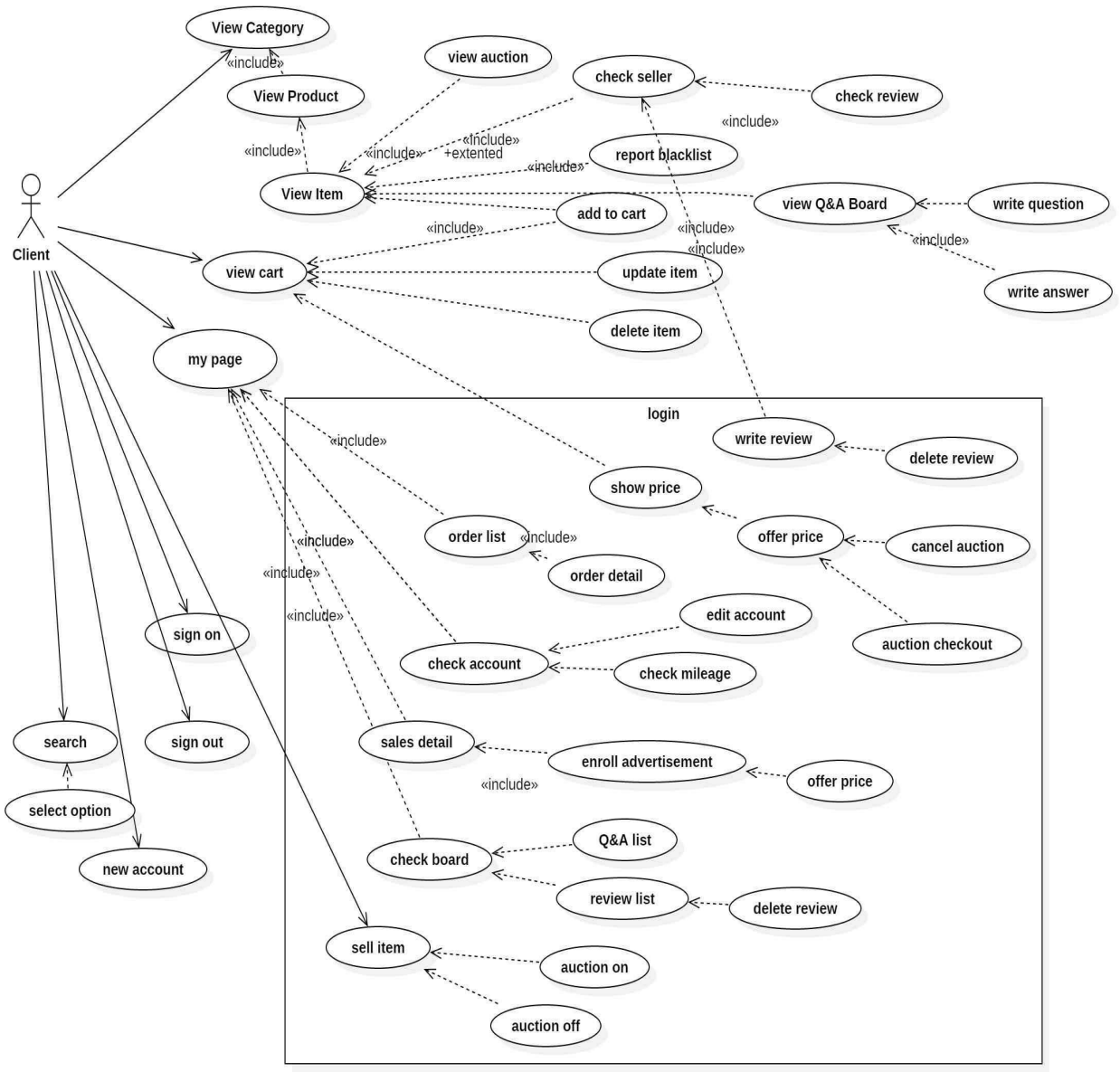
□ 문제(요구사항) 분석

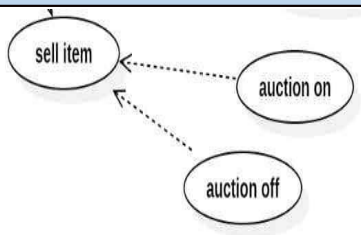
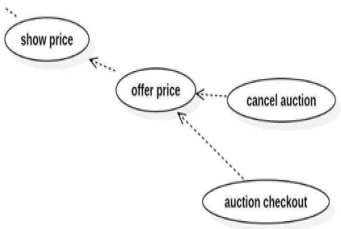
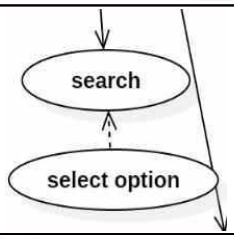
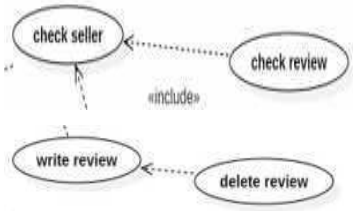
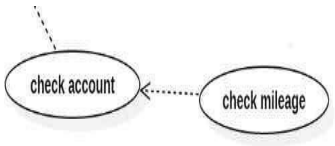
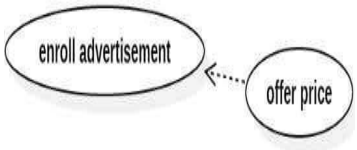
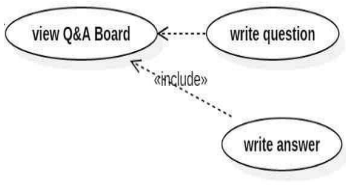
세부 내용	
사실 확인	**▶ 기존 시스템 기능** - 애완동물을 판매하는 웹(B2C). - 로그인과 회원가입 서비스가 구현되어 있음. - 사용자는 구매하고자 하는 물품을 장바구니에 담을 수 있지만, 로그아웃하면 내용물이 리셋됨. - 제품 검색 기능이 있긴 하나, 오직 물품의 name으로만 검색이 가능함. - 제품 주문이 가능하지만, 주문할 때는 반드시 로그인 상태여야 함. **▶ 요구 사항** 1. 기존 B2C 서비스로 운영되던 Pet Store에 C2C 서비스를 추가하길 원함. 2. 경매 방식으로 판매를 하고자 함. 3. 기타 제안 3-1. 고객 편의성 향상 3-2. 매출 증가
가설/해결안	**▶ <요구 사항 1>에 대한 해결안** **[기능1. 상품 등록]** 1-1. 일반 사용자, 판매자, 관리자를 위한 페이지를 구분함. 1-2. 원활한 거래를 위해 신뢰성을 확보해야하므로, 허위등록을 한 계정을 블랙리스트에 추가함(게시물의 신고 횟수가 일정 수를 넘어가면 해당 게시물을 올린 계정이 블랙리스트에 추가됨). 블랙리스트에 오른 계정은 판매 글을 올릴 수 없도록 제한함. **▶ <요구 사항 2>에 대한 해결안** **[기능2. 경매]** 2-1. 기간 내에 가장 높은 가격을 부른 구매자에게 판매함. 2-2. 가격 제시는 최고가 초과만 가능함. 2-3. 사용자들이 진행 상황을 한눈에 볼 수 있도록 그래프를 제시함. 2-4. 낙찰된 사람에게 알림이 가고 하루 안에 입금을 하지 않으면 다음 낙찰가를 부른 사람이 낙찰. 이 과정을 거래가 이루어질 때까지 반복함. **▶ <요구 사항 3>에 대한 해결안** **[기능3. 상품 검색]** 3. 사용자가 원하는 상품에 빠르게 접근할 수 있도록 검색 옵션 부여 기능 도입. 가격 범위 지정, 제목으로 검색, 판매자로 검색, 전체 검색 등을 선택하여 검색 가능. **[기능4. 판매자에 대한 후기]** 4. 판매자에 대한 신뢰성 재고를 할 수 있도록, 후기 공유 기능 제공.

	[기능5. 마일리지 제도] 5. 마일리지 제도를 추가. (다만 회사 수익을 위해 적절한 양의 마일리지를 부여). **[기능6. 메인 광고]** 6. 제품 구매를 유도를 위해 메인 화면에 광고란을 생성. (일반 사용자도 이용 가능하도록 하여 이익 창출) **[기능7. Q&A]** 7. 고객의 판단을 돕기 위해 필요. 제품에 대한 질문과 그에 대한 판매자의 답변으로 이루어진 기능.

