

# 최종 결과 보고서

(2022학년도 1학기)

과제명	아 맞다! - 쇼핑리마인더 App 개발
제출일자	2022. 06. 06
연구참여자	팀장: 김범수(20180895) 팀원: 정병윤(20151035) 팀원: 김성범(20180897)
수업명/조번호	모바일 프로그래밍 / 안드로이드 만들어조

# 목 차

1. 설계과제 제목 .....	
2. 설계목적 .....	
3. 설계과제 내용 .....	
4. 설계과제의 목표 .....	
5. 설계과정 .....	
6. 제작 .....	
7. 시험 .....	
8. 평가 .....	
9. 추진체계 .....	
10. 설계 추진일정 .....	
11. 설계 과정을 통해 배운 점 .....	

## 1. 설계과제의 제목

- 사고 싶은 물건을 까먹지 않도록 챙겨주는 쇼핑 리마인더 앱  
'아! 맞다!' 앱 설계

## 2. 설계 목적

- 1) Android Studio 코드 구성을 통한 상세 동작 이해
- 2) Application 개발에 사용되는 JAVA 언어 실력 향상
- 3) 프로그램 구현을 통해 Application 사용자들의 일상생활에 편리함을 제공
- 4) Application 개발에 사용되는 Android Studio 사용 능력 향상 및 프로그램 동작 이해
- 5) 팀 프로젝트를 통한 협업 능력 향상
- 6) 모든 파일과 구성 요소들을 일관되게 하는 네이밍 정의

## 3. 설계과제 필요성 및 내용

다양한 쇼핑 플랫폼이 생겨남에 따라, 자신이 구매하고자 하는 물건을 한눈에 모아보기가 쉽지 않고, 구매를 원했던 제품도 곧잘 까먹게 됩니다. 따라서 안드로이드 스튜디오를 사용하여 쇼핑 리마인더 앱을 만들기 하였습니다.

이 앱은 구매할 물품의 이름, 이미지, 가격, 메모, 중요도, 판매 링크 등을 사용자가 입력할 수 있도록 하고, 사용자가 등록해둔 물품을 최신순, 오래된순, 가격 낮은순, 높은 가격순, 관심목록 등으로 정렬하여 볼 수 있도록 하였습니다. 사용자가 등록해둔 물품 키워드 검색 기능을 제공합니다. 또한, 구매하지 않은 물품에 대하여 정기적으로 푸시알림을 받을 수 있도록 하고, 사용자가 등록한 물품을 수정 및 삭제할 수 있는 기능을 제공합니다.

## 4. 설계과제의 목표

본 과제에서 달성하고자 하는 기능은 다음과 같으며, 이외에도 본 과제에서 표 1과 같은 현실적인 제한요소들은 달성합니다.

- (1) 구매할 물건의 정보를 저장
- (2) 저장된 정보들을 설정한 기준에 따라 정렬
- (3) 사용자에게 주기적으로 알림 전송
- (4) 목록 리스트를 앨범형 또는 목록형으로 선택 기능
- (5) 메모 수정, 삭제 기능
- (6) 상품 사진 추가 기능
- (7) 저장된 물건 클릭 시 자세한 정보를 확인하는 기능

표 1. 본 프로젝트의 현실적 제한요소 항목

현실적 제한 요소들	내 용 (Content)
경제	<ul style="list-style-type: none"> <li>- 무료로 사용 가능한 안드로이드 스튜디오를 활용하여 개발하기에 초기 자본이 필요하지 않음</li> <li>- 개발 제품은 무료 배포함으로써 가격 경쟁력 확보에 용이</li> </ul>
편리	<ul style="list-style-type: none"> <li>- 다양한 쇼핑 애플리케이션으로 나누어져 있는 제품들을 사용자가 관심 있는 제품들만 따로 선별하여 비교할 수 있어서 매우 편리함</li> <li>- 사용자가 잊어버릴 수 있는 정보를 메모할 수 있고, 가격순, 최신순으로 나열하여 한눈에 볼 수 있어서 매우 편리함</li> <li>- 주기적으로 상품 알림 기능이 있어서 잊어버리지 않고 내용을 확인할 수 있음</li> </ul>
윤리	<ul style="list-style-type: none"> <li>- 사용자 일상생활에 도움이 되는 제품으로 불법 또는 도덕적 윤리에 어긋나지 않고 삶의 질을 향상하는데 도움이 될 것으로 예상함</li> </ul>
안전성	<ul style="list-style-type: none"> <li>- 사용자 개인정보가 들어가는 요소가 없고 간단한 정보 확인이나 사용자가 관심이 있는 제품을 비교하는 편의성 위주의 애플리케이션이므로 안전함</li> </ul>
유지관리 용이성	<ul style="list-style-type: none"> <li>- 스마트폰 전원만 연결되어 있으면 언제든지 정보를 확인할 수 있고 메모 내용이나 관심 제품의 목록을 추가하거나 삭제함으로써 유지하거나 관리하는데 유용할 것임</li> <li>- RecyclerView를 사용하여 ListView의 비해 데이터의 낭비가 적어짐</li> </ul>
사회	<ul style="list-style-type: none"> <li>- 사용자가 중요하다고 생각하는 정보를 알림 받고 관심 있는 제품을 사용자가 원하는 방식으로 비교, 나열하는 방식으로 특정 제품을 우대하거나 하나의 쇼핑물의 독점이 불가능하므로 사회적 영향력이 크지 않을 것으로 생각됨.</li> </ul>

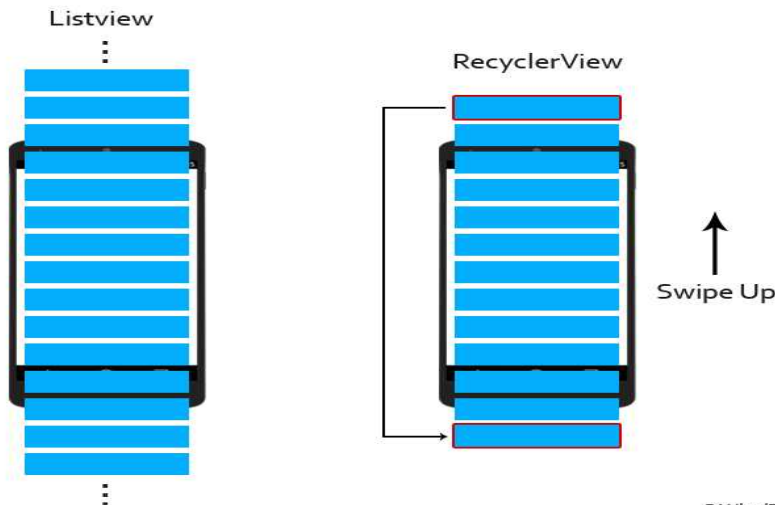
## 5. 설계과정

### 5.1 설계 기초이론

#### <사용자에게 리스트를 보여주며 스크롤 동작>

사용자는 등록 창에 등록한 상품 정보를 볼 수 있어야합니다. 이때 동적으로 추가하는 상품 정보와 상품의 개수가 많아지면 스크롤하면서 상품의 동작을 확인하기 위해 <RecyclerView>를 사용했습니다. 사용자가 추가하는 아이템에 대하여 리스트를 추가시켜 <ListView>를 사용하여도 괜찮지만, 이는 <ListView>의 단점이 있어 <RecyclerView>를 사용했습니다.

## Recycler view의 재활용성



기존 <ListView>를 사용하면 생성한 List의 개수만큼 스크롤을 할수록 값이 추가가 되며 100개의 List가 존재한다면 100개의 View 객체가 계속해서 사용이 됩니다. 그러나 <RecyclerView>의 경우 10개의 뷰 객체를 가지고 스크롤을 Up하면 화면을 넘어가 보이지 않는 View객체를 아래로 가져와 재활용하는 것이며, 뷰 객체만 재활용 하는 것이므로 데이터는 새로 갱신되기 때문에 <ListView>보다 훨씬 효율적인 기능입니다.

<RecyclerView>의 구성으로 필요한 것을 설명 드리겠습니다.

10개의 뷰 객체만 사용하기 위해 처음 객체를 기억하는 ViewHolder가 있습니다.

다음으로는 Adapter와 LayoutManager가 있습니다.

Adapter는 100개의 아이템의 데이터 값들을 <RecyclerView>에 Binding 시켜주기 위한 작업을 하는 객체입니다.

LayoutManager는 생성한 <RecyclerView>의 동작방식을 결정하는 기능을 합니다.

즉, <RecyclerView>의 기능을 정리하자면 Adapter를 통해 일정한 양의 ViewHolder를 만듭니다. 이후 데이터를 ViewHolder에 Bind해주면서 생성된 View객체에 데이터를 저장하는 것입니다. 이후 스크롤 동작을 하면 ViewHolder는 더 이상 생성하지 않고 Bind만 이용하여 값을 갱신하여 사용자가 화면을 볼 수 있도록 하는 것입니다.

## <앨범형, 리스트형 선택 동작>

이는 LayoutManager를 통해 gridLayoutManager와 LinearLayoutManager를 통해 제어했습니다. 또한 사용자에게 출력하는 사진의 크기와 텍스트 위치 변경 등 UI 변경을 위하여 각각의 View 객체를 다른 형태의 모습으로 나타내줍니다.

LinearLayoutManager	GridLayoutManager
<div>item</div> <div>item</div> <div>item</div> <div>item</div> <div>item</div>	<div> <div>item</div> <div>item</div> <div>item</div> <div>item</div> <div>item</div> </div> <div> <div>item</div> <div>item</div> <div>item</div> <div>item</div> <div>item</div> </div>

## <순서 정렬 메뉴 동작>

순서 정렬 메뉴의 처리 방식은 다음과 같습니다.

데이터 값을 저장한 List 배열이 있습니다. 이를 Comparator를 통해 비교합니다. Comparator는 객체를 비교할 수 있는 인터페이스이며 compare(T o1, T o2) 메서드를 통해 매개변수를 비교하는 용도입니다.

이를 비교하여 return 값으로 객체의 List 배열을 전부 비교하여 오름차, 내림차순서와 날짜 값의 년/월/일을 비교하여 최신순, 오래된순 그리고 관심 데이터 값을 확인하여 비교하고 이를 Collections.sort()를 이용하여 DataSet의 배열을 정렬했습니다.

## <검색창에 키워드 탐색하기>

사용자가 등록한 상품 메모 중에 자신이 원하는 메모를 빠르게 찾기 위해서 상품의 이름, 가격, 등록된 메모의 내용을 검색하는 기능을 넣으면 좋을 것 같았습니다. 따라서 우리의 메인화면을 구성하고 있는 <RecyclerView>에서 동작 가능한 검색 기능을 구현하기로 하였습니다. <EditText>를 사용하여 검색바를 구현하였고, filterList 메소드를 이용하여 검색할 때 검색어가 포함되면 그 목록을 보여주는 방식으로 검색 기능을 구현하였습니다.

처음에는 <SearchView>를 사용하여 검색바를 구현하고 검색 기능을 추가하려 하였으나 배열로 생성된 데이터들을 받아와서 검색기능을 실현하는데 어려움을 겪어 EditText로 변경하였음.

## <검색 중복 제거>

검색창에 키워드를 입력했을 때 키워드에서 발생하는 문제를 해결하기 위해 HashSet을 사용했습니다.

HashSet은 List의 모든 항목을 비교하여 중복 발생을 체크하여 중복이 발생한다면 List에서 제외합니다.

## <상품 저장 및 저장한 정보 이동하기>

사용자가 등록창에 저장한 정보를 확인하거나 초기화면으로 정보를 전달하기위해 intent를 활용하였습니다. 상품 정보를 등록하는 등록창에서 이름, 가격, 링크, 메모할 정보들은 문자열로 저장 후 putExtra와 getStringExtra를 활용하여 정보를 보내고 받으며, 사진 정보를 이동할 때에는 bitmap을 활용하여 문자열로 바꾼 후 intent로 정보를 보내는 것을 활용하여 RecyclerView의 Item 객체를 추가했습니다.

## <상품 수정 및 삭제>

사용자가 등록한 상품을 수정하고 삭제하는 작업을 하기위해 상품의 정보와 같이 상품의 포지션(위치)값을 넘겼습니다.

수정작업의 경우 값이 변경되었다면 해당 위치와 변경된 상품 정보를 가지고와 해당 리스트의 값을 바뀐 값으로 변경해준 뒤 어댑터의 연결하였습니다.

삭제작업의 경우 삭제의 대한 요청을 받으면 리스트 배열의 해당 위치를 삭제하고 어댑터를 다시 연결해줬습니다.

## <SQLite를 이용한 상품 정보를 데이터 베이스에 저장하기>

어플리케이션 동작을 종료하였을 때 이전에 입력하였던 정보를 다시 불러오기 위해 SQLite를 사용하였습니다. 처음에는 상품 등록창을 통해 상품 정보를 모두 입력한 후 저장을 누르게 되면 SQLite 테이블 생성을 한 후 각각의 행 안에 상품의 정보가 저장됩니다.

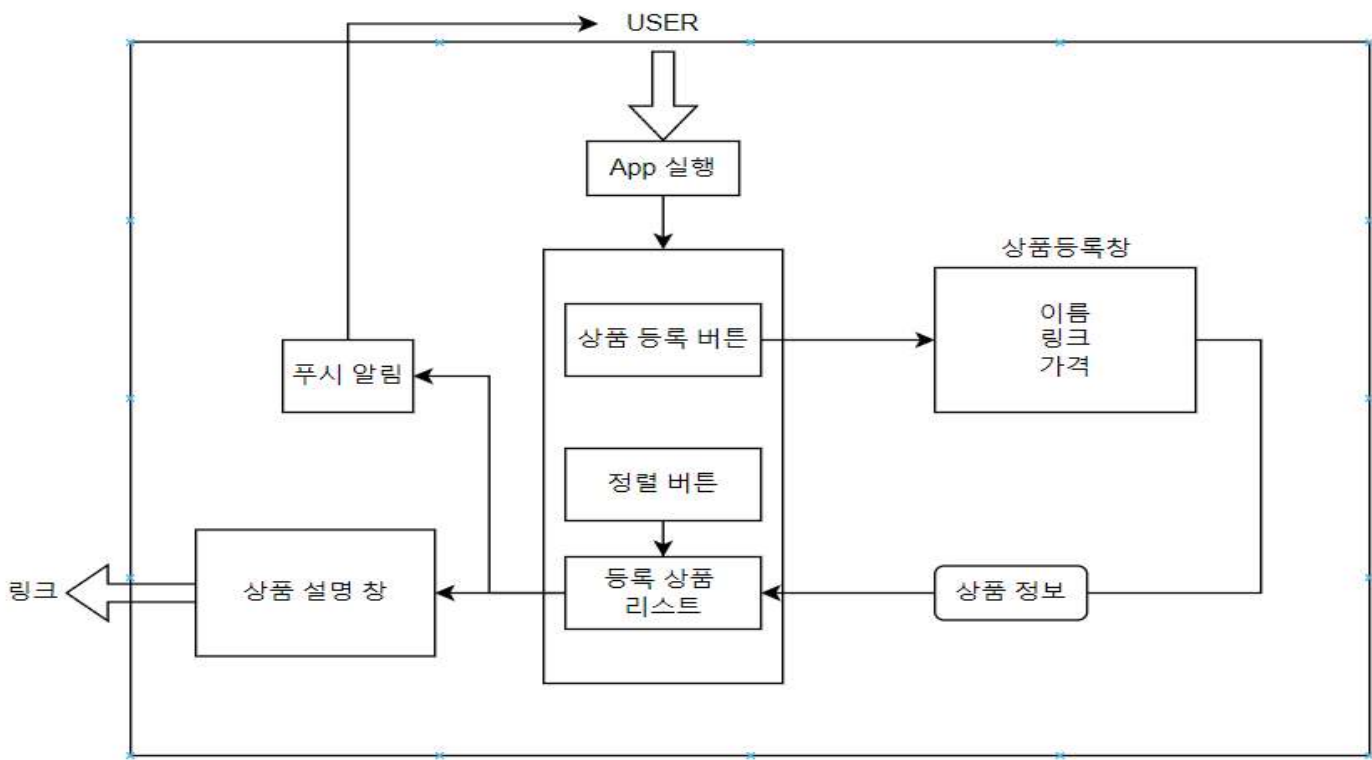
### <dialog를 이용한 설명서 만들기>

사용자에게 간단한 설명서를 제공하기 위해 dialog를 이용해 설명서를 만들었습니다. 대화상자를 생성할 때는 Alert Dialog.builder 클래스를 활용하였습니다. 그리고 사진으로 설명서를 제공하기 위해 image.setImageResource(R.drawable.add)를 사용하였고 마지막으로 show() 메소드를 사용하여 화면에 출력하였습니다.

### <푸시알림>

Notification 함수를 사용하여 사용자가 알림을 활성화하면 일정 시간 뒤에 상단바에서 알림창이 나타나고, 알림음이 울리도록 설정하였습니다. 기존에 버튼을 누르면 알림창고 알림음이 울리던 상태에서 동작 지연 함수를 추가하여 일정시간 이후에 기능이 동작하도록 구현하였습니다.

