

꿀조합 팩토리

HONEY COMBO FACTORY

버그잡자 찍찍찍



Ver 6.28.38
2025.04.11

김동현(팀장), 정규민, 고강희, 장지현

프로젝트 시연



팀 구성 및 역할



김동현 (팀장)

Service (파트장)

- Map API
- GPS API
- [블로그](#)



정규민

Front (파트장)

- 카카오 로그인 API
- 별점 플러그인
- 이벤트 트리거
- [블로그](#)



고강희

DB (파트장)

- 네이버 로그인 API
- 이메일 API
- [블로그](#)



장지현

DB

- 웹 크롤링
- 회원 정보 담당
- [블로그](#)

목 차

- 01_ 프로젝트 개요
- 02_ 설계
- 03_ 기능
- 04_ API
- 05_ 오류 원인 분석 및 해결 방안
- 06_ 향후 개발 과제 및 소감



HONEY-COMBO
CONVENIENCE STORE COMBO SHOPPING MALL

01 프로젝트 개요

- 기획 배경 및 목적, 기대효과
- 기능 요약 및 개발 환경
- 개발 일정
- 버전 히스토리

01_ 프로젝트 개요

프로젝트 개요 1/2

기획 배경

- 각각의 앱을 사용하는 불편함
- 다양한 브랜드를 사용하고자 하는 소비자 니즈 증가
- 통합 플랫폼으로 확장할 필요성

기획 목적

- 상품 정보의 통합
- 브랜드 구분없는 세트 구매 가능
- 사용자의 구매 편의성과 선택 폭 확대

기대 효과

- 클라이언트 시간 절약, 맞춤형 구매
- 브랜드 상품의 노출, 새로운 유입 경로
- 플랫폼 사용자 재방문율 증가
- 시장 전반 브랜드 간 경쟁과 협업 촉진

01_ 프로젝트 개요

프로젝트 개요 2/2

기능 요약

- 상품 정보 통합 검색 및 필터링
- 브랜드 혼합 세트 상품 제공
- 장바구니
- 리뷰 및 별점

개발 환경

언어



DB



개발
도구

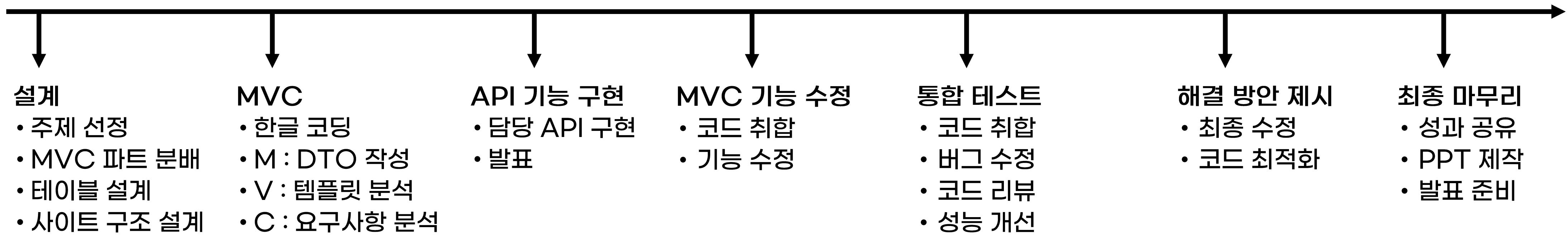


협업



01_프로젝트 개요

개발 일정



3.12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
설계															
	MVC				취합		MVC					API 기능 구현			
28	29	30	31	4.1	2	3	4	5	6	7	8	9	10	11	12
발표															
	MVC 기능 수정			통합 테스트				해결 방안 제시							
									최종 마무리						
										PPT		발표			

Ver 6.28.38

6 : 코드 취합

28 : Front 업데이트

38 : Service 업데이트

Latest Release : 2025.04.09

02 프로젝트 설계

- 타겟 사이트
- 유저 요구사항 분석
- 기능 리스트
- 페이지 리스트
- Service 리스트
- 정보 구조도
- ERD
- USER FLOW
- LOGIC PROCESS
- 와이어 프레임

02 _ 프로젝트 설계

타겟 사이트



GS25



OLIVE O YOUNG

편의점 통합 플랫폼

주요 편의점 브랜드를 포함하여

소비자 선호도가 높은 Subway와 Olive Young을 참고
각자의 사용 경험을 바탕으로 소비자 중심의 플랫폼 개발 진행

02 _ 프로젝트 설계

유저 요구사항 분석 1/2

기능명	요구사항 설명	비고	구현유무
회원가입	사용자 정보 입력받아 회원으로 등록	선택사항 : 주소	o
로그인	사용자 로그인정보 입력 시 인증 수행	-	o
소셜 로그인	소셜 네트워크로 로그인 수행	카카오, 네이버 로그인	o
회원정보수정	사용자의 정보수정	비밀번호 변경, 재설정	o
로그아웃	사용자 로그인 상태를 해제	-	o
상품조회	사용자의 상품조회	카테고리, 정렬	o
상품검색	사용자의 상품검색	개별, 꿀조합 상품	o
장바구니담기	사용자가 구매하려는 상품 장바구니에 저장	-	o

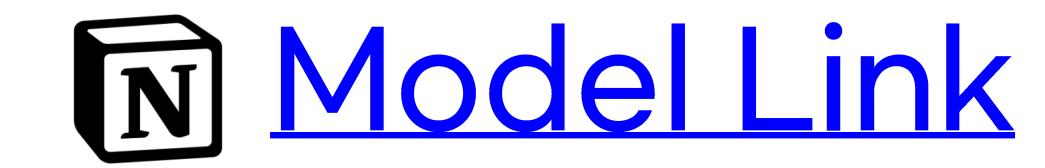
02 _ 프로젝트 설계

유저 요구사항 분석 2/2

기능명	요구사항 설명	비고	구현유무
주문	사용자가 구매하려는 상품 주문	꿀조합	X
게시판 글작성	사용자가 게시판에 글을 입력, 수정, 삭제	꿀조합 게시판	X
좋아요 기능	사용자가 꿀조합 상품에 좋아요/좋아요 취소	-	O
결제	사용자의 상품 결제	API	X
별점	사용자가 상품에 별점 부여	5점(플러그인)	O
이메일 수신	사용자가 결제한 주문내역 수신	API	O
지도	사용자 현재 위치 주변 편의점 위치 제공	API	O
주소	회원가입 시 주소 검색 기능 제공	API	O

02 _ 프로젝트 설계

기능 리스트 1/2



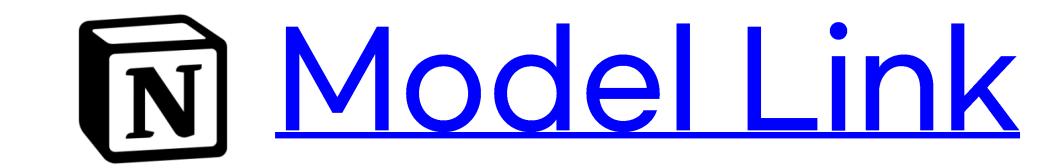
	메서드명	기능
MEMBER	selectAll	회원 조회
	selectOne	로그인
		회원 조회
		내 정보 보기
		아이디 찾기
		비밀번호 찾기
		이메일 중복 검사
		휴대폰 중복 검사
	insert	회원가입
	update	비밀번호 변경
		회원 탈퇴
		내 정보 수정

	메서드명	기능
PRODUCT SINGLE	selectAll	상품 조회
		상품 검색
		인기순 정렬
		가격순 정렬
		카테고리순 정렬
		재고순 정렬
	selectOne	상품 상세 정보
	insert	개별 상품 추가

	메서드명	기능
PRODUCT COMBO	selectAll	인기순 전체 조회
		가격순 전체 조회
		카테고리별 전체 조회
		카테고리별 조회
	selectOne	상품 검색(가격순)
	insert	상품 상세 조회
	메서드명	기능
PURCHASE	selectAll	주문 목록 조회
	selectOne	주문정보 1개 조회
	insert	주문 추가
	delete	주문 취소

02 _ 프로젝트 설계

기능 리스트 2/2



	메서드명	기능
BOARD COMBO LIKED	selectAll	좋아요 개수 게시물 조회
	selectOne	좋아요 한 회원
	insert	좋아요 추가
	delete	좋아요 삭제

	메서드명	기능
REVIEW	selectAll	리뷰 목록 조회
		리뷰 상세 조회
		하나의 상품 리뷰 조회
	selectOne	리뷰 상세 조회
	insert	리뷰 작성
	update	리뷰 수정
	delete	리뷰 삭제

	메서드명	기능
PURCHASE DETAIL	selectAll	상품 상세정보 조회
		상품 상세정보 추가

	메서드명	기능
PRODUCT COMBO COMPONENT	selectAll	구성품 전체 조회
	insert	구성품 추가
	delete	구성품 삭제

	메서드명	기능
BOARD COMBO	selectAll	게시판 글 조회
		게시판 글 검색
		회원 글 목록 조회
	selectOne	게시판 상세글 조회
	insert	게시판 글 작성

02 _ 프로젝트 설계

페이지 리스트



[View Link](#)

메인	main.do
로그인	login.do
아이디/비밀번호 찾기	findAccount.do
회원가입	join.do
내정보	myInfo.do
장바구니	cart.do
꿀조합 게시판	comboBoard.do
상품 상세	productDetail.do
글 상세	boardDetail.do
글 등록	updateBoard.do
주문 상세	purchaseDetail.do
CU 상품	CUProduct.do
GS 상품	GSProduct.do
꿀조합 상품	comboProduct.do
에러	error.do

02 _ 프로젝트 설계

Service 리스트

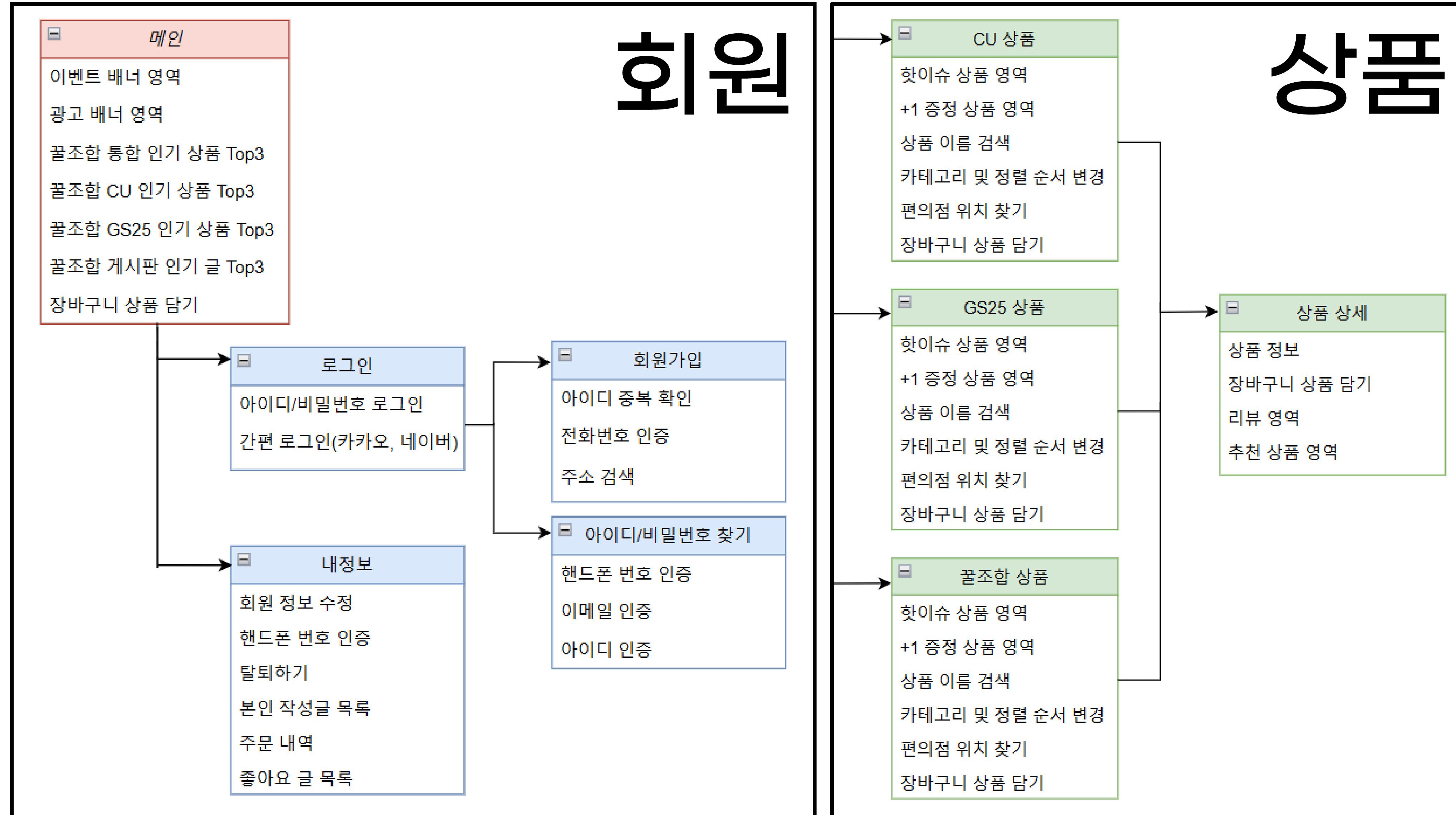


[Controller Link](#)

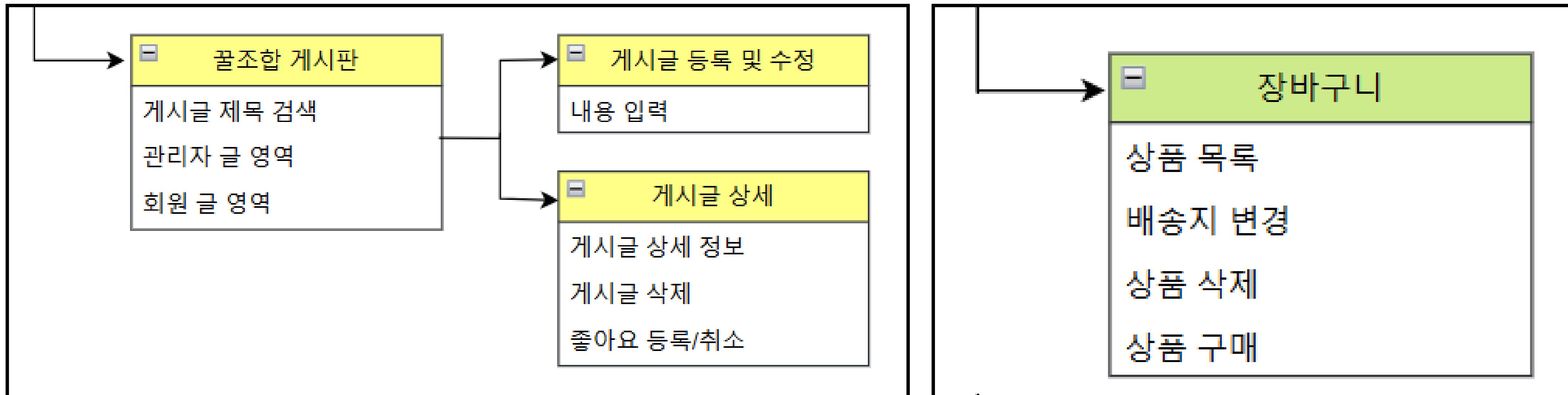
장바구니 추가	AddCartProductServlet	게시글 수정	MemberUpdateBoardServlet
로그인	LoginServlet	좋아요 등록 / 취소	MemberLikedBoardServlet
아이디 찾기	FindIdServlet	게시글 등록	MemberInsertBoardServlet
비밀번호 찾기	FindPasswordServlet	댓글 수정	ProductDetailUpdateReviewServlet
비밀번호 재설정	FindPasswordAndUpdatePasswordServlet	댓글 삭제	ProductDetailDeleteReviewServlet
아이디 중복검사	CheckJoinIdServlet	댓글 등록	ProductDetailInsertReviewServlet
회원가입	MemberJoinServlet	리뷰 정보	ProductDetailReviewServlet
본인이 작성한 글	MyInfoMyBoardServlet	장바구니	AddCartProductServlet
주문 내역	MyInfoMyPurchaseServlet	CU 정렬 조건	OrderCuProductServlet
좋아요 글 내역	MyInfoMyLikedBoardServlet	CU 핫이슈 상품	CuHotIssueProductServlet
내정보 변경	UpdateMyInfoServlet	CU 증정상품	CuPlusOneProductServlet
회원탈퇴	WithdrawServlet	CU 검색	SearchCuProductServlet
본인확인	ConfirmPasswordServlet	GS 정렬 조건	OrderGsProductServlet
비밀번호 재설정	ChangePasswordServlet	GS 핫이슈 상품	GsHotIssueProductServlet
장바구니 삭제	DeleteCartProductServlet	GS 증정상품	GsPlusOneProductServlet
장바구니 수량 변경	ChangeCartProductCountServlet	GS 검색	SearchGsProductServlet
장바구니 구매	BuyCartProductServlet	꿀조합 정렬 조건	OrderProductComboServlet
좋아요 여부 확인	BoardComboLikedServlet	꿀조합 검색	SearchComboProductServlet
게시글 삭제	MemberDeleteBoardServlet		

02_프로젝트 설계

정보 구조도 1/2



정보 구조도 2/2

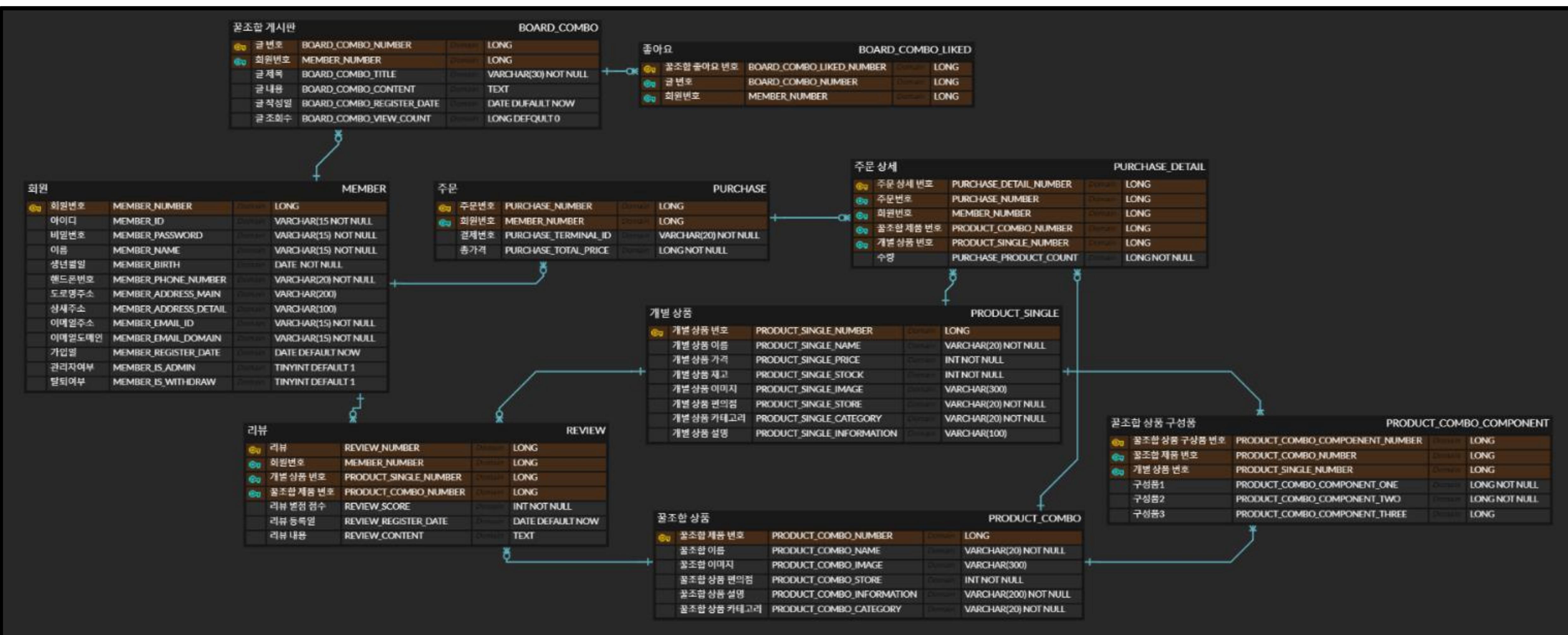


게시판

장바구니

02_프로젝트 설계

ERD 1/5



02_프로젝트 설계

ERD 2/5

꿀조합 상품		PRODUCT_COMBO	
PK	꿀조합 제품 번호	PRODUCT_COMBO_NUMBER	Domain LONG
	꿀조합 이름	PRODUCT_COMBO_NAME	Domain VARCHAR(20) NOT NULL
	꿀조합 이미지	PRODUCT_COMBO_IMAGE	Domain VARCHAR(300)
	꿀조합 상품 판의점	PRODUCT_COMBO_STORE	Domain INT NOT NULL
	꿀조합 상품 설명	PRODUCT_COMBO_INFORMATION	Domain VARCHAR(200) NOT NULL
	꿀조합 상품 카테고리	PRODUCT_COMBO_CATEGORY	Domain VARCHAR(20) NOT NULL

꿀조합 상품 구성품		PRODUCT_COMBO_COMPONENT	
PK	꿀조합 상품 구상품 번호	PRODUCT_COMBO_COMPONENT_NUMBER	Domain LONG
PK	꿀조합 제품 번호	PRODUCT_COMBO_NUMBER	Domain LONG
PK	개별 상품 번호	PRODUCT_SINGLE_NUMBER	Domain LONG
	구성품1	PRODUCT_COMBO_COMPONENT_ONE	Domain LONG NOT NULL
	구성품2	PRODUCT_COMBO_COMPONENT_TWO	Domain LONG NOT NULL
	구성품3	PRODUCT_COMBO_COMPONENT_THREE	Domain LONG

ERD 3/5

? 고민

꿀조합 상품은 각 편의점의 개별 상품으로 구성됨

1. 꿀조합 상품의 개별 상품은 어떻게 저장해야 하는가?
2. 꿀조합 상품의 개별 상품 개수는 다른데
어떻게 저장해야 하는가?