□ 세부 기능 정의

▶ Use Case Diagram

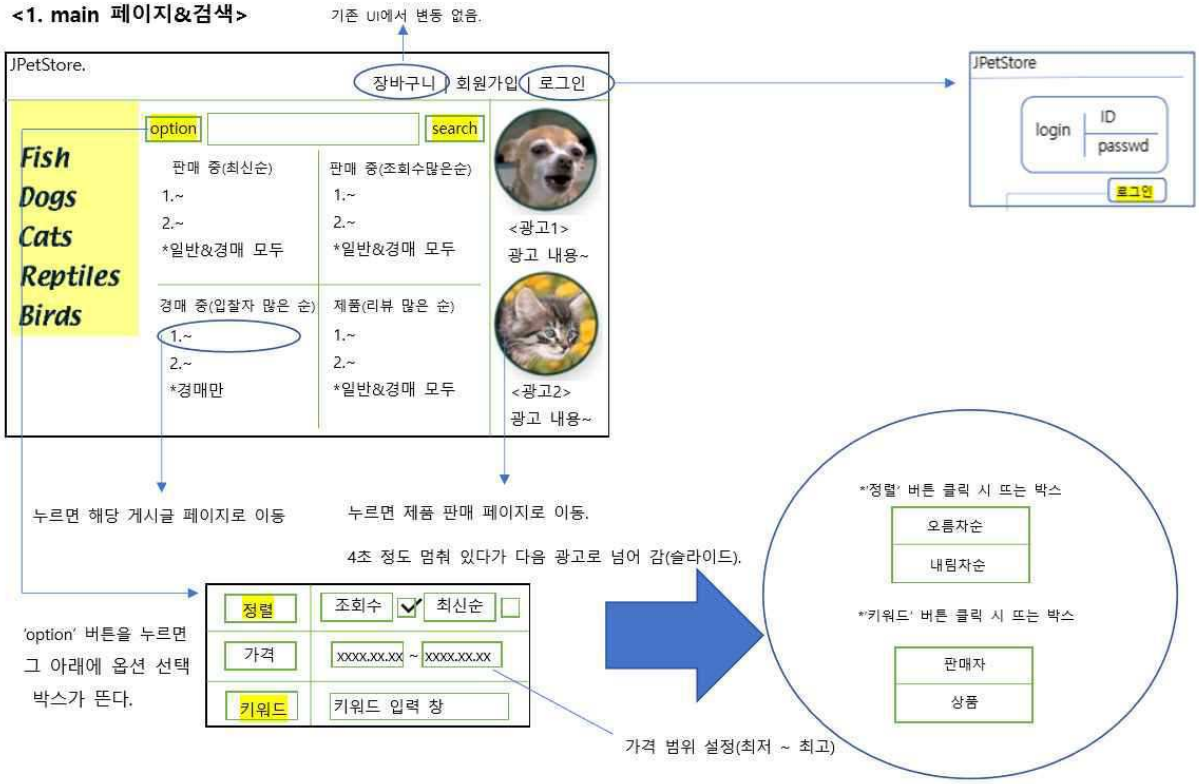


세부 내용		
기능 소개		**▶ 기능1. 상품 등록** my page에서 회원만 상품 등록 가능. 일반 판매/경매 판매 구분하여 등록함.
		**▶ 기능2. 경매** 사고자 하는 제품의 가격을 제시함. (가격 제시는 최고가 초과만 가능), 경매는 낙찰되거나 취소될 수 있음. 사용자 편의성 증대를 위해 경매 상황을 그래프를 통해 시각적으로 제시함.
		**▶ 기능3. 상품 검색** 회원/비회원 모두 상품 검색 가능. 메인 화면의 리스트는 상품을 등록한 순으로 나열되어 있으며 가격 범위, 제목, 판매자 검색 옵션을 부여하여 사용자의 편의성을 증대시킴.
		**▶ 기능4. 판매자에 대한 후기** 판매자 신뢰도를 알 수 있도록 후기 기능 추가. 판매 페이지에서 판매자에 대한 리뷰를 볼 수 있음. 리뷰 쓰기와 삭제, 수정은 로그인이 반드시 필요.
		**▶ 기능5. 마일리지 제도** My page에서 마일리지 확인 가능. 판매/구매 활동 시 일정 비율의 마일리지가 적립되고 구매 시 마일리지 결제가 가능함. 마일리지는 회원 간 양도가 가능함.
		**▶ 기능6. 메인 광고** <my page> 카테고리에 광고 등록이 가능한 페이지를 생성하여 판매자가 일정 비용을 지불한 후 자신의 제품을 홍보할 수 있음. 관리자의 승인을 거쳐 하루에 등록 되어지는 광고의 개수는 총 5개이며 리스트의 형식으로 메인화면 오른편에 배너 형식으로 생성됨.
		**▶ 기능7. Q&A 기능** 고객 판단을 돕기 위한 기능. 판매 페이지에서 물품에 대한 Q&A를 확인할 수 있다. Question은 쓰기, 수정, 삭제는 로그인이 필요하며 권한을 가진 사용자만 수행할 수 있다. Answer은 해당 아이템 판매자만 쓰기, 수정, 삭제할 수 있음.

□ 구성 요소 설계

▶ UI

<1. main 페이지&검색>



<2. 제품 리스트가 뜨는 창>

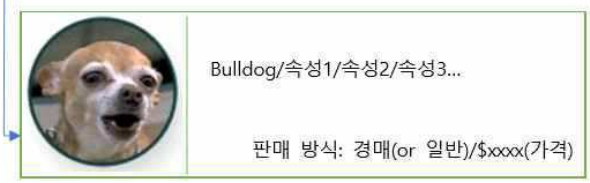
- 검색하면 뜨는 제품 리스트 & 카테고리 별로 제품 볼 때 뜨는 리스트

*** 현재 화면**

Koi

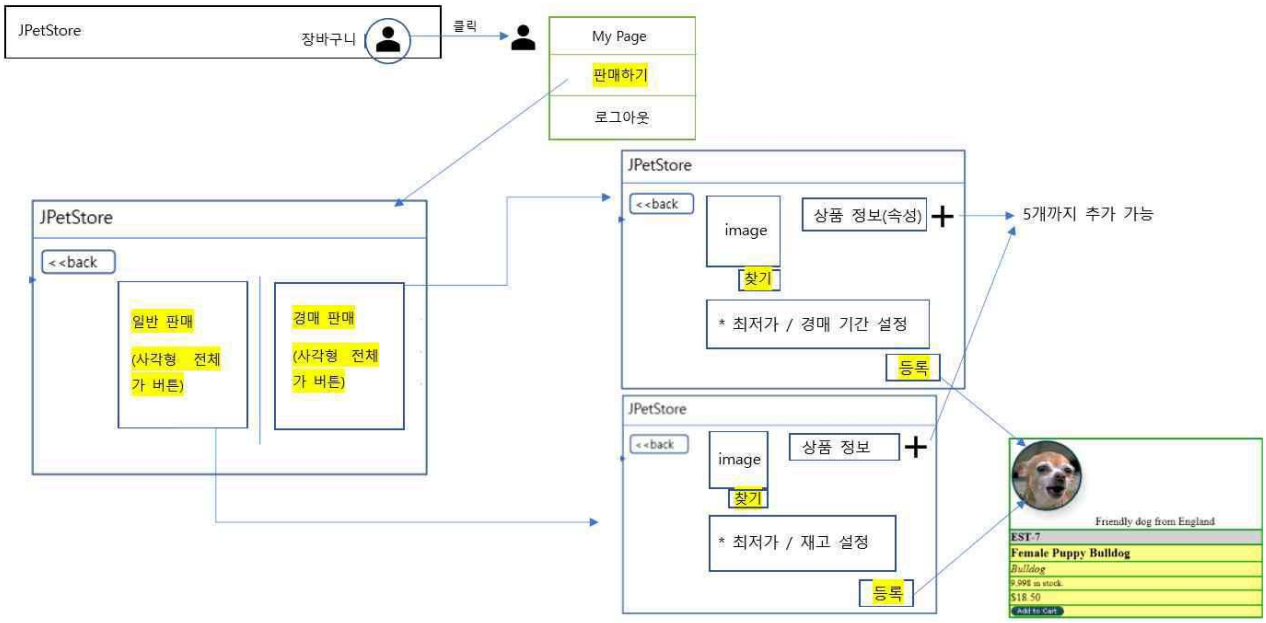
Item ID	Product ID	Description	List Price	
EST-4	FI-FW-01	Spotted Koi	$18.50	Add to Cart
EST-5	FI-FW-01	Spotless Koi	$18.50	Add to Cart

