
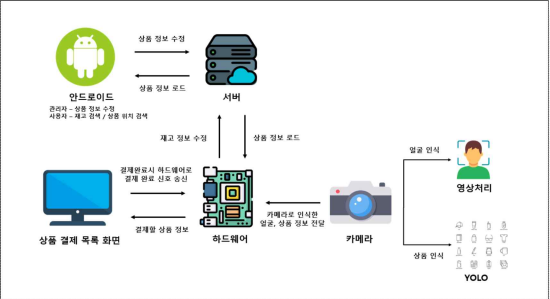
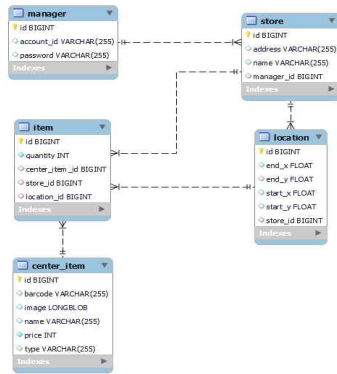


종합설계1 과제 수행 보고서

작품과제명	스마트 매장 관리 및 결제 시스템 (SSMPS: Smart Store Management and Payment System)
과제 개요	<ul style="list-style-type: none"> 과제 선정 배경 <p>- 다이소나 일반 무인편의점을 사용할때의 불편한 점은 계산이 오래 걸린다는 점입니다. 상품의 개수가 많다면 하나하나 바코드를 찾아 찍기 때문에 속도가 빠르지 않을뿐더러 번거롭기 때문입니다. 그리고 사용자 입장에서 해당 편의점과 같은 상품이 진열된 매장을 이용할 때 원하는 상품의 재고가 있는지를 확인하는 것 또한 직접 찾아가서 보지 않는다면 알 수가 없습니다.</p> <p>따라서 해당 문제들을 개선할 수 있도록 하는 시스템을 개발하려고 합니다.</p> <ul style="list-style-type: none"> 과제의 필요성 <ol style="list-style-type: none"> 빠른 결제와 편의성 제공: 현대 사회에서는 시간이 중요한 가치로 여겨집니다. 상품의 바코드를 일일이 찍어 결제하는 시스템은 시간이 오래 걸리며, 사용자에게 불편함을 주는 요소입니다. 바코드를 하나하나 찍지 않고 이미지인식을 통해 결제하는 시스템을 개발함으로써, 사용자는 더 빠르고 편리하게 결제를 진행할 수 있습니다. 실시간 재고 관리와 투명성: 현대의 소비자들은 상품의 재고 상태와 위치에 대한 정보를 실시간으로 알고 싶어합니다. 이를 통해 불필요한 시간 낭비와 헛걸음을 줄일 수 있습니다. 상품의 재고 수량과 위치를 실시간으로 확인할 수 있는 시스템은 소비자에게 편의성을 제공하며, 매장 운영자 또한 실시간으로 재고를 관리하고 더욱 효과적인 상품 배치를 구성할 수 있습니다. 무인 매장 운영의 효율성 향상: 바코드 기반 결제 시스템은 사용자의 결제 과정을 지체시키며, 매장 운영자의 재고 관리를 어렵게 만듭니다. 상품의 이름을 통한 결제와 실시간 재고 관리 시스템은 무인 매장 운영의 효율성을 향상시키는 핵심 요소가 될 수 있습니다.
과제 내용	<ul style="list-style-type: none"> 과제 구성 <p>- 시스템 설계 및 데이터베이스 구축: 상품에 대한 정보를 관리하는 데이터베이스를 구축합니다. 이 데이터베이스에는 상품 이름, 가격, 재고 수량, 위치 등의 정보가 포함됩니다.</p> <p>- 앱개발(사용자 앱/ 결제 화면 앱):</p> <ul style="list-style-type: none"> [사용자 앱] <ul style="list-style-type: none"> 사용자는 상품의 재고 수량과 위치를 확인할 수 있고, 관리자는 앱과 웹에서 재고를 관리하고 위치를 수정할 수 있습니다. [결제화면 앱] <ul style="list-style-type: none"> 결제화면 앱에서는 인식된 제품의 결과가 출력되고 결제를 진행할 수 있습니다. <p>- 상품 인식 AI 기능 구현 : 상품을 인식하고, 이를 바탕으로 결제를 진행하는 AI 기능을 개발합니다.</p> <p>- 얼굴 인식 기능 구현: 결제는 얼굴이 인식되거나 결제화면 앱에서 결제 시작을 누르면 시작됩니다. 해당 부분에서 얼굴을 인식할 수 있는 Cascade를 이용하여 얼굴을 인식하였습니다.</p> <ul style="list-style-type: none"> 과제 주요 특징 <p>- 결제 편의성 증대 : 상품을 이미지 인식을 통해 한번에 처리하기 때문에 바코드를 하나하나 찍는 불편함을 줄일 수 있습니다.</p> <p>- 실시간 재고 및 위치 확인: 애플리케이션을 이용하여 해당 제품의 대략적인 위치와 재고 확인을 통해 사용자의 편의성을 높일 수 있습니다.</p>