! 방안

1. 꿀조합 구성품을 담을 테이블을 새로 생성
2. 조합이기 때문에
최소 두 개는 NOT NULL 제약조건을 설정하고
나머지는 NULL 허용

02_프로젝트 설계

ERD 4/5

주문			PURCHASE	
PK	주문번호	PURCHASE_NUMBER	Domain	LONG
PK	회원번호	MEMBER_NUMBER	Domain	LONG
	결제번호	PURCHASE_TERMINAL_ID	Domain	VARCHAR(20) NOT NULL
	총가격	PURCHASE_TOTAL_PRICE	Domain	LONG NOT NULL

주문 상세			PURCHASE_DETAIL	
PK	주문 상세 번호	PURCHASE_DETAIL_NUMBER	Domain	LONG
PK	주문번호	PURCHASE_NUMBER	Domain	LONG
PK	회원번호	MEMBER_NUMBER	Domain	LONG
PK	풀조합 제품 번호	PRODUCT_COMBO_NUMBER	Domain	LONG
PK	개별 상품 번호	PRODUCT_SINGLE_NUMBER	Domain	LONG
	수량	PURCHASE_PRODUCT_COUNT	Domain	LONG NOT NULL

ERD 5/5

? 고민

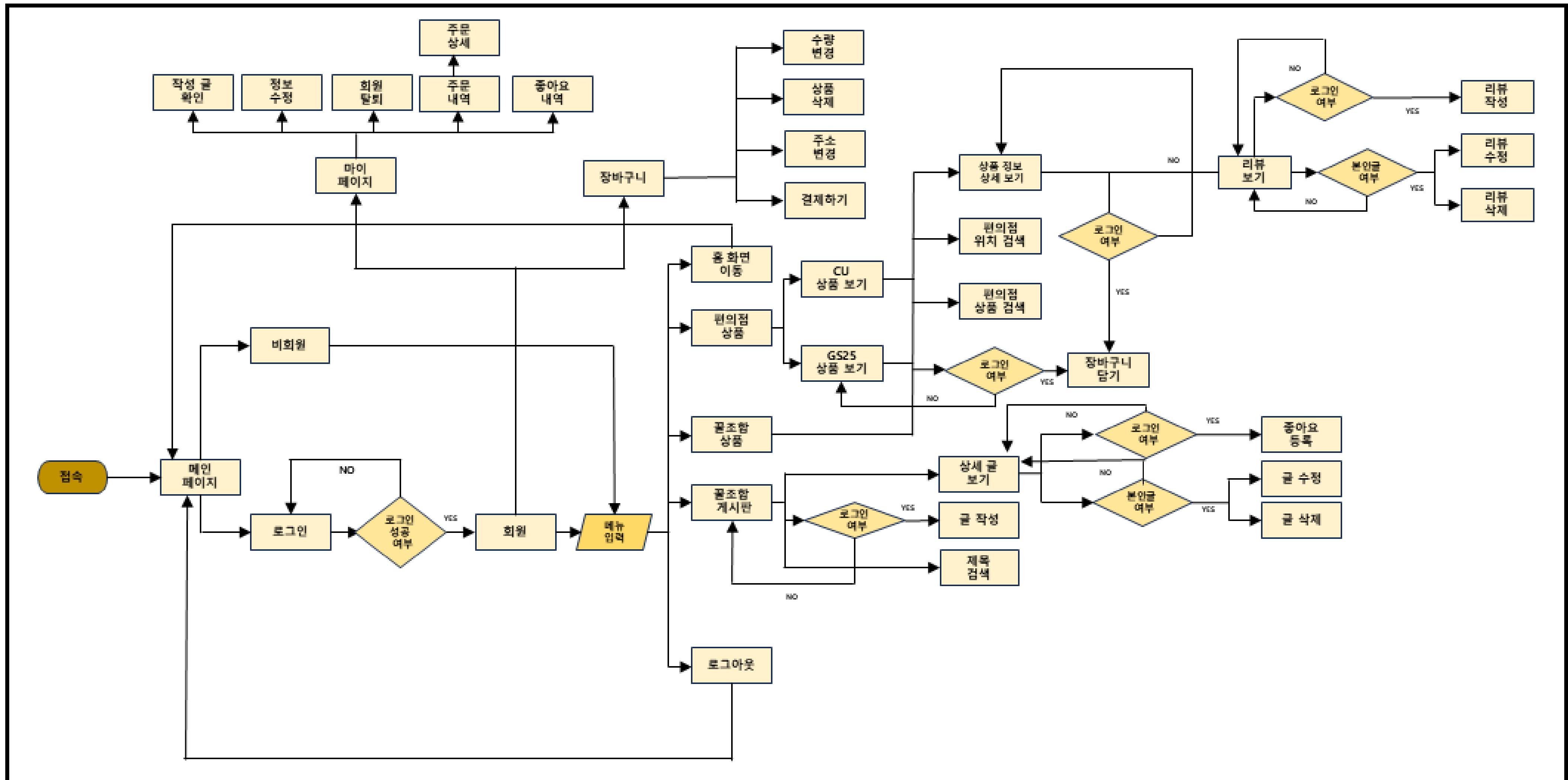
상품 구매 시 여러 개의 상품을 동시에 구매할 경우
각 상품의 정보는 어떻게 저장해야 하는가?

! 방안

주문 테이블에 한 번 저장한 후
주문상세 테이블에 상품을 각각 저장 → 주문번호를 FK로 설정

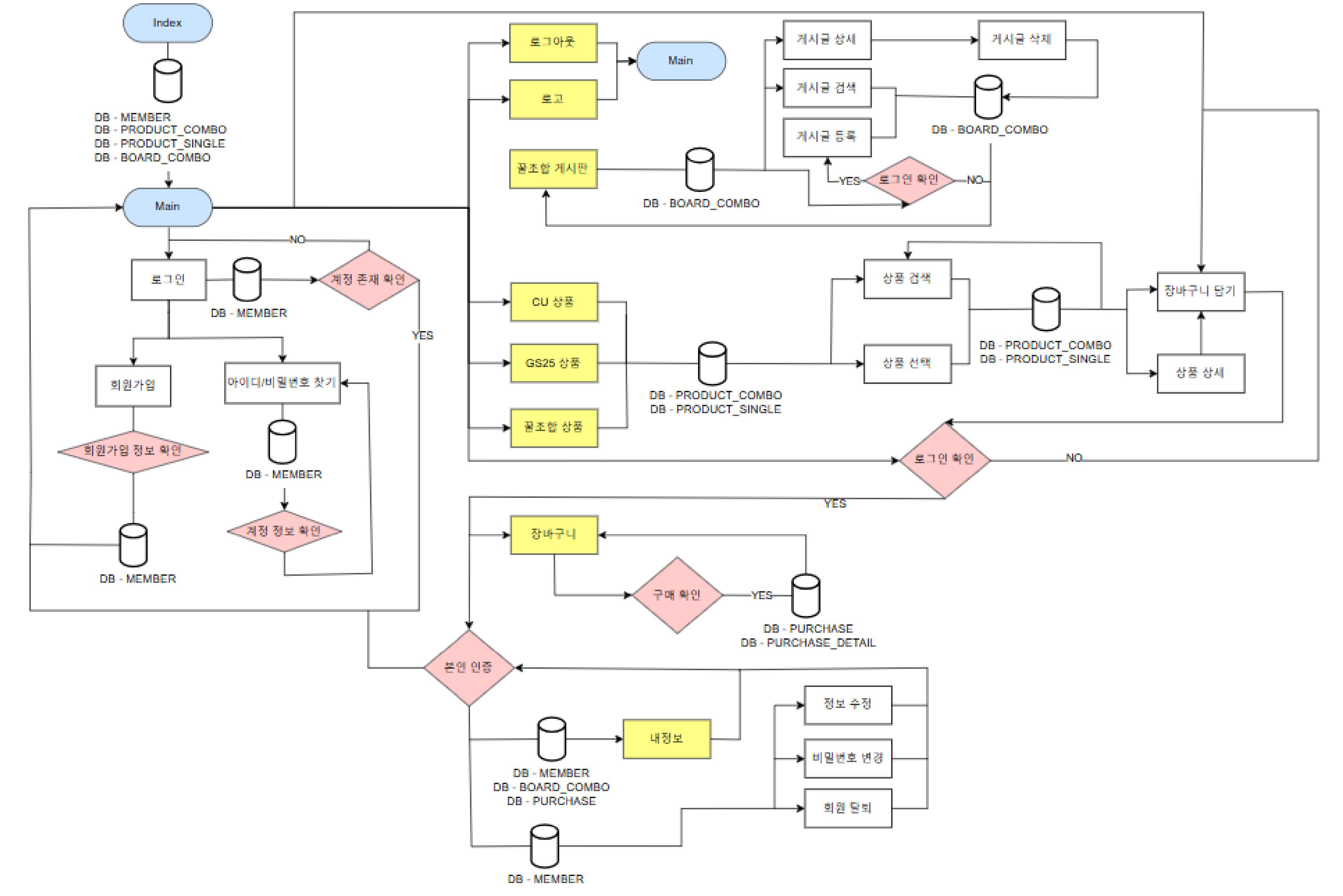
02_프로젝트 설계

USER FLOW



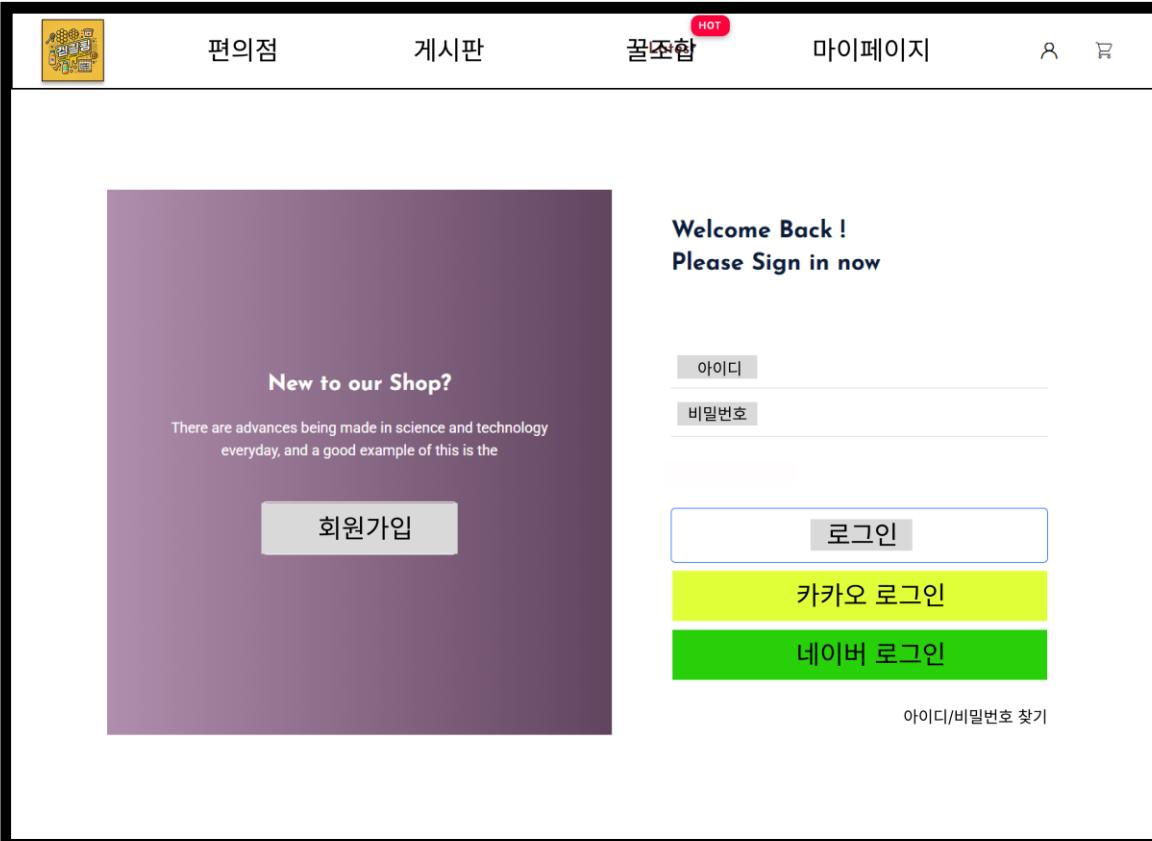
02_프로젝트 설계

LOGIC PROCESS



02_프로젝트 설계

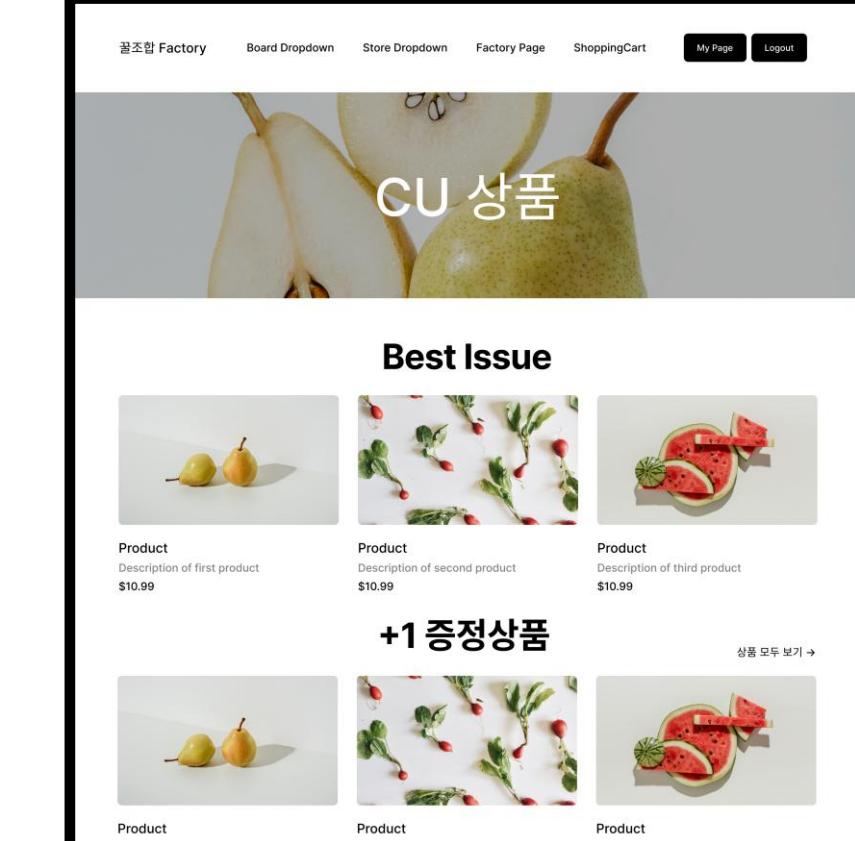
와이어 프레임



로그인 페이지



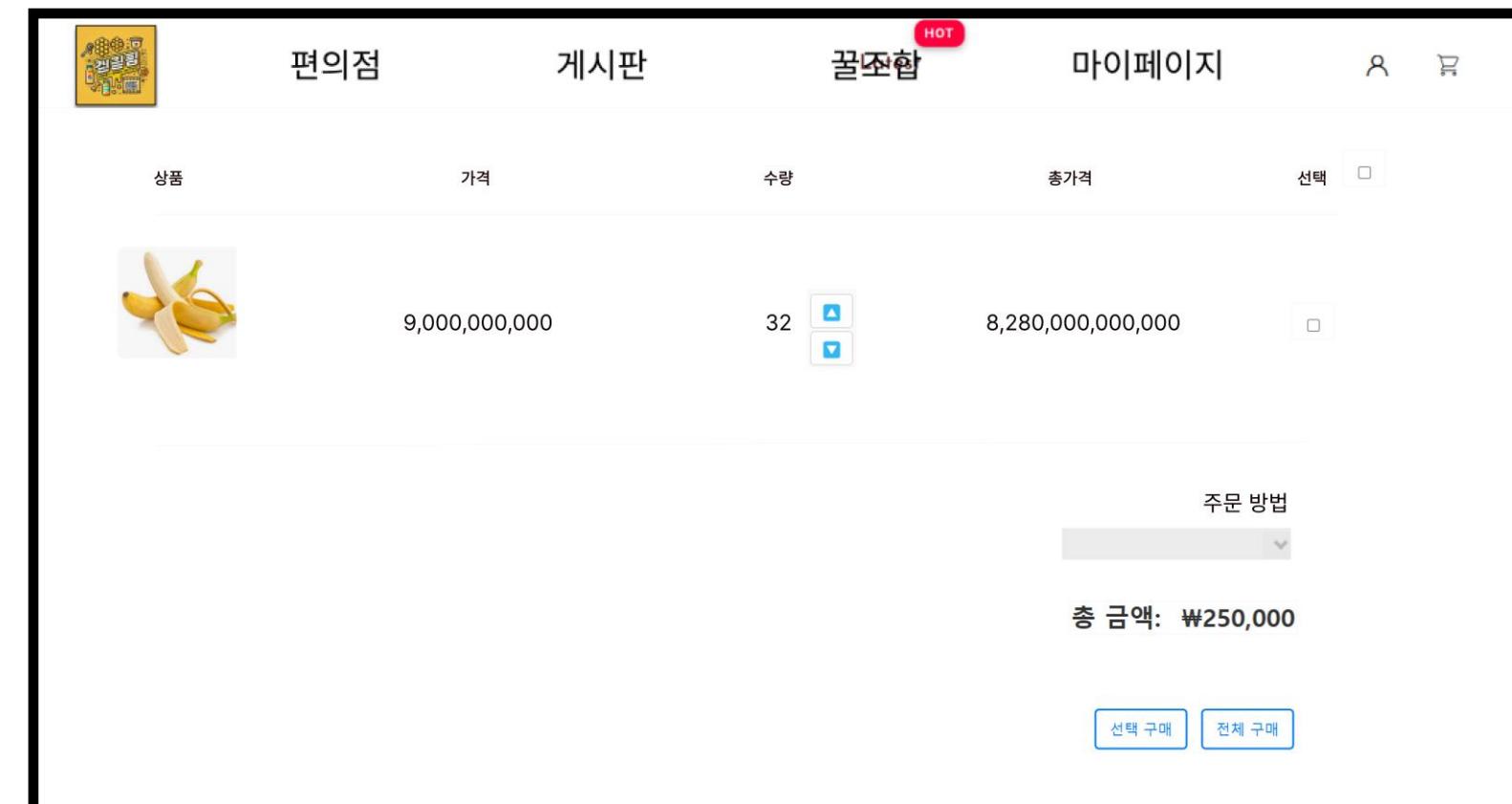
게시판 페이지



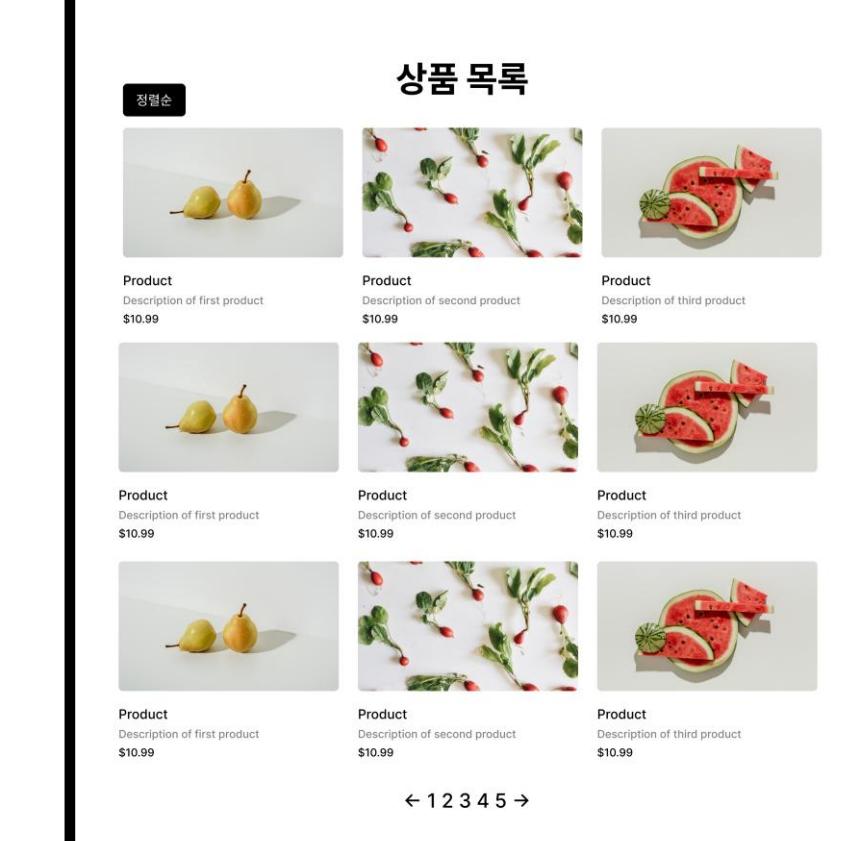
검색창 엔트리
검색창 입력란 : _____
모든 상품
카테고리 1
카테고리 2



회원가입 페이지



장바구니 페이지



상품 판매 페이지

03 프로젝트 가능

- 장바구니
- 게시글 좋아요
- Crawling
- 주문 상세 내역 조회
- 페이지네이션
- 아이디 중복 검사
- 페이지 이동
- 검색, 정렬
- 카테고리, 정렬

04 API 문서

- 플러그인
- 라이브러리
- API

03 _ 프로젝트 기능

01. 장바구니

img-cap 클래스 요소에 클릭 이벤트 등록



장바구니 담기

해태)레몬에이드P350ml

2000원

```
47      // 장바구니 담기 기능 이벤트 위임
48•     $(document).on("click", ".img-cap", function() {
49         let productSingleNumber = $(this).data("product-id");
50         insertCart(productSingleNumber, 1, false);
51     });

```

컴포넌트화 한 메서드 실행

```
320• <div class="img-cap" data-product-id="` + productData.productSingleNumber + '`">
321     <span>장바구니 담기</span>
322 </div>
```

장바구니 담기 클릭 시

```
397 // 장바구니 상품 담기 기능
398 const insertCart = (productNumber, cartProductCount, isComboProduct) => {
399     console.log("CU 장바구니 상품 담기 실행");
400     if (!loginedMemberNumber) { // 로그인하지 않은 경우
401         console.log("로그인 없이 장바구니 담기 요청");
402         alert("로그인이 필요한 기능입니다!");
403         return;
404     }
405     console.log("장바구니 담을 상품 번호 : [" + productNumber + "]");
406     console.log("장바구니 담을 상품 수량 : [" + cartProductCount + "]");
407     console.log("장바구니 담을 상품 꿀조합 여부 : [" + isComboProduct + "]");
```

로그인 여부 검사

로그 출력으로 디버깅 용이

```
409$.ajax({  
410    type: "POST", // 방식  
411    url: insertCartUrl, // 찾아갈 주소  
412    data: { // 보낼 값  
413        productNumber: productNumber,  
414        cartProductCount: cartProductCount,  
415        isComboProduct: isComboProduct  
416    },  
417    dataType: "text", // 받을 타입  
418    success: (response) => { // 성공 시 처리  
419        if (response === "true") { // 잘 담겼다면  
420            console.log("장바구니 상품 담기 성공");  
421            alert("선택하신 상품이 장바구니에 담겼습니다.");  
422        } else { // 안 담겼다면  
423            console.log("장바구니 상품 담기 실패");  
424            alert("선택하신 상품을 장바구니에 담기 실패했습니다.");  
425            location.href = "error.do";  
426        }  
427    },  
428    error: (xhr, status, error) => { // 에러 처리  
429        console.error("AJAX 요청 에러 발생", xhr.status, status, error);  
430        alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");  
431        location.href = "error.do";  
432    }  
433});  
434};
```

비동기로 백단에

1. 상품 번호
2. 장바구니 담을 개수
3. 꿀조합 상품 여부

값들 전달

localhost:8088 내용:

선택하신 상품이 장바구니에 담겼습니다.