## 5.2 기능 블록도

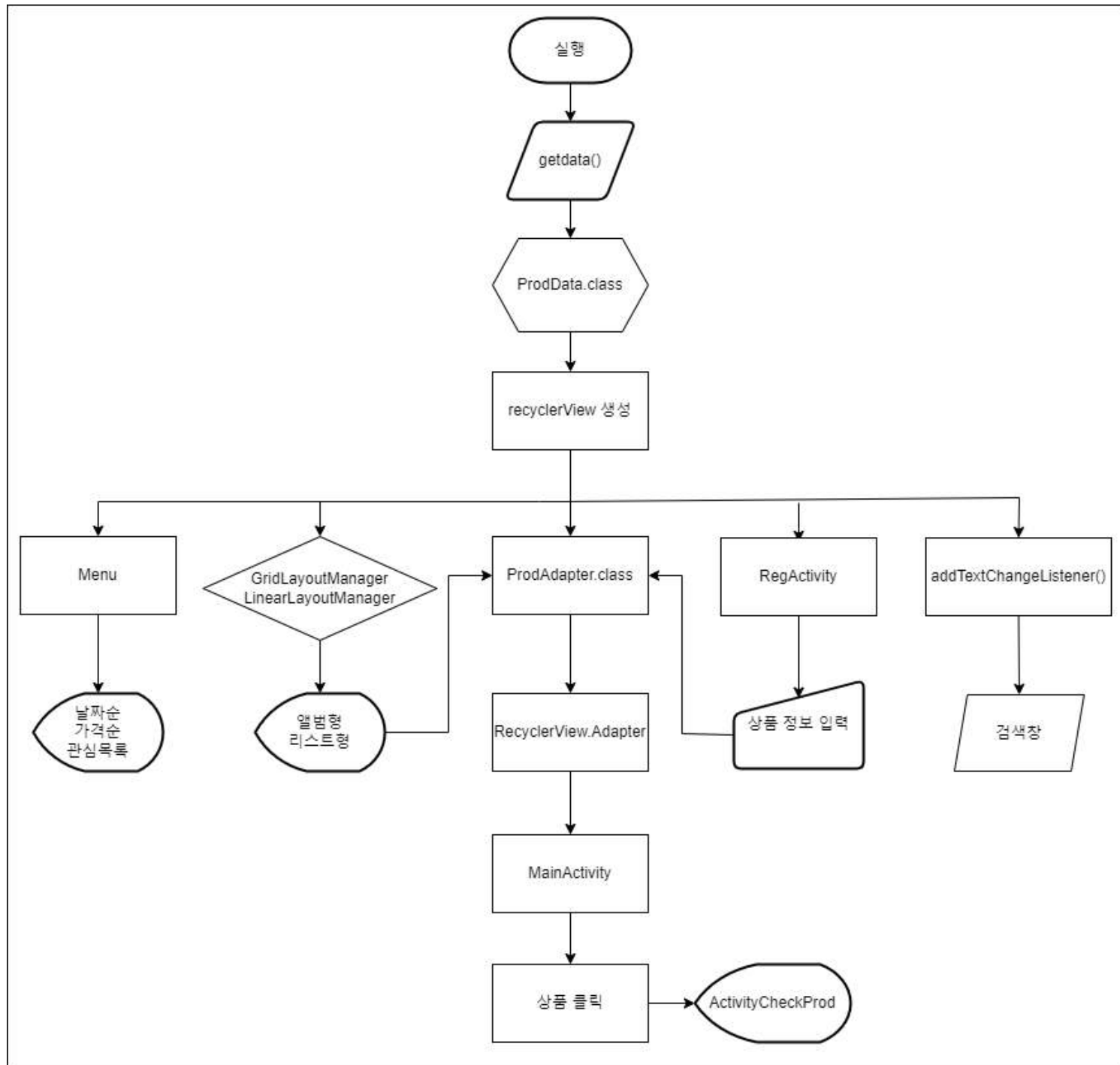


### 5.3 기능적 요구사항 명세서

요구사항	내용	우선 순위	설명
순서 정렬 아이콘	사용자가 나열하고 싶은 방식으로 보여주는 아이콘	3	사용자가 등록한 정보를 토대로 최신순, 오래된순, 가격 낮은순, 높은 가격순, 관심목록을 정렬하는 기능
상품 등록	상품에 대한 정보를 등록하는 창	2	등록창에서 이름, 가격, 링크, 사진, 메모 등을 기입할 수 있으며 수정이 가능합니다.
알림 메시지	사용자에게 주기적으로 푸시알림	7	사용자가 표시한 정보를 통해 핸드폰 푸시 알림을 보냅니다.
등록 상품 시각화	사용자가 등록한 상품을 편리하게 확인하는 창	1	사용자가 등록한 상품을 리스트 형이나 앨범형으로 나눠서 RecyclerView를 통해 사용자가 보기 편리하게 해줍니다.
중요도 표시	사용자가 중요하게 생각하는 상품을 체크	4	중요도 체크를 할 수 있는 버튼과 체크를 하면 색상이 바뀌며 중요한 상품만 관리하는 기능 또한 제공합니다.
등록 상품 상세 정보 확인	등록 상품을 터치하면 상세한 정보를 보여주는 창	6	리스트로 출력된 상품을 클릭 시 상세 정보를 확인하는 창으로 연결하기.
검색 기능	등록한 상품 검색 기능	5	키워드 검색을 통해 해당 키워드의 상품만 출력하는 동작



## 5.4 전체/부분 순서도 (상세설계)



## 6. 제작(Implementation)

### 6.1 제작 과정

#### 가. XML 처리

##### (1) activity\_main.xml

###### <상품 추가 버튼과 정렬메뉴 버튼 UI>

UI를 구성할 때, 전체적인 틀은 <LinearLayout>으로 구성하였습니다.

가장 위에 상품을 추가할 수 있는 버튼과 등록된 상품을 원하는 순서대로 정렬할 수 있는 메뉴 버튼을 <ImageButton>으로 추가하였다. '+'버튼을 누르면 상품을 등록할 수 있는 창이 뜨고, 오른쪽의 메뉴버튼을 길게 누르면 어떤 종류로 정렬할 수 있는지 고를 수 있는 메뉴창이 뜬다. 이미지버튼을 오른쪽으로 정렬하기 위해 <View> 위젯을 사용하여 왼쪽에 여백을 만들어줬습니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="50sp"
    android:orientation="horizontal">
    /*View로 왼쪽 여백 설정*/
    <View
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_weight="1"/>
    <ImageButton
        android:id="@+id/add_product"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:src="@drawable/add" />
    <ImageButton
        android:id="@+id/btnmenu"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:src="@drawable/menu2" />
</LinearLayout>
```

###### <검색창 UI>

```
/*검색창 리니어, EditText 사용*/
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <EditText
        android:id="@+id/search_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="검색어를 입력하십시오"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:singleLine="true"
        android:imeOptions="actionDone"
        android:layout_marginTop="10sp"
        android:layout_marginHorizontal="10sp"
        android:lines="1" />
</LinearLayout>
```



<EditText>를 이용해 검색바를 구현하였고, android:hint 명령어를 통해 검색어를 입력하는 곳이라는 정보를 제공할 수 있도록 했습니다. <EditText> 검색바에 검색어를 입력하게 되면 <RecyclerView>에 저장된 정보를 탐색하여 검색 동작을 합니다.

### <저장된 상품 목록을 테이블형식으로 보여주는 버튼, 리스트형식으로 보여주는 버튼 UI>

<View> 위젯으로 왼쪽에 여백을 주고 오른쪽 끝에 이미지 버튼을 구성하였습니다.

왼쪽 버튼을 누르면 테이블 형식으로, 오른쪽 버튼을 누르면 리스트 형식으로 상품을 볼 수 있습니다.

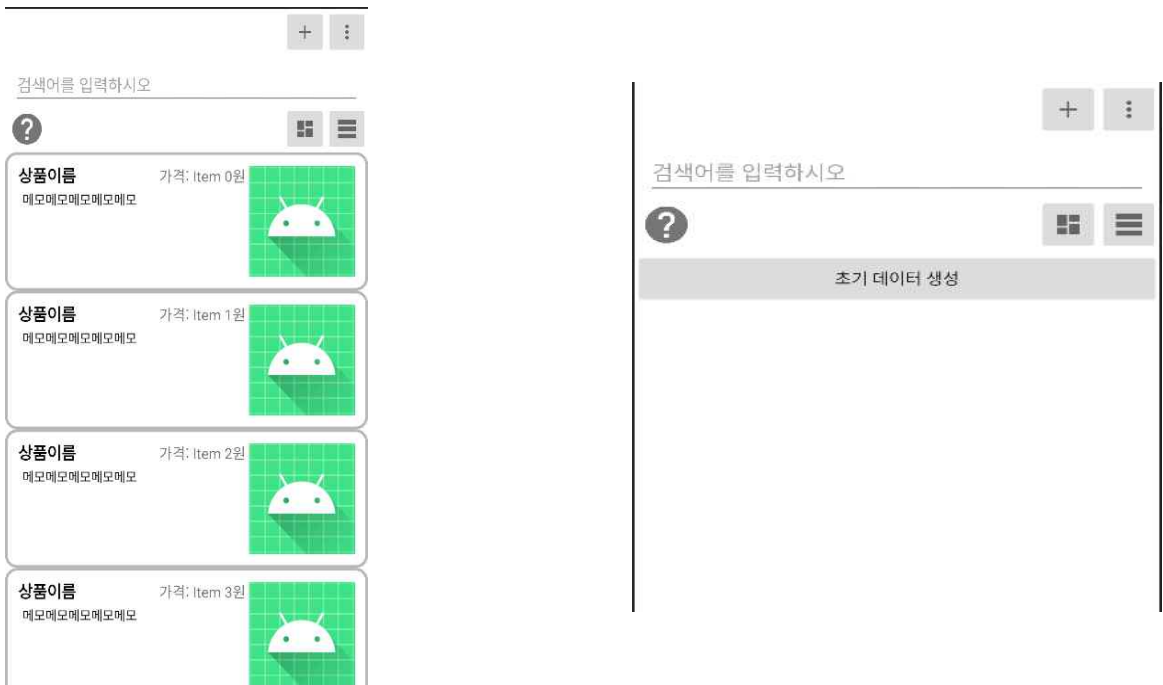
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="50sp"
    android:orientation="horizontal">
    /*View로 왼쪽 여백 설정*/
    <View
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_weight="1"/>
    <ImageButton
        android:id="@+id/btnalbum"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:src="@drawable/table" />
    <ImageButton
        android:id="@+id/btnvertical"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:src="@drawable/tableerow" />
</LinearLayout>
```

### <RecyclerView 생성>

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:listitem="@layout/recyclerview_item"/>
```

- 리사이클러뷰 생성은 위젯을 제외한 모든 공간으로 설정하며 이때 상품의 리스트가 적용되는 방식을 확인하기 위해 recyclerview\_item.xml을 사용합니다.

### <전체적인 UI>



왼쪽 사진이 설명한 위젯들과 <RecyclerView>를 다 포함한 전체 UI입니다.  
오른쪽 사진은 상품이 존재하지 않을 때 보이는 화면입니다.

## (2) recyclerview\_item.xml

위에서 보이는 RecyclerView의 들어가는 하나의 Item View객체입니다.



그림 5. item.xml(1)

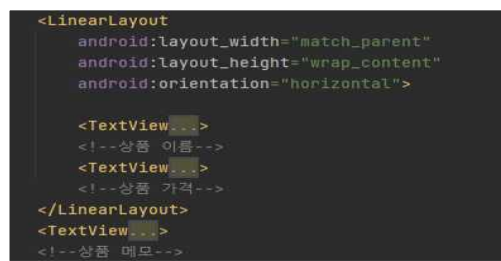
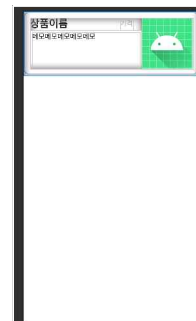


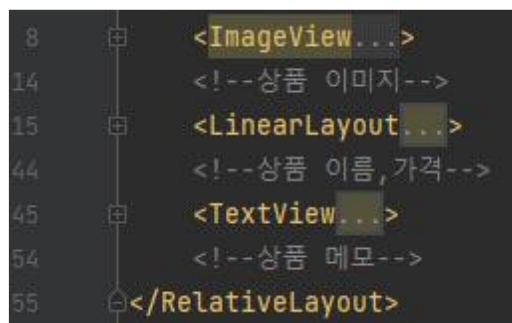
그림 6. item.xml(2)



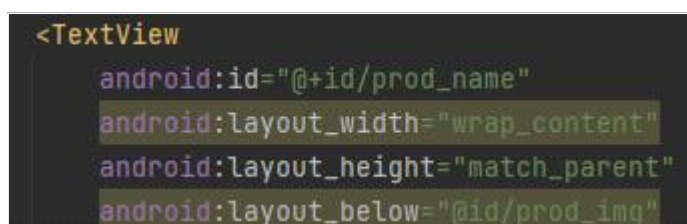
RecyclerView에 리스트형으로 들어갈 하나의 View 객체를 이와 같이 만듭니다.

## (3) recyclerview\_item2.xml

RecyclerView의 들어가는 다른 형태의 Item View객체입니다.



RecyclerView에 앨범형 출력에 들어가는 View 객체입니다.



이때 사용한 Layout은 상대적 위치를 표현하기 위해 <RelativeLayout>을 사용했습니다.

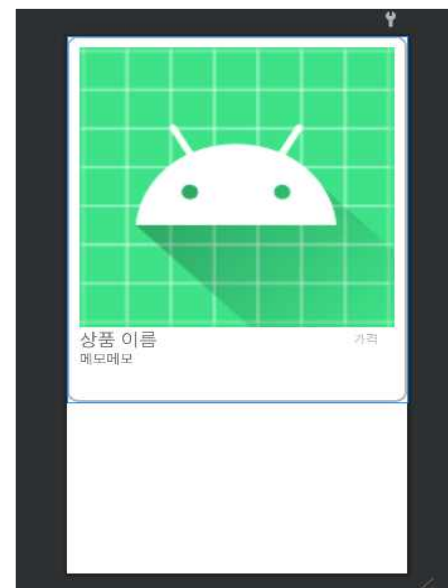


그림 8. item2.xml

#### (4) prod\_activity\_reg.xml

사진, 이름, 가격, 링크, 메모장을 입력할 수 있는 화면을 만듭니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center">
    <ImageView
        android:layout_width="300dp"
        android:background="@drawable/edge"
        android:src="@drawable/add"
        android:id="@+id/prod_img"
        android:layout_height="200dp"
    />
</LinearLayout>
```

[그림 1. 이미지]

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@+id/prod_name"
        android:id="@+id/prod_name"
        android:nextFocusDown="@+id/prod_price"
        android:inputType="text"/>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@+id/prod_price"
        android:id="@+id/prod_price"
        android:inputType="number"
        android:nextFocusDown="@+id/prod_link"/>
</LinearLayout>
```

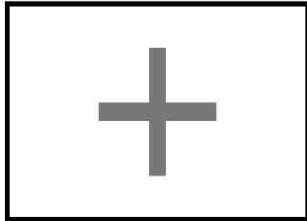
[그림 2]

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@+id/prod_link"
        android:id="@+id/prod_link"
        android:nextFocusDown="@+id/prod_note"
        android:inputType="textUri"/>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@+id/prod_note"
        android:id="@+id/prod_note"
        android:imeOptions="actionDone"/>
</LinearLayout>
```

[그림 3]

6:27 6

목록 추가 취소 저장



이름 \_\_\_\_\_

가격 \_\_\_\_\_ 원

링크 \_\_\_\_\_

메모를 입력하세요 \_\_\_\_\_

☆

[그림 4]

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:hint="@+id/prod_price"
    android:id="@+id/prod_price"
    android:nextFocusDown="@+id/prod_link"
    android:inputType="number"/>
```

가격 \_\_\_\_\_ 원

1	2	3	-
4	5	6	←
7	8	9	✕
,	0	.	→

가격 입력시 숫자패드가 나오게 하기 위해서 inputType을 number로 지정하였습니다.



```
<CheckBox
    android:id="@+id/prod_star"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:button="@drawable/btn_star" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/empty_star"
        android:state_checked="false"/>
    <item android:drawable="@drawable/full_star"
        android:state_checked="true"/>
</selector>
```

관심목록을 체크할 수 있는 체크박스를 구현하고, 체크박스에 사용할 이미지를 drawable에서 xml파일로 가져왔습니다. 이때, xml파일에 Destiny 자격을 부여하여 xxhdpi형식에 .png이미지를 추가하는 방식으로 사용하였습니다.

## (5) prod\_activity\_check.xml

전체적으로 등록창과 비슷하게 진행했습니다.

```
<ImageView
    android:background="@drawable/edge"
    android:id="@+id/prod_img"
    android:layout_width="300dp"
    android:layout_height="200dp" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="확인"
    android:id="@+id/ok_btn" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="이름 : "
    android:textSize="18sp" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:id="@+id/prod_name" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="가격 : "
    android:textSize="18sp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:id="@+id/prod_price" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="원"
    android:textSize="16sp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="메모 : "
    android:textSize="18sp" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:id="@+id/prod_note" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="링크 : "
    android:textSize="18sp" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:id="@+id/prod_link"
    android:linksClickable="true"
    android:autoLink="web" />
```

링크 클릭 시 해당 링크로 이동할 수 있게 동작했습니다.

## (6) prod\_activity\_modify.xml

등록창과 확인창의 동작을 합친 부분입니다.

```
<ImageView  
    android:layout_width="300dp"  
    android:background="@drawable/edge"  
    android:src="@drawable/image"  
    android:id="@+id/mod_img"  
    android:layout_height="200dp"
```

```
<CheckBox...> //관심
```

```
<LinearLayout
```


```
    <Button...> //수정 취소  
    <Button...> //수정 완료  
</LinearLayout>
```

```
<EditText...> //가격
```

```
<EditText...> //이름
```

```
<EditText...> //링크
```

```
<EditText...> //메모
```



이름

가격 원

링크

메모를 입력하세요

☆

수정 취소   수정 완료

3:46



콧스 COX CK87 게이트론 텐키리스 키보드

49,900 원

<http://naver.me/Gx0tp3oe>

키보드, 검정색, 빨간색

☆

수정 취소   수정 완료

위와 같이 사용하여 등록창과 동일한 UI를 구성하지만 아래 수정 완료와 취소 버튼이 생겼으며, 상품 정보와 사진을 EditText와 ImageView로 가져와 사용자가 다시 수정할 수 있습니다.

## 나. 기타 작업 코드

### <설명서>

#### - dialog.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical" >

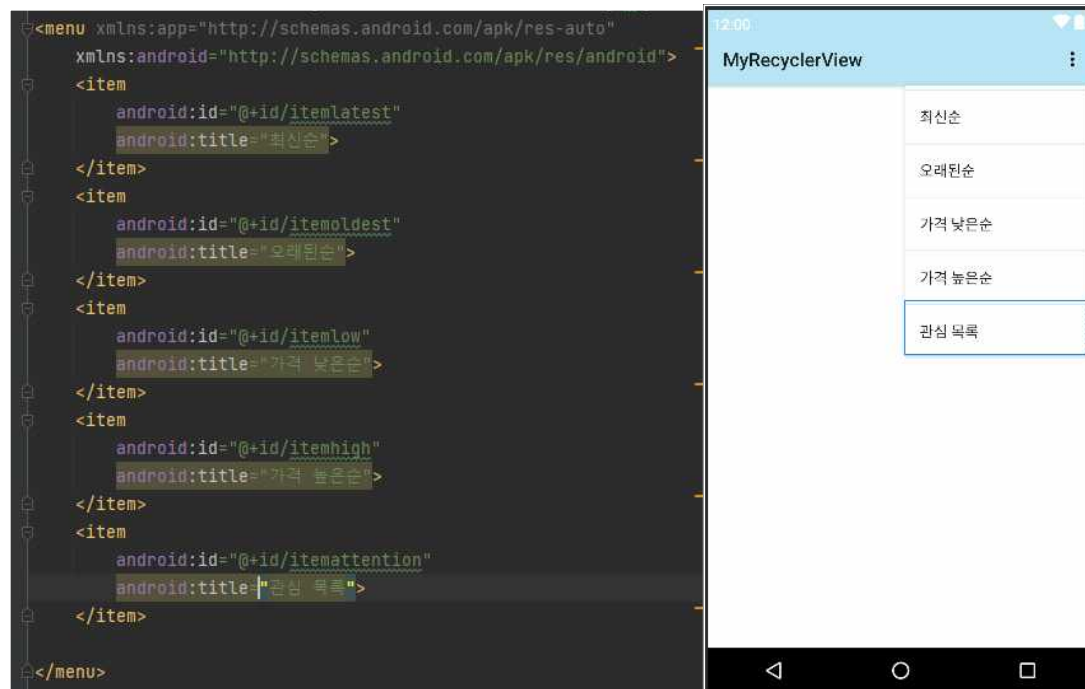
    <ImageView
        android:id="@+id/ivPoster"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

설명서 버튼을 클릭하였을 때 사진으로 저장된 설명서가 나오도록 합니다.

### <menu 폴더>

#### - main\_menu.xml

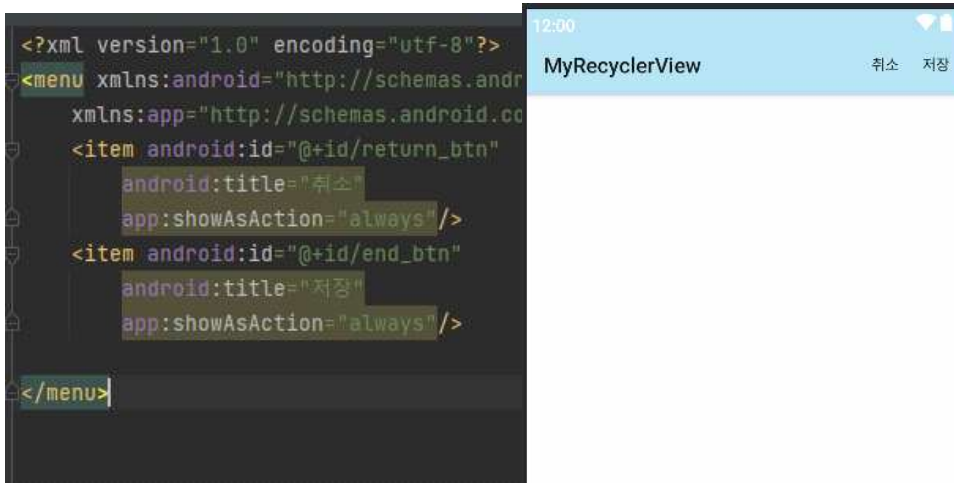


