

WEB3 전자지갑 풀스택 개발자

# Sony Store

---

박재한 박진수 이승현

---

# 목차

CONTENT

1

## 프로젝트 개요

- 개발환경
- Gantt Chart & WBS
- 마인드맵
- 요구사항 명세서

2

## 화면구현

- 프로토타입 설계 (MockUp)
- 화면 구현

3

## 데이터베이스

- 테이블 구성 (ERD)
- 테이블 명세서

4

## 구현기능

- 클래스 다이어그램
- 구현 기능

# 소개



SONY

## 소니 스토어 홈페이지 클론코딩

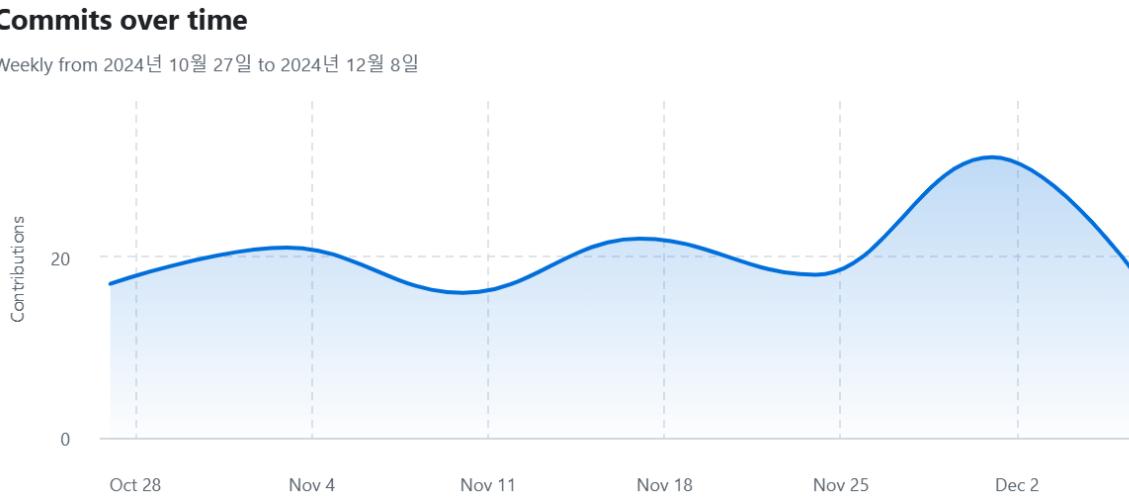
쇼핑몰의 핵심적인 사용자 경험(UX)을 고려하여 웹 페이지를 설계하고 구현하였습니다.  
이를 통해 쇼핑몰의 전반적인 흐름을 이해했습니다.

## 쇼핑몰 선정 기준

- 직관적이고 명확한 구조
- 보편적으로 실무에서 사용되는 기본 기능 포함  
(회원관리, 상품 카테고리, 검색, 쇼핑몰 기본 기능 등의 구현)

# GIT

## Team



### 박진수



### 이승현



### 박재한



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

PART 1

# 프로젝트 개요

# 개발환경



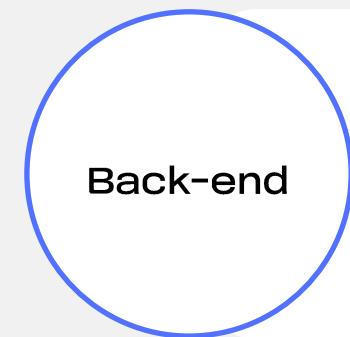
HTML5



HTML5, SCSS



Javascript



JAVA



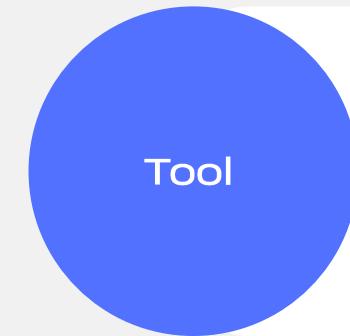
Spring Framework



MySQL



MyBatis



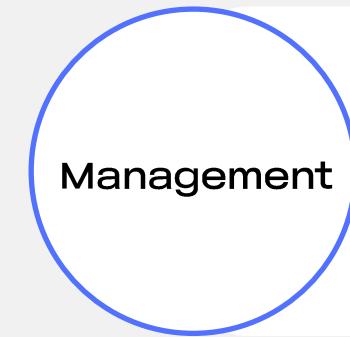
Visual Studio Code



MySQL Workbench

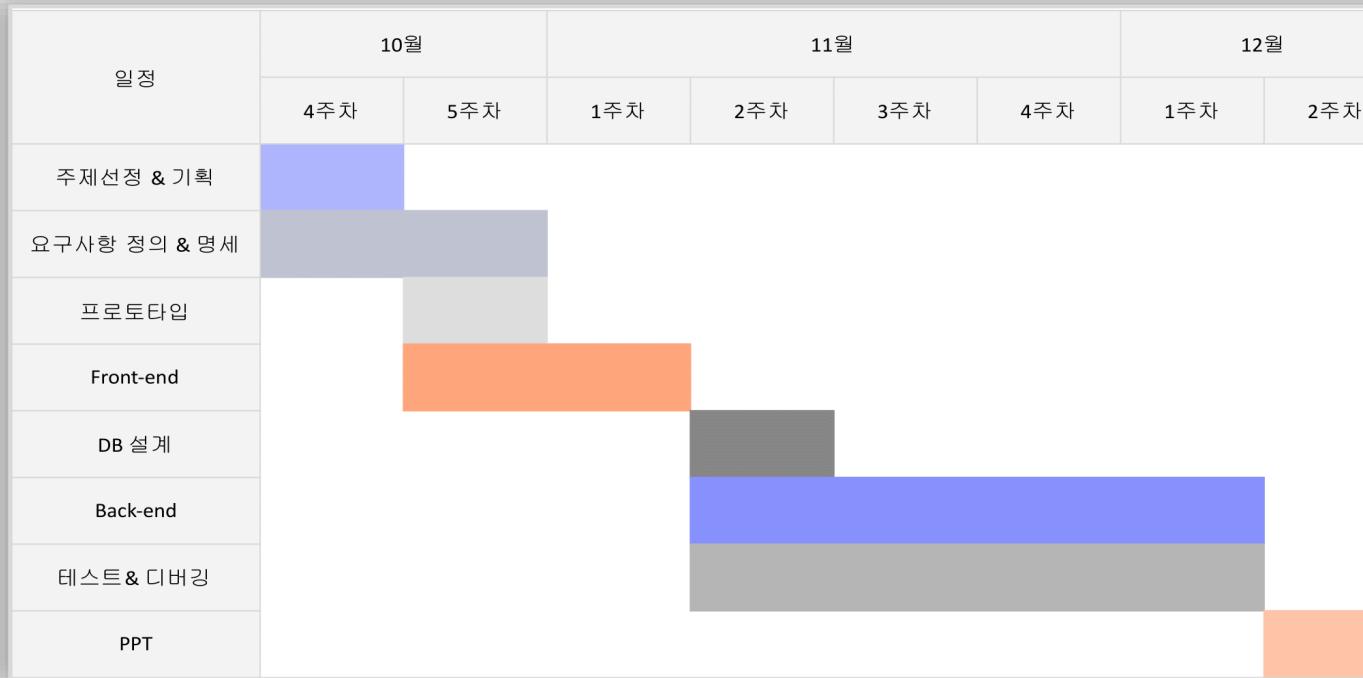


HeidiSQL



GitHub

# Gantt Chart & WBS



SONY팀 백엔드				Enter Project Range																																																																																																																																																																																																																																																																																																																																																																																																																																																														
				Start Date	End Date																																																																																																																																																																																																																																																																																																																																																																																																																																																													
				24/11/20	24/12/10																																																																																																																																																																																																																																																																																																																																																																																																																																																													
Zoom (enter 1 for Daily, 7 for Weekly)--->																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
1depth	2depth	3depth	담당자	시작일	종료일	기간	진척율 (%)	상태	수	24/11/20	24/11/21	24/11/22	24/11/23	24/11/24	24/11/25	24/11/26	24/11/27	24/11/28	24/11/29	24/11/30	24/12/01	24/12/02	24/12/03	24/12/04	24/12/05	24/12/06	24/12/07	24/12/08	24/12/09	24/12/10	24/12/11	24/12/12	24/12/13	24/12/14	24/12/15	24/12/16	24/12/17	24/12/18	24/12/19	24/12/20	24/12/21	24/12/22	24/12/23	24/12/24	24/12/25	24/12/26	24/12/27	24/12/28	24/12/29	24/12/30	24/12/31	24/12/32	24/12/33	24/12/34	24/12/35	24/12/36	24/12/37	24/12/38	24/12/39	24/12/40	24/12/41	24/12/42	24/12/43	24/12/44	24/12/45	24/12/46	24/12/47	24/12/48	24/12/49	24/12/50	24/12/51	24/12/52	24/12/53	24/12/54	24/12/55	24/12/56	24/12/57	24/12/58	24/12/59	24/12/60	24/12/61	24/12/62	24/12/63	24/12/64	24/12/65	24/12/66	24/12/67	24/12/68	24/12/69	24/12/70	24/12/71	24/12/72	24/12/73	24/12/74	24/12/75	24/12/76	24/12/77	24/12/78	24/12/79	24/12/80	24/12/81	24/12/82	24/12/83	24/12/84	24/12/85	24/12/86	24/12/87	24/12/88	24/12/89	24/12/90	24/12/91	24/12/92	24/12/93	24/12/94	24/12/95	24/12/96	24/12/97	24/12/98	24/12/99	24/12/100	24/12/101	24/12/102	24/12/103	24/12/104	24/12/105	24/12/106	24/12/107	24/12/108	24/12/109	24/12/110	24/12/111	24/12/112	24/12/113	24/12/114	24/12/115	24/12/116	24/12/117	24/12/118	24/12/119	24/12/120	24/12/121	24/12/122	24/12/123	24/12/124	24/12/125	24/12/126	24/12/127	24/12/128	24/12/129	24/12/130	24/12/131	24/12/132	24/12/133	24/12/134	24/12/135	24/12/136	24/12/137	24/12/138	24/12/139	24/12/140	24/12/141	24/12/142	24/12/143	24/12/144	24/12/145	24/12/146	24/12/147	24/12/148	24/12/149	24/12/150	24/12/151	24/12/152	24/12/153	24/12/154	24/12/155	24/12/156	24/12/157	24/12/158	24/12/159	24/12/160	24/12/161	24/12/162	24/12/163	24/12/164	24/12/165	24/12/166	24/12/167	24/12/168	24/12/169	24/12/170	24/12/171	24/12/172	24/12/173	24/12/174	24/12/175	24/12/176	24/12/177	24/12/178	24/12/179	24/12/180	24/12/181	24/12/182	24/12/183	24/12/184	24/12/185	24/12/186	24/12/187	24/12/188	24/12/189	24/12/190	24/12/191	24/12/192	24/12/193	24/12/194	24/12/195	24/12/196	24/12/197	24/12/198	24/12/199	24/12/200	24/12/201	24/12/202	24/12/203	24/12/204	24/12/205	24/12/206	24/12/207	24/12/208	24/12/209	24/12/210	24/12/211	24/12/212	24/12/213	24/12/214	24/12/215	24/12/216	24/12/217	24/12/218	24/12/219	24/12/220	24/12/221	24/12/222	24/12/223	24/12/224	24/12/225	24/12/226	24/12/227	24/12/228	24/12/229	24/12/230	24/12/231	24/12/232	24/12/233	24/12/234	24/12/235	24/12/236	24/12/237	24/12/238	24/12/239	24/12/240	24/12/241	24/12/242	24/12/243	24/12/244	24/12/245	24/12/246	24/12/247	24/12/248	24/12/249	24/12/250	24/12/251	24/12/252	24/12/253	24/12/254	24/12/255	24/12/256	24/12/257	24/12/258	24/12/259	24/12/260	24/12/261	24/12/262	24/12/263	24/12/264	24/12/265	24/12/266	24/12/267	24/12/268	24/12/269	24/12/270	24/12/271	24/12/272	24/12/273	24/12/274	24/12/275	24/12/276	24/12/277	24/12/278	24/12/279	24/12/280	24/12/281	24/12/282	24/12/283	24/12/284	24/12/285	24/12/286	24/12/287	24/12/288	24/12/289	24/12/290	24/12/291	24/12/292	24/12/293	24/12/294	24/12/295	24/12/296	24/12/297	24/12/298	24/12/299	24/12/300	24/12/301	24/12/302	24/12/303	24/12/304	24/12/305	24/12/306	24/12/307	24/12/308	24/12/309	24/12/310	24/12/311	24/12/312	24/12/313	24/12/314	24/12/315	24/12/316	24/12/317	24/12/318	24/12/319	24/12/320	24/12/321	24/12/322	24/12/323	24/12/324	24/12/325	24/12/326	24/12/327	24/12/328	24/12/329	24/12/330	24/12/331	24/12/332	24/12/333	24/12/334	24/12/335	24/12/336	24/12/337	24/12/338	24/12/339	24/12/340	24/12/341	24/12/342	24/12/343	24/12/344	24/12/345	24/12/346	24/12/347	24/12/348	24/12/349	24/12/350	24/12/351	24/12/352	24/12/353	24/12/354	24/12/355	24/12/356	24/12/357	24/12/358	24/12/359	24/12/360	24/12/361	24/12/362	24/12/363	24/12/364	24/12/365	24/12/366	24/12/367	24/12/368	24/12/369	24/12/370	24/12/371	24/12/372	24/12/373	24/12/374	24/12/375	24/12/376	24/12/377	24/12/378	24/12/379	24/12/380	24/12/381	24/12/382	24/12/383	24/12/384	24/12/385	24/12/386	24/12/387	24/12/388	24/12/389	24/12/390	24/12/391	24/12/392	24/12/393	24/12/394	24/12/395	24/12/396	24/12/397	24/12/398	24/12/399	24/12/400	24/12/401	24/12/402	24/12/403	24/12/404	24/12/405	24/12/406	24/12/407	24/12/408	24/12/409	24/12/410	24/12/411	24/12/412	24/12/413	24/12/414	24/12/415	24/12/416	24/12/417	24/12/418	24/12/419	24/12/420	24/12/421	24/12/422	24/12/423	24/12/424	24/12/425	24/12/426	24/12/427	24/12/428	24/12/429	2

# 마인드맵



# 요구사항 명세서

1Depth	2Depth	3Depth	기능내용	공통기능
카메라	렌즈 교환식 카메라	전체보기	카메라 전체를 보여줌(카메라 탭을 클릭했을 때 기본으로 실행)	
		전체보기	렌즈 교환식 카메라 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(렌즈 교환식 카메라 버튼을 클릭했을 때 기본으로 실행)	
		풀프레임	풀프레임 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		APS-C	APS-C 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
	컴팩트 카메라		컴팩트 카메라 제품만 보여줌	
		전체보기	비디오 카메라 전체를 보여줌(비디오 카메라 탭을 클릭했을 때 기본으로 실행)	
	비디오카메라	시네마 라인 카메라	시네마 라인 카메라 제품만 보여줌	
		전체보기	캠코더 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(캠코더 버튼을 클릭했을 때 기본으로 실행)	
		캠코더	4K 핸디캠 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		4K 핸드헬드	4K 핸드헬드 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		전체보기	렌즈 전체를 보여줌(렌즈 탭을 클릭했을 때 기본으로 실행)	
		풀프레임 렌즈	풀프레임 렌즈 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(풀프레임 렌즈 버튼을 클릭했을 때 기본으로 실행)	
렌즈	APS-C 렌즈	전체보기	표준 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		표준	망원 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		망원	광각 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		광각	APS-C 렌즈 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(APS-C 렌즈 버튼을 클릭했을 때 기본으로 실행)	-3Depth는 목록을 박스형태로 구현(전체보기 포함)
		표준	표준 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	-3Depth 목록중에 하위 목록이 존재한다면 탭메뉴 형식으로 하위목록 구현
		망원	망원 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		광각	광각 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
	헤드폰/이어폰	전체보기	오디오 전체를 보여줌(오디오 탭을 클릭했을 때 기본으로 실행)	-최신순, 높은가격순, 낮은가격순 정렬 버튼 구현 (최신순 정렬 버튼 기본 적용. Active시 글씨 굵게, 밝출 적용)
		전체보기	헤드폰/이어폰 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(헤드폰/이어폰 버튼을 클릭했을 때 기본으로 실행)	
		헤드폰	헤드폰 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		무선이어폰	무선이어폰 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	-카테고리별 제품의 개수 표현, 한줄에 최대 4개의 제품 보여줌(제품 이미지를 클릭하면 해당 제품으로 이동)
제품	오디오	유선이어폰	유선이어폰 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		무선 스피커	무선 스피커 제품만 보여줌	
		총오디오	총오디오 제품만 보여줌	
		전체보기	워크맨/녹음기 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(워크맨/녹음기 버튼을 클릭했을 때 기본으로 실행)	-제품 색상에 따른 색상 변경 버튼 구현 (마우스 오버시 색상에 맞는 이미지 변경. 색상이 없으면 보이지 않음)
	워크맨/녹음기	워크맨	워크맨 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		녹음기	녹음기 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		전체보기	액세서리 전체를 보여줌(액세서리 탭을 클릭했을 때 기본으로 실행)	
	액세서리	전체보기	카메라 액세서리 전체를 보여줌, 버튼 클릭 시 파란색으로 Active(카메라 액세서리 버튼을 클릭했을 때 기본으로 실행)	
		메모리카드/SSD	메모리카드/SSD 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		배터리, 충전기/어댑터	배터리, 충전기/어댑터 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		그립 액세서리	그립 액세서리 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		플래시/조명	플래시/조명 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		マイ크	マイ크 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		뷰파인더/모니터	뷰파인더/모니터 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		삼각대/리모콘	삼각대/리모콘 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		가방/케이스/스트랩	가방/케이스/스트랩 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		기타	기타 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
		필터/보호필름	필터/보호필름 제품만 보여줌, 버튼 클릭 시 파란색으로 Active	
	오디오 액세서리		오디오 액세서리 제품만 보여줌	

# 요구사항 명세서

1Depth	2Depth	3Depth	기능내용	공통기능
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	
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	
	회원 정보 찾기	로그아웃	로그아웃 (메인 페이지로 이동)	
		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	