	<p>- Jetson Nano 설정</p> <p>Jetson Nano 하드웨어 구성 :</p> <ul style="list-style-type: none"> CSI 케이블 : 1개 MIPI CSI-2 카메라 : 1개 Wi-Fi 모듈 <p>Jetson Nano 환경 구성 :</p> <ul style="list-style-type: none"> 개발환경 : Python 3.6 운영체제 : Ubuntu Linux 모델 : JetPack 4.6 [L4T 32.6] CUDA : 10.2.300 cuDNN : 8.2.1.32 OpenCV : 4.5.4
	 <p>- 실제 조립된 모습 : 카메라의 경우 원래 2대를 사용해야 하지만, CSI 소켓이 파손되어 2개를 사용하진 못하였습니다.</p> <p>Jetson Nano가 Server 가 되어 Client(결제 앱) 과 통신을 합니다. 그리고 DB 관리를 위해 Spring Boot 와도 통신을 진행합니다.</p>
과제 내용	 <p>하드웨어(Jetson Nano)에서 상품 결제 목록 화면(결제 앱)</p> <p>서버(Spring boot)또한 함께 하드웨어에 정보를 주고, 요청을 받으면 재고 정보 수정 또한 진행합니다.</p> <p>서버 <--> 하드웨어 <--> 결제 앱</p> <p>이러한 형태로 각각 양방향 통신을 이용해 통신을 하였습니다.</p> <p>전달되는 정보의 형태는</p> <p>서버 -> 하드웨어 : 하드웨어에서 요청한 상품의 정보</p> <p>하드웨어 -> 서버 : 결제 시, 계산된 상품의 이름 및 개수</p> <p>하드웨어 -> 결제 앱 : 인식된 상품의 정보 (서버로부터 받아온)</p> <p>결제 앱 -> 하드웨어 : 결제 버튼이 눌렸는지 신호 전송</p>

- 시스템 설계 및 데이터베이스 구축:



Database 설계 구조

1. 데이터베이스 구성

데이터베이스 설계를 위와 같이 하였습니다.

- **center_item 테이블** : 편의점 본사에서 가지고 있는 물건 정보를 활용하기 위해서 임시로 테이블을 이용하였습니다. 물건의 바코드, 이미지, 이름, 가격, 종류 정보를 가지고 있습니다.
- **item 테이블** : 각각의 매장이 가지고 있는 물건에 대한 정보를 저장합니다. 본사 물건 테이블, 매장 테이블, 매대 테이블을 참조합니다.
- **location 테이블** : 편의점이 가지고 있는 매장 구조에 대한 정보를 저장합니다. 매대의 위치 좌표를 저장하고 매장 테이블을 참조합니다.
- **manager 테이블** : 매장 관리자의 정보를 저장합니다. 이 때 비밀번호는 hash 값을 저장합니다.
- **store 테이블** : 매장 정보를 저장합니다. 이름과 주소를 저장하고 관리자 테이블을 참조합니다.

2. 서버 구성 및 통신

- **서버 구성** : 스프링 부트 2.7 버전을 사용하였습니다. 자바 11을 이용하였고 gradle로 빌드하였습니다.

REST API로 통신 하였습니다. ORM(Object Relational Mapping)인 JPA를 사용해 데이터베이스를 제어하였습니다.

```

@DeleteMapping("/api/store/item/{id}")
public ResponseEntity<Object> deleteItem(@PathVariable Long id){
    Item findItem = itemService.getItemById(id);
    Item deletedItem = itemService.deleteItem(findItem);
    ItemResponse resultItem = new ItemResponse(deletedItem);
    return ResponseEntity.ok(resultItem);
}
  
```

서버에서 사용된 물건 삭제 통신 컨트롤러 코드

```

public List<Item> findItemName(String itemName, Store store) {
    return em.createQuery("select i from Item i where i.item.name = :name and i.store = :store")
        .setParameter("name", itemName)
        .setParameter("store", store)
        .getResultList();
}
  