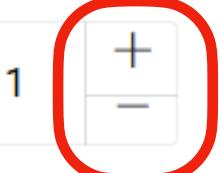
확인

```
43     int productNumber = Integer.parseInt(request.getParameter("productNumber"));
44     int cartProductCount = Integer.parseInt(request.getParameter("cartProductCount"));
45     boolean isComboProduct = Boolean.parseBoolean(request.getParameter("isComboProduct"));
46
47     boolean alreadyIn = false;
48
49     for (Map<String, Object> cartItem : shoppingCart) {
50         if ((int) cartItem.get("productNumber") == productNumber && (boolean) cartItem.get("isComboProduct") == isComboProduct) {
51             int currentCount = (int) cartItem.get("cartProductCount");
52             cartItem.put("cartProductCount", currentCount + cartProductCount);
53             alreadyIn = true;
54             break;
55         }
56     }
57
58     if (!alreadyIn) {
59         Map<String, Object> newCartItem = new HashMap<>();
60         newCartItem.put("productNumber", productNumber);
61         newCartItem.put("cartProductCount", cartProductCount);
62         newCartItem.put("isComboProduct", isComboProduct);
63         shoppingCart.add(newCartItem);
64     }
}
```

사용자가 요청한 상품 번호를 받은 후
장바구니에서의 존재 유무 파악

존재할 경우 새로 담지 않고 수량 증가

존재하지 않는 상품이라면
장바구니에 추가

장바구니 상품	가격	수량	총 가격	
	해태)레몬에이드P350ml	2000	1 	2000 원 <input checked="" type="checkbox"/>
	김)돈까스샐러드김밥	3400	1 	3400 원 <input checked="" type="checkbox"/>
	코카)스프라이트P500ml	2100	1 	2100 원 <input type="checkbox"/>
총 구매 가격			5400 원	

장바구니 상품 개수 증가/감소 가능

```
143 // 장바구니 상품 개수 증가/감소 기능
144 const changeCartProductCount = (cartProductNumber, cartProductCondition) => {
145     // 장바구니 증감할 개수
146     const productCount = 1;
147
148     console.log("개수 변경 장바구니 상품 실행");
149     console.log("개수 변경 장바구니 상품 번호 : [" + cartProductNumber + "]");
150     console.log("개수 변경 장바구니 상품 조건 : [" + cartProductCondition + "]");
151
152     // 장바구니 최소/최대 개수 검사 함수 호출
153     if (checkMaxMinProductCount(cartProductNumber, cartProductCondition, productCount)) {
154         return;
155     }
```

```
162 // 장바구니 최소/최대 개수 검사 기능
163 const checkMaxMinProductCount = (cartProductNumber, cartProductCondition, productCount) => {
164     let flag = false;
165     const nowCount = parseInt($("#count-" + cartProductNumber).val());
166
167     if (cartProductCondition === "upCartProductCount") { // 상품 수량 증가라면
168         const maxValue = parseInt($("#count-" + cartProductNumber).prop("max"));
169         if (nowCount + parseInt(productCount) > maxValue) { // 재고보다 많아진다면
170             console.log("재고보다 많을 수 없음");
171             flag = true;
172         }
173     } else { // 상품 수량 감소라면
174         if (nowCount - parseInt(productCount) < 1) { // 1보다 작아진다면
175             console.log("1보다 작을 수 없음");
176             flag = true;
177         }
178     }
179 }
180 return flag;
181 }
```

0 < 장바구니 수 < 재고 수

```
157$.ajax({  
158    type: "POST", // 방식  
159    url: changeCartProductCountUrl, // 찾아갈 주소  
160    data: { // 보낼 값  
161        cartProductNumber: cartProductNumber,  
162        cartProductCondition: cartProductCondition,  
163        productCount: productCount  
164    },  
165    dataType: "text", // 받을 타입  
166    success: (response) => { // 성공적이라면  
167        if (response === "true") { // 상품 개수 변경 성공 시  
168            console.log("장바구니 상품 개수 변경 성공");  
169  
170            // 장바구니 상품 수량 표시 변경 함수 호출  
171            changeInputCartProductCount(cartProductNumber, cartProductCondition, productCount);  
172  
173            // 장바구니 각 상품 구매 가격 변경 함수 호출  
174            calculationPrice(cartProductNumber);  
175  
176            console.log("증감 상품 장바구니 선택 여부 : [" +  
177                $("#productCheckBox-" + cartProductNumber).prop("checked") + "]");  
178            if ($("#productCheckBox-" + cartProductNumber).prop("checked")) { // 선택된 상품이었다면  
179                // 장바구니 상품 가격 총합 계산 함수 호출  
180                calculationTotalAmount();  
181            }  
182        }  
183    }  
184}
```

비동기로 백단에

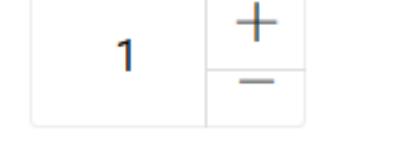
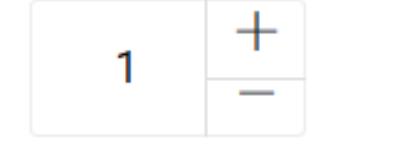
- 1.장바구니 상품 번호
- 2.조건(개수 증가/감소)
- 3.변경할 수량 단위

값들 전달

상품 수량 표시 변경

선택 상품 구매 가격 변경

장바구니 총 구매 가격 변경

장바구니 상품	가격	수량	총 가격	
	해태)레몬에이드P350ml	2000	6  12000 원	<input checked="" type="checkbox"/>
	김)돈까스샐러드김밥	3400	1  3400 원	<input type="checkbox"/>
	코카)스프라이트P500ml	2100	1  2100 원	<input type="checkbox"/>
총 구매 가격			12000 원	

장바구니 상품 개수
변경 확인

장바구니 상품	가격	수량	총 가격				
	2000	<table><tr><td>1</td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/></td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/></td></tr></table>	1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>	2000 원	<input style="border: 2px solid red; width: 20px; height: 20px; border-radius: 50%; padding: 0; margin: 0;" type="checkbox"/>
1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>					
	3400	<table><tr><td>1</td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/></td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/></td></tr></table>	1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>	3400 원	<input checked="" style="border: 2px solid red; width: 20px; height: 20px; border-radius: 50%; padding: 0; margin: 0;" type="checkbox"/>
1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>					
	2100	<table><tr><td>1</td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/></td><td><input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/></td></tr></table>	1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>	2100 원	<input style="border: 2px solid red; width: 20px; height: 20px; border-radius: 50%; padding: 0; margin: 0;" type="checkbox"/>
1	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="+"/>	<input style="width: 20px; height: 20px; border: 1px solid black; border-radius: 50%; padding: 0; margin: 0;" type="button" value="-"/>					
총 구매 가격			5400 원				

장바구니 상품 선택

```
196 // 장바구니 상품 삭제 기능
197 const deleteCartProduct = () => {
198     console.log("장바구니 상품 삭제 실행");
199
200     let cartProductNumberDatas = ""; // 선택된 상품 번호들을 저장할 문자열
201
202     // 체크된 상품들의 번호 저장
203     $('.productCheckBox:checked').each(function() {
204         const productNumber = $(this).val(); // 상품 번호
205
206         // 상품 번호를 저장 (처음이 아니면 + 추가)
207         if (cartProductNumberDatas.length > 0) {
208             cartProductNumberDatas += "+";
209         }
210
211         cartProductNumberDatas += productNumber;
212     });
213     console.log("체크된 상품들의 번호 정보 : [" + cartProductNumberDatas + "]");
```

선택된 상품 번호
+로 연결하여 저장

```
215$.ajax({  
216    type: "POST", // 방식  
217    url: deleteCartProductUrl, // 찾아갈 주소  
218    data: { cartProductNumberDatas: cartProductNumberDatas }, // 보낼 값  
219    datatype: "text", // 받을 타입  
220    success: (response) => { // 성공적이라면  
221        if (response === "true") { // 상품 삭제 성공 시  
222            console.log("장바구니 상품 삭제 성공");  
223  
224            // "+"를 기준으로 모든 값을 배열로 분리  
225            let deleteParts = cartProductNumberDatas.split("+");  
226            console.log("총 개수:", deleteParts.length); // 몇 개인지 확인  
227  
228            // 배열 반복문으로 행 삭제 처리  
229            deleteParts.forEach((deletePart) => {  
230                // 해당 상품 행 삭제  
231                $("#cartRow-" + deletePart).remove();  
232            });  
233  
234            // 장바구니 상품 가격 총합 계산 함수 호출  
235            calculationTotalAmount();
```

비동기로 백단에
+로 연결된 상품 번호
값 전달

선택된 상품
화면에서 삭제

```
51 String cartProductNumberDatas = request.getParameter("cartProductNumberDatas");
52
53 if (cartProductNumberDatas != null && !cartProductNumberDatas.isEmpty()) {
54     String[] productNumbers = cartProductNumberDatas.split("\\\\+");
55     ArrayList<Integer> productNumberList = new ArrayList<>();
56     for (String productNumber : productNumbers) {
57         try {
58             productNumberList.add(Integer.parseInt(productNumber.trim()));
59         } catch (NumberFormatException e) {
60             System.err.println("잘못된 상품 번호: " + productNumber);
61         }
62     }
63
64 Iterator<Map<String, Object>> iterator = shoppingCart.iterator();
65 while (iterator.hasNext()) {
66     Map<String, Object> cartItem = iterator.next();
67     int cartProductNumber = (int) cartItem.get("productNumber");
68     if (productNumberList.contains(cartProductNumber)) {
69         iterator.remove();
70     }
71 }
72 }
```

장바구니에서 선택한
+로 연결된 모든 상품의 번호 값을 요청
+를 기준으로 값들을 나눈 후
새로운 배열에 저장

요청 상품 번호와
장바구니 상품의 번호가 일치하면
장바구니에서 해당 상품을 삭제

장바구니 상품	가격	수량	총 가격	
	코카)스프라이트P500ml	2100	<div style="display: flex; align-items: center; justify-content: space-around;">1+-</div>	2100원
총 구매 가격				0 원

장바구니에서
상품 삭제 완료

```
31• protected void doPost(HttpServletRequest request, HttpServletResponse response)
32    throws ServletException, IOException {
33
34    HttpSession session = request.getSession();
35
36    ArrayList<Map<String, Object>> shoppingCart = (ArrayList<Map<String, Object>>) session.getAttribute("shoppingCart");
37    if (shoppingCart == null) {
38        shoppingCart = new ArrayList<>();
39        session.setAttribute("shoppingCart", shoppingCart);
40    }
41
42    int productNumber = Integer.parseInt(request.getParameter("productNumber"));
43    int cartProductCount = Integer.parseInt(request.getParameter("cartProductCount"));
44    boolean isComboProduct = Boolean.parseBoolean(request.getParameter("isComboProduct"));
45
46    boolean alreadyIn = false;
47
48    for (Map<String, Object> cartItem : shoppingCart) {
49        if ((int) cartItem.get("productNumber") == productNumber && (boolean) cartItem.get("isComboProduct") == isComboProduct) {
50            int currentCount = (int) cartItem.get("cartProductCount");
51            cartItem.put("cartProductCount", currentCount + cartProductCount);
52            alreadyIn = true;
53            break;
54        }
55    }
56}
```

모든 장바구니 기능은
비동기 처리로 진행
페이지 새로고침 없이 진행

04 _ API 문서

01. Map & GPS API



```
220     <script type="text/javascript"  
221         src="https://dapi.kakao.com/v2/maps/sdk.js?appkey=발급받은 자신의 API 키 삽입&libraries=services,clusterer,drawing"></script>
```

1. CDN 방식을 사용하여 Kakao Map API 호출

→ API 정상적 호출을 위해 Kakao Developer로부터 AppKey를 발급

```
<div id='mapWrapper' style="width: 100%; height: 500px;"></div>  
  
228     // 지도 초기화 함수  
229     function initMap() {  
230         map = new kakao.maps.Map(document.getElementById('mapWrapper'), {  
231             center: new kakao.maps.LatLng(37.5665, 126.9780), // 초기 지도 중심 좌표 설정  
232             level: 3 // 초기 지도 확대 레벨 설정  
233         });  
234     }
```

2. <div>를 사용하여 Map 객체를 호출할 위치와 id 생성 → 부여한 id와 이름이 다르다면 맵 출력 불가능

```
236 // 현재 위치 찾기 함수
237 function getLocation() {
238     if (navigator.geolocation) {
239         navigator.geolocation.getCurrentPosition(function(position) {
240             const pos = new kakao.maps.LatLng(position.coords.latitude, position.coords.longitude)
241             map.setCenter(pos); // 지도 중심을 현재 위치로 설정
242
243             if (userMarker) userMarker.setMap(null); // 기존 마커 삭제
244
245             // 새로운 마커 생성
246             userMarker = new kakao.maps.Marker({
247                 position: pos,
248                 map: map,
249                 title: '내 위치'
250             });
251
252             searchNearbyCU(pos); // 주변 300m 반경 내 편의점 검색
253         });
254     }
255 }
```

3. 웹 페이지에서 사용자의 편의성을 위해 geoLocation() 내장API를 사용하여 사용자의 현재 위치 파악 편의점 검색 함수 호출

```
310 // 주변 300m 반경 내 CU 편의점 검색
311 function searchNearbyCU(location) {
312     const ps = new kakao.maps.services.Places();
313     ps.categorySearch('CS2', function(result, status) { // 'CS2'는 편의점 카테고리
314         if (status === kakao.maps.services.Status.OK) {
315             result.forEach(function(place) {
316                 if (place.place_name.includes("CU")) { // 이름에 "CU"가 포함된 편의점만 처리
317                     displayMarker(place);
318                 }
319             });
320
321             if (!result.some(place => place.place_name.includes("CU"))) {
322                 alert('주변에 CU 편의점을 찾을 수 없습니다.');
323             }
324         } else {
325             alert('주변에 편의점을 찾을 수 없습니다.');
326         }
327     }, {
328         location: location,
329         radius: 300 // 검색 반경 (미터)
330     });
331 }
```

4. 현재 위치로 이동이 완료 된 후

위치 기준 300m 반경 내
편의점 검색

- 'CS2' = Kakao에서 사용하는
카테고리
- radius = 반경 지정

```
261 // 사용자가 입력한 검색어로 장소 검색
262 function searchStore() {
263     const keyword = document.getElementById('searchStoreKeyword').value.trim()
264     console.log("편의점 검색어 [", keyword, "]");
265     if (!keyword) {
266         alert('편의점 이름을 입력해 주세요.');
267         return;
268     }
269
270     const ps = new kakao.maps.services.Places();
271
272     ps.keywordSearch(keyword, function(data, status, pagination) {
273         if (status === kakao.maps.services.Status.OK) {
274             // 검색 결과 중 첫 번째 장소를 기준으로 지도 이동
275             const firstPlace = data[0];
276             map.setCenter(new kakao.maps.LatLng(firstPlace.y, firstPlace.x));
277         }
278     });
279 }
280
281 // 지도 중심좌표 설정
282 map.setCenter(new kakao.maps.LatLng(37.5, 127.0));
283
284 // 지도 확대/축소 조작
285 map.setZoom(10);
286
287 // 마커 클릭 시 해당 장소 정보 제공
288 map.on('click', function(e) {
289     const place = e.place;
290     if (place) {
291         alert(`장소명: ${place.name}, 주소: ${place.address_name}`);
292     }
293 });
294
295 // 키보드 엔터 키로 검색 실행
296 document.addEventListener('keydown', function(e) {
297     if (e.key === 'Enter') {
298         searchStore();
299     }
300 });
301
302 // 편의점 검색 결과 표시
303 const resultList = document.getElementById('list');
304
305 resultList.innerHTML = '';
306
307 for (const place of places) {
308     const item = document.createElement('div');
309     item.innerHTML = `

장소명: ${place.name}, 주소: ${place.address_name}

`;
310     resultList.appendChild(item);
311 }
```

5. 검색창으로부터 검색어를 입력받고
첫 번째 검색된 장소의 위치로
지도 이동
마커 클릭시 해당 장소 정보 제공

03 _ 프로젝트 기능

02. 게시글 좋아요

```
// 좋아요 추가  
  
final String INSERTBOARDCOMBOLIKED =  
    "INSERT INTO BOARD_COMBO_LIKED  
        (BOARD_COMBO_LIKED_NUMBER, MEMBER_NUMBER, BOARD_COMBO_NUMBER)  
        SELECT IFNULL(MAX(BOARD_COMBO_LIKED_NUMBER), 0) + 1, ?, ?  
    FROM BOARD_COMBO_LIKED;
```

좋아요 번호(PK) 값을 수동으로 증가시키는 코드

AUTO_INCREMENT 대신 사용

가장 큰 값에 1을 더하여 설정하며 만약 NULL이라면 0으로 치환

```
// 좋아요 누른 총 개수, 좋아요 누른 게시물 목록(글 번호, 글 제목, 글 작성자) 출력 - 최신순
final String SELETEALL =
    SELECT
        BOARD_COMBO_LIKED.BOARD_COMBO_LIKED_NUMBER,
        BOARD_COMBO_LIKED.BOARD_COMBO_NUMBER,
        BOARD_COMBO.BOARD_COMBO_TITLE,
        MEMBER.MEMBER_NAME,
        BOARD_COMBO_LIKED.MEMBER_NUMBER,
        COUNT(BOARD_COMBO_LIKED.BOARD_COMBO_NUMBER) OVER() AS TOTAL_COUNT_NUMBER
    FROM BOARD_COMBO_LIKED
        JOIN BOARD_COMBO ON BOARD_COMBO_LIKED.BOARD_COMBO_NUMBER = BOARD_COMBO.BOARD_COMBO_NUMBER
        JOIN MEMBER ON BOARD_COMBO.MEMBER_NUMBER = MEMBER.MEMBER_NUMBER
    WHERE BOARD_COMBO_LIKED.MEMBER_NUMBER = ?
    ORDER BY BOARD_COMBO_LIKED.BOARD_COMBO_NUMBER DESC LIMIT ?, ?;
```

윈도우 함수 사용

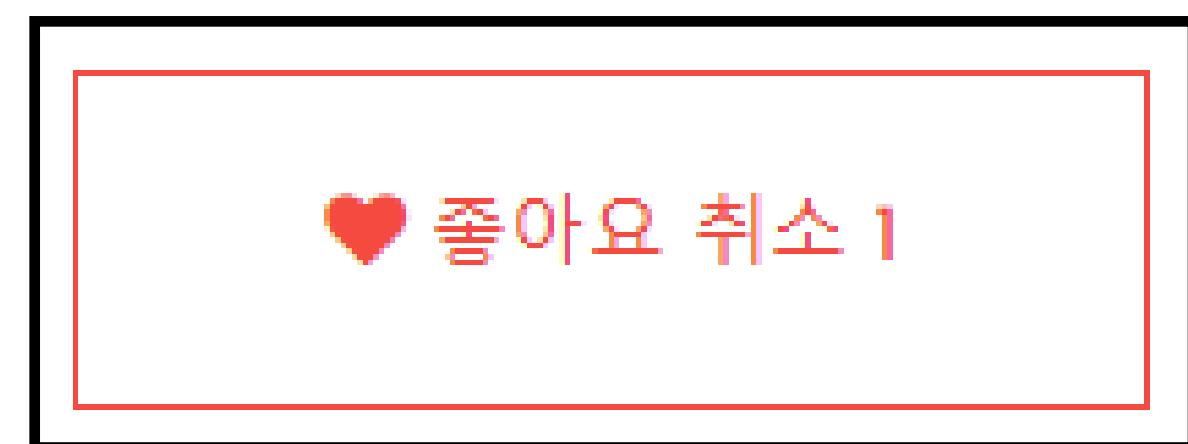
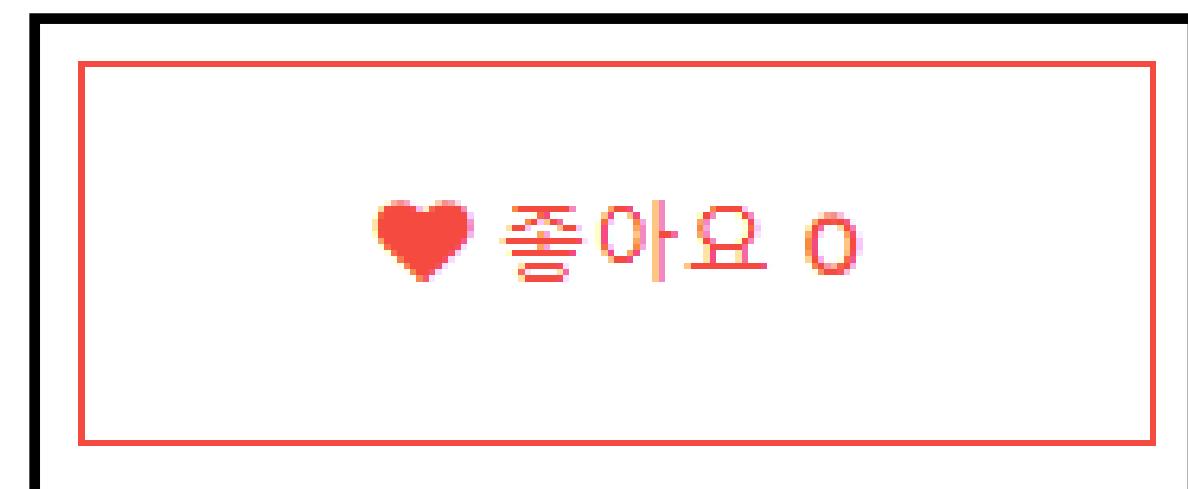
좋아요 테이블의 게시글 번호와 꿀조합 게시판의 게시글 번호 조인
꿀조합 게시판의 회원 번호와 회원 테이블의 회원 번호 조인

```
// 글 번호와 회원 번호가 둘 다 있다면 정보를 가져옴 (좋아요 여부 판단)
final String SELECTONE =
    SELECT
        BOARD_COMBO_LIKED_NUMBER,
        MEMBER_NUMBER,
        BOARD_COMBO_NUMBER
    FROM BOARD_COMBO_LIKED
    WHERE MEMBER_NUMBER = ? AND BOARD_COMBO_NUMBER = ?;
```