# 요구사항 명세서

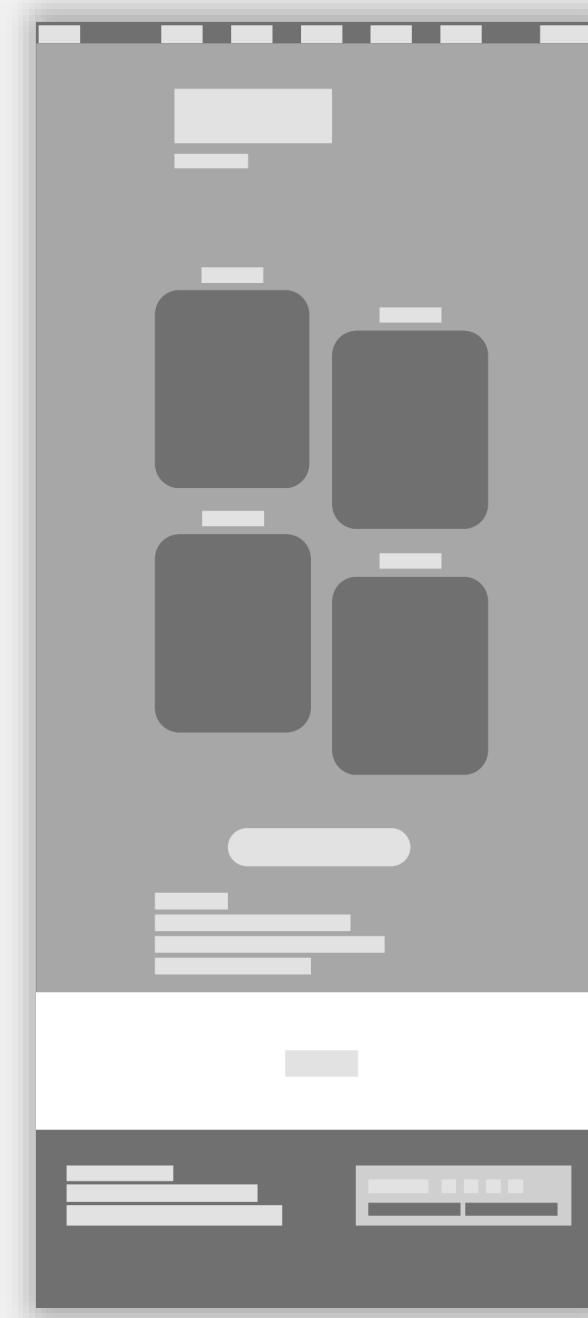
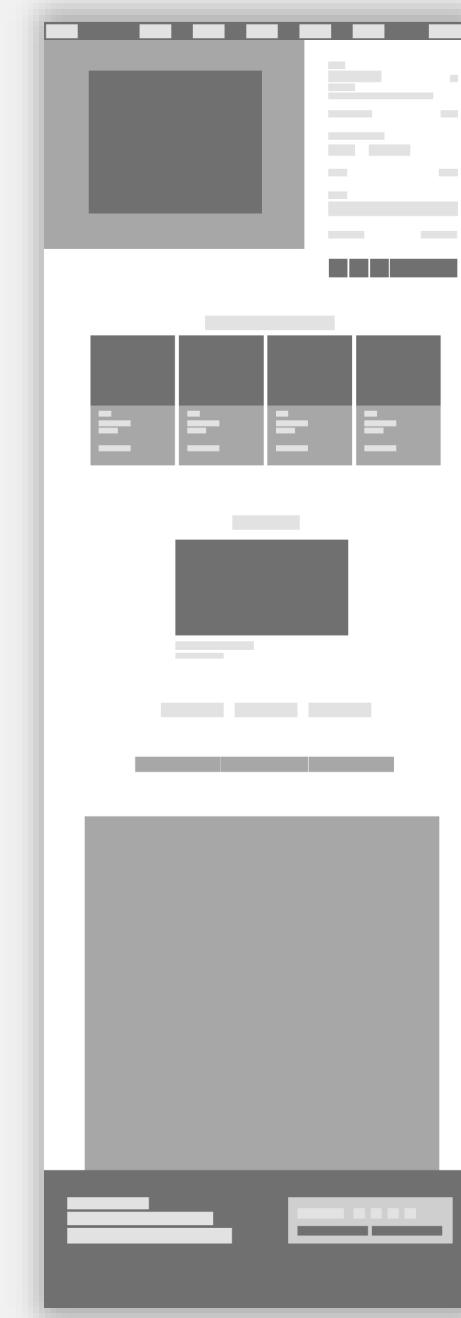
1Depth	2Depth	3Depth	기능내용	공통기능
장바구니			장바구니에 담긴 제품 내역 확인, 수량변경 기능 선택 제품 삭제 기능 결제 예정 금액, 총 수량 확인 *쇼핑 계속하기" 버튼 클릭 시, 메인 페이지로 이동 *구매하기" 버튼 클릭 시, *주문·결제" 페이지로 이동	
주문·결제	결제		주문할 제품 정보 표시 주문자 정보, 배송지 정보, 할인 정보, 결제 방법 입력 및 선택 결제 예정 금액 표시 결제 후 이메일 발송, "결제완료" 페이지로 이동	
	완료		결제 완료 안내, 주문번호 표시 *계속 쇼핑하기" 클릭 시, 메인 페이지로 이동 *주문/배송 조회" 클릭 시, *주문 배송 조회" 페이지로 이동	
검색 결과			검색 결과 키워드가 포함된 제품 목록 표시. 검색 결과가 없을 시 도움말 표시, 잘못된 검색 시 404에러페이지	
기획전		전체		
		소니스토어 단독		
		혜택존	슬라이더로 진행 중인 이벤트 표시	
		예약판매	"전체", "소니스토어 단독", "혜택존", "예약판매", "정품등록 이벤트", "LIVE ON" 탭으로 분류	
		정품등록 이벤트		
스토어 추천 제품		LIVE ON		
		추천 제품	추천 제품 이미지 표시. 마우스 호버 시, 이미지 확대 효과	
		선물제안	선물로 제안하는 제품 표시. 클릭 시, 각 제품의 "제품 상세" 페이지로 이동 *관련 기획전 바로가기" 클릭 시, "기획전" 페이지로 이동	
멤버십	등급&혜택 안내	교육 활동	교육 활동 정보 표시 *목록" 클릭 시, "기획전" 페이지로 이동	
		마일리지	마일리지 안내, 적립내역 확인, 마일리지 사용방법 표기	
		프리미엄 서비스	선물하기 서비스, 무료배송/ 친환경 패키지 서비스, 영상 변환 서비스, 멤버십 서비스 플랜 표기	탭 메뉴 형식으로 구성 (마우스 오버, 클릭 시 파란색으로 민출 Active)
하단 메뉴		쿠폰안내	발급 받을 수 있는 쿠폰을 표시	
		이용약관	소니스토어 사이트 관련 제반 서비스의 이용과 관련하여 필요한 사항을 규정안내, *개정 소니스토어 이용약관 보기" 버튼 클릭 시, 변경 전 이용약관 페이지로 이동	
		개인정보처리방침	소니 본사 페이지의 개인정보처리방침으로 이동	
		소비자 피해 보상보합	소비자 피해 보상보합 관련 내용 modal로 표시, "서비스 가입 사실 확인하기" 버튼 클릭 시, 쇼핑몰 보증 보합 사이트로 이동	
고객 서비스		사이트맵	사이트맵 페이지로 이동, GNB메뉴의 항목들만 a태그로 2Depth까지 표시.	
		직영점 안내	매장 정보 및 교통편 안내 AS센터 찾기 클릭 시, 소니 본사 페이지의 엔터 검색 페이지로 이동 *지도보기" 버튼 클릭 시, *지도접기" 버튼으로 변경하고 지도표시	
		동영상 강좌	제품의 사용설명 영상을 유튜브로 연결	
메인페이지				

PART 2

# 화면구현

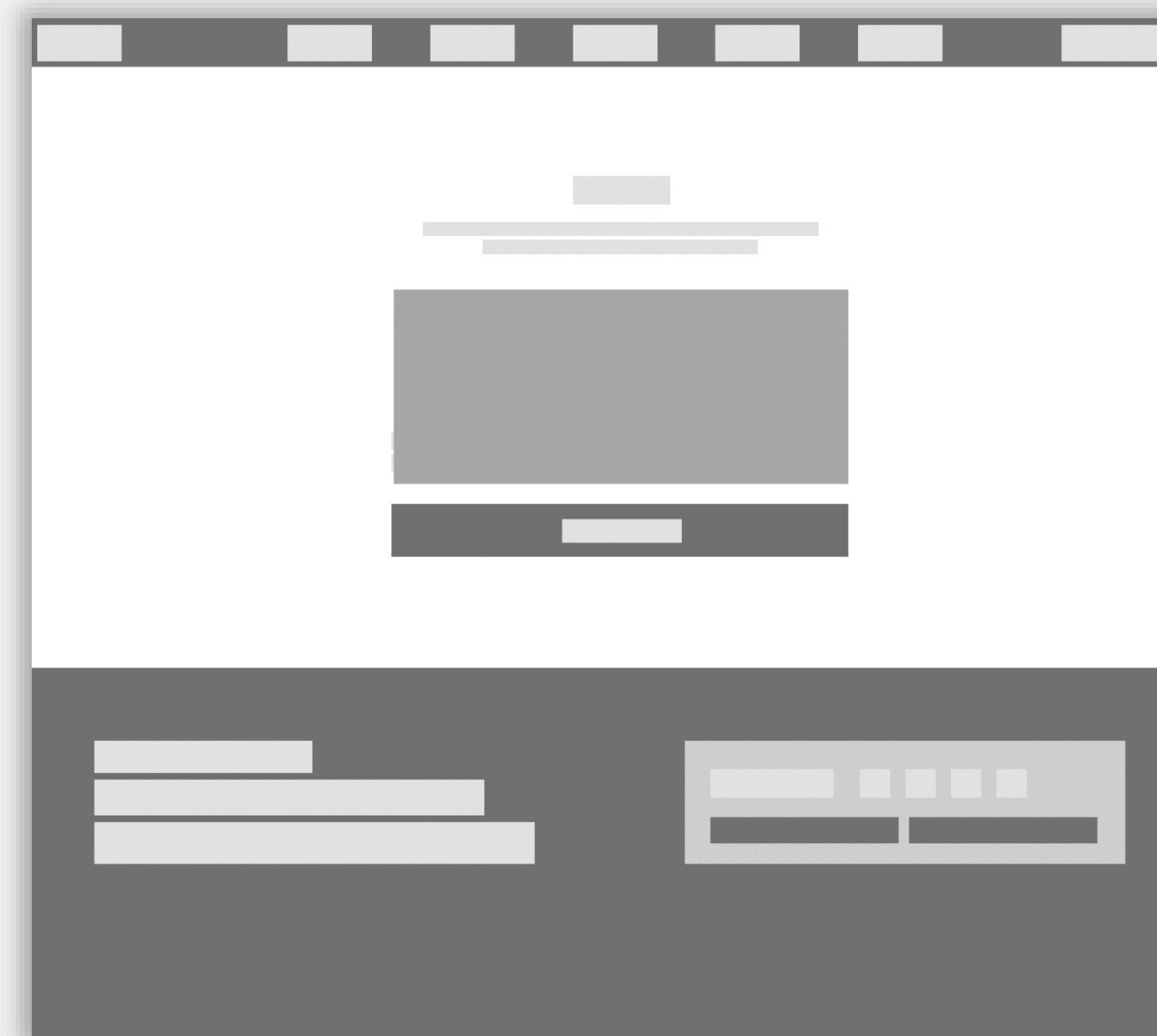
2-1

# MockUp



2-1

# MockUp



2-2

# 화면 구현

**기록의 완성**

기록의 완성

ZV-E10M2

**EVENT**

풀프레임으로 시작하자  
시작부터 FULL

WF-C510 출시 기념 프로모션

사진도, 영상도 ZV로 완벽하게

**PRODUCT**

Camera Video Camera Lens Audio

일파 아카데미 9월 수강 신청

무엇을 도와드릴까요?

카메라

전체보기 렌즈교환식 카메라 컴팩트 카메라

제품

최신순 높은가격순 낮은가격순

제품	가격
ILCE-7CM2	2,690,000원
ZV-E10M2	1,340,000원
ZV-E10M2K	1,490,000원
ILCE-9M3	7,980,000원
ZV-1F	759,000원
ZV-1	999,000원

FOLLOW US

**ZV-E10M2**

기록의 완성

1,340,000 원

회원별 마일리지 적립혜택

53,600 M MEMBERSHIP 2%

색상

제품선택

총 상품금액

바로 구매하기

**함께 구매하시면 좋은 추천 제품**

제품	가격
ZV-1	999,000원
ZV-1F	759,000원
ILCE-7CM2	2,690,000원
ZV-E10M2K	1,490,000원

**진행중인 기획전**

사진도, 영상도 ZV로 완벽하게

2024-08-12 ~ 2024-12-01

소니 일파 유니버스 소니 블로그 소니 블로그

카메라

카메라에 대한 검색결과는 총 5건입니다.

**제품**

제품	가격
ZV-1	999,000원
ZV-1F	759,000원
ILCE-7CM2	2,690,000원

**ZV-E10M2**

기록의 완성

1,340,000 원

**ZV-E10M2K**

기록의 완성

1,490,000 원

2-2

# 화면 구현

스토어 추천 제품

제품

기획전

멤버쉽

고객 서비스

# 장바구니



장바구니



주문 결제



주문 완료

전체

선택 삭제

제품	가격	수량	합계
	1,490,000 원	- 1 +	1,490,000 원
	7,980,000 원	- 1 +	7,980,000 원

**결제 예정 금액 (총 2개) 9,470,000 원**

\* 최종 결제금액은 고객님의 쿠폰 / 마일리지 적용에 따라 달라질 수 있습니다.

쇼핑 계속 하기 구매하기

장바구니 이용 안내

AS관련 제품 주의사항

인터넷 주문이 어려우세요?

| 소비자 피해 보상보험 | 사이트맵

는 바, 상업적 목적의 무단전재, 복사, 배포 등을 금합니다.

해 현금 등으로 결제 시 저희 쇼핑몰에서 기입한 구매안전서비스를 이용할 수 있습니다.

기 확인 >

금융으로 10 원아이피씨 24F 사업자 등록번호 : 106-81-23810 등신판매번호 : 2012-서울영등포-1038 소니코리아주

책임자 : Okura Kikuo TEL : 소니코리아 고객센터 1588-0911 E-MAIL : cshelf@sony.co.kr

All rights reserved.

