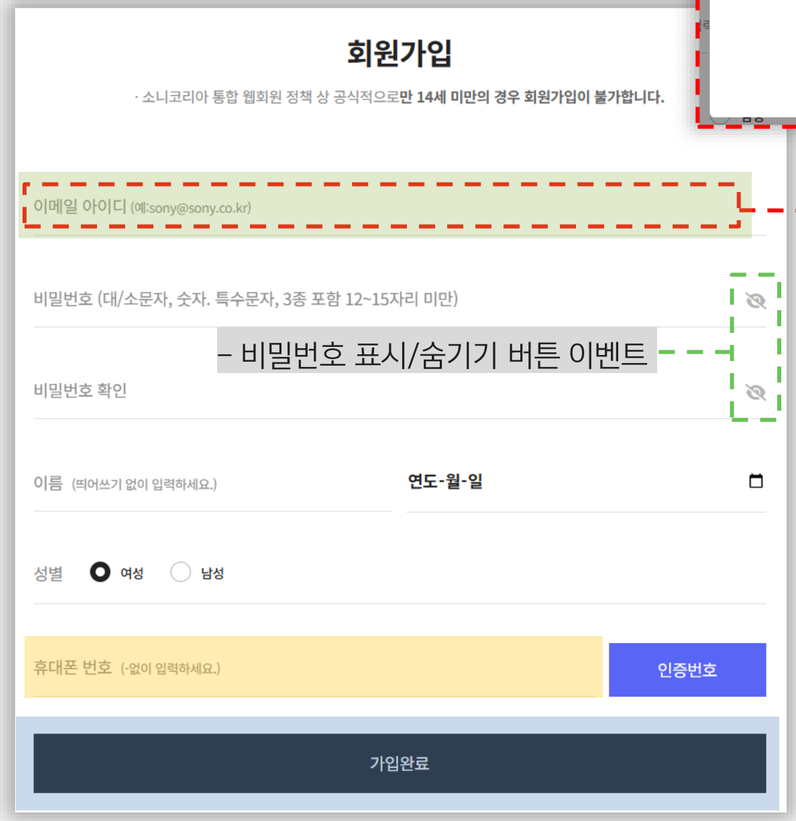
- [필수] 항목 체크 유무 검사 => 미체크 시 모달창 안내
- [선택] 항목 체크 유무 => 다음페이지로 queryString 전송
- 전체 체크 및 전체 해제 컨트롤 (하나라도 미동의 시 false)

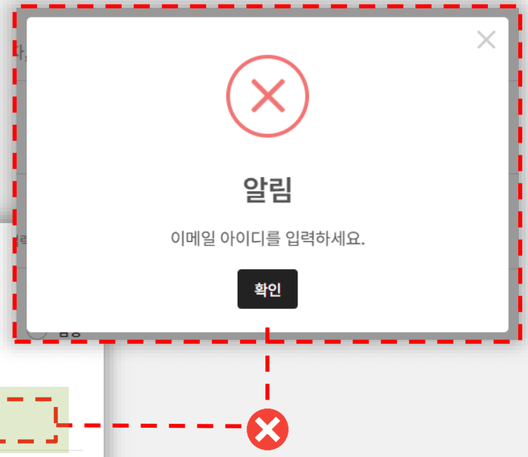
localhost:8080/sign/sign_up_input?receiveemail=Y&receivesms=Y

localhost:8080/sign/sign_up_input?receiveemail=N&receivesms=N

4-2 기능설명 - 회원가입 (2)



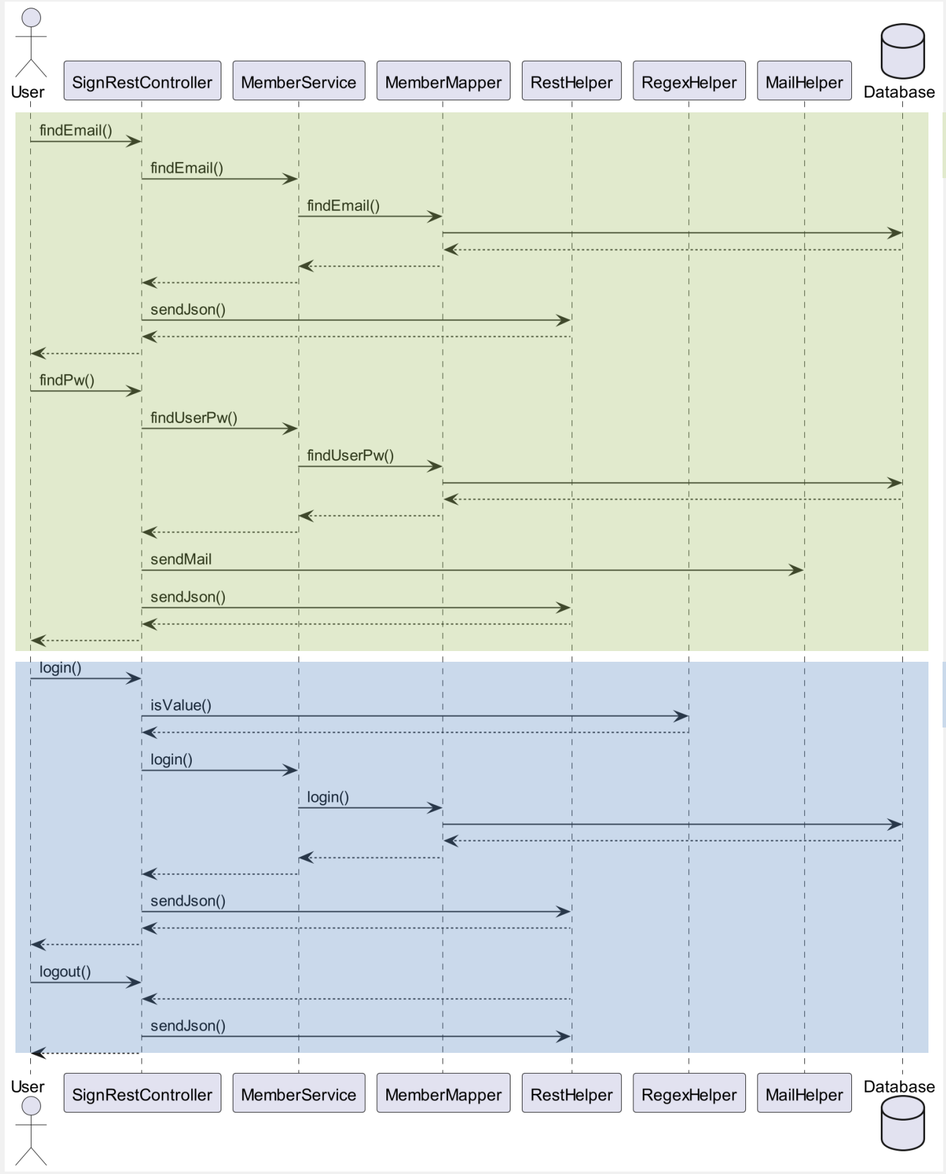




- 입력 가능한 값들을 유효성 검사 후 유효하지 않으면 alert 으로 메시지 출력
- 이메일 아이디와 휴대폰 번호는 중복 검사 진행
- 이메일 형식 검사, 비밀번호 영어 대/소문자와 특수문자 및 숫자 포함하는지 검사, 12글자 이상 20글자 이하인지 검사
- 생년월일 현재시간보다 이전인지 검사

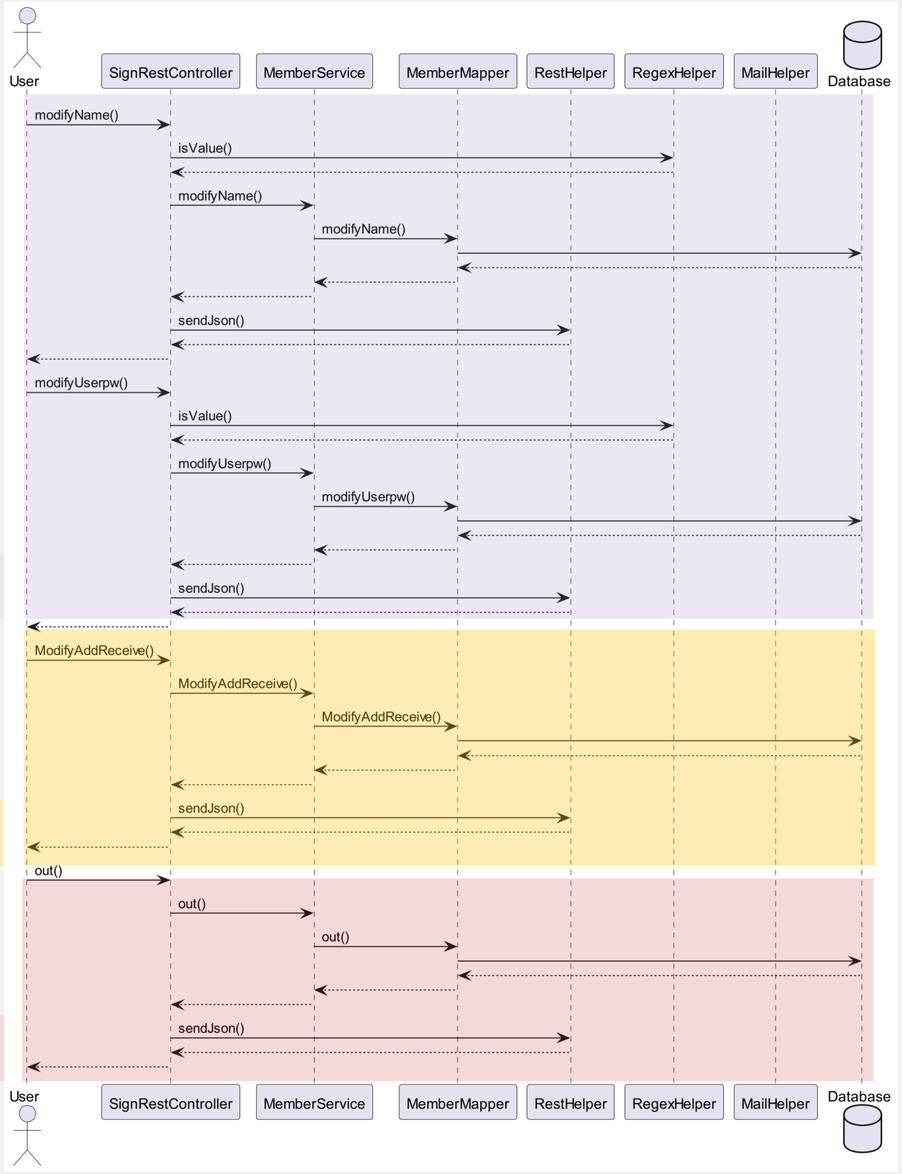
- 이전 페이지에서 쿼리스트링으로 보낸 문자 및 이메일 광고 수신 여부를 hidden 속성의 input 박스의 value 값 설정

4-2 기능설명 - 회원 정보 (시퀀스다이어그램)



아이디 · 비밀번호 찾기

로그인 · 로그아웃

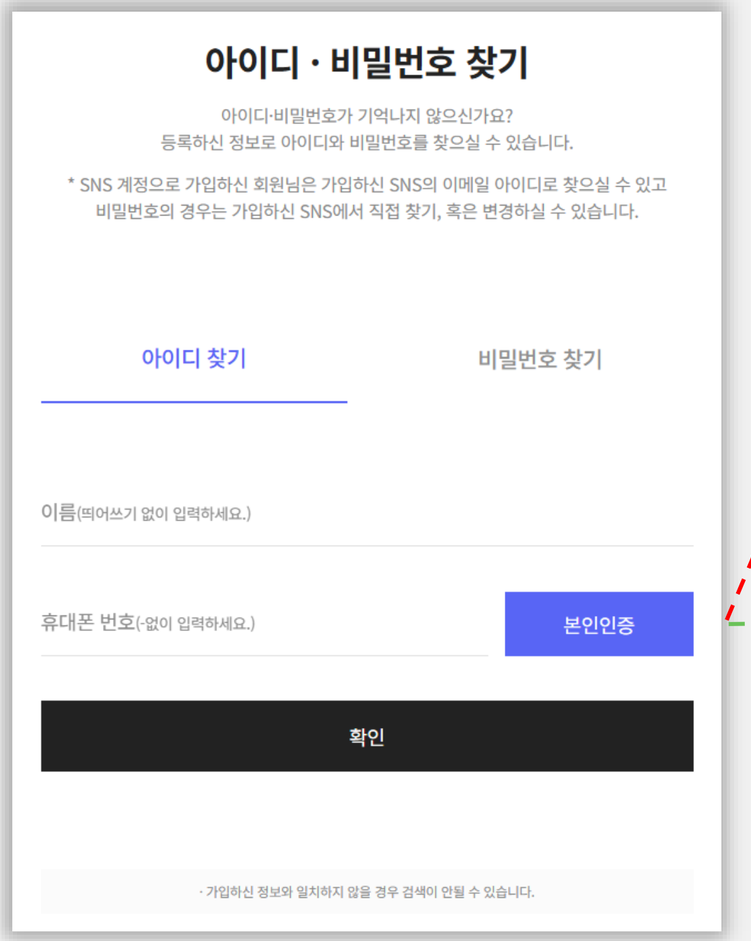


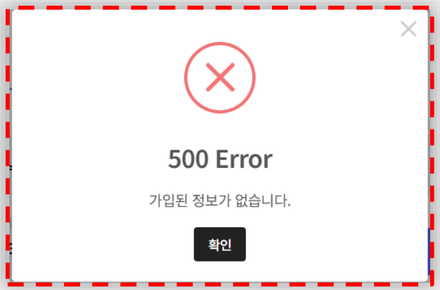
이름 · 비밀번호 변경

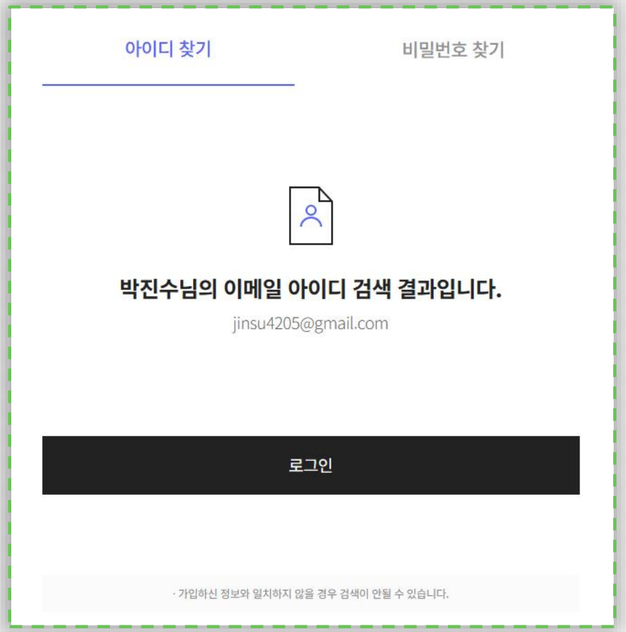
주소 · 광고 수신 여부 변경

회원 탈퇴

4-2 기능설명 - 아이디 찾기





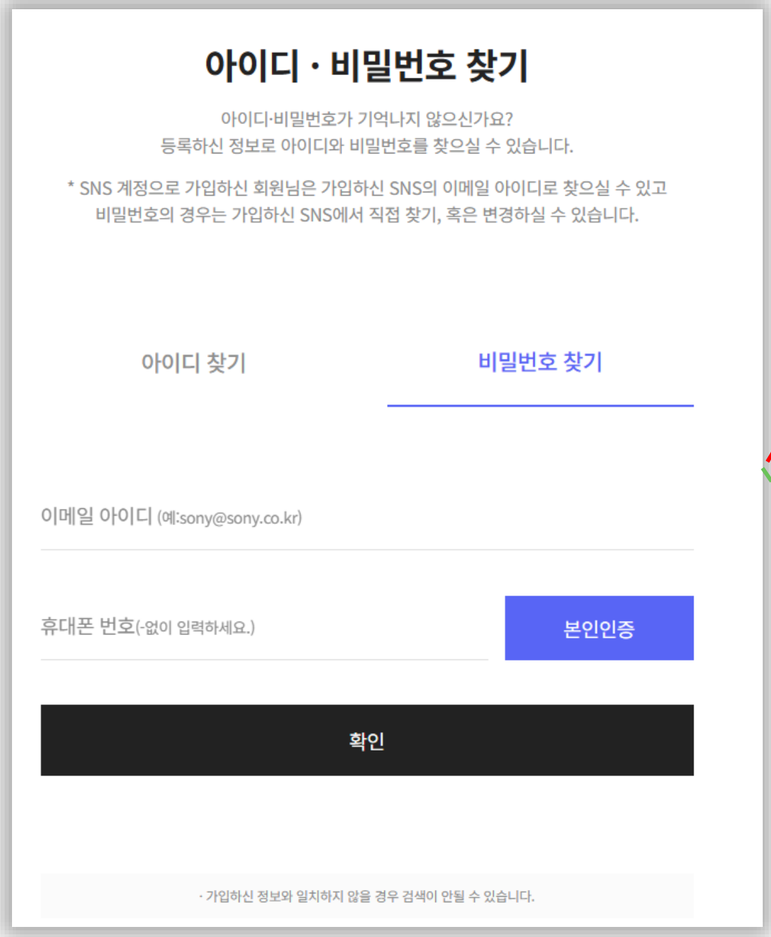


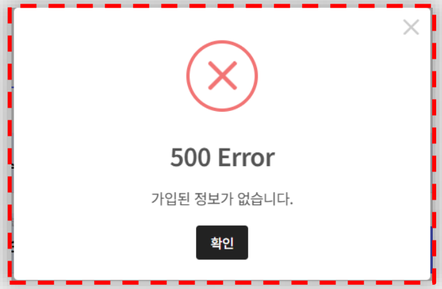
MemberMapper.java

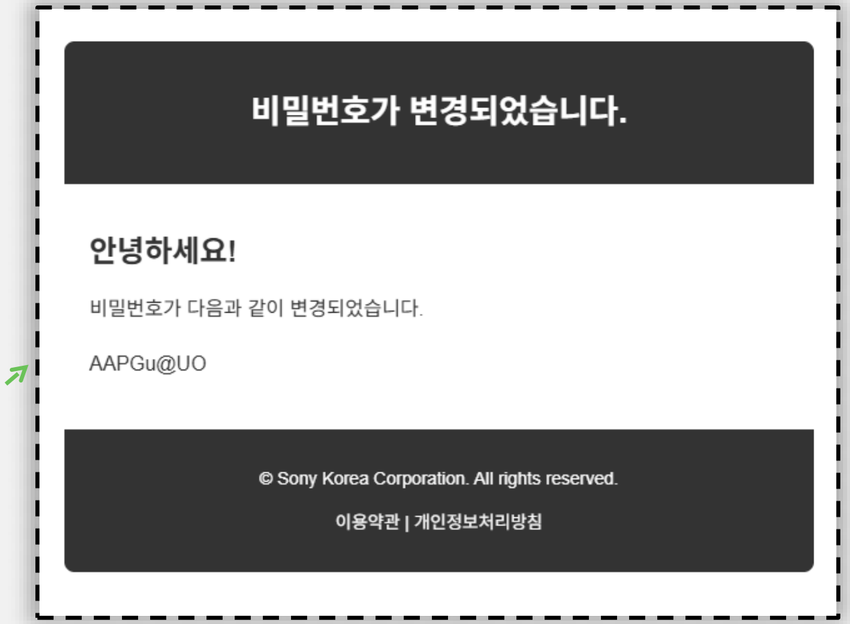
```
@Select("select email from members " +
                "where username = #{username} and phone = #{phone}")
@ResultMap("memberMap")
Member findEmail(Member input);
```