*** 수정(사용자들에게 친숙한 UI로)**

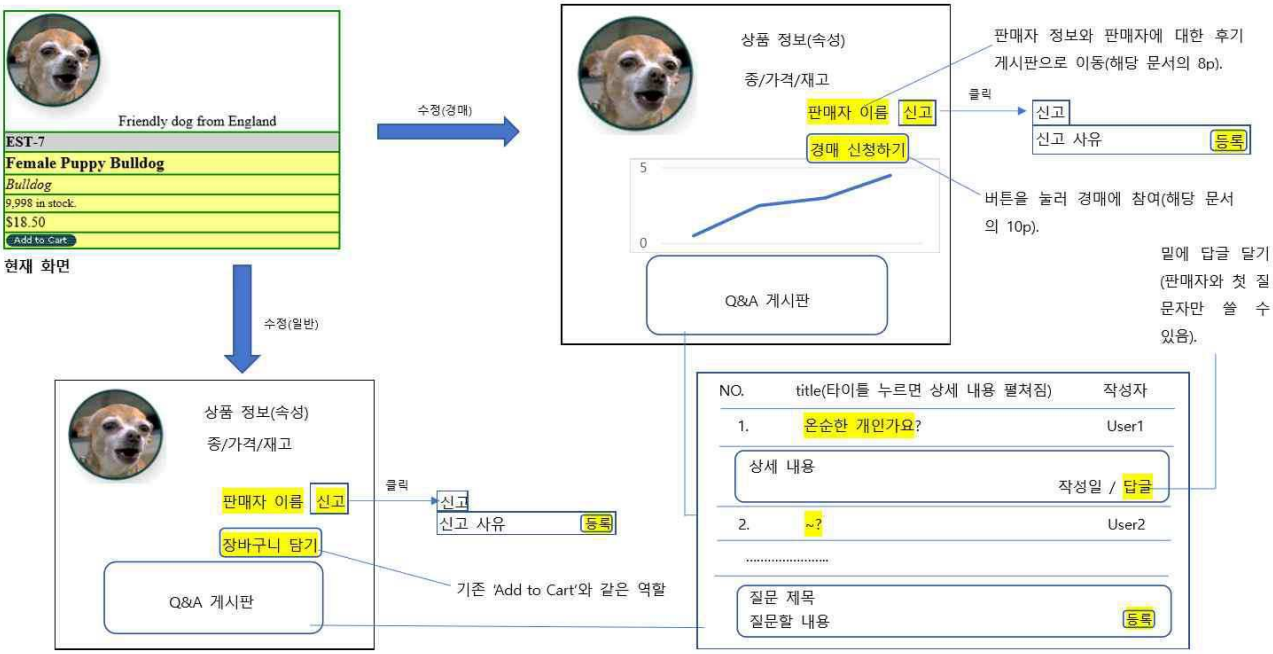


<3. 판매 등록>

Main 페이지 로그인 후 우측 상단

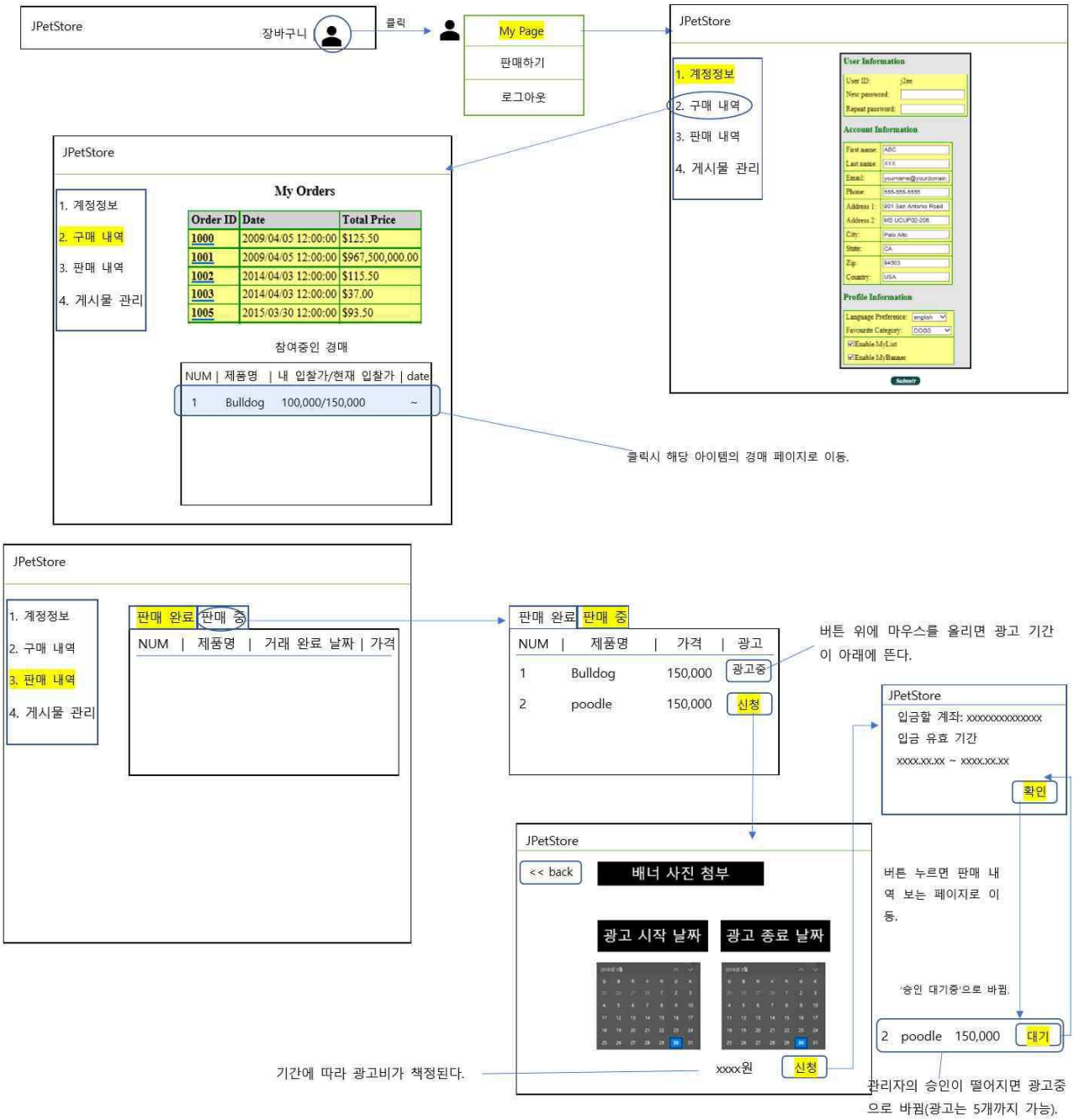


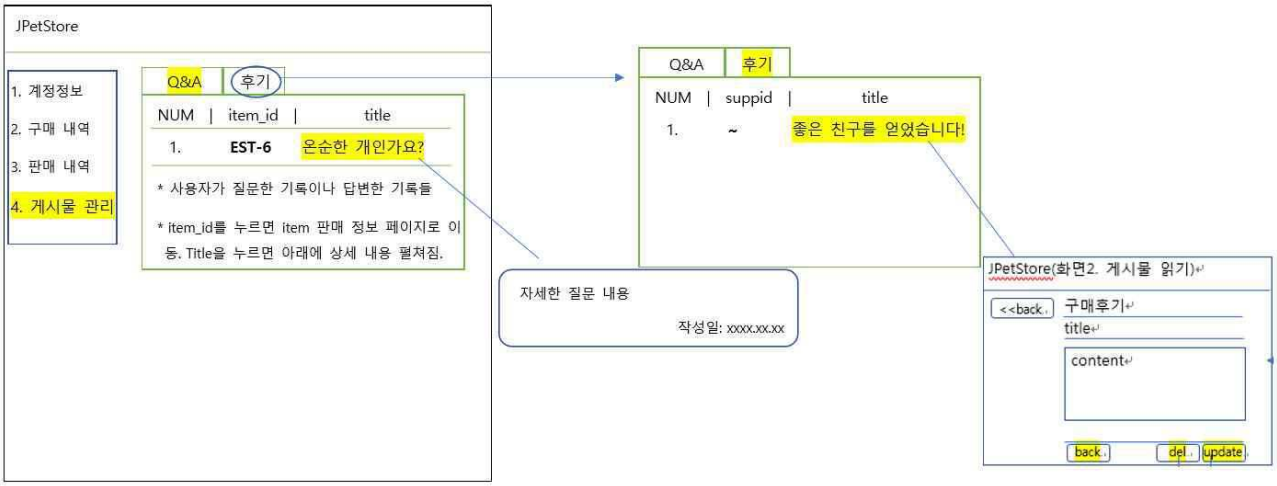
<4. 판매 중인 아이템 상세 보기>



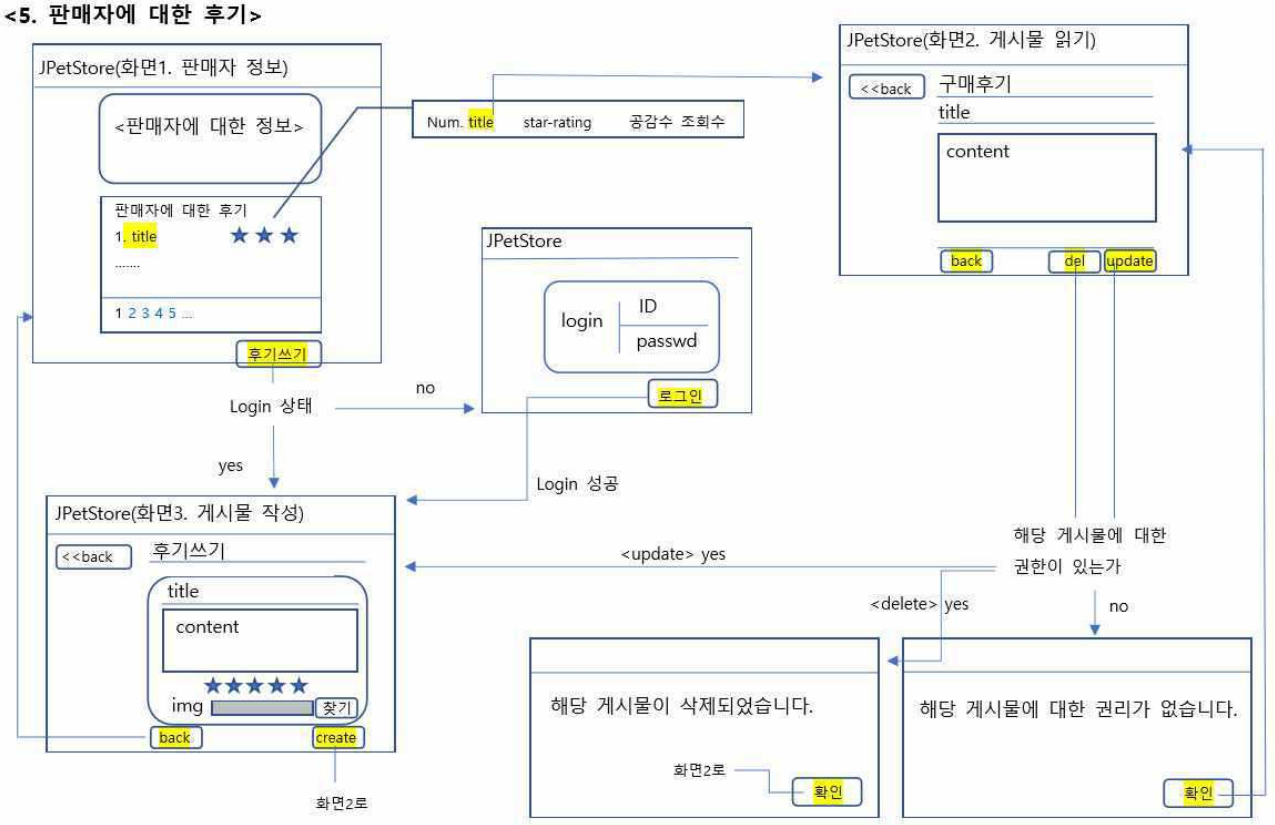
<5. My Page>

로그인 후 메인 우측 상단

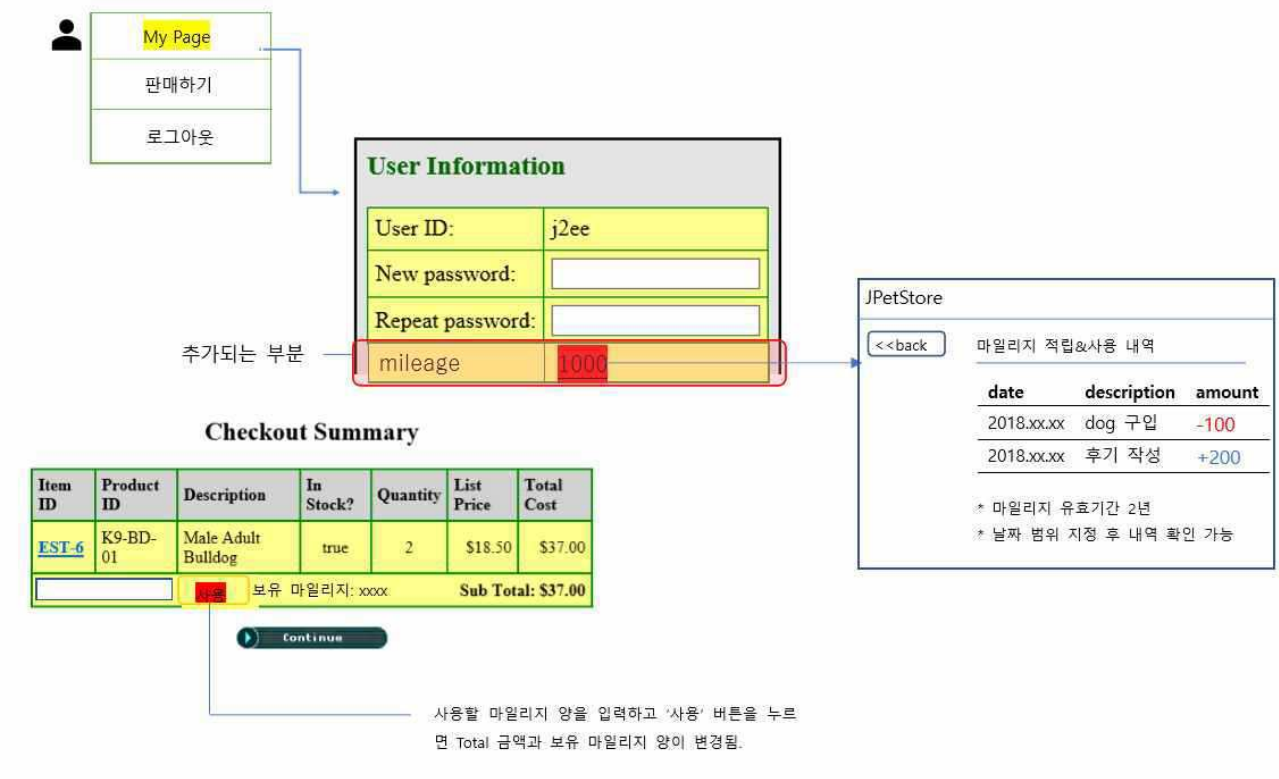



JPetStore
1. 계정정보
2. 구매 내역
3. 판매 내역
4. 게시물 관리
Q&A
후기
NUM | item_id | title
1. EST-6 온순한 개인가요?
* 사용자가 질문한 기록이나 답변한 기록들
* item_id를 누르면 item 판매 정보 페이지로 이동. Title을 누르면 아래에 상세 내용 펼쳐짐.
Q&A
후기
NUM | suppid | title
1. ~ 좋은 친구를 얻었습니다!
자세한 질문 내용
작성일: xxxx.xx.xx
JPetStore(화면2. 게시물 읽기)
<<back
구매후기
title
content
back
del
update

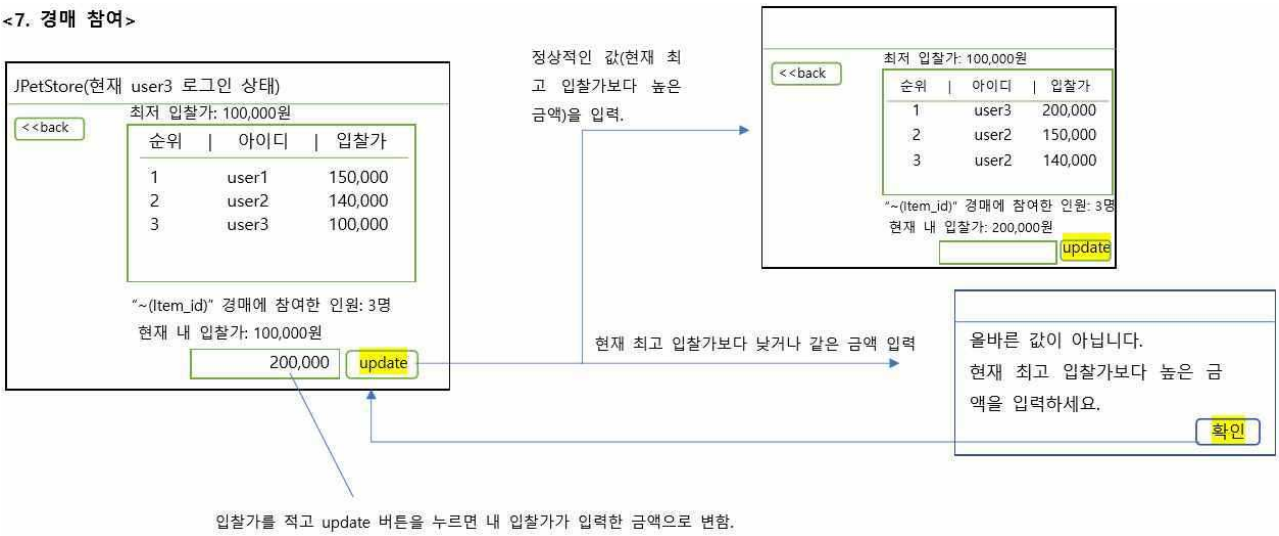
<5. 판매자에 대한 후기>


JPetStore(화면1. 판매자 정보)
<판매자에 대한 정보>
판매자에 대한 후기
1. title ★★★
.....
1 2 3 4 5 ...
후기쓰기
Num. title star-rating 공감수 조회수
JPetStore(화면2. 게시물 읽기)
<<back
구매후기
title
content
back
del
update
JPetStore
login
ID
passwd
로그인
no
Login 상태
yes
Login 성공
JPetStore(화면3. 게시물 작성)
<<back
후기쓰기
title
content
★★★★★
img
찾기
back
create
화면2로
<update> yes
해당 게시물에 대한
권한이 있는가
<delete> yes
no
해당 게시물이 삭제되었습니다.
화면2로
확인
해당 게시물에 대한 권리가 없습니다.
확인