FOLLOW US

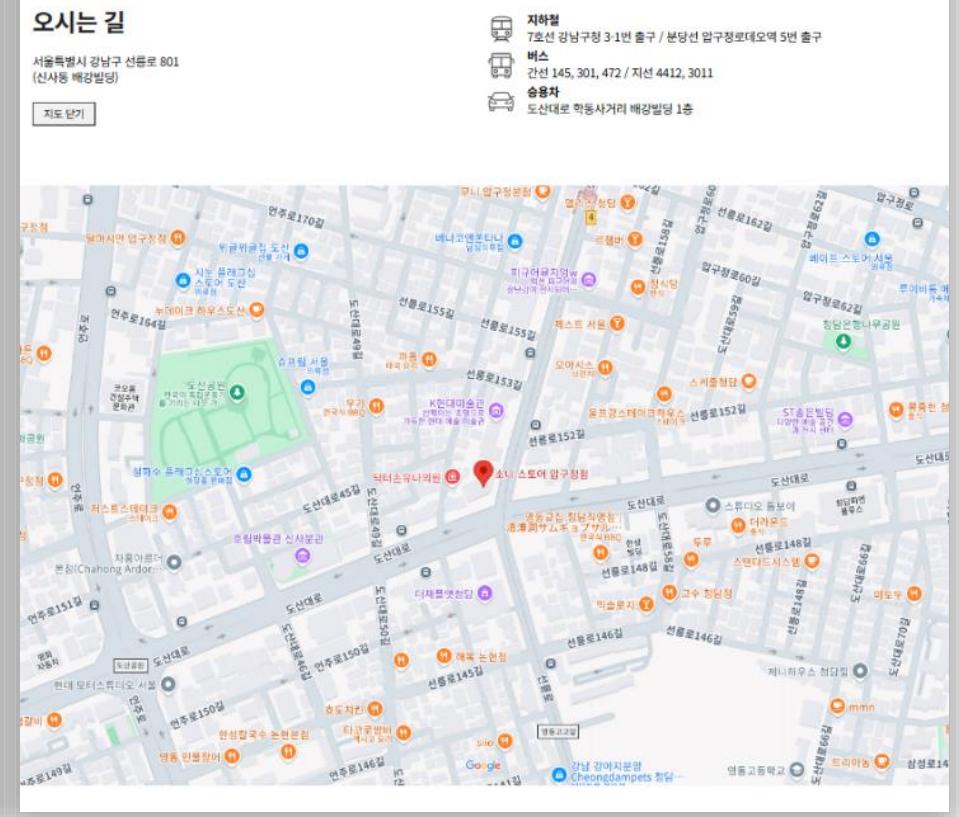
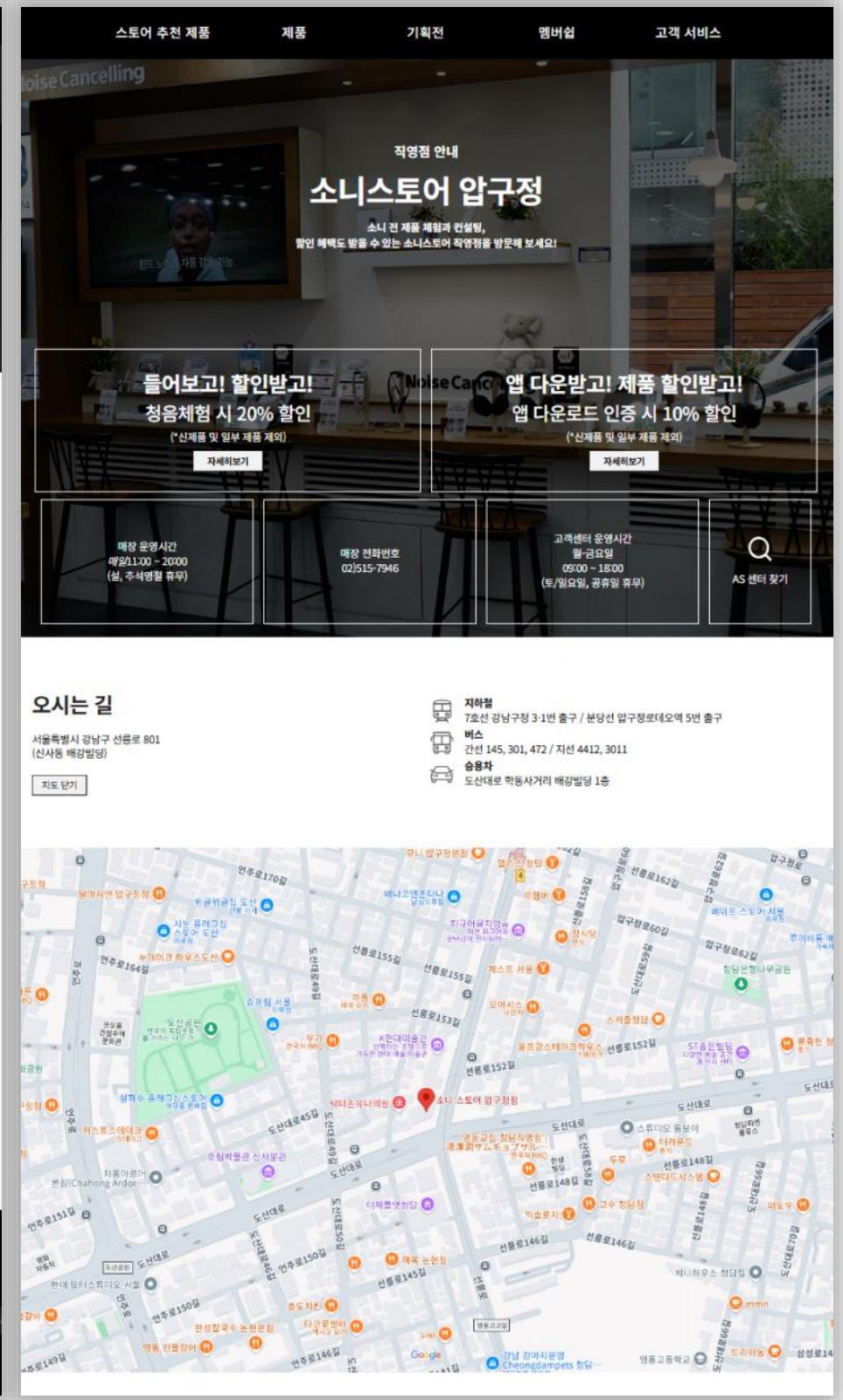
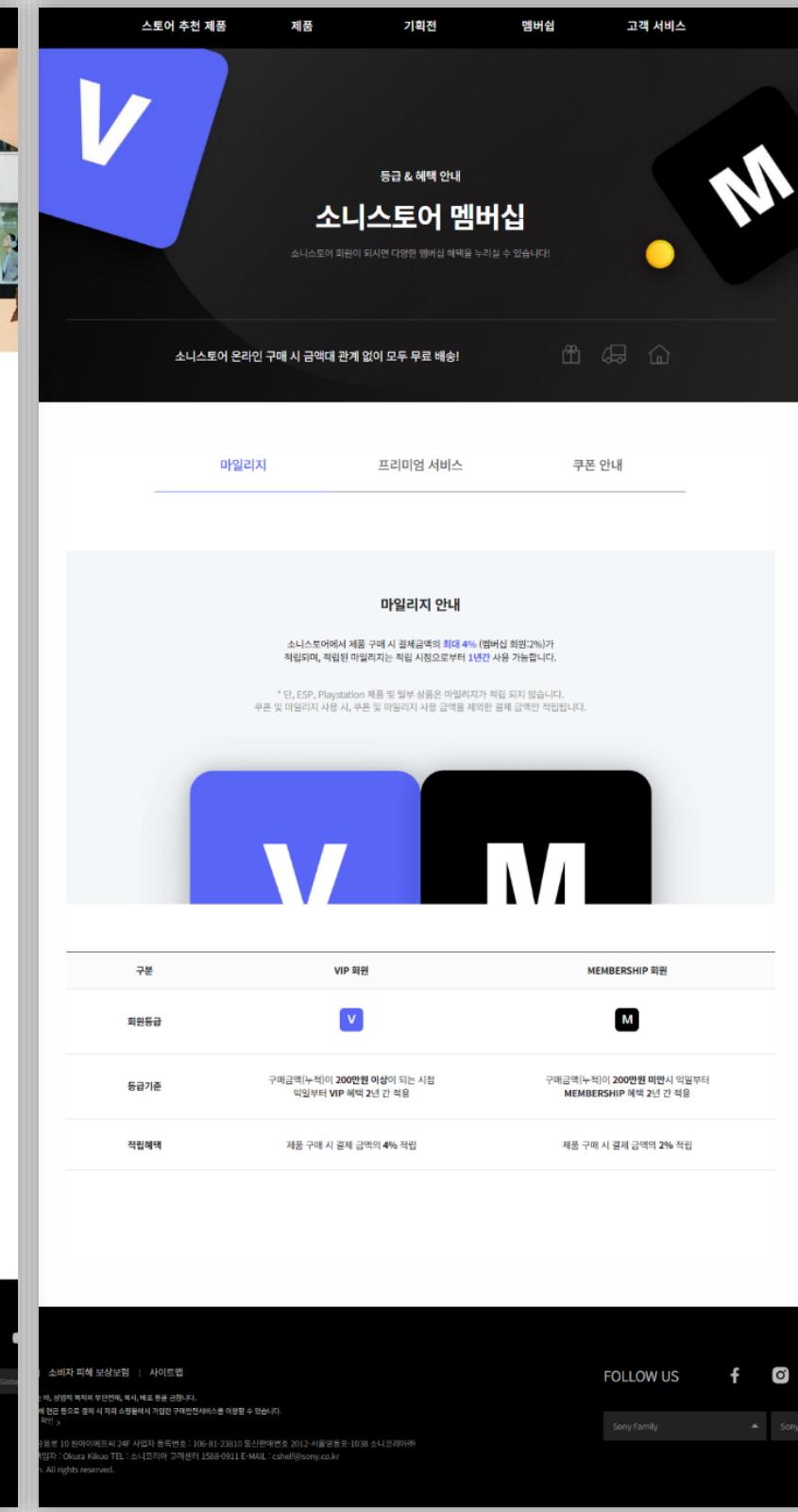
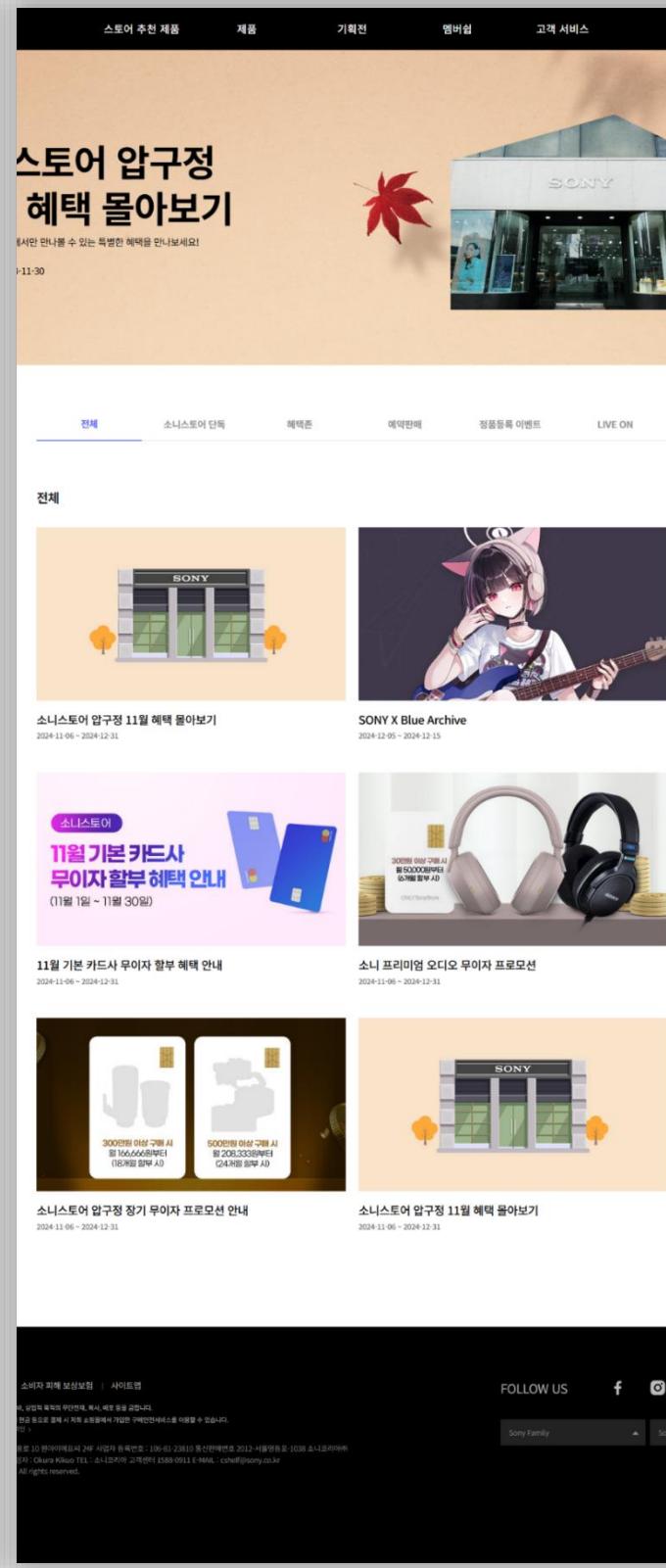
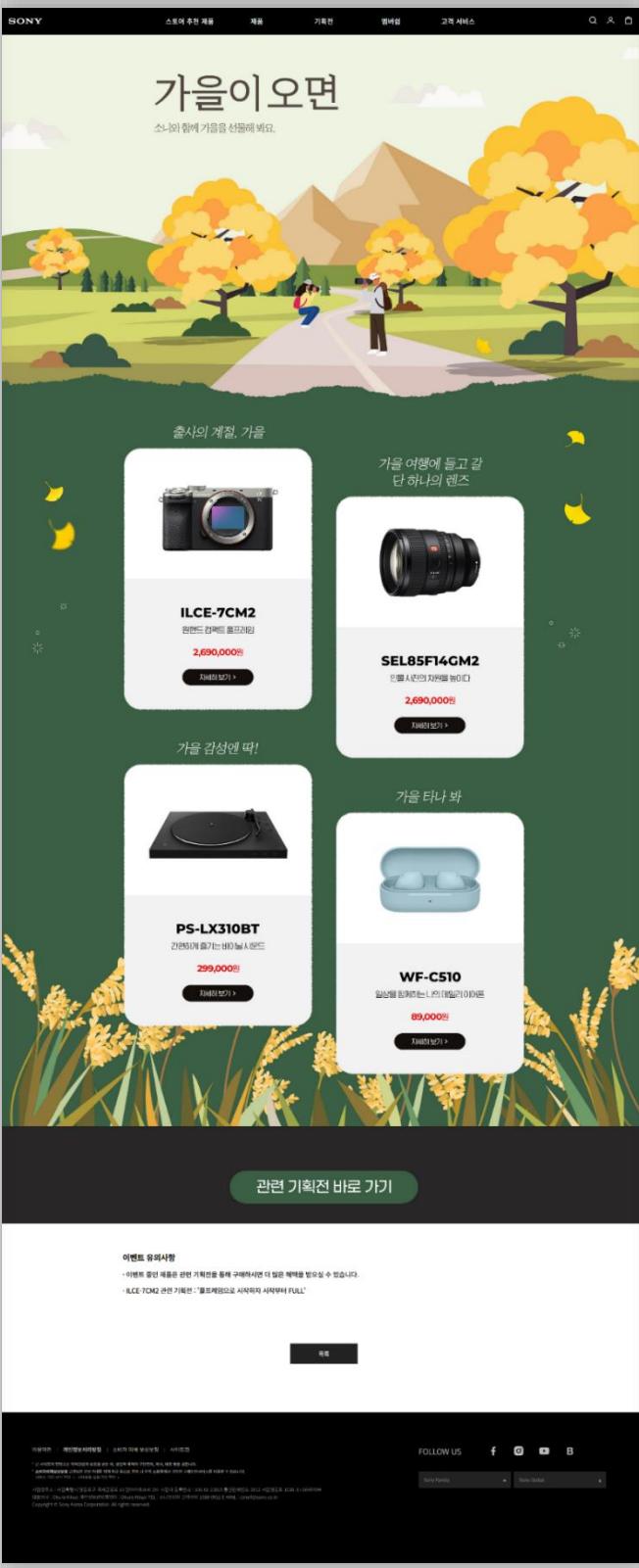
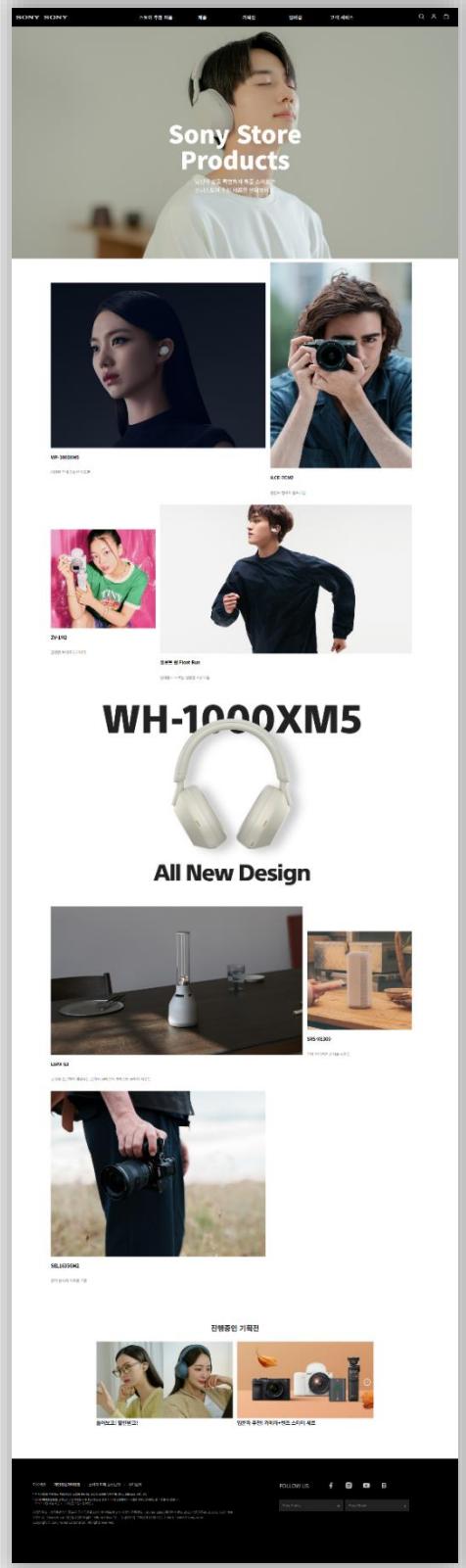


Sony Family

A screenshot of a mobile application or website for Sony Store. The top navigation bar has three items: '제품' (Products), '기획전' (Promotions), and '멤버쉽' (Membership). Below the navigation is a large blue circular icon containing a white stylized 'E' character. The main content area features a large bold text '주문 완료' (Order Complete) and a message '소니스토어를 이용해 주셔서 감사합니다!' (Thank you for using Sony Store!). Below this, another message '주문이 완료 되었습니다.' (Your order has been completed.) is displayed. A large blue box contains the text '주문번호 : 20241216195047000079'. At the bottom, there are two buttons: '계속 쇼핑 하기' (Continue Shopping) in a white box and '주문/배송 조회' (Order/Shipping Inquiry) in a black box. The footer contains legal text: '판매번호 2012-서울영등포-1038 소니코리아(주)' and 'MAIL : cshelf@sony.co.kr'. There is also a small 'F' icon in the bottom right corner.

2-2

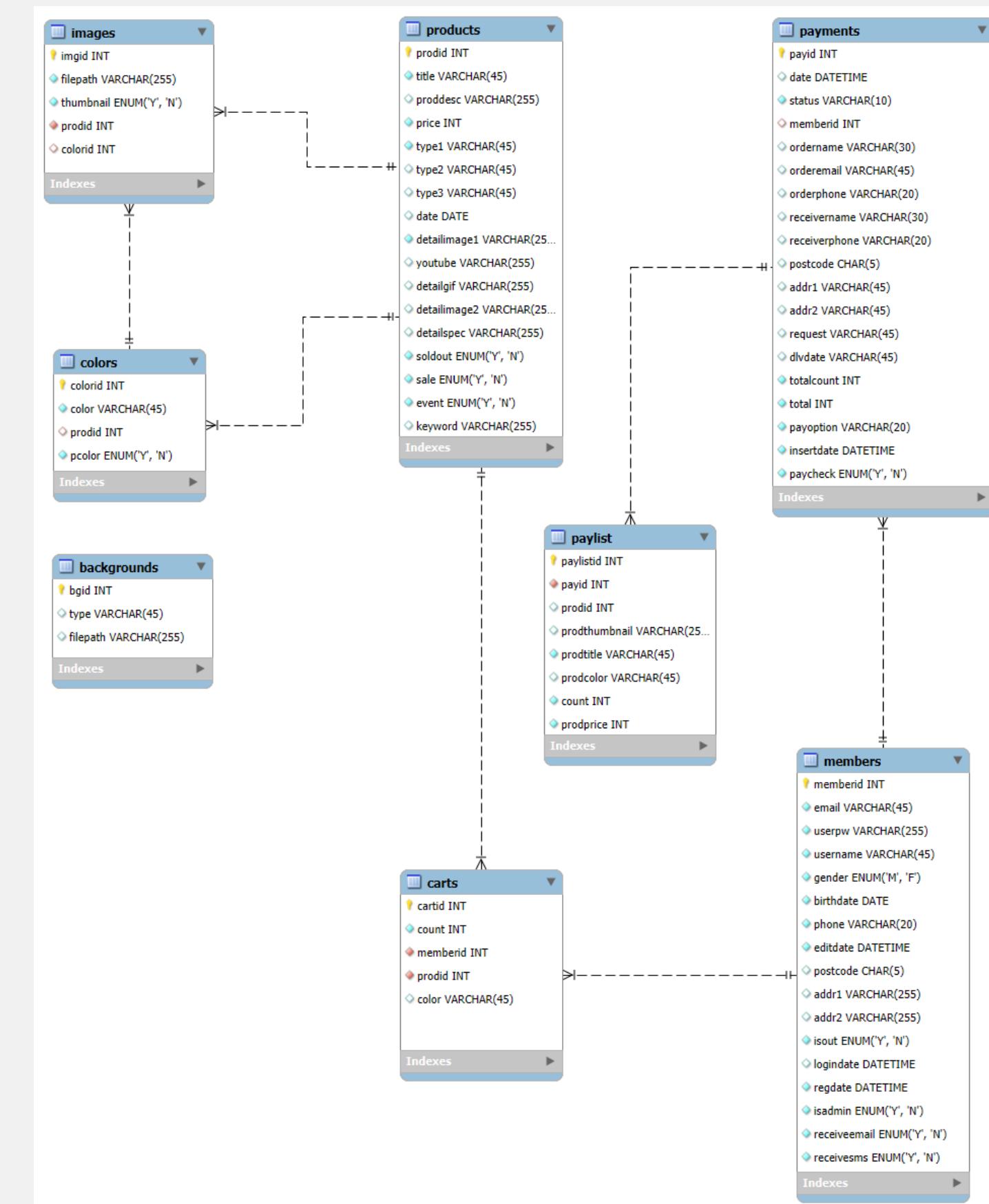
# 화면 구현



PART 3

# 데이터베이스

# 테이블 구성 (ERD)



# 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

## | 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

## | 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

## | 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

## | 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

# 테이블 명세서

## | 결제 목록

TableName		playlist				
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)

PART 4

# 구현기능

# 주요 기능

## 헤더

- thymeleaf를 이용하여 각 페이지를 컨트롤러로 연결
- 로그인 이전 아이콘메뉴와 로그인 이후 아이콘메뉴를 세션정보확인을 통해 구현

## 회원가입/로그인

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

## 아이디/ 비밀번호찾기

- 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 후 새비밀번호로 유효성 검사 후 변경 가능
- 주소 및 광고성 이메일 수신, 문자 수신 여부를 수정 가능
- 회원 탈퇴 – 회원을 탈퇴하면 회원정보에 탈퇴회원 여부의 값이 "Y"로 변경 => 스케줄러를 통해 정해진 시간마다 해당 값이 "Y"인 컬럼의 회원 정보 delete 실행

## 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
- DB에 저장된 색상값으로 색상이 없는 제품을 제외한 모든 제품에 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 pagination으로 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

## 장바구니

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

## 주문·결제

- '장바구니 데이터' 또는 제품상세페이지의 '제품 데이터'를 결제 테이블에 삽입 (주문정보 제외)
- 삽입된 결제 테이블의 데이터를 조회하여 뷰에 전달
- 회원의 결제테이블에서 최근 배송지 5개까지 조회
- 다음 주소찾기 API 사용하여 주소 입력
- 주문·배송 정보 입력품 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 한시간마다 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