좋아요 여부 판단

이미 좋아요를 누른 글에는 다시 좋아요를 누를 수 없도록 함

```
57 // 좋아요 버튼 생성 기능
58 const createLikedBtn = () => {
59     console.log("좋아요 버튼 생성 시 좋아요 여부 : [" + isLiked + "]");
60     console.log("좋아요 버튼 생성 시 좋아요 수 : [" + boardLikedCount + "]");
61     // 좋아요 버튼 내용 비우기
62     $("#likedBtnWrapper").empty();
63     let contentText = "";
64
65     if (isLiked) { // 좋아요 상태라면
66         contentText = "좋아요 취소";
67     }
68     else { // 좋아요 상태가 아니라면
69         contentText = `좋아요`;
70     }
71
72     // 화면에 생성
73     $("#likedBtnWrapper").append(`
74         <a href="javascript:void(0);"
75             id="likeBtn" class="genric-btn primary-border e-large danger-border">
76             <i class="fa fa-heart" aria-hidden="true"></i>
77             <span>
78                 `+ contentText + `&ampnbsp` + boardLikedCount + `
79             </span>
80         </a>
81     `);
82 }
```



```
84 // 좋아요 여부 변경 기능
85 const clickLiked = () => {
86     let likedCondition = "";
87     console.log("좋아요 버튼 클릭");
88     console.log("좋아요 버튼에 요청받은 글 번호 : [" + boardComboNumber + "]");
89     console.log("좋아요 버튼에 요청받은 회원 번호 : [" + loginedMemberNumber + "]");
90     console.log("좋아요 버튼에 요청받은 좋아요 여부 : [" + isLiked + "]");
91
92 if (!loginedMemberNumber) { // 비회원이라면
93     alert("로그인이 필요한 기능입니다.");
94     return;
95 }
96
97 if (isLiked) { // 좋아요 글이라면
98     likedCondition = "DELETELIKED";
99 }
100 else { // 좋아요 글이 아니라면
101     likedCondition = "INSERTLIKED";
102 }
103 console.log("좋아요 등록/취소 조건 : [" + likedCondition + "]");
```

로그인 여부 검사

상황에 맞는 조건 설정

```
105$.ajax({ // 비동기
106  url: clickLikedUrl, // 보낼 주소
107  type: 'POST', // 방식
108  data: { // 보낼 값
109    boardComboNumber: boardComboNumber,
110    likedCondition: likedCondition
111  },
112  dataType: 'text', // 받을 타입
113  success: (response) => { // 성공적이라면
114    console.log("받은 좋아요 여부 : [" + response + "]"); // 로그 찍기
115  }
}
```

비동기로 백단에

1. 게시글 번호

2. 조건(좋아요 등록/삭제)

값들 전달

```
122
123  isLiked = !isLiked; // 회원 좋아요 여부 값 변경
124  boardLikedCount = response // 글 좋아요 수 값 변경
125  // 좋아요 버튼 생성 함수 호출
126  createLikedBtn();
127  },
128  error: (xhr, status, error) => { // 에러 처리
129    console.error("AJAX 요청 에러 발생", xhr.status, status, error);
130    alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
131    location.href = "error.do";
132  }
133});
```

버튼 값 변경

```
67     if("INSERTLIKED".equals(orderCondition)) {  
68         isLiked = boardComboLikedDAO.insert(boardComboLikedDTO);  
69         boardComboLikedCount++;  
70         System.out.println("수정된 좋아요 수 [" +boardComboLikedCount+ "]");  
71  
72         System.out.println(isLiked ? "좋아요 등록 성공" : "좋아요 등록 실패");  
73     }  
74     else if(("DELETELIKED").equals(orderCondition)){  
75         isLiked = boardComboLikedDAO.delete(boardComboLikedDTO);  
76         boardComboLikedCount--;  
77         System.out.println("수정된 좋아요 수 [" +boardComboLikedCount+ "]");  
78  
79         System.out.println(isLiked ? "좋아요 취소 성공" : "좋아요 취소 실패");  
80     }  
81  
82     response.setContentType("text/plain");  
83     response.setCharacterEncoding("UTF-8");  
84  
85     if(isLiked) {  
86         response.getWriter().write(String.valueOf(boardComboLikedCount));  
87     }  
88     else {  
89         response.getWriter().write("false");  
90     }  
91 }  
92 }
```

해당 회원의 버튼 입력에 따라 좋아요 또는 좋아요 취소

변경된 좋아요 수 또는 false를 반환하여
좋아요 여부 비동기 처리

03 _ 프로젝트 기능

03. Crawling

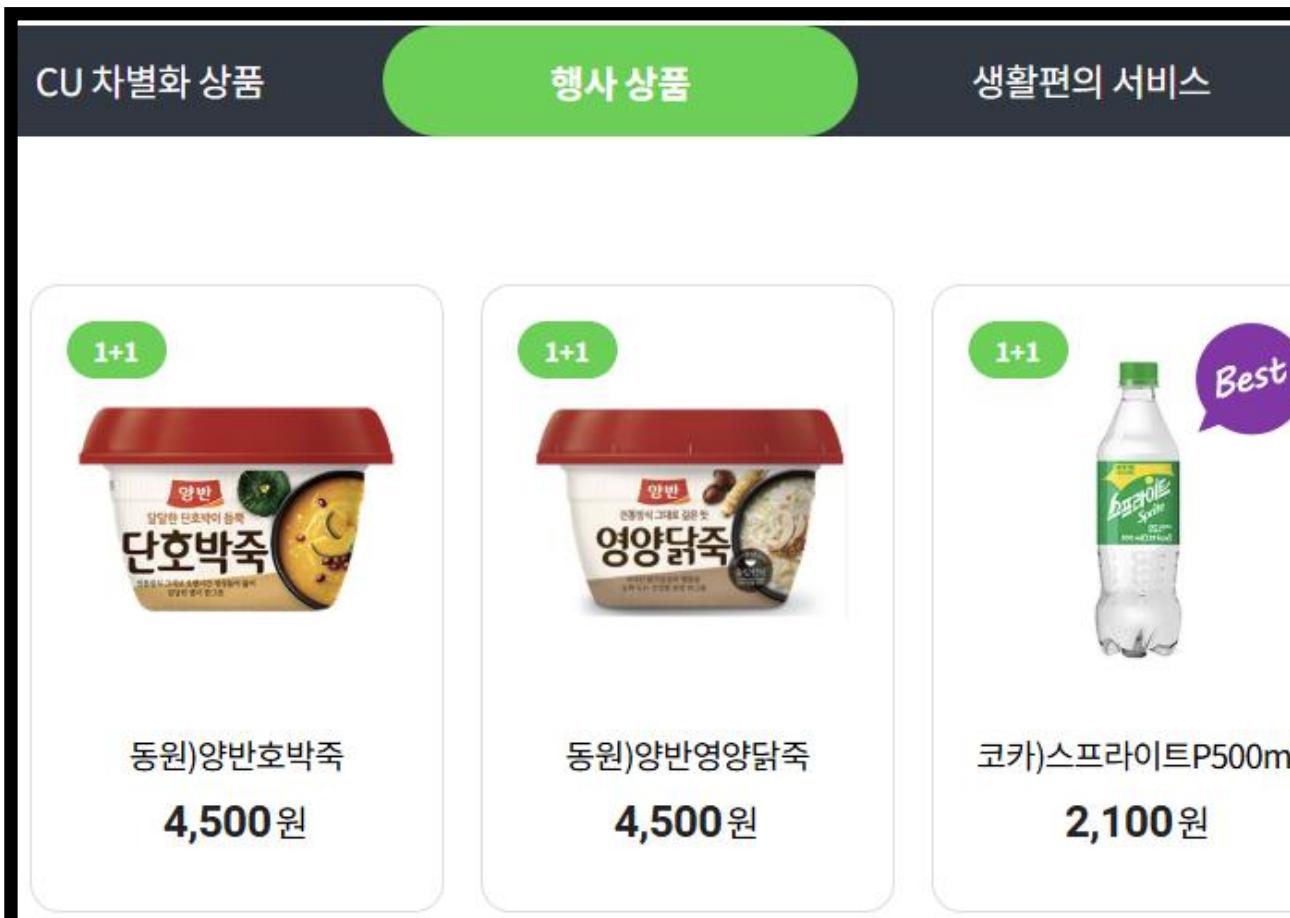
```
23● @Override  
24  public void contextInitialized(ServletContextEvent sce) {  
25      System.out.println("[리스너] 서버 시작됨, 크롤링 시작!");  
26  
27      try {  
28          // CU 상품 크롤링  
29          new StoreCU().makeSampleCU();  
30          System.out.println("[리스너] CU 크롤링 완료!");  
31  
32          // GS25 상품 크롤링  
33          new StoreGS25().makeSampleGS25();  
34          System.out.println("[리스너] GS25 크롤링 완료!");  
35  
36      } catch (Exception e) {  
37          System.out.println("[리스너] 크롤링 도중 오류 발생!");  
38          e.printStackTrace();  
39      }  
40  }
```

Listener

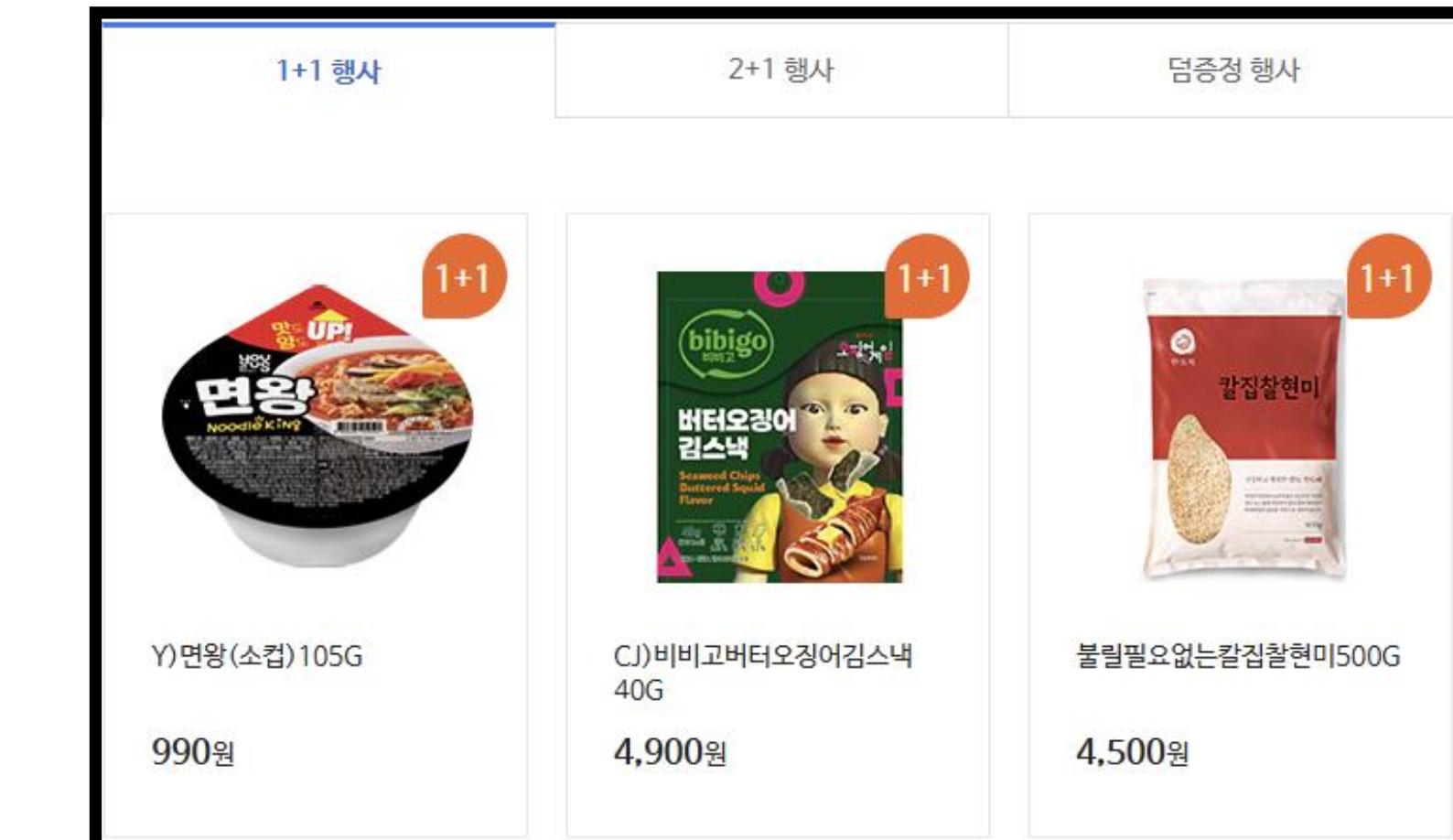
ServletContextListener를 활용하여 서버 시작 시 자동으로 크롤링 실행

관리자가 따로 실행할 필요 없어 운영 효율성이 높아짐

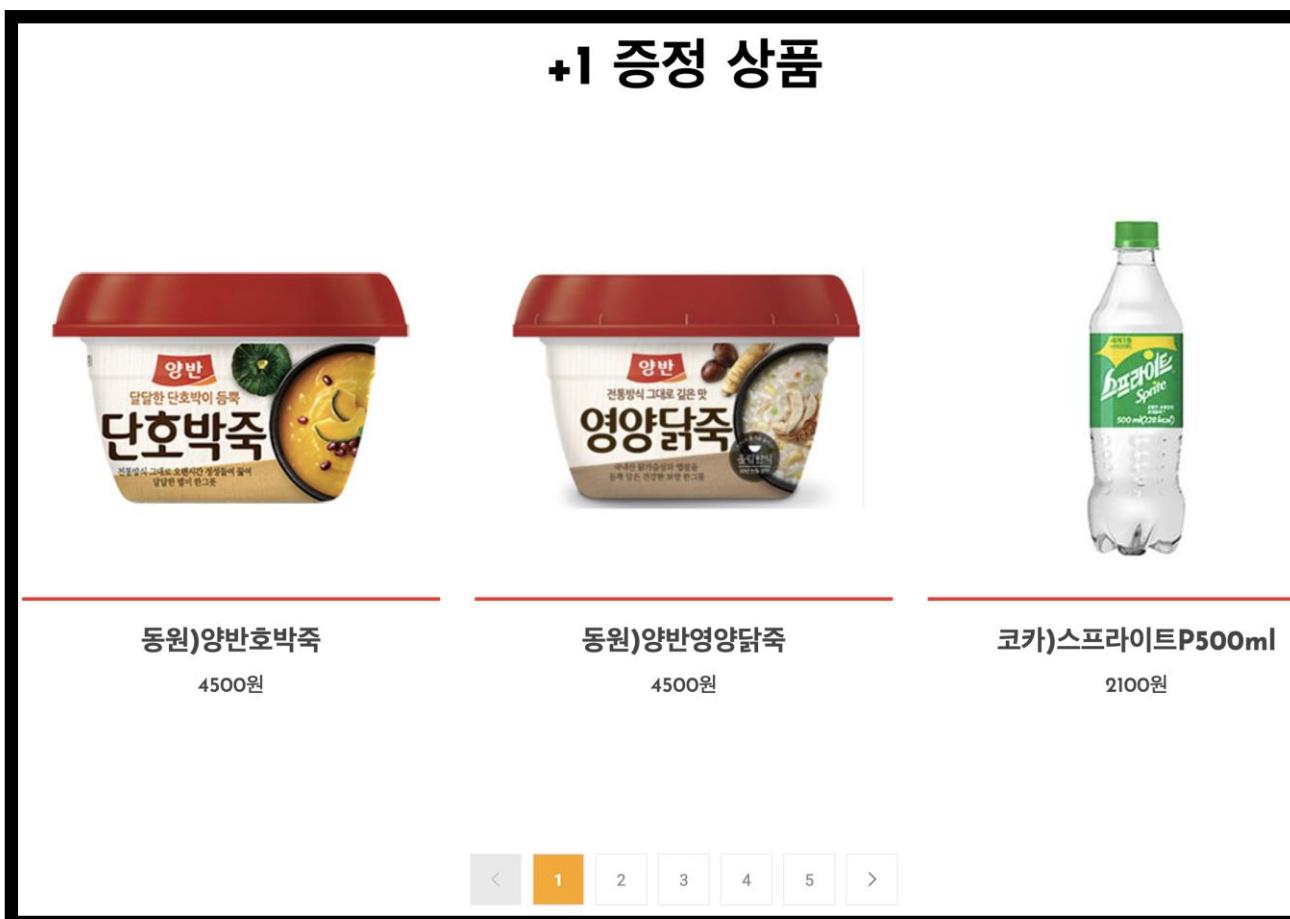
[CU 상품]



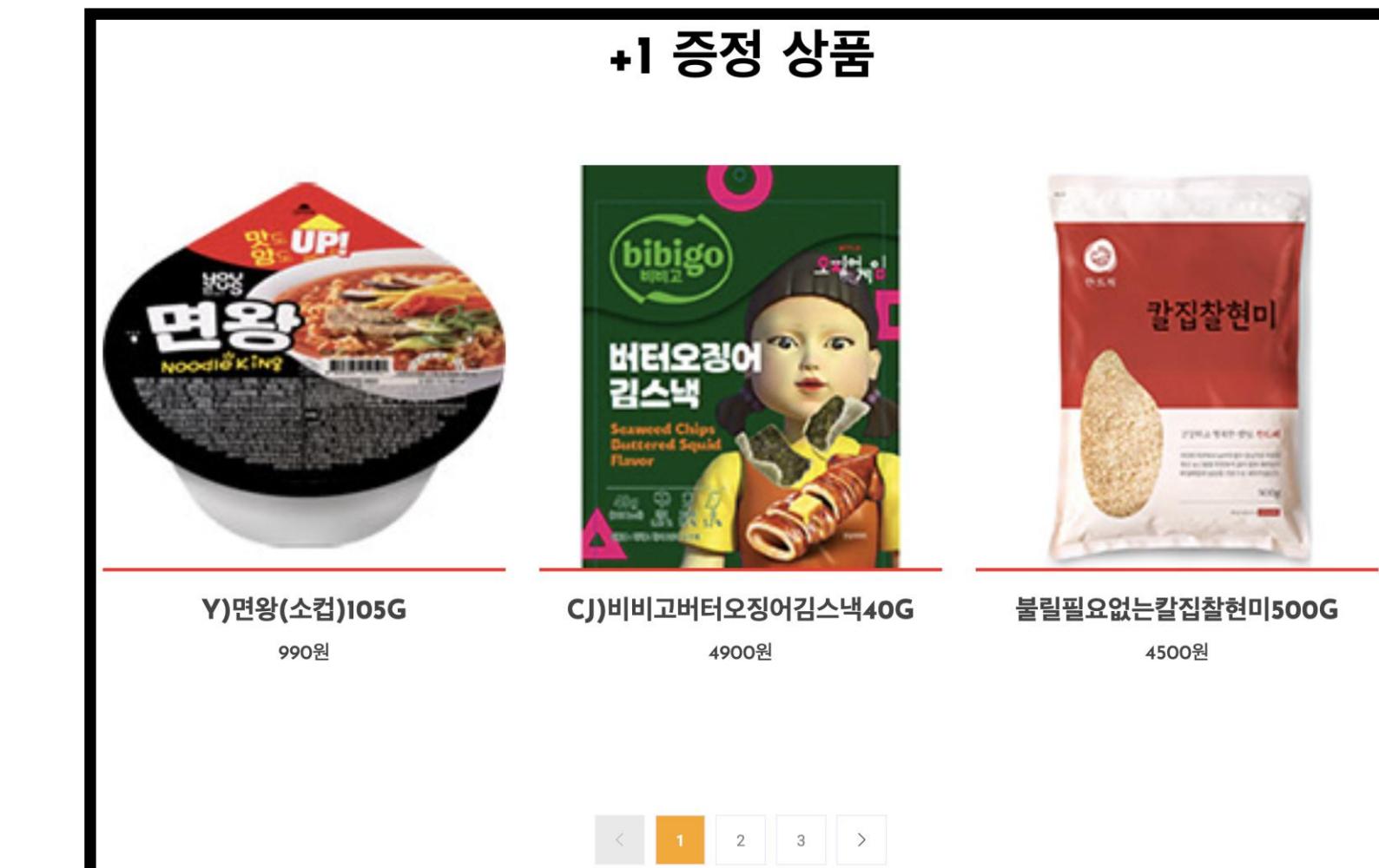
[GS 상품]



[Factory 상품]



[Factory 상품]



```
17 public class StoreCU {  
18     public void makeSampleCU() {  
19         System.out.println("[CU] : 크롤링 시작");  
20         // System.setProperty("webdriver.chrome.driver", "googleDriver/chromedriver.exe");  
21         ChromeOptions options = new ChromeOptions();  
22         options.addArguments("user-agent=Mozilla/5.0");  
23         WebDriver driver = new ChromeDriver(options);  
24         WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));  
25  
26         try {  
27             crawlNewCU(driver, wait);          // 핫이슈  
28             crawlEventCU(driver, wait);      // 1+1 증정상품  
29             crawlCategoryCU(driver, wait);   //식품/음료/생활용품  
30         } catch (Exception e) {  
31             System.out.println("[CU]: 전체 크롤링 중 예외 발생");  
32             e.printStackTrace();  
33         } finally {  
34             driver.quit();  
35             System.out.println("[CU]: 크롤 드라이버 종료");  
36         }  
37     }  
}
```

크롤링 흐름을 하나의 함수로 캡슐화

```
17 public class StoreGS25 {  
18     public void makeSampleGS25() {  
19         // System.setProperty("webdriver.chrome.driver", "googleDriver/chromedriver.exe");  
20         System.out.println("[GS25] 크롤링 시작");  
21         ChromeOptions options = new ChromeOptions();  
22         options.addArguments("user-agent=Mozilla/5.0");  
23         WebDriver driver = new ChromeDriver(options);  
24         WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));  
25  
26         try {  
27             crawlEvent(driver, wait);        // 1+1 상품  
28             crawlHotIssue(driver, wait);    // 신상품 (span.new)  
29             crawlCategory(driver, wait);    // 식품 / 음료 / 생활용품  
30         } catch (Exception e) {  
31             System.out.println("[GS25] 전체 크롤링 중 예외 발생:");  
32             e.printStackTrace();  
33         } finally {  
34             driver.quit();  
35             System.out.println("[GS25] 크롤 드라이버 종료");  
36         }  
37     }  
}
```

역할 분담이 명확하고 유지보수에 유리

```
134●    private void crawlCategoryCU(WebDriver driver, WebDriverWait wait) throws Exception {  
135        String[] urls = { //url이 다른  
136            "https://cu.bgfretail.com/product/product.do?category=product&depth2=4&depth3=1", // 식품  
137            "https://cu.bgfretail.com/product/product.do?category=product&depth2=4&depth3=6", // 음료  
138            "https://cu.bgfretail.com/product/product.do?category=product&depth2=4&depth3=7" // 생활용품  
139        };  
140        String[] categories = { "FOODPRODUCT", "BEVERAGEPRODUCT", "DAILYSUPPLIESPRODUCT" };  
141        //db카테고리 컬럼에 들어갈 값  
142        String[] conditions = { "food", "drink", "goods" };  
143        //dto 컨디션필드에 들어갈 검색 조건 키워드  
144  
145        ProductSingleDAO dao = new ProductSingleDAO();  
146    }
```