<6. 마일리지>



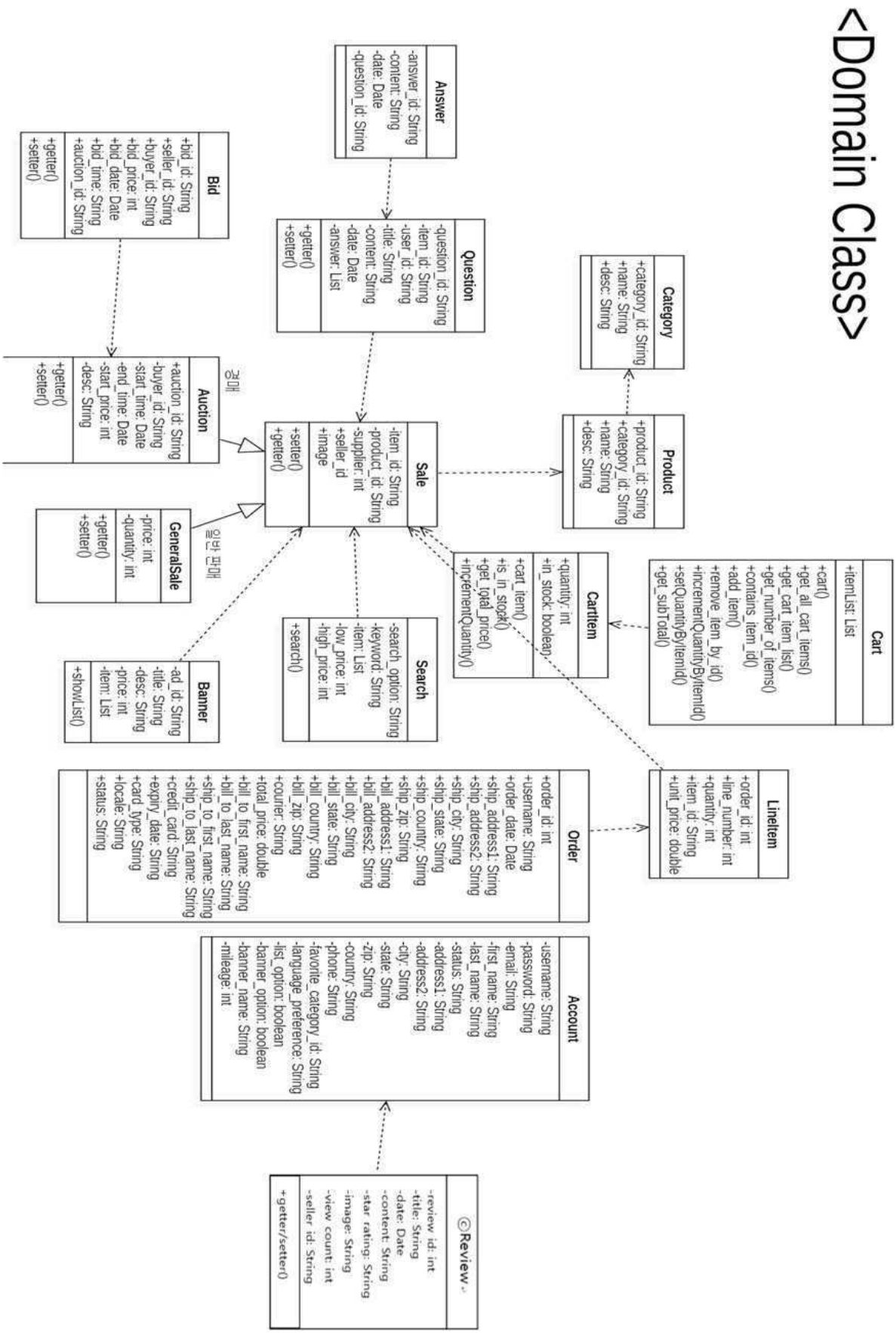
<7. 경매 참여>

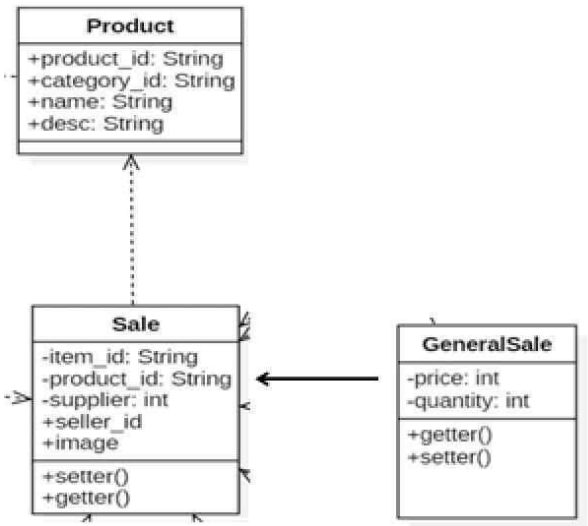
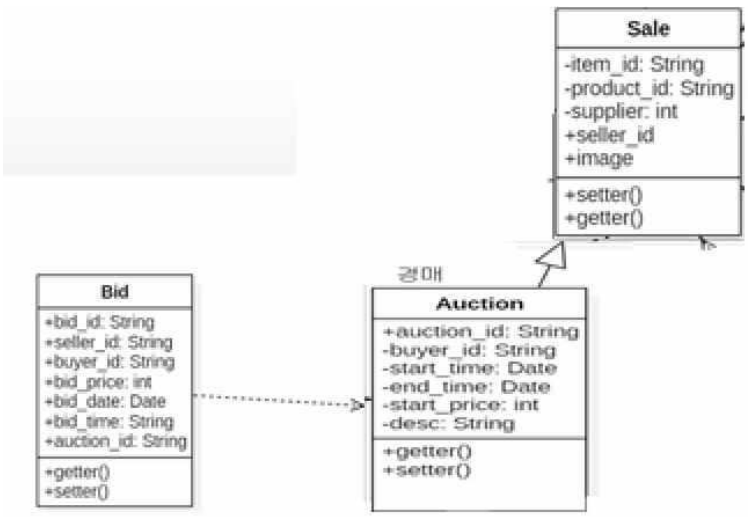
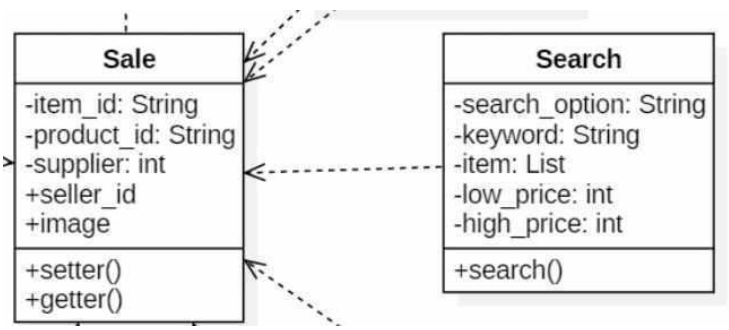


*보증금 시스템 추가할지 추후 논의.

*낙찰 알림 시스템 어떻게 구현할지 논의

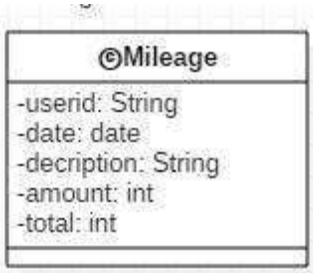
▶ Domain Class



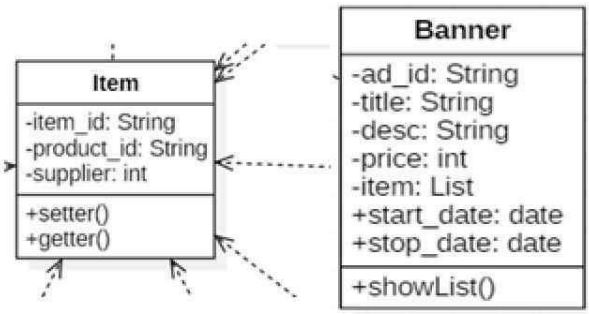
세부 내용	
기능별 Domain class	**▶ 기능1. 상품 등록**  **▶ 기능2. 경매**  **▶ 기능3. 검색** 