```
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/itemlatest"
        android:title="최신순">
    </item>
    <item
        android:id="@+id/itemoldest"
        android:title="오래된순">
    </item>
    <item
        android:id="@+id/itemlow"
        android:title="가격 낮은순">
    </item>
    <item
        android:id="@+id/itemhigh"
        android:title="가격 높은순">
    </item>
    <item
        android:id="@+id/itemattention"
        android:title="관심 목록">
    </item>
</menu>
```

이처럼 아이템을 생성하고 메뉴 동작을 합니다.



## - reg\_menu.xml



이처럼 아이템을 생성하고 메뉴 동작을 합니다.

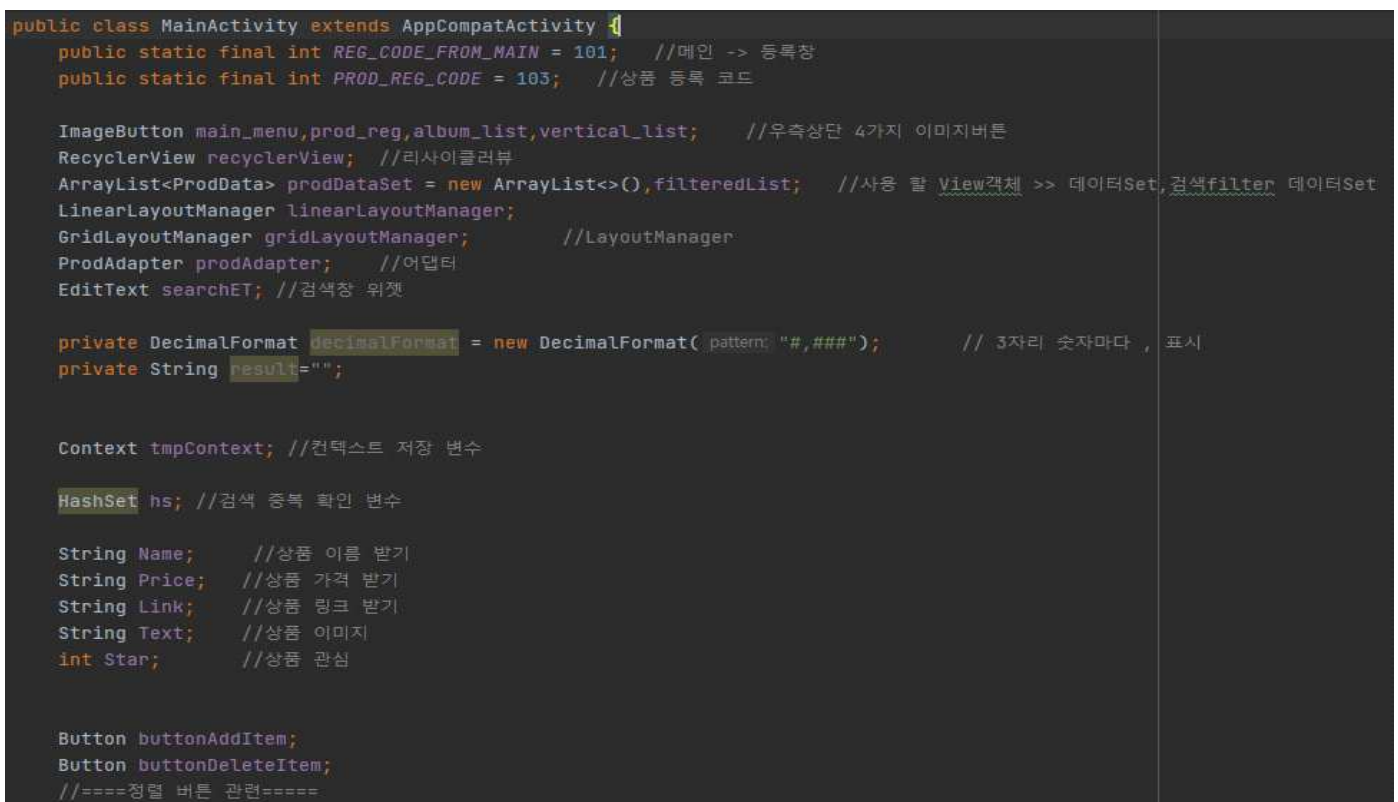
## <AndroidManifest.xml>



Activity를 사용하는 만큼 추가하기 << Intent 동작 시 필요

## 다. MainActivity.java

### (1) 기초 작업



사용 변수 선언

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //====어플리케이션 이름 숨기기====
    getSupportActionBar().hide();
    //====어플리케이션 이름 숨기기====

```

어플리케이션 이름을 숨기기 위한 작업입니다.

```

//====리사이클러뷰 필수 =====
tmpContext = (Context) this;
recyclerView = (RecyclerView) findViewById(R.id.recyclerView); //리사이클러뷰

//tmpContext = (Context) this;
LinearLayoutManager = new LinearLayoutManager(tmpContext);
recyclerView.setLayoutManager(linearLayoutManager); // LayoutManager 설정

```

```

//어댑터 생성
prodAdapter = new ProdAdapter(getApplicationContext(), prodDataSet, db, new ProdAdapter.OnItemClickListener() {

recyclerView.setAdapter(prodAdapter); // 어댑터 설정, 그 담겨져있는 데이터 어댑터를 리사이클러뷰의 세팅
//====리사이클러뷰 필수 =====

```

이는 리사이클러뷰를 사용하기 위해 하는 작업들입니다.

리사이클러뷰에 LinearLayoutManager를 설정해주고 현재 지닌 데이터 값의 리스트로 어댑터를 설정해줍니다.

이때 어댑터를 생성할 때 다음과 같이 ItemClick 동작을 추가합니다.

```

//어댑터 생성
prodAdapter = new ProdAdapter(getApplicationContext(), prodDataSet, db, new ProdAdapter.OnItemClickListener() {
    @Override
    //=====상품 클릭시=====
    public void onItemClick(View v, int position) {
        Intent intent2 = new Intent( getApplicationContext(),ActivityCheckProd.class);

        //이미지 가져오기
        Bitmap prod_Bitmap = prodDataSet.get(position).getProd_bit_image();

        if(prod_Bitmap ==null){
            byte[] byteArray = null;
            intent2.putExtra( name: "image",byteArray);
        }else {
            ByteArrayOutputStream stream = new ByteArrayOutputStream();
            prod_Bitmap.compress(Bitmap.CompressFormat.JPEG, /quality: 100, stream);
            byte[] byteArray = stream.toByteArray();
            intent2.putExtra( name: "image",byteArray);
        }
    }
}

```

```

String prod_name = prodDataSet.get(position).getProd_name();
String prod_price = prodDataSet.get(position).getProd_price();
String prod_note = prodDataSet.get(position).getProd_note();

//이름, 가격, 메모, 링크, 이미지, 관심, 날짜, 포지션
intent2.putExtra( name: "name",prod_name);
intent2.putExtra( name: "price",prod_price);
intent2.putExtra( name: "note", prod_note);

String prod_link1 = prodDataSet.get(position).getProd_link();
intent2.putExtra( name: "link",prod_link1);

int prod_star1 = prodDataSet.get(position).getProd_star();
intent2.putExtra( name: "star",prod_star1);

String prod_date1 = prodDataSet.get(position).getProd_date();
intent2.putExtra( name: "date",prod_date1);

//intent2.putExtra("image",byteArray);

intent2.putExtra( name: "position",position);

startActivityForResult(intent2,CHECK_PROD);
}
//=====상품 클릭시=====

```

위와 같이 어댑터를 생성할 때 각 상품정보 클릭을 추가하여, 해당 상품 클릭 시 상품의 대한 정보와 position(위치)이라는 위치 값과 함께 Intent를 통해 확인창으로 넘어갈 수 있게끔 하는 동작을 합니다.

## (2) 메뉴 동작

```

//=====메뉴 동작 =====
main_menu =(ImageButton) findViewById(R.id.btnmenu);
registerForContextMenu(main_menu);
//=====메뉴 동작 =====

```

메뉴동작은 우선 menu 이미지를 눌렀을 때 menu 창이 뜨게끔 동작을 하게끔 합니다. 이에 대한 동작 코드는 아래와 같습니다.

```

//=====메뉴 동작 =====
@Override
public void onCreateContextMenu(ContextMenu menu, View view,
                                ContextMenu.ContextMenuInfo menuInfo){

    super.onCreateContextMenu(menu, view, menuInfo);

    MenuInflater mmInflater = getMenuInflater();
    if(view == main_menu){
        mmInflater.inflate(R.menu.main_menu, menu);
    }
}

```

<main\_menu.xml>로 메뉴를 생성하여 이를 화면에 적용시키고 아래에서 동작합니다.

```

@Override
public boolean onContextItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.itemlatest: //최신순
            Comparator<ProdData> latest = new Comparator<ProdData>() {...} ;
            Collections.sort(prodDataSet, latest);
            prodAdapter.notifyDataSetChanged();
            return true;
        case R.id.itemoldest: //오래된순
            Comparator<ProdData> oldest = new Comparator<ProdData>() {...} ;
            Collections.sort(prodDataSet, oldest);
            prodAdapter.notifyDataSetChanged();
            return true;
        case R.id.itemlow: //가격 낮은순
            Comparator<ProdData> noAsc = new Comparator<ProdData>() {
                @Override
                public int compare(ProdData item1, ProdData item2) {...}
            } ;
            Collections.sort(prodDataSet, noAsc);
            prodAdapter.notifyDataSetChanged();
            return true;
        case R.id.itemhigh: //가격 높은순
            Comparator<ProdData> noDesc = new Comparator<ProdData>() {...} ;
            Collections.sort(prodDataSet, noDesc) ;
            prodAdapter.notifyDataSetChanged() ;
            return true;
        case R.id.itemattention: //관심목록
            /*...*/
            //관심목록 상단출력
            Comparator<ProdData> statusStar = new Comparator<ProdData>() {...} ;
            Collections.sort(prodDataSet, statusStar);
            prodAdapter.notifyDataSetChanged();
            return true;
    }
    return false;
}

//=====메뉴 동작 =====

```

메뉴 아이템의 선택된 항목에 따라서 동작을 합니다.



이는 Comparator와 Collections.sort를 이용하여 다음과 같이 사용합니다.

```
case R.id.itemLow: //가격 낮은순
    Comparator<ProdData> noAsc = new Comparator<ProdData>() {
        @Override
        public int compare(ProdData item1, ProdData item2) {
            int ret ;

            if (Integer.parseInt(item1.getProd_price()) < Integer.parseInt(item2.getProd_price()))
                ret = -1 ;
            else if (Integer.parseInt(item1.getProd_price()) == Integer.parseInt(item2.getProd_price()))
                ret = 0 ;
            else
                ret = 1 ;

            return ret ;

            // 위의 코드를 간단히 만드는 방법.
            // return (item1.getNo() - item2.getNo()) ;
        }
    };
    Collections.sort(prodDataSet, noAsc);
    prodAdapter.notifyDataSetChanged();
    return true;
```

```
if (Integer.parseInt(StringReplace(item1.getProd_price())) < Integer.parseInt(StringReplace(item2.g
    ret = -1 ;
else if (Integer.parseInt(StringReplace(item1.getProd_price())) == Integer.parseInt(StringReplace(i
    ret = 0 ;
```

이는 ProdData에 데이터 형식으로 getter로 상품의 가격을 가져와서 비교합니다. 이후 비교된 상품을 통해 첫 번째 아이템의 가격이 적으면 return을 (-1)로 하는 방식으로 정렬한 뒤 이렇게 만들어진 새로운 리스트 배열을 정렬합니다. 이때 가격의 비교는 정수를 비교하지만 실제 텍스트입력은 문자열로 들어가기 때문에 Integer.parseInt를 사용하여 문자열을 정수로 변환시켜서 비교합니다. 마지막으로 들어가는 prodAdapter.notifyDataSetChanged();는 상품어댑터의 정보가 변경된 것을 새로 고침 하는 것입니다. 위 과정에서 ret의 return값이 (-1)와 1이 바뀌면서 가격의 대한 제어를 합니다. 이때 ','와 같이 반점의 특수문자가 들어가는 경우가 있어 StringReplace를 한 것을 코드에 넣어서 처리하는 방식으로 변경했습니다.

사용되는 StringReplace는 아래 코드입니다.

```
/*=====특수문자 제거 함수=====*/
public static String StringReplace(String str){
    String match = "[^\\uAC00-\\uD7A30-9a-zA-Z]";
    str = str.replaceAll(match, replacement: "");
    return str;
}
/*=====특수문자 제거 함수=====*/
```

//모든 문자, 숫자 들을 제외한 값을 없애는 코드

그렇다면 위 동작처럼 비교하는 동작을 하기위해 날짜 데이터 비교는 다음과 같이 합니다.

```
public int compare(ProdData item1, ProdData item2) {
    int ret;
    int year1, mon1, day1;
    int year2, mon2, day2;

    year1 = Integer.parseInt(item1.getProd_date()) / 10000;
    year2 = Integer.parseInt(item2.getProd_date()) / 10000;

    mon1 = (Integer.parseInt(item1.getProd_date()) % 10000) / 100;
    mon2 = (Integer.parseInt(item2.getProd_date()) % 10000) / 100;

    day1 = (Integer.parseInt(item1.getProd_date()) % 10000) % 100;
    day2 = (Integer.parseInt(item2.getProd_date()) % 10000) % 100;
```

```
if (year1 != year2) {
    if (year1 < year2)
        ret = -1;
    else if (year1 == year2)
        ret = 0;
    else
        ret = 1;
    return ret;
} else if (mon1 != mon2) {
    if (mon1 < mon2)
        ret = -1;
    else if (mon1 == mon2)
        ret = 0;
    else
        ret = 1;
    return ret;
} else {
    if (day1 < day2)
        ret = -1;
    else if (day1 == day2)
        ret = 0;
    else
        ret = 1;
    return ret;
}
```

프로그램의 상품을 등록한 날짜를 통해 최신순과 오래된순을 비교하기 위해서 각 item에 대하여 날짜의 년/월/일을 나눕니다. 이후 가격을 비교한 코드와 동일하게 날짜를 비교하는데 이때 년/월/일에 우선순위를 차례대로 해서 정렬했습니다. 관심목록에 대한 메뉴는 상품의 관심(Prod\_star)의 값을 비교하여 1과 0으로 설정하여, 이를 높은 순 정렬을 통해 상단에는 관심메뉴가 우선하여 출력하도록 했습니다.

### (3) 앨범형, 리스트형 출력

#### <MainActivity.java>

```
//=====출력 리스트 모양 동작 =====
album_list = (ImageButton) findViewById(R.id.btnalbum);
vertical_list = (ImageButton) findViewById(R.id.btnvertical);

album_list.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        prodAdapter.setItemViewType(ProdAdapter.ALBUM_LIST);
        gridLayoutManager = new GridLayoutManager(tmpContext, spanCount 2);
        recyclerView.setLayoutManager(gridLayoutManager);
    }
});

vertical_list.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        prodAdapter.setItemViewType(ProdAdapter.VERTICAL_LIST);
        linearLayoutManager = new LinearLayoutManager(tmpContext);
        recyclerView.setLayoutManager(linearLayoutManager); // LayoutManager 설정
    }
});
//=====출력 리스트 모양 동작 =====
```

앨범형과 리스트형의 대한 동작은 위 사진처럼 앨범 또는 리스트형의 이미지 버튼을 클릭할 시 클릭한 Image Button에 따라서 앨범형은 GridLayoutManager를 통하여 레이아웃매니저를 설정하고 어댑터의 ViewType을 ALBUM\_LIST로 설정합니다. 리스트형은 기존 레이아웃 매니저처럼 설정한 뒤 어댑터의 ViewType을 VERTICAL\_LIST로 설정합니다.

#### <ProdAdapter.java>

```
//===== ViewHolder 생성 RecyclerView Adapter 필수 구현 항목 =====
@NonNull
@Override // ViewHolder 객체를 생성하여 리턴한다.
public ProdAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    mContext = parent.getContext();
    if( viewType == VERTICAL_LIST){
        View view = LayoutInflater.from(mContext)
            .inflate(R.layout.recyclerview_item, parent, attachToRoot false);
        ProdAdapter.ViewHolder viewHolder = new ProdAdapter.ViewHolder(view);
        return viewHolder;
    }
    else{
        View view = LayoutInflater.from(mContext)
            .inflate(R.layout.recyclerview_item2, parent, attachToRoot false);
        ProdAdapter.ViewHolder viewHolder = new ProdAdapter.ViewHolder(view);
        return viewHolder;
    }
}
//===== RecyclerView Adapter 필수 구현 항목 =====
```

위에서 Click된 Image 버튼의 따라서 viewType에 따라 어댑터(ProdAdapter.java)의 ViewHolder 형식이 변경됩니다

## (4) 검색창 기능

<ProdAdapter.java>

```
/*필터리스트 설정*/
public void filterList(ArrayList<ProdData> filteredList) {
    prodDataSet = filteredList;
    notifyDataSetChanged();
}
```

어댑터에 filterList 메소드를 선언해주었다. 검색할 때, 검색어가 포함되어 있다면 그 목록을 보여주는 기능을 한다.

<MainActivity.java>

```
//=====검색 동작 =====
searchET = findViewById(R.id.search_view); //써치뷰
filteredList=new ArrayList<>(); //필터리스트 생성

searchET.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void afterTextChanged(Editable editable) {
        String searchText = searchET.getText().toString();
        searchFilter(searchText);
        if(searchText.replace( target: " ", replacement: "").equals("")) {
            searchET.clearFocus();
            prodAdapter.filterList(prodDataSet);
        }
    }
});
//=====검색 동작 =====
```

필터리스트 배열을 생성하여 입력된 정보들의 비교를 통해 검색하고자 하는 것을 찾을 수 있도록 합니다. 이때 add TextChangedListener에 TextWatcher라는 인터페이스를 연결해주어 사용자가 EditText에 텍스트를 입력할 때, 입력 시점에 따라 이벤트를 주는 방법을 사용하여 필터링을 구현하였습니다. 입력하여 변화가 생기기 전에 처리는 before TextChanged() 부분, 변화와 동시에 처리를 해주하고자 할 때는 onTextChanged() 부분, 입력이 끝났을 때 처리는 after TextChanged() 부분에서 탐색을 하고 실시간으로 결과를 반영하여 나타냅니다.