편의점 카테고리별 크롤링

각 카테고리가
독립된 URL로 분리되어 있어
확장성이 높음

탭 클릭이 필요 없으므로
요소 탐색 예외 발생 가능성이 낮음

```
148●    private void crawlCategory(WebDriver driver, WebDriverWait wait) throws Exception {  
149        driver.get("https://gs25.gsretail.com/gscvs/ko/products/youus-different-service");  
150  
151        String[] tabIds = { "productRamen", "productDrink", "productGoods" };  
152        //탭 버튼의 id값  
153        String[] categoryValues = { "FOODPRODUCT", "BEVERAGEPRODUCT", "DAILYSUPPLIESPRODUCT" };  
154    }
```

탭 클릭을 통해
하나의 페이지에서 모든 데이터를 가져옴

```

91     Thread.sleep(2000);
92     wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("li.prod_list")));
44     WebElement tab = wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//a[contains(text(), '1+1')]")));
45     //해당요소가 화면에 표시되고, 클릭 가능한 상태가 될 때까지 기다리고,
46     //<a>태그 중 텍스트에 1+1이 포함된 요소를 찾음(1+1을 클릭할 수 있게 될 때까지 기다린후, 해당요소를 반환)
47     tab.click();
49     wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("ul.prod_list li")));
50     //특정요소가 DOM에 존재하면 화면에 실제로 보여 통과되는 조건
150    Thread.sleep(2000); 40
151    wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("li.prod_list")));
41    wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("li.prod_list")));

```

조건	상황	설명
presenceOfElementLoacated	DOM에 존재 여부	요소가 보이지 않아도, 존재를 확인하고 싶을 때
visibilityOfElementLocated	화면에 나타나는지 여부	텍스트를 읽거나 사용자가 보는 요소를 대상으로 할 때
elementToBeClickable	보이는 것 뿐 아니라 클릭 가능해야 함	버튼, 탭 등을 클릭하기 직전

```
280 private boolean existsByName(String productName) { //데이터가 있는지 확인 insert문 때문에
281     String sql = "SELECT COUNT(*) FROM PRODUCT_SINGLE WHERE PRODUCT_SINGLE_NAME = ?";
282     //한번만 사용하기 때문에 내부에 사용
283     try {
284         Connection conn = JDBCUtil.connect();
285         PreparedStatement pstmt = conn.prepareStatement(sql)
286     ) {
287         pstmt.setString(1, productName);
288         try (ResultSet rs = pstmt.executeQuery()) {
289             if (rs.next()) {
290                 return rs.getInt(1) > 0; // 이미 존재함
291             }
292         }
293     } catch (Exception e) {
294         e.printStackTrace();
295     }
296
297     return false; // 예외가 생기거나 없으면 insert 하게끔 처리
298 }
```

ProductSingleDAO

함수명 : existsByName

DB에 중복되는 상품명을
insert() 전에 여과

DAON내부에서 은닉
→ private 접근제어자 사용

```

300     private static final Map<String, Long> storeMap = new HashMap<>();
301     static {
302         storeMap.put("CU", 10000L); // CU는 10000~
303         storeMap.put("GS25", 20000L); // GS25는 20000~
304         // 확장성을 위해 다른 store도 추가 쉽게 할 수 있음
305     }

```

HashMap을 사용
→ 키-값 구조로 데이터를 관리

```

306     private long getNextProductSingleNumberByBrand(String store) { // 번호 생성
307         Connection conn = null;
308         PreparedStatement pstmt = null;
309         ResultSet rs = null;
310
311         long base = storeMap.getOrDefault(store, 30000L);
312         // CU나 GS25 가 아니면 기본값 30000부터 시작
313         long maxRange = base + 9999;
314         // 해당 브랜드가 사용할 최대 범위
315         long nextNumber = base;
316         // 기본으로 시작 번호로 초기화
317         // 이후 db에 따라 갱신
318         try {
319             conn = JDBCUtil.connect();
320             String sql = "SELECT IFNULL(MAX(PRODUCT_SINGLE_NUMBER), ?) AS MAXNUMBER FROM PRODUCT_SINGLE WHERE PRODUCT_SINGLE_NUMBER BETWEEN ? AND ?";
321             // 현재 db에서 해당 구간내에서 가장 큰 번호를 조회
322             pstmt = conn.prepareStatement(sql);
323             pstmt.setLong(1, base - 1); // 데이터가 없을 때는 base-1>>+1하면서 시작
324             pstmt.setLong(2, base); // 범위 시작
325             pstmt.setLong(3, maxRange); // 범위 끝
326
327             rs = pstmt.executeQuery();
328             if (rs.next()) {
329                 long max = rs.getLong("MAXNUMBER");
330                 nextNumber = (max == 0) ? base : max + 1;
331                 // max가 0이면 base부터, 아니면 +1부터
332             }
333         } catch (Exception e) {
334             e.printStackTrace();
335         } finally {
336             JDBCUtil.disconnect(conn, pstmt);
337         }
338
339         return nextNumber;
340     }

```

함수명
: getNextProductSingleNumberByBrand

max 번호 계산으로 중복 방지

범위 기반 관리
→ 각 브랜드마다 10,000개의 상품번호 보장

03 _ 프로젝트 기능

04. 주문 상세 내역 조회

? 고민

주문-주문 상세, 주문 상세-꿀조합 상품, 개별 상품으로
조인하여 사용하는데 SQL구문이 길어질 것으로 보임
DB를 3번 접근하는 것이 옳은가?

! 방안

하나의 큰 기능으로 판단,
DB에 한 번 접근하여 필요한 값들을 모두 가져옴

주문 정보		
주문 번호	: 15	배송지
결제번호	: TID015	도로명 주소 상세 주소
결제 가격	: 17600원	: 서울시 강남구 테헤란로 : 101동 202호
주문 상세정보		
상품	수량	총 가격
치즈볶이	2	2600원
도영Pick	2	15000원
총합		17600원

주문 상세 내역에는
주문 번호, 주문 상세정보, 배송지 정보 등이 포함됨

- DTO에 멤버변수를 추가하여 확장하고
DB 테이블 컬럼은 유지
- DB 정규화 및 무결성 유지

```
// 일치하는 주문번호에 맞춰서 상품 상세 정보 출력
final String SELECTALL =
    SELECT
        COALESCE(PRODUCT_SINGLE.PRODUCT_SINGLE_NAME, '') PRODUCT_SINGLE_NAME,
        COALESCE(PRODUCT_SINGLE.PRODUCT_SINGLE_PRICE, 0) PRODUCT_SINGLE_PRICE,
        COALESCE(PRODUCT_COMBO.PRODUCT_COMBO_NAME, '') PRODUCT_COMBO_NAME,
        COALESCE(PRODUCT_COMBO_COMPONENT_PRICE.PRODUCT_COMBO_PRICE, 0) PRODUCT_COMBO_PRICE,
        PURCHASE_DETAIL.PURCHASE_PRODUCT_COUNT AS PURCHASE_PRODUCT_COUNT,
        SUM(
            (COALESCE(PRODUCT_SINGLE.PRODUCT_SINGLE_PRICE, 0)
            + COALESCE(PRODUCT_COMBO_COMPONENT_PRICE.PRODUCT_COMBO_PRICE, 0))
            * PURCHASE_DETAIL.PURCHASE_PRODUCT_COUNT
        ) OVER() PURCHASE_TOTAL_PRICE
    FROM PURCHASE_DETAIL
```

개별상품, 꿀조합 상품 중 구매하지 않는 상품이 있을 수도 있으므로
NULL 값이 들어올 경우에 String 타입인 이름은 빈 문자열, LONG 타입인 가격은 0으로 처리

```
JOIN PURCHASE
    ON PURCHASE_DETAIL.PURCHASE_NUMBER = PURCHASE.PURCHASE_NUMBER
LEFT JOIN PRODUCT_SINGLE
    ON PURCHASE_DETAIL.PRODUCT_SINGLE_NUMBER = PRODUCT_SINGLE.PRODUCT_SINGLE_NUMBER
LEFT JOIN PRODUCT_COMBO
    ON PURCHASE_DETAIL.PRODUCT_COMBO_NUMBER = PRODUCT_COMBO.PRODUCT_COMBO_NUMBER
LEFT JOIN (SELECT PRODUCT_COMBO_COMPONENT.PRODUCT_COMBO_NUMBER PRODUCT_COMBO_NUMBER,
COALESCE(PRODUCT_SINGLE_1.PRODUCT_SINGLE_PRICE, 0)
+ COALESCE(PRODUCT_SINGLE_2.PRODUCT_SINGLE_PRICE, 0)
+ COALESCE(PRODUCT_SINGLE_3.PRODUCT_SINGLE_PRICE, 0) AS PRODUCT_COMBO_PRICE
FROM PRODUCT_COMBO_COMPONENT
```

1. 주문 상세 테이블과 주문 테이블, 개별상품 테이블, 꿀조합 상품 테이블을 조인
2. 꿀조합 상품 가격은 개별상품들의 가격 합과 동일하므로 COALESCE를 통해 계산
 $\text{NULL} + N = \text{NULL}$ 을 막기 위해 NULL인 경우는 더해지지 않도록 함
3. 주문 상세에 포함된 상품들의 상세정보를 한번에 조회

```
LEFT JOIN PRODUCT_SINGLE PRODUCT_SINGLE_1
      ON PRODUCT_COMBO_COMPONENT.PRODUCT_COMBO_COMPONENT_ONE = PRODUCT_SINGLE_1.PRODUCT_SINGLE_NUMBER
LEFT JOIN PRODUCT_SINGLE PRODUCT_SINGLE_2
      ON PRODUCT_COMBO_COMPONENT.PRODUCT_COMBO_COMPONENT_TWO = PRODUCT_SINGLE_2.PRODUCT_SINGLE_NUMBER
LEFT JOIN PRODUCT_SINGLE PRODUCT_SINGLE_3
      ON PRODUCT_COMBO_COMPONENT.PRODUCT_COMBO_COMPONENT_THREE = PRODUCT_SINGLE_3.PRODUCT_SINGLE_NUMBER
) PRODUCT_COMBO_COMPONENT_PRICE
      ON PRODUCT_COMBO.PRODUCT_COMBO_NUMBER = PRODUCT_COMBO_COMPONENT_PRICE.PRODUCT_COMBO_NUMBER
WHERE PURCHASE.MEMBER_NUMBER = ?
AND PURCHASE.PURCHASE_NUMBER = ?
```

1. 꿀조합 구성품 테이블에 있는 개별상품들을 개별상품 테이블과 LEFT JOIN해서 가격을 가져옴
2. 3개의 개별상품 가격을 더해서 꿀조합 상품 가격을 계산
3. 꿀조합 상품 테이블과 조인하여 회원번호와 주문번호로 해당 주문의 상세내역 조회

04 _ API 문서

02. 네이버 로그인 API

N 네이버 로그인

네이버 로그인 API 흐름

1. login.jsp(로그인페이지)에서 사용자가 '네이버 로그인' 버튼 클릭
2. 네이버 로그인 화면 → 로그인 후 사용자 정보 동의 → callback.jsp로 인증코드 등 전송
3. callback.jsp에서 받은 코드를 통해 Access Token 요청
4. callback.jsp에서 Access Token 획득 후 토큰을 이용해 네이버에 사용자 정보 요청
5. 네이버는 JSON 데이터로 정보 응답 → 애플리케이션 등록 시 선택한 정보 응답
6. callback.jsp는 네이버로부터 받은 정보를 세션에 저장한 후 로그인 처리
7. LoginAction으로 이동해서 DB에 존재하는 이메일이면 로그인, 없으면 INSERT

코드 리뷰

```
14     String scope = "name,email,birthday,birthyear,mobile"; // 네이버에게 받을 값
15     String clientId = "네이버 애플리케이션 Client ID";
16     String redirectURI = URLEncoder.encode("http://localhost:8088/honeyComboFactory/client/callback.jsp", "UTF-8"); // 로그인 성공 후 리디렉션될 콜백 주소(네이버에 등록한 것과 동일)
17     SecureRandom random = new SecureRandom();
18     String state = new BigInteger(130, random).toString();
19     String apiURL = "https://nid.naver.com/oauth2.0/authorize?response_type=code" // 네이버 인증 요청 URL
20     // 필수 파라미터
21     + "&client_id=" + clientId
22     + "&redirect_uri=" + redirectURI // 네이버 개발자 센터에 등록한 것과 일치
23     + "&state=" + state
24     + "&scope=" + scope;
25     session.setAttribute("state", state);
```

네이버 로그인 또한 카카오 로그인과 동일하게 이메일 아이디(@ 앞부분) 아이디 설정
비밀번호는 NAVER로 설정
→ NOT NULL인 MEMBER_ID, MEMBER_PASSWORD 해결

네이버 로그인 사전 작업

네이버 로그인
Callback URL (최대 5개)

`http://localhost:8088/honeyComboFactory/client/callback.jsp` +

텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.
Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

Windows App X ^

다운로드 URL

로그인 오픈 API
서비스 환경 ?

1. 네이버 개발자 센터로 접속하여 API 신청 및 애플리케이션 등록
2. Client ID, Client Secret 발급

naverLogin.jsp

```
14     String scope = "name,email,birthday,birthyear,mobile"; // 네이버에게 받을 값
15     String clientId = "네이버 애플리케이션 Client ID";
16     String redirectURI = URLEncoder.encode("http://localhost:8088/honeyComboFactory/client/callback.jsp", "UTF-8"); // 로그인 성공 후 리디렉션될 콜백 주소(네이버에 등록한 것과 동일)
17     SecureRandom random = new SecureRandom();
18     String state = new BigInteger(130, random).toString();
19     String apiURL = "https://nid.naver.com/oauth2.0/authorize?response_type=code" // 네이버 인증 요청 URL
20     // 필수 파라미터
21     + "&client_id=" + clientId
22     + "&redirect_uri=" + redirectURI // 네이버 개발자 센터에 등록한 것과 일치
23     + "&state=" + state
24     + "&scope=" + scope;
25     session.setAttribute("state", state);
```

발급받은 클라이언트 아이디를 통해 네이버 로그인 API 호출
호출을 위한 URL 작성

login.jsp

```
3      <%@ include file="naverlogin.jsp" %>
70<a> <br> <br> <a href="<%=_apiURL %>">
71     
72</a> <br> <br> <a class="lost_pass"
```

사용자가 화면에서 네이버로 로그인 버튼 클릭 시 네이버 로그인 요청 화면으로 이동하게 됨

callback.jsp

```
27     URL url = new URL(apiURL);
28     HttpURLConnection con = (HttpURLConnection) url.openConnection();
29     con.setRequestMethod("GET");
30
31     BufferedReader br = new BufferedReader(new InputStreamReader(con.getInputStream(), "UTF-8"));
32     String inputLine;
33     StringBuilder responseBuffer = new StringBuilder();
34     while ((inputLine = br.readLine()) != null) {
35         responseBuffer.append(inputLine);
36     }
37     br.close();
38
39     JSONParser parser = new JSONParser();
40     JSONObject json = (JSONObject) parser.parse(responseBuffer.toString());
41     accessToken = (String) json.get("access token");
```

사용자가 네이버 로그인을 성공하고 나면 callback.jsp에서는 네이버로부터 인증코드와 상태 파라미터를 전달받으며 이를 가지고 네이버에게 토큰 요청
앞에서 구성한 URL을 통해 토큰을 요청하고, 이 요청을 받은 네이버는 ACCESS TOKEN을 발급

callback.jsp

```
54 // 사용자 정보 요청
55 try {
56     String token = "Bearer " + accessToken; // 발급받은 토큰 사용
57     // 인증된 정보임을 네이버에게 알림
58     URL profileUrl = new URL("https://openapi.naver.com/v1/nid/me"); // 사용자 정보를 요청할 URL
59     HttpURLConnection profileCon = (HttpURLConnection) profileUrl.openConnection();
60     profileCon.setRequestMethod("GET");
61     profileCon.setRequestProperty("Authorization", token);
62     // 네이버로부터 받아온 사용자 정보 읽기
63     BufferedReader profileBr = new BufferedReader(new InputStreamReader(profileCon.getInputStream(), "UTF-8"));
64     StringBuilder profileResponse = new StringBuilder();
65     // 사용자 정보 추출
66     // API 애플리케이션 등록 당시 받아오기로 한 값들
67     String email = (String) responseObj.get("email");
68     String name = (String) responseObj.get("name");
69     String birthYear = (String) responseObj.get("birthyear");
70     String birthday = (String) responseObj.get("birthday");
71     String mobile = (String) responseObj.get("mobile");
```

네이버로부터 발급받은 ACCESS TOKEN을 통해 사용자 정보를 요청
GET 방식으로 요청하며, JSON 방식으로 응답을 받아옴

callback.jsp

```
97         MemberDAO memberDAO = new MemberDAO();
98         MemberDTO memberDTO = new MemberDTO();
99         memberDTO.setMemberEmailId(emailId);
100        memberDTO.setMemberEmailDomain(emailDomain);
101        memberDTO.setCondition("SELECTONEEMAIL");
102
103        MemberDTO existingUser = memberDAO.selectOne(memberDTO);
104
105        String memberId;
106        String memberPassword = "NAVER"; // 네이버는 고정 비번
107
108        memberDTO.setMemberBirth(Date.valueOf(birth));
109        memberDTO.setMemberPhoneNumber(mobile);
110        memberDTO.setCondition("INSERT");
111
112
113        // 네이버 로그인 후 세션 저장
114        session.setAttribute("loginedMemberNumber", memberDTO.getMemberNumber());
115        session.setAttribute("loginedMemberName", memberDTO.getMemberName());
116        session.setAttribute("loginedmemberIsAdmin", memberDTO.isMemberIsAdmin());
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
```

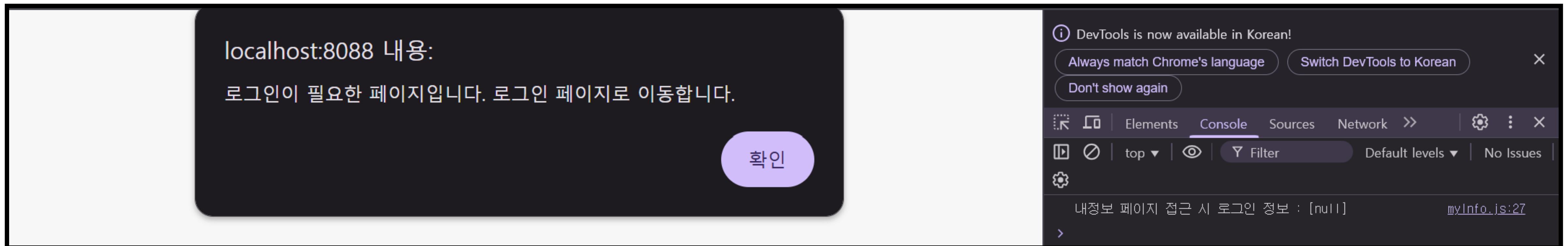
1. 존재하는 이메일인지 검사 후
만약 존재한다면 로그인 처리

2-1. 존재하지 않는 이메일 이라면
NOT NULL 방지를 위해
비밀번호와 아이디를 생성

2-2. 파싱한 값들을 세팅하여 INSERT

2-3. 세션에 회원 번호, 이름, 관리자 여부
저장하여 메인으로 리다이렉트

트러블 슈팅



```
java.lang.NullPointerException: Cannot invoke "model.dto.MemberDTO.getMemberNumber()" because "memberDTO" is null
```

❓ 발생 원인 세션에 정보를 저장하기 전에 DTO가 NULL이 되고 있음

❗ 해결 방안 DAO에서 관리자 여부, 회원 이름도 함께 조회하도록 수정 후
JS, LoginServlet에서도 세션 정보를 저장하도록 수정

04 _ API 문서

03. 이메일 API



이메일 API 흐름

1. 사용자가 이메일 인증을 선택하고, 이메일과 생년월일 입력
→ [인증번호 전송] 버튼 클릭
2. AJAX가 EmailVerificationServlet 서블릿에 POST 요청
3. EmailVerificationServlet이 DB에서 회원 확인
→ 인증번호 생성 및 회원 아이디 세션에 저장
4. 이메일 전송
5. 사용자 입력 인증번호와 비교 → FindAccountAction에서 검증 후 결과 처리
→ JSON으로 반환
6. JS에서 alert()를 통해 인증 성공 여부를 사용자에게 알림

이메일 API 사전 작업

앱 비밀번호를 사용하면 최신 보안 표준을 지원하지 않는 오래된 앱 및 서비스에서 Google 계정에 로그인할 수 있습니다.

앱 비밀번호는 최신 보안 표준을 사용하는 최신 앱 및 서비스보다 보안 수준이 낮습니다. 앱 비밀번호를 만들기 전에 앱에 로그인하려면 비밀번호가 필요한지 확인해야 합니다.