```

JPA를 이용해 물건 이름으로 데이터베이스 조회 하는 코드

```

C:\Program Files\Java\jdk-11.0.16\bin\java.exe ...
...
:: Spring Boot :: (v2.7.11)

2023-06-17 15:52:35.980 INFO 5564 --- [main] c.ssmgs_backend.SsmgsBackendApplication : Starting SsmgsBackendApplication using Java 11
2023-06-17 15:52:35.984 INFO 5564 --- [main] c.ssmgs_backend.SsmgsBackendApplication : No active profile set, falling back to 1 default profile: 'default'
2023-06-17 15:52:35.984 DEBUG 5564 --- [main] o.s.boot.SpringApplication : Loading source class capstone_design_1.ssmgs_backend.SsmgsBackendApplication
2023-06-17 15:52:35.988 DEBUG 5564 --- [main] org.springframework.boot.SpringApplication : Refreshing org.springframework.boot.web.servlet
  
```

스프링 부트 서버 구동시 로그

통신 : 안드로이드에서 서버와의 통신을 위해 Retrofit2 라이브러리를 사용하였습니다.

```

@GET("/api/storeList/{id}")
Call<List<StoreResponse>> findStoreList(@Path("id") Long id);
  
```

Retrofit2 라이브러리의 네트워크 통신에 사용된 매장 리스트 조회 인터페이스

```

Call<List<StoreResponse>> findStoreList = service.findStoreList(nowManager.getId());
findStoreList.enqueue(new Callback<List<StoreResponse>>() {
    @Override
    public void onResponse(Call<List<StoreResponse>> call,
        Response<List<StoreResponse>> response) {
        if(!response.isSuccessful()){
            Toast.makeText(ManagerStoreSelectActivity.this, "매장 불러오기 에러",
                Toast.LENGTH_SHORT).show();
            Log.e("store finding error", response.errorBody().toString());
            return;
        }
        Log.e("store finding success", "매장 불러오기 성공");
        storeList = response.body().stream()
            .map(s -> new Store(s.getId(), s.getName(), s.getAddress(),
                s.getLocationList()))
            .collect(Collectors.toList());
        Log.e("store size", storeList.size() + "");
        setRecyclerView();
    }
    @Override
    public void onFailure(Call<List<StoreResponse>> call, Throwable t) {
        Toast.makeText(ManagerStoreSelectActivity.this, "매장 불러오기 실패",
            Toast.LENGTH_SHORT).show();
        Log.e("store finding", "error");
    }
});
  
```

매장 리스트 조회 인터페이스 실제 사용 코드

- 앱 개발 :

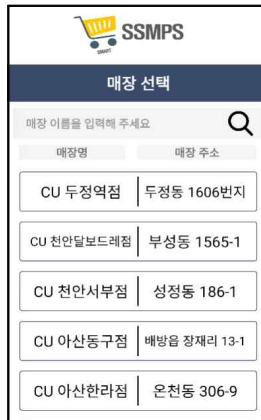
[사용자 앱] - 고객 기능 -> 상품의 재고 수량과 위치를 확인



[그림 1 : 사용자 앱의 초기화면]

과제 내용

-> 사용자 앱은 관리자 기능과 고객 기능이 있습니다.
고객은 별도 가입 없이 고객 로그인 버튼을 누르면 실행되도록 하였습니다.



[그림 2 매장 선택 조회 화면]

-> 조회하고 싶은 편의점을 선택 하는 화면입니다.
등록되어있는 편의점의 리스트가 나오고 검색 기능을 사용할 수 있습니다.

```
private void setStoreList(){
    Call<List<Store>> findAllStore = service.findAllStore();
    findAllStore.enqueue(new Callback<List<Store>>() {
        @Override
        public void onResponse(Call<List<Store>> call,
            Response<List<Store>> response) {
            if(response.isSuccessful()){
                Log.e("find store list error",
                    response.errorBody().toString());
                Toast.makeText(GuestStoreSelectActivity.this, "매장 불러오기 에러", Toast.LENGTH_SHORT).show();
                return;
            }
            storeList = response.body();
            Log.e("store size", storeList.size() + "");
            setRecyclerviewData();
        }
        @Override
        public void onFailure(Call<List<Store>> call, Throwable t) {
            Log.e("find store list fail", t.getMessage());
            Toast.makeText(GuestStoreSelectActivity.this, "매장 불러오기 실패", Toast.LENGTH_SHORT).show();
        }
    });
}
```

[실제 사용 코드]



[그림 3 매장 구성 조회 화면]
-> 조회하고 싶은 매장을 선택하면 해당 편의점의 구조(매대)와 등록되어있는 제품의 종류 이름이 보입니다.