## (5) 검색창 중복 문제

```
hs = new HashSet(filteredList);
```

이와 같이 검색으로 만들어진 filteredList를 사용해 HashSet을 생성합니다.

```
if(!(hs.contains(filteredList))){
    filteredList.add(prodDataSet.get(i));
    hs.addAll(filteredList);
}
```

이후 이처럼 생성된 HashSet에 문자열이 포함되어 있지 않으면 값을 추가하고 아닌 경우 값을 추가하는 다른 동작을 하지 않습니다.

## 라. ProdAdapter.java

```
public class ProdAdapter extends RecyclerView.Adapter<ProdAdapter.ViewHolder>{
    public static final int VERTICAL_LIST = 0;
    public static final int ALBUM_LIST = 1;
    int mItemViewType;

    private ArrayList<ProdData> prodDataSet;
    private Context mContext;
```

우선 ProdAdapter의 extends를 통해 RecyclerView.Adapter를 해줍니다.  
이후 사용할 변수들을 선언해줍니다.

```
//=====뷰홀더 클래스 =====
public class ViewHolder extends RecyclerView.ViewHolder {
    protected TextView prod_name;
    protected TextView prod_price;
    protected TextView prod_link;
    protected TextView prod_note;
    protected ImageView prod_img;

    public ViewHolder(@NonNull View itemView) { //View 객체 홀더
        super(itemView);

        this.prod_name=(TextView) itemView.findViewById(R.id.prod_name);
        this.prod_price=(TextView) itemView.findViewById(R.id.prod_price);
        this.prod_link=(TextView) itemView.findViewById(R.id.prod_link);
        this.prod_note=(TextView) itemView.findViewById(R.id.prod_note);
        this.prod_img = (ImageView) itemView.findViewById(R.id.prod_img);
    }

    //View 객체 값 설정
    void onBind(ProdData data, final OnItemClickListener listener) {
        prod_name.setText(data.getProd_name());
        prod_price.setText(data.getProd_price());
        prod_note.setText(data.getProd_note());
        prod_img.setImageBitmap(data.getProd_bit_image());

        //상품 클릭 동작
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { listener.onItemClick(view, getAdapterPosition()); }
        });
    }
}
//=====뷰홀더 클래스 =====
```

이 동작은 뷰 홀더 클래스 동작입니다. 이를 통해 ViewHolder의 구조를 생성하고 아래 있는 onBind를 통해 실제로 View객체에 데이터 값을 적용시켜 사용자가 보는 ViewHolder를 만드는 것입니다.

이때 itemView의 Click 동작을 할 수 있습니다.

위 Click 동작을 통해 확인창으로 넘어가는 작업을 하는데 이는 아래에서 나타내겠습니다.



```
//===== 생성자 =====
// 생성자를 통해서 데이터를 전달받도록 함
public ProdAdapter (ArrayList<ProdData> dataSet) { prodDataSet = dataSet; }
//===== 생성자 =====
```

이를 통해 Main과 같이 다른 외부 클래스에서 ProdAdapter를 생성하였을 때 데이터의 값을 전달시킵니다.

```
//===== ViewHolder 생성 RecyclerView Adapter 필수 구현 항목 =====
@NonNull
@Override // ViewHolder 객체를 생성하여 리턴한다.
public ProdAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    mContext= parent.getContext();
    if( viewType == VERTICAL_LIST){
        View view = LayoutInflater.from(mContext)
            .inflate(R.layout.recyclerview_item, parent, attachToRoot false);
        ProdAdapter.ViewHolder viewHolder = new ProdAdapter.ViewHolder(view);
        return viewHolder;
    }
    else{
        View view = LayoutInflater.from(mContext)
            .inflate(R.layout.recyclerview_item2, parent, attachToRoot false);
        ProdAdapter.ViewHolder viewHolder = new ProdAdapter.ViewHolder(view);
        return viewHolder;
    }
}
//===== RecyclerView Adapter 필수 구현 항목 =====
```

위 코드로 ViewHolder를 생성합니다. 이때 ViewHolder를 생성하는데, ViewType이 무엇인지 동작을 Main.java에서 받아와 받은 값에 따라 view객체를 recyclerviewitem.xml로 정해야할지 recyclerviewitem2.xml로 할지정하고 이러한 ViewHolder를 생성합니다.

```
//=====앨범형 리스트형 변경 ViewType=====
@Override
public int getItemViewType(int position) { return mItemViewType; }

public void setItemViewType(int viewType) {
    this.mItemViewType = viewType;
    notifyDataSetChanged();
}
//=====앨범형 리스트형 변경 ViewType=====
```

View 객체를 변경하기 위해 추가하는 코드입니다. ViewType의 대하여 받아오는 setter와 getter입니다.

```
@Override
public void onBindViewHolder(@NonNull ProdAdapter.ViewHolder holder, int position) {
    // Item을 하나, 하나 집어넣어주는(bind 되는) 함수입니다.
    holder.onBind(prodDataSet.get(position));
}

void addItem(ProdData data) { prodDataSet.add(data); }

@Override // 전체 데이터의 갯수를 리턴한다.
public int getItemCount() { return prodDataSet.size(); }
```

onBindViewHolder는 생성한 ViewHolder에 실제로 데이터 값을 매칭해주는 것입니다. addItem()과 getItemCount()는 어댑터 동작에 있어서 도움이 되는 추가적인 요소들입니다. 값을 추가하는 메서드와 전체 데이터의 개수를 알 수 있는 메서드입니다.

## 마. ProdData.java

상품의 정보를 사용하기 위해 상품의 대한 정보에 여러 데이터들을 사용합니다. 이러한 데이터들을 모아서 처리해주는 java 코드입니다.

```
public class ProdData {  
    private String prod_name;    //상품명  
    private String prod_price;  //상품가격  
    private String prod_note;   //상품메모  
    private int prod_img;       //상품이미지  
    private int prod_star;      //상품 관심  
    private String prod_date;   //상품 등록 날짜  
    private Bitmap prod_bit_image;  
    private String prod_link;
```

우선 변수를 선언합니다. 각각 프로그램의 제어에 필요하거나 사용자에게 보여주는 데이터 값으로, 이는 추가되거나 변경될 여지가 있습니다.

```
public Bitmap getProd_bit_image() { return prod_bit_image; }  
  
public void setProd_bit_image(Bitmap prod_bit_image) { this.prod_bit_image = prod_bit_image; }  
  
public int getProd_star() { return prod_star; }  
  
public String getProd_link() { return prod_link; }  
  
public void setProd_link(String prod_link) { this.prod_link = prod_link; }  
  
public void setProd_star(int prod_star) { this.prod_star = prod_star; }  
  
public String getProd_date() { return prod_date; }  
  
public void setProd_date(String prod_date) { this.prod_date = prod_date; }  
  
public String getProd_price() { return prod_price; }  
  
public void setProd_price(String prod_price) { this.prod_price = prod_price; }  
  
public String getProd_name() { return prod_name; }  
  
public void setProd_name(String prod_name) { this.prod_name = prod_name; }  
  
public String getProd_note() { return prod_note; }  
  
public void setProd_note(String prod_note) { this.prod_note = prod_note; }  
  
public int getProd_img() { return prod_img; }  
  
public void setProd_img(int prod_img) { this.prod_img = prod_img; }
```

각 항목의 대한 getter, setter입니다. 이를 통해 리사이클러뷰 아이템에 따른 값을 받아오거나 또는 설정할 수 있습니다.

```
public ProdData(String prod_name, String prod_price, String prod_note,  
                String prod_link, int prod_star, String prod_date) {  
    this.prod_name = prod_name;  
    this.prod_price = prod_price;  
    this.prod_note = prod_note;  
    this.prod_link = prod_link;  
    this.prod_star = prod_star;  
    this.prod_date = prod_date;  
}
```

이는 Constructor로써 생성자입니다. 만약 ProdData의 형식으로 데이터를 생성하는 경우 위 파라미터를 바탕으로 ProdData 데이터 객체가 생성됩니다.

## 바. ActivityRegProd.java

등록하는 창과 연결되는 java입니다.

```
public class ActivityRegProd extends AppCompatActivity {
    // 아래는 사용 request code
    private static final int PROD_REG_CODE = 103;    //상품 등록 코드
    private static final int IMAGE_REQUEST_CODE = 111; //이미지 받아오기

    public static final int REG_CODE_FROM_MAIN = 101;    //메인 -> 등록창
    public static final int PROD_43REG = 105;

    ImageView imageView;
    byte[] byteArray;
    EditText prod_name;    // 상품 이름
    EditText prod_price;    // 가격
    EditText prod_link;    // 링크
    EditText prod_note;    // 메모를 입력하세요
    int prod_star;    // 평요도

    private DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
    private String result="";
}
```

사진, 이름, 가격, 링크, 메모 선언과 가격의 3자리마다 ,를 넣어주기 위해 DecimalFormat을 사용하였습니다.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.prod_activity_reg);
    setTitle("등록 추가");

    imageView = findViewById(R.id.prod_img);

    prod_price = (EditText) findViewById(R.id.prod_price);
    prod_price.addTextChangedListener(watcher);

    imageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setType("image/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(intent, IMAGE_REQUEST_CODE);
        }
    });
}
```

```
final CheckBox cb_star = (CheckBox)findViewById(R.id.prod_star);

cb_star.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean checked) {

        if(checked){
            prod_star = 1;
            Toast.makeText(getApplicationContext(), text "관심 설정", Toast.LENGTH_SH
        }else{
            prod_star = 0;
        }
    }
});
```

3자리 숫자마다 ,를 적기 위해서 addTextCangedListener(watcher)를 사용하였고 사진을 넣기 위해 이미지 부분을 클릭하면 갤러리 애플리케이션을 실행할 수 있도록 intent를 사용하였습니다.

또한 체크박스 선택 시 prod\_star = 1;을 관심 상품 구분을 했습니다.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    /*...*/
    //=====갤러리 동작=====
    if (requestCode == IMAGE_REQUEST_CODE) {    //사진받아오기
        if (resultCode == RESULT_OK) {
            try {
                InputStream in = getContentResolver().openInputStream(data.getData());

                Bitmap bitmap = BitmapFactory.decodeStream(in); //이미지 파일을 변환할 때는 BitmapFactory 클래스의 decodeStream 함수
                ByteArrayOutputStream stream = new ByteArrayOutputStream();
                bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 100, stream);
                byteArray = stream.toByteArray();

                imageView.setImageBitmap(bitmap);    // img 를 이미지뷰에 출력
                Toast.makeText(getApplicationContext(), text "사진 불러오기 성공", Toast.LENGTH_SHORT).show();    // 토스트 메시지 출력
            } catch (Exception e) {

            }
        } else if (resultCode == RESULT_CANCELED) {    // 뒤로가기로 작업 취소한때
            Toast.makeText( context this, text "사진 선택 취소", Toast.LENGTH_LONG).show();
        }
    }
    //=====갤러리 동작=====
}
```

사진을 이미지 뷰에 넣어주기 위한 작업입니다. 이전 코드에서 갤러리에 보낸 요청코드와 일치하면 동작하도록 되어있습니다. 이미지를 불러옴과 동시에 이미지를 문자열로 변환하기위한 작업을 같이 수행하기 위해 이미지 정보를 BitmapFactory.decodeStream을 사용하였고 이미지 파일을 변환한 후 ByteArray로 Bitmap 이미지 정보를 바꿨습니다. 사진을 불러온 후 완료 토스트 메시지를 출력하고 취소한 경우에는 취소 토스트 메시지가 출력되도록 하였습니다.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.end_btn: // 액션바 저장버튼 눌렀을때

            Intent intent = new Intent( packageContext, ActivityRegProd.this, MainActivity.class); // 보여지는 액티비티 자바

            prod_name = (EditText) findViewById(R.id.prod_name); // 상품 이름
            prod_price = (EditText) findViewById(R.id.prod_price); // 가격
            prod_link = (EditText) findViewById(R.id.prod_link); // 링크
            prod_note = (EditText) findViewById(R.id.prod_note); // 메모

            intent.putExtra( name: "name", prod_name.getText().toString()); // 키값은 name
            intent.putExtra( name: "price", prod_price.getText().toString());
            intent.putExtra( name: "link", prod_link.getText().toString());
            intent.putExtra( name: "note", prod_note.getText().toString());
            intent.putExtra( name: "star", prod_star);
            intent.putExtra( name: "image",byteArray);

            setResult(PROD_REG_CODE,intent); //Resultcode 설정
            Toast.makeText(getApplicationContext(), text: "저장완료하였습니다", Toast.LENGTH_SHORT).show();
            finish(); //등록창 -> 메인창

            return true;
        case R.id.return_btn: //취소버튼 동작
            Toast.makeText(getApplicationContext(), text: "취소합니다", Toast.LENGTH_SHORT).show();
            finish();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

가장 상단에 입력한 정보를 저장하기 위한 저장, 취소 버튼이 추가된 액션바를 만들었습니다. 저장버튼을 눌렀을 때 이름, 가격, 링크, 메모, 관심의 정보를 보내기 위해 문자열로 변환 후 intent를 사용하였으며, 사진을 보낼 때는 byteArray를 putExtra해줬습니다.

이때 Resultcode를 설정하여 finish()하였을 때, 메인액티비티에서 putExtra한 정보를 토대로 값을 추가하는 작업을 진행합니다.

취소버튼은 바로 finish() 동작합니다.

```
TextWatcher watcher = new TextWatcher() { // 글자 변경을 감지해줌 (3자리 숫자마다 , 나오기)
@Override
public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
}

@Override
public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    if(!TextUtils.isEmpty(charSequence.toString()) && !charSequence.toString().equals(result)){
        // EditText 가 비어있지 않을 때만 실행
        result = decimalFormat.format(Double.parseDouble(charSequence.toString().replaceAll( regex: " ", replacement: "")));
        prod_price.setText(result);
        prod_price.setSelection(result.length());
    }
}

@Override
public void afterTextChanged(Editable editable) { // 가격입력시 3자리 마다 , 를 생기기 위해 쓰는 함수
}
}
```

가격을 입력할 때 3자리 숫자마다 ,가 자동으로 입력되기 위한 코드 구성입니다. 다양한 숫자 데이터를 자신이 원하는 형식으로 나타낼 수 있도록 도와주는 DecimalFormat을 이용해서 매개변수로 들어가는 #의 수와 상관없이 format메서드에 들어가는 숫자형 변수의 자리에 맞춰서 출력해줍니다.



## 사. ActivityCheckProd.java

저장된 상품 정보를 확인하는 창과 연결되는 java 코드입니다.

```
public class ActivityCheckProd extends AppCompatActivity {
    public static final int PROD_DATA_CHANGE = 108; //상품 수정
    String Name;    //이름
    String Price;    //가격
    String Link;    //링크
    String Note;    //메모
    Button ok_btn;  //확인버튼
    int Star;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.prod_activity_check);

    //====어플리케이션 이름 숨기기====
    getSupportActionBar().hide();
    //====어플리케이션 이름 숨기기====

    //텍스트 작업
    TextView prod_name = (TextView) findViewById(R.id.prod_name);
    TextView prod_price = (TextView) findViewById(R.id.prod_price);
    TextView prod_link = (TextView) findViewById(R.id.prod_link);
    TextView prod_note = (TextView) findViewById(R.id.prod_note);

    Intent intent2 = getIntent();
    Name = intent2.getStringExtra( name: "name");
    Price = intent2.getStringExtra( name: "price");
    Link = intent2.getStringExtra( name: "link");
    Note = intent2.getStringExtra( name: "note");

    prod_name.setText(Name);           // 이름 출력
    prod_price.setText(Price);         // 가격 출력
    prod_link.setText(Link);           // 링크 출력
    prod_note.setText(Note);           // 텍스트 출력
```

```
//관심도 작업
ImageView prod_star = (ImageView) findViewById(R.id.prod_star);
Star = intent2.getIntExtra( name: "star", defaultValue: 0);
if(Star == 1){
    prod_star.setImageResource(R.drawable.full_star);
}else
    prod_star.setImageResource(R.drawable.empty_star);
```

```
//이미지 작업
byte[] arr = intent2.getByteArrayExtra( name: "image"); //비트맵이미지 받기
Bitmap image = BitmapFactory.decodeByteArray(arr, offset 0, arr.length); //바이트배열 -> 비트맵
ImageView prod_img = (ImageView) findViewById(R.id.prod_img);
prod_img.setImageBitmap(image); //비트맵으로 이미지 출력*/

//확인버튼 동작
ok_btn = findViewById(R.id.ok_btn);
ok_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { finish(); }
});
}
```

메인 화면에서 아이템 클릭 시 정보를 확인 할 수 있는 액티비티 코드입니다. RecyclerView Item에서 intent로 보낸 문자열 정보와 관심 표시를 위한 prod\_star값, Bitmap byte를 가져온 후 출력할 수 있도록 하였으며, 확인 버튼 클릭 시 동작을 마무리 합니다.

## 아. ActivityModProd.java

수정 창에서의 작업은 확인창과 등록 창에서의 작업을 합친 버전입니다.

확인 창(activity\_check)처럼 상품의 정보를 받아 이를 사용자에게 출력합니다. 이때 수정창은 상품의 이름을 제외하고 TextView가 아닌 EditText로 지정해줍니다.

```
prod_name = (TextView) findViewById(R.id.mod_name); // 상품 이름
prod_price = (EditText) findViewById(R.id.mod_price); // 가격
prod_link = (EditText) findViewById(R.id.mod_link); // 링크
prod_note = (EditText) findViewById(R.id.mod_note); // 메모
prod_star = (CheckBox) findViewById(R.id.mod_star);
imageView = (ImageView) findViewById(R.id.mod_img);
```

이후 등록 창(activity\_reg)과 동일하게 이미지 클릭 시 이미지 가져오기, EditText값을 읽어 상품 정보로 등록하여 MainActivity로 보내는 작업을 합니다.

이때 기존의 사진에서 변경이 된 경우를 구분하기 위해 이미지 변경이 된 것을 구분하는

```
change_img = 1; //이미지 변경

imageView.setImageBitmap(bitmap);
```

change\_img 변수를 사용하여 갤러리에서 새로운 값을 가져오면 1로 설정합니다.

```
if(change_img==1) //이미지 변경 시
    intent1.putExtra( name: "image", change_image_arr);
else
    intent1.putExtra( name: "image", get_image_arr);
setResult( PROD_DATA_CHANGE, intent1);
```

이후 경우에 따라 변경 시에는 새로 받은 이미지 값, 변경이 안됐으면 기존 이미지 값을 반환합니다.

```
//=====취소=====
btn_cancel = (Button) findViewById(R.id.btn_cancel);
btn_cancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(getApplicationContext(), text: "수정 취소", Toast.LENGTH_SHORT).show();
        finish();
    }
});
//=====취소=====
```

//수정취소 시

```
//=====수정완료=====
btn_ok = (Button) findViewById(R.id.btn_ok);
btn_ok.setOnClickListener(new View.OnClickListener() {...});
//=====수정완료=====
```

//수정 시

```
Intent intent1 = new Intent( packageContext: ActivityModProd.this, ActivityCheckProd.class);
```

이때 상품수정의 정보는 메인 -> 확인 -> 수정 -> 메인의 순서로 가지 않습니다.

중간에 수정을 거쳐 메인 -> 확인 -> 수정 -> 확인 -> 메인의 순서로 확인 창을 한 번 들립니다. 이렇게 하지 않으면 중간에 값이 사라지는 문제가 발견되었습니다.