- 이름과 휴대폰 번호로 이메일 아이디를 검색 후 유효하다면 입력 form 을 숨기고 검색 결과 표시

4-2 기능설명 - 비밀번호 찾기







MemberMapper.java

```java
@Update("update members set userpw = #{userpw} " +
        "where email = #{email} and phone = #{phone}")
@ResultMap("memberMap")
int findUserPw(Member input);
```

- 이메일 아이디와 휴대폰 번호로 회원정보 검색 후 유효하다면 랜덤 문자열로 비밀번호 update 후 해당 이메일로 비밀번호 전송

4-2 기능설명 - 로그인, 로그아웃

header.html

```
<div class="mypage_drop_down flex_center" th:if="${session.memberInfo == null}">
    <a class="mypage_drop_down_login pointer" th:href="@{/sign/login}">로그인</a>
    <a class="mypage_drop_down_menu pointer" th:href="@{/sign/sign_up}">회원가입</a>
    <a class="mypage_drop_down_menu pointer" th:href="@{/my-page/order-list}">주문/배송 조회</a>
</div>
```

header.html

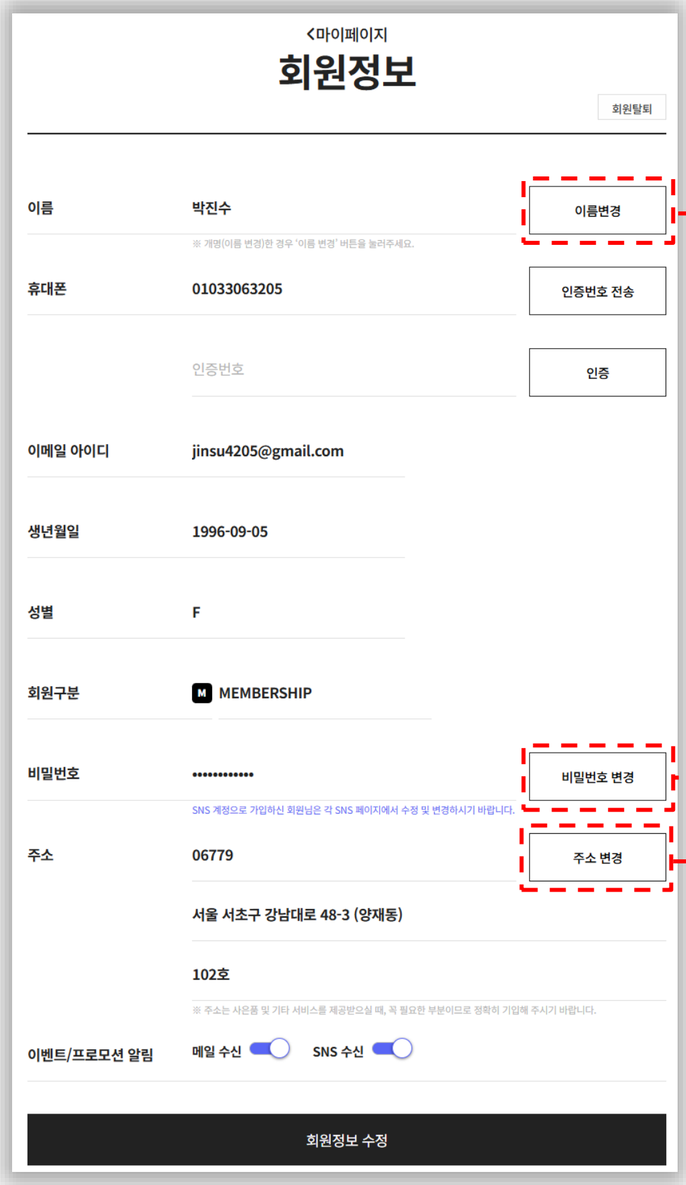
```
<div class="mypage_drop_down flex_center" style="height: 200px; text-align: center;;" th:unless="${session.memberInfo == null}">
    <a class="mypage_drop_down_login pointer"><span th:text="${session.memberInfo.username}" />님<br /> 안녕하세요!</a>
    <a class="mypage_drop_down_menu pointer" th:href="@{/members/my_page}">마이페이지</a>
    <a class="mypage_drop_down_menu pointer" th:href="@{/my-page/order-list}">주문/배송 조회</a>
    <a id="btn_logout" class="mypage_drop_down_menu pointer" style="text-decoration: none;">로그아웃</a>
    <script>
        document.querySelector('#btn_logout').addEventListener('click', async (e) => {
            e.preventDefault();

            const data = await axiosHelper.get("[[@{/api/sign/logout}]]");

            window.location = "[[@{/index}]]"
        });
    </script>
</div>
```

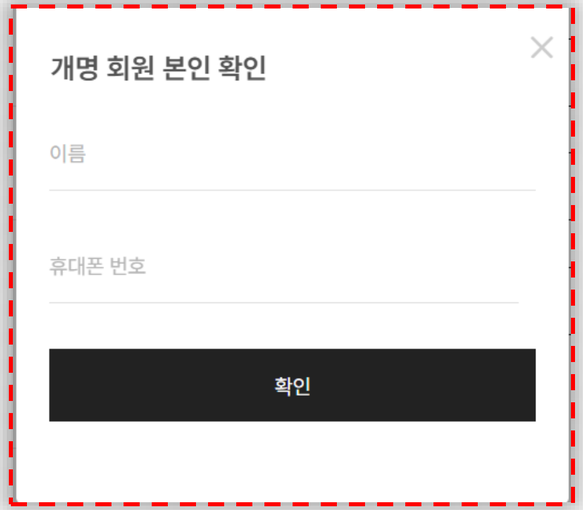
SignRestController.java

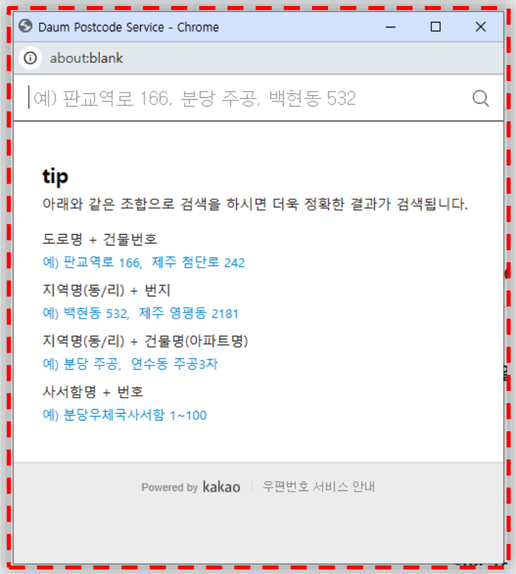
```
@GetMapping("/api/sign/logout")
public Map<String, Object> logout(HttpServletRequest request) {
    HttpSession session = request.getSession();
    session.invalidate();

    return restHelper.sendJson();
}
```

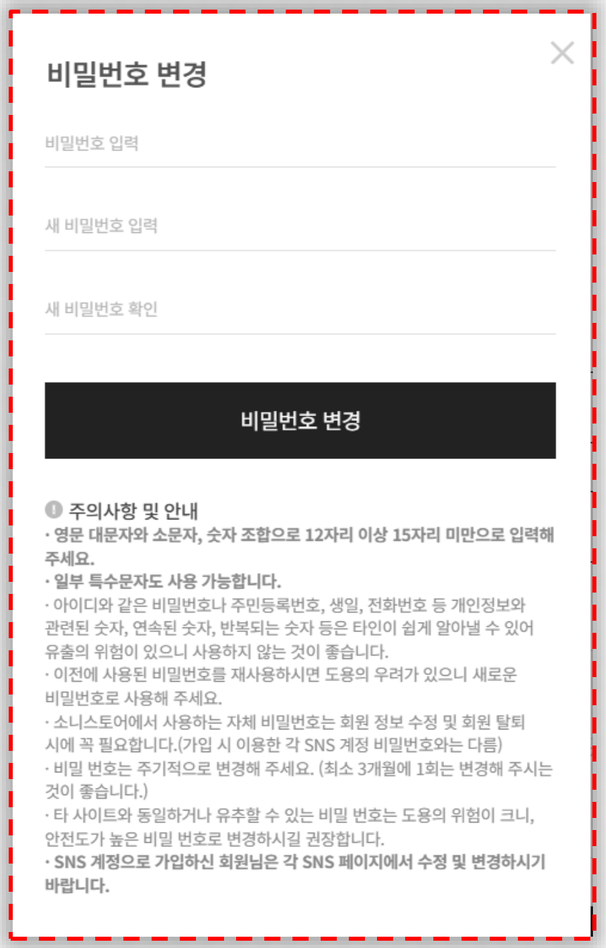
- 로그인 성공 시 해당 데이터를 session에 저장 하고 메인페이지로 이동
- thymeleaf를 이용하여 세션정보가 있다면 header 회원정보의 드롭다운 메뉴 구성 변경
- 로그아웃 클릭 시 세션정보를 삭제하고 메인페이지로 이동

4-2 기능설명 - 회원정보 수정



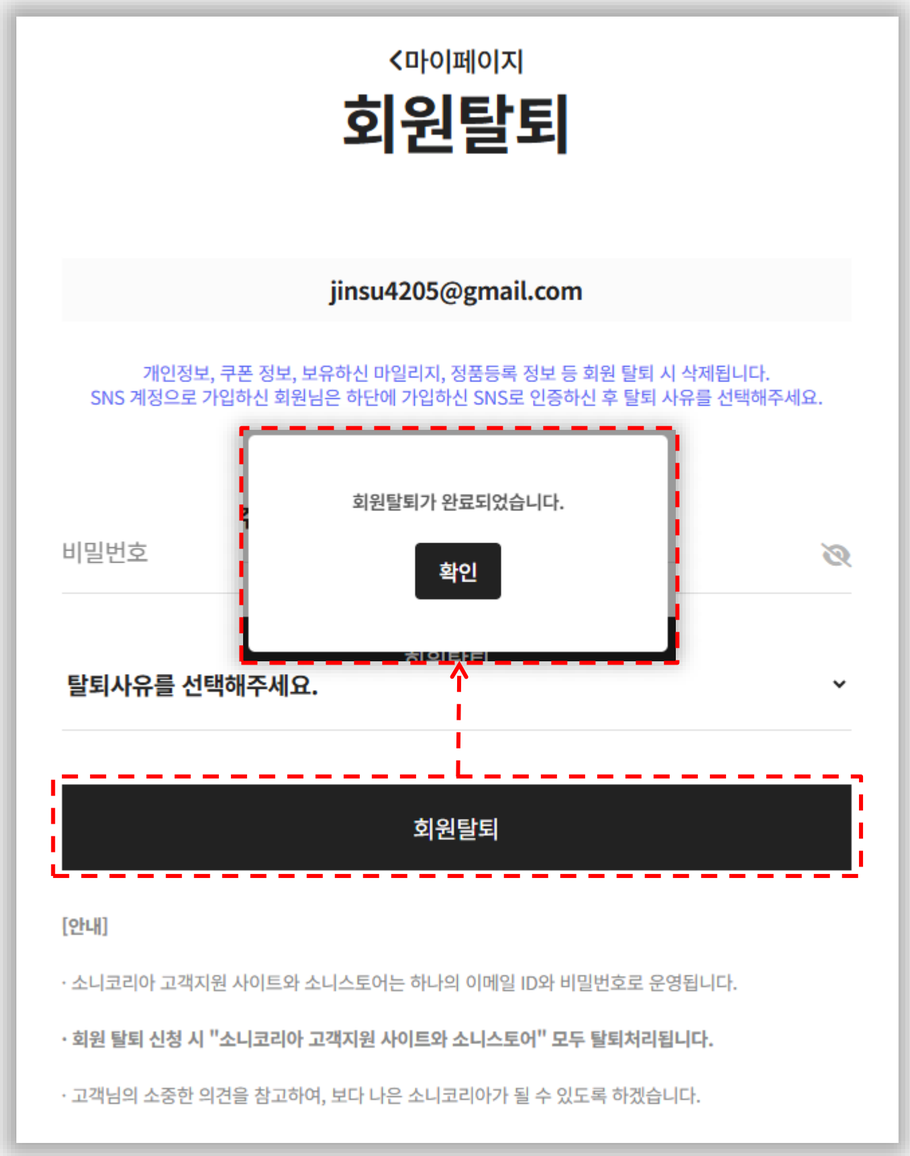






- 이름변경 시 휴대폰 번호가 세션에 저장된 번호와 일치하는지 검사
- 비밀번호 변경 시 회원가입과 동일한 유효성 검사
- 이름과 비밀번호는 개별 수정이며, 그 외 정보 수정 시 본인인증 유무 검사
- 변경된 정보는 즉시 세션을 갱신하여 화면 렌더링

4-2 기능설명 - 회원탈퇴



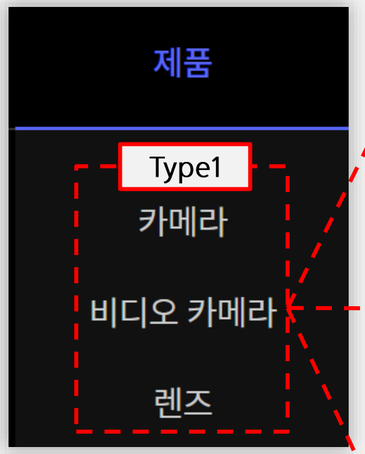
MemberMapper.java

```
@Update("update members \n" +
                "set isout = 'Y', editdate = now() \n" +
                "where memberid = #{memberid} and userpw = #{userpw} and isout = 'N'")
public int out(Member input);
```

```
@Delete("delete from members \n" +
                "where isout='Y' and \n" +
                "editdate < date_add(now(), interval -7 day)")
public int deleteOutMembers();
```

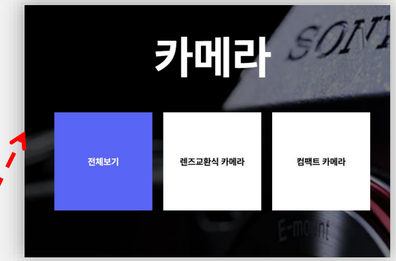
- 회원탈퇴 시 세션의 비밀번호와 비밀번호 입력값이 일치하면 isout = “Y”로 값 수정
- 스케줄러를 이용해 매주 월요일 00시 isout이 “Y”로 수정된 시간이 7일이 지난 데이터 삭제

4-2 기능설명 - 제품 목록 페이지 (1)