▶ 기능4. 판매자에 대한 후기

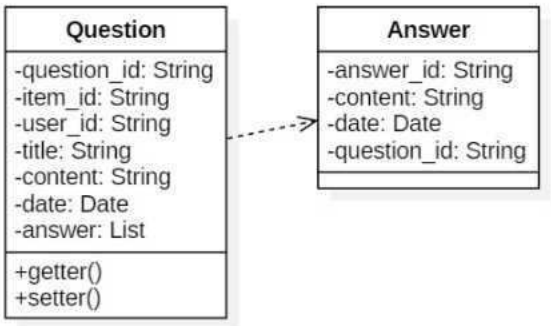
▶ 기능5. 마일리지 제도



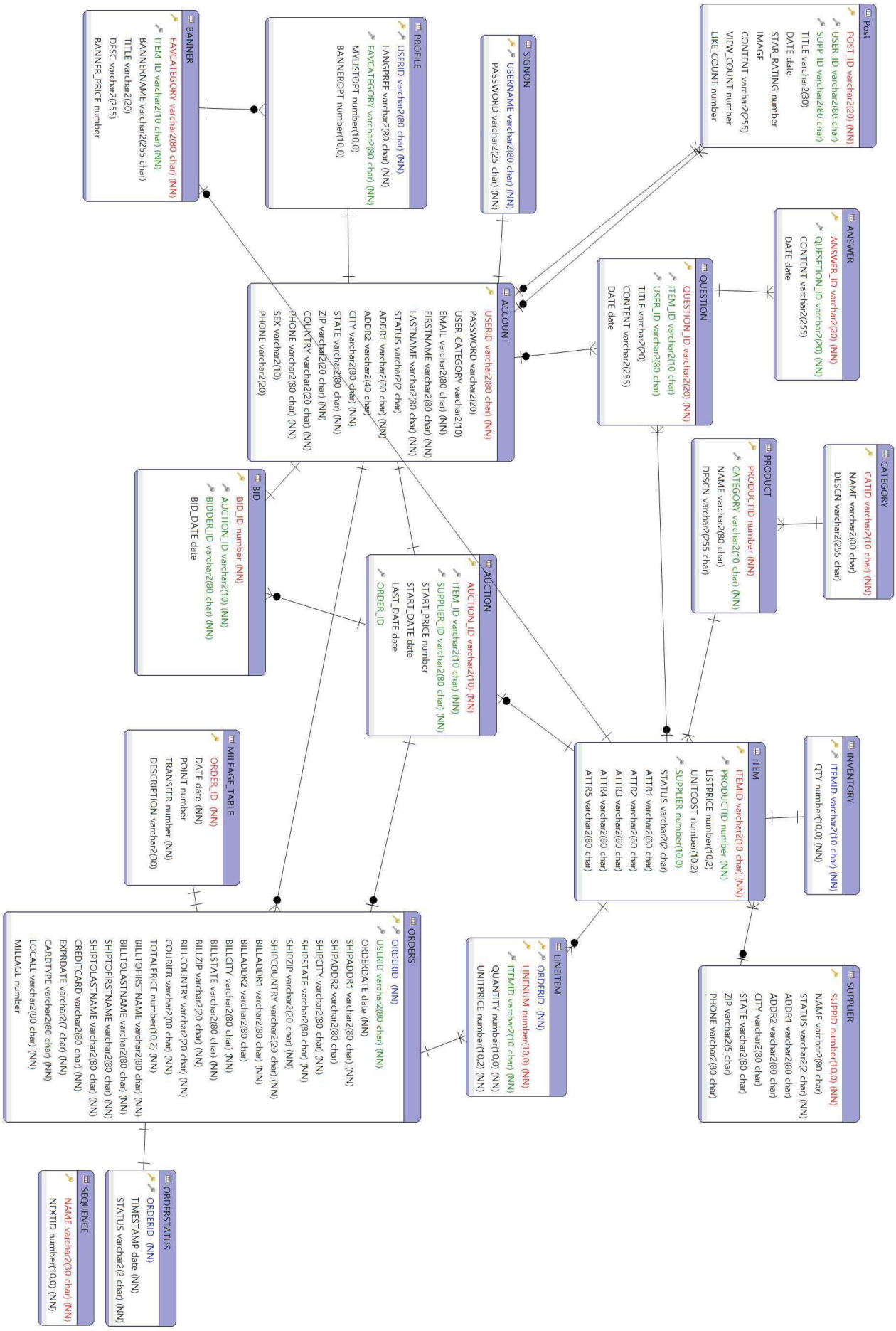
▶ 기능6. 메인 광고

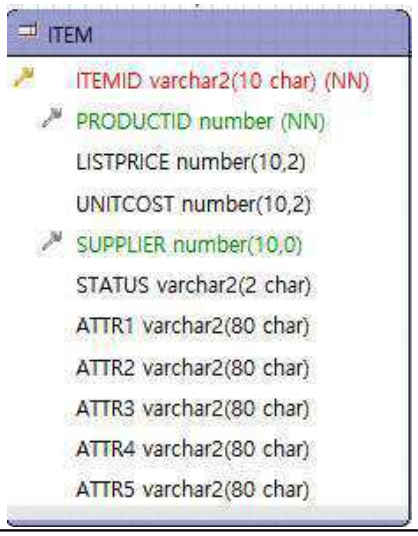
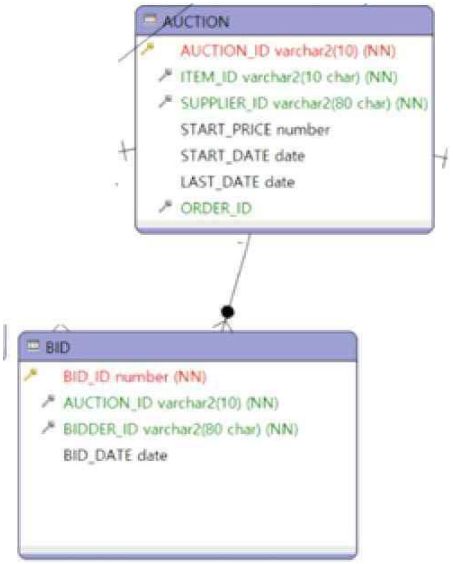


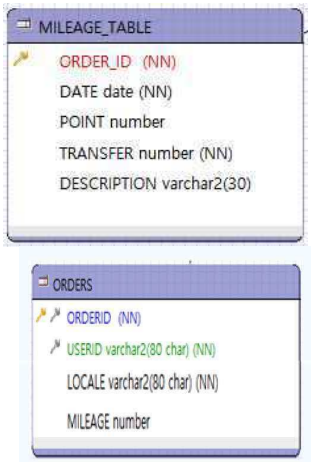
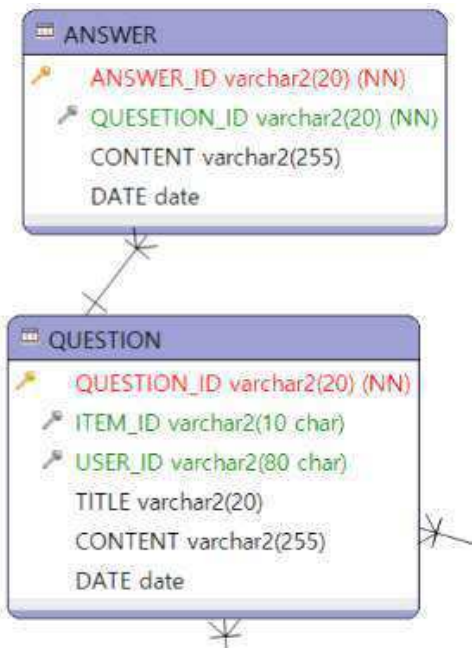
▶ 기능7. Q&A



▶ DB Schema


Post
POST_ID varchar2(20) (NN)
USER_ID varchar2(80 char)
SUPP_ID varchar2(80 char)
TITLE varchar2(30)
DATE date
STAR_RATING number
IMAGE
CONTENT varchar2(255)
VIEW_COUNT number
LIKE_COUNT number
ANSWER
ANSWER_ID varchar2(20) (NN)
QUESETION_ID varchar2(20) (NN)
CONTENT varchar2(255)
DATE date
QUESTION
QUESTION_ID varchar2(20) (NN)
ITEM_ID varchar2(10 char)
USER_ID varchar2(80 char)
TITLE varchar2(20)
CONTENT varchar2(255)
DATE date
SIGNON
USERNAME varchar2(80 char) (NN)
PASSWORD varchar2(25 char) (NN)
ACCOUNT
USERID varchar2(80 char) (NN)
PASSWORD varchar2(20)
USER_CATEGORY varchar2(10)
EMAIL varchar2(80 char) (NN)
FIRSTNAME varchar2(80 char) (NN)
LASTNAME varchar2(80 char) (NN)
STATUS varchar2(2 char)
ADDR1 varchar2(80 char) (NN)
ADDR2 varchar2(40 char)
CITY varchar2(80 char) (NN)
STATE varchar2(80 char) (NN)
ZIP varchar2(20 char) (NN)
COUNTRY varchar2(20 char) (NN)
PHONE varchar2(80 char) (NN)
SEX varchar2(10)
PHONE varchar2(20)
PROFILE
USERID varchar2(80 char) (NN)
LANGPREF varchar2(80 char) (NN)
FAVCATEGORY varchar2(80 char) (NN)
MYLISTOPT number(10,0)
BANNEROPT number(10,0)
BANNER
FAVCATEGORY varchar2(80 char) (NN)
ITEM_ID varchar2(10 char) (NN)
BANNERNAME varchar2(255 char)
TITLE varchar2(20)
DESC varchar2(255)
BANNER_PRICE number
CATEGORY
CATID varchar2(10 char) (NN)
NAME varchar2(80 char)
DESCN varchar2(255 char)
PRODUCT
PRODUCTID number (NN)
CATEGORY varchar2(10 char) (NN)
NAME varchar2(80 char)
DESCN varchar2(255 char)
ITEM
ITEMID varchar2(10 char) (NN)
PRODUCTID number (NN)
LISTPRICE number(10,2)
UNITCOST number(10,2)
SUPPLIER number(10,0)
STATUS varchar2(2 char)
ATTR1 varchar2(80 char)
ATTR2 varchar2(80 char)
ATTR3 varchar2(80 char)
ATTR4 varchar2(80 char)
ATTR5 varchar2(80 char)
INVENTORY
ITEMID varchar2(10 char) (NN)
QTY number(10,0) (NN)
SUPPLIER
SUPPID number(10,0) (NN)
NAME varchar2(80 char)
STATUS varchar2(2 char) (NN)
ADDR1 varchar2(80 char)
ADDR2 varchar2(80 char)
CITY varchar2(80 char)
STATE varchar2(80 char)
ZIP varchar2(5 char)
PHONE varchar2(80 char)
BID
BID_ID number (NN)
AUCTION_ID varchar2(10) (NN)
BIDDER_ID varchar2(80 char) (NN)
BID_DATE date
AUCTION
AUCTION_ID varchar2(10) (NN)
ITEM_ID varchar2(10 char) (NN)
SUPPLIER_ID varchar2(80 char) (NN)
START_PRICE number
START_DATE date
LAST_DATE date
ORDER_ID
LINEITEM
ORDERID (NN)
LINENUM number(10,0) (NN)
ITEMID varchar2(10 char) (NN)
QUANTITY number(10,0) (NN)
UNITPRICE number(10,2) (NN)
MILEAGE_TABLE
ORDER_ID (NN)
DATE date (NN)
POINT number
TRANSFER number (NN)
DESCRIPTION varchar2(30)
ORDERS
ORDERID (NN)
USERID varchar2(80 char) (NN)
ORDERDATE date (NN)
SHIPADDR1 varchar2(80 char) (NN)
SHIPADDR2 varchar2(80 char)
SHIPCITY varchar2(80 char) (NN)
SHIPSTATE varchar2(80 char) (NN)
SHIPZIP varchar2(20 char) (NN)
SHIPCOUNTRY varchar2(20 char) (NN)
BILLADDR1 varchar2(80 char) (NN)
BILLADDR2 varchar2(80 char)
BILLCITY varchar2(80 char) (NN)
BILLSTATE varchar2(80 char) (NN)
BILLZIP varchar2(20 char) (NN)
BILLCOUNTRY varchar2(20 char) (NN)
COURIER varchar2(80 char) (NN)
TOTALPRICE number(10,2) (NN)
BILLTOFIRSTNAME varchar2(80 char) (NN)
BILLTOLASTNAME varchar2(80 char) (NN)
SHIPTOFIRSTNAME varchar2(80 char) (NN)
SHIPTOLASTNAME varchar2(80 char) (NN)
CREDITCARD varchar2(80 char) (NN)
EXPRDATE varchar2(7 char) (NN)
CARDTYPE varchar2(80 char) (NN)
LOCALE varchar2(80 char) (NN)
MILEAGE number
ORDERSTATUS
ORDERID (NN)
TIMESTAMP date (NN)
STATUS varchar2(2 char) (NN)
SEQUENCE
NAME varchar2(30 char) (NN)
NEXTID number(10,0) (NN)

세부 내용		
DB 설명	**▶ 기능1. 상품 등록** 	- **ITEMID** : 아이템이 등록될 때 마다 자동생성 - **PRODUCTID** : 아이템등록 시 먼저 제품에 등록되고 그 제품 아이디를 아이템 내역에 참조 - **LISTPRICE :** 정가 (전체 개수 * 단가) - **UNITPRICE** : 단가(하나당 가격) - **SUPPLIER** : 판매인 ID를 참조 - **STATUS :** 상품에 대한 상태 - **ATTR1~5** : 상품의 특이점을 입력(ex> 사진)
	▶ 기능2. 경매 	**[AUCTION : 경매]** - **AUCTION_ID** : 경매의 ID - **ITEM_ID** : 제품의 ID - **SUPPLIER_ID** : 판매자의 ID - **START_PRICE** : 경매의 시작 가격 - **START_DATE** : 경매의 시작 날짜 - **LAST_DATE** : 경매의 마감 날짜 - **ORDER_ID** : 마지막 낙찰자의 정보와 결재 정보 **[BID : 입찰]** - **BID_ID** : 입찰의 ID - **AUCTION_ID** : 경매 ID - **BIDDER_ID** : 입찰자 ID - **BID_DATE** : 입찰 날짜
	▶ 기능4. 판매자에 대한 후기 	- **POST_ID** : 해당 테이블의 기본키. - **USER_ID :** Account 테이블의 기본키를 참조. 어떤 사용자가 올린 글인지 구분할 수 있음. - **SUPP_ID :** Account 테이블의 기본키를 참조. 어떤 판매자에 대한 후기인지 알 수 있음. - **TITLE :** 글의 제목 - **DATE :** 글이 게시된 날짜. 혹은 최근 수정된 날짜. - **STAR_RATING:** 판매자에 대한 별점. - **IMAGE:** 판매 물품 사진.