## SignRestController

회원 관련 정보를 처리하고 JSON형태로  
객체 데이터를 반환

## OutScheduler

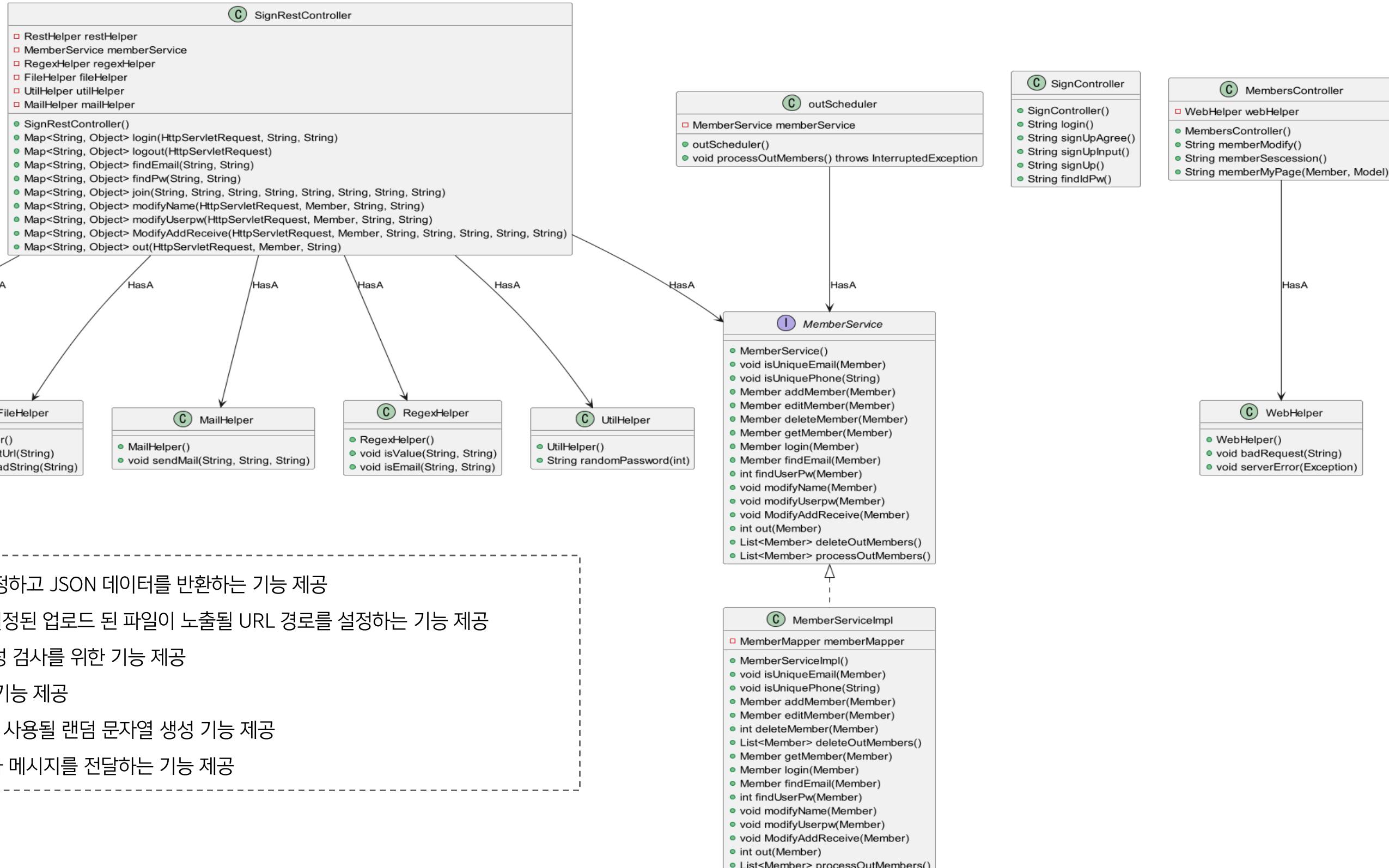
회원 탈퇴한 회원들을 정해진 시간에  
주기적으로 데이터를 지우는 기능

## SignController, MemberController

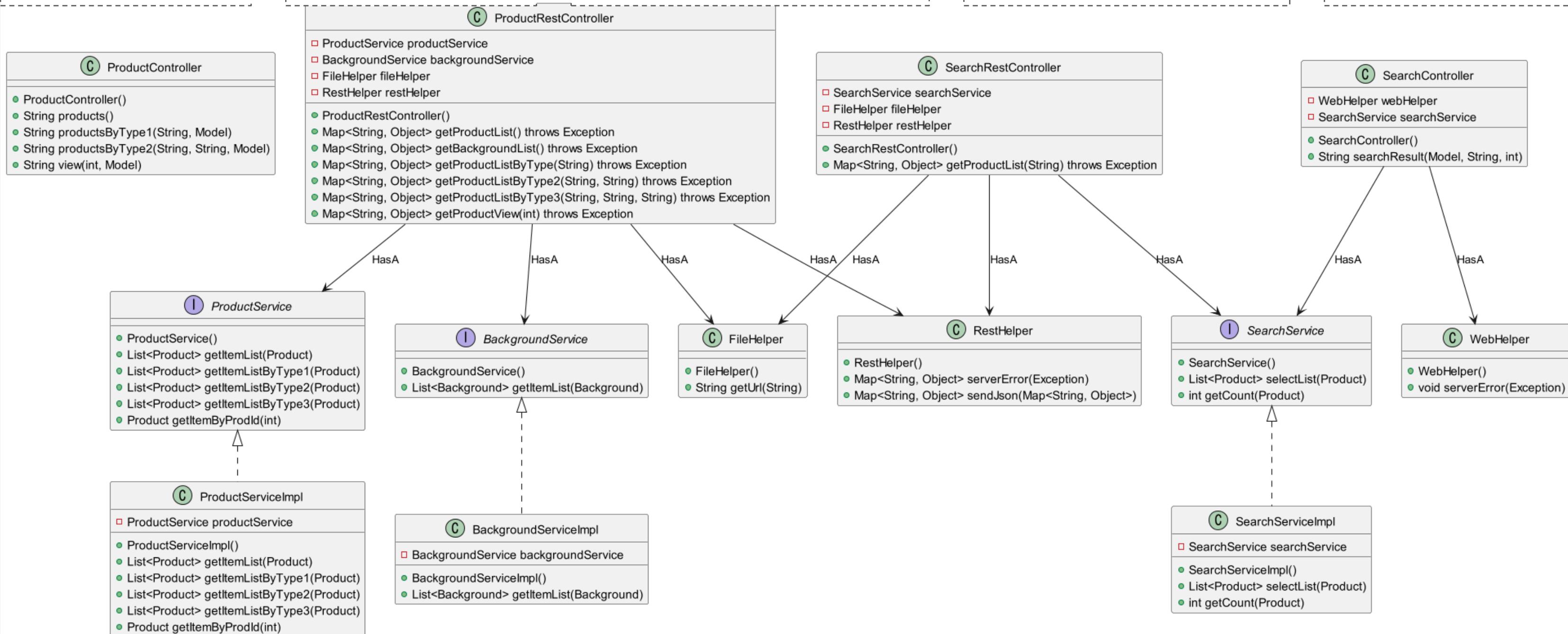
View를 반환

## MemberService

회원 정보를 추가, 수정, 조회, 삭제하는 기능



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 Type1,2,3에 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

# 클래스 다이어그램

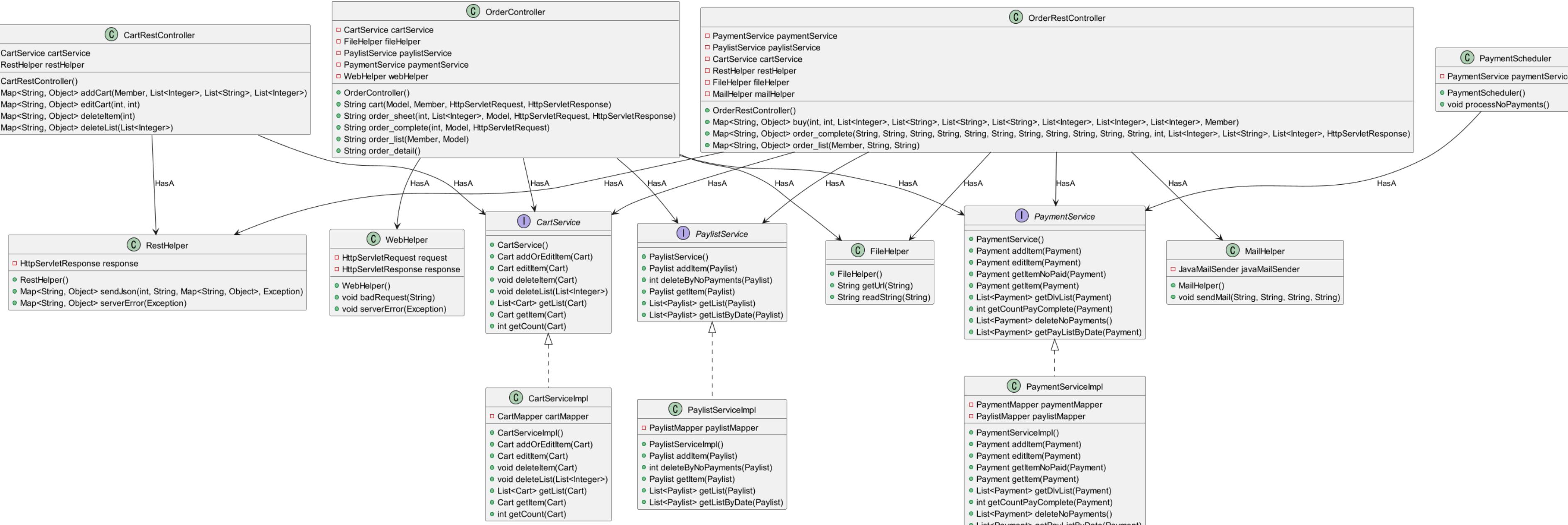
**CartRestController**  
장바구니에 상품을 추가, 수정,  
삭제하는 RESTful API 제공

**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

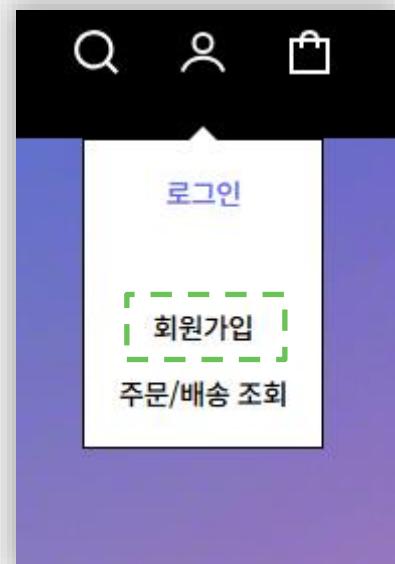
**PaylistService**  
결제 목록에 항목을 추가, 조회,  
삭제

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

**PaymentService**  
결제 항목을 추가, 수정, 조회,  
삭제하는 기능

4-2

# 구현 기능 - 회원가입 (1)



## 회원가입

소니스토어와 소니 고객지원 사이트는 하나의 아이디와 비밀번호로 운영됩니다.  
회원가입을 통해 다양한 서비스를 이용하실 수 있습니다.

**소니스토어 회원 가입**

· 소니코리아 통합 웹회원 정책 상 공식적으로만 14세 미만의 경우 회원가입이 불가합니다.

## 소니스토어

소니 고객지원 사이트(SCS) 와 소니스토어는 하나의 ID 와 비밀번호로 운영됩니다.  
소니 고객지원 사이트(SCS)의 이용약관에 함께 동의하시면,  
하나의 ID로 고객지원 사이트(SCS)를 편리하게 이용하실 수 있습니다.

**약관 전체 동의** (선택 항목 포함)

[필수] 소니스토어 쇼핑몰 이용약관 동의

[필수] 소니 고객지원 사이트(SCS) 이용약관 동의

[필수] 회원가입 개인정보 수집에 관한 동의

[선택] 마케팅 목적의 개인정보 처리 및 광고성 정보 수신에 관한 동의

전체보기 >

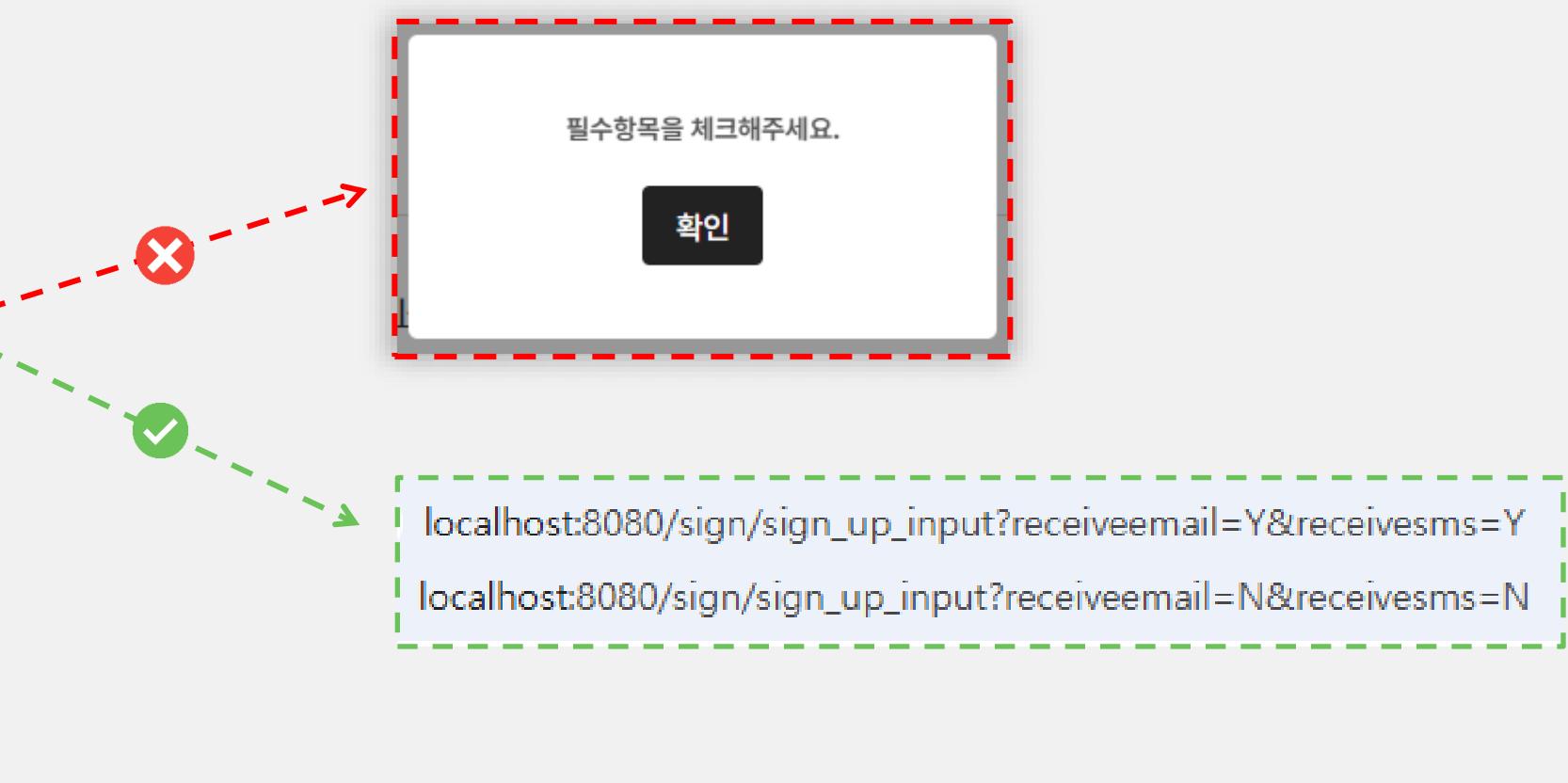
전체보기 >

전체보기 >

전체보기 >

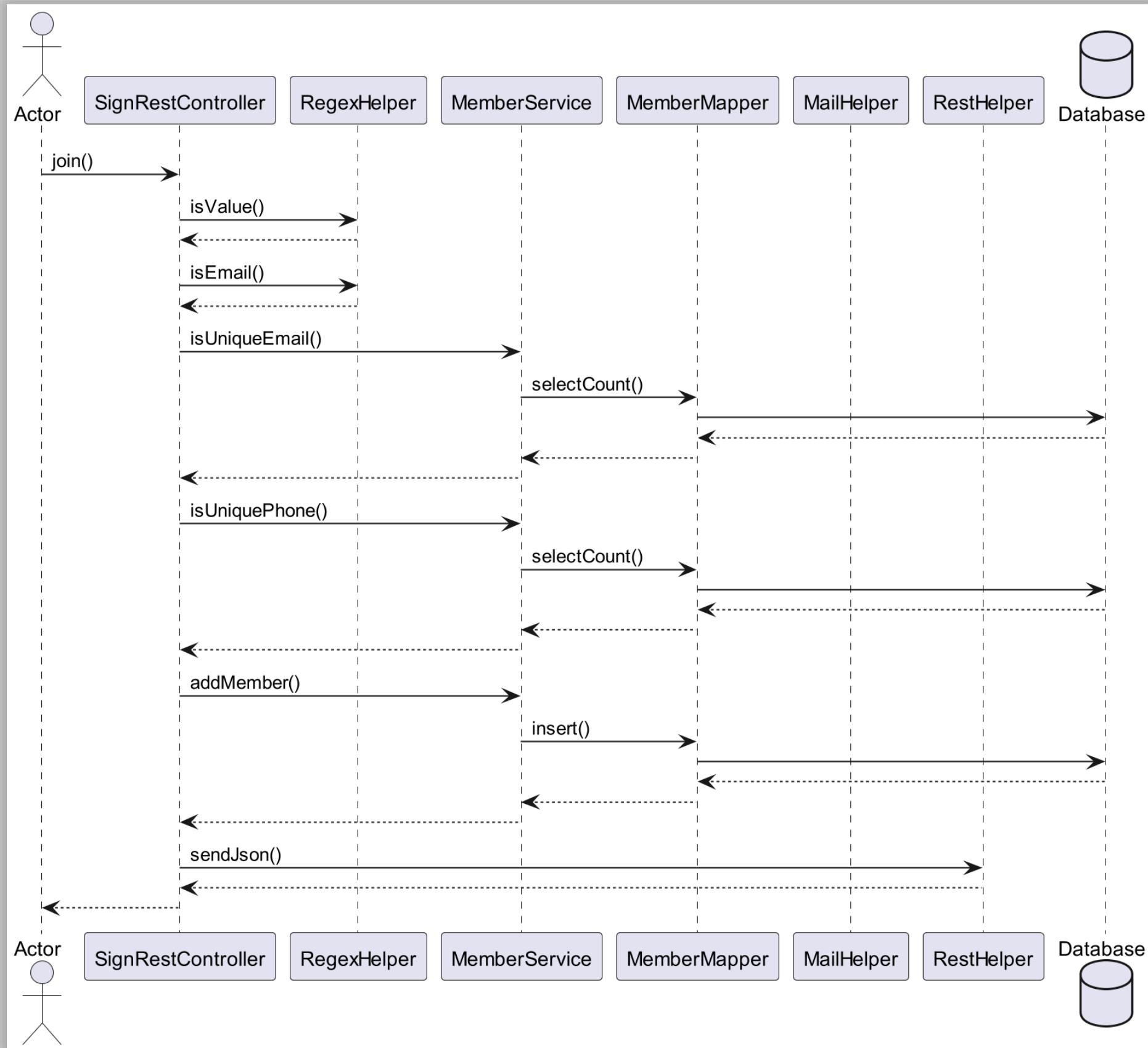
**동의**

- [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)



4-2

# 구현 기능 - 회원가입 (2)



**회원가입**

소니코리아 통합 웹회원 정책 상 공식적으로만 14세 미만의 경우 회원가입이 불가합니다.

이메일 아이디 (예:sony@sony.co.kr)

비밀번호 (대/소문자, 숫자, 특수문자, 3종 포함 12~15자리 미만)

- 비밀번호 표시/숨기기 버튼 이벤트

비밀번호 확인

이름 (띄어쓰기 없이 입력하세요.)

연도-월-일

성별  여성  남성

휴대폰 번호 (-없이 입력하세요.)

인증번호

가입완료

- 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

# 구현 기능 - 아이디 찾기

회원 로그인    비회원 로그인

이메일 아이디(예:sony@sony.co.kr)

비밀번호 (대/소문자, 숫자, 특수문자, 3종 포함 12~15자리 미만)

로그인

아이디 · 비밀번호 찾기      회원가입

### 아이디 · 비밀번호 찾기

아이디·비밀번호가 기억나지 않으신가요?  
등록하신 정보로 아이디와 비밀번호를 찾으실 수 있습니다.