## 자. NotificationHelper.java

```
public class NotificationHelper extends ContextWrapper {

    public static final String channelId = "channel1ID";
    public static final String channelName = "channel 1";

    private NotificationManager mManager;

    public NotificationHelper(Context base) {
        super(base);

        //오레오보다 같거나 크면
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            createChannels();
        }
    }
}
```

```
//채널 생성
@RequiresApi(api = Build.VERSION_CODES.O)
public void createChannels(){
    NotificationChannel channel1 = new NotificationChannel(channelId, channelName, NotificationManager.IMPORTANCE_DEFAULT);
    channel1.enableLights(true);
    channel1.enableVibration(true);
    channel1.setLightColor(R.color.purple_700);
    channel1.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    getManager().createNotificationChannel(channel1);
}

public NotificationManager getManager(){
    if(mManager == null){
        mManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    }

    return mManager;
}

public NotificationCompat.Builder getChannel1Notification(String title, String message){

    return new NotificationCompat.Builder(getApplicationContext(), channelId)
        .setContentTitle(title)
        .setContentText(message)
        .setSmallIcon(R.mipmap.ic_andro_foreground);
}
}
```

푸시알림을 구현하기 위한 Class입니다. 푸시알림에서는 Notification을 사용하는데, Android O(API 26)부터는 Notification을 사용하려면 Channel을 먼저 생성해주어야 합니다. channel1이라는 이름을 가진 채널을 생성합니다. 채널을 생성하고, 해당 채널로 Notification을 생성합니다. setContentTitle(), setContentText(), setSmallIcon()으로 제목 텍스트, 본문 텍스트, 알림 아이콘 등을 설정해주었습니다.

## 차. ProdDB.java

### <데이터 베이스 선언과 테이블 생성>

```
public ProdDB(Context context){
    super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
    this.context = context;
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE prod_List ( " +
        "prod_name TEXT, " +
        "prod_price TEXT, " +
        "prod_link TEXT, " +
        "prod_note TEXT, " +
        "prod_star INTEGER, " +
        "prod_date TEXT, " +
        "prod_img BLOB);");
}
```

SQLite의 선언과 테이블을 생성해 줍니다. 각 항목에는 이름, 가격, 링크, 메모, 관심등록 유무, 날짜, 사진이 들어 갑니다.

## 카. 기타 작업들

### <상품 정보 클릭 시 ActivityCheckProd 이동>

```
//어댑터 생성
prodAdapter = new ProdAdapter(getApplicationContext(), prodDataSet, db, new ProdAdapter()
@Override
//=====상품 클릭시=====
public void onItemClick(View v, int position) {
    Intent intent2 = new Intent( getApplicationContext(),ActivityCheckProd.class);

    //이미지 가져오기
    Bitmap prod_Bitmap = prodDataSet.get(position).getProd_bit_image();

    if(prod_Bitmap ==null){
        byte[] byteArray = null;
        intent2.putExtra( name: "image",byteArray);
    }else {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        prod_Bitmap.compress(Bitmap.CompressFormat.JPEG, quality:100, stream);
        byte[] byteArray = stream.toByteArray();
        intent2.putExtra( name: "image",byteArray);
    }

    String prod_name = prodDataSet.get(position).getProd_name();
    String prod_price = prodDataSet.get(position).getProd_price();
    String prod_note = prodDataSet.get(position).getProd_note();

    //이름,가격,메모,링크,이미지,관심,날짜,포지션
    intent2.putExtra( name: "name",prod_name);
    intent2.putExtra( name: "price",prod_price);
    intent2.putExtra( name: "note", prod_note);

    String prod_link1 = prodDataSet.get(position).getProd_link();
    intent2.putExtra( name: "link",prod_link1);

    int prod_star1 = prodDataSet.get(position).getProd_star();
    intent2.putExtra( name: "star",prod_star1);

    String prod_date1 = prodDataSet.get(position).getProd_date();
    intent2.putExtra( name: "date",prod_date1);

    //intent2.putExtra("image",byteArray);

    intent2.putExtra( name: "position",position);

    startActivityForResult(intent2,CHECK_PRODD);

    =====상품 클릭시=====
}
```

Main.java에서 어댑터를 생성할 때 하는 작업입니다.

RecyclerView의 Item 클릭 시 해당 아이템의 포지션 값과 상품 정보를 가져옵니다.

이후 상품정보를 Intent를 생성하여 putExtra로 보냅니다. 이때 상품정보는 ProdData의 getter를 이용하여 가져옵니다.

이후 이것을 메인 -> 확인 창으로 넘겨주는 작업을 했습니다.



## <Main 액티비티 StartActivityResult 정보 받기 동작>

### <상품 등록>

```
//=====Intent 응답 =====
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == PROD_REG_CODE) { //등록코드가 resultCode시 동작
        // Toast.makeText(getApplicationContext(), "등록 완료", Toast.LENGTH_LONG).show(); //확인 메세지

        byte[] arr = data.getByteArrayExtra( name: "image"); //비트맵이미지 받기
        Bitmap image = BitmapFactory.decodeByteArray(arr, 0, arr.length); //바이트배열 -> 비트맵
        ImageView prod_img = (ImageView)findViewById(R.id.prod_img);
        prod_img.setImageBitmap(image); //비트맵으로 이미지 출력*/

        Name = data.getStringExtra( name: "name"); //상품 이름 받기
        Price = data.getStringExtra( name: "price");//상품 가격 받기
        Link = data.getStringExtra( name: "link");//상품 링크 받기
        Text = data.getStringExtra( name: "note");//상품 메모 받기
        Star = data.getIntExtra( name: "star", defaultValue: 0);

        ProdData mainData = new ProdData(Name,Price,Text,Link //ProdData 데이터 설정
            ,Star, prod_date: "220204");
        mainData.setProd_bit_image(image); //bitmap이미지 저장

        prodDataSet.add(mainData); //데이터 설정한 dataSet을 ProdData 리스트에 추가
        prodAdapter.notifyItemInserted(prodDataSet.size()); //어댑터의 Data추가
        prodAdapter.notifyDataSetChanged(); //새로고침
    }
}
```

상품 등록창에서 상품을 등록할 때 ResultCode로 설정한 code가 들어왔을 경우 각각의 데이터 값을 받아 ProdData 타입의 dataSet의 데이터를 설정합니다. 설정한 데이터를 상품 데이터 List의 추가하고 어댑터의 새롭게 추가한 Data를 어댑터의 추가하는 작업을 합니다.

### <수정 및 삭제>

수정 및 삭제는 위처럼 onActivityResult를 통해 받습니다.

두 동작 모두 확인 창에서 이루어지기 때문에 else if를 통해 requestCode를 비교하여 확인 창에서 온 작업인지 확인합니다. 이후 resultCode를 통해 삭제 작업인지 수정 작업인지 확인합니다.

```
else if(requestCode == CHECK_PROD){ //확인창 req code
    if(resultCode == PROD_DATA_DELETE){...} //삭제

    else if(resultCode == PROD_DATA_CHANGE){ //수정
    }
}
```

### - 수정

#### <ActivityCheckProd.java>

```
//=====수정=====
modify_btn = findViewById(R.id.btn_modify);
modify_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(),ActivityModProd.class);
        //이름, 가격, 메모, 링크, 이미지, 관심, 날짜, 포지션
        intent.putExtra( name: "position",Pos);
        intent.putExtra( name: "name",Name);
        intent.putExtra( name: "price",Price);
        intent.putExtra( name: "note",Note);
        intent.putExtra( name: "link",Link);
        intent.putExtra( name: "date",Date);
        intent.putExtra( name: "star",Star);
        intent.putExtra( name: "image",arr);
        startActivityForResult(intent, PROD_DATA_CHANGE);
    }
}
```

Main에서부터 상품의 대한 정보를 모두 받아와 수정하기 버튼 클릭 시 수정창 액티비티로 넘어갑니다.

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == PROD_DATA_CHANGE) { //수정완료 시 지나침 Mod -> Check -> Main
        //이름, 가격, 링크, 메모, 관심, 날짜, 이미지, 포지션
        Name = data.getStringExtra( name: "name");
        Price = data.getStringExtra( name: "price");
        Link = data.getStringExtra( name: "link");
        Note = data.getStringExtra( name: "note");
        image_arr = data.getByteArrayExtra( name: "image");
        Pos = data.getIntExtra( name: "position", defaultValue: 0);
        Date = data.getStringExtra( name: "date");
        Star = data.getIntExtra( name: "star", defaultValue: 0);

        Intent intent = new Intent(getApplicationContext(), MainActivity.class); //Check -> Main
        //이름, 가격, 링크, 메모, 관심, 날짜, 이미지, 포지션
        intent.putExtra( name: "name", Name); // 키값은 name
        intent.putExtra( name: "price", Price);
        intent.putExtra( name: "link", Link);
        intent.putExtra( name: "note", Note);
        intent.putExtra( name: "position", Pos);
        intent.putExtra( name: "star", Star);
        intent.putExtra( name: "image", image_arr);
        intent.putExtra( name: "date", Date);
        setResult(PROD_DATA_CHANGE, intent);

        finish();
    }
}

```

해당 작업은 수정하였을 때 수정 -> 확인 -> 메인으로 잠시 거쳐 가는 코드이며 해당 내용은 문제점 및 해결방안에서 다룰 내용입니다.

### <ActivityModProd.java>

```

Intent intent = getIntent();

//이름, 가격, 링크, 메모, 관심, 날짜, 이미지, 포지션
String Name = intent.getStringExtra( name: "name");
String Price = intent.getStringExtra( name: "price");
String Link = intent.getStringExtra( name: "link");
String Note = intent.getStringExtra( name: "note");
Pos = intent.getIntExtra( name: "position", defaultValue: 0);
Star = intent.getIntExtra( name: "star", defaultValue: 0);
Date = intent.getStringExtra( name: "date");

final CheckBox cb_star = (CheckBox) findViewById(R.id.mod_star);
//관심인터페이스
if(Star == 1) //관심 상종아면
    cb_star.setChecked(true); //체크 상태
else
    cb_star.setChecked(false); //un 체크 상태

cb_star.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean checked) {
        if(checked){ //상종 체크 시
            Star = 1;
            Toast.makeText(getApplicationContext(), text: "관심 설정\n", Toast.LENGTH_SHORT).show(); // 토스트 메시지 출력
        }else{
            Star = 0;
        }
    }
});

```

수정 창에서는 우선 getIntent를 통해 값들을 받아줍니다. 이때 각 데이터 형식대로 값을 받고, 체크박스는 값이 1인지 0인지에 따라 체크박스의 표기 상태를 true 또는 false로 출력합니다.  
이후 아래에서 체크박스의 체크 동작을 추가해줍니다.

```
String Name = prod_name.getText().toString();
String Price = prod_price.getText().toString();
String Link = prod_link.getText().toString();
String Note = prod_note.getText().toString();

Intent intent1 = new Intent( packageContext ActivityModProd.this, ActivityCheckProd.class); //Mod -> Check
//이름, 가격, 링크, 메모, 관심, 날짜, 이미지, 포지션
intent1.putExtra( name: "name", Name); // 키값은 name
intent1.putExtra( name: "position", Pos);
intent1.putExtra( name: "price", Price);
intent1.putExtra( name: "link", Link);
intent1.putExtra( name: "note", Note);
intent1.putExtra( name: "star", Star);
intent1.putExtra( name: "date", Date);
if(change_img==1) //이미지 변경 시
    intent1.putExtra( name: "image", change_image_arr);
else
    intent1.putExtra( name: "image", get_image_arr);
setResult( PROD_DATA_CHANGE , intent1);
```

다음으로는 수정한 값을 넘겨줍니다. 이때 넘기는 곳은 Main이 아닌 Check로 넘겨줍니다.

### <MainActivity.java>

메인에서는 수정 -> 확인 -> 메인을 통해 코드를 받았을 시 해당 상품의 포지션, 이미지, 해당 내용들을 전부 받습니다. 이후 받은 데이터로 새로운 데이터를 생성합니다.

```
Name = data.getStringExtra( name: "name"); //상품 이름 받기
Price = data.getStringExtra( name: "price");//상품 가격 받기
Link = data.getStringExtra( name: "link");//상품 링크 받기
Text = data.getStringExtra( name: "note");//상품 메모 받기
Star = data.getIntExtra( name: "star", defaultValue: 0); //관심 날짜
Date = data.getStringExtra( name: "date");

Toast.makeText(getApplicationContext(), text: "수정 완료", Toast.LENGTH_SHORT).show();

//데이터 생성
ProdData modData = new ProdData(Name, Price, Link, Text, Star, Date);
modData.setProd_bit_image(image);
```

이렇게 생성한 데이터를 ArrayList 메서드를 사용하여 해당 위치의 값을 변경합니다.

```
prodDataSet.set(pos, modData); //현재 위
prodAdapter.notifyDataSetChanged();
```

## - 삭제

### <ActivityCheckProd.java>

```
//=====삭제=====
delete_btn = findViewById(R.id.btn_delete);
delete_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent( packageContext ActivityCheckProd.this, MainActivity.class);
        intent.putExtra( name: "position", Pos);
        setResult(PROD_DATA_DELETE, intent);
        finish();
    }
});
```

삭제 시 동작은 Check창에서 삭제 버튼 클릭 시 해당 포지션 값을 받아옵니다.

### <MainActivity.java>

이후 메인에서는 resultCode를 통해 삭제 동작이 들어오면 ArrayList의 메서드를 통해 리스트에서 해당 위치를 삭제합니다.

```
prodDataSet.remove(pos); //리스트 현재 위치 삭제
prodAdapter.notifyDataSetChanged();
```

## <상품 등록 날짜 데이터>

### <ActivityRegProd.java>

```
//날짜관련
long mNow;
Date mDate;
SimpleDateFormat mFormat = new SimpleDateFormat( pattern: "yyMMdd");
//날짜관련

String prod_date1; //날짜
```

상품을 등록하면 순서 정렬을 위한 날짜 데이터가 존재합니다. 이러한 상품 등록 날짜를 받기 위해 상품을 등록하는 창에서 날짜 데이터를 생성합니다. 이때 날짜 형식은 yyMMdd(=20yy년 MM월 dd일)의 형식으로 받습니다. 클래스 하단에 getTime() 메서드를 생성합니다.

```
//현재 시각
private String getTime(){
    mNow = System.currentTimeMillis();
    mDate = new Date(mNow);
    return mFormat.format(mDate);
}
```

이는 시스템으로부터 현재 시간을 받아서 위 설정한 DateFormat의 형식으로 현재 시간을 반환하는 메서드입니다.

```
case R.id.end_btn:// 액션바 저장버튼 눌렀을때

    prod_date1 = getTime(); //현재 날짜 yyymmdd 꼴 intent.putExtra( name: "date",prod_date1);
```

상품 저장 클릭 시 다음과 같이 prod\_date1 변수의 저장하여 이를 putExtra를 통해 날짜 값을 전달합니다.

### <ActivityCheckProd.java>

상품날짜의 형식이 yyMMdd의 Format으로 저장되어 있기 때문에 사용자가 정보를 확인하는 Check 액티비티에서는 다음과 같이 String 메서드를 사용하여 날짜 형식을 바꿔줍니다.

```
Date_p = "20"+Date.substring(0,2)+"."+Date.substring(2,4)+"."+Date.substring(4,Date.length());
//20yy.MM.dd 형태로 출력(print)
```



2022.05.03

그럼 이와 같은 형태의 날짜 데이터로 표시가 됩니다.

## <푸시알림 동작>

### <ActivityCheckProd.java>

```
//푸시알림 관련
private NotificationHelper notificationHelper;
private Button btn_alarm;
RadioGroup rPush;
RadioButton sec10,day7,day14;
int push_msec;
int push_ctr;
String push_msg;
```

푸시알림 관련 변수를 선언합니다.



```

rPush = (RadioGroup) findViewById(R.id.rPush);
sec10 = (RadioButton) findViewById(R.id.sec10);
day7 = (RadioButton) findViewById(R.id.day7);
day14 = (RadioButton) findViewById(R.id.day14);
String today = getTime(); //현재 시간
btn_alarm = findViewById(R.id.btn_alarm);
notificationHelper = new NotificationHelper( base: this);
String date_alarm = Date.substring(2,4)+"월"+Date.substring(4,Date.length())+"일"; //MM월dd일 형태
btn_alarm.setOnClickListener(new View.OnClickListener() {

```

```

@Override
public void onClick(View view) {
    switch (rPush.getCheckedRadioButtonId()) {
        case R.id.sec10:
            push_msec = 10000;
            push_msg = "10초";
            push_ctr = 0;
            break;
        case R.id.day7:
            push_msec = 604800000; //7일 1000(1초)*60(1분)*60(1시)*24(1일)*7(7일)
            push_msg = "7일";
            push_ctr = 0;
            break;
        case R.id.day14:
            push_msec = 1209600000;
            push_msg = "14일";
            push_ctr = 0;
            break;
        default:
            push_msec = -1;
            Toast.makeText(getApplicationContext(), text: "푸시 알림 시간을 선택해주세요", Toast.LENGTH_SHORT).show();
    }
}

```

```

if (push_msec != -1&&push_ctr!=1) {
    Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            sendOnChannel1(
                title: Name + "구매하셨나요?",
                message: "상품 담은 " + date_alarm + "로 부터 " + 7 + "일 지났습니다!!");
        }
    }, push_msec); //라디오버튼만큼 delay
    push_ctr = 1; //같은 창에서 여러번 클릭 시에도 한번만 올리게 함
    Toast.makeText(getApplicationContext(), text: push_msg + " 뒤에 알림이 갑니다", Toast.LENGTH_LONG).show();
}
}

```

라디오버튼으로 10초 뒤, 7일 뒤, 14일 뒤에 푸시알림을 전송하는 버튼을 구성하였습니다. 10초후 버튼은 시연을 위해 임시로 만들어 둔 버튼입니다. switch-case문을 이용하여 해당 라디오 버튼이 눌릴 때마다 정해진 동작을 하도록 하였습니다. 버튼을 누른 후에 실행되는 동작을 지연시키는 함수를 추가하여 일정 시간이 지난 뒤에 해당 기능이 동작하도록 구현하였습니다. default:문은 ActivityCheckProd 화면에서 푸시알림을 따로 설정하지 않고 넘어갔을 때 어플이 튕기는 것을 방지하기 위해 아무것도 선택하지 않았더라도 그대로 에러없이 넘어갈 수 있도록 하기 위해 작성 하였습니다. Handler()를 통해 설정한 push\_msec의 시간이 지난 후에 run() 함수가 동작하도록 하였습니다. 또한, push\_ctr를 통해 버튼이 여러 번 눌리더라도 알림은 한 번만 동작할 수 있도록 하였습니다. 알림이 설정되었을 때 설정되었다는 알림이 토스트 메시지를 통해서도 전달됩니다.

## <데이터베이스>

### <상품 등록한 정보 데이터베이스에 저장하기>

```
Name = data.getStringExtra( name: "name");           //상품이름 받기
Price = data.getStringExtra( name: "price");          //가격
Link = data.getStringExtra( name: "link");            //링크
Note = data.getStringExtra( name: "note");            //메모
Star = data.getIntExtra( name: "star", defaultValue: 0); //관심
Date = data.getStringExtra( name: "date");            //날짜