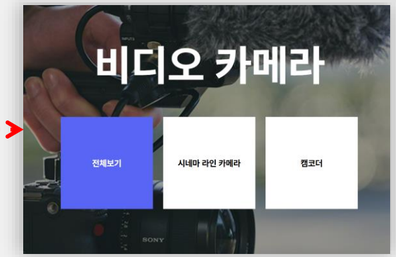


- 상단 헤더 제품 부분에서 해당 제품 목록 페이지로 이동

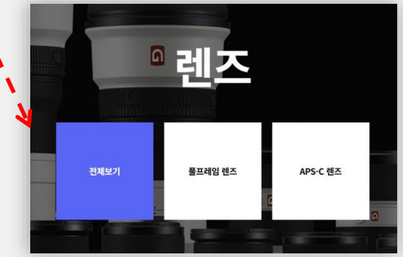
localhost:8080/products/camera



localhost:8080/products/video



localhost:8080/products/lens

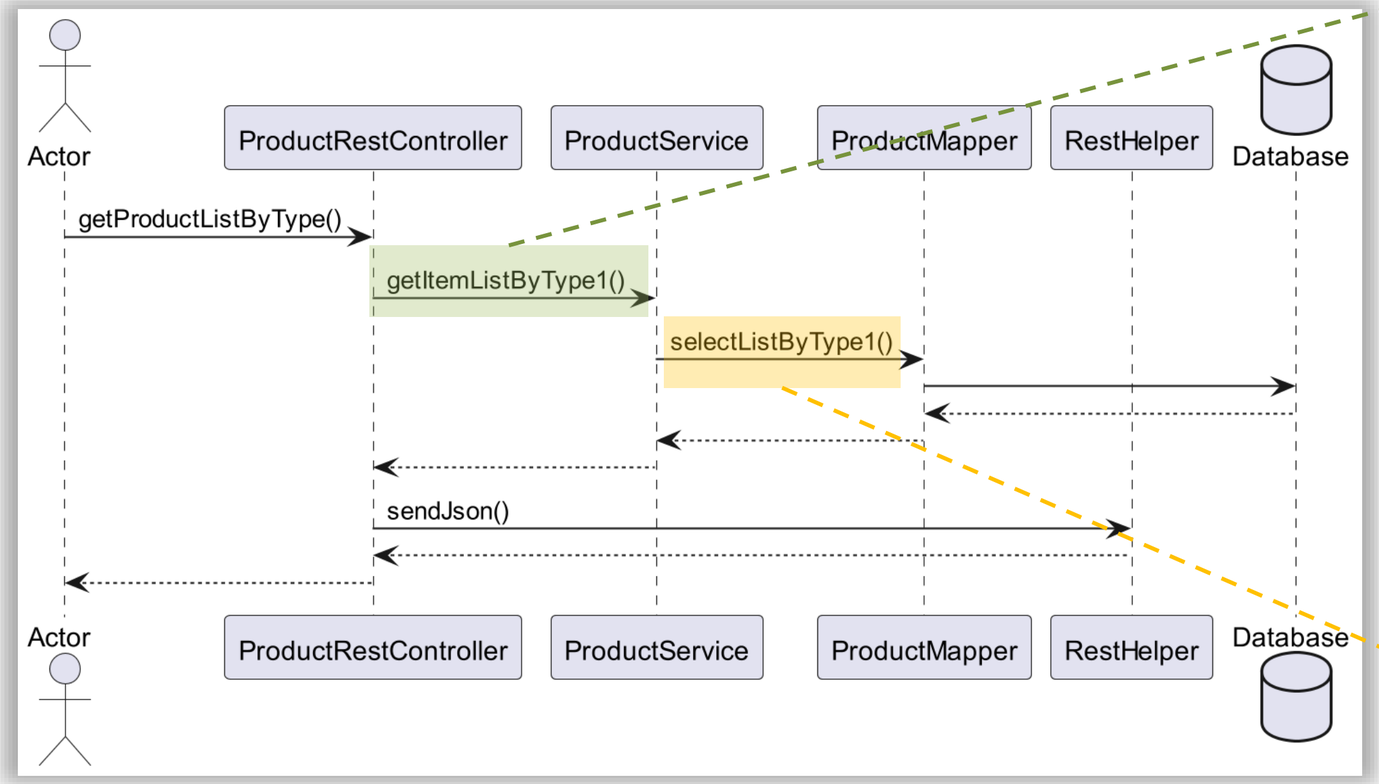


ProductRestController.java

```java
// 타입별 제품 목록 조회
http://127.0.0.1:8080/api/products/{type} (Count=10 Total=2.25s Max=0.00s)
@GetMapping("/api/products/{type}")
public Map<String, Object> getProductListByType(@PathVariable("type") String type) throws Exception {
    List<Product> output;
    try {
        Product input = new Product();
        input.setType1(type);
        output = productService.getItemListByType1(input);
        if (output != null) {
            output.forEach(product -> {
                if (product.getImages() != null) {
                    product.getImages().forEach(image -> {
                        image.setFilepath(fileHelper.getUrl(image.getFilepath()));
                    });
                }
            });
        }
    } catch (Exception e) {
        return restHelper.serverError(e);
    }
    Map<String, Object> data = new LinkedHashMap<>();
    data.put(key:"list", output);

    return restHelper.sendJson(data);
}
```

- 제품의 Type1 에 따라 제품 목록을 JSON 형식으로 반환하는 RESTful API

4-2 기능설명 - 제품 목록 페이지 (2)



ProductServiceImpl.java

```java
@Override
public List<Product> getItemListByType1(Product input) throws Exception {
    List<Product> result = productMapper.selectListByType1(input);

    if (result == null) {
        throw new Exception(message:"상품 정보 조회에 실패했습니다.");
    }

    // 중복된 제품 제거
    result = result.stream().distinct().collect(Collectors.toList());

    for (Product temp : result) {
        Image image = new Image();
        image.setProdid(temp.getProdid());

        temp.setImages(imageMapper.selectImagesByProductId(temp));
        temp.setColors(colorMapper.selectColorsByProductId(temp));
    }

    return result;
}
```

- 제품의 Type1 값에 따라 필터링된 제품 목록을 조회하고, 중복된 항목을 제거한 후 각 제품에 이미지와 색상 정보를 추가하여 반환

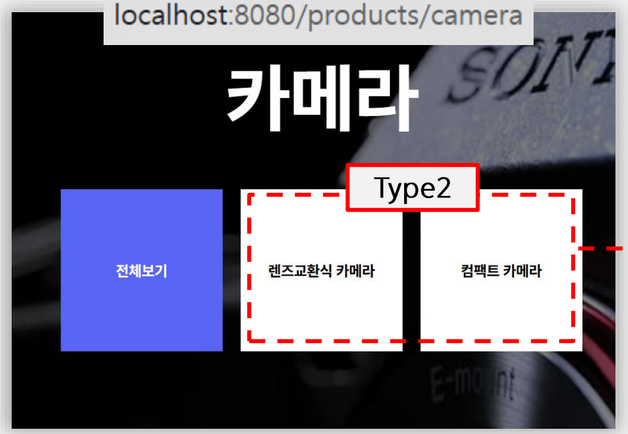
ProductMapper.java

```java
// type1에 따른 상품 목록 조회
@Select("SELECT p.prodid, p.title, p.proddesc, p.price, p.type1, p.type2, p.type3, " +
        "p.date, p.detailimage1, p.detailimage2, p.youtube, p.detailgif, " +
        "p.detailspec, p.soldout, p.sale, p.event, " +
        "i.imgid, i.filepath, i.thumbnail, " +
        "c.colorid, c.color, c.pcolor " +
        "FROM products p " +
        "LEFT JOIN images i ON p.prodid = i.prodid " +
        "LEFT JOIN colors c ON p.prodid = c.prodid " +
        "WHERE p.type1 = #{type1} " +
        "ORDER BY p.prodid DESC ")
        @ResultMap("productMap")
        List<Product> selectListByType1(Product input);
```

- 제품의 Type1 에 따른 제품 목록(이미지와 색상 포함)을 조회하는 SQL 쿼리를 정의

4-2 기능설명 - 제품 목록 페이지 (3)

localhost:8080/products/camera

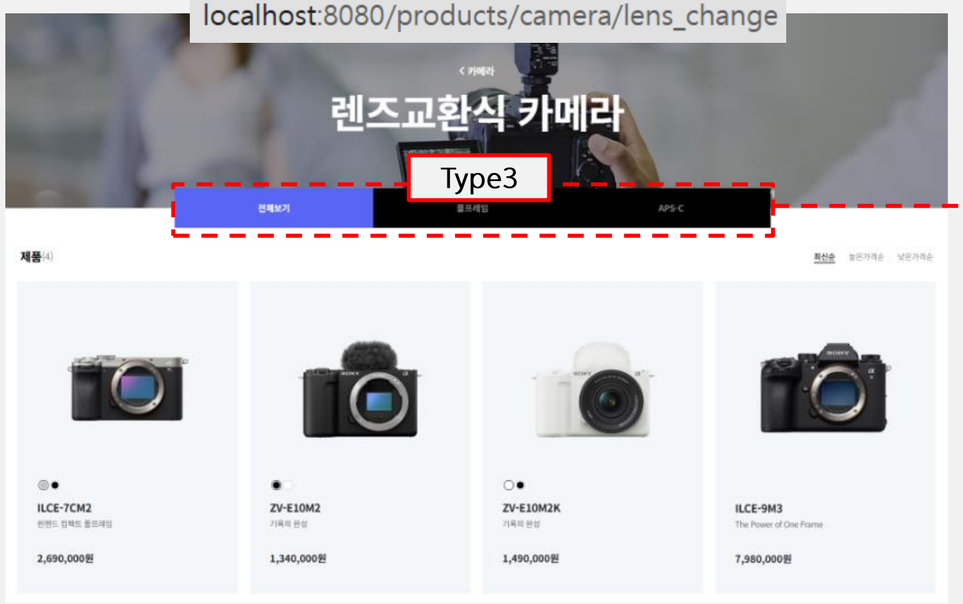


ProductRestController.java

```java
@GetMapping("/api/products/{type}/{type2}")
public Map<String, Object> getProductListByType2(@PathVariable("type") String type, @PathVariable("type2") String type2) throws Exception {
    List<Product> output;
    try {
        Product input = new Product();
        input.setType1(type);
        input.setType2(type2);
        output = productService.getItemListByType2(input);
        if (output != null) {
            output.forEach(product -> {
                if (product.getImages() != null) {
                    product.getImages().forEach(image -> {
                        image.setFilepath(fileHelper.getUrl(image.getFilepath()));
                    });
                }
            });
        }
    } catch (Exception e) {
        return restHelper.serverError(e);
    }
    Map<String, Object> data = new LinkedHashMap<>();
    data.put(key:"list", output);

    return restHelper.sendJson(data);
```

- 제품의 Type1, 2에 따라 제품 목록을 JSON 형식으로 반환하는 RESTful API
- 만약 Type3 값이 없는 제품이라면 탭메뉴를 생성하지 않음

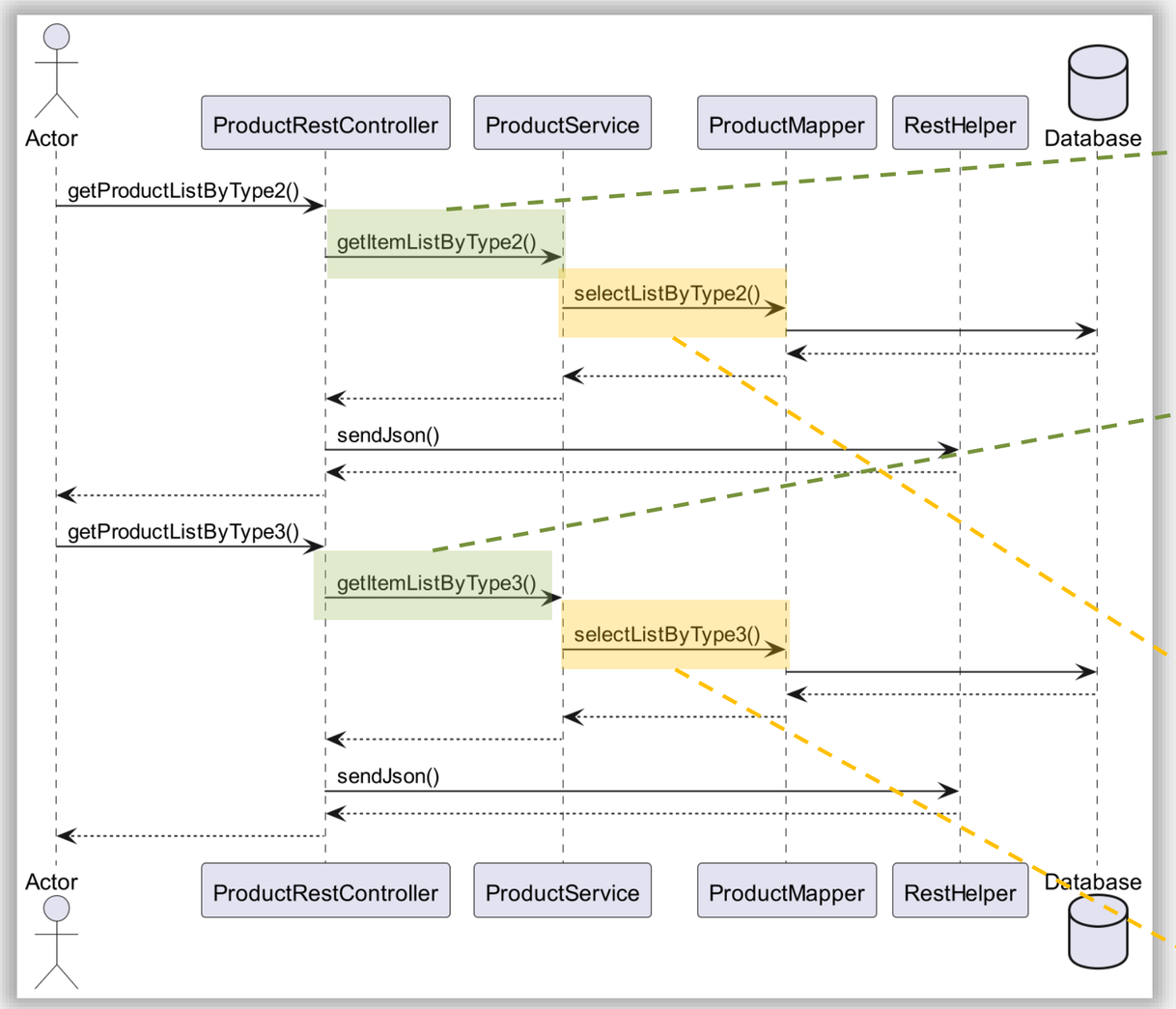
localhost:8080/products/camera/lens_change




```java
@GetMapping("/api/products/{type}/{type2}/{type3}")
public Map<String, Object> getProductListByType3(@PathVariable("type") String type, @PathVariable("type2") String type2, @PathVariable("type3") String type3) throws Exception {
    List<Product> output;
    try {
        Product input = new Product();
        input.setType1(type);
        input.setType2(type2);
        input.setType3(type3);
        output = productService.getItemListByType3(input);
        if (output != null) {
            output.forEach(product -> {
                if (product.getImages() != null) {
                    product.getImages().forEach(image -> {
                        image.setFilepath(fileHelper.getUrl(image.getFilepath()));
                    });
                }
            });
        }
    } catch (Exception e) {
        return restHelper.serverError(e);
    }
    Map<String, Object> data = new LinkedHashMap<>();
    data.put(key:"list", output);

    return restHelper.sendJson(data);
```

- 페이지 이동 없이 type3 값에 따라 탭 메뉴로 제품 목록 분류 (컨트롤러 구현X)

4-2 기능설명 - 제품 목록 페이지 (4)



ProductServiceImpl.java

```java
@Override
public List<Product> getItemListByType2(Product input) throws Exception {
    List<Product> result = productMapper.selectListByType2(input);

    if (result == null) {
        throw new Exception(message:"상품 정보 조회에 실패했습니다.");
    }

    // 중복된 제품 제거
    result = result.stream().distinct().collect(Collectors.toList());

    for (int i = 0; i < result.size(); i++) {
        Product temp = result.get(i);

        Image image = new Image();
        image.setProdid(temp.getProdid());

        temp.setImages(imageMapper.selectImagesByProductId(temp));
        temp.setColors(colorMapper.selectColorsByProductId(temp));
    }

    return result;
}
```

```java
public List<Product> getItemListByType3(Product input) throws Exception {
    List<Product> result = productMapper.selectListByType3(input);

    if (result == null) {
        throw new Exception(message:"상품 정보 조회에 실패했습니다.");
    }

    // 중복된 제품 제거
    result = result.stream().distinct().collect(Collectors.toList());

    for (int i = 0; i < result.size(); i++) {
        Product temp = result.get(i);

        Image image = new Image();
        image.setProdid(temp.getProdid());

        temp.setImages(imageMapper.selectImagesByProductId(temp));
        temp.setColors(colorMapper.selectColorsByProductId(temp));
    }

    return result;
}
```