[자세히 알아보기](#)

앱 비밀번호

honeyComboFactory

생성일: 4월 4일, 최종 사용일: 4월 8일



1. 라이브러리 [webapp] → [WEB-INF]
→ [lib]에 추가
activation.jar, mail.jar

2. 구글 → [설정] → [계정관리]
→ [보안] → [앱비밀번호] 생성

findAccount.jsp

```
82          <input type="radio" name="passwordAuthMethod"  
83              value="email">본인확인 이메일로 인증
```

findAccount.js

```
139 ①    $.ajax({  
140     type: "POST",  
141     url: "/honeyComboFactory/verifyEmailCode.do",  
142     data: {  
143         memberEmailId: emailId,  
144         memberEmailDomain: emailDomain,  
145         memberBirth: birth  
146     },  
147     dataType: "json",  
148     success: (res) => {  
149         alert(res.message); // 인증번호가 이메일로 전송되었습니다  
150         if (res.code) {  
151             sendingConfirmNumber = res.code; // 서버에서 받은 인증번호 저장  
152             console.log("서버로부터 받은 인증번호 : " + sendingConfirmNumber);  
153         }  
154         stopBtn(true);  
155         inputConfirm.prop("readonly", false);  
156         startPhoneNumberConfirmTimer(findType);  
157     },  
158     error: (xhr, status, error) => {  
159         console.error("이메일 인증 AJAX 에러 발생", xhr.status, status, error);  
160         alert("이메일 인증번호 전송 중 오류가 발생했습니다.");  
161         stopBtn(false);  
162     }  
163 }) ;
```

사용자가 라디오 버튼을 통해
이메일로 인증 클릭

사용자가 정보를 입력하고
“인증번호 발송” 버튼 클릭 시
JS에서 AJAX로 Servlet에게 요청 전달

EmailVerificationServlet [아이디 찾기]

```
42     try { // AJAX에서 전달받은 데이터 받음
43         String emailId = request.getParameter("memberEmailId");
44         String emailDomain = request.getParameter("memberEmailDomain");
45         String birth = request.getParameter("memberBirth");
46
47         // DTO에 값 세팅
48         MemberDTO memberDTO = new MemberDTO();
49         memberDTO.setMemberEmailId(emailId);
50         memberDTO.setMemberEmailDomain(emailDomain);
51         memberDTO.setMemberBirth(Date.valueOf(birth));
52         memberDTO.setCondition("SELECTONEFINDID");
53
54         // DB에서 일치하는 회원이 있는지 확인
55         MemberDAO memberDAO = new MemberDAO();
56         memberDTO = memberDAO.selectOne(memberDTO);
57
58         // 일치하는 회원이 있다면
59         // 인증번호 생성해서 세션에 저장
60         // 해당 회원의 아이디도 함께 세션에 저장해두었다가 화면에 표시해줄 때 사용
61         String verificationCode = String.format("%06d", new Random().nextInt(1000000));
62         session.setAttribute("verificationCode", verificationCode);
63         session.setAttribute("findId", memberDTO.getMemberId());
64
65         // 이메일 전송
66         // 구글 계정으로 로그인 설정
67         String to = emailId + "@" + emailDomain;
68         String from = "kokanghee0014@gmail.com"; // 보낼 이메일
69         Properties props = new Properties();
70         props.put("mail.smtp.host", "smtp.gmail.com");
71         props.put("mail.smtp.port", "587");
72         props.put("mail.smtp.auth", "true");
73         props.put("mail.smtp.starttls.enable", "true");
```

사용자가 입력한 이메일과 생년월일을 받아온 후 DTO에 정보를 담아 해당 DTO로 DB에서 일치하는 회원 정보를 조회

일치하는 회원이 있다면 6자리 인증번호 생성해서 세션에 해당 회원의 아이디와 인증번호를 저장

앱 비밀번호를 통해 이메일 전송 설정
이메일 전송 후 JSON으로 결과 반환

EmailVerificationServlet [비밀번호 찾기]

```
336 ①     $.ajax({
337     type: "POST",
338     url: updateMemberPasswordUrl,
339 ②     data: {
340         memberId: memberId,
341         memberPassword: memberPassword
342     },
343     dataType: "text",
344 ③     success: (response) => {
345 ④         if (response === "true") {
346             alert("비밀번호가 성공적으로 변경되었습니다.");
347             location.href = "main.do";
348 ⑤         } else {
349             alert("비밀번호 변경에 실패했습니다.");
350             location.href = "error.do";
351         }
352     },
353 ⑥     error: () => {
354         alert("서버 오류가 발생했습니다.");
355         location.href = "error.do";
356     }
}
```

비밀번호 찾기의 경우,
인증이 완료되면 개인정보 보안을 위해
새로운 비밀번호를 설정하도록 구현

같은 페이지 내에서 비동기 처리

03 _ 프로젝트 기능

05. 페이지네이션

```
5 // 꿀조합 상품 불러오기 서블릿 url  
6 const loadMoreComboUrl = "/honeyComboFactory/OrderProductComboServlet";  
  
9 // 전역 객체 선언  
10 const productState = {  
11     pageGroupSize: 5, // 한 번에 보여줄 페이지 개수  
12     pageNumber: { // 페이지 수  
13         combo: 1  
14     },  
15     contentCount: { // 한 번에 보여줄 데이터 수  
16         combo: 6  
17     },  
18     pageNation: { // 페이지네이션 생성할 영역 id  
19         combo: "comboPageNation"  
20     },  
21     // 꿀조합 전용  
22     orderCondition: "ORDERPOPULAR", // 상품 정렬 조건  
23     category: "ALLPRODUCT", // 상품 카테고리  
24     searchKeyword: "", // 검색 상품이름  
25     searchKeywordPageFlag: true // 검색 상품 페이지 1로 설정 여부  
26 };
```

유지보수를 위해 최상단 상수화

맵 전역 객체 선언

장점

- 공유 데이터 쉽게 처리
- 다양한 키 타입 가능
- 빠르고 깔끔한 캐싱

```
49 // 꿀조합 상품 페이지네이션 이벤트 등록
50 $(document).on("click", ".combo", function(event) { // 페이지네이션 클릭 시
51     event.preventDefault(); // 기본 이벤트 방지
52
53     changePage("combo", $(this)); // 페이지 이동
54 });
```

combo 클래스 요소에 클릭 이벤트 등록

모듈화 한 메서드 호출

```
159 // 꿀조합 상품 페이지네이션 페이지 이동 기능
160 const changePage = (type, element) => {
161     console.log("상품 페이지네이션 클릭 타입 : [" + type + "]");
162     console.log("상품 페이지네이션 클릭 번호 : [" + element.data('page') + "]");
163     console.log("상품 페이지네이션 클릭 아이디 : [" + element.attr('id') + "]");
164
165     if (element.attr('id') === 'Previous') { // "<" 버튼 클릭 시
166         productState.pageNumber[type]--;
167     } else if (element.attr('id') === 'Next') { // ">" 버튼 클릭 시
168         productState.pageNumber[type]++;
169     } else { // 페이지 번호 클릭 시
170         productState.pageNumber[type] = element.data('page');
171     }
172
173     if (productState.searchKeyword) { // 검색어가 있다면
174         // 꿀조합 상품 이름 검색 함수 호출
175         searchComboProductName();
176     }
177     else { // 검색어가 없다면
178         // 꿀조합 상품 불러오기 함수 호출
179         loadDatas(type);
180     }
181 }
```

빠른 실행을 위해 화살표 함수 사용

1. <

2. >

3. 페이지 번호

구분 후 전역변수의 페이지 번호 값 변경

상품 정보 불러오기 함수 실행

```
183 // 꿀조합 상품 불러오기 기능
184 const loadDatas = (type) => {
185     console.log("불러올 상품 타입 : [" + type + "]");
186
187     $.ajax({ // 비동기
188         url: loadMoreComboUrl, // 보낼 주소
189         type: 'GET', // 방식
190         data: { // 보낼 값
191             comboProductPageNumber: productState.pageNumber[type],
192             comboProductContentCount: productState.contentCount[type],
193             comboProductOrderCondition: productState.orderCondition,
194             comboProductCategory: productState.category
195         },
196         datatype: 'json', // 받을 타입
197         success: (response) => { // 성공적이라면
198             console.log("받은 꿀조합 상품 데이터 : [" + response + "]"); // 로그 찍기
199
200             // 꿀조합 상품 생성 함수 호출
201             insert(type, response);
202         },
203         error: (xhr, status, error) => { // 에러 처리
204             console.error("AJAX 요청 에러 발생", xhr.status, status, error);
205             alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
206             location.href = "error.do";
207         }
208     });
209 }
```

비동기로 백단에

1. 페이지 번호
2. 받아올 데이터 수
3. 정렬 조건
4. 카테고리 전달

상품 출력 함수 실행

```

34     int boardPageNumber = Integer.parseInt(request.getParameter("boardPageNumber"));
35     int boardContentCount = Integer.parseInt(request.getParameter("boardContentCount"));
36
37     int startIndex = (boardPageNumber - 1) * boardContentCount;
38
39     boardComboDTO.setBoardComboIndex(startIndex);
40     boardComboDTO.setBoardComboContentCount(boardContentCount);
41
42
43     if ("ORDERUPTODATE".equals(orderCondition)) {
44         boardComboDTO.setCondition("SELECTALLMEMBERCONTENTDESC");
45     } else if ("ORDEROLD".equals(orderCondition)) {
46         boardComboDTO.setCondition("SELECTALLMEMBERCONTENTASC");
47     } else if ("ORDERPOPULAR".equals(orderCondition)) {
48         boardComboDTO.setCondition("SELECTALLMEMBERCONTENTPOPULAR");
49     }
50
51     BoardComboDAO boardComboDAO = new BoardComboDAO();
52     ArrayList<BoardComboDTO> boardComboDatas = boardComboDAO.selectAll(boardComboDTO);

```

페이지 번호와
해당 페이지에 보여줄 데이터의 수를 받아옴

시작 인덱스 값을 구하고
출력할 데이터 수와 함께 DB에 요청

DB에서 받은 데이터들을 배열에 저장

```

63     JSONArray boardComboArray = new JSONArray();
64     for (BoardComboDTO boardCombo : boardComboDatas) {
65         JSONObject boardComboObject = new JSONObject();
66         boardComboObject.put("boardComboNumber", boardCombo.getBoardComboNumber());
67         boardComboObject.put("memberName", boardCombo.getMemberName());
68         boardComboObject.put("boardComboTitle", boardCombo.getBoardComboTitle());
69         boardComboObject.put("boardComboViewCount", boardCombo.getBoardComboViewCount());
70         boardComboObject.put("boardComboLikedCount", boardCombo.getBoardComboLikedCount());
71         boardComboObject.put("boardComboRegisterDate", boardCombo.getBoardComboRegisterDate().toString());
72         boardComboObject.put("totalCountNumber", boardCombo.getTotalCountNumber());
73         boardComboArray.add(boardComboObject);
74
75         System.out.println("내보내는 데이터 [" + boardComboObject + "]");
76
77     }
78     JSONObject boardComboData = new JSONObject();
79
80     boardComboData.put("boardComboDatas", boardComboArray);

```

완성된 배열을 JSONArray를 통해 전달

DTO 설계

productComboIndex : 출력을 시작할 인덱스 번호 (데이터 번호)

productComboContentCount : 한 페이지에 출력할 데이터 개수

totalCountNumber : 총 데이터 개수 (상품의 총 개수)

```
ORDER BY TOTAL_SALES DESC  
LIMIT ?, ?;
```

파라미터에 데이터를 바인딩하여 페이지네이션

```
211 // 반복하며 꿀조합 상품 생성 기능
212 const insert = (type, response) => {
213     console.log("받은 상품 생성 타입 : [" + type + "]");
214     console.log("받은 상품 생성 정보 : [" + response + "]");
215
216     const wrapper = $("#" + type + "Wrapper");
217
218     // 상품 초기화
219     wrapper.empty();
220
221     if (response.length === 0) { // 응답이 비어있다면
222         console.log(type + " 상품 존재하지 않음");
223
224         // 꿀조합 상품 비우고 생성
225         wrapper.append(``+
226             <div class="col-xl-4 col-lg-4 col-md-6 col-sm-6">
227                 <div class="single-popular-items mb-50 text-center"></div>
228             </div>
229             <div class="col-xl-4 col-lg-4 col-md-6 col-sm-6">
230                 <div class="single-popular-items mb-50 text-center"><h1>상품 준비중입니다.</h1></div>
231             </div>
232             <div class="col-xl-4 col-lg-4 col-md-6 col-sm-6">
233                 <div class="single-popular-items mb-50 text-center"></div>
234             </div>
235         `);
236
237         // 꿀조합 상품 페이지네이션 생성 함수 호출
238         makePageNation(type, 1);
239     return;
240 }
```

받은 상품 정보가 없다면
화면에 상품 준비중 출력

페이지 생성 함수 실행

```
242 // 반복하며 꿀조합 상품 생성
243 response.forEach(productData => {
244     wrapper.append(` 
245         <div class="col-xl-4 col-lg-4 col-md-6 col-sm-6">
246             <div class="single-popular-items mb-50 text-center">
247                 <div class="popular-img">
248                     <img src=`+ productData.productComboImage + `"
249                         alt="`+ productData.productComboName + ` 상품 이미지">
250                     <div class="img-cap" data-product-id="`+
251                         productData.productComboNumber + `">
252                         <span>장바구니 담기</span>
253                     </div>
254                 </div>
255                 <div class="popular-caption">
256                     <h3>
257                         <a
258                             href="productDetail.do?productComboNumber=`
259                             + productData.productComboNumber + `">
260                             + productData.productComboName + `</a>
261                     </h3>
262                     <span>`+ productData.productComboPrice + `원</span>
263                 </div>
264             </div>
265         `);
266     });
267 });
268 
269 // 꿀조합 페이지네이션 생성 함수 호출
270 makePageNation(type, response[0].totalCountNumber);
271 }
```

화면에 상품 출력

페이지 생성 함수 실행

```
271 // 꿀조합 페이지네이션 생성 기능
272 function makePageNation(type, totalNumber) {
273     console.log("페이지네이션 생성 타입 : [" + type + "]");
274     console.log("총 데이터 수 : [" + totalNumber + "]");
275
276     // 총 페이지 수
277     const totalPageNumber = Math.ceil(totalNumber / productState.contentCount[type]);
278     console.log("총 페이지 수 : [" + totalPageNumber + "]");
279
280     // 현재 그룹
281     let group = Math.ceil(productState.pageNumber[type] / productState.pageGroupSize);
282     // 그룹의 처음 페이지 수
283     let startPage = (group - 1) * productState.pageGroupSize + 1;
284     // 그룹의 마지막 페이지 수
285     let endPage = Math.min(group * productState.pageGroupSize, totalPageNumber);
286
287     // 1 페이지 혹은 마지막 페이지면 비활성 클래스 추가
288     let prevClass = productState.pageNumber[type] <= 1 ? 'disabled-link' : '';
289     let nextClass = productState.pageNumber[type] >= totalPageNumber ? 'disabled-link' : '';
290
291     console.log("현재 페이지 수 : [" + productState.pageNumber[type] + "]");
292     console.log("이전 버튼 값 : [" + prevClass + "]");
293     console.log("다음 버튼 값 : [" + nextClass + "]");
```

총 페이지 수 계산

1. 그룹 수 결정
2. 처음 페이지 수 계산
3. 마지막 페이지 수 계산

버튼 비활성화 여부 결정

04 _ API 문서

04. 카카오 로그인 API



카카오 로그인



Web

삭제 수정

사이트 도메인	http://localhost:8088
---------	---

Redirect URI

삭제 수정

Redirect URI	http://localhost:8088/honeyComboFactory/KakaoLoginServlet
--------------	---

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

★ 입력 필수 ★

```
<script  
type="text/javascript"src="https://developers.kakao.com/sdk/js/kakao.min.js">  
</script>
```

Request

url : 카카오 서버 URL
JavaScript key : 앱 (JS) 키

Response

id	: 회원 번호
connected_at	: 접속일
kakao_account	: 설정한 계정 정보

히스토리

**고민**

간편 로그인한 회원의 정보를 DB에 저장해야 상품 구매내역과 게시판 관련 이력을 알 수 있음
바로 저장 하기엔 받은 정보만으로는 부족하며 회원가입 창으로 보내면 간편 로그인의 의미가 흐려짐
어떻게 해야 하는가?

**방안**

아이디는 이메일 아이디(@ 앞부분)로 설정
비밀번호는 간편 로그인한 타입으로 설정(KAKAO)

히스토리

? 고민

저장은 해결이 됐지만 로그인은 여러 번 가능함
계속 저장이 된다면 DB의 중복성을 위반
그리고 간편 로그인한 회원의 정보는 다른 곳에서 바뀌어 올 수 있음
어떻게 해야 하는가?

! 방안

1단계 : 회원 정보를 조회
2단계 : 간편 로그인 정보와 조회한 회원 정보를 비교
3단계 : 1. 조회 정보가 없다면 새로 저장
 2. 같다면 로그인만 처리
 3. 다르다면 정보 업데이트

코드 리뷰

```
56     // JSONParser를 사용하여 카카오 로그인 응답을 JSON 객체로 변환
57     JSONParser parser = new JSONParser();
58     JSONObject jsonResponse = (JSONObject) parser.parse(kakaoLoginResponse.toString());
59
60     // JSON 응답에서 'kakao_account' 객체를 추출하여 필요 정보 가져오기
61     JSONObject kakaoAccount = (JSONObject) jsonResponse.get("kakao_account");
62     long number = Long.parseLong(jsonResponse.get("id").toString());
63     String name = kakaoAccount.get("name").toString();
64     // 카카오에게 받은 이메일 주소에서 @ 앞부분만 추출
65     String emailId = kakaoAccount.get("email").toString().split("@")[0];
66     // 카카오에게 받은 이메일 주소에서 @ 뒷부분만 추출
67     String emailDomain = kakaoAccount.get("email").toString().split("@")[1];
68     // 카카오에게 받은 핸드폰번호에서 +82는 0으로 변경, "-"는 없애기
69     String phoneNumber = kakaoAccount.get("phone_number").toString().replace("+82 ", "0").replaceAll("-", "").trim();
70
71     // 생년월일 하나로 만들기 작업
72     String birthyear = kakaoAccount.get("birthyear").toString(); // 생년
73     String birthday = kakaoAccount.get("birthday").toString(); // 월일 (MMDD 형식)
74     // 생년월일 (YYYY-MM-DD 형식)
75     String birth = birthyear + "-" + birthday.substring(0, 2) + "-" + birthday.substring(2);
76
77     // 로그 찍기
78     System.out.println("카카오 로그인 번호 : [" + number + "]");
79     System.out.println("카카오 로그인 이름 : [" + name + "]");
80     System.out.println("카카오 로그인 이메일아이디 : [" + emailId + "]");
81     System.out.println("카카오 로그인 이메일도메인 : [" + emailDomain + "]");
82     System.out.println("카카오 로그인 핸드폰번호 : [" + phoneNumber + "]");
83     System.out.println("카카오 로그인 생년월일 : [" + birth + "]");
```



코드 리뷰

```
97         // 회원 정보 조회
98         MemberDTO isJoinedMember = memberDAO.selectOne(memberDTO);
99         System.out.println("카카오 로그인-DB 저장 기록 확인 : [" + isJoinedMember + "]");
100
101        boolean flag = true;
102        if (isJoinedMember != null) { // DB에 회원번호가 있다면
103            System.out.println("카카오 로그인-저장 기록 있음");
104
105            // 두 객체의 값이 하나라도 같지 않다면
106            if (!(memberDTO.equals(isJoinedMember))) {
107                System.out.println("두 객체의 값이 하나라도 다름");
108
109                // 로그인 정보 최신화
110                flag = memberDAO.update(memberDTO);
111            }
112        } else { // DB에 회원번호가 없다면
113            System.out.println("카카오 로그인-저장 기록 없음");
114
115            // 로그인 정보 저장
116            flag = memberDAO.insert(memberDTO);
117        }
118
119        PrintWriter out = response.getWriter();
120        if (!flag) { // 저장/업뎃에 실패했다면
121            System.out.println("카카오 로그인-DB 정보 저장/업뎃 실패");
122            out.print("false");
123            out.flush();
124        }
```

코드 리뷰

```
179e    @Override
180     public boolean equals(Object obj) { // 최상위클래스 오버라이딩
181         // 객체가 자신과 동일할 경우
182         if (this == obj) {
183             return true;
184         }
185         // obj가 null이거나 두 객체가 서로 다른 클래스 타입일 경우
186         if (obj == null || getClass() != obj.getClass()) {
187             return false;
188         }
189         // obj가 MemberDTO 타입인 경우 형변환
190         MemberDTO other = (MemberDTO) obj;
191         // memberNumber (long)은 기본형이므로 == 연산자를 사용하여 값 비교
192         // 비교할 값이 null이라면 문제 발생하므로 Objects.equals() 메서드를 사용하여 비교
193         return (this.memberNumber == other.memberNumber
194             && Objects.equals(this.memberId, other.memberId)
195             && Objects.equals(this.memberPassword, other.memberPassword)
196             && Objects.equals(this.memberName, other.memberName)
197             && Objects.equals(this.memberBirth, other.memberBirth)
198             && Objects.equals(this.memberPhoneNumber, other.memberPhoneNumber)
199             && Objects.equals(this.memberAddressMain, other.memberAddressMain)
200             && Objects.equals(this.memberAddressDetail, other.memberAddressDetail)
201             && Objects.equals(this.memberEmailId, other.memberEmailId)
202             && Objects.equals(this.memberEmailDomain, other.memberEmailDomain));
203     }
```

MemberDTO
최상위 Object 클래스
오버라이딩



트러블 슈팅

```
67 // 카카오 로그인 기능
68•const kakaoLogin = () => {
69     console.log("카카오 로그인 실행");
70
71•     Kakao.Auth.login({
72•         success: () => {
73             // 로그인 성공 후 이메일 정보 요청
74•             Kakao.API.request({
75                 url: kakaoServerUrl,
76•                 success: (response) => {
77                     console.log("카카오 로그인 성공");
78                     // 서블릿으로 반환 정보 전달
79                     window.location.href = '/honeyComboFactory/KakaoLoginServlet?
80                 },
81•                 error: (xhr, status, error) => {
82                     console.error("AJAX 요청 에러 발생", xhr.status, status, error);
83                     alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
84                     location.href = "error.do";
85                 }
86             });
87         },
88•         error: (xhr, status, error) => {
89             console.error("AJAX 요청 에러 발생", xhr.status, status, error);
90             alert("카카오 서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
91             location.href = "error.do";
92         }
93     });
94 }
```

트러블 슈팅

✖ ► Uncaught [kakao.min.js:108](#)
an {name: 'KakaoError', message: 'Invalid parameter
keys: error at Auth.login', stack: 'Error at https://
developers.kakao.com/sdk/js/k.../develope
rs.kakao.com/sdk/js/kakao.min.js:108:215'}

- ? 발생 원인 Kakao.Auth.login을 쓸 때 콜백 이름으로 error를 지원하지 않음
- ! 해결 방안 콜백 이름을 error 대신 fail로 변경

04 _ API 문서

05. 별점 플러그인



script

- src="https://ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js
- src="https://google-code-prettify.googlecode.com/svn/loader/run_prettify.js
- src="js/jquery.star-rating-svg.js"

link

- rel="stylesheet" type="text/css"
 href="css/star-rating-svg.css"

initialRating	: 점수
strokeColor	: 선 색상
strokeWidth	: 선 두께
starSize	: 크기
useFullStars	: 별 하나 단위
readOnly	: 값 고정
disableAfterRate	: 지속적 값 변경

트러블 슈팅

```
335         // 해당 리뷰 삭제
336         $("#review-" + reviewNumber).remove();
337         // 총 리뷰 수 -1으로 설정
338         $("#totalReviewCount").text(Number($("#totalReviewCount").text()) - 1);
339         // 리뷰 없음 글 설정
340         makeNoReviewText();
```

```
355 // 리뷰 존재 여부 안내 문구 기능
356 const makeNoReviewText = () => {
357     console.log("리뷰 존재 여부 확인");
358     if ($("#reviewWrapper").is(':empty')) {
359         console.log("리뷰가 화면에 없음");
360         $("#noReview").show();
361     }
362     else {
363         console.log("리뷰가 화면에 있음");
364         $("#noReview").hide();
365     }
366 }
```

트러블 슈팅

삭제할 리뷰 번호 : 13]	productDetail.js:316
삭제할 의사 : true]	productDetail.js:319
리뷰 삭제 성공	productDetail.js:333
리뷰 존재 여부 확인	productDetail.js:357
리뷰가 화면에 있음	productDetail.js:363

- ? 발생 원인** ArrayList의 .isEmpty() 메서드와 닮아서 사용해본 is(':empty')는 내부에 공백만 있어도 false를 반환
- ! 해결 방안** 대신에 확실한 .children().length === 0으로 변경

트러블 슈팅

```
376     // 수정할 리뷰의 disabled 속성 제거
377     $("#reviewComment-" + reviewNumber).prop("disabled", false);
378     $('#rating-reply-' + reviewNumber).starRating('setOptions', {
379         readOnly: false,      // 별점 수정 가능
380         disableAfterRate: false // 별점 클릭 후에도 수정 가능
381     });

```

트러블 슈팅

```
✖ ► Uncaught Error: Method jquery.min.js:4  
    setOptions does not exist on starRating.js  
        at x.error (jquery.min.js:4:3549)  
        at $.fn.<computed> [as starRating] (  
            jquery.star-rating-svg.js:308:11)  
        at updateReview (productDetail.js:378:38)  
        at HTMLButtonElement.<anonymous> (  
            productDetail.js:44:3)  
        at HTMLDocument.dispatch (jquery.min.js:5:9505  
)  
        at y.handle (jquery.min.js:5:6288)
```

? 발생 원인

사용 중인 star-rating 플러그인 ([jquery.star-rating-svg.js](#))에는 "setOptions" 같은 메서드가 정의되어 있지 않음

! 해결 방안

비동기 생성된 DOM 요소에 별점 플러그인을 다시 적용하려면 기존 요소 상태가 꼬이므로 `replaceWith()`로 해당 DOM 요소를 새로 만들고 그 후에 `.starRating()`을 적용

04 _ API 문서

06. 카카오 주소 API



```
<script  
src="//t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>  
<script type="text/javascript" src="//dapi.kakao.com/v2/maps/sdk.js?appkey=  
{개인JS앱키}&libraries=services"></script>
```

Request

url : 카카오 서버 URL
JavaScript key : 앱 (JS) 키

Response

addr : 주소 변수 (도로명, 지번)
extraAddr : 참고항목 변수 (건물명 등)

The screenshot shows two instances of the Daum Postcode Service search interface. The left window shows the search bar with '코리아' and a red box highlighting it. Below the search bar is a 'tip' section with search tips. The right window shows the search results for '코리아', displaying two entries: one for zip code 12446 and another for 12463. Each entry includes the address and a link to '영문보기' (English view) and '지도' (Map).