ProdData regData = new ProdData(Name,Price,Link,Note //data 데이터
                                ,Star,Date);
regData.setProd_bit_image(image); //bitmap이미지 저장

sqlDB = db.getWritableDatabase();
SQLiteStatement p = sqlDB.compileStatement( sql: "INSERT INTO prod_List VALUES(?,?,?,?,?,?,?);");
p.bindString( index: 1,regData.getProd_name());
p.bindString( index: 2,regData.getProd_price());
p.bindString( index: 3,regData.getProd_link());
p.bindString( index: 4,regData.getProd_note());
p.bindLong( index: 5,regData.getProd_star());
p.bindString( index: 6,regData.getProd_date());
if(Image_byte != null)p.bindBlob( index: 7,Image_byte);
p.execute();
```

상품 등록창에서 저장한 정보를 받아와서 INSERT INTO를 통해 prod\_list에 추가합니다. 이름, 가격, 링크, 메모, 관심등록, 날짜, 이미지 순서로 데이터베이스에 저장합니다.

## <getData()>

### <데이터베이스 값 가져오기>

```
private void getData() {

    cursor = sqlDB.rawQuery( sql: "SELECT * FROM prod_List;", selectionArgs: null);

    while (cursor.moveToNext()) {
        ProdData data = new ProdData( prod_name: null, prod_price: null, prod_link: null, prod_note: null, prod_star: -1, prod_date: null);
        data.setProd_name(cursor.getString( columnIndex: 0));
        data.setProd_price(cursor.getString( columnIndex: 1));
        data.setProd_link(cursor.getString( columnIndex: 2));
        data.setProd_note(cursor.getString( columnIndex: 3));
        data.setProd_star(cursor.getInt( columnIndex: 4));
        data.setProd_date(cursor.getString( columnIndex: 5));
        byte[] image = cursor.getBlob( columnIndex: 6);
        if(image != null) {
            Bitmap bm = BitmapFactory.decodeByteArray(image, offset: 0, image.length);
            data.setProd_bit_image(bm);
        }
        else {
            data.setProd_bit_image(null);
        }
        prodDataSet.add(data);
    }

    //adapter에 값이 변경되었다는 것을 알려줍니다.
    prodAdapter.notifyDataSetChanged(); //새로고침
}
```

이후 getData() 메서드에서 데이터 베이스에서 가져온 데이터를 처리하기 위해서 Cursor라는 인터페이스를 사용하였습니다. 데이터 베이스에 저장된 정보를 받을 ProdData data는 null과 -1로 초기화하였고 데이터 베이스에 저장되어 있는 각 데이터 형식에따라 String, Int, Blob(바이트)의 형태를 가져와서 데이터 리스트를 생성하여 어댑터에 추가해줍니다. 이때 데이터 리스트는 이름, 가격, 링크, 메모, 관심정보, 날짜, 사진 데이터를 ProdData data에 저장하였습니다.

## <getData() 초기 데이터 생성>

```
private void getData() {
    // 임의의 데이터입니다.
    sqlDB = db.getReadableDatabase();
    Cursor cursor;
    cursor = sqlDB.rawQuery( sql: "SELECT * FROM prod_List;", selectionArgs: null);
    while (cursor.moveToNext()) {
        tmp = cursor.getString(10);
    }
    if (tmp == null) {
        data_get.setVisibility(View.VISIBLE);
        data_get.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                getInitialData();
            }
        });
    }
}
```



우선 데이터베이스를 조회하여 존재하는 데이터가 없다면 상품의 확인을 위한 초기 데이터 생성 버튼을 보여주고 클릭 시 getInitialData() 메서드를 호출합니다.

```
public void getInitialData(){
    List<String> listName = Arrays.asList(
        "ARCH TEE",
        "시그니처 로고 반팔티",
        "미니멀프로젝트 로버스트 헤비 오버핏 반팔티셔츠",
        "Durgod 토러스 K320KR 텐키리스 투톤 키보드",
        "바밀로 고래 VA87M 텐키리스 키보드",
        "콕스 COX CK87 게이트론 텐키리스 키보드",
        "뉴발란스 CM878MC1",
        "나이키 에어 포스107"
    );

    List<String> listPrice = Arrays.asList(
```

```
for (int i = 0; i < listName.size(); i++) {
    // 각 List의 값들을 data 객체에 set 해줍니다.
    Bitmap get_prod_image = BitmapFactory.decodeResource(getApplication().getResources(), list
    Image_byte = bitmapToByteArray(get_prod_image);
    ProdData data = new ProdData(listName.get(i), listPrice.get(i), listlink.get(i), listNote.get(i),
    data.setProd_bit_image(get_prod_image);

    sqlDB = db.getWritableDatabase();
    SQLiteStatement p = sqlDB.compileStatement( sql: "INSERT INTO prod_List VALUES(?,?,?,?,?,?,?);");
    p.bindString( index 1, listName.get(i));
    p.bindString( index 2, listPrice.get(i));
    p.bindString( index 3, listlink.get(i));
    p.bindString( index 4, listNote.get(i));
    p.bindLong( index 5, listStar.get(i));
    p.bindString( index 6, listDate.get(i));
    p.bindBlob( index 7, Image_byte);
    p.execute();
}
```

```
restart(getApplicationContext());
```

이 메서드에서는 프로그램의 진행을 확인하기 위해 일종의 데이터를 미리 저장해놨습니다. 각각의 List를 생성하여 저장하고, 데이터 베이스에도 저장을 해줍니다. 이후 restart() 메서드를 생성하여 재시작합니다.

## <restart() 재시작 메서드>

```
private void restart(Context context) {
    PackageManager packageManager = context.getPackageManager();
    Intent intent = packageManager.getLaunchIntentForPackage(context.getPackageName());
    ComponentName componentName = intent.getComponent();
    Intent mainIntent = Intent.makeRestartActivityTask(componentName);
    context.startActivity(mainIntent);
    Runtime.getRuntime().exit( status: 0);
}
```

프로그램을 재시작하기 위해 만들어준 함수입니다.



### <getData() 데이터베이스 값 가져오기>

```
private void getData() {  
    cursor = sqldb.rawQuery( sql: "SELECT * FROM prod_List;", selectionArgs: null);  
  
    while (cursor.moveToNext()) {  
        ProdData data = new ProdData( prod_name: null, prod_price: null, prod_link: null, prod_note: null, prod_star: -1,  
        data.setProd_name(cursor.getString( 0));  
        data.setProd_price(cursor.getString( 1));  
        data.setProd_link(cursor.getString( 2));  
        data.setProd_note(cursor.getString( 3));  
        data.setProd_star(cursor.getInt( 4));  
        data.setProd_date(cursor.getString( 5));  
        byte[] image = cursor.getBlob( 6);  
        if (image != null) {  
            Bitmap bm = BitmapFactory.decodeByteArray(image, offset: 0, image.length);  
            data.setProd_bit_image(bm);  
        } else {  
            data.setProd_bit_image(null);  
        }  
        prodDataSet.add(data);  
    }  
}
```

getData 메서드에서는 데이터베이스를 조회하여 테이블에 존재하는 모든 값을 가져옵니다. 이때 커서를 이용하여 각 테이블의 열 값을 가져오는데, 각 데이터 형식에 따라 String, Int, Blob(바이트)의 형태를 가져와서 데이터 리스트를 생성하여 어댑터에 추가해줍니다.

### <데이터 베이스 정보 삭제 동작>

```
else if(requestCode == CHECK_PROD){ //확인창 req code  
    if(resultCode == PROD_DATA_DELETE){ //삭제  
  
        sqldb.execSQL("DELETE FROM prod_List WHERE prod_name = '"  
            + prodDataSet.get(pos).getProd_name() + "';");  
        prodDataSet.remove(pos); //리스트 현재 위치 삭제  
        prodAdapter.notifyDataSetChanged();  
    } //삭제
```

상품 정보를 확인하는 창에서 삭제 버튼 클릭시 데이터 베이스에 있는 정보도 함께 삭제되도록 구현하였습니다. 테이블 정보 중에서 삭제 기능이 동작하는 상품 정보의 이름과 같은 테이블이 삭제되도록 작성하였습니다.

### <데이터 베이스 정보 수정 동작>

```
prodDataSet.set(pos, modData); //현재 위치에 데이터 변경  
prodAdapter.notifyDataSetChanged();  
  
Toast.makeText(getApplicationContext(), text: "수정 완료", Toast.LENGTH_SHORT).show();  
sqldb.execSQL("UPDATE prod_List "+"SET prod_price = " + Price +  
    " WHERE prod_name = '" + Name + "';");  
sqldb.execSQL("UPDATE prod_List "+"SET prod_link = " + Link +  
    " WHERE prod_name = '" + Name + "';");  
sqldb.execSQL("UPDATE prod_List "+"SET prod_note = " + Note +  
    " WHERE prod_name = '" + Name + "';");  
sqldb.execSQL("UPDATE prod_List "+"SET prod_star = " + Star +  
    " WHERE prod_name = '" + Name + "';");  
if(Image_byte != null) {  
    SQLiteStatement p = sqldb.compileStatement( sql: "UPDATE prod_List SET prod_img = ? WHERE prod_name = '"  
        + Name + "';");  
    p.bindBlob( index: 1, Image_byte);  
    p.execute();  
}
```

상품 확인창에서 수정 버튼이 눌리고 난 후 수정한 상품 정보들이 각각의 테이블 안에 새로운 정보를 넣어주도록 하는 작성하였습니다. 상품의 이름은 고정으로 하고 가격, 링크, 메모, 관심등록이 변경될 수 있도록 작성하였습니다.



<상품 추가 후 데이터 베이스 저장 화면>

6:07 3G

목록 추가 취소 저장



그리디파머스 맥커브 태블릿 거치대

21,500 원

http://naver.me/x3b1ZFYo

높이조절 2단 / 각도조절 2단



6:07 3G


+ ⋮

검색어를 입력하십시오

AMD


뉴발런스 신발, 회색

가격: 129,000원



나이키 에어 포스 신발, 흰색

가격: 119,000원



그리디파머스 맥커브 높이조절 2단 / 각도조절 2단

가격: 21,500원



DB Browser for SQLite - C:\Users\Kown Youngin\Desktop\prodList.db

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W) 변경사항 취소하기(R) 프로젝트 열기(P)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): prod\_List

	prod_name	prod_price	prod_link	prod_note	prod_star	prod_date	prod_img
	필터	필터	필터	필터	필터	필터	필터
1	그리디파머스 맥커브...	21,500	http://naver.me/x3b1ZFYo	높이조절 2단 / 각도조절 2단	0	220602	[B@2f152c2

상품 등록 후 DB Browser for SQLite를 통해 상품 정보가 데이터 베이스에 저장됨을 확인할 수 있습니다.

## <설명창>

```
ImageButton manual = (ImageButton) findViewById(R.id.manual);
manual.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        View dialogView = (View) View.inflate(context: MainActivity.this, R.layout.dialog, root: null);
        AlertDialog.Builder dlg = new AlertDialog.Builder(context: MainActivity.this);
        ImageView ivPoster = (ImageView) dialogView.findViewById(R.id.ivPoster);
        ivPoster.setImageResource(R.drawable.add1);
        dlg.setView(dialogView);
        dlg.setIcon(R.mipmap.ic_andro_round);
        dlg.setNegativeButton(text: "cancel", listener: null);
        dlg.setPositiveButton(text: "next", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
    View dialogView = (View) View.inflate(context: MainActivity.this, R.layout.dialog, root: null);
    AlertDialog.Builder dlg = new AlertDialog.Builder(context: MainActivity.this);
    ImageView ivPoster = (ImageView) dialogView.findViewById(R.id.ivPoster);
    ivPoster.setImageResource(R.drawable.add3);
    dlg.setView(dialogView);
    dlg.setIcon(R.mipmap.ic_andro_round);
    dlg.setNegativeButton(text: "cancel", listener: null);
    dlg.show();
}
}
```

물음표 모양의 버튼 클릭 시 dialog를 사용한 설명창이 나오도록 작성하였습니다. 대화상자 화면은 dialog.xml를 사용하였고 각각 화면은 add1, add2, add3으로 사진을 넣었습니다. 첫 번째와 두 번째 화면은 next, cancel버튼이 나오고 세 번째 화면은 cancel이 동작하는 버튼을 추가하고 마지막에는 dlg.show()를 통해 화면에 출력되도록 하였습니다.

## 6.2 제작시 문제점 및 개선사항

### <이미지 Bitmap 통일 및 Intent 문제>

사용자가 상품 등록 정보를 저장할 때 이미지를 받아왔습니다. 이때 코드를 합치기 전에는 R.drawable폴더에 있는 이미지 파일을 통해 코드 구현을 하였는데, 갤러리를 통해 이미지를 받아오는 형태는 Bitmap의 데이터였고, drawable은 int로 값을 저장했습니다. 이를 개선하기 위해 기존에 있던 int로 선언된 prod\_img를 prod\_bit\_img로 변경했으며, 이와 맞게 코드를 조작했습니다.

또한 Intent로 Bitmap을 넘길 때 Bitmap => 문자열로 변경하는 코드를 사용했습니다. 그러나 문자열이 Intent되는 과정에서 앱이 종료되는 문제가 발생하여 String의 문자열이 아닌 byte[]의 바이트 배열을 사용하여 Intent를 시켜줬습니다.

### <가격순 정렬 시 숫자 문자열 특수문자 문제>

사용자가 메뉴를 동작할 시 가격순 정렬이 있는데, 숫자를 1000 => 1,000 꼴로 나타내니 ,을 parseInt하는데 문제가 있었습니다. 이를 해결하기 위해 replaceAll을 사용해줬습니다.

```
/*=====특수문자 제거 함수=====*/
public static String StringReplace(String str){
    String match = "[^\\uAC00-\\uD7A30-9a-zA-Z]";
    str = str.replaceAll(match, replacement: "");
    return str;
}
/*=====특수문자 제거 함수=====*/
```

이를 통해 특수 문자를 제거하고 특수문자를 공백으로 둡니다.

```
if (Integer.parseInt(StringReplace(item1.getProd_price())) < Integer.parseInt(StringReplace(item2.getProd_price()))
    ret = -1 ;
else if (Integer.parseInt(StringReplace(item1.getProd_price())) == Integer.parseInt(StringReplace(item2.getProd_price()))
    ret = 0 ;
else
    ret = 1 ;
```

이후 숫자를 비교하는 코드에서 parseInt를 하기 전, StringReplace를 통해 문자열만 가져옵니다.

## <검색창 중복 출력 문제>

검색창의 상품을 입력할 때 동작은 다음과 같습니다.

```
/*상품이름으로 검색*/
for (int i = 0; i < prodDataSet.size(); i++) {
    if (prodDataSet.get(i).getProd_name().toLowerCase().contains(searchText.toLowerCase())) {
        filteredList.add(prodDataSet.get(i));
        hs.add(filteredList);
    }
}
/*가격으로 검색*/
for (int i = 0; i < prodDataSet.size(); i++) {...}
/*메모로 검색*/
for (int i = 0; i < prodDataSet.size(); i++) {...}
prodAdapter.filterList(filteredList);
```

상품의 이름(가격, 메모)을 가지고와 EditText를 비교하여 같은 것이 있다면 list의 추가합니다.

근데 이것을 이름과 가격, 메모의 모두 같은 동작을 하니 이름의 '키보드', 메모의 '키보드'가 있으면 중복으로 list의 추가돼 아이템이 2개가 추가되는 문제가 있었습니다.

```
hs = new HashSet(filteredList); //HashSet 배열
```

이를 해결하기 위해 filteredList.add()를 하며 동시에 hs.add(filteredList);를 통해 hs라는 HashSet 배열에 추가합니다. 이후 아래 작업에서는 hash배열의 존재한다면 추가하지 않고 아이템이 없다면 추가합니다.

```
if(!(hs.contains(filteredList))){
    filteredList.add(prodDataSet.get(i));
    hs.addAll(filteredList);
}
```

이를 통해 해결했습니다.

## <메인창 아이템 클릭 시 해당 정보 확인창 이동>

메인창의 아이템 클릭 시 동작을 할 때 해당 view 객체에서 상품 정보를 가져올 수 있습니다.

그러나 저희의 View객체는 다음 사진과 같이 상품의 이름, 가격, 메모만 Text로 있습니다.

이러한 값을 가져와서 putExtra로 넘겨주는 것입니다.



```
Intent intent2 = new Intent(view.getContext(), ActivityCheckProd.class);
intent2.putExtra( name: "name", prod_name.getText().toString());
intent2.putExtra( name: "price", prod_price.getText().toString());
intent2.putExtra( name: "note", prod_note.getText().toString());
```

그래서 위와는 다르게 Bitmap을 출력하기 위한 byte[]와 Star, Link에 대한 값의 출력은 다른 Text와는 다르게 동작해야 합니다.

이를 사용하기 위해 해당 ProdData를 저장한 dataSet에서 getter를 이용하여 가져왔습니다.

```
String prod_link1 = data.getProd_link();    int prod_star1 = data.getProd_star();
intent2.putExtra( name: "link", prod_link1); intent2.putExtra( name: "star", prod_star1); //Link, Star
```

```
Bitmap prod_Bitmap = data.getProd_bit_image();
ByteArrayOutputStream stream = new ByteArrayOutputStream();
prod_Bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 100, stream);
byte[] byteArray = stream.toByteArray(); //Bitmap 이미지
```



## <등록, 수정 시 사진X 저장>

사용자가 상품을 등록할 때 사진을 직접 고르기 때문에 사진이 없이 동작하는 경우가 있습니다.

그러나 상품 정보가 이동하면서 Bitmap을 set할 때 값이 존재하지 않는 경우 오류가 발생하여 에뮬레이터 동작이 멈췄습니다.

이를 해결하기 위해 다음과 같이 코딩을 했습니다. <ActivityRegProd.java>

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //=====갤러리 동작=====
    if (requestCode == IMAGE_REQUEST_CODE) { //사진받아오기
        if (resultCode == RESULT_OK) {
            try {
                InputStream in = getContentResolver().openInputStream(data.getData());

                Bitmap bitmap = BitmapFactory.decodeStream(in); //이미지 파일을 변환할 때는 BitmapFactory 클래스
                ByteArrayOutputStream stream = new ByteArrayOutputStream();
                bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 100, stream);
                byteArray = stream.toByteArray();
                img_ok = true;
                imageView.setImageBitmap(bitmap); // img 를 이미지뷰에 출력
                Toast.makeText(getApplicationContext(), text "사진 불러오기 성공", Toast.LENGTH_SHORT).show();
            } catch (Exception e) {
            }
        } else if (resultCode == RESULT_CANCELED) { // 뒤로가기로 작업 취소한때
            Toast.makeText( context: this, text "사진 선택 취소", Toast.LENGTH_LONG).show();
            if(img_ok != true)img_ok=false;
        }
    }
    //=====갤러리 동작=====
}
```

이처럼 img\_ok라는 boolean 변수를 생성하여 구분 짓습니다.

아래는 갤러리에서 사진을 가져온 뒤 다시 갤러리에서 취소 동작을 하는 것을 구분짓기 위해 if문을 사용했습니다.

이후 저장을 클릭할 시 다음과 같이 동작합니다.

```
if(img_ok == true) {
    intent.putExtra( name: "image", byteArray);
}
else if(img_ok == false){
    byte[] tmp = null;
    intent.putExtra( name: "image",tmp);
}
```

이처럼 이미지가 바뀌었을 경우 가져온 이미지를 putExtra로 넘기는 동작을 하고, 이미지가 없다면 null값을 보내줍니다.