- Type2, Type3 값에 따라 필터링된 제품 목록을 조회하고, 중복된 항목을 제거한 후 각 제품에 이미지와 색상 정보를 추가하여 반환

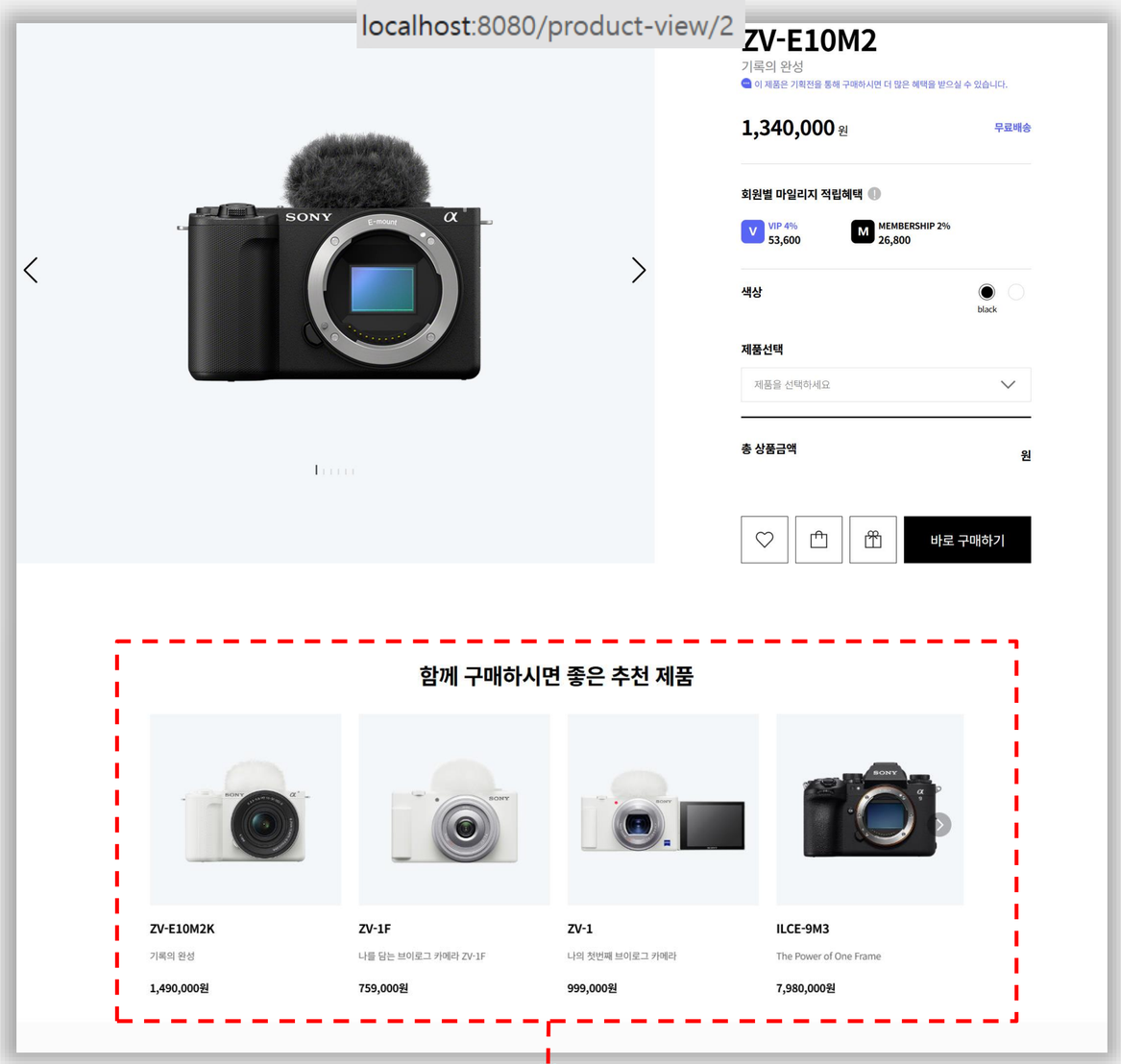
ProductMapper.java

```java
// type2에 따른 상품 목록 조회
@Select("SELECT p.prodid, p.title, p.proddesc, p.price, p.type1, p.type2, p.type3, " +
        "p.date, p.detailimage1, p.detailimage2, p.youtube, p.detailgif, " +
        "p.detailspec, p.soldout, p.sale, p.event, " +
        "i.imgid, i.filepath, i.thumbnail, " +
        "c.colorid, c.color, c.pcolor " +
        "FROM products p " +
        "LEFT JOIN images i ON p.prodid = i.prodid " +
        "LEFT JOIN colors c ON p.prodid = c.prodid " +
        "WHERE p.type2 = #{type2} " +
        "ORDER BY p.prodid DESC ")
        @ResultMap("productMap")
        List<Product> selectListByType2(Product input);
```

```java
// type3에 따른 상품 목록 조회
@Select("SELECT p.prodid, p.title, p.proddesc, p.price, p.type1, p.type2, p.type3, " +
        "p.date, p.detailimage1, p.detailimage2, p.youtube, p.detailgif, " +
        "p.detailspec, p.soldout, p.sale, p.event, " +
        "i.imgid, i.filepath, i.thumbnail, " +
        "c.colorid, c.color, c.pcolor " +
        "FROM products p " +
        "LEFT JOIN images i ON p.prodid = i.prodid " +
        "LEFT JOIN colors c ON p.prodid = c.prodid " +
        "WHERE p.type3 = #{type3} " +
        "ORDER BY p.prodid DESC ")
        @ResultMap("productMap")
        List<Product> selectListByType3(Product input);
```

- 제품의 Type2, Type3 에 따른 제품 목록(이미지와 색상 포함)을 조회하는 SQL 쿼리를 정의

4-2 기능설명 - 제품 상세 페이지



- 해당 제품의 Type과 같은 제품 목록을 받아 랜덤으로 상품 10개를 슬라이드 형식으로 구현

ProductRestController.java

```java
//제품 상세 조회
http://127.0.0.1:8080/api/product-view/{prodid}
@GetMapping("/api/product-view/{prodid}")
public Map<String, Object> getProductView(@PathVariable("prodid") int prodid) throws Exception {
    Product output;
    try {
        output = productService.getItemByProdId(prodid);
        if (output != null) {
            if (output.getImages() != null) {
                output.getImages().forEach(image -> {
                    image.setFilepath(fileHelper.getUrl(image.getFilepath()));
                });
            }
            if (output.getDetailimage1() != null) {
                output.setDetailimage1(fileHelper.getUrl(output.getDetailimage1()));
            }
            if (output.getDetailgif() != null) {
                output.setDetailgif(fileHelper.getUrl(output.getDetailgif()));
            }
            if (output.getDetailimage2() != null) {
                output.setDetailimage2(fileHelper.getUrl(output.getDetailimage2()));
            }
            if (output.getDetailspec() != null) {
                output.setDetailspec(fileHelper.getUrl(output.getDetailspec()));
            }
        }
    } catch (Exception e) {
        return restHelper.serverError(e);
    }
    Map<String, Object> data = new LinkedHashMap<>();
    data.put(key:"product", output);

    return restHelper.sendJson(data);
}
```

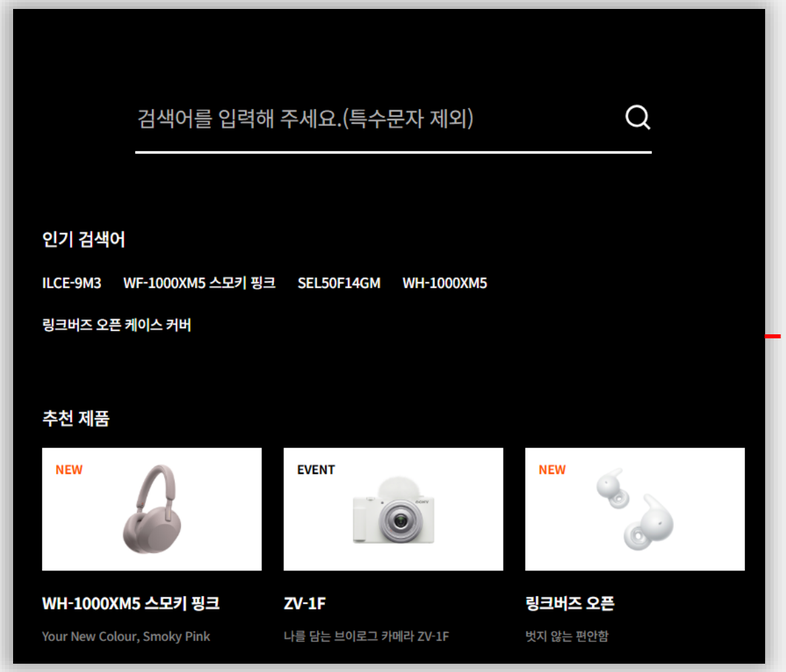
- 제품의 prodid에 따라 JSON 형식으로 반환하는 RESTful API

ProductMapper.java

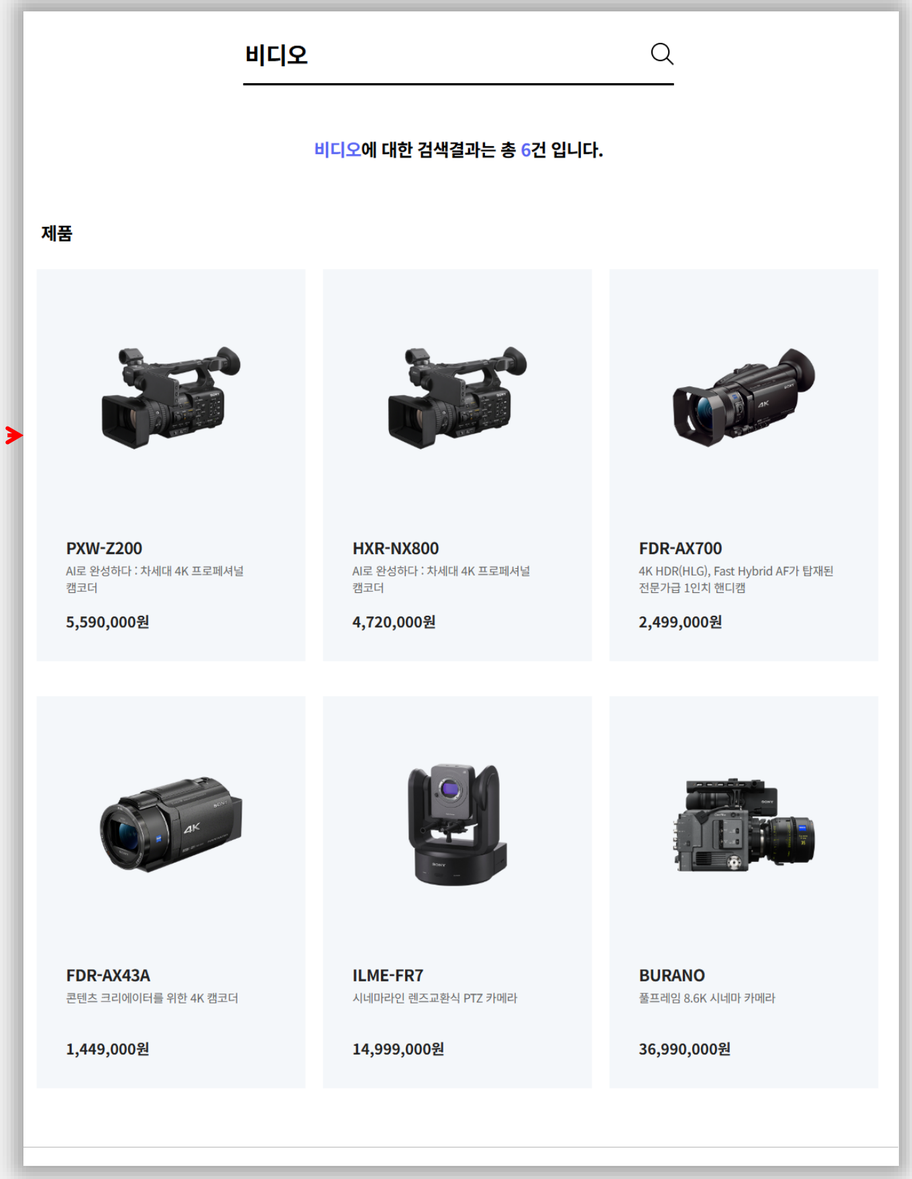
```java
// prodid에 따른 상품 조회
@Select("SELECT p.prodid, p.title, p.proddesc, p.price, p.type1, p.type2, p.type3, " +
        "p.date, p.detailimage1, p.detailimage2, p.youtube, p.detailgif, " +
        "p.detailspec, p.soldout, p.sale, p.event, " +
        "i.imgid, i.filepath, i.thumbnail, " +
        "c.colorid, c.color, c.pcolor " +
        "FROM products p " +
        "LEFT JOIN images i ON p.prodid = i.prodid " +
        "LEFT JOIN colors c ON p.prodid = c.prodid " +
        "WHERE p.prodid = #{prodid} " +
        "ORDER BY p.prodid DESC " +
        "LIMIT 1")
@ResultMap("productMap")
Product selectItemByProdid(int prodid);
```

- prodid에 따른 제품(이미지와 색상 포함)을 조회하는 SQL 쿼리를 정의

4-2 기능설명 - 제품 검색



- header에서 돋보기 아이콘을 눌러 검색 창 접근 가능
- 키워드가 포함된 문자열 검색 후 검색된 수와 제품 리스트 렌더링

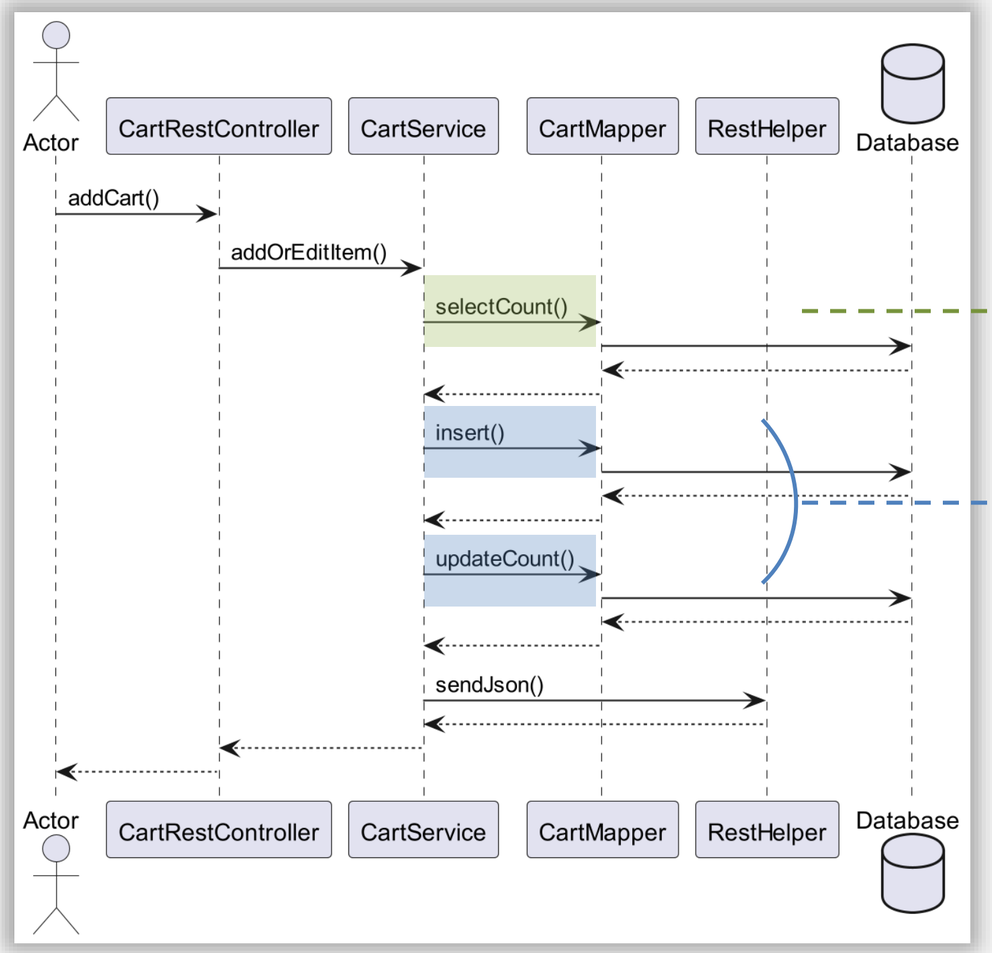


SearchMapper.java

```
@Select("<script>" +
        "select distinct " +
        "p.prodid, p.title, p.proddesc, p.price, p.type1, " +
        "p.type2, p.type3, p.date, p.soldout, p.sale, " +
        "p.event, i.imgid, i.filepath, i.thumbnail, c.colorid as ccolorid, " +
        "i.colorid as icolorid, c.color, c.pcolor " +
        "from products p " +
        "inner join images i on p.prodid = i.prodid " +
        "inner join colors c on p.prodid = c.prodid" +
        "<where>" +
        "<if test='keyword != null'>keyword like concat('%', #{keyword}, '%')</if>" +
        "and ((c.pcolor = 'Y' and i.colorid = c.colorid and i.thumbnail = 'Y') or (i.colorid is null and i.thumbnail = 'Y'))" +
        "</where>" +
        "order by p.prodid desc " +
        "</script>")
```

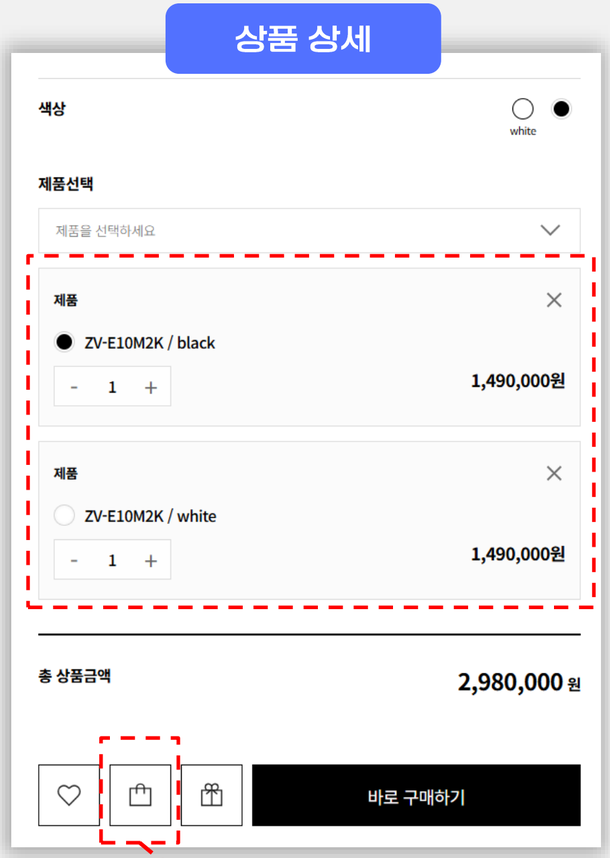
4-2 기능설명 - 장바구니 담기

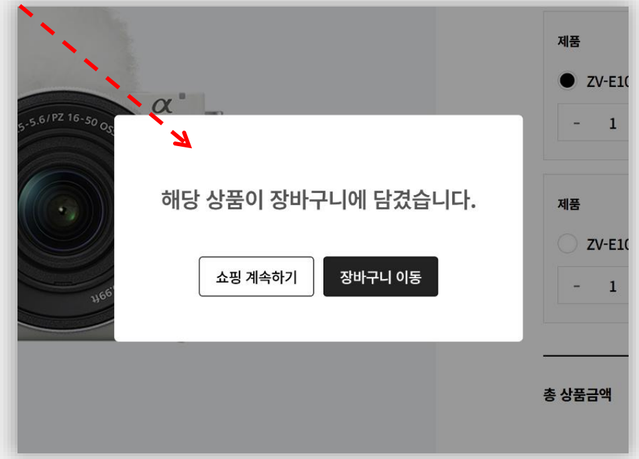
상품 상세 페이지에서 선택한 제품을 장바구니에 담기.