		- **CONTENT:** 게시글의 내용 - **VIEW_COUNT:** 해당 글의 조회수. 레코드 생성시 처음에는 모두 값이 0임.
	▶ **기능5. 마일리지 제도** 	- **ORDERID :** 각 거래에 의해 마일리지 부여 (거래 ID를 PK로 사용) - **DATE :** 거래 날짜 - **POINT :** 적립된 금액 - **TRANSFER :** 양도여부(0:NO, 1:YES) - **DESCRIPTION :** 마일리지에 대한 설명 **ORDERS 테이블에도 MILEAGE property존재**
	▶ **기능6. 메인 광고** 	- **FAVCATEGORY**: 사용자가 선호하는 제품 카테고리() - **ITEM_ID**: 제품 ID (ITEM 테이블의 PK 참조) - **BANNERNAME**: 광고 리스트(배너) 이름 - **TITLE**: 광고 제목 - **DESC**: 광고 내용 소개 - **BANNER_PRICE**: 광고 가격
	▶ **기능7. Q&A** 	**[ANSWER** : 질문에 대한 답변] - **ANSWER_ID**: 해당 테이블의 기본키. - **QUESTION_ID**: 어떤 질문에 대한 답변인지를 구분(QUESTION 테이블의 PK 참조). - **USER_ID**: Account 테이블의 기본키를 참조. - **CONTENT**: 질문 내용. - **DATE**: 글을 올린 날짜. **[QUESTION** : 제품에 대한 질문] - **QUESTION_ID**: 해당 테이블의 기본키. - **ITEM_ID**: 제품 ID (ITEM 테이블의 PK 참조) - **USER_ID**: Account 테이블의 기본키를 참조. 어떤 사용자가 올린 글인지 구분할 수 있음. - **TITLE**: 질문 제목. - **CONTENT**: 질문 내용. - **DATE**: 글을 올린 날짜.

▶ **DAO**

세부 내용	
DAO	**▶ 기능1. 상품 등록** **[CategoryDAO] - 관리자만 다룰 수 있음** public void insert_cat(Category c) : (고객이 선택할 수 있는) 상품 카테고리 등록 public List<Category> show_cat() : DetailSellInsert Fom에서 선택해야하는 판매 cat리스트를 보여줌. public void update_cat(String a, String b) : a라는 카테고리를 b로 변경. **[ProductDAO] - 상품 등록 시 제품 먼저 등록되고 item이 등록됨** public void insert_product(Product d) : 상품등록 전 제품을 먼저 등록. public void update_product(List<String> a, List<String> b) : 제품 수정 (a의 속성들을 b로 바꾼다) public void delete_product(int product_id) : 제품 삭제 public List<Product> show_product() : 제품 보여주기 **[ItemDAO]** public void insert_item(GeneralSale sale) : 상품 등록 public int delete_item(int item_id) : 상품 삭제 public void update_item(List<String> a, List<String> b) : 상품 변경 public Item show_detail(String item_id) : 상품 상세정보 보여주기
	▶ 기능2. 경매 **[AuctionDAO]** public void insertAuction(Auction auction) : 경매 아이템 등록 public void updateAuction(Auction auction) : 경매 아이템 수정 public void deleteAuction(Auction auction) : 경매 아이템 삭제 public void beAuctionedOff(Auction auction, bid_id) : 낙찰

▶ **기능3. 상품 검색**

[searchDAO]

public List<Item> sortList (String option, String keyword)

public List<Item> sortList (int low_price, int high_price)

: 검색 옵션 값(판매자, 이름, 가격 범위)과 검색 값을 매개변수로 받아 옵션에 맞게 정렬한 뒤 리스트로 반환.

▶ **기능4. 판매자에 대한 후기**

[postDao]

public void createPost(Post post)

: 후기 생성

public void deletePost(int post_id)

: 후기 삭제

public void updatePost(Post post)

: 후기 변경

public Post detailPost(int post_id)

: 후기 내용 보여주기

public List<Post> getPostList()

: 후기 리스트 가져오기

▶ **기능5. 마일리지 제도**

[mileageDAO]

public void insert_mileage_history(String user_id, int mileage)

: 해당 계정의 마일리지 내역 보여주기.

public int display_mileage(String user_id)

: 현재 마일리지 보여주기.

public void update_mileage(String user_id, int mileage)

: 마일리지 변경하기.

▶ **기능6. 메인 광고**

[bannerDAO]

public void insertBannner(Banner banner)

: 최근 승인되어진 광고를 배너에 등록.

public void deleteBanner(Banner banner)

: 기간이 만료된 광고를 배너에서 삭제.

public void updateBanner(Banner banner)

: 등록, 삭제에 따라 리스트를 재정렬.

▶ **기능7. Q&A**

[QuestionDAO]

public void createQuestion(Question q)

	: 질문을 생성. public void deleteQuestion(int q_id) : 질문을 삭제. public void updateQuestion(Question q) : 질문을 수정 public List<Question> getQuestionList() : 질문 list 가져오기(Q&A 게시판) **[AnswerDAO]** public void createAnswer(Answer a) : 답변을 생성. public void deleteAnswer(int a_id) : 답변을 삭제. public void updateAnswer(Answer a) : 답변을 수정 public List<Answer> getAnswerList() : 답변 list 가져오기(Q&A 게시판)

▶ Controller

세부 내용	

Controller 설명	**▶ 기능1. 상품 등록 [ProductController]**
	
	@RequestMapping("sell.do")
	Public ModelAndView sellCat(){
	return "redirect:sellCatForm"
	}
	// URL주소로 '/shop/Item/sell.do'라는 명령이 들어오면 우선 판매카테고리(일반/경매) form으로 보낸다.
	
	@RequestMapping(value="sellCatClicked.do",method="RequestMethod.GET")
	public String sellCatSubmitted(@RequestParam("cat") int num){
	if(num==0){ return "redirect:GeneralSellForm" }
	return "redirect:AuctionSellForm"
	}
	// 앞서 보여준 두 개의 창 중 하나를 클릭하면 'sellCatClicked' 라는 url이 전송되고,
	//일반판매는 "0" 경매 판매는 "1"이라는 숫자도 같이 넘어온다.
	//넘어온 인수에 따라 각각 상품을 등록할 수 있는 form을 넘겨준다.
	
	@RequestMapping(value="sellSubmit.do",method="RequestMethod.POST")
	public ModelAndView sellDetailSubmit(@ModelAttribute Item item){

```
            boardService.itemCreate(item);
            ModelAndView mav = new ModelAndView("itemDetail");
            mav.addObject("newItem",item)j;
            return mav
        }

//item판매 작성을 완성하고 완료 버튼을 click하면 해당 폼에 작성된 내역이 mav형태로 넘어오고
//해당 mav를 item서비스에 존재하는 itemCreate() 메소드를 실행한다.
//그 후 등록된 판매디테일 내역을 돌려주어 띄어준다.


@RequestMapping("listShow.do")
   Public ModelAndView   showItemList(@RequestParam("user_id") int num){
            ModelAndView mav = new ModelAndView("itemList");
            mav.addObject("ItemList",boardService.showItemList(num));
            return mav
            }
//URL "/shop/Item/listShow.do"가 클릭됐을 때  본인의 판매내역을 띄어준다.

@RequestMapping("deleteItem.do")
   Public String   deleteItem(@RequestParam("item_id") int num){
            boardService.deleteItem(num);
            return "redirect:listShow.do"
}
// URL "/shop/Item/deleteItem.do"가 넘어올 땐 삭제하고자 하는 판매의 아이디도
//같이 넘어온다. 아이템 서비스에 deleteItem()이라는 메소드에 아이디를 변수로 주어
//해당 판매내역을 삭제하도록 한다. 그 후 판매내역을 갱신시켜 보여준다.


@RequestMapping("updateItemShow.do")
   Public String sellCat(@RequestParam("updateItem") Item item){
            boardService.itemUpdate(item);
            return "redirect:listShow.do"
}
// URL "/shop/Item/update.do"가 넘어올 땐 삭제하고자 하는 판매의 아이디도
//같이 넘어온다. 아이템 서비스에 itemUpdate()이라는 메소드에 Item Object를 변수로 주어
//해당 판매내역을 수정한다. 그 후 판매내역을 갱신시켜 보여준다.
```

▶ 기능2. 경매 [AuctionController]

```
@RequestMapping("list.do")
```

```
  Public ModelAndView list(@RequestParam("auction_id") String auction_id) {
 ModelAndView mav = new ModelAndView("bidList");
          Mav.addObject("bidList",boardService.getBidList(auction_id));
          Return mav;
}
//URL /shop/auction/list.do가 클릭되면 해당 acution_id의 경매에 대한 입찰가격들의 리스트를 DB에 검색하고 View에 보여준다.

@RequestMapping(value="createBid.do", method=RequestMethod.POST)
  Public String create(@ModelAttribute("bid") Bid bid) {
          boardService.createBid(bid);
          return "redirect:confirmAuction";
}
// URL shop/auction/createBid.do를 클릭하면 가격과 입찰자 정보가 입찰 테이블에 추가된다.

@RequestMapping("deleteBid.do")
  Public String deleteBid(@RequestParam int num) {
          boardService.deleteBId(num);
          return "redirect:list.do"
}
// URL shop/auction/deleteBid.do가 클릭되면 입찰을 삭제한다.
```

▶ 기능3. 상품 검색 [SearchItemController]

```
@RequestMapping(value="searchSubmit.do",method="RequestMethod.POST")
  Public String searchItemSubmit(@RequestParam("option") String option,
                                 @RequestParam("keyword") String keyword){
          boardService.getList(option, keyword);
           ...
          return "redirect:ItemList.do"
        }

//검색창에 검색옵션과 검색 키워드을 선택하고 검색 버튼을 click하면 작성된 검색 옵션과 키워드가 같이 넘어옴.
//서비스에 있는 getList()이라는 메소드에 옵션과 키워드를 매개변수로 주어 그에 맞게 아이템 리스트를 정렬한다.
//재정렬 된 리스트를 갱신시켜 보여준다.
```