\* SNS 계정으로 가입하신 회원님은 가입하신 SNS의 이메일 아이디로 찾으실 수 있고  
비밀번호의 경우는 가입하신 SNS에서 직접 찾기, 혹은 변경하실 수 있습니다.

아이디 찾기      비밀번호 찾기

이름(띄어쓰기 없이 입력하세요.)

휴대폰 번호(없이 입력하세요.)

본인인증

확인

· 가입하신 정보와 일치하지 않을 경우 검색이 안될 수 있습니다.

500 Error

가입된 정보가 없습니다.

확인

아이디 찾기      비밀번호 찾기

박진수님의 이메일 아이디 검색 결과입니다.

jinsu4205@gmail.com

로그인

· 가입하신 정보와 일치하지 않을 경우 검색이 안될 수 있습니다.

### 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

# 구현 기능 - 비밀번호 찾기

**아이디 · 비밀번호 찾기**

아이디·비밀번호가 기억나지 않으신가요?  
등록하신 정보로 아이디와 비밀번호를 찾으실 수 있습니다.

\* SNS 계정으로 가입하신 회원님은 가입하신 SNS의 이메일 아이디로 찾으실 수 있고  
비밀번호의 경우는 가입하신 SNS에서 직접 찾기, 혹은 변경하실 수 있습니다.

아이디 찾기      **비밀번호 찾기**

---

이메일 아이디 (예:sony@sony.co.kr)

휴대폰 번호(-없이 입력하세요.)

**분인인증**

**확인**

· 가입하신 정보와 일치하지 않을 경우 검색이 안될 수 있습니다.



## MemberMapper.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

# 구현 기능 - 로그인, 로그아웃

회원 로그인   비회원 로그인

---

이메일 아이디(예:sony@sony.co.kr)

---

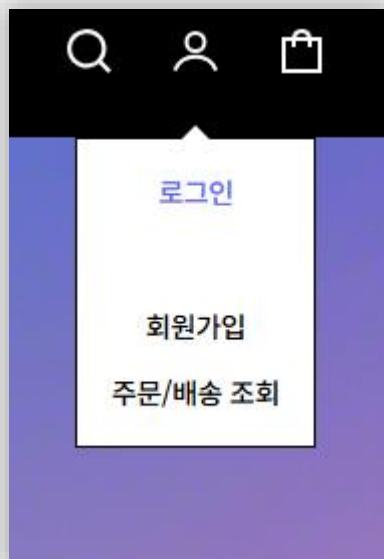
비밀번호 (대/소문자, 숫자, 특수문자, 3종 포함 12~15자리 미만)

---

**로그인**

---

아이디 · 비밀번호 찾기 | 회원가입



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>
```

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

SignRestController.java

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

    return restHelper.sendJson();
}
```

4-2

# 구현 기능 - 회원정보 수정

<마이페이지

## 회원정보

회원탈퇴

이름	박진수	<input type="button" value="이름변경"/>
휴대폰	01033063205	<input type="button" value="인증번호 전송"/>
인증번호		<input type="button" value="인증"/>
이메일 아이디	jinsu4205@gmail.com	
생년월일	1996-09-05	
성별	F	
회원구분	<input checked="" type="checkbox"/> MEMBERSHIP	
비밀번호	*****	<input type="button" value="비밀번호 변경"/>
주소	06779 서울 서초구 강남대로 48-3 (양재동) 102호	<input type="button" value="주소 변경"/>
이벤트/프로모션 알림	<input checked="" type="checkbox"/> 메일 수신 <input checked="" type="checkbox"/> SNS 수신	
<input type="button" value="회원정보 수정"/>		

개명 회원 본인 확인

이름

휴대폰 번호

Daum Postcode Service - Chrome  
about:blank  
예) 판교역로 166, 분당 주공, 백현동 532

**tip**  
아래와 같은 조합으로 검색을 하시면 더욱 정확한 결과가 검색됩니다.  
도로명 + 건물번호  
예) 판교역로 166, 제주 첨단로 242  
지역명(동/리) + 번지  
예) 백현동 532, 제주 영평동 2181  
지역명(동/리) + 건물명(아파트명)  
예) 분당 주공, 연수동 주공3차  
사서함명 + 번호  
예) 분당우체국사서함 1~100

Powered by kakao | 우편번호 서비스 안내

- 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

비밀번호 변경

비밀번호 입력

새 비밀번호 입력

새 비밀번호 확인

**① 주의사항 및 안내**

- 영문 대문자와 소문자, 숫자 조합으로 12자리 이상 15자리 미만으로 입력해 주세요.
- 일부 특수문자도 사용 가능합니다.
- 아이디와 같은 비밀번호나 주민등록번호, 생일, 전화번호 등 개인정보와 관련된 숫자, 연속된 숫자, 반복되는 숫자 등은 타인이 쉽게 알아낼 수 있어 유출의 위험이 있으니 사용하지 않는 것이 좋습니다.
- 이전에 사용된 비밀번호를 재사용하시면 도용의 우려가 있으니 새로운 비밀번호로 사용해 주세요.
- 소니스토어에서 사용하는 자체 비밀번호는 회원 정보 수정 및 회원 탈퇴 시에 꼭 필요합니다.(가입 시 이용한 각 SNS 계정 비밀번호와는 다른)
- 비밀 번호는 주기적으로 변경해 주세요. (최소 3개월에 1회는 변경해 주시는 것이 좋습니다.)
- 타 사이트와 동일하거나 유추할 수 있는 비밀 번호는 도용의 위험이 크니, 안전도가 높은 비밀 번호로 변경하시길 권장합니다.
- SNS 계정으로 가입하신 회원님은 각 SNS 페이지에서 수정 및 변경하시기 바랍니다.

# 구현 기능 - 회원 탈퇴

<마이페이지  
회원탈퇴

jinsu4205@gmail.com

개인정보, 쿠폰 정보, 보유하신 마일리지, 정품등록 정보 등 회원 탈퇴 시 삭제됩니다.  
SNS 계정으로 가입하신 회원님은 하단에 가입하신 SNS로 인증하신 후 탈퇴 사유를 선택해주세요.

회원탈퇴가 완료되었습니다.

비밀번호

확인

탈퇴사유를 선택해주세요.

회원탈퇴

[안내]

- 소니코리아 고객지원 사이트와 소니스토어는 하나의 이메일 ID와 비밀번호로 운영됩니다.
- 회원 탈퇴 신청 시 "소니코리아 고객지원 사이트와 소니스토어" 모두 탈퇴처리됩니다.
- 고객님의 소중한 의견을 참고하여, 보다 나은 소니코리아가 될 수 있도록 하겠습니다.

## 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 설정
- 스케줄러를 이용해 매주 월요일 00시 isout이 "Y"로 수정된 시간이 7일이 지난 데이터 삭제

# 구현 기능 - 제품 목록 페이지 (1)



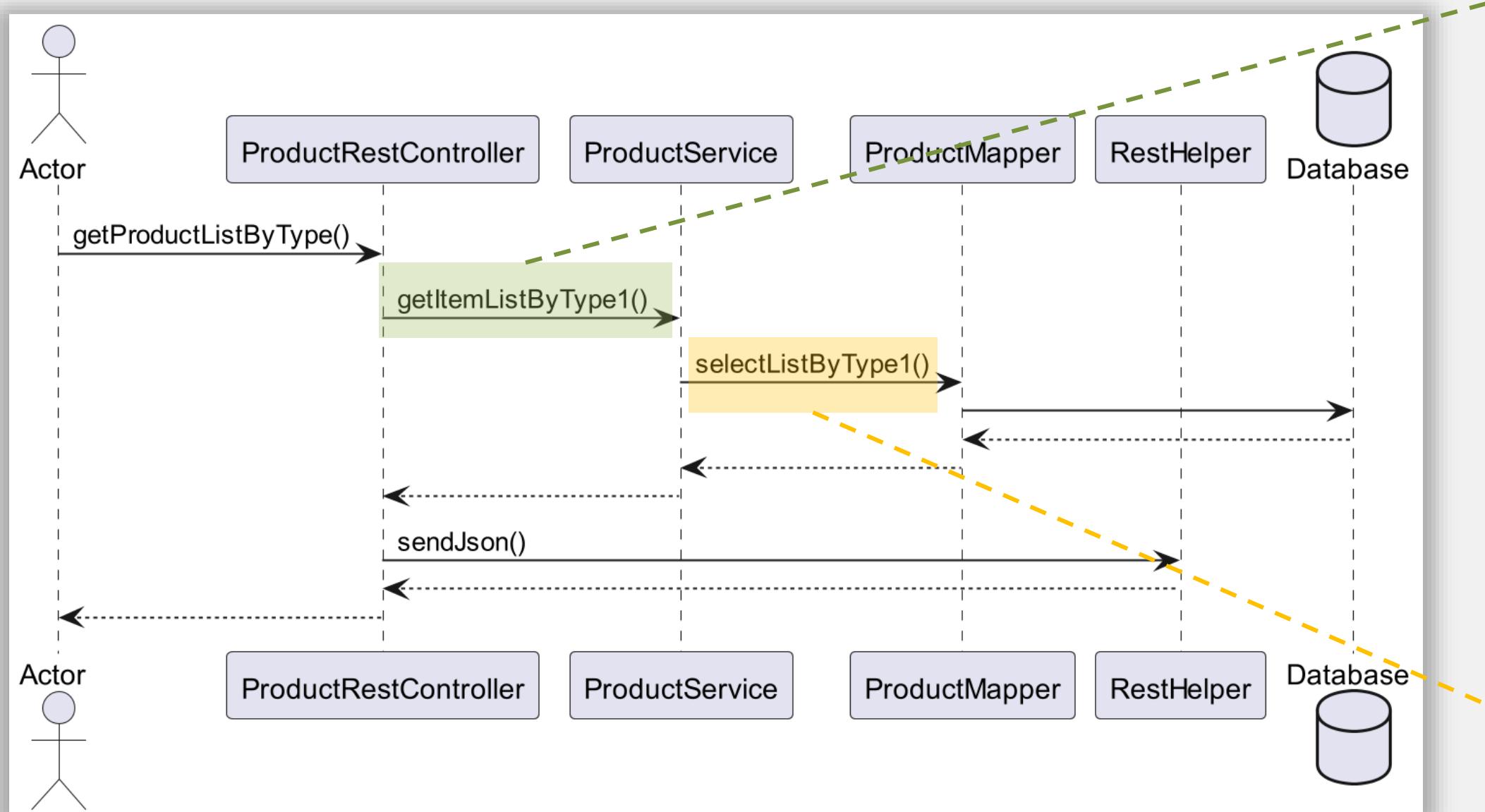
## ProductRestController.java

```
// 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> getProductsListByType(@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 Type1에 따라 제품 목록을 JSON 형식으로 반환하는 RESTful API

# 구현 기능 - 제품 목록 페이지 (2)



## ProductServiceImpl.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

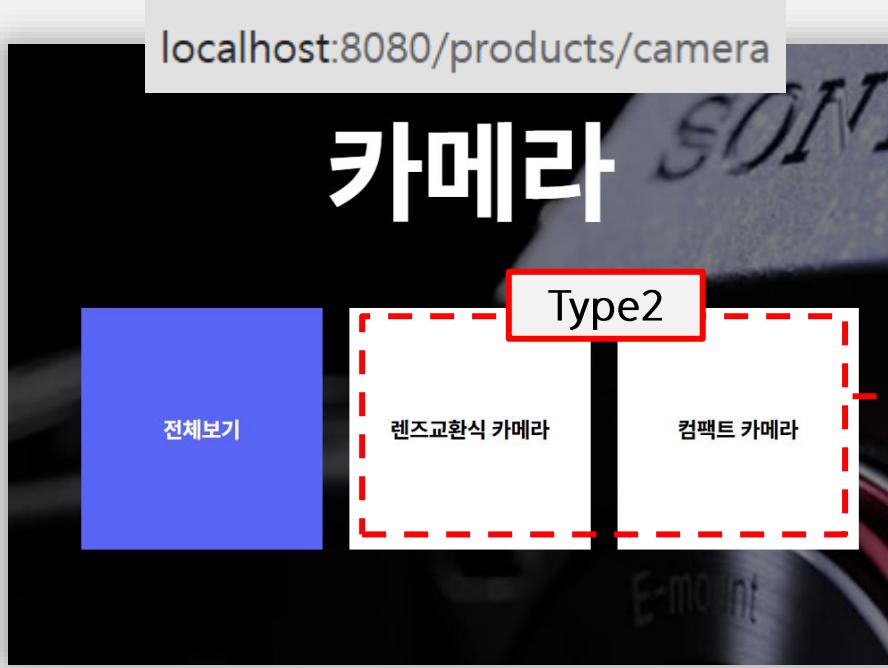
```

// 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 Type1에 따른 제품 목록(이미지와 색상 포함)을 조회하는 SQL 쿼리를 정의

4-2

# 구현 기능 - 제품 목록 페이지 (3)

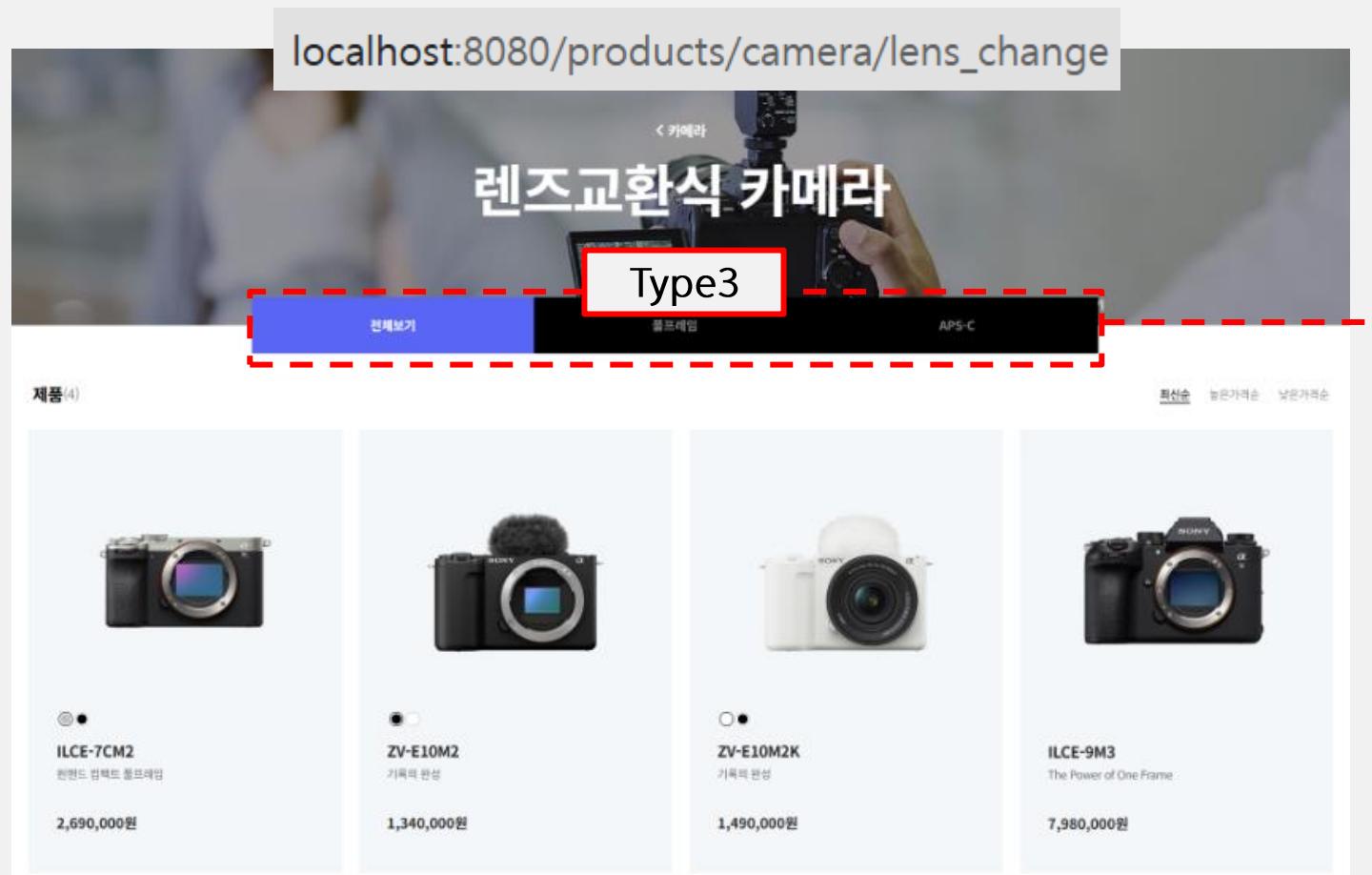


ProductRestController.java

```
@GetMapping("/api/products/{type}/{type2}")
public Map<String, Object> getProductsByType(@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

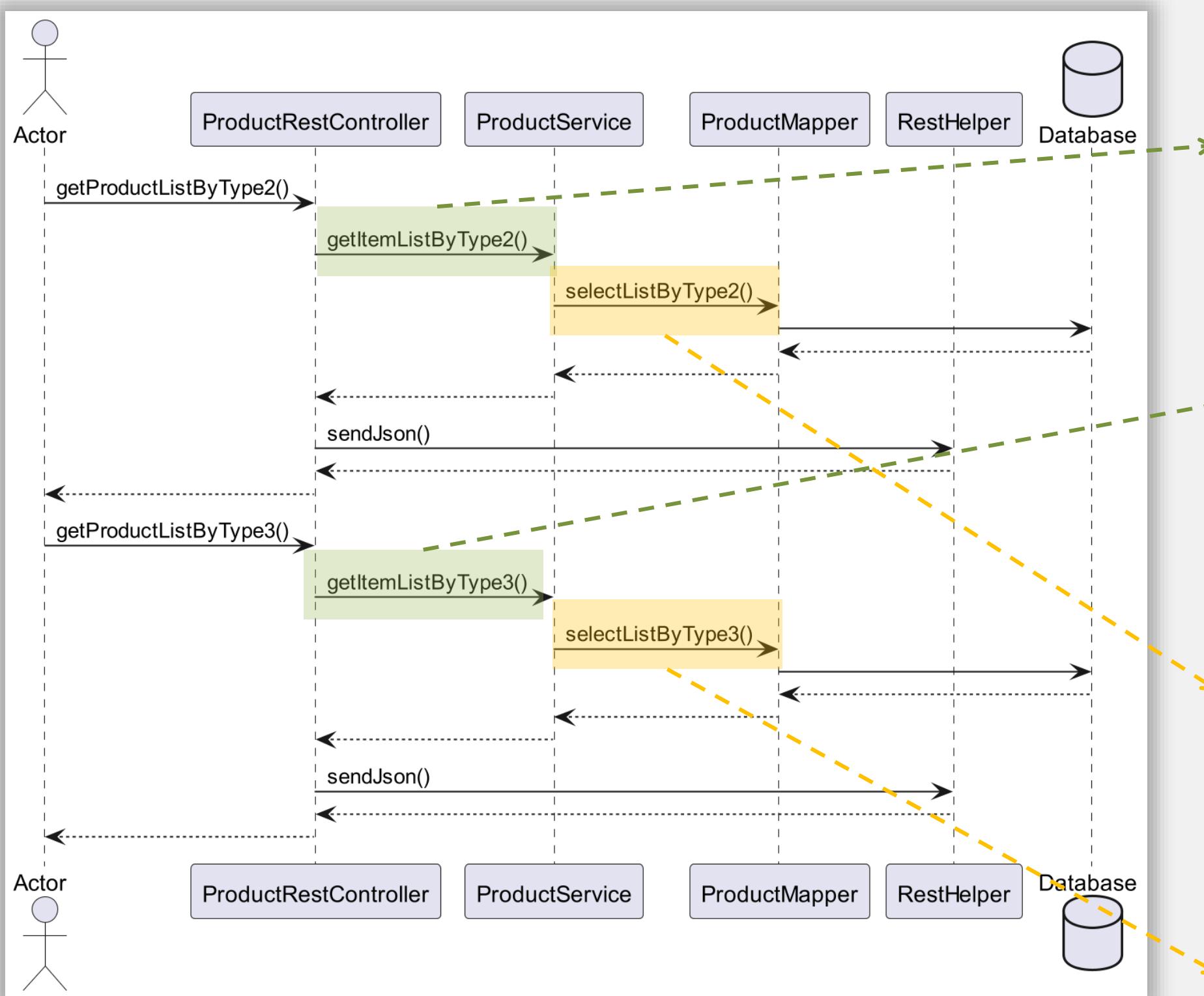