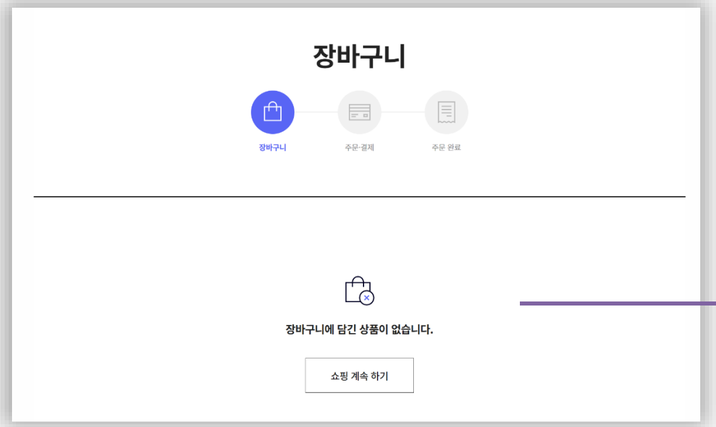
장바구니에 중복된 상품이 있는지 조회

장바구니 테이블에 해당상품의 데이터가 없으면 장바구니에 데이터를 추가.
있으면 데이터의 수량을 증가시킴.





4-2 기능설명 - 장바구니 조회

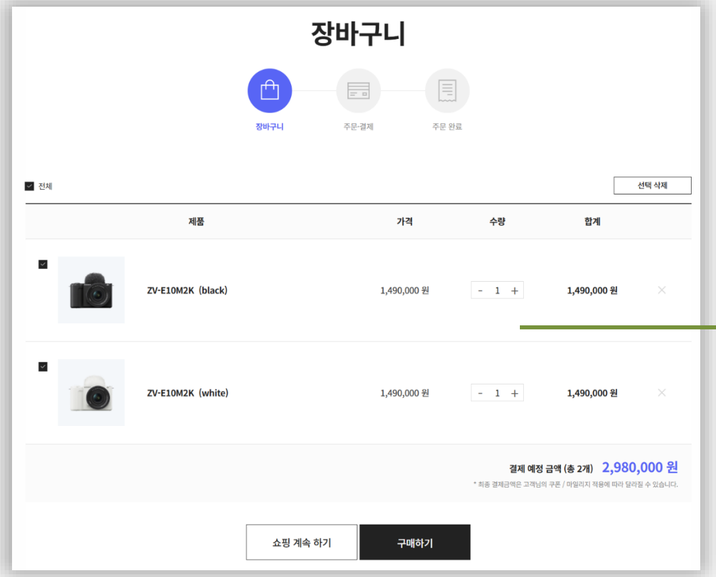


cart.html

```
<!-- 장바구니 데이터 없을 때 -->
<!-- 장바구니에 담긴 상품이 없습니다 -->

<div class="empty_cart_box" th:if="${session.memberInfo == null or carts==null or carts.size()==0}">
    <i class="empty_ico"></i>
    <p class="emptyinfo_tit">장바구니에 담긴 상품이 없습니다.</p>
    <div class="btn_wrap">
        <a class="continue btn_200_64 bg_white" th:href="@{/}"> 쇼핑 계속 하기 </a>
    </div>
</div>
```

- 로그인 되어있지 않거나,

- 로그인 된 상태에서
  회원의 장바구니 테이블의 데이터가 없다면
  “장바구니에 담긴 상품이 없습니다“를 표시.



CartMapper.java

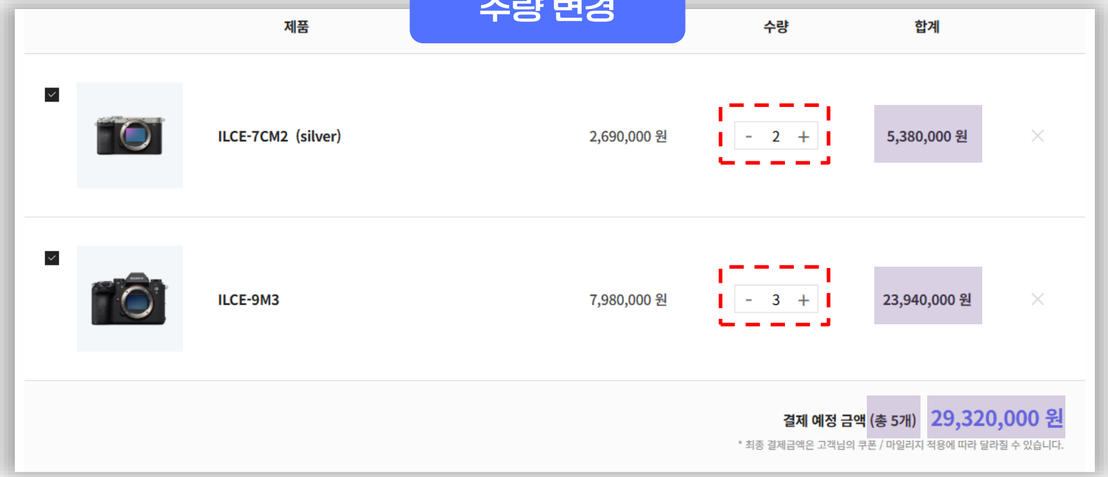
```
/**
 * 장바구니 목록을 조회한다
 * @param input - 조회할 장바구니 정보에 대한 모델 객체
 * @return 조회된 장바구니 목록
 */
@Select (
    "SELECT cartid, memberid, c.prodid, title, i.colorid, \n" +
    "c.color, filepath, price, count, price*count AS sum \n" +
    "FROM carts c \n" +
    "INNER JOIN products p ON c.prodid = p.prodid \n" +
    "LEFT JOIN colors clr ON c.prodid = clr.prodid AND c.color = clr.color \n" +
    "LEFT JOIN images i ON \n" +
        "c.prodid = i.prodid AND (clr.colorid = i.colorid OR clr.colorid IS NULL) \n" +
        "AND i.thumbnail = 'Y' \n" +
    "WHERE memberid = #{memberid} \n" +
    "ORDER BY cartid"
)
public List<Cart> selectList(Cart input);
```

- 로그인 된 상태에서
회원의 장바구니 테이블의 데이터가 있다면
조회된 데이터를 모델을 통해 뷰로 전달.

← 상품 관련 데이터베이스 테이블을 조인하여
장바구니 정보를 조회하는 SQL문.

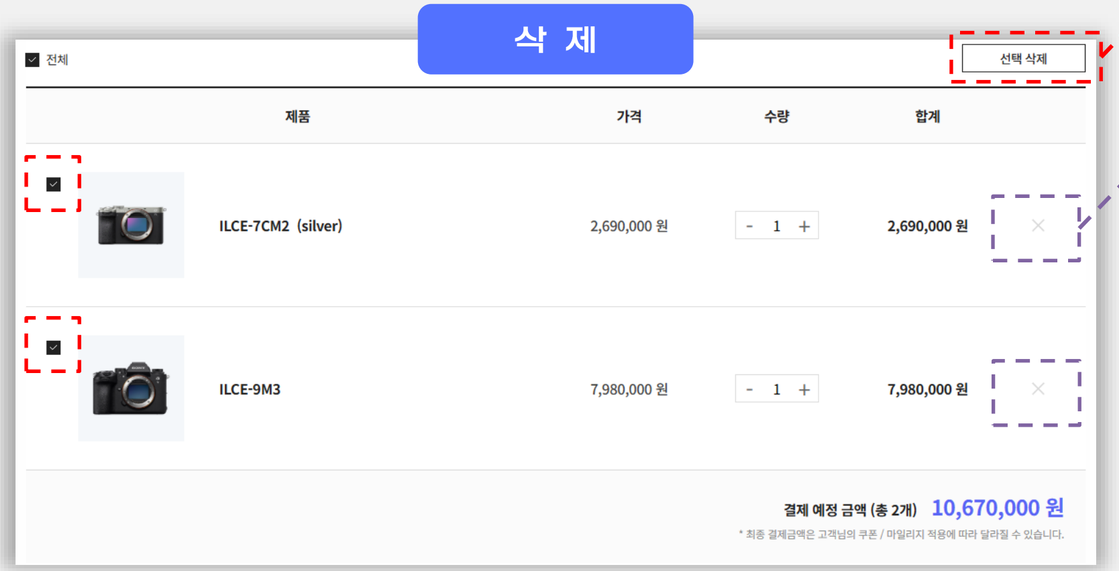
4-2 기능설명 - 장바구니 수량 변경 / 삭제

수량 변경



- "-" 또는 "+" 버튼 클릭 시, 장바구니 테이블의 데이터 수량을 변경. (PUT요청)
- 버튼 클릭 시, 화면의 '수량', '합계', '총 개수', '결제 예정 금액'을 변경.

삭 제



< 다중 삭제 >

삭제 하고 싶은 상품을 체크하여 장바구니에서 "선택 삭제" .

< 단일 삭제 >

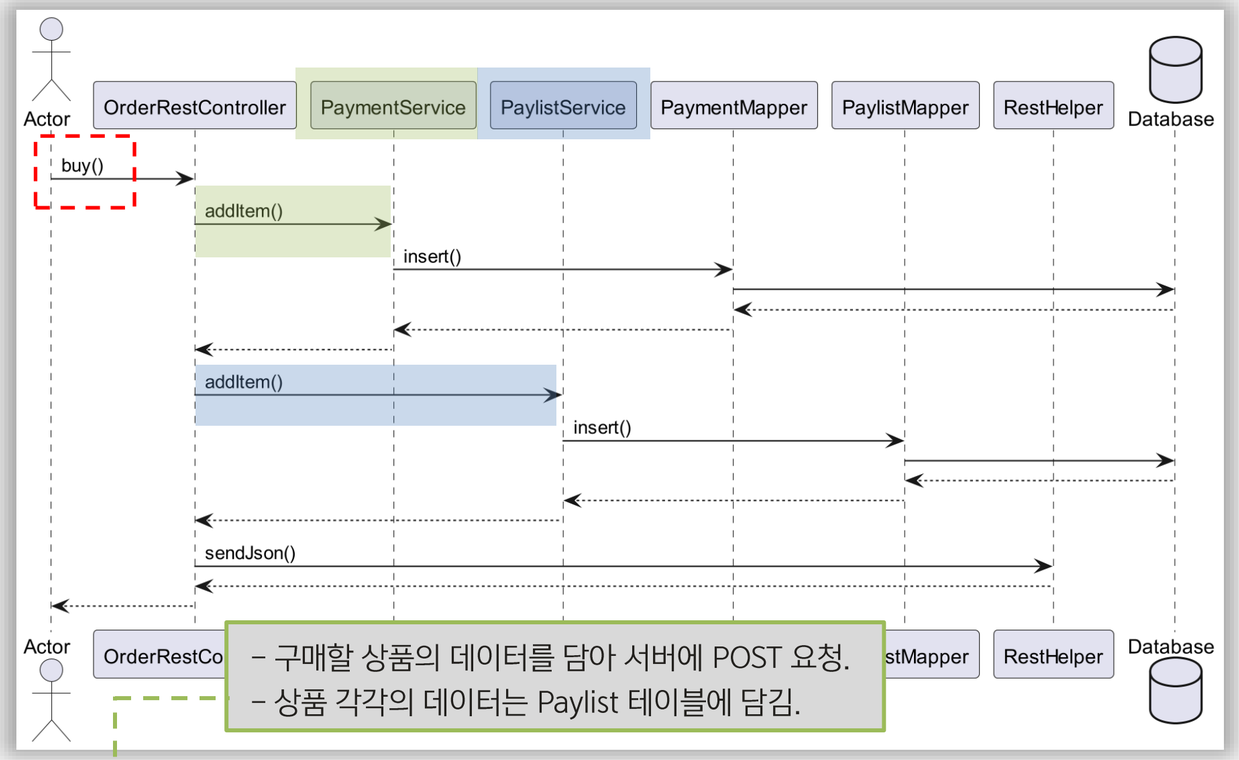
"X" 버튼 클릭 시, 장바구니에서 삭제.

↓ 장바구니 DB의 PK(cartid)로 반복을 돌려 다중 삭제하는 SQL문.

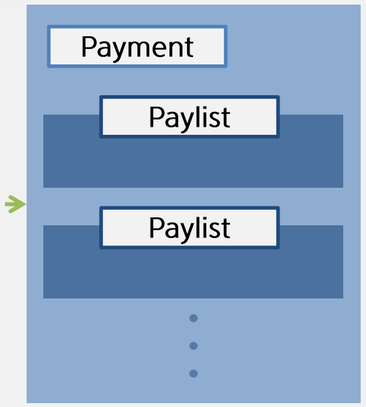
CartMapper.java

```
@Delete (
    "<script> \n" +
    "DELETE FROM carts WHERE cartid IN \n" +
    "<foreach item='cartid' collection='cartidList' open='(' separator=',' close=')' > \n" +
    "#{cartid} \n" +
    "</foreach> \n" +
    "</script>" )
public int deleteList(@Param("cartidList") List<Integer> cartidList);
```

4-2 기능설명 - 구매하기



- 구매할 상품의 데이터를 담아 서버에 POST 요청.
- 상품 각각의 데이터는 Paylist 테이블에 담김.



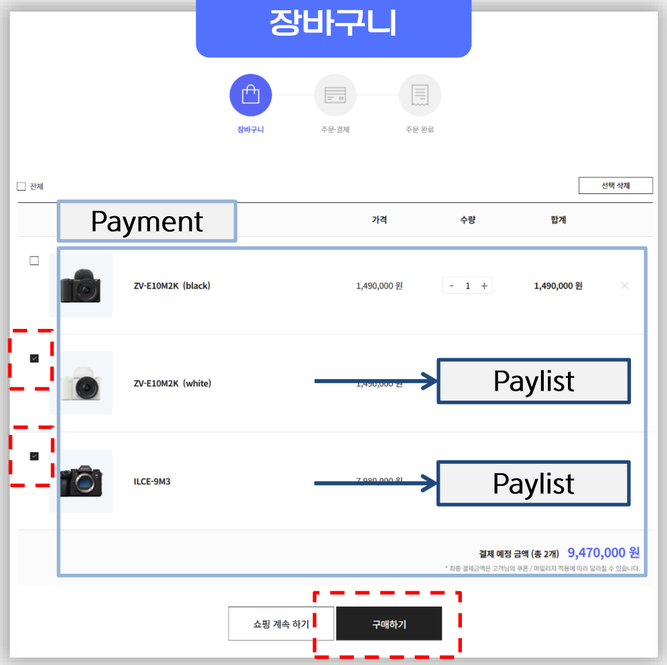
결제 DB

- Payment : 결제 테이블
- Paylist : 결제 제품 목록 테이블

<Payment 와 Paylist 테이블의 구조>

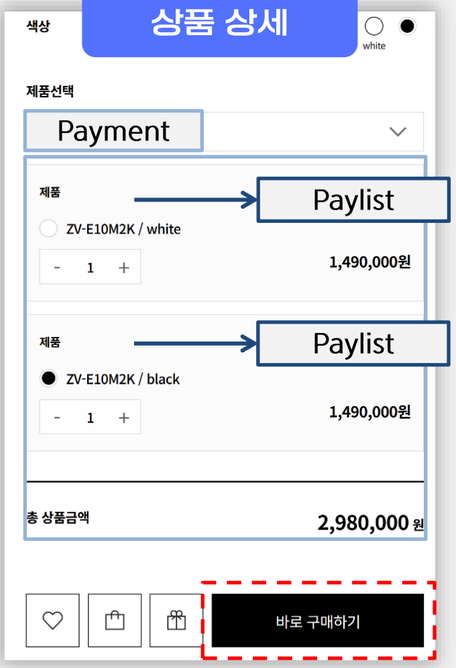
- Payment : Paylist = 1 : N

장바구니



체크한 제품들만 결제 테이블에 추가.