▶ 기능4. 판매자에 대한 후기 [BoardController]

```
@RequestMapping("list.do")
  Public ModelAndVIew list(@RequestParam("username") String username) {
          ModelAndView mav = new ModelAndView("list");
```

```
            Mav.addObject("postList", boardService.list(username));
            Return mav;
}
// 매핑 주소 - /shop/board/list.do
// 매개변수로 username(판매자 아이디)을 받고 있으며, 이것으로 DB에 접근해서 쿼리를 수행. 해당 아이디에 대한 후기 정보만 뽑아온다. 그 뒤에 postList에 데이터를 저장.
// list.jsp로 postList를 전달.

@RequestMapping(value="create.do", method=RequestMethod.GET)
  Public String createPost() {
            Return "createPost"
}
//매핑 주소 - /shop/board/create.do, 전송 방식 – GET
//리뷰 작성 페이지로 넘어가게 하는 역할.


@RequestMapping(value="createPost.do", method=RequestMethod.POST)
  Public String Create(@ModelAttribute("post") Post post) {
            boardService.create(post);
            return "redirect:list.do";
}
//매핑 주소 - /shop/board/insert.do, 전송 방식 – POST
//게시글 작성을 처리. 매개변수로 받은 post를 DB에 저장하고 list.do로 넘어간다.

@RequestMapping("deletePost.do")
  Public String deletePost(@RequestParam int num) {
            boardService.delete(num);
            return "redirect:list.do"
}
//매핑 주소 - /shop/board/delete.do
//매개변수로 들어오는 값을 post_id로 가진 게시글 삭제 처리. 삭제하고 나서 list.do로 이동.

@RequestMapping(value="update.do, method=RequestMethod.Get)
  Public String updatePost(@RequestParam int num) {
            return "updatePost"
}
//매핑 주소 - /shop/board/update.do, 전송 방식 – GET
//게시글 수정 페이지로 넘어간다. 매개변수로 post_id가 들어오고, 이를 이용하여 DB에서 사용자가 수정할 게시물의 내용을 뽑아온다.

@RequestMapping(value="updatePost.do", method=RequestMethod.POST)
  Public String update(@ModelAttribute Post post) {
            boardService.update(post);
```

```
        return "redirect:list.do"
}
//매핑 주소 - /shop/board/updatePost.do, 전송 방식 – POST
//게시글 수정을 처리. 매개변수로 받은 post를 DB에서 수정하고 list.do로 넘어간다.

@RequestMapping("detail.do", method=RequestMethod.GET)
  Public ModelAndVIew DetailPost(@RequestParam int num, HttpSession session) {
        boardService.increaseCnt(num, session);
        ModelAndView mav = new ModelAndView("detail");
        mav.addObject("postDto", boardService.detail(num));
        return mav;
}
//매핑 주소 - /shop/board/detail.do, 전송 방식 – GET
//게시글의 디테일한 내용을 볼 수 있음. 조회수 증가 처리.
```

▶ 기능5. 마일리지 제도 [MileageController]

```
@RequestMapping("/shop/UserInfo.do")
  Public String ClickUserInfo(String userId ){
        boardService.getMileage(userId);
              ...
        return "redirect:userInfoForm"
        }
//마이페이지를 클릭하면 마일리지 내역이 포함된 userInfoForm이 뜬다.
//해당 폼에 사용자의 현 마일리지가 뜬다.

@RequestMapping("/shop/showDetailMileage.do")
  Public ModelAndView shoeDetailMileage(@RequestParam("userId") String id){
        ModelAndView mav = new ModelAndView("mList");
        mav.addObject("mList",boardService.getMileageDetail(id););
        return mav;
        }
//userInfoForm에서 마일리지(숫자)를 클릭하면 마일리지 상세보기 창으로 넘어간다
//여태까지 마일리지의 적립/사용내역이 Table형태로 뜬다.

@RequestMapping("/shop/insertMileage.do")
  Public void insertMileage(@RequestParam("userId") String id,
                                        @RequestParam("price") int price){
          boardService.insertMileage(userid,price);
        }
//insertMileage.do라고 입력되면 사용자 아이디와 가격을 받아서 마일리지를
//적립한다.

@RequestMapping("/shop/deleteMileage.do")
  Public void deleteMileage(@RequestParam("orderId") String id){
```

boardService.deleteMileage(id);
}
//주문id를 받아 Mileage 테이블에서 마일리지 내역을 삭제한다.

▶ **기능6. 메인 광고 [BannerController]**

@RequestMapping(value="addBanner.do", method=RequestMethod.POST)
Public String createBanner(@ModelAttribute("banner") Banner banner) {
boardService.addBanner(banner);
return "redirect:confirmBanner";
}
//URL shop/banner/addBanner.do을 클릭하면 배너 정보를 삽입하고 관리자에게 허가를 받는다. 그 후 입금 확인 View를 띄운다.

@RequestMapping("deleteBanner.do")
Public String deleteBanner(@RequestParam int banner_id) {
boardService.deleteBanner(banner_id);
return "redirect:/confirmBanner"
}
// URL deleteBanner.do가 클릭되면 해당 배너를 삭제하고 확인창에 배너가 삭제되었음을 알린다.

▶ **기능7. Q&A**

[ViewItemController]
@RequestMapping("/shop/viewItem.do")
Public String handlerRequest(@RequestParam("itemId") String itemId, ModelMap model) {
//기존에 있는 내용에 Q&A 불러오는 내용 추가.
Return "item";
}
// 매핑 주소 - /shop/viewItem.do?itemId=
// item 내용과 item에 대한 Q&A 내용을 반환.

[QuestionAnswerController]
@RequestMapping(value="createQuestion.do", method=RequestMethod.POST)
Public String Create(@ModelAttribute("question") Question question) {
qService.createQ(question);
return "redirect:viewItem.do"
}
//매핑 주소 - /shop/viewItem.do?itemId=, 전송 방식 – POST
//질문 작성을 처리. 매개변수로 받은 question을 DB에 저장하고 모든 작업이 끝나면 원래 있던 페이지로 돌아간다.

```
@RequestMapping("deleteQuestion.do")
Public String deleteQuestion (@RequestParam int num) {
        qService.deleteQ(num);
        return "redirect:viewItem.do"
}
//매핑 주소 - /shop/viewItem.do?itemId=
//매개변수로 들어오는 값을 question_id로 가진 게시글 삭제 처리. 삭제하고 나서 원래 페이지로 이동.

@RequestMapping(value="update.do, method=RequestMethod.Get)
Public String updateQuestion(@RequestParam int num) {
        return mav;
}
//매핑 주소 - /shop/viewItem/update.do, 전송 방식 – GET
//수정하기 버튼을 누르면 수정하는 폼을 담아서 viewItem.do로 다시 돌아오게 함.

@RequestMapping(value="updateQeustion.do", method=RequestMethod.POST)
Public String updateQuestion2(@ModelAttribute Question question) {
        qService.updateQ(question);
        return "redirect:viewItem.do"
}
//매핑 주소 - /shop/viewItem.do?itemId=, 전송 방식 – POST
//질문 수정을 처리. 매개변수로 받은 question을 DB에서 수정하고 원래 페이지로 돌아간다.

@RequestMapping(value="createAnswer.do", method=RequestMethod.POST)
Public String Create(@ModelAttribute("answer") Answer a) {
        aService.createA(answer);
        return "redirect:viewItem.do"
}
//매핑 주소 - /shop/viewItem.do?itemId=, 전송 방식 – POST
//질문에 대한 답변 작성을 처리. 매개변수로 받은 answer을 DB에 저장하고 모든 작업이 끝나면 원래 있던 페이지로 돌아간다.

@RequestMapping("deleteAnswer.do")
Public String deleteAnswer(@RequestParam int num) {
        aService.deleteA(num);
        return "redirect:viewItem.do"
}
//매핑 주소 - /shop/viewItem.do?itemId=
//매개변수로 들어오는 값을 answer_id로 가진 게시글 삭제 처리. 삭제하고 나서 원래 페이지로 이동.

@RequestMapping(value="update.do, method=RequestMethod.Get)
Public String updateAnswer (@RequestParam int num) {
```

	return mav;
	}
	//매핑 주소 - /shop/viewItem.do?itemId=, 전송 방식 – GET
	//수정하기 버튼을 누르면 수정하는 폼을 담아서 viewItem.do로 다시 돌아오게 함.
	
	@RequestMapping(value="updateAnswer.do", method=RequestMethod.POST)
	Public String updateAnswer2(@ModelAttribute Answer answer) {
	aService.updateA(question);
	return "redirect:viewItem.do"
	}
	//매핑 주소 - /shop/viewItem.do?itemId=, 전송 방식 – POST
	//답변 수정을 처리. 매개변수로 받은 answer을 DB에서 수정하고 원래 페이지로 돌아간다.

▶ **Service**

세부 내용	

Service 설명	▶ **기능1. 상품 등록**
	[itemFacade에 추가해야할 Method]
	List<Item> showItemList(int user_num);
	: UserId로 판매상품내역 질의
	void itemUpdate(Item item);
	: Item내역 수정
	void deleteItem(int item_num);
	: ItemId로 해당 아이템 삭제

	▶ **기능2. 경매**
	[PetStoreFacade에 추가해야할 Method]
	ArrayList<Bid> getBidList(String auction_id));
	: 경매 ID에 따른 입찰 리스트 가져오기
	void createBid(Bid bid);
	: 입찰 정보 저장
	void deleteBId(int bid_id);
	: 입찰 정보 삭제
	void beAuctionedOff(String auction_id);
	: 낙찰자 선정하고 낙찰자에게 쪽지 보내기