Daum Postcode Service - Chrome

about:blank

코리아

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

Daum Postcode Service - Chrome

about:blank

코리아

검색결과가 많습니다. 검색어에 아래와 같은 조합을 이용하시면 더욱 정확한 결과가 검색됩니다. '[도로명+건물번호](#)', '[지역명+지번](#)', '[지역명+건물명\(아파트명\)](#)', '[사서함명+번호](#)'

도로명 전체 지역명 전체

12446 영문보기 | 지도
도로명 경기 가평군 상면 녹수계곡로 157-72 (뉴스킨코리아랑연쉼터)

경기 가평군 상면 녹수계곡로 157-72

지번 경기 가평군 상면 덕현리 599-1
↳ 외 지번주소 1건 더보기

12463 영문보기 | 지도
도로명 경기 가평군 설악면 한서로 421-259 (한국경노원&코리아실버비전센터)

경기 가평군 설악면 위곡리 33-1
↳ 외 지번주소 2건 더보기

03 _ 프로젝트 기능

06. 아이디 중복 검사

```
18     // 2. 로그인
19•     final String SELECTONELOGIN = "SELECT MEMBER_NUMBER, MEMBER_ID, MEMBER_NAME,
20             + " MEMBER_IS_ADMIN FROM MEMBER WHERE MEMBER_ID = ? AND MEMBER_PASSWORD = ?";
21
```

아이디와 비밀번호가 일치하면 해당 회원의 정보를 불러옴

아이디

풀조합팩토리

가입할 아이디 입력

```
37      <label for="memberId">아이디</label> <input type="text"
38          name="memberId" class="form-control" id="memberId"
39          placeholder="아이디를 입력해주세요" onblur="checkJoinMemberId(event)"
40          pattern="^[a-zA-Z0-9]{2,15}$" maxlength="15" required>
41      <div class="invalid-feedback" id="memberIdFeedback">영어와 숫자만
42          세글자 이상 입력</div>
```

포커스아웃 시 작동
네이버 회원가입 참고

```
110 // 아이디 사용 여부 검사 기능
111 const checkJoinMemberId = (event) => {
112   // 아이디 입력 패턴
113   const memberIdPattern = /^[a-z][a-z0-9]{2,15}$/;
114   // 행동이 일어난 대상의 값(양쪽 여백 제거) 받아오기
115   let joinMemberId = $(event.target).val().trim();
116   const isPatternValid = memberIdPattern.test(joinMemberId);
117   const memberId = $("#memberId");
118   const memberIdFeedback = $("#memberIdFeedback");
119
120   // 로그 찍기
121   console.log("JS 입력받은 joinMemberId : [" + joinMemberId + "]");
122
123   // 입력값이 없으면 요청하지 않음
124   if (!joinMemberId) {
125     memberId.removeClass("is-valid is-invalid");
126     memberIdFeedback.hide();
127     console.log("입력값이 비어 있어 요청하지 않음");
128     return;
129   }
130
131   // 입력 패턴에 맞지 않다면
132   if (!isPatternValid) {
133     // UI/UX
134     memberId.addClass("is-invalid").removeClass("is-valid");
135     // 사용자에게 패턴 오류 알림
136     memberIdFeedback.text("영어와 숫자만 세글자 이상 입력").show();
137     return;
138 }
```

아이디 입력 패턴 상수화

입력 값이 없을 시 종료
서버, DB 비용 절감

입력 패턴 불통과 시 종료

```
140 $.ajax({
141   type: "POST", // 방식
142   url: checkJoinMemberIdUrl, // 찾아갈 주소
143   data: { memberId: joinMemberId }, // 보낼 값
144   dataType: "text", // 받을 타입
145   success: (response) => { // 비동기 정상 작동 시
146     console.log("받은 아이디 중복 여부 데이터:", response);
147     if (response === "true") { // 사용 가능
148       memberId.addClass("is-valid").removeClass("is-invalid");
149       memberIdFeedback.hide();
150     } else if (response === "false") { // 아이디 중복 (사용 불가)
151       memberId.addClass("is-invalid").removeClass("is-valid");
152       memberIdFeedback.text("이미 사용 중인 아이디입니다.").show();
153     }
154   },
155   error: (xhr, status, error) => { // 비동기 작동 실패 시
156     console.error("AJAX 요청 에러 발생", xhr.status, status, error);
157     alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
158     location.href = "error.do";
159   }
160 });
161 }
```

비동기로 Service에
가입 아이디 값 전달

text 타입으로 받는 이유

1. 간단하고 직관적임
2. JSON보다 크기가 작음
3. 전송 속도가 빠름

화면에 사용 가능 여부 표시



< Home 이동

회원가입

아이디

 ×

영어와 숫자만 세글자 이상 입력

비밀번호

비밀번호 확인

아이디 사용 가능 여부 확인

```
38 String joinMemberId = request.getParameter("memberId");
39
40 boolean isAvailable = false;
41
42 if (joinMemberId != null && !joinMemberId.trim().isEmpty()) {
43     MemberDAO memberDAO = new MemberDAO();
44     MemberDTO memberDTO = new MemberDTO();
45     memberDTO.setCondition("SELECTONEMEMBER");
46     memberDTO.setMemberId(joinMemberId);
47
48     memberDTO = memberDAO.selectOne(memberDTO);
49
50     if(memberDTO == null) {
51         isAvailable = true;
52     } else if(memberDTO != null) {
53         isAvailable = false;
54     }
55 }
56
57 System.out.println("중복 여부 ["+isAvailable+"]");
```

사용자가 입력한 아이디 데이터를 전송, 확인 요청

입력한 아이디 데이터를 DB에서 검색

회원이 존재하지 않는다면 true

회원이 존재하면 false 반환

프론트로 결과 값을 반환하여

사용자에게 아이디 사용 가능 여부 안내

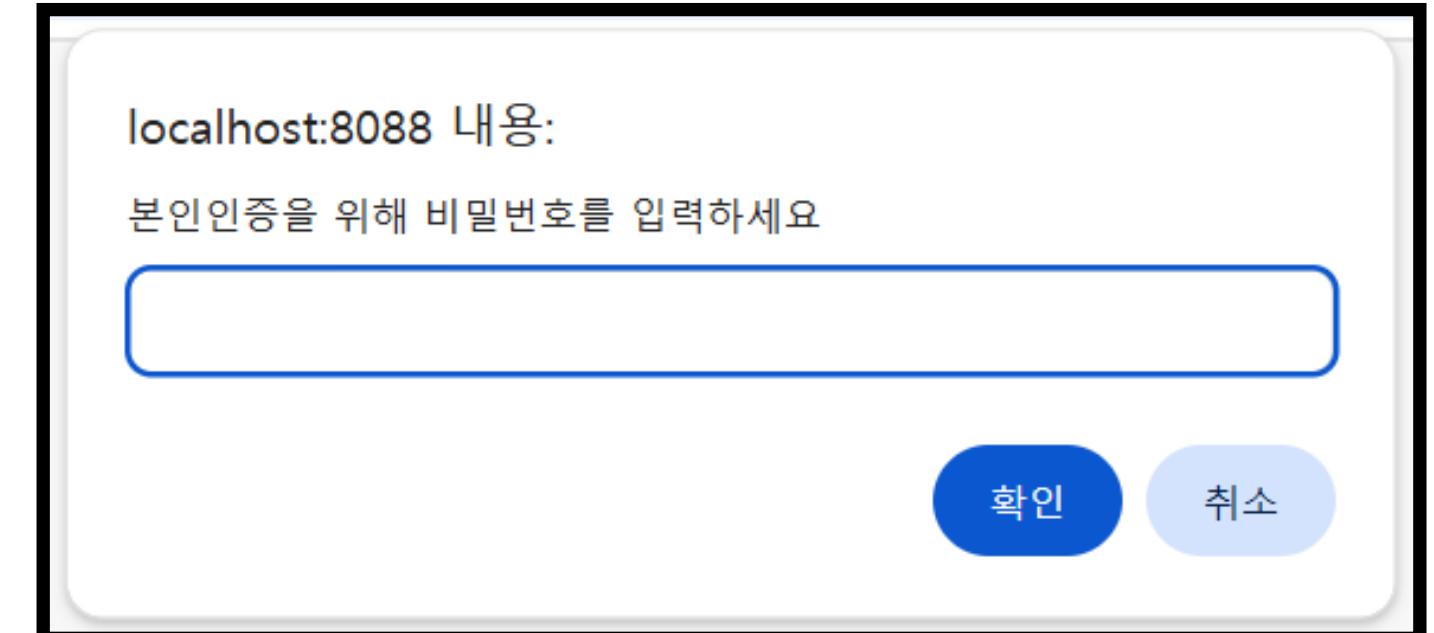
03 _ 프로젝트 기능

07. 페이지 이동

```

365 // 본인확인 기능
366 const checkLoggedInMember = () => {
367   return new Promise((resolve) => {
368     let loginedMemberPassword = prompt("본인인증을 위해 비밀번호를 입력하세요");
369
370     // 사용자가 취소 버튼을 눌렀을 경우
371     if (loginedMemberPassword === null) {
372       console.log("본인인증 입력 취소됨");
373       alert("본인 인증이 취소되었습니다.");
374       resolve(false);
375       return;
376     }
377
378     console.log("입력받은 본인확인 비밀번호 : [" + loginedMemberPassword + "]");
379
380     $.ajax({
381       type: "POST", // 방식
382       url: checkLoggedInMemberUrl, // 찾아갈 주소
383       data: { loginedMemberPassword: loginedMemberPassword }, // 보낼 값
384       datatype: "text", // 받을 타입
385       success: (response) => { // 성공적이라면
386         if (response === "true") { // 인증 성공 시
387           console.log("인증 성공");
388           resolve(true);
389         } else { // 인증 실패 시
390           console.log("인증 실패");
391           alert("본인 인증에 실패하셨습니다. 비밀번호를 확인해주세요.");
392           resolve(false);
393         }
394       },

```



인증 취소 시 비동기 사용 X

비동기로 백단에
입력받은 비밀번호
값 전달

```
33     if (session.getAttribute("loginedMemberNumber") == null) {  
34         forward.setPath("login.jsp");  
35         forward.setRedirect(true);  
36         return forward;  
37     }
```

사용자가 로그인에 성공하지 못하면
내 정보 페이지, 장바구니 페이지 등 본인 확인이 필요한 페이지 이동에 대해서는
접근을 차단
로그인 페이지로 유도

03 _ 프로젝트 기능

08. 검색, 정렬

검색 시 인기순 우선 노출

1. 개별상품으로 가격 계산
2. 재고 결정
3. 인기순 정렬을 위한 판매수량 집계
4. 검색을 위한 CONCAT 함수
5. 집계를 위한 GROUP 함수

```
111 // 꿀조합 상품 이름 검색 기능
112•const searchComboProductName = () => {
113     console.log("꿀조합 상품 이름 검색 실행");
114     productState.searchKeyword = $("#" + "#searchKeyword").val();
115     console.log("검색한 꿀조합 상품 이름 : [" + productState.searchKeyword + "]");
116
117•    if (!productState.searchKeyword) { // 빈 내용을 검색한다면
118        // 꿀조합 상품 불러오기 함수 호출
119        loadComboDatas();
120        // 첫 페이지 설정 필요
121        productState.searchKeywordPageFlag = true;
122    }
123
124•    if (productState.searchKeywordPageFlag) { // 처음 검색이라면
125        // 페이지 번호 초기화
126        productState.pageNumber["combo"] = 1;
127        // 첫 페이지 설정 불필요
128        productState.searchKeywordPageFlag = false;
129    }
```

빈 내용 검색 시
상품 출력 기준 초기화

비연속적으로 검색 시
페이지 번호 1로 초기화

```
131     console.log("검색할 조건 : [" + productState.orderCondition + "]");
132     $.ajax({ // 비동기
133         url: searchComboProductNameUrl, // 보낼 주소
134         type: 'GET', // 방식
135         data: { // 보낼 값
136             searchKeyword: productState.searchKeyword,
137             comboProductPageNumber: productState.pageNumber["combo"],
138             comboProductContentCount: productState.contentCount["combo"],
139             comboProductOrderCondition: productState.orderCondition
140         },
141         dataType: 'json', // 받을 타입
142         success: (response) => { // 성공적이라면
143             console.log("받은 검색 꿀조합 상품 데이터 : [" + response + "]"); // 로그 찍기
144             // 카테고리는 전부로 설정
145             productState.category = "ALLPRODUCT";
146
147             // 꿀조합 상품 생성 함수 호출
148             insert("combo", response);
149         },
150         error: (xhr, status, error) => { // 에러 처리
151             console.error("AJAX 요청 에러 발생", xhr.status, status, error);
152             alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
153             location.href = "error.do";
154         }
155     });
156 }
```

비동기로 백단에

1.검색어

2.페이지 수

3.받아올 데이터 수

4.정렬 조건

값들 전달

카테고리는 ALLPRODUCT
화면에 상품 출력

```
79 // 꿀조합 상품 정렬 기능
80 const setComboProductOrderCondition = (orderCondition) => {
81   console.log("요청받은 상품 정렬 조건 : [" + orderCondition + "]");
82
83 if (productState.orderCondition === orderCondition) { // 정렬 조건이 변함없다면
84   console.log("변동 없으므로 중단");
85   return;
86 }
87
88 // 정렬 조건 변경
89 productState.orderCondition = orderCondition;
90 // 검색어 변경
91 const nowSearchKeyword = $("#searchKeyword").val();
92 productState.searchKeyword = nowSearchKeyword;
93 // 페이지 번호 초기화
94 productState.pageNumber["combo"] = 1;
95
96 console.log("정렬 시 현재 검색된 꿀조합 상품 정렬 조건 : [" + productState.orderCondition + "]");
97 console.log("정렬 시 현재 검색된 검색어 : [" + productState.searchKeyword + "]");
98
99 if (!productState.searchKeyword) { // 검색된 상품 이름이 없다면
100   // 꿀조합 불러오기 함수 호출
101   loadDatas("combo");
102 }
103 else { // 검색된 꿀조합 상품 이름이 있다면
104   // 첫 페이지 설정 필요
105   productState.searchKeywordPageFlag = true;
106   // 꿀조합 상품 이름 검색 함수 호출
107   searchComboProductName();
108 }
109 }
```

정렬 조건 변경 여부 검사

정렬 조건 변경 후

1. 검색어가 있다면

상품 이름 검색

2. 검색어가 없다면

일반 상품 정렬 변경

```
183 // 꿀조합 상품 불러오기 기능
184 const loadDatas = (type) => {
185     console.log("불러올 상품 타입 : [" + type + "]");
186
187     $.ajax({ // 비동기
188         url: loadMoreComboUrl, // 보낼 주소
189         type: 'GET', // 방식
190         data: { // 보낼 값
191             comboProductPageNumber: productState.pageNumber[type],
192             comboProductContentCount: productState.contentCount[type],
193             comboProductOrderCondition: productState.orderCondition,
194             comboProductCategory: productState.category
195         },
196         dataType: 'json', // 받을 타입
197         success: (response) => { // 성공적이라면
198             console.log("받은 꿀조합 상품 데이터 : [" + response + "]"); // 로그 찍기
199
200             // 꿀조합 상품 생성 함수 호출
201             insert(type, response);
202         },
203         error: (xhr, status, error) => { // 에러 처리
204             console.error("AJAX 요청 에러 발생", xhr.status, status, error);
205             alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
206             location.href = "error.do";
207         }
208     });
209 }
```

비동기로 백단에

- 페이지 번호
- 받아올 데이터 수
- 정렬 조건
- 카테고리
값들 전달

화면에 상품 출력 함수 실행

03 _ 프로젝트 기능

09. 카테고리, 정렬

- 카테고리 조회 시 가격 낮은 순으로 정렬 구현
- 개별상품의 가격으로 콤보상품 가격 계산
- 카테고리별로 개별상품 가격 계산
- 가격순 정렬을 위해 개별상품들의 가격들을 조인

```
183 // 꿀조합 상품 불러오기 기능
184 const loadDatas = (type) => {
185     console.log("불러올 상품 타입 : [" + type + "]");
186
187     $.ajax({ // 비동기
188         uri: 'loadmorecombouri', // 보낼 주소
189         type: 'GET', // 방식
190         data: { // 보낼 값
191             comboProductPageNumber: productState.pageNumber[type],
192             comboProductContentCount: productState.contentCount[type],
193             comboProductOrderCondition: productState.orderCondition,
194             comboProductCategory: productState.category
195         },
196         dataType: 'json', // 받을 타입
197         success: (response) => { // 성공적이라면
198             console.log("받은 꿀조합 상품 데이터 : [" + response + "]"); // 로그 찍기
199
200             // 꿀조합 상품 생성 함수 호출
201             insert(type, response);
202         },
203         error: (xhr, status, error) => { // 에러 처리
204             console.error("AJAX 요청 에러 발생", xhr.status, status, error);
205             alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
206             location.href = "error.do";
207         }
208     });
209 }
```

비동기로 백단에

- 페이지 번호
- 받아올 데이터 수
- 정렬 조건
- 카테고리
값들 전달

05 오류 원인 분석 및 해결 방안

05. 오류 원인 분석 및 해결 방안

01. DB

트러블 슈팅

```
java.sql.SQLSyntaxErrorException: Column 'MEMBER_ID' not found.  
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException (SQLError.java:110)  
    at com.mysql.cj.jdbc.result.ResultSetImpl.findColumn (ResultSetImpl.java:1000)  
    at com.mysql.cj.jdbc.result.ResultSetImpl.getString (ResultSetImpl.java:1030)  
  
[74]         System.out.println("쿼리 실행 조건: " + condition);  
[75] //         System.out.println("결과: MEMBER_NUMBER = " + rs.getLong("MEMBER_NUMBER"));  
[76] //         System.out.println("결과: MEMBER_ID = " + rs.getString("MEMBER_ID"));  
[77]  
[78]         data = new MemberDTO();
```

- ? 발생 원인** MEMBER_ID를 SELECT한 적이 없는데 로그를 사용해서
not found 에러 발생
- ! 해결 방안** 해당 쿼리에서는 MEMBER_ID가 필요 없으므로 로그 삭제

```
 ${not empty purchaseProductDetailData.productComboName}">
  lspan="2"><span> ${purchaseProductDetailData.productComboName} </spa
  purchaseProductDetailData.purchaseProductCount} </th>
  pan> ${purchaseProductDetailData.purchaseProductCount * purchasePro
  ${not empty purchaseProductDetailData.productSingleName}">
  lspan="2"><span> ${purchaseProductDetailData.productSingleName} </sp
  purchaseProductDetailData.purchaseProductCount} </th>
  pan> ${purchaseProductDetailData.purchaseProductCount * purchasePro
```

```
 productSingleName=, productComboName=,
 :eNumber=15, productSingleName=, produ
```

？ 발생 원인

view에서는 if문을 통해서 productComboName, productSingleName을 순회하고 있는데 model에서 보내는 데이터는 name이 NULL값이어서 발생

! 해결 방안

보내는 데이터에 productComboName과 productSingleName을 추가하여 NULL 방지

```
java.sql.SQLException: Parameter index out of range (1 > number of parameters, which is 0).
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:121)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:89)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:81)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:55)
    at com.mysql.cj.jdbc.ClientPreparedStatement.checkBounds(ClientPreparedStatement.java:1482)
    at com.mysql.cj.jdbc.ClientPreparedStatement.getCoreParameterIndex(ClientPreparedStatement.java:1497)
    at com.mysql.cj.jdbc.ClientPreparedStatement.setInt(ClientPreparedStatement.java:1769)
    at model.dao.BoardComboDAO.selectAll(BoardComboDAO.java:60)
    at controller.move.action.BoardComboMoveAction.execute(BoardComboMoveAction.java:48)
    at controller.common.FrontControllerDo.doAction(FrontControllerDo.java:32)
    at controller.common.FrontControllerDo.doGet(FrontControllerDo.java:54)
```

？ 발생 원인

SQL과 파라미터 바인딩 수가 불일치해서 발생한 에러
페이지네이션을 위해 시작 인덱스 번호와 한 페이지에
출력될 데이터 수를 보내줘야 하는데 SQL문에 LIMIT절을 빠뜨렸기 때문임
SQL문에 LIMIT ?, ?를 추가하고 바인딩에는
그대로 setInt~getIndex, getCount

! 해결 방안

번호	작성자	제목	조회수	좋아요	작성일
9	관리자	꿀조합 콘테스트 안내	201	-	2025-03-12
4	관리자	재고 입고 문의 관련 안내드립니다.	167	-	2025-02-19
3	관리자	꿀조합 팩토리 관리자가 추천하는 조합	311	-	2025-01-13
2	관리자	꿀조합 게시판 사용 시 이용수칙	382	-	2025-01-02
1	관리자	꿀조합 게시판에 오신 걸 환영합니다	441	-	2025-01-01
15	최수진	스트레스 날릴 때 이거!	205	0	2025-03-30
14	김석환	따뜻하게 먹으면 꿀맛	231	0	2025-03-30
13	이승기	GS 강추 조합 나왔습니다	241	0	2025-03-27
12	김민수	CU에서만 가능한 조합!	50	0	2025-03-21
11	John Doe	친구랑 먹다 감탄함	124	0	2025-03-19