상품 상세



선택한 제품의 데이터를
결제 테이블에 추가.

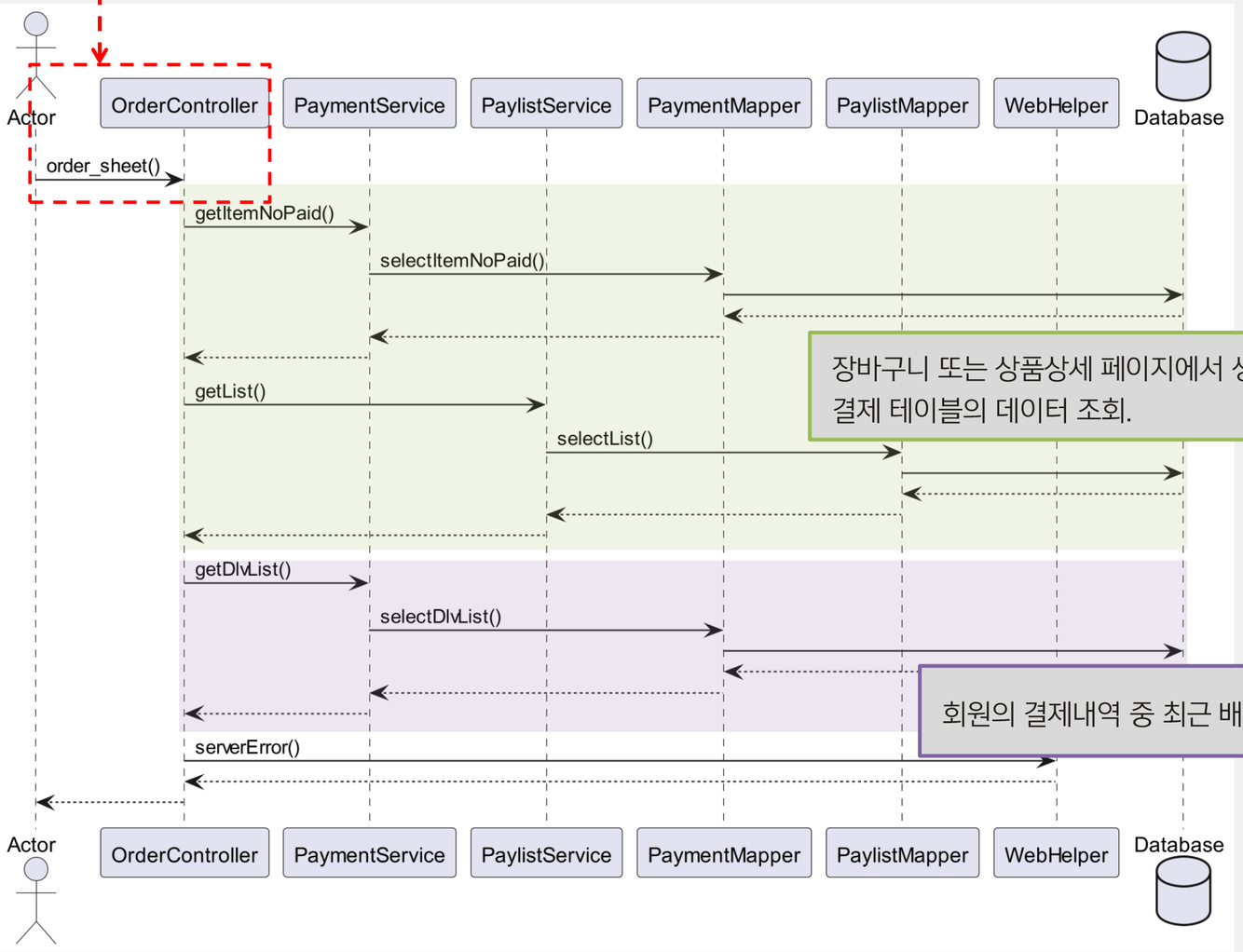
```
const cartids = data.cartids.map( v => `cartid=${v}`).join('&');
window.location = `/order/sheet?orderSheetNo=${data.item.payid}&${cartids}`;
```

결제 후, 장바구니의 데이터를 삭제하기 위해 PK인 cartid를 쿼리스트링으로 전달

```
localhost:8080/order/sheet?orderSheetNo=73&cartid=46&cartid=49
```

4-2 기능설명 - 결제페이지 (1)

주문·결제 페이지를 구성하는데 필요한 데이터를 조회하여 모델에 추가하고 해당 페이지로 이동.



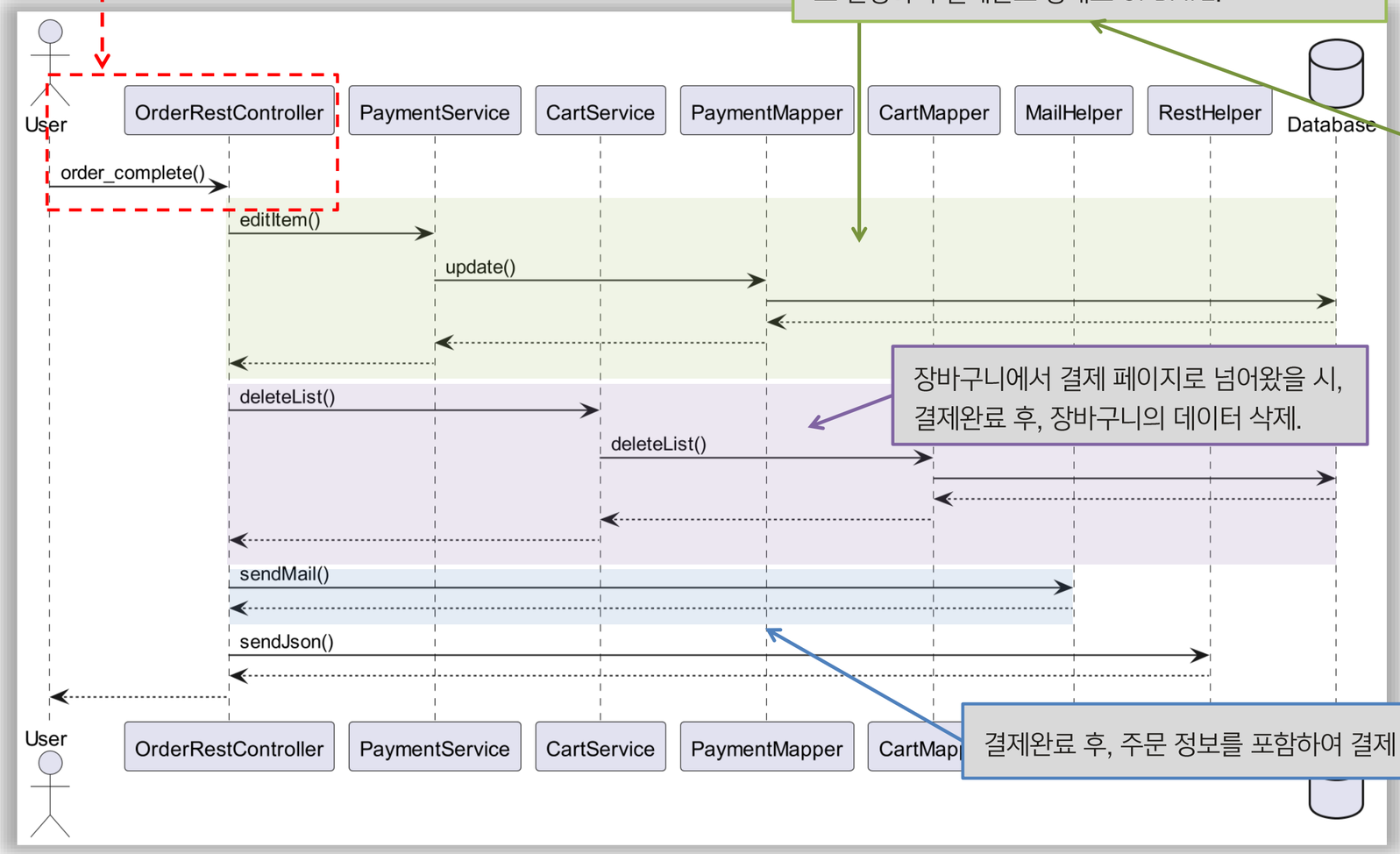
장바구니 또는 상품상세 페이지에서 생성된 결제 테이블의 데이터 조회.

회원의 결제내역 중 최근 배송지 5개까지 조회.

4-2 기능설명 - 결제페이지 (2)

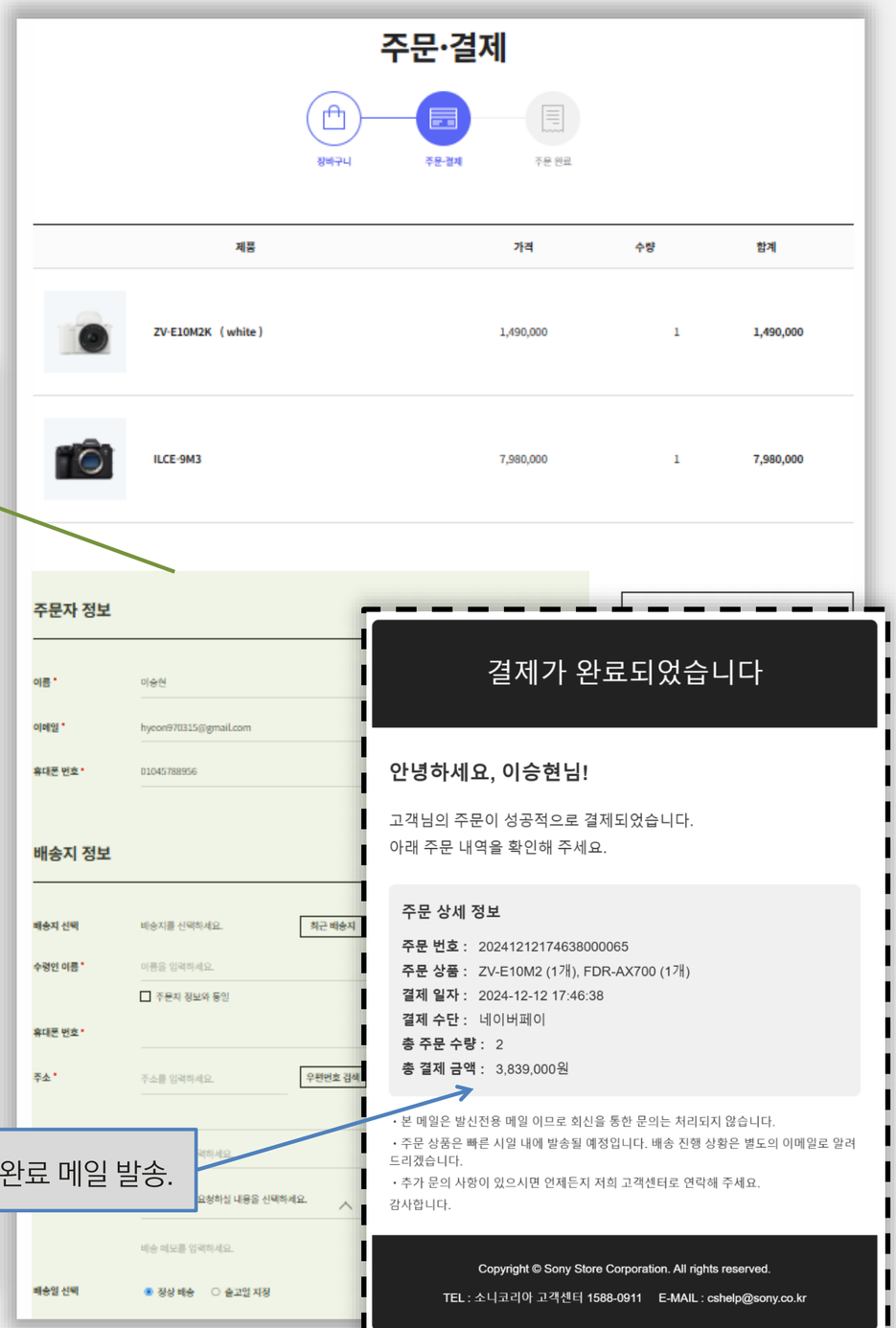
결제 하기 기능의 RestController

결제 테이블의 데이터를
주문·결제 페이지에서 입력받은 주문 및 배송 정보와
status="결제완료", paycheck="Y", date=NOW()
로 변경하여 결제완료 상태로 UPDATE.

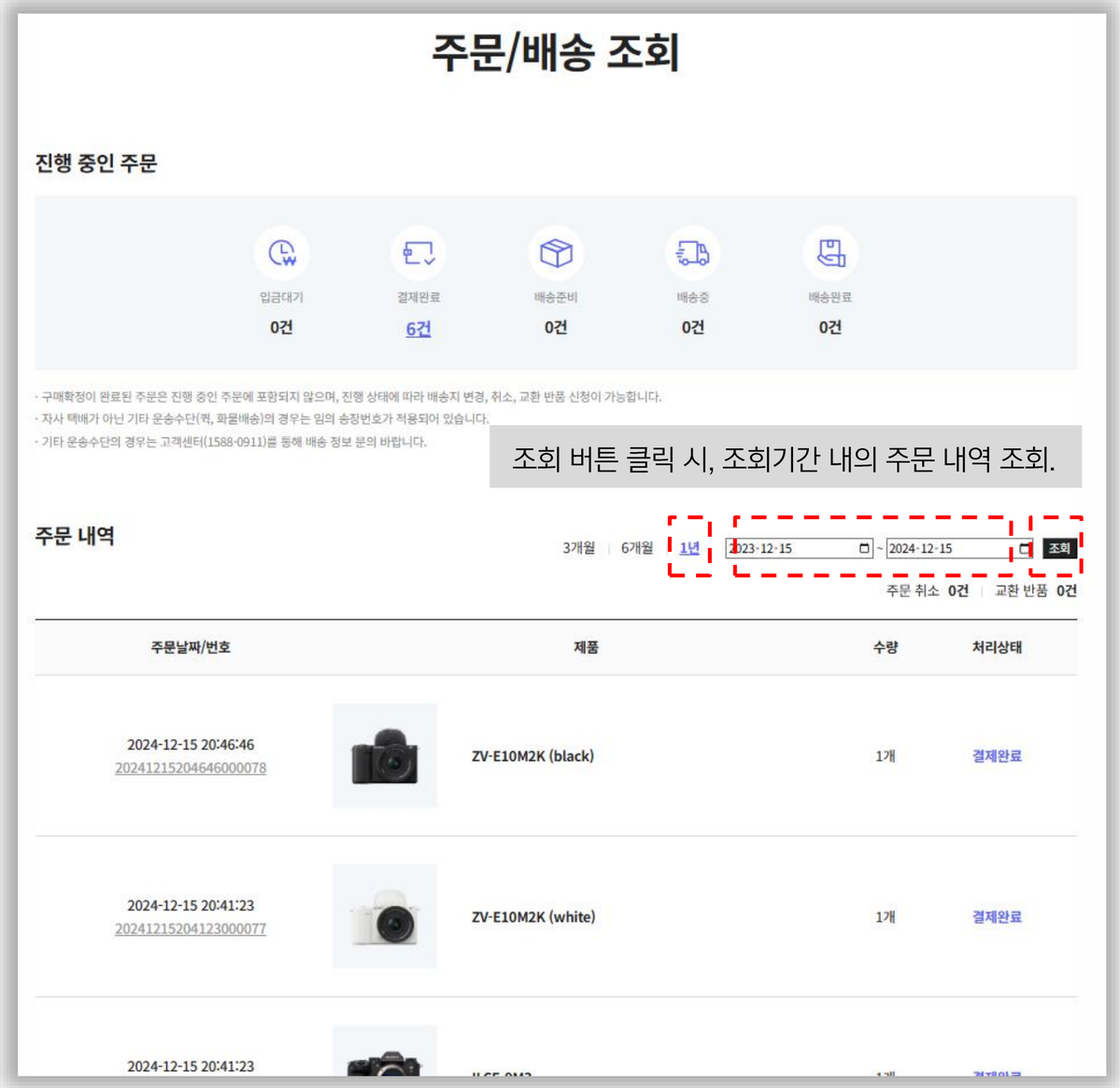


장바구니에서 결제 페이지로 넘어왔을 시,
결제완료 후, 장바구니의 데이터 삭제.

결제완료 후, 주문 정보를 포함하여 결제 완료 메일 발송.