이러한 동작은 수정 시에도 마찬가지였습니다.

사진이 있는 상품을 클릭 후 이미지 변경을 하지 않는다면 오류가 발생했습니다.

<ActivityModProd.java>

```
if(change_img==1) //이미지 변경 시
    intent1.putExtra( name: "image",change_image_arr);
else
    intent1.putExtra( name: "image",get_image_arr);
```

그래서 위와 같이 change\_img라는 int형 변수를 통해 제어하여 변경이 됐을 경우와 안 된 경우 처리를 해줬습니다.

## <관심 상품 정렬>

메뉴의 관심목록 클릭 시 filterList를 통해 관심목록만 출력하였으나, 관심목록에서 다시 일반 모드로 돌아가는 동작을 진행하던 중 관심목록만 상단의 출력하는 게 어떨까라는 생각이 들어 그렇게 진행했습니다.

기존 filterList를 통해 관심목록만 출력한 작업을, 관심도의 값인 int prod\_star를 만들어 prod\_star가 1이면 관심 상품, 0이면 일반 상품으로 뒤 이를 크기 비교했습니다.

## <수정 시 Intent.Extra() 오류>

상품 정보는 Main에서 상품 클릭 시 startActivityForResult를 통해 Check로 넘어가는 동작을 합니다.

이후 Check에서는 Mod로 넘어가 수정하는 작업을 합니다. 이후 수정된 값을 Main으로 가져오는 방식입니다.

따라서 처음에는 Main -> Check -> Mod -> Main 으로 돌아가는 방식을 생각했습니다.

그러나 이 방법으로 코드를 작성했을 때 Main에 제대로된 값이 도착하지 않는 것을 확인하였으며, 관련 자료를 참고하여 Main -> Check -> Mod -> Check -> Main으로 돌아오는 방식으로 해결했습니다.

## <AlarmManager 구현 문제>

처음 푸시알림을 구현하기 위해 Notification을 통한 푸시알림의 동작과 AlarmManager와 Broadcast receiver, TimePicker를 통한 특정 시간을 지정하여 사용자에게 푸시알림을 나타내는 동작을 구현하고자 했습니다.

그러나 관련된 자료를 통해 구현하고자 했던 동작들을 찾아봤지만 참고한 자료 중 성공했던 자료가 없었기 때문에 저희는 다른 방법을 찾았습니다. 이 방법은 Handler를 이용한 방법입니다.

Handler라는 인터페이스를 사용하면 이 Handler에 있는 메서드인 postDelayed()를 사용했습니다.

그러면 사용자가 지정한 ms 만큼 지나면 run()함수가 동작하는 지연 동작을 합니다. 이를 구현하기 위해 다음과 같이 코드를 작성했습니다. <ActivityCheckProd.java>

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {...}
}, push_msec); //라디오버튼만큼 delay
```

지정한 push\_msec의 시간만큼(단위 ms) 지나면 run()함수가 동작하게 됩니다. 이 run()함수에는 푸시알림 함수를 집어넣어서 동작했습니다.

그러나 위와 같은 방식의 문제점이 발견됐습니다.

어플리케이션을 background에서 동작하고 있는 경우에는 잘 동작하지만, 완전히 종료된 경우 이 Handler를 통한 이벤트가 사라지는 문제점이 있었습니다. 기한 내에 Broadcast receiver를 적용시키지 못하여 이대로 최종 진행하였지만 향후 Broadcast receiver를 구현한다면 푸시 알림의 구현을 해결할 수 있을 것이라 생각합니다.



## <데이터베이스 상품 구분>

데이터베이스에 상품이 저장될 때 각 상품은 아래와 같은 테이블의 형태로 저장됩니다.

	prod_name	prod_price	prod_link	prod_note	prod_star	prod_date	prod_img
	필터	필터	필터	필터	필터	필터	필터
1	ARCH TEE	18,000	http://mss.kr/2409894?_gf=A	반팔, 검정	1	220503	BLOB
2	시그니처 로고 반팔티	19,900	https://www.hiver.co.kr/products/b/66706556	반팔, 흰색	0	220501	BLOB
3	미니멀프로젝트 로버스트 헤비 오버핏 반팔티셔츠	19,000	https://www.hiver.co.kr/products/b/15819478	반팔, 회색	1	220504	BLOB
4	Durgod 토러스 K320KR 텐키리스 투톤 키보드	129,500	http://naver.me/GazatNbY	키보드, 검정, 흰색	0	220507	BLOB
5	바일로 고래 VA87M 텐키리스 키보드	168,000	http://naver.me/G5l7lrFb	키보드, 하늘색, 흰색	1	220405	BLOB
6	콕스 COX CK87 게이머톤 텐키리스 키보드	49,900	http://naver.me/GxOtp3oe	키보드, 검정색, 빨간색	1	220431	BLOB
7	뉴발란스 CM878MC1	129,000	https://www.nbkorea.com/product/...	신발, 회색	0	220427	BLOB
8	나이키 에어 포스107	119,000	https://www.nike.com/kr/ko_kr/t/men/fw/nik...	신발, 흰색	0	220421	BLOB

이때 각 상품에 수정이나 삭제 등 여러 동작을 할 때 데이터베이스를 제어하기 위해 고유한 값을 가지는 ID를 부여해야했습니다.

처음에는 상품에 ID를 추가하여 코드를 변경하려 하였지만, 저희는 상품의 이름(prod\_name)을 고정 값으로 두고 이를 통해 제어를 하기로 결정했습니다.

사용자가 상품을 등록할 때 가격이나 링크, 메모 할 사항 이미지 관심 등은 사용자가 계속해서 변경할 이유가 발생할 수 있지만 상품 이름은 변하지 않을 거란 생각으로 그렇게 진행했습니다.

그렇기 때문에 수정 창에서는 상품의 이름은 수정하지 못하게 prod\_name을 TextView로 선언했습니다.

```
prod_name = (TextView) findViewById(R.id.mod_name);
```

이후 데이터의 수정과 삭제 동작에서 데이터베이스 코드의 WHERE를 prod\_name으로 제어를 했습니다.

```
sqlDB.execSQL("UPDATE prod_List "+"SET prod_price = " + Price +  
" WHERE prod_name = " + Name + " "); //수정 시 DB 동작
```

```
sqlDB.execSQL("DELETE FROM prod_List WHERE prod_name = "  
+ prodDataSet.get(pos).getProd_name() + " "); //삭제 시 DB 동작
```

## <데이터베이스 이미지 출력>

프로젝트의 마무리 단계에서 데이터베이스를 모두 구현하고 실행했을 때 데이터베이스의 이미지가 출력되지 않는 문제가 있었습니다. 이를 해결하기 위해 Log와 Toast메시지를 통해 확인해본 결과 다음과 같은 문제가 있었습니다.

```
@Override  
public void onCreate(SQLiteDatabase db) {  
    db.execSQL("CREATE TABLE prod_List ( "  
        "prod_name TEXT, " +  
        "prod_price TEXT, " +  
        "prod_link TEXT, " +  
        "prod_note TEXT, " +  
        "prod_star INTEGER, " +  
        "prod_date TEXT, " +  
        "prod_img BLOB);");  
}
```

//SQLite 테이블 생성

위 코드는 저희 데이터베이스의 테이블을 생성하는 부분입니다. 이때 이미지는 Bitmap으로 되어 있는데, 이러한 Bitmap은 byteArray와 연관되어 있기 때문에 BLOB의 형태로 생성해줬습니다.

이후 다른 동작들과 마찬가지로 SQLiteDatabase.execSQL("SQLite 명령어");를 통해 생성해줬습니다.

그러나 위 동작으로 실행했을 때 byteArray값이 저장된 것은 String의 형태로 저장이 되었으며, 이를 다시 getBlob을 통해 받아오면 가져오는 byteArray는 전혀 다른 값으로 바뀌어 이미지 출력이 되지 않았습니다.

1 | 나야 | 124 | 532 | 325 | 1 | 220602 | [B@546f199 //당시 DB 저장 (col : 7)

최종 기한까지 얼마 남지 않았지만 프로그램의 가장 중요한 기능이라고 생각된 등록한 상품의 저장문제이며 가장 시각적으로 눈에 보이는 이미지 부분이었기 때문에 여러 가지 자료를 찾아봤습니다.

그렇게 찾은 해결 방법은 다음과 같았습니다.

```
SQLiteStatement p = sqldb.compileStatement(
    sql: "INSERT INTO prod_List VALUES(?, ?, ?, ?, ?, ?, ?);");
p.bindString( index: 1, regData.getProd_name());
p.bindString( index: 2, regData.getProd_price());
p.bindString( index: 3, regData.getProd_link());
p.bindString( index: 4, regData.getProd_note());
p.bindLong( index: 5, regData.getProd_star());
p.bindString( index: 6, regData.getProd_date());
if(Image_byte != null)p.bindBlob( index: 7, Image_byte);
p.execute();
```

//상품 등록 시 코드

SQLiteStatement라는 인터페이스를 생성하여 이를 통해 데이터베이스의 값을 넣어줬습니다. 이때 INSERT INTO를 사용하여 값을 집어넣을 때 VALUES의 값을 ?로 준 뒤, bindXXX(?위치, 들어갈 값)을 통해 값을 집어넣었습니다.

ARCH TEE	18,000	http://mss.kr/2409894?_gf=A	반팔, 검정	1	220503	BLOB
----------	--------	-----------------------------	--------	---	--------	------

그럼 위와 같이 DB에 값이 저장되었으며, 이는 DB에서도 다음과 같이 확인이 가능했습니다.

데이터(T): prod\_List

prod_name	prod_price	prod_link	prod_note	prod_star	prod_date	prod_img
ARCH TEE	18,000	http://mss.kr/2409894?_gf=A	반팔, 검정	1	220503	BLOB
시그니처 로고 반팔티	19,900	https://www.hiver.co.kr/products/b/66706556	반팔, 흰색	0	220501	BLOB
미니멀프로젝트 로바스트 세비 오버핏 반팔티셔츠	19,000	https://www.hiver.co.kr/products/b/15819478	반팔, 흰색	1	220504	BLOB
Durgod 토러스 K320R 텐키리스 투톤 키보드	129,500	http://naver.me/GazatNbY	키보드, 검정, 흰색	0	220507	BLOB
버릴로 고대 VA87M 텐키리스 키보드	168,000	http://naver.me/GS97rFb	키보드, 하얀색, 흰색	1	220405	BLOB
콕스 COX CK87 게이밍 텐키리스 키보드	49,900	http://naver.me/GxOtp3oe	키보드, 검정색, 빨간색	1	220431	BLOB
뉴발란스 CM879MC1	129,000	https://www.nbkorea.com/product/...	신발, 흰색	0	220427	BLOB
나이키 에어 포스107	119,000	https://www.nike.com/kr/ko_kr/men/tw/nik...	신발, 흰색	0	220421	BLOB

모드: 이미지

이를 통해 값이 제대로 저장되는 것을 확인할 수 있었으며, 이를 getBlob으로 가져오는 작업을 통해 해결할 수 있었습니다.

### <첫 번째 상품 생성 오류 및 초기데이터 버튼 생성>

상품 정보가 아무것도 없을 때 이미지가 들어간 파일을 생성하면 오류가 발생하는 문제가 있었습니다.



그림 1. 사진X 저장



그림 2. 사진O 저장

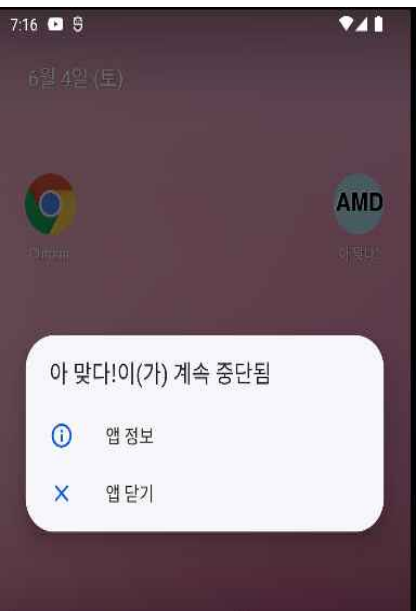


그림 3. 그림2 이후 동작

위를 해결하고자 코드에서 문제가 되는 부분을 Toast메시지와 Log를 통해 찾았습니다.

```
//=====Intent 응답 =====
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == PROD_REG_CODE) { //등록코드
        byte[] Image_byte = data.getBytesExtra(name: "image"); //비트맵이미지 받기