```
@GetMapping("/api/products/{type}/{type2}/{type3}")
public Map<String, Object> getProductsByType3(@PathVariable("type") String type, @PathVariable("type2") String type2, @PathVariable("type3") String type3) throws Exception {
    List<Product> output;
    try {
        Product input = new Product();
        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)

# 구현 기능 - 제품 목록 페이지 (4)



## ProductServiceImpl.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;
}

```

```

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

```

// 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.imgur, 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);

```

```

// 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.imgur, 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, Type3에 따른 제품 목록(이미지와 색상 포함)을 조회하는 SQL 쿼리를 정의

4-2

# 구현 기능 - 제품 상세 페이지

localhost:8080/product-view/2

ZV-E10M2

기록의 완성  
이 제품은 기획전을 통해 구매하시면 더 많은 혜택을 받으실 수 있습니다.

1,340,000 원 무료배송

회원별 마일리지 적립혜택 ⓘ

VIP 4% 53,600 MEMBERSHIP 2% 26,800

색상

black

제품선택

제품을 선택하세요

총 상품금액 원

바로 구매하기

함께 구매하시면 좋은 추천 제품

ZV-E10M2K	ZV-1F	ZV-1	ILCE-9M3
기록의 완성	나를 담는 브이로그 카메라 ZV-1F	나의 첫번째 브이로그 카메라	The Power of One Frame
1,490,000원	759,000원	999,000원	7,980,000원

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

## ProductRestController.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

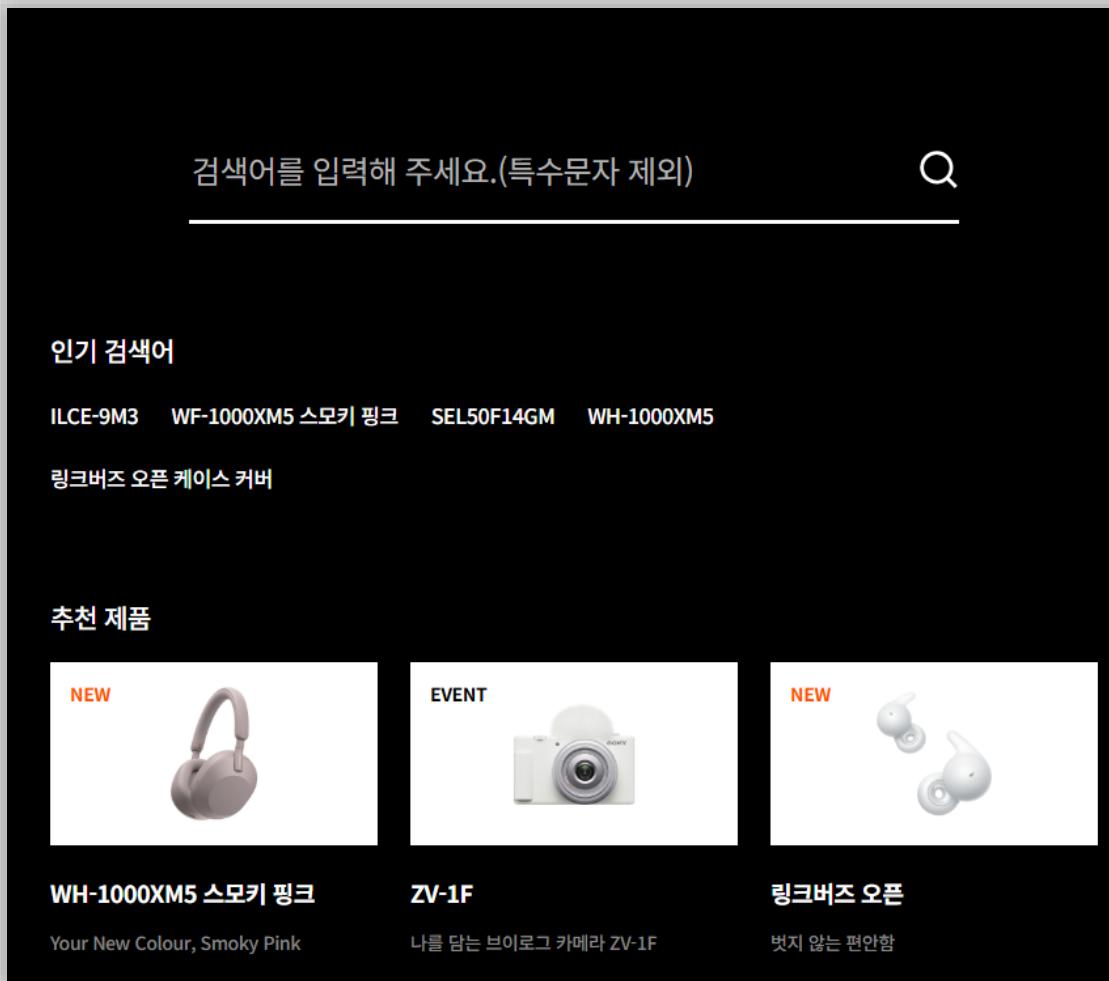
## ProductMapper.java

```
// 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.imgur, 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

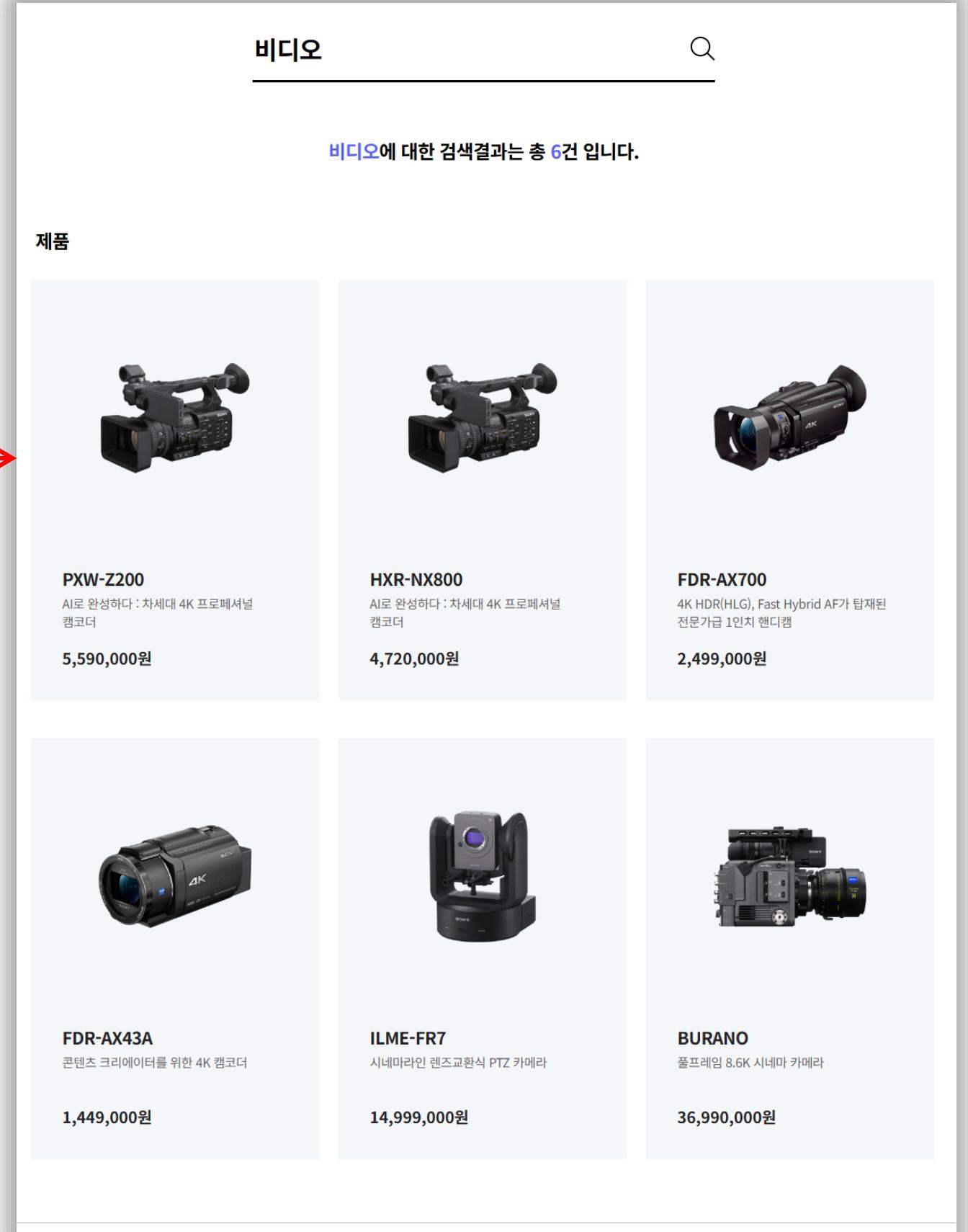
4-2

# 구현 기능 - 제품 검색



```
@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.imgur, 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>")
```

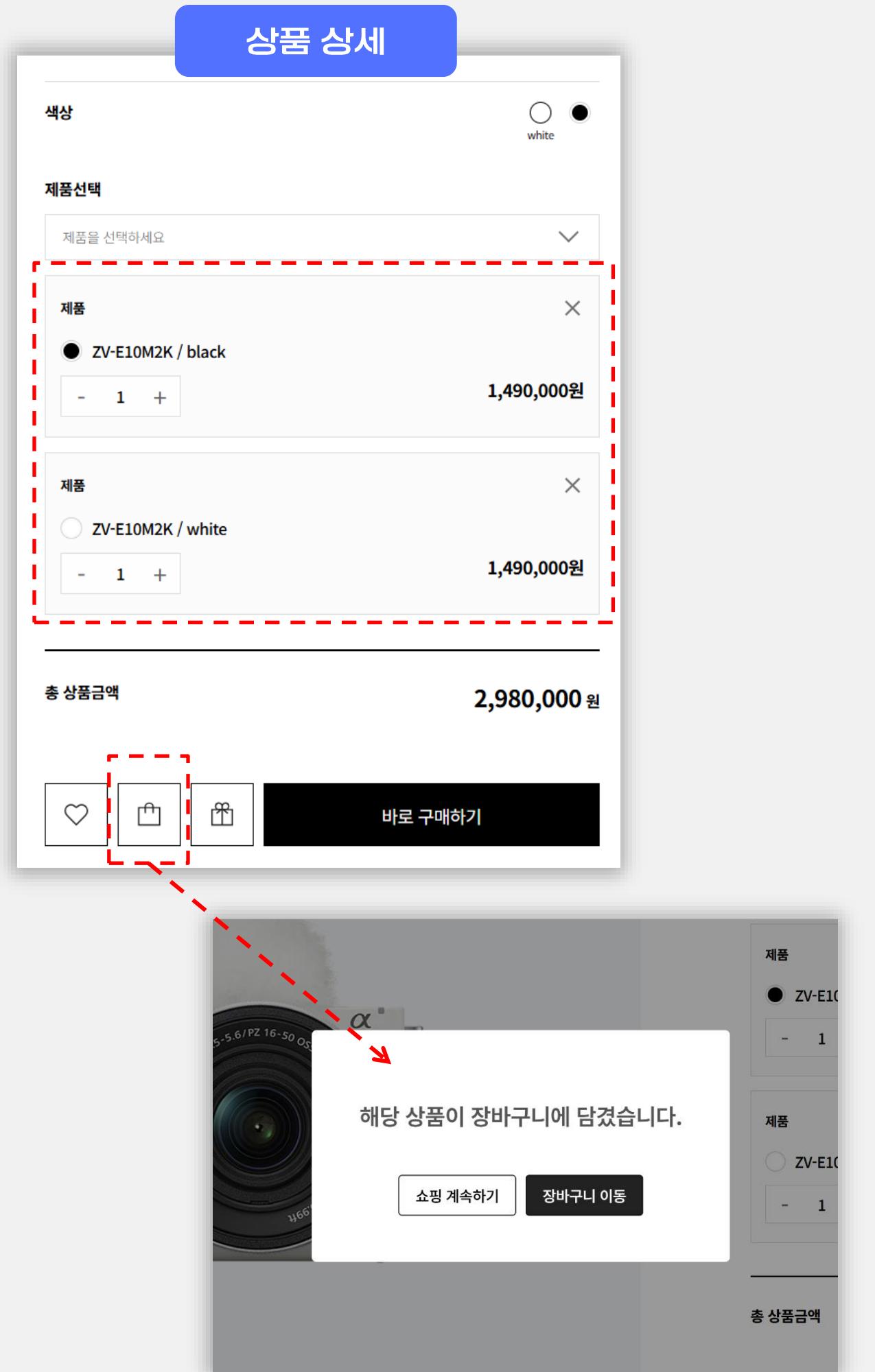
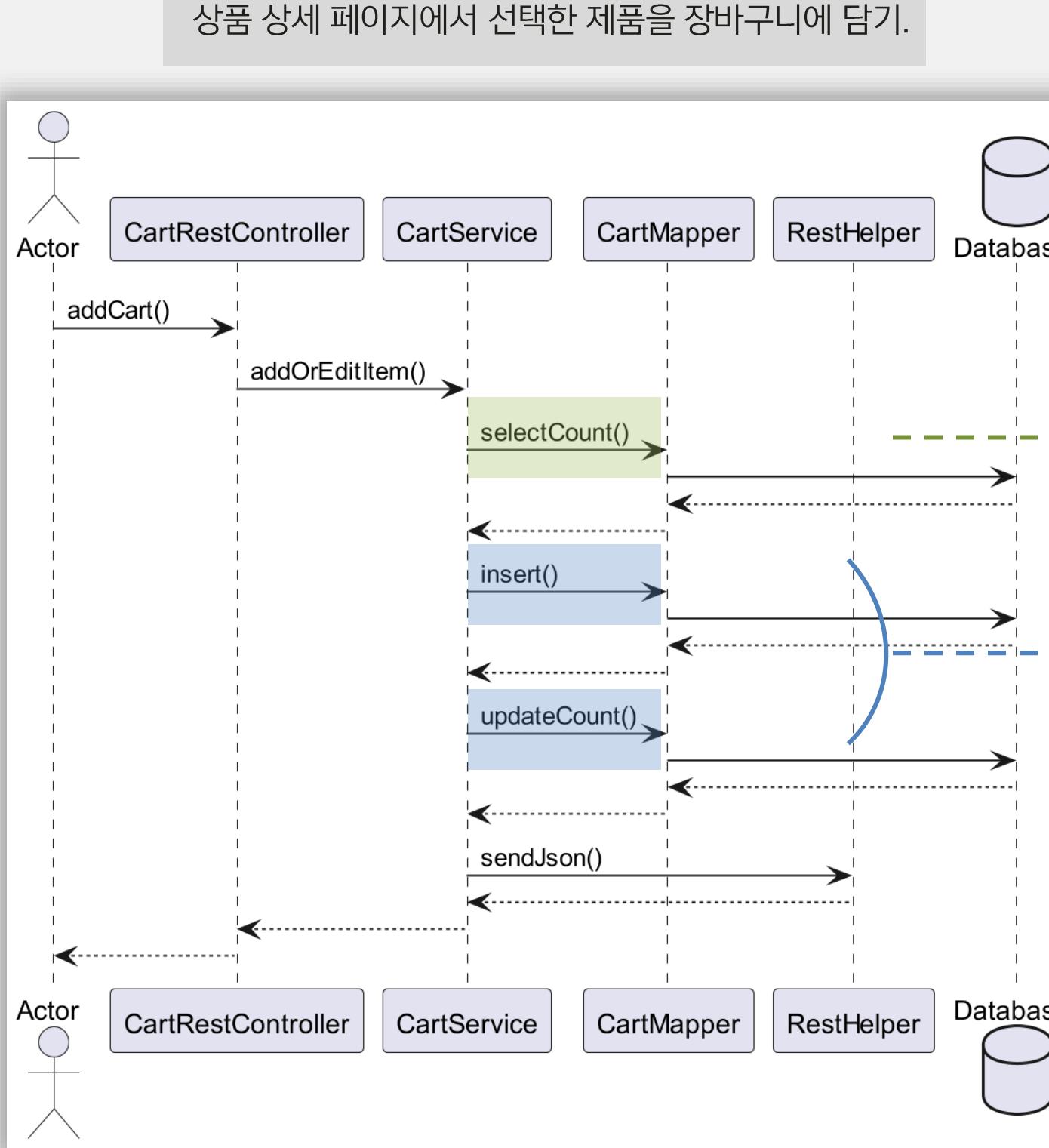
SearchMapper.java