4-2 기능설명 - 주문 조회



조회 버튼 클릭 시, 조회기간 내의 주문 내역 조회.

order_list.html – JavaScript

```javascript
document.addEventListener('DOMContentLoaded', () => {
    document.querySelector('.term.year').click();
    document.querySelector('.search').click();
} );

const colTableWrap = document.querySelector('.col_table_wrap.order_list.order_sheet_list');
const colTable = document.querySelector('.col_table');
const colTableBody = document.querySelector('.col_table_body.order_sheet_body');

document.querySelector('.search').addEventListener( 'click', async e => {
    e.preventDefault();

    const fromdate = document.querySelector('.date1').value + ' 00:00:00';
    const todate = document.querySelector('.date2').value + ' 23:59:59';
    // console.log(fromdate, todate);

    const formData = new FormData();
    formData.append('fromdate', fromdate);
    formData.append('todate', todate);

    let data = await axiosHelper.get('[[@{/api/order_list_by_date}]]', formData);
```

페이지가 완전히 로드된 후,
조회기간을 " 1년 " 으로 클릭. => "조회" 버튼 클릭.
클릭이벤트로 서버에 GET 요청을 보내고 조회된 데이터를 페이지에 렌더링.

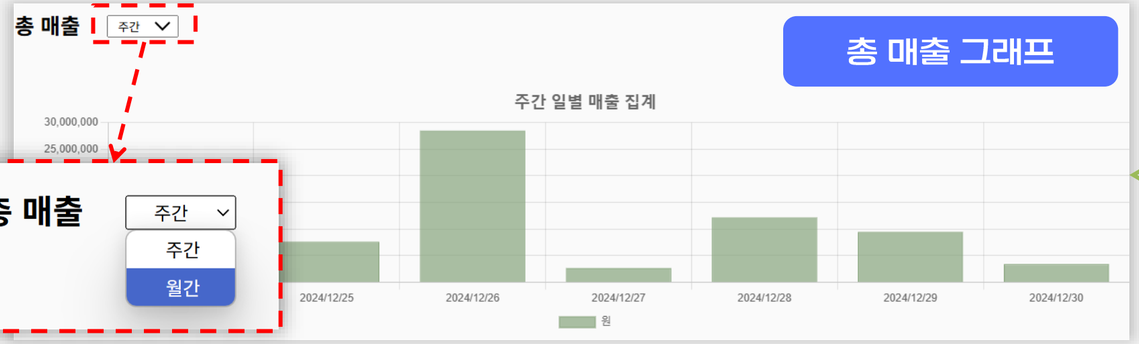
PaylistMapper.java

```java
@Select (
    "SELECT \n" +
    "pl.payid, pm.date, pm.status, \n" +
    "CONCAT( DATE_FORMAT(pm.date, '%Y%m%d%H%i%s'), LPAD(pl.payid,6,'0') ) AS orderno, \n" +
    "paylistid, prodid, prodthumbnail, prodtitle, \n" +
    "prodcolor, prodprice, count, prodprice*count AS sum \n" +
    "FROM paylist pl \n" +
    "INNER JOIN payments pm ON pl.payid = pm.payid \n" +
    "WHERE memberid = #{memberid} AND \n" +
    "pm.date BETWEEN #{fromdate} AND #{todate} \n" +
    "ORDER BY pm.date DESC"
)
@ResultMap("paylistMap")
public List<Paylist> selectListByDate(Paylist input);
```

↑ 조회기간 내의 주문내역을 조회하는 Mapper

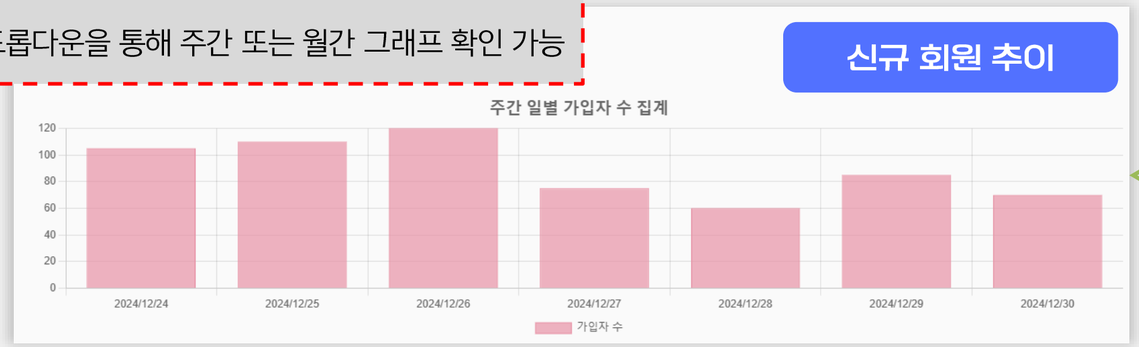
4-2 기능설명 - 대시보드 (1)

SpringBoot로 구축한 백엔드에서 데이터를 받아와 React로 대시보드 페이지 구현

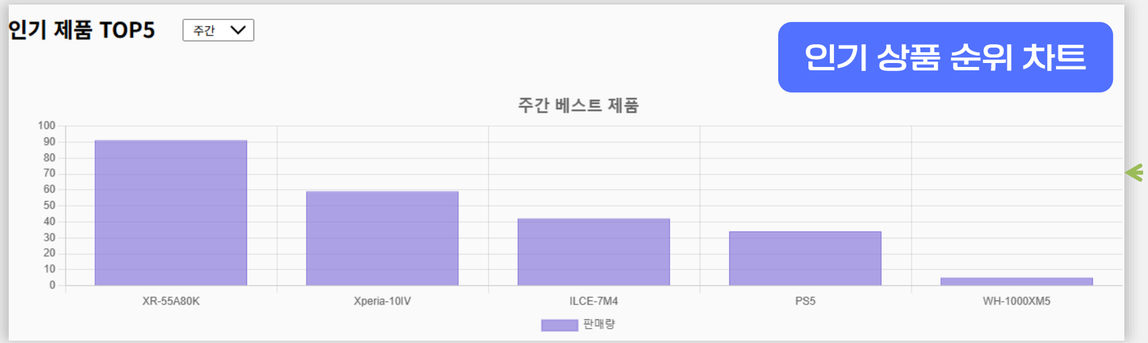


총 매출 그래프

드롭다운을 통해 주간 또는 월간 그래프 확인 가능



신규 회원 추이



인기 상품 순위 차트

매일 오전 1시에 스케줄러를 통해 전날 데이터를 집계하고,
각각의 일별 집계 DB 테이블에 데이터를 저장한다.
↓ Mapper의 데이터 삽입문

```java
/**
 * 일별 매출 집계 추가 (매출이 있을 경우)
 * @return 추가된 데이터 수
 */
@Insert(
    "INSERT INTO today_sales (date, total) \n"+
        "SELECT DATE(p.date) AS dt, SUM(p.total) AS total \n"+
            "FROM payments p \n"+
        "WHERE paycheck = 'Y' AND \n" +
            "DATE(p.date) = DATE(DATE_ADD(NOW(), INTERVAL -1 DAY)) \n"+
        "GROUP BY dt"
)
public int insert();
```

```java
// 가입인원수 집계 후 테이블에 넣기
@Insert("insert into today_member (date, count) " +
        "select DATE(regdate) date, COUNT(*) count " +
        "from members " +
        "where DATE(regdate) = DATE(DATE_ADD(NOW(), INTERVAL -1 DAY)) " +
        "group by date")
public int insertNewMemberCount();
```

```java
// 집계 결과를 테이블에 바로 넣기
@Insert("INSERT INTO today_bestproduct (title, date, cnt) " +
                "SELECT pl.prodtitle, DATE(pm.date) as dt, COUNT(*) AS cnt " +
                "FROM paylist pl " +
                "INNER JOIN payments pm ON pl.payid = pm.payid " +
                "WHERE DATE(pm.date) = DATE(DATE_ADD(NOW(), INTERVAL 0 DAY)) " +
                "GROUP BY DATE(pm.date), pl.prodtitle " +
                "ORDER BY cnt DESC " +
                "LIMIT 10")
@Options(useGeneratedKeys = true, keyProperty = "id", keyColumn = "id")
public int insert(Today_BestProduct input);
```

4-2 기능설명 - 대시보드 (2)

예시 : 총 매출

Redux Slice

```
export const getList = createAsyncThunk('SaleSlice/getList', async ( {url}, {rejectWithValue}) =>{
    let result = null;
    let args = {_sort: 'id', _order: 'desc'};

    try {
        result = await axiosHelper.get(url, args);
    } catch(err) {
        result = rejectWithValue(err);
    }

    return result;
});

const SaleSlice = reduxHelper.getDefaultSlice('SaleSlice', [getList]);

export default SaleSlice.reducer;
```

Axios를 통해 백엔드로부터 데이터 수신

React Component

```
useEffect(() => {
    const url = period === 'weekly' ? '/api/today_sales/day' : '/api/today_sales/day?day=28';
    dispatch(getList({ url }));
}, [dispatch, period]);
```

훅을 사용하여 드롭다운의 값이 변경될 때마다
"주간" 또는 "월간" 데이터 가져오기

Request URL

```
http://localhost:8080/api/today_sales/day?day=7
```

Server response

Code | Details

200

Response body


```
{
  "timestamp": "2024-12-31T11:40:48.180578300",
  "status": 200,
  "message": "OK",
  "item": [
    {
      "id": 72,
      "date": "2024/12/24",
      "total": 0
    },
    {
      "id": 109,
      "date": "2024/12/25",
      "total": 7590000
    },
    {
      "id": 127,
      "date": "2024/12/26",
```


RESTful API

백엔드로부터 받아온 집계 데이터를
React 프론트엔드로 전달

Request URL

```
http://localhost:8080/api/today_sales/day?day=28
```

Server response

Code | Details

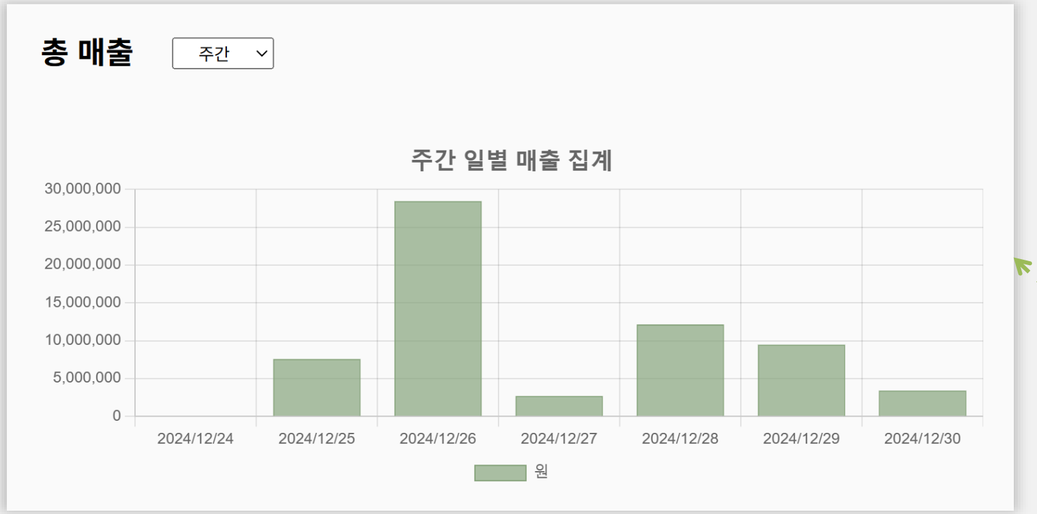
200

Response body


```
{
  "timestamp": "2024-12-31T11:47:07.892213",
  "status": 200,
  "message": "OK",
  "item": [
    {
      "id": 100,
      "date": "2024/12/03",
      "total": 9470000
    },
    {
      "id": 101,
      "date": "2024/12/04",
      "total": 0
    },
    {
      "id": 102,
      "date": "2024/12/05",
      "total": 3409000
```

4-2 기능설명 - 대시보드 (3)

예시 : 총 매출



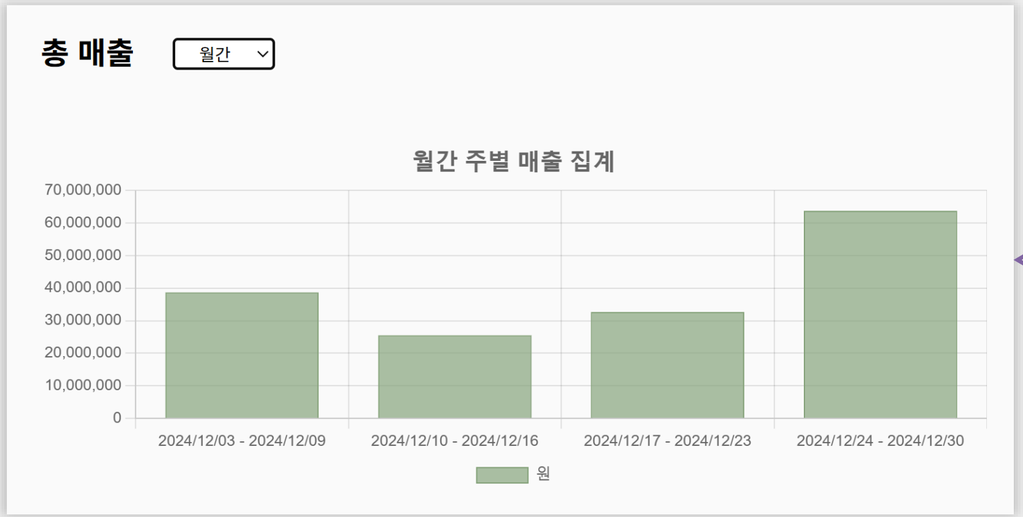


Chart.js 라이브러리를 이용한 그래프 구현

React Component

```jsx
const dispatch = useDispatch();
const {item} = useSelector( state => state.SaleSlice );
const [period, setPeriod] = useState('weekly');
const [data, setData] = useState({ keys: [], values: [] });
```

훅을 사용하여 Redux 상태를 관리

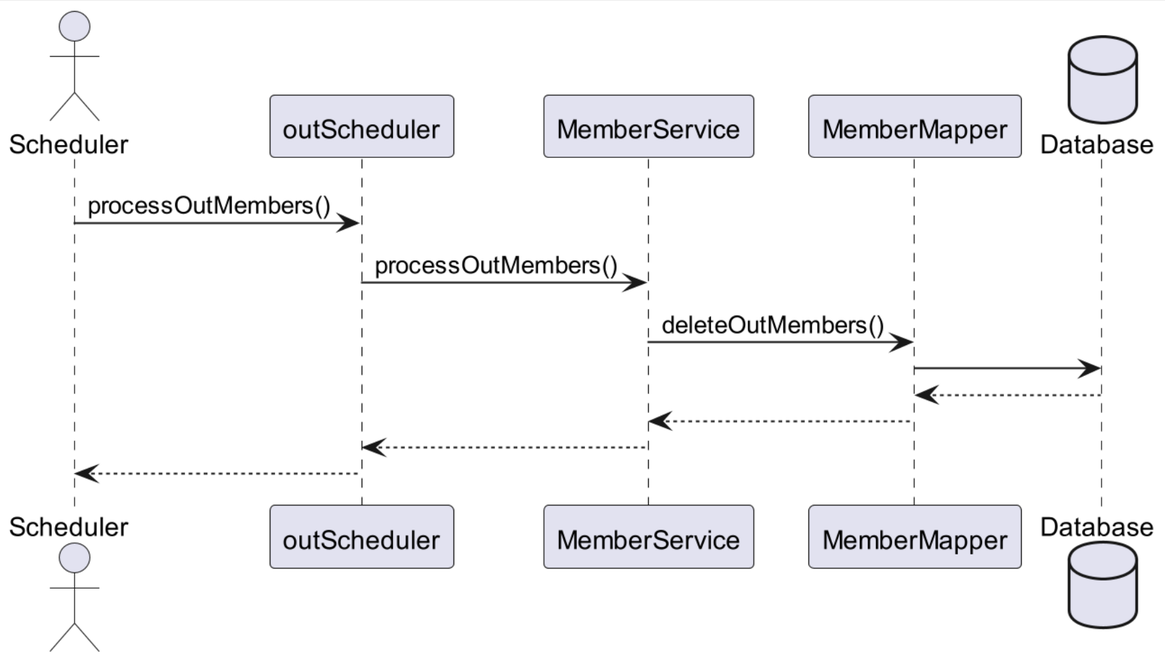
```jsx
useEffect( () => {

    if (!item) return;

    let keys = [];
    let values = [];

    if (period === 'weekly') {
        keys = item.map(v => v.date);
        values = item.map(v => v.total);
    } else {
        const weeks = [];
        for (let i=0; i<item.length; i+=7) {
            weeks.push( item.slice( i, i+7 ) );
        }

        const weeklyData = weeks.map( week => week.reduce( (acc, cur) => acc + cur.total, 0 ) );

        keys = weeks.map( (v,i) => `${v[0].date} - ${v[v.length-1].date}`);
        values = weeklyData;
    }

    setData({ keys, values });

}, [item, period]);
```

주간, 월간 에 따라 집계 데이터를 반복문 처리하여 key와 value에 해당하는 값을 배열에 저장

4-2 기능설명 - 스케줄러 (1)