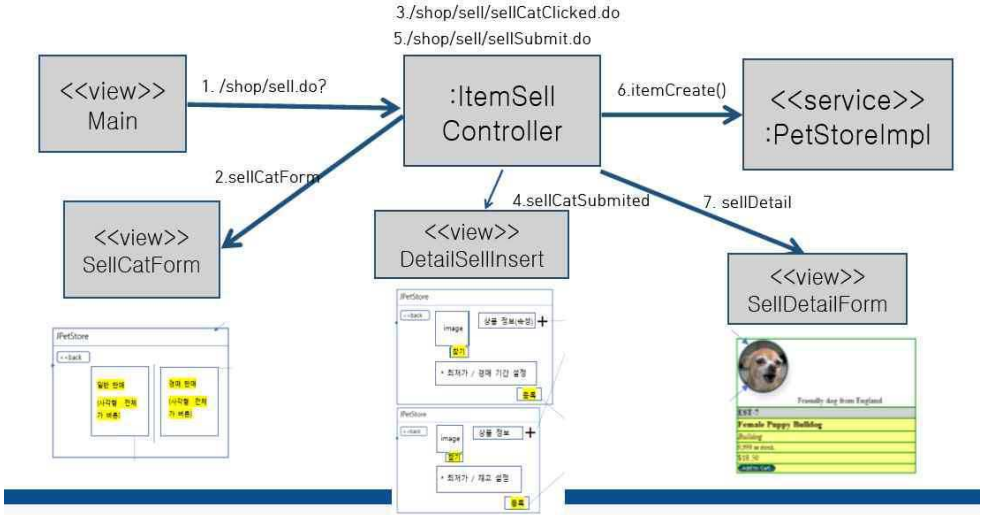
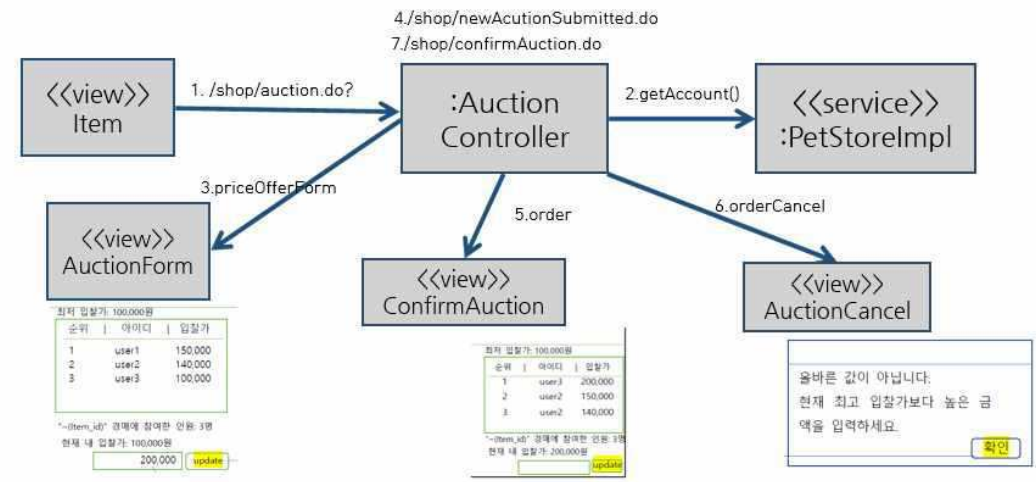
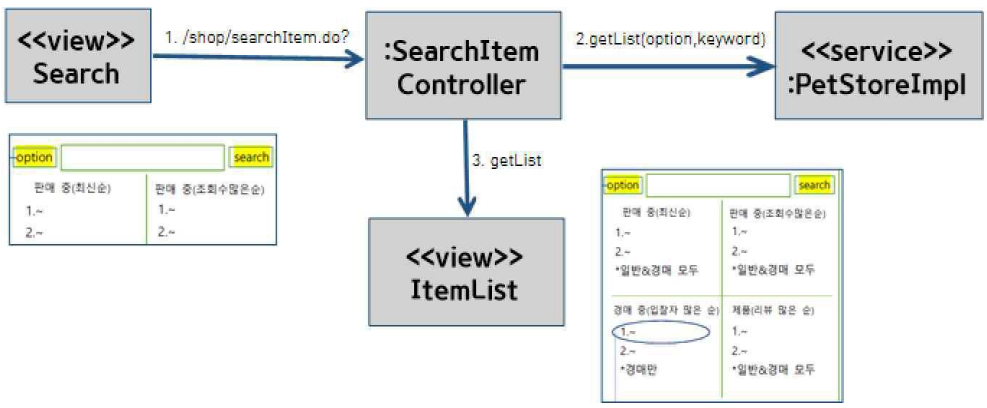
	▶ **기능3. 상품 검색** **[itemFacade에 추가해야할 Method]** List<Item> searchItemList(String option, String keyword); **:** option, keyword로 아이템 리스트 정렬.
	▶ **기능4. 판매자에 대한 후기** **[PetStoreFacade에 추가해야할 Method]** List<Post> list(String username)

	: 쿼리를 통해, username을 id로 가진 사용자에 대한 후기만을 불러오고, 이를 반환. void create(Post post) : post를 Post 테이블에 저장. void delete(int num) : num 값을 POST_ID로 가진 게시물을 삭제. void update(Post post) : post에 있는 값으로 게시물을 수정. void increaseCnt(int num, HttpSession session) : 조회수 증가 처리. Post detail(int num) : num값을 POST_ID로 가지는 게시물만을 불러오고, 이를 반환.
	▶ 기능5. 마일리지 제도 **[PetStoreFacade에 추가해야할 Method]** Int getMileage(int user_num); : 사용자아이디로 현 마일리지 조회 List<Mileage> getMileageDetail(int user_num); : 사용자아이디로 마일리지 적립/사용/양도 내역조회 void insertMileage(int order_id,price); : 주문번호와 가격을 받아 마일리지 적립 void deleteMileage(int order_id) : 주문번호를 받아 마일리지 내역 삭제

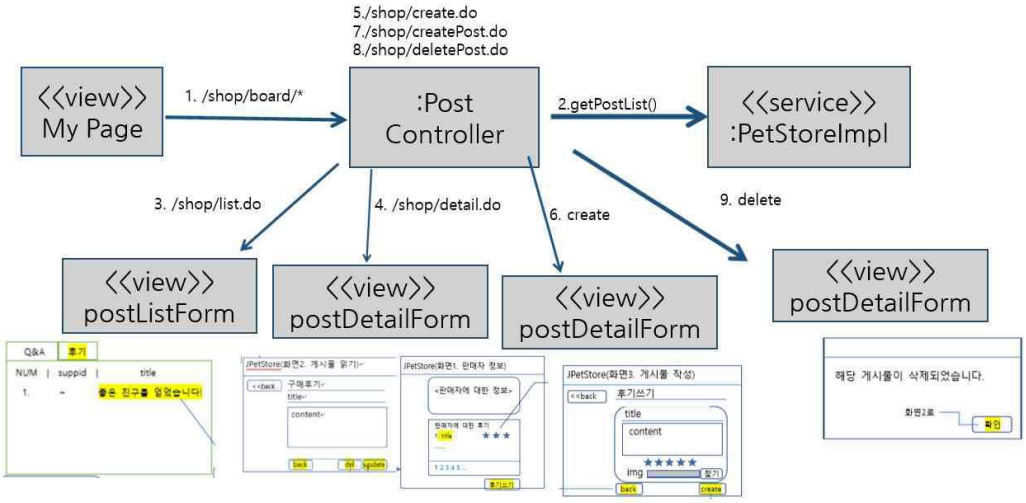
	▶ 기능6. 메인 광고
	[PetStoreFacade에 추가해야할 Method]
	void addBanner(Banner banner);
	: 광고 추가
	void deleteBanner(int banner_id);
	: 광고 삭제

	▶ 기능7. Q&A **[PetStoreFacade에 추가해야할 Method]** List<QAndA> listQAndA(int num) : Quesetion 테이블에서 num 값을 itme_id 필드 값으로 갖는 레코드들을 뽑아내고, 각각 Question의 question_id 값을 이용하여 Answer에서 쿼리 실행. 마지막으로 Question(1) : Answer(n) 형태의 Map 자료형으로 반환. void createQ(Question q) : q를 Question 테이블에 저장. void deleteQ(int num) : num 값을 QUESTION_ID로 가진 게시물을 삭제. void updateQ(Question q) : q에 있는 값으로 게시물을 수정. void createA(Answer a) : a를 Answer 테이블에 저장. void deleteA(int num) : num 값을 ANSWER_ID로 가진 게시물을 삭제. void updateA(Answer a) : a에 있는 값으로 게시물을 수정.

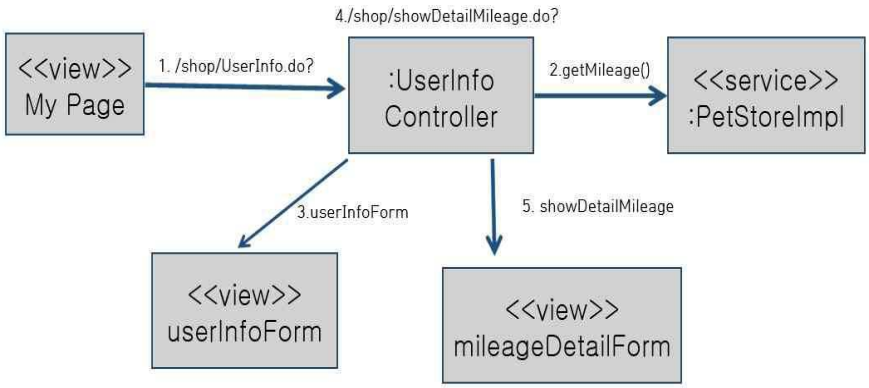
□ Request 처리 흐름

세부 내용	
Request 흐름	▶ 기능1. 상품 등록  ▶ 기능2. 경매  ▶ 기능3. 상품 검색 

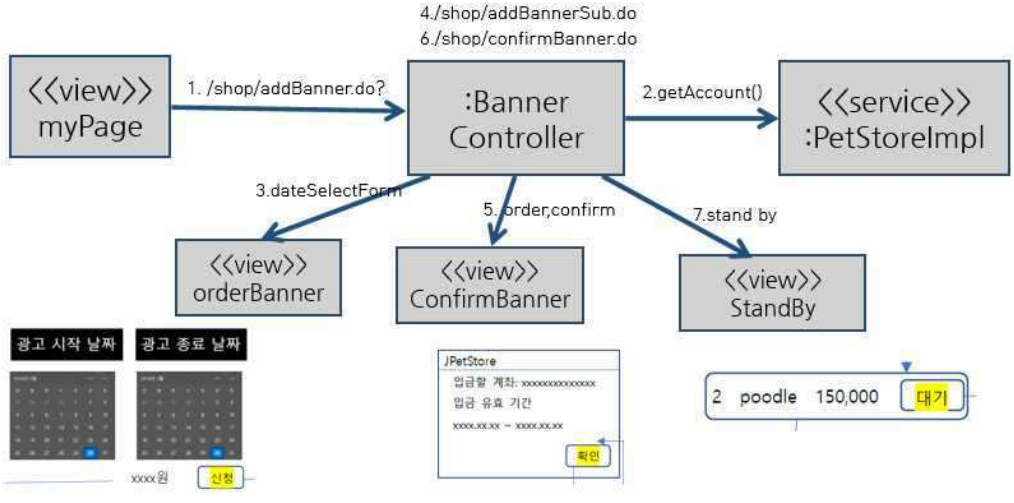
▶ 기능4. 판매자에 대한 후기



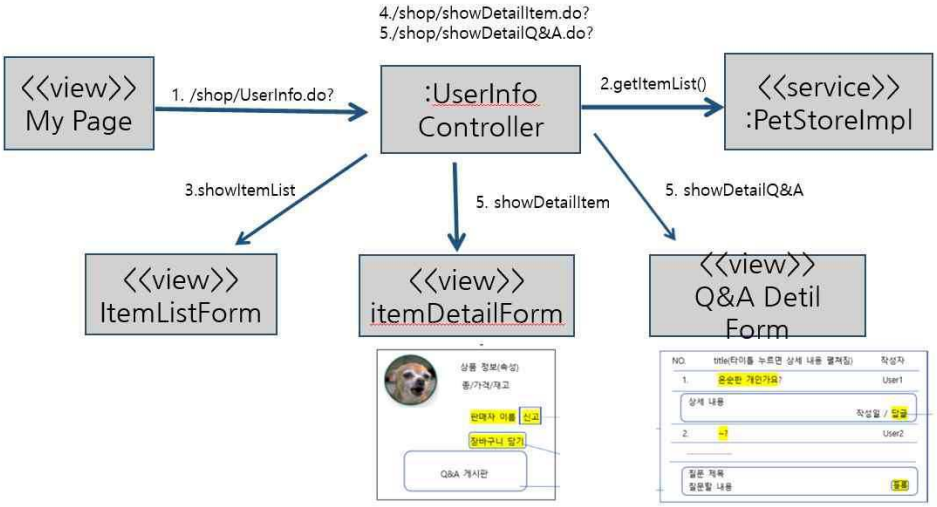
▶ 기능5. 마일리지 제도



▶ 기능6. 메인 광고



▶ **기능7. Q&A**


4./shop/showDetailItem.do?
5./shop/showDetailQ&A.do?
<<view>>
My Page
1. /shop/UserInfo.do?
:UserInfo
Controller
2.getItemList()
<<service>>
:PetStoreImpl
3.showItemList
5. showDetailItem
5. showDetailQ&A
<<view>>
ItemListForm
<<view>>
itemDetailForm
<<view>>
Q&A Detil
Form
상품 정보(속성)
종/가격/재고
판매자 이름
신고
장바구니 담기
Q&A 게시판
NO.
title(타이틀 누르면 상세 내용 열려짐)
작성자
1.
User1
상세 내용
작성일 / 답글
2.
User2
질문 제목
질문할 내용
질문