4-2

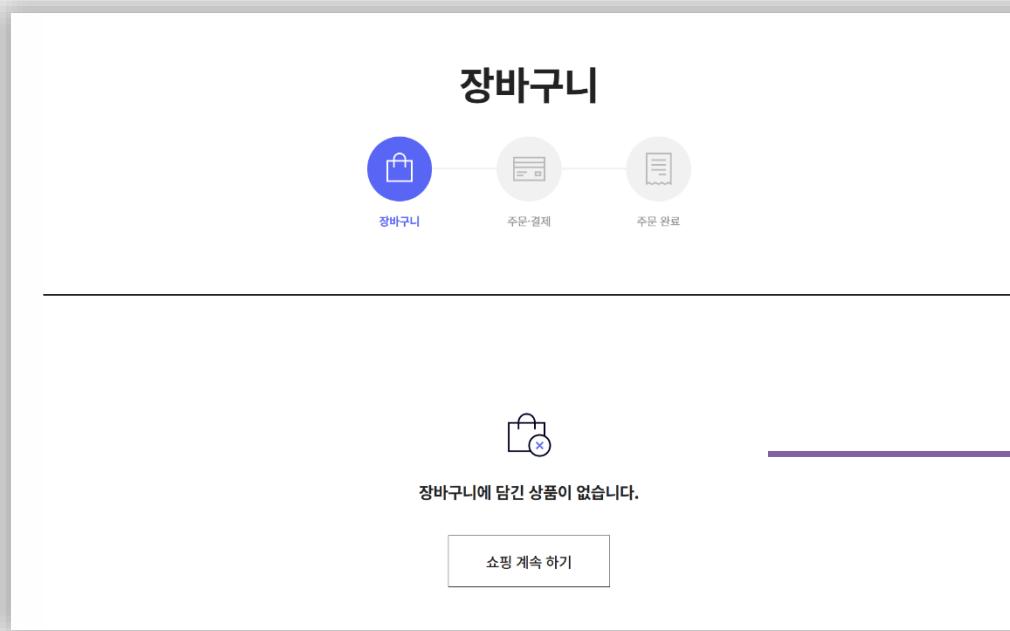
# 구현 기능 - 장바구니 담기

상품 상세



4-2

# 구현 기능 - 장바구니 조회

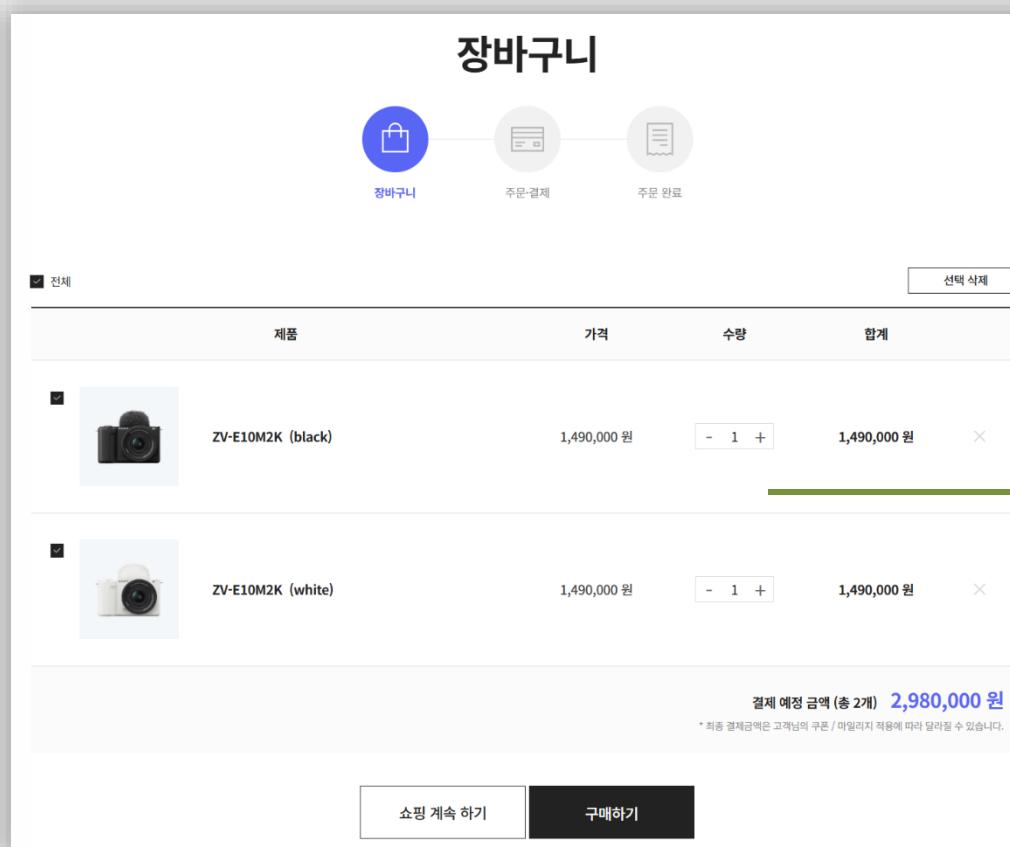


cart.html

```
<!-- 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

# 구현 기능 - 장바구니 수량 변경 / 삭제

**수량 변경**

제품	수량	합계
ILCE-7CM2 (silver)	2,690,000 원	<input type="text" value="2"/> <span style="border: 1px dashed red; padding: 2px;">-</span> <span style="border: 1px dashed red; padding: 2px;">+</span> 5,380,000 원 <span style="color: purple;">×</span>
ILCE-9M3	7,980,000 원	<input type="text" value="3"/> <span style="border: 1px dashed red; padding: 2px;">-</span> <span style="border: 1px dashed red; padding: 2px;">+</span> 23,940,000 원 <span style="color: purple;">×</span>
<span style="background-color: #e0e0ff; border: 1px solid #ccc; padding: 2px;">결제 예정 금액 (총 5개) 29,320,000 원</span> <small>* 최종 결제금액은 고객님의 쿠폰 / 마일리지 적용에 따라 달라질 수 있습니다.</small>		

**삭제**

제품	가격	수량	합계
ILCE-7CM2 (silver)	2,690,000 원	<input type="text" value="1"/> <span style="border: 1px dashed red; padding: 2px;">-</span> <span style="border: 1px dashed red; padding: 2px;">+</span>	2,690,000 원 <span style="border: 1px dashed purple; padding: 2px;">×</span>
ILCE-9M3	7,980,000 원	<input type="text" value="1"/> <span style="border: 1px dashed red; padding: 2px;">-</span> <span style="border: 1px dashed red; padding: 2px;">+</span>	7,980,000 원 <span style="border: 1px dashed purple; padding: 2px;">×</span>
<span style="background-color: #e0e0ff; border: 1px solid #ccc; padding: 2px;">선택 삭제</span>			
<span style="background-color: #e0e0ff; border: 1px solid #ccc; padding: 2px;">결제 예정 금액 (총 2개) 10,670,000 원</span> <small>* 최종 결제금액은 고객님의 쿠폰 / 마일리지 적용에 따라 달라질 수 있습니다.</small>			

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

## <다중 삭제>

삭제하고 싶은 상품을 체크하여 장바구니에서 “선택 삭제” .

## <단일 삭제>

“X” 버튼 클릭 시, 장바구니에서 삭제.

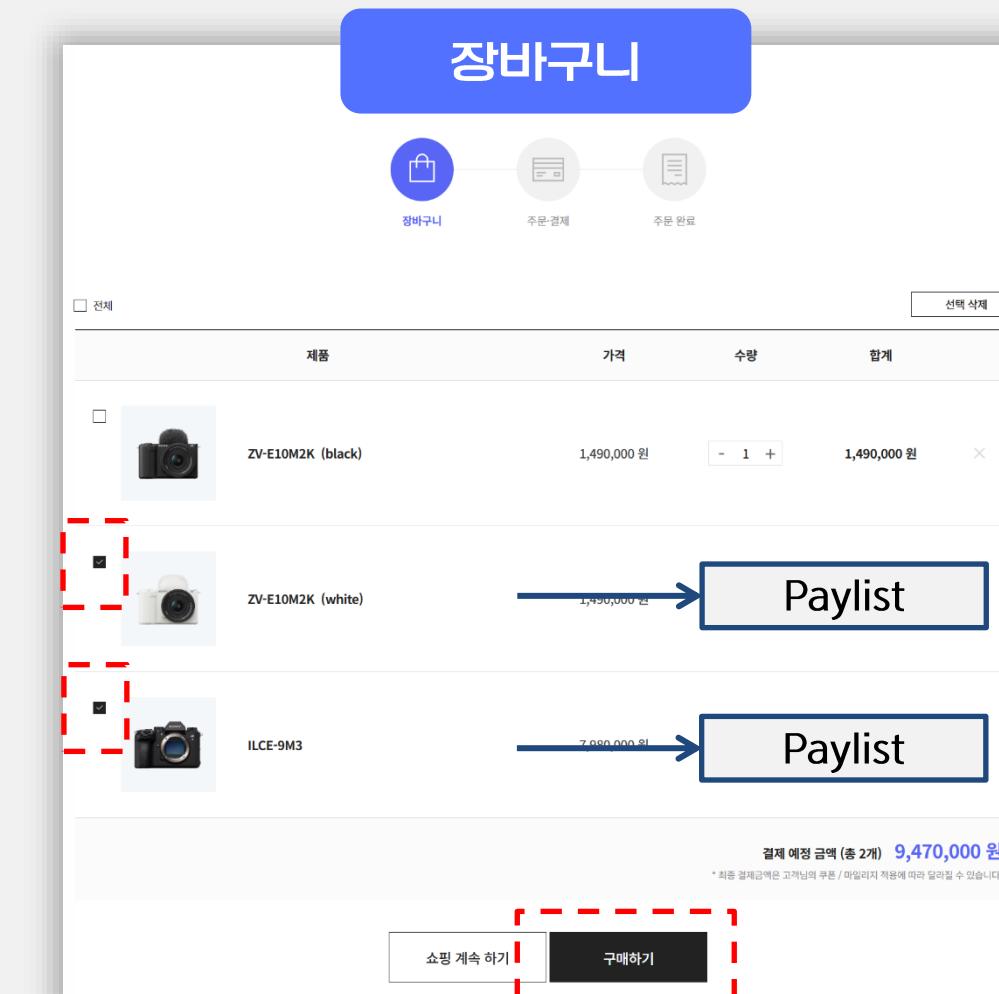
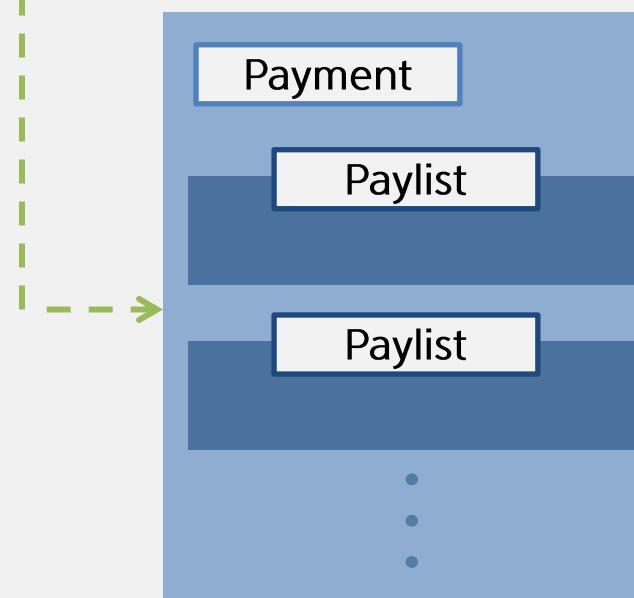
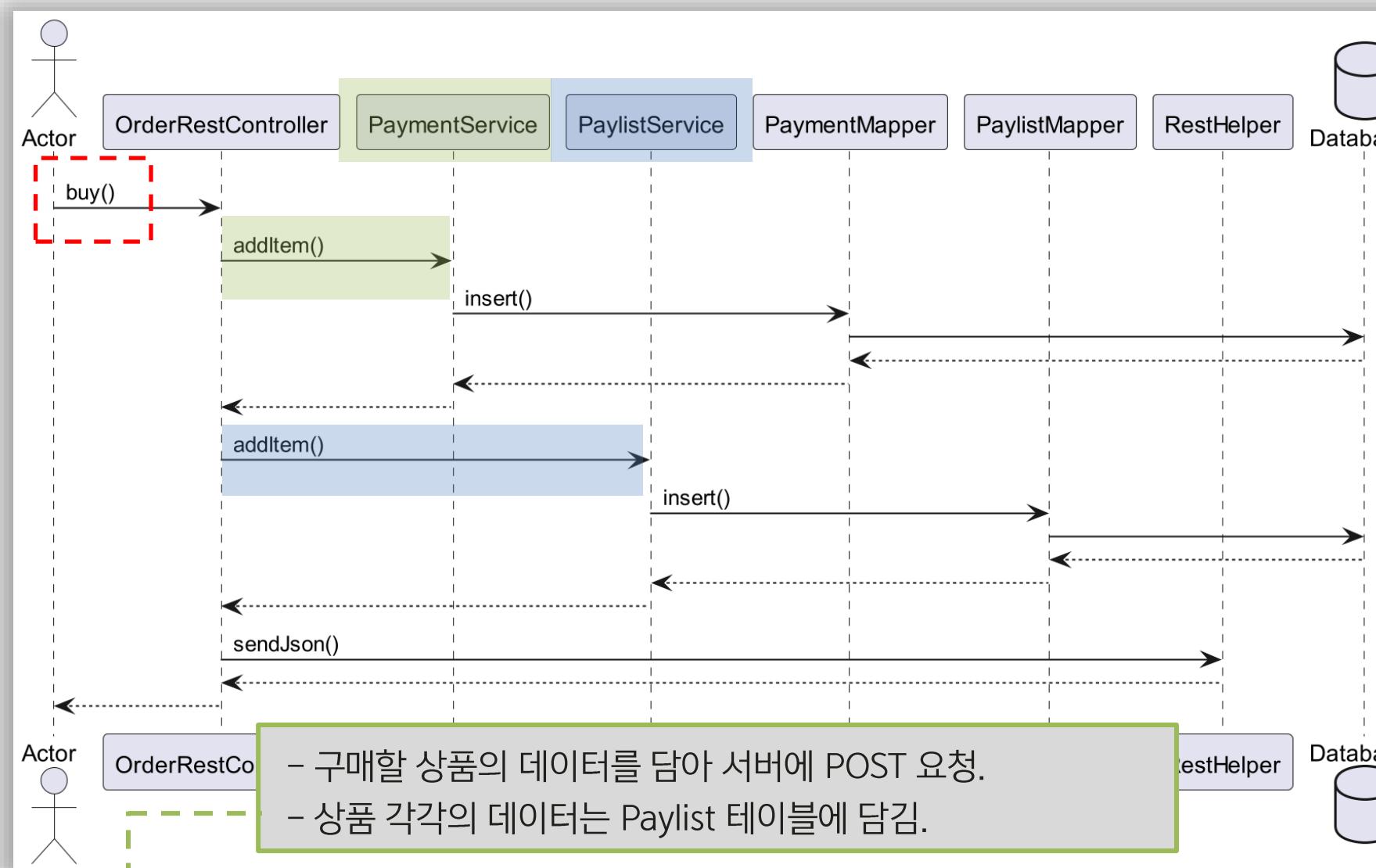
↓ 장바구니 DB의 PK(cartid)로 반복을 돌려 다중 삭제하는 SQL문.