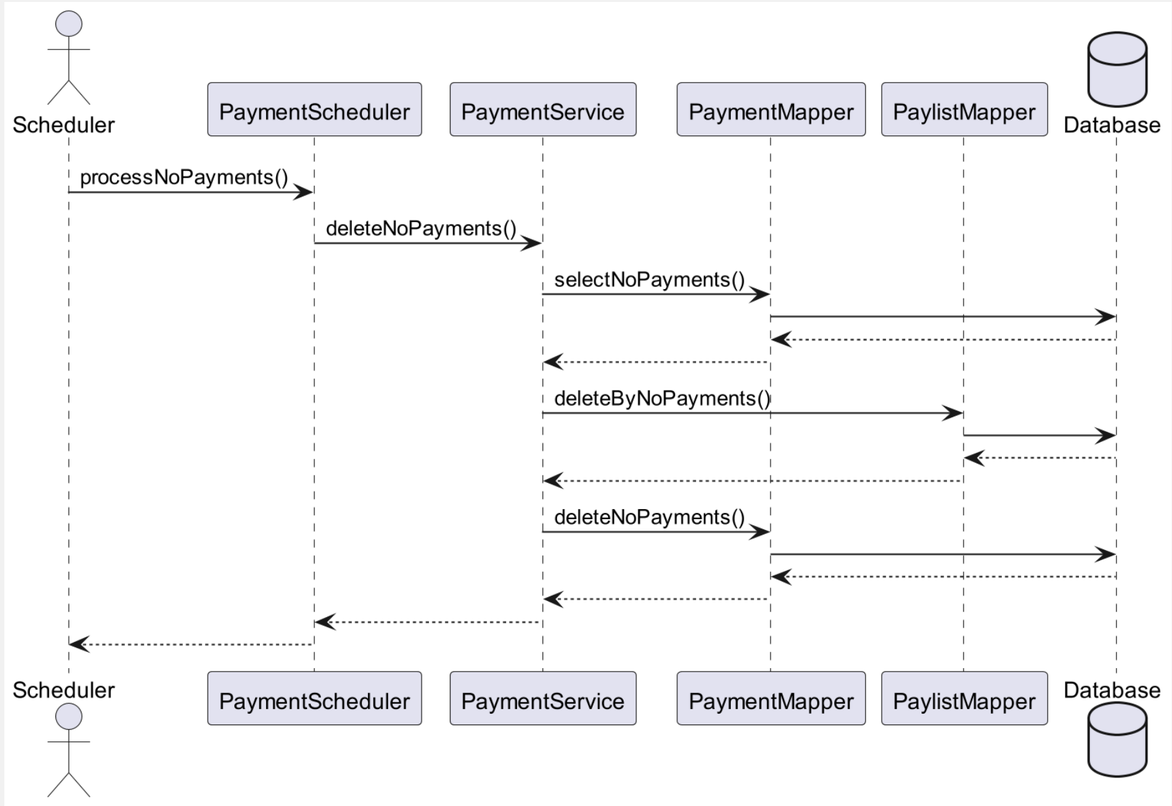
회원 탈퇴

- 회원탈퇴 시 세션의 비밀번호와 비밀번호 입력값이 일치하면 isout = “Y”로 값 수정
- 스케줄러를 이용해 매주 월요일 00시에 isout이 “Y”로 수정된 후, 7일이 지난 데이터 삭제



결제 취소

- paycheck=“N” 이고, 결제 데이터가 추가된 지 1시간이 지난 주문내역 조회
- 결제완료 하지 않고 페이지를 이동하는 등의 미결제 내역은 스케줄러로 매일 오전 4시에 데이터 삭제

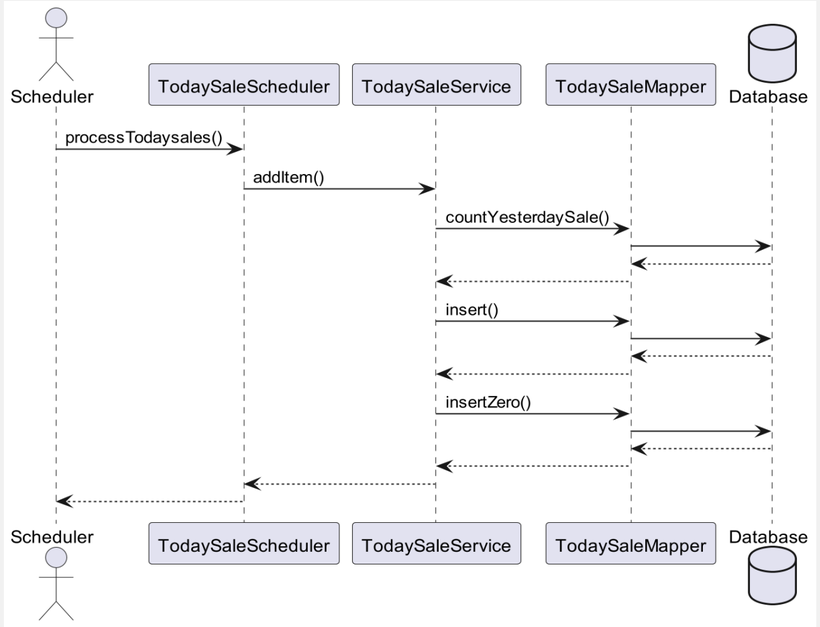


4-2 기능설명 - 스케줄러 (2)

< 관리자 대시보드 기능을 위한 데이터 INSERT >
매일 오전 1시에 스케줄러를 통해 전날 데이터를 집계하고,
각각의 일별 집계 DB 테이블에 저장

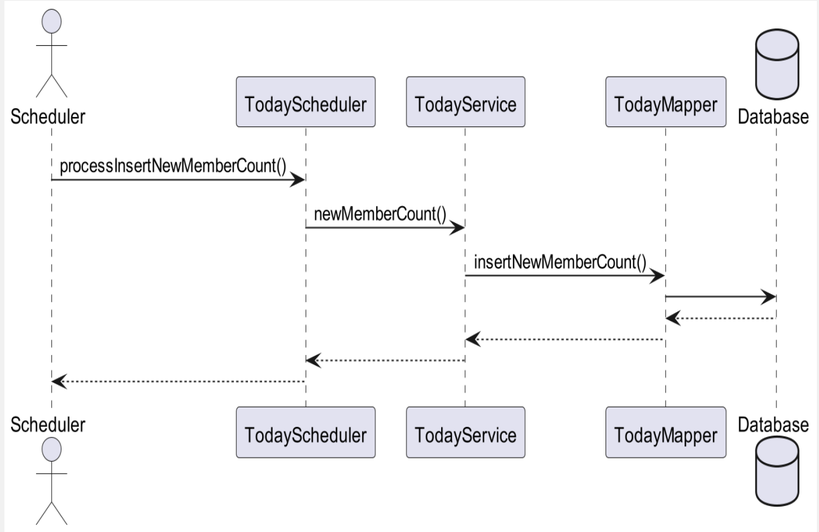
총 매출

결제테이블에서 결제일이 하루 전인 매출과 날짜를
today_sales 테이블에 저장



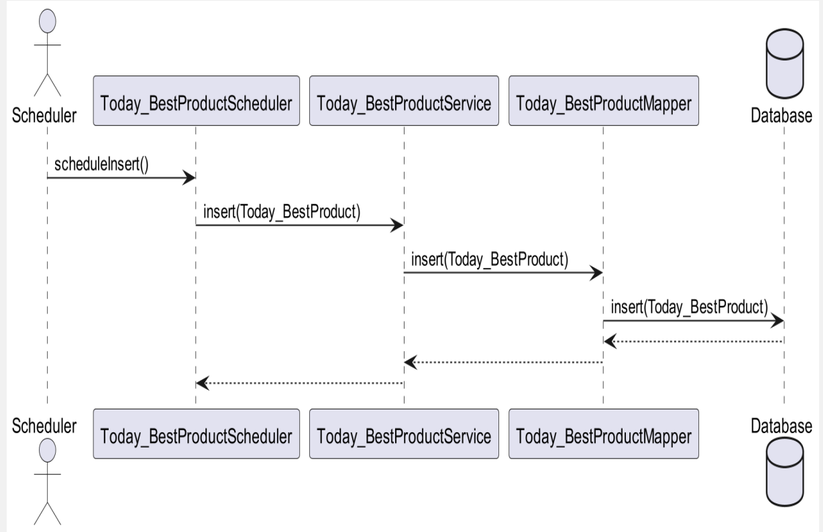
신규 가입자 수

등록 일자가 하루 전인 회원의 수와 날짜를
today_member 테이블에 저장

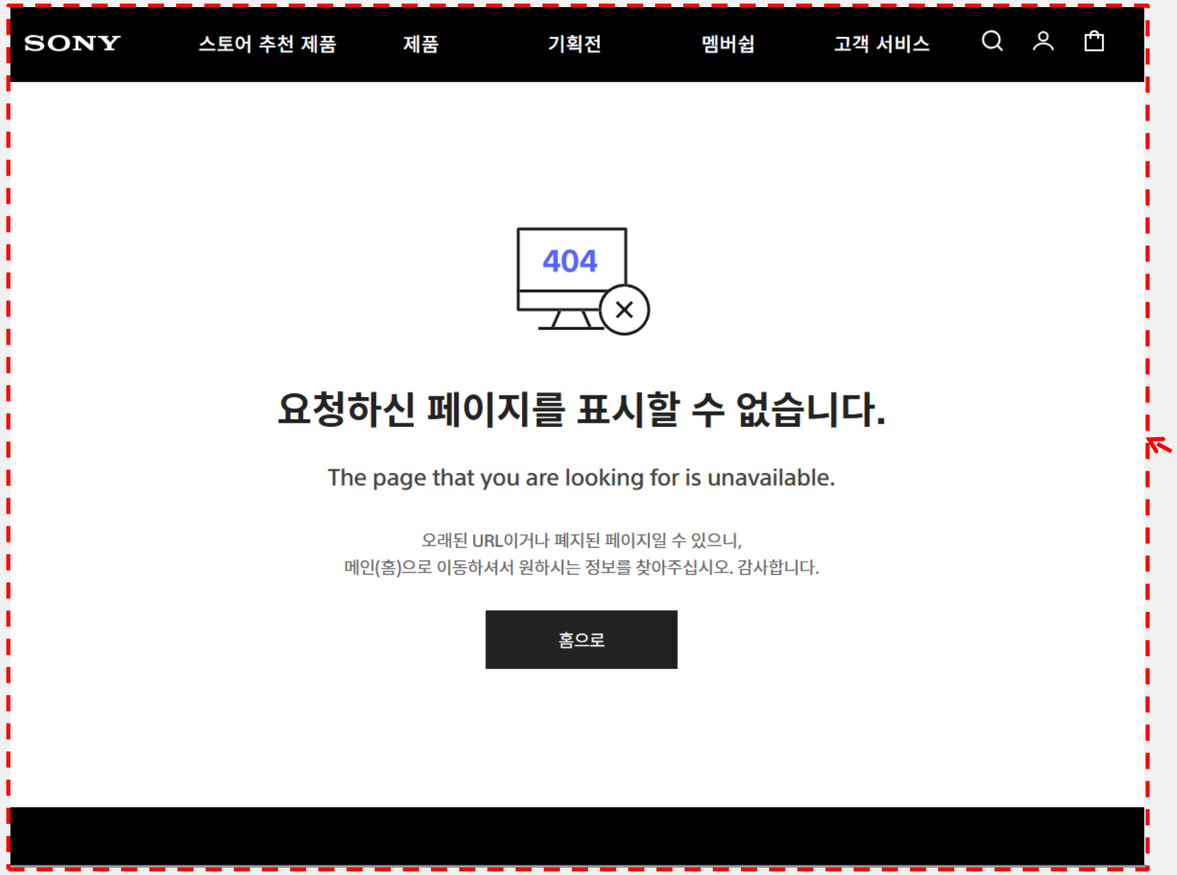


판매량

결제 테이블에서 결제일이 하루 전인 상품이름과 날짜, 수량을
today_bestproduct 테이블에 저장



4-2 기능설명 - 전역 에러 처리



CustomErrorController.java

```java
@Slf4j
@Controller
public class CustomErrorController implements ErrorController {
    private String VIEW_PATH = "/error/";

    @RequestMapping(value = "/error")
    public String handleError(HttpServletRequest request, HttpServletResponse response) {
        Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE);

        if (status != null) {
            int statusCode = Integer.valueOf(status.toString());

            if (statusCode == 404) {
                return VIEW_PATH + "error_404";
            }
            if (statusCode == 500) {
                return VIEW_PATH + "error_404";
            }
            if (statusCode == 400) {
                return VIEW_PATH + "error_404";
            }
        }
        return "error";
    }
}
```

SpringBoot 전반에서 발생하는 에러 일관되게 처리

상태 코드가 404, 500, 400일 경우 error_404.html 페이지로 연결

4-3 Restful API 명세서

SonyStore Swagger 1.0.0 OAS 3.0

/v3/api-docs

SonyStore REST API

localhost:8080/api-docs

Swagger-UI를 통해 개발한 Restful API의 목록 확인, 테스트

Order API 주문 관련 API

- PUT /api/order/complete 주문·결제 완료
- POST /api/order 주문·결제 시작
- GET /api/order_list_by_date 주문 조회

Cart API 장바구니 관련 API

- PUT /api/cart/edit 장바구니 수량 변경
- POST /api/cart/add 장바구니 추가
- DELETE /api/cart/deleteList 장바구니 상품 다중 삭제
- DELETE /api/cart/deleteItem/{cartid} 장바구니 상품 단일 삭제

Sign API 회원 관련 API

- PUT /api/sign/modify_userpw 비밀번호 수정
- PUT /api/sign/modify_name 이름 수정
- PUT /api/sign/modify_add_receive 주소 및 수신 설정 수정
- POST /api/sign/sign_up_input 회원가입
- POST /api/sign/login 로그인
- POST /api/sign/find_pw 비밀번호 찾기
- POST /api/sign/find_id_pw 아이디/비밀번호 찾기
- GET /api/sign/logout 로그아웃
- DELETE /api/sign/member_secession 회원 탈퇴

Search API 제품 검색 관련 API

- GET /api/search/search_result 제품 검색 결과 목록 조회

Product API 제품 관련 API

- GET /api/products 제품 목록 조회
- GET /api/products/{type} 타입별 제품 목록 조회
- GET /api/products/{type}/{type2} 타입2별 제품 목록 조회
- GET /api/products/{type}/{type2}/{type3} 타입3별 제품 목록 조회
- GET /api/product-view/{prodid} 제품 상세 조회
- GET /api/backgrounds 제품 배경화면 목록 조회

New Member API 주간, 월간 신규 가입자 수 API

- GET /api/week_member 주간 일별 신규 회원 조회
- GET /api/month_member 월간 주별 신규 회원 조회

Sale API 주간, 월간 총 매출 API

- GET /api/today_sales/day 총 매출 조회

Best Product API 주간, 월간 인기 상품 순위 API

- GET /api/today_best_products 인기 상품 순위 조회

프로젝트 후기

재한

데이터베이스에서 데이터를 불러와 페이지를 구현하는 게 생각보다 많이 힘들었습니다.
구현 과정에서 처음 구상과 다르게 많은 수정이 있었습니다.
이 과정에서 기획 단계의 중요성을 다시 한번 깨달았습니다.
팀원과 함께 Git을 사용하면서 다른 사람의 코드를 이해하고 제 코드와 연결 시키기 위해 많은 의사소통이 필요했던 것 같습니다.
팀워크와 전체적인 흐름을 배울 수 있는 유익하고 값진 시간이었습니다.

진수

처음으로 기획, 설계를 거쳐 실제로 웹사이트를 구현하고 이 과정을 문서화 하는 프로세스를 경험해보고자 했습니다. 역할과 작업 기간을 나누었더니 책임감이 생겼고, 작업 과정을 기술함으로써 현재 진행 상태를 파악하고 이슈가 발생했을 때 문제점을 찾아내는데 큰 도움이 되었습니다. 프로젝트를 진행하면서 스프링의 구조가 점차 이해되었고, 코드 한 줄 한 줄의 의미를 되새길 수 있어 많은 공부가 되었습니다. HTML, CSS, Javascript, Spring, Database, React 등 서로 독립적인 주제가 통합되는 과정에서 성취감을 느끼며 더욱 흥미를 느낄 수 있었습니다.

승현

이전에는 지금까지 배운 프로그래밍 언어 수준으로 과연 무엇을 할 수 있을까 막막했는데, Spring MVC 프로세스를 이해하게 되면서 프로그램의 구조와 흐름을 파악할 수 있게 되었습니다. 프로젝트를 경험해보니, 처음에는 복잡하게만 느껴졌던 MVC 로직이 여러 작업을 효과적으로 관리해 줄 수 있다는 것을 알게 되었고, 분리된 로직이 코드 간의 독립성, 결합성, 용이성을 얼마나 잘 높여주는지 확실히 느낄 수 있었습니다. 또한, 결제 프로세스를 설계하면서 제가 생각한 로직대로 맵퍼, 서비스, 컨트롤러를 한 단계 한 단계 구현해 나갈 때, 프로그래밍에 매력을 느낄 수 있었습니다. 쇼핑몰 페이지의 기능을 구현해 보고 나니, 이제 다른 사이트도 만들어보고 싶다는 생각이 들었고, 자신감과 성취감을 얻을 수 있는 유익한 시간이었습니다. 더 공부하여 코드를 효율적으로 만들 수 있는 개발자가 되고 싶습니다.

감사합니다