? 발생 원인

서브쿼리에 사용되는 AS LIKEDCOUNT와 좋아요 수를 저장할 멤버변수 BOARDCOMBOLIKEDCOUNT가 혼동되어서 발생

DTO와 DAO를 모두 수정

서블릿에도 BOARDCOMBOLIKEDCOUNT로 통일하여 좋아요 수 반영 성공

! 해결 방안

```
경고: Unable to find CDP implementation matching 134
4월 01, 2025 9:07:56 오후 org.openqa.selenium.chromium.ChromiumDriver lambda$new$4
경고: Unable to find version of CDP to use for 134.0.6998.178. You may need to include a dependency
4월 01, 2025 9:08:41 오후 io.opentelemetry.api.GlobalOpenTelemetry maybeAutoConfigureAndSetGlobal
정보: AutoConfiguredOpenTelemetrySdk found on classpath but automatic configuration is disabled. To
[GS25/+1] Y)면왕(소컵)105G / 990원
java.sql.SQLIntegrityConstraintViolationException: Duplicate entry '0' for key 'product_single.PR
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:109)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapp
[GS25/+1] CJ)비비고버터오징어김스낵40G / 4900원
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:10
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:14
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:13
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeLargeUpdate(ClientPreparedStatement.java:18
    at com.mysql.ci.jdbc.ClientPreparedStatement.executeUpdate(ClientPreparedStatement.java:100)
```

?**발생 원인** 기본키 중복 오류 → 직접 번호를 생성해서 발생하는 오류

!**해결 방안** 중복방지 로직 추가

```
com.mysql.cj.jdbc.exceptions.MySQLDataTruncation: Data truncation: Data too long for column 'PRODUCT_SINGLE_NAME' at row 1
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:96)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:990)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:1168)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:1103)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeLargeUpdate(ClientPreparedStatement.java:1450)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdate(ClientPreparedStatement.java:1086)
    at model.dao.ProductSingleDAO.insert(ProductSingleDAO.java:230)
    at model.crawling.StoreGS25.crawlCategory(StoreGS25.java:125)
    at model.crawling.StoreGS25.makeSampleGS25(StoreGS25.java:28)
    at model.crawling.StoreGS25.main(StoreGS25.java:136)
[GS25/GOODS] 유어스(P)NEW3단우산55CM(블랙) / 12000원
```

？ 발생 원인 문자 길이 초과 오류

! 해결 방안 DB테이블에서 칼럼길이(상품명) 늘리기

```
at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:4330)
at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:164)
... 25 more
Caused by: java.lang.ClassNotFoundException: ProductDAO
at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1332)
at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1144)
... 38 more

4월 01, 2025 2:20:21 오후 org.apache.coyote.AbstractProtocol pause
정보: 프로토콜 핸들러 ["http-nio-8088"]을(를) 일시 정지 중
4월 01, 2025 2:20:21 오후 org.apache.catalina.core.StandardService stopInternal
정보: 서비스 [Catalina]을(를) 중지시킵니다.
4월 01, 2025 2:20:21 오후 org.apache.coyote.AbstractProtocol destroy
정보: 프로토콜 핸들러 ["http-nio-8088"]을(를) 소멸시킵니다.
```

❓ 발생 원인 클래스 누락(ClassNotFoundException) 오류

❗ 해결 방안 Eclipse/IDE에서 Project >> Clean 후 다시 Build

[GS25/GOODS] 유어스(P)니트릴장갑50매(블랙) / 9900원

이미지 URL: https://image.woodongs.com/imgsvr/item/GD_8801222002887_002.jpg

```
java.sql.SQLException: You can't specify target table 'PRODUCT_SINGLE' for update in FROM clause
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:121)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:510)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:644)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdateInternal(ClientPreparedStatement.java:633)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeLargeUpdate(ClientPreparedStatement.java:670)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeUpdate(ClientPreparedStatement.java:161)
    at model.dao.ProductSingleDAO.insert(ProductSingleDAO.java:208)
    at model.crawling.StoreGS25.crawlCategory(StoreGS25.java:125)
    at model.crawling.StoreGS25.makeSampleGS25(StoreGS25.java:28)
    at model.crawling.StoreGS25.main(StoreGS25.java:136)
```

[GS25/GOODS] 유어스)세이프함균커퍼칼 / 3500원

이미지 URL: https://image.woodongs.com/imgsvr/item/GD_8801067940283_002.jpg

? 발생 원인

DB(MYSQL) 테이블에서 UPDATE 대상과 DELETE 대상을
FROM 절에서 동시에 사용 (MySQL 제약)

! 해결 방안

자바 분기 방식으로 해결
IF문으로 조건을 설정하고 만족하는 경우 SQL문 수행

05. 오류 원인 분석 및 해결 방안

02. Front

```

9 let loginedMemberNumber; // 로그인한 회원 번호
10 let isLiked = false; // 게시글 좋아요 여부
11
12 $(document).ready(function() {
13     $(document).on("click", "#likeBtn", function(event) { // 좋아요 클릭 시
14         event.preventDefault(); // 기본 이벤트 방지
15
16         clickLiked(); // 좋아요 여부 변경 함수 호출
17     });
18
19     // 로그인 상태 확인
20     loginedMemberNumber = sessionStorage.getItem('loginedMemberNumber');
21     console.log("꿀조합 게시글 상세 페이지 접근 시 로그인 정보 : [" + loginedMemberNumber + "]");
22
23     // 좋아요 여부 불러오기 함수 호출
24     checkLiked();
25 });

```

꿀조합 게시글 상세 페이지 접근 시 로그인 정보 : [mainPage.js:10](#)
[null]

? 발생 원인

js 외부 파일에서는 session을 사용할 수가 없음
해결하기 위해 sessionStorage를 사용하였지만 둘은 별개로 작동

! 해결 방안

로그인 시 sessionStorage.setItem을 사용하여
JS만의 session을 사용

✖ ► Uncaught TypeError: Cannot read properties of undefined (reading '0')
at insertHot ([CUProduct.js:548:33](#))
at Object.success ([CUProduct.js:485:4](#))
at i ([jquery-1.12.4.min.js:2:27449](#))
at Object.fireWith [as resolveWith] ([jquery-1.12.4.min.js:2:28213](#))
at y ([jquery-1.12.4.min.js:4:22721](#))
at XMLHttpRequest.c ([jquery-1.12.4.min.js:4:26925](#))

❓ 발생 원인 Controller에게 받은 response의 구조가 코딩해둔 반복문과 맞지 않기 때문

❗ 해결 방안 out.print(jsonObj,Array)라면 response.datas[0].totalCountNumber로 수정
out.print(jsonObj,obj)라면 response[0].totalCountNumber로 반복

```

365 // 본인확인 기능
366 const checkLoggedInMember = () => {
367   return new Promise((resolve) => {
368     let loginedMemberPassword = prompt("본인인증을 위해 비밀번호를 입력하세요");
369
370     console.log("입력받은 본인확인 비밀번호 : [" + loginedMemberPassword + "]");
371
372   $.ajax({
373     type: "POST", // 방식
374     url: checkLoggedInMemberUrl, // 찾아갈 주소
375     data: { loginedMemberPassword: loginedMemberPassword }, // 보낼 값
376     dataType: "text", // 받을 타입
377     success: (response) => { // 성공적이라면
378       if (response === "true") { // 인증 성공 시
379         console.log("인증 성공");
380         resolve(true);
381     } else { // 인증 실패 시
382       console.log("인증 실패");
383       alert("본인 인증에 실패하셨습니다. 비밀번호를 확인해주세요.");
384       resolve(false);
385     }
386   },
387   error: (xhr, status, error) => { // 에러 처리
388     console.error("AJAX 요청 에러 발생", xhr.status, status, error);
389     alert("서버에 문제가 발생했습니다. 지속될 시 관리자에게 문의하세요.");
390     // 에러 상황에서도 흐름을 멈추지 않도록 false 반환
391     resolve(false);
392     location.href = "error.do";
393   }
394 });
395 });
396 };

```

```

37 // 본인 인증 확인
38 if(checkLoggedInMember()){
39   location.href = "main.do";
40   return;
41 }

```

? 발생 원인

비동기는 백그라운드에서 동작
 success 콜백은
 언젠가 서버가 응답을 주면 실행
 하지만 JS는 기다리지 않고 다음 줄을 실행

! 해결 방안

1. 콜백을 사용
2. Promise + async/await을 사용

jQuery의 `$.ajax()`는 Promise 기반이
 아니라서 `await`은 사용 불가



312 // 반복하며 화면에 상품 생성
313 response.forEach((type+productData => {

？ 발생 원인 forEach의 매개변수는 반드시 변수 하나를 받아야함

! 해결 방안 type+productData를 변수에 담고 window[변수명]으로 사용
window[변수명]으로 사용할 변수 타입이 let, const일 경우 사용 불가

149
150

```
// 페이지 번호 초기화  
window[pageNumber] = 1;
```

? 발생 원인

키값으로 window[]의 값이 1로 저장될 뿐이며 전역 변수로 선언한 변수값이 10이 되는 것이 아니기 때문

! 해결 방안

const productState = {}
전역 맵 객체를 선언하여 productState.pageNumber[type] 형태로 값을 뽑거나 변경하여 사용

05_오류 원인 분석 및 해결 방안

03. Service

05 _ 오류 원인 분석 및 해결 방안

```

60 // JSON 배열로 상품 정보를 변환
61 JSONArray boardComboArray = new JSONArray();
62 for (BoardComboDTO boardCombo : boardComboDatas) {
63     JSONObject boardComboObject = new JSONObject();
64     boardComboObject.put("boardComboNumber", boardCombo.getBoardComboNumber());
65     boardComboObject.put("memberName", boardCombo.getMemberName());
66     boardComboObject.put("boardComboTitle", boardCombo.getBoardComboTitle());
67     boardComboObject.put("boardComboViewCount", boardCombo.getBoardComboViewCount());
68     boardComboObject.put("boardComboLikedCount", boardCombo.getBoardComboLikedCount());
69     boardComboObject.put("boardComboRegisterDate", boardCombo.getBoardComboRegisterDate().toString());
70     boardComboObject.put("totalCountNumber", boardCombo.getTotalCountNumber());
71     boardComboArray.add(boardComboObject);
72
73     System.out.println("내보내는 데이터 [" + boardComboObject + "]");
74 }
75
76 JSONObject boardComboData = new JSONObject();
77
78 boardComboData.put("boardComboDatas", boardComboArray);

```

```

60 // JSON 배열로 상품 정보를 변환
61 JSONArray boardComboArray = new JSONArray();
62 for (BoardComboDTO boardCombo : boardComboDatas) {
63     JSONObject boardComboObject = new JSONObject();
64     boardComboObject.put("boardComboNumber", boardCombo.getBoardComboNumber());
65     boardComboObject.put("memberName", boardCombo.getMemberName());
66     boardComboObject.put("boardComboTitle", boardCombo.getBoardComboTitle());
67     boardComboObject.put("boardComboViewCount", boardCombo.getBoardComboViewCount());
68     boardComboObject.put("boardComboLikedCount", boardCombo.getBoardComboLikedCount());
69     boardComboObject.put("boardComboRegisterDate", boardCombo.getBoardComboRegisterDate().toString());
70     boardComboObject.put("totalCountNumber", boardCombo.getTotalCountNumber());
71     boardComboArray.add(boardComboObject);
72
73     System.out.println("내보내는 데이터 [" + boardComboObject + "]");
74 }
75
76
77 response.setContentType("application/json");
78 response.setCharacterEncoding("UTF-8");
79
80 response.getWriter().print(boardComboArray.toJSONString());
81 response.getWriter().flush();
82 response.getWriter().close();
83

```

? 발생 원인

**JSONArray 타입으로 내보낼 경우
View에서 데이터를 인식하지 못하는
상황 발생**

! 해결 방안

**JSONObject로 데이터를
재포장 한 후 데이터 전송
→ 데이터 올바르게 파싱 및 사용
JSONObject를 사용하여
데이터 구조를 명확하게 정의
→ 데이터 처리의 정확성과 효율성
높임**

```
49      // 생년월일 날짜 형변환  
50      memberDTO.setMemberBirth(memberBirth);  
51      System.out.println("아이디 찾을 생년월일 [" + memberBirth + "]");
```

 50 The method setMemberBirth(Date) in the type MemberDTO is not applicable for the arguments (String)

생년월일

2025. 04. 01.



? 발생 원인

MemberDTO에는 Date로 선언이 되어있지만
클라이언트에서 보내주는 날짜의 형식을 그대로 저장할 수 없음

! 해결 방안

java.sql.Date.valueOf() 메소드는 문자열을
java.sql.Date 타입으로 직접 변환하는데 사용

! 코드 개선

```

58 ArrayList<Object> shoppingCart = (ArrayList<Object>)session.getAttribute("shoppingCart");
59
60 for(Object cartProduct : shoppingCart) {
61     // 개별상품이 이미 장바구니에 담겨 있다면
62     if(cartProduct instanceof ProductSingleDTO && productSingleDTO != null) {
63         ProductSingleDTO alreadyInProductSingle = (ProductSingleDTO) cartProduct;
64         if(alreadyInProductSingle.getProductSingleNumber() == productSingleDTO.getProductSingleNumber()) {
65             alreadyInProductSingle.setProductSingleCount(alreadyInProductSingle.getProductSingleCount() + 1);
66             alreadyIn = true;
67
68             System.out.println("저장 여부 [" +alreadyIn + "]");
69
70             break;
71     }
72
73     // 조합상품이 이미 장바구니에 담겨 있다면
74     else if(cartProduct instanceof ProductComboDTO && productComboDTO != null) {
75         ProductComboDTO alreadyInProductCombo = (ProductComboDTO) cartProduct;
76         if(alreadyInProductCombo.getProductComboNumber() == productComboDTO.getProductComboNumber()) {
77             alreadyInProductCombo.setProductComboCount(alreadyInProductCombo.getProductComboCount() + 1);
78             alreadyIn = true;
79
80             System.out.println("저장 여부 [" +alreadyIn + "]");
81
82             break;
83     }
84 }
85 }
86

```

기존 코드에서는 instanceof를 사용하여 타입을 확인 후, 적절한 타입으로 캐스팅 후 필드를 업데이트하는 복잡한 과정을 거침

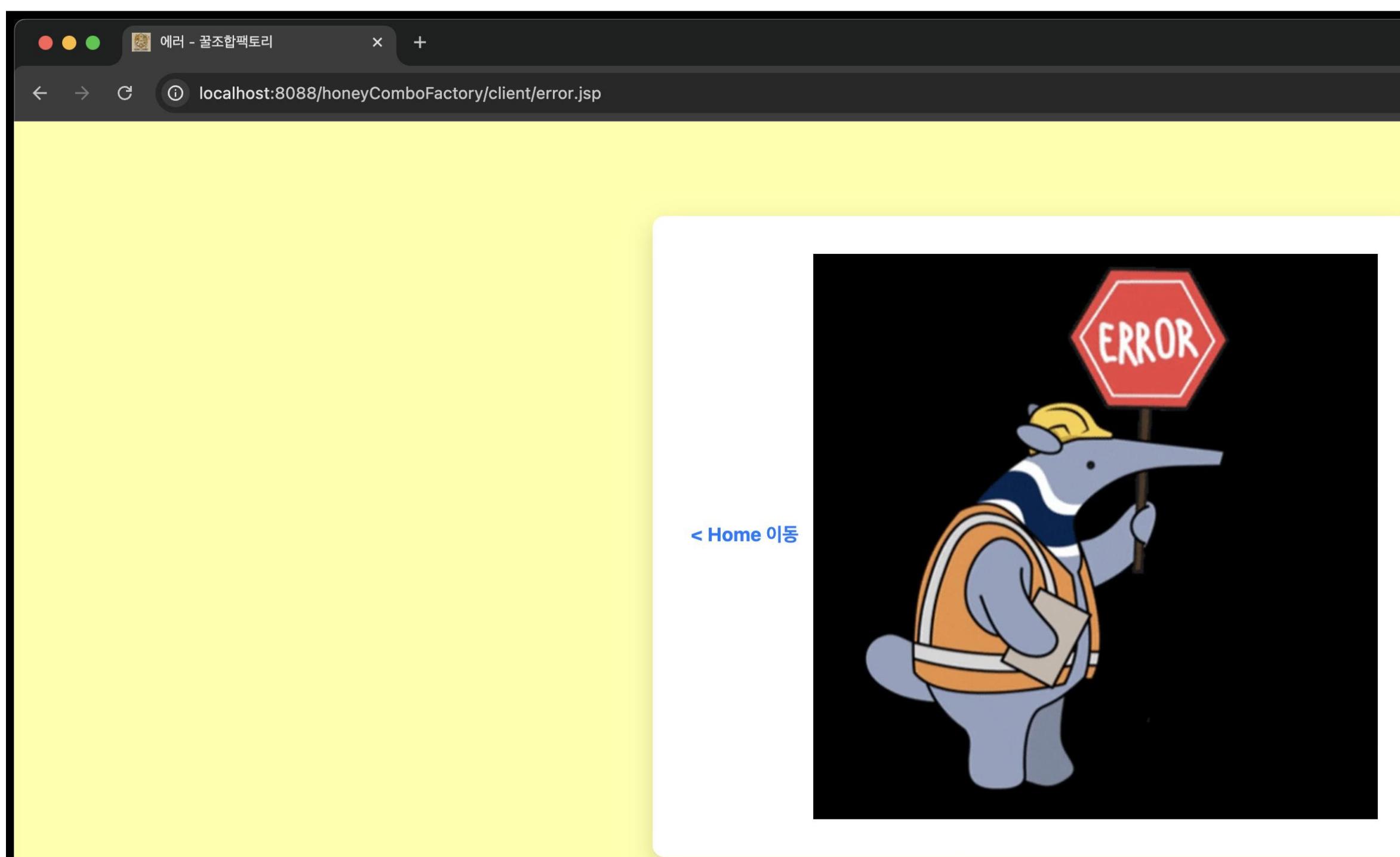
- 개선된 코드에서는 Map을 사용하여 필요한 값만 간단하게 추출하고 즉시 업데이트하는 방식으로 변경
- 반복문과 조건문의 구조가 간결, 처리해야 할 로직이 명확하게 표현 → 코드의 가독성과 유지보수성이 개선
- 개선된 코드는 Map<String, Object>를 사용하여 다양한 타입의 데이터를 유연하게 다루고, 필요한 데이터를 쉽게 접근하고 수정 가능
→ 데이터 구조의 변화에 대응하기 더 용이

```

52     for (Map<String, Object> cartItem : shoppingCart) {
53         if ((int) cartItem.get("productNumber") == productNumber && (boolean) cartItem.get("isComboProduct") == isComboProduct) {
54             int currentCount = (int) cartItem.get("cartProductCount");
55             cartItem.put("cartProductCount", currentCount + cartProductCount);
56             alreadyIn = true;
57             break;
58         }
59     }

```

productComboName=, productSingleName=치즈볶이, productComboName=,



❓ 발생 원인

Controller에서 정상적으로 값을 뽑는 것처럼 보이지만
빈 값을 보냄

c:if 태그는 빈값이나 NULL 값에 대해 별도의 처리 로직
제공하지 않음

→ 위와 같은 값이 입력되었을 때 예외 처리가 없으면 오류 발생 가능

빈값 또는 null일 경우 c:if 조건이 거짓으로 평가되어
조건 내의 코드 실행되지 않음

❗ 해결 방안

View와 소통하여, c:choose 태그 및 c:when 태그로 수정
→ 여러 조건 효율적으로 관리

c:when 조건이 충족되지 않아도 c:otherwise를 통해
대체 텍스트 설정

예외적인 입력에 대한 처리를 수정한 c 태그를 통해 명확히 구분

❓ 발생 원인

페이지네이션 기능을 구현하기 위해 필요한 총 데이터의 수를 Controller에서 배열의 사이즈를 구하여 전송 시도

❗ 해결 방안

접근 방식 자체가 틀린 상황 발생

배열의 사이즈를 구하기 위해선 모든 정보를 불러 와야 가능하기 때문에 DB를 더욱 많이 사용

페이지 번호와 한 페이지에서 보여줄 데이터의 수를 전송하고, DB에서 데이터의 수만큼 출력하면서 총 데이터 수를 구하여 DTO에 저장
 → DTO에 저장된 수만 받아와 모든 데이터를 배열에 저장할 필요 없음

```

69     productSingleDTO.setCondition("SELECTALL_1");
70     productSingleDTO.setProductSingleCategory("PLUSPRODUCT");
71     productSingleDTO.setstartIndex(plusstartIndex);
72     productSingleDTO.setLimitNumber(plusPerPage);
73     ArrayList<ProductSingleDTO> cuPlusOneList = productSingleDAO.selectAll(productSingleDTO);
74     request.setAttribute("plusCUProductDatas", cuPlusOneList);
75
76     int totalPageNumber = cuPlusOneList.size();
77     request.setAttribute("CUTotalPageNumber", totalPageNumber);
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
  
```

06 향후 개발 과제 및 소감

향후 개발 과제

- 기능 개선
내 정보 페이지에서 정보 수정 시 간편 로그인한 SNS의 정보 수정 불가
- 사용자 기능 추가
카카오페이 결제 API, 문자 API, CKEditor 5
- 관리자 기능 추가
관리자 공지사항 작성, FAQ 기능
상품 추가, 수정, 삭제

고강희

좋았던 점

- DB에 컬럼을 추가하지 않고 DTO만 확장시켜 요구사항을 처리하고 무결성 규칙을 유지하는 방법을 학습

아쉬운 점

- 하나의 프로젝트를 끝낼 때마다 항상 설계의 미흡함에 아쉬움을 느낌
- 최종 프로젝트 때는 구글 캘린더, Slack 등 다양한 협업 도구를 사용해보고 싶음

장지현

좋았던 점

- 셀레니움에 대한 이해도가 향상됨
- 이번 프로젝트를 통해 처음보다 로그 분석에 능숙해졌다는 것을 체감

아쉬운 점

- SQL문에 대한 이해도 부족으로 시간이 예상보다 오래 걸려 아쉬움
- 충분히 설계를 했다고 생각했지만 부족했음

정규민

좋았던 점

- 화살표 함수와 jQuery에 익숙해짐
- 개발자 도구 및 로그를 통해 어떤 파트의 오류인지 확인 가능

아쉬운 점

- 로그를 충분히 작성했다고 생각했는데 통합할 때
특히 비동기 처리에서 파라미터 등을 확인하기가 어려울 때가 있었음
- 비동기 처리를 ajax에서 fetch+async/await으로 변경 적용할 예정

김동현

좋았던 점

- MVC 각 파트에 대해서 전체적으로 이해할 수 있었던 좋은 경험
- Front에서 보내주는 비동기 요청을 서블릿을 통해 처리하는 능력이 부족했으나 이를 완성해내며 극복함

아쉬운 점

- 설계 단계에서 변수명 등을 확실히 정했다고 생각했으나
기능이 추가되면서 각자 다른 변수명을 사용하여 데이터 전달에 오류가 많았음
- 서비스 설계 단계에서 한글 코딩이 부족하여 가독성과 효율성이 아쉬운 코드가 작성된 부분을 개선할 예정

Q & A

감사합니다

버그잡자 찍찍찍 