```
@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);
```

CartMapper.java

4-2

# 구현 기능 - 구매하기



체크한 제품들만 결제 테이블에 추가

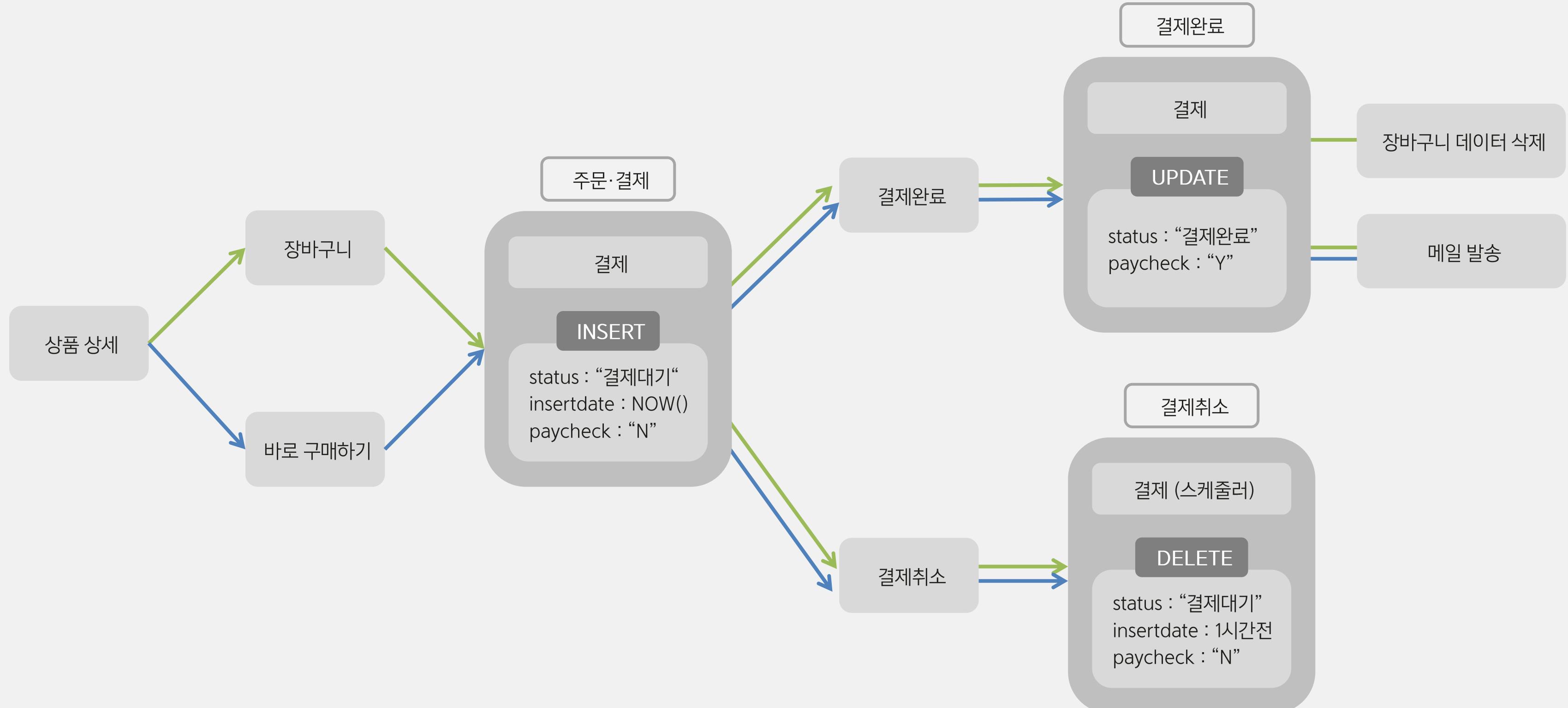
```
const cartids = data.cartids.map( v => `cartid=${v}`).join('&');  
window.location = `/order/sheet?orderSheetNo=${data.item.payid}&${cartids}`;
```

결제 후, 장바구니의 데이터를 삭제하기 위해 PK인 cartid를 쿼리스트링으로 가지고감.

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

4-2

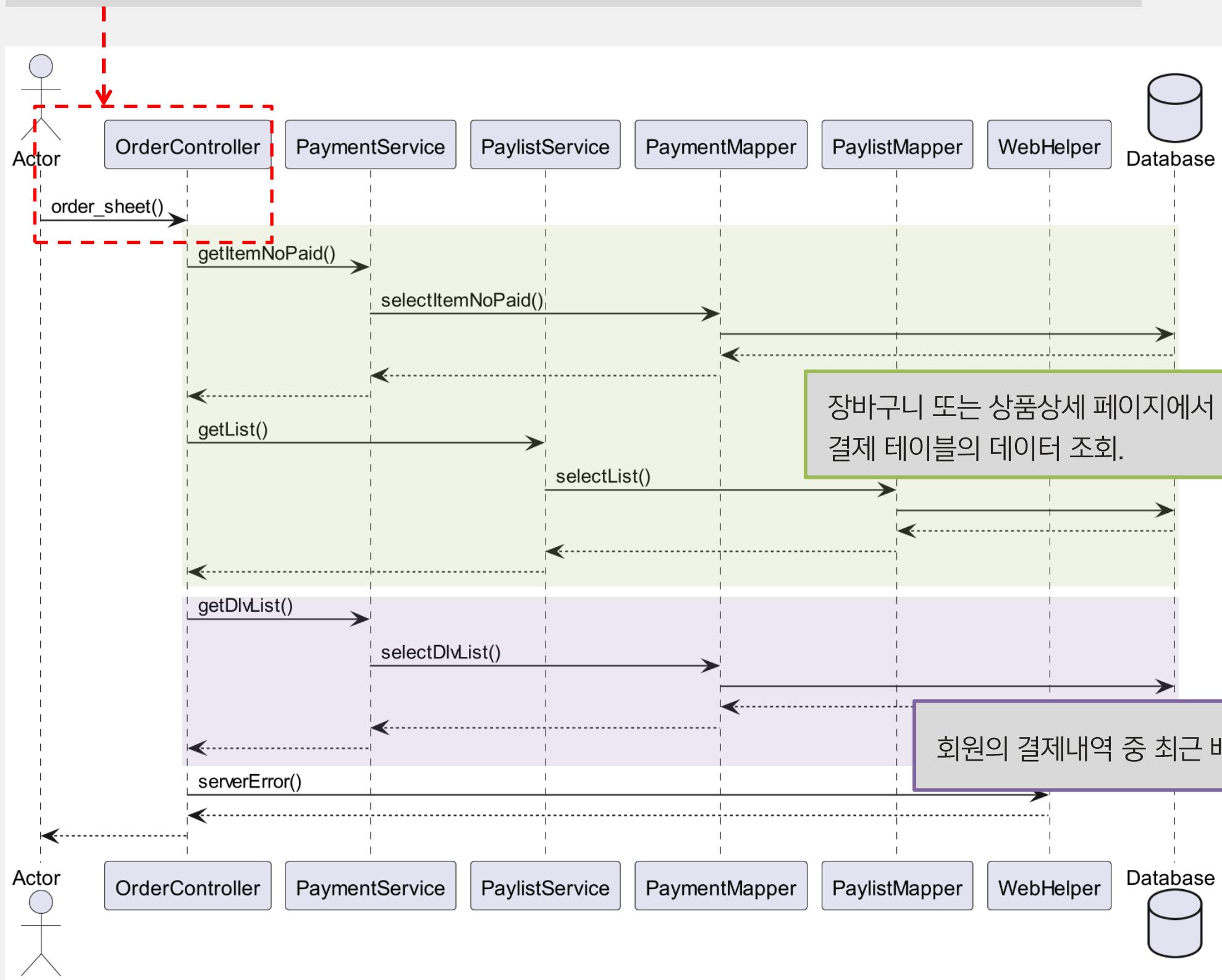
# 구현 기능 - 결제 프로세스



4-2

# 구현 기능 - 결제페이지 (1)

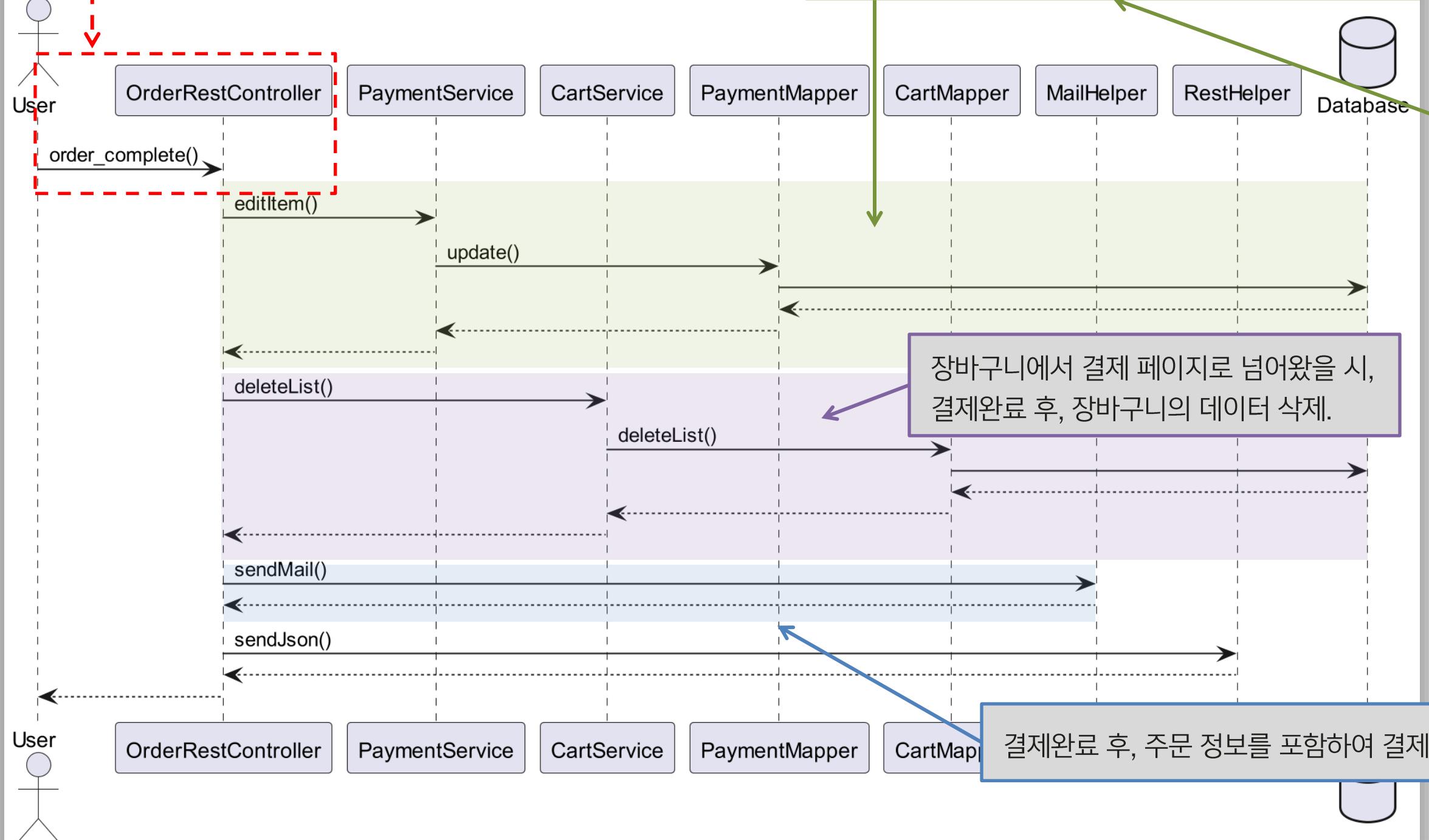
주문·결제 페이지를 구성하는데 필요한 데이터를 조회하여 모델에 추가하고 해당 페이지로 이동.



4-2

# 구현 기능 - 결제페이지 (2)

결제 하기 기능의 RestController



주문·결제

장바구니
 주문·결제
 주문 원료

---

제품

	ZV-E10M2K (white)	1,490,000	1	1,490,000
	ILCE-9M3	7,980,000	1	7,980,000

가격

수량

합계

---

주문자 정보

이름: 이승현  
이메일: hyeon970315@gmail.com  
휴대폰 번호: 01045788956

배송지 정보

배송지 선택: 최근 배송지  
수령인 이름: 미입력  
수령인 주소: 미입력  
휴대폰 번호: 미입력  
배송 메모: 미입력

결제가 완료되었습니다.

안녕하세요, 이승현님!

고객님의 주문이 성공적으로 결제되었습니다.  
아래 주문 내역을 확인해 주세요.

주문 상세 정보

주문 번호: 20241212174638000065  
주문 상품: ZV-E10M2 (1개), FDR-AX700 (1개)  
결제 일자: 2024-12-12 17:46:38  
결제 수단: 네이버페이  
총 주문 수량: 2  
총 결제 금액: 3,839,000원

• 본 메일은 발신전용 메일 이므로 회신을 통한 문의는 처리되지 않습니다.  
• 주문 상품은 빠른 시일 내에 발송될 예정입니다. 배송 진행 상황은 별도의 이메일로 알려드리겠습니다.  
• 추가 문의 사항이 있으시면 언제든지 저희 고객센터로 연락해 주세요.  
감사합니다.

Copyright © Sony Store Corporation. All rights reserved.

TEL : 소니코리아 고객센터 1588-0911 E-MAIL : cshelp@sony.co.kr

4-2

# 구현 기능 - 주문 조회

**주문/배송 조회**

진행 중인 주문

상태	개수
입금대기	0건
결제완료	6건
배송준비	0건
배송중	0건
배송완료	0건

· 구매확정이 완료된 주문은 진행 중인 주문에 포함되지 않으며, 진행 상태에 따라 배송지 변경, 취소, 교환 반품 신청이 가능합니다.  
 · 자사 택배가 아닌 기타 운송수단(택, 화물배송)의 경우는 임의 송장번호가 적용되어 있습니다.  
 · 기타 운송수단의 경우는 고객센터(1588-0911)를 통해 배송 정보 문의 바랍니다.

조회 버튼 클릭 시, 조회기간 내의 주문 내역 조회.

주문 내역

주문날짜/번호	제품	수량	처리상태
2024-12-15 20:46:46 <a href="#">20241215204646000078</a>	ZV-E10M2K (black)	1개	결제완료
2024-12-15 20:41:23 <a href="#">20241215204123000077</a>	ZV-E10M2K (white)	1개	결제완료
2024-12-15 20:41:23	ZV-E10M2K	1개	결제완료

3개월 | 6개월 | **1년** | 2023-12-15 ~ 2024-12-15 | 조회

주문 취소 0건 | 교환 반품 0건

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

order\_list.html – JavaScript

```
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.

```
@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);
```

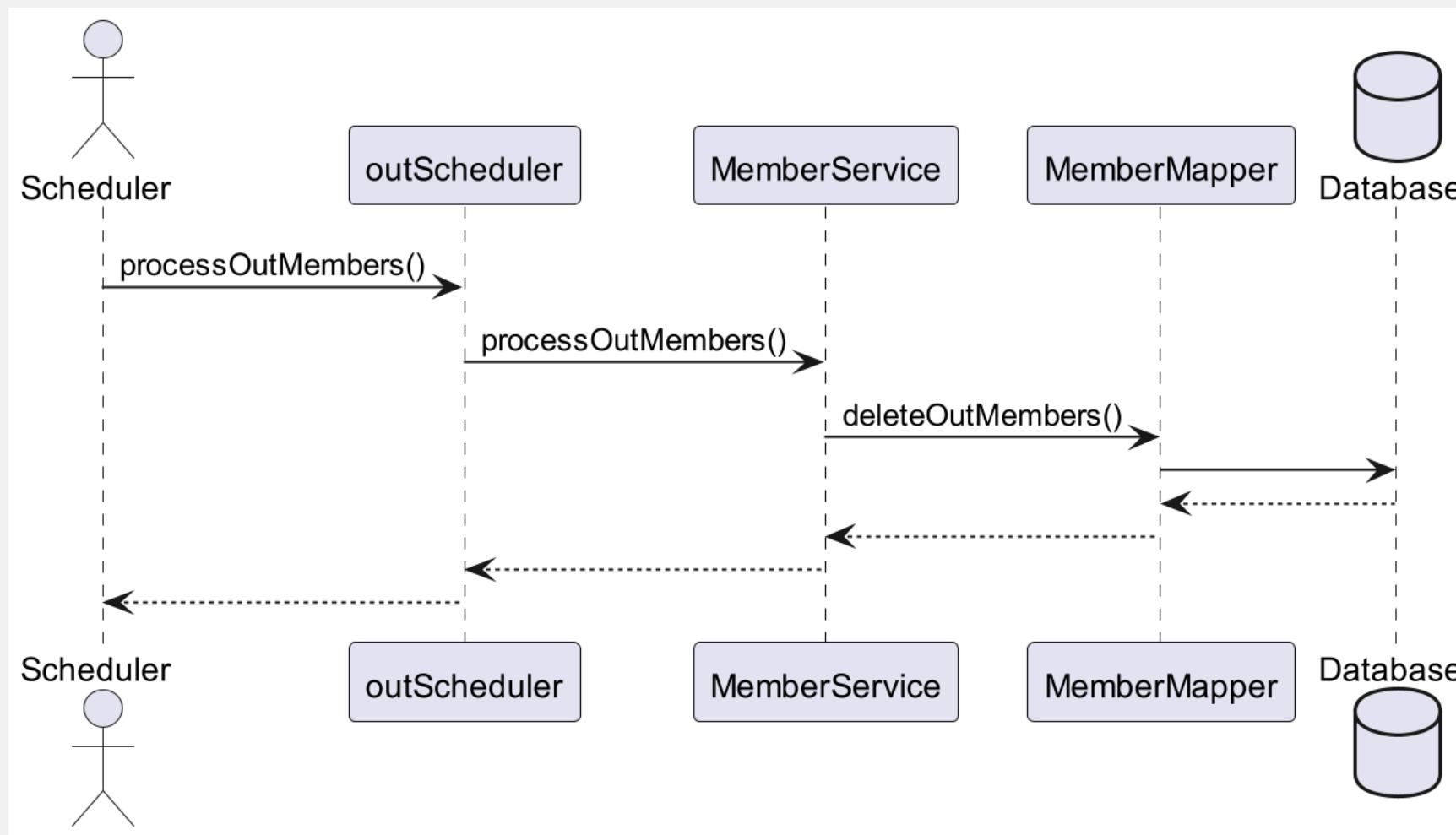
PaylistMapper.java

↑ 조회기간 내의 주문내역을 조회하는 Mapper

# 구현 기능 - 스케줄러

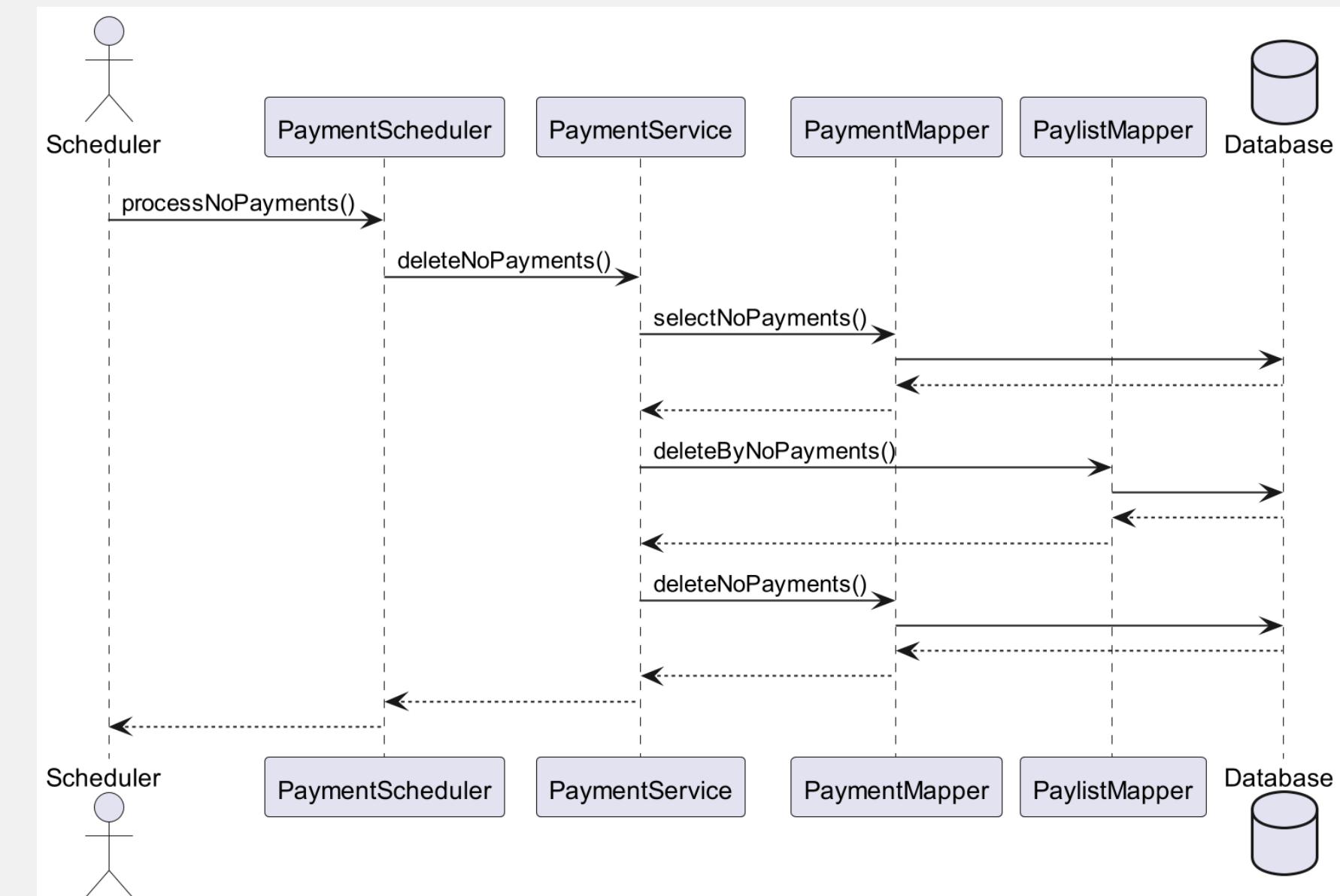
## 회원 탈퇴

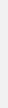
- 회원탈퇴 시 세션의 비밀번호와 비밀번호 입력값이 일치하면 isout = “Y”로 값 수정
- 스케줄러를 이용해 매주 월요일 00시에 isout이 “Y”로 수정된 후, 7일이 지난 데이터 삭제



## 결제 취소

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





# 프로젝트 후기

## 재한

데이터베이스에서 데이터를 불러와 페이지들을 구현하는 게 생각보다 많이 힘들었습니다.

처음에 구상했던 부분과 실제 구현한 부분이 달라서 많은 수정이 있었는데 이를 계기로 기획 단계의 중요성을 다시 한번 깨달았습니다.

팀원과 함께 Git을 사용하면서 코드를 하나 수정하는 것과 다른 사람의 코드를 이해하는 데 많은 의사소통이 필요했던 것 같습니다.

소프트웨어를 분석, 설계, 구현 까지 전부 함께 하니 서로 정도 많이 들었고 유익하고 값진 시간이었습니다.

## 진수

기획부터 작업 기간과 작업 내용, 과정을 문서로 작업하며 사이트 하나를 통째로 구현한 것은 처음이었습니다.

처음에는 실무에서 하게 될 팀프로젝트의 과정을 먼저 경험해보고 덜 해매자. 해서 시작한 팀 작업이었지만 역할을 나누어 작업 기간을 나누었더니 책임감이 생겼고, 작업 내용과 과정을 기술하니 현재 진행 상태 파악과 이슈가 발생했을 때 문제점을 찾아내는데 큰 도움이 되었습니다. 그리고 사이트를 처음부터 끝까지 만들면서 기능별로 배웠던 코드와 만들었던 코드들, 사용할 일이 있을 것 같아 보관해 두었던 코드들을 모아 하나의 파일이 되는 작업이 재미있었습니다. 문서 작업을 하면서 그저 하라고 해서, 좋다고 해서 했었던 스프링의 구조가 점차 이해되었고, 코드 한 줄 한 줄의 의미를 되새길 수 있어 많은 공부가 되었습니다. 이전에는 자바와 데이터베이스, 그리고 프론트 단의 작업을 배우긴 했지만 저에겐 모두 별개의 느낌이었습니다. 자바에서 클래스를 만들 수는 있었지만 그걸 HTML파일에서 어떻게 사용하지? DB에 INSERT는 할 수 있었지만 그걸 자바파일에서 어떻게 사용하지? 하고 생각만 하는 상태였는데 서로를 연결하면서 궁금증이 해소되고 성취감을 느끼며 더욱 흥미를 느낄 수 있었습니다.

## 승현

이전에는 지금까지 배운 컴퓨터 언어 수준으로 내가 과연 뭘 할 수 있을까 막막했는데, Spring MVC 프로세스를 이해하게 되면서 프로그램의 구조와 흐름을 파악할 수 있게 되었습니다. 프로젝트를 진행하면서 직접 구현해보니, 처음에는 복잡하게만 느껴졌던 MVC 패턴이 여러 작업을 효과적으로 관리해줄 수 있다는 것을 알게 되었고, 분리된 로직이 코드간의 독립성, 결합성, 용이성을 얼마나 잘 높여주는지 확실히 느낄 수 있었습니다. 또한, 결제 프로세스를 설계하면서 제가 생각한 로직대로 맵퍼, 서비스, 컨트롤러를 한 단계 한 단계 진행할 때, 코드의 흐름과 제어에 매력을 느낄 수 있었습니다. 쇼핑몰 웹 페이지의 기능을 구현해 보고 나니, 이제 다른 페이지도 만들어보고 싶다는 생각도 들었고, 자신감과 성취감을 얻을 수 있는 유익한 시간이었습니다. 더 공부하여 코드를 조금 더 효율적으로 만들 수 있는 개발자가 되고 싶다는 생각이 들었습니다.

감사합니다