        Bitmap image;
        if(Image_byte != null) {
            image = BitmapFactory.decodeByteArray(Image_byte, 0, Image_byte.length);
            ImageView prod_img = (ImageView) findViewById(R.id.prod_img);
            prod_img.setImageBitmap(image); //비트맵으로 이미지 출력
        }
        else{
            Image_byte = null;
            image = null;
        }
    }
}
```

해당 부분의 코드가 문제였지만 값의 index나 position값과 같은 것이 존재하지도 않았으며, getByteArrayExtra를 통한 Image\_byte의 byteArray도 값이 제대로 들어가 있는 것을 확인했습니다.

이런 원인 모를 문제가 생겨났으며 첫 번째 상품을 저장할 때 사진이 있으면 오류가나고 사진이 없는 경우 생성이 되었습니다. 또한 2번째 상품부터는 이미지가 존재해도 저장이 잘 됐기 때문에 데이터 값을 미리 생성하고자 했습니다.

이러한 데이터 값을 무엇으로 할지 고민하다가 기존에 프로젝트를 진행하면서 원활한 확인을 위해 만들어둔 데이터 값들이 있었습니다. 이를 초기 데이터로 사용하면 어떨까 싶어 초기데이터 버튼을 생성했습니다.

초기데이터 버튼을 통해 저희가 미리 만든 8개의 아이템을 DB의 추가하고 프로그램을 다시 껐다가 켜야 데이터베이스가 적용이 되기 때문에 이를 사용자가 따로 동작을 안 해도 할 수 있도록 restart() 시키는 메서드를 Main에 추가하여 초기데이터 버튼 클릭 시 동작이 되게 했습니다.

또한 오류 동작이 발생하지 않는 것을 목표로 하기 위해 다음과 같은 대처방안을 뒀습니다.



그림 1. + 아이콘 클릭 시(DB X)



그림 2. 마지막 상품 삭제 시



그림 3. 그림2 이후 동작

## 7. 동작 시험



어플리케이션 실행화면입니다.

우선 사용자가 등록한 상품들을 리스트로 보여주는 화면이자 메인화면입니다.

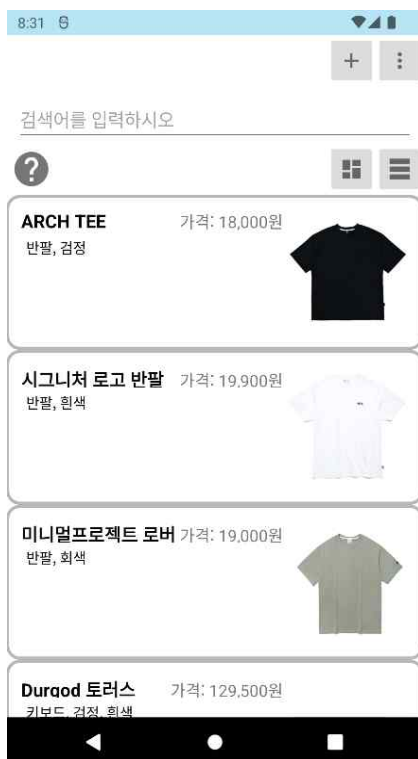


그림 1. 목록형 리스트



그림 2. 목록형 리스트2

위처럼 사용자가 등록한 상품의 리스트를 스크롤하며 볼 수 있습니다.

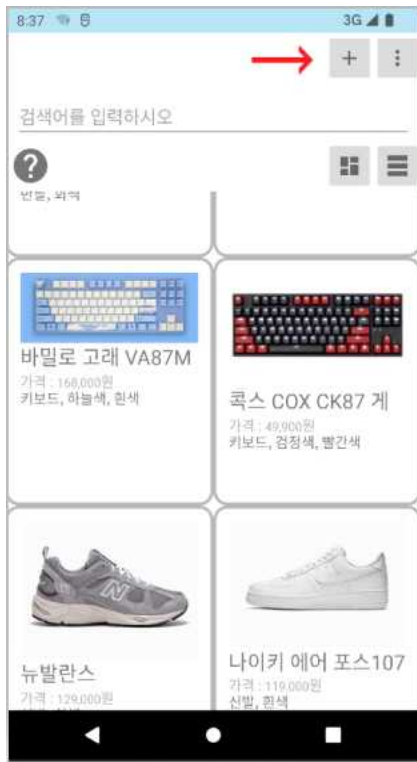


그림 3. 앨범형 리스트

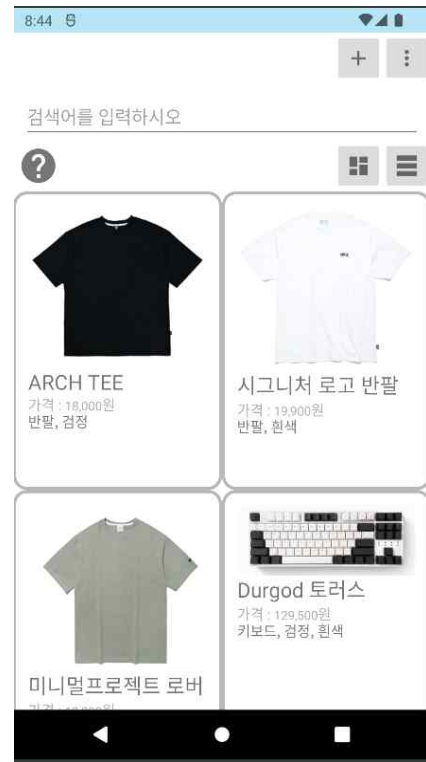


그림 4. 앨범형 리스트2

앨범형 선택 시 사용자에게 보여주는 리스트 형태가 변경됩니다.

다음은 사용자가 상품을 등록하는 화면입니다.

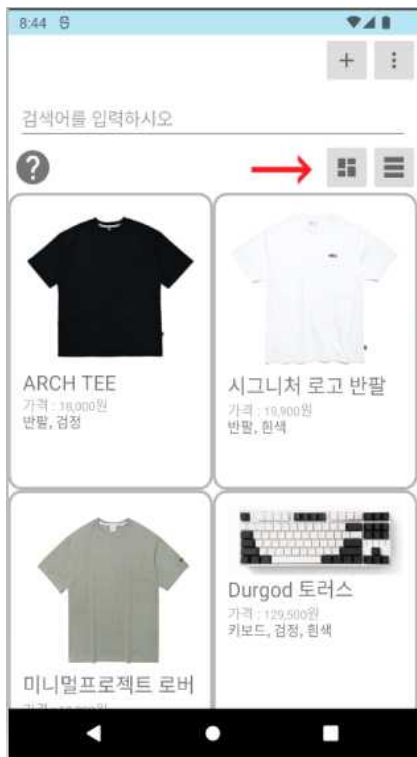


그림 5. 등록 버튼 클릭

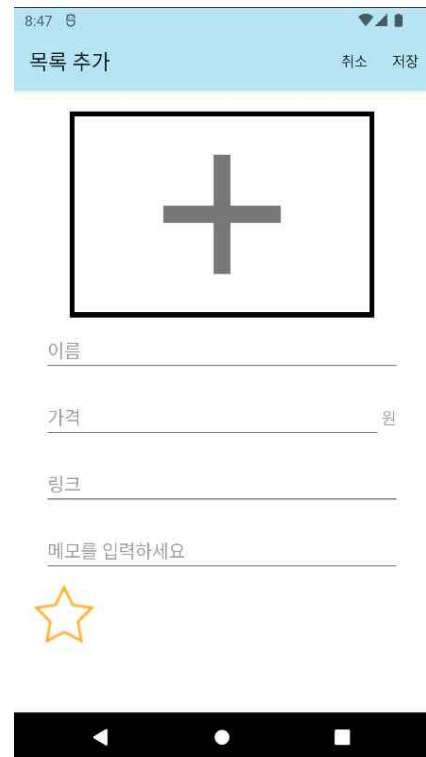


그림 6. 등록 창

위 + 모양의 이미지를 누르면 이동됩니다. 추가창에 값을 입력합니다.



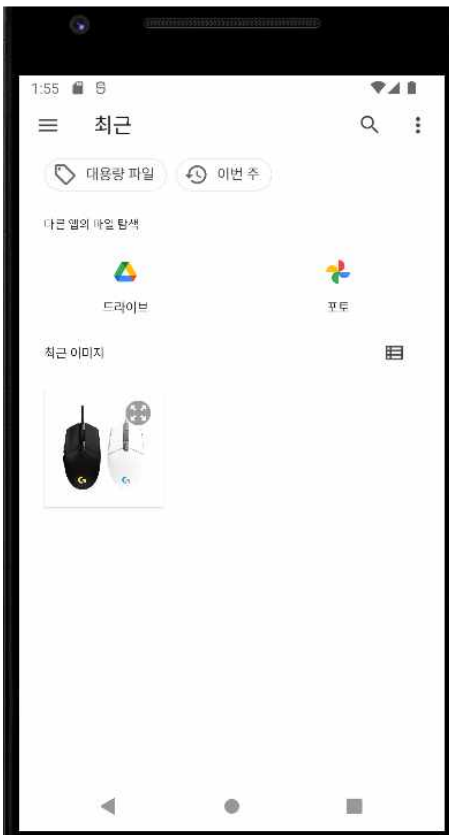


그림 7. + 이미지 클릭 시



그림 8. 사진 선택 취소 시



그림 9. 사진 선택 시

가운데 큰 + 이미지를 누르면 갤러리에서 사진을 가져옵니다.

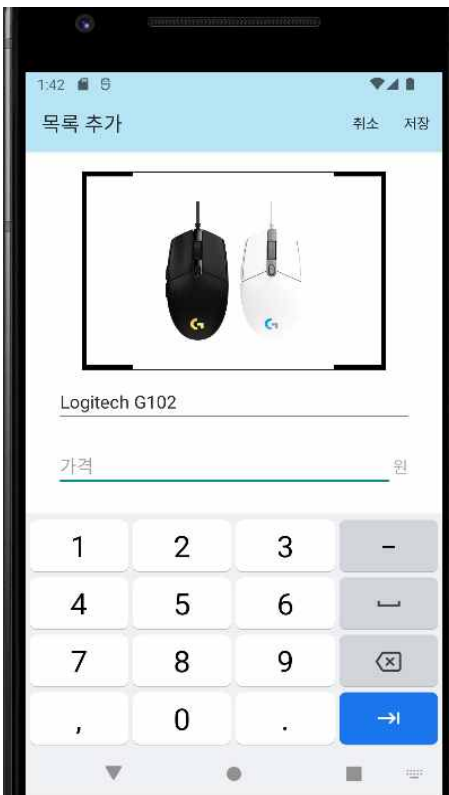


그림 10. 가격 입력(숫자)

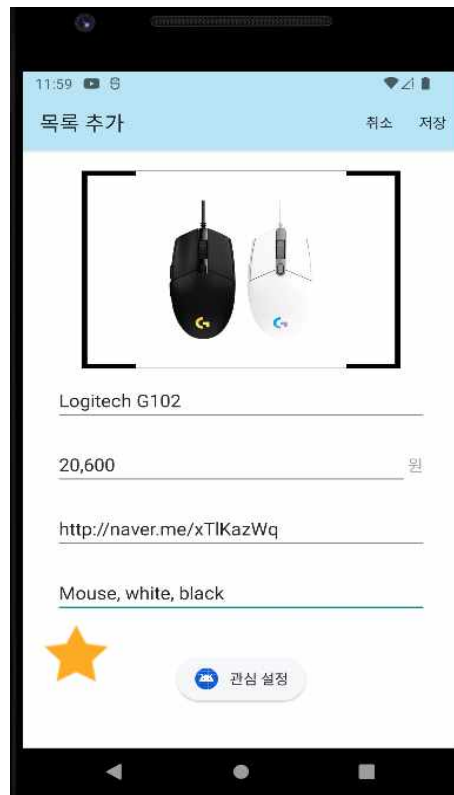


그림 11. 정보 입력

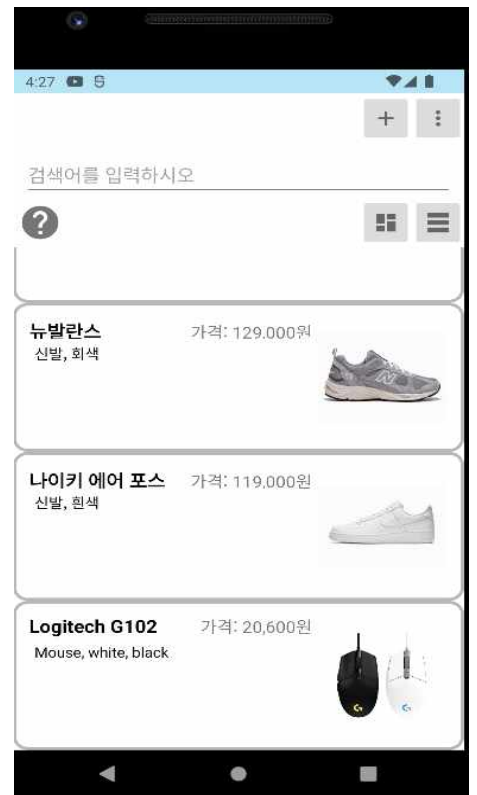


그림 12. 등록 완료 시

추가 창에 이미지와 값을 입력합니다. 숫자의 경우 입력을 숫자로만 받게 했습니다.  
이후 저장 버튼을 누르면 등록 완료와 함께 등록이 됩니다.

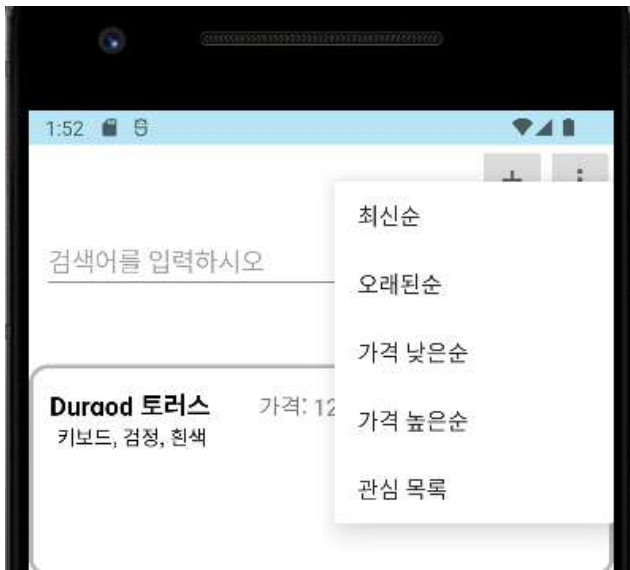


그림 13. 메뉴 버튼  
메뉴 버튼 클릭 시 동작입니다.



그림 14. 최신순



그림 15. 오래된 순

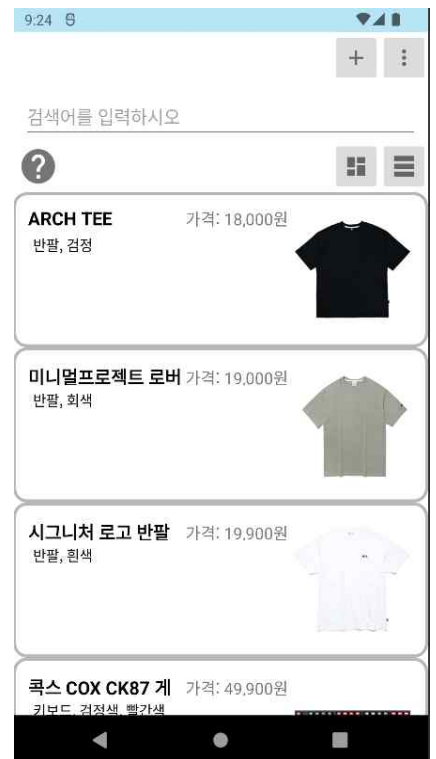


그림 16. 가격 낮은순



그림 17. 가격 높은순



그림 18. 관심 목록

각 메뉴 정렬 버튼 누를 시 동작입니다.

이때 최신순과 오래된 순은 내부에 저장된 날짜 값을 통해 제어되며, 관심 목록은 등록된 상품을 우선하여 상단에 출력해줍니다.



그림 19. 아이템 정보 클릭 시 동작

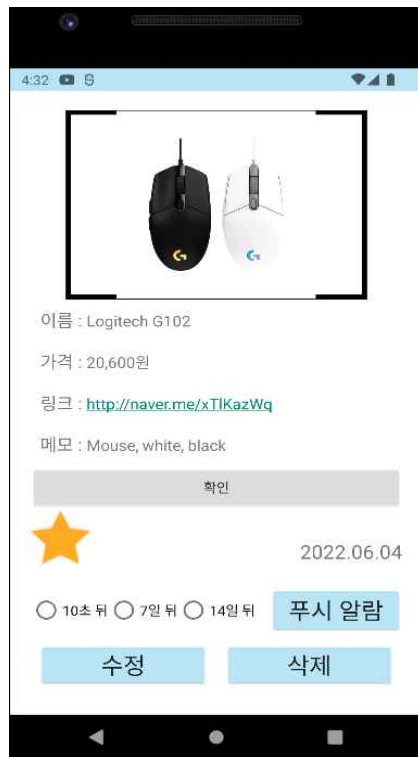


그림 20. 아이템 정보 클릭 시 동작2

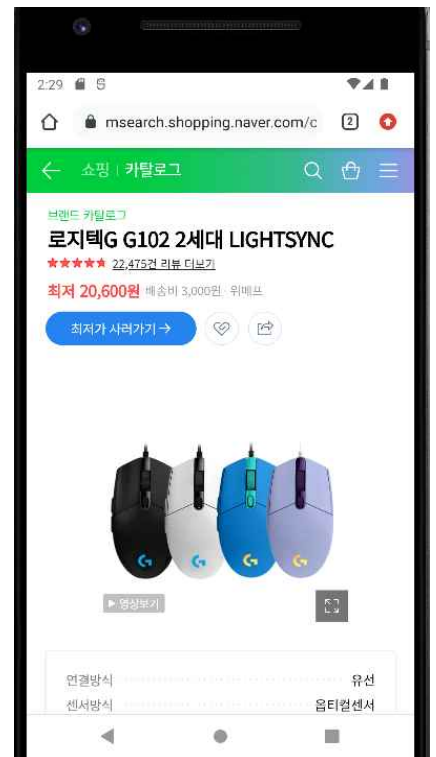


그림 21. 링크 클릭 시 연결

리스트의 아이템을 클릭 시 아이템 정보의 상세 정보를 보여주고 링크 클릭 시 해당 링크로 이동하는 동작을 합니다.



그림 22. 검색 동작



그림 23. 검색 동작 2



그림 24. 검색 동작 3

상단 검색창의 입력 시 검색 키워드에 따라 상품이 출력되는 것을 볼 수 있습니다.



그림 25. 등록 상품 확인창



그림 26. 상단바 푸시 알림 확인



그림 27. 수정 버튼을 눌렀을 때

등록된 상품 목록을 클릭하면 입력된 정보를 수정하거나 상품 구매 알림 설정을 할 수 있습니다.



그림 28. 입력된 메모 수정



그림 29. 수정된 상품 리스트

등록된 상품 목록을 클릭하면 입력된 정보를 수정하거나 상품 구매 알림 설정을 할 수 있습니다.



그림 30. 설명서 버튼



그림 31. 설명서1

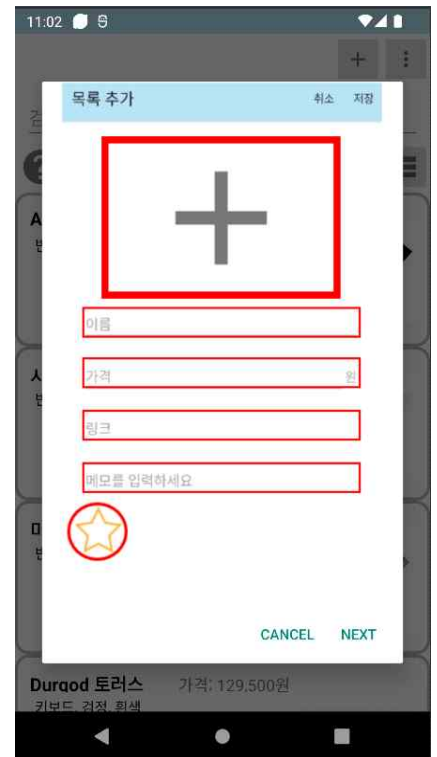


그림 32. 설명서2

왼쪽 물음표 버튼을 누르면 이 어플이 어떻게 기능을 하는지 알 수 있도록 간단한 설명서를 볼 수 있습니다.



8. 평가

8.1 기능적 요구사항 목표달성도 평가

요구사항	목표 달성도	우선 순위	구현 기능
순서 정렬 아이콘	100%	3	앨범형, 리스트형 동작과 상품의 정보를 가지고 메뉴 버튼을 통해 순서 정렬 동작까지 구현 완료.
상품 등록	100%	2	기능을 모두 구현하여 등록하면 아이템이 추가되며, 데이터베이스에 상품이 저장됨
알림 메시지	80%	7	상품을 등록한 뒤 푸시알림을 설정하여 일정 시간이 흐른 뒤 사용자에게 푸시알림 전송이 가능함. 그러나 어플리케이션 완전히 종료 시 딜레이 이벤트가 동작하지 않아 어플리케이션이 켜진 상태로 유지되어야 함.
등록 상품 시각화	100%	1	사용자가 등록한 상품을 스크롤을 통한 확인과 아이템 클릭 시 확인창이 나타나 링크로 연결하는 작업까지 완료했음.
중요도 표시	90%	4	중요도 체크하여 값을 설정하는 기능은 추가하였으나, 중요상품만 출력하는 것이 아닌 중요 상품을 상단에 출력함.
등록 상품 상세 정보 확인	100%	6	등록된 상품은 데이터베이스에서 가져와 어플리케이션을 종료해도 상품 정보가 존재하며, 아이템 클릭 시 해당 아이템 정보를 확인할 수 있으며, 링크 클릭 시 해당 링크로 넘어가는 동작을 함.
검색 기능	100%	5	키워드 검색을 하면 이름, 가격, 메모의 해당하는 키워드가 들어가면 해당 내용의 상품만 사용자에게 보여줌.

8.2 현실적 제한요소 달성도 평가

현실적 제한 요소들	목표	달성결과
경제	- 인적 자원 외 다른 자본 불필요 - 무료배포로 인한 가격 경쟁력 확보	프로그램 개발의 있어 비용이 발생하지 않았으며 사용자에게 무료로 배포를 함으로써 발생하는 비용은 줄이고 광고 같은 것을 통해 수익도 기대할 수 있을 거라고 생각됨
편리	- 사용자가 관심 있는 제품만 선별하여 비교 - 사용자가 원하는 방식으로 비교 나열 가능	여러 개의 쇼핑물을 이동하지 않고 저장한 상품을 한 곳에 모아놓은 뒤 확인할 수 있어 편리함
윤리	- 도덕적 윤리에 어긋나지 않고 삶의 질 향상	아 맞다 애플리케이션을 통해 삶의 질을 향상시킬 수 있고 도덕적 윤리에 어긋나지 않았음.
안전성	- 단순 정보 저장과 이동으로 메모리 영향이 낮음	사용자에 개인 정보의 접근하는 일 없이 직접 정보를 추가하는 것이므로 안전성이 보장됨.
유지관리 용이성	- 사용자가 원할 때 수정 삭제 가능	사용자가 리스트를 삭제할 수 있는 기능은 구현하였지만 수정기능은 추후에 추가할 예정임
사회	- 특정 제품의 우대나 쇼핑물의 독점의 불가	제작된 프로그램은 특정 쇼핑물이 아닌 인터넷에 있는 여러 개의 쇼핑물을 타깃으로 하기 때문에 특정 쇼핑물에 대한 독점 문제가 없음

## 9. 추진체계

### - 조장(김범수)

: 프로젝트 총괄 및 일정 조율

: 오류 수정 및 제작

### - 팀원 (김성범)

: 프로그램 제작, 알고리즘 연구

: 자료조사

: 프로그램 Simulation

### - 팀원 (정병윤)

: 프로그램 제작, 알고리즘 연구

: 자료조사

: 아이디어 및 기능 구현

## 10. 설계 추진 일정: 2022년 4월 ~ 2022년 6월

수행 내용		일정 (1주 단위)						
		1	2	3	4	5	6	7
목표와 기준 설정	- 설계목표 설정 - 기능 블록도 구성 - 역할 분담							
구조 설계	- 프로그램 요구사항 파악 - 통일된 네이밍 설정 - 세부 기능 블록도							
분석	- 기능별 구현방법 결정 - 적용할 이론 및 기술 - 프로그램 코딩							
합성 및 점검	- 프로그램 코딩 - 중간점검 및 보고서 작성							
제작	- UI 구현 및 설계							
시험	- 시험 및 검증 - 오류 수정 및 최종 검증							
평가	- 결과보고 및 시연							

## 11. 설계과제를 통해 배운 점

약 7주간 팀 프로젝트를 진행하면서 모르는 것은 스스로 찾아보고 또는 팀원에게 물어보고 알려주면서 배우는 것이 배움에 있어서 가장 많이 도움이 된다는 것을 느꼈습니다.

그리고 배운 것을 토대로 결과물이 완성되어 가는 것을 보면서 큰 기쁨도 느낄 수 있었습니다. 처음 목표했던 대로 Android studio의 상세 동작을 이해할 수 있었으며 파일 이름과 구성요소들을 일관되게 정의함으로써 통일된 네이밍이 왜 필요한지 한 번 더 느끼게 되는 계기가 되었고 서로 배려하고 대화도 많이 하면서 팀 프로젝트를 통한 협업 능력이 향상됨을 느낄 수 있었습니다. 힘들고 지루할 수도 있는 긴 시간이었지만 끝까지 완성시키고 포기하지 않으려고 함으로써 책임감도 많이 향상되었음을 느낄 수 있었습니다. 앞으로 남은 학기동안 다른 팀 프로젝트들이 많이 남았지만 이번 경험을 통해 좀 더 발전되고 좋은 작품들을 만들 수 있게 되었으면 좋겠습니다.

## 부록

<프로그램 소스코드>

- 라인별 커멘트 작성
- 캡스톤지원시스템에 업로드