과제 내용



[그림 4 매장 제품 조회 화면]
-> 제품 조회 버튼을 누르면 등록되어있는 제품들의 리스트를 조회할 수 있습니다.
해당 화면에는 제품의 위치를 확인할 수 있는 버튼과 제품의 상세 설명을 볼 수 있는 버튼이 있습니다.



[그림 5 제품 상세 조회 화면]
-> 매장 제품 조회 화면에서 제품의 상세 버튼을 누르면 나오는 화면입니다. 이 화면에서는 해당 제품의 이미지와 가격, 제품의 종류, 재고를 확인할 수 있습니다.



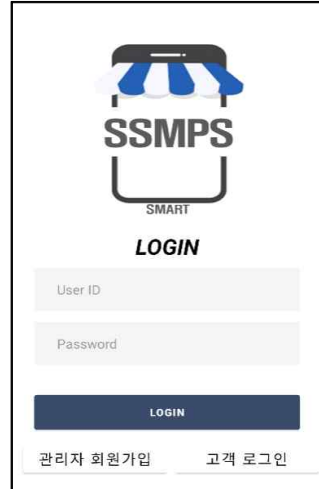

[그림 6 제품 진열 위치 조회 화면]
-> 매장 제품 조회 화면에서 제품의 위치 버튼을 누르면 나오는 화면입니다. 이 화면에서는 가게 제품의 위치를 확인할 수 있습니다. 선택한 제품의 위치는 진한 회색 색상으로 나타났습니다.

과제 내용	<pre>private void setData() { Call<List<Item>> findAllItem = service.findAllItem(nowStore.getId()); findAllItem.enqueue(new Callback<List<Item>>() { @Override public void onResponse(Call<List<Item>> call, Response<List<Item>> response) { if(response.isSuccessful()){ Log.e("find all item error", response.errorBody().toString()); Toast.makeText(GuestItemListActivity.this, "매장 물건 가져오기 에러", Toast.LENGTH_SHORT).show(); return; } itemList = response.body(); setRecyclerView(); } @Override public void onFailure(Call<List<Item>> call, Throwable t) { Log.e("find all item fail", t.getMessage()); Toast.makeText(GuestItemListActivity.this, "매장 물건 가져오기 실패", Toast.LENGTH_SHORT).show(); } }); }</pre>
	<p>[매장 제품 조회 사용 코드]</p>
	<pre>private void setData() { itemName.setText(nowItem.getName()); itemPrice.setText(Integer.toString(nowItem.getPrice())); itemType.setText(nowItem.getType()); itemQuantity.setText(Integer.toString(nowItem.getQuantity())); itemImg.setImageBitmap(TolImage(nowItem.getImage())); } private void getLocationData() { Intent intent = getIntent(); Store store = (Store) intent.getSerializableExtra("store"); Long storeId = store.getId(); } private void searchItem() { String itemName = itemNameInput.getText().toString(); // DB에서 itemName과 같 가져오기 Location location = locationList.get(0); }</pre> <p>[제품 상세 조회 화면]</p> <p>[제품 진열 위치 조회 코드]</p>

[매장 제품 조회 사용 코드]

[제품 상세 조회 화면]

[제품 진열 위치 조회 코드]

과제 내용	<p>[사용자 앱] - 관리자 기능 - 재고 관리와 위치 수정</p>  <p>관리자 회원가입 고객 로그인</p>
	<p>[그림 7 사용자 앱의 초기화면]</p> <p>-> 사용자 앱은 관리자 기능과 고객 기능이 있습니다. 등록되어있지 않은 관리자는 회원가입을 통해 앱을 이용할 수 있습니다. 등록되어있는 관리자는 로그인을 통해 앱을 이용할 수 있습니다.</p>  <pre>if(password.equals(passwordCheck)){ Toast.makeText(SignActivity.this, "비밀번호가 같지 않습니다", Toast.LENGTH_SHORT).show(); return; } if(id.isEmpty() password.isEmpty()){ Toast.makeText(SignActivity.this, "아이디 비밀번호를 입력하세요", Toast.LENGTH_SHORT).show(); return; } Log.e("size: ", storeNameList.size() + ""); Log.e("size: ", storeAddressList.size() + ""); if (storeNameList.size() == 0 storeAddressList.size() == 0 (storeNameList.size() != storeAddressList.size())){ Toast.makeText(this, "매장 정보를 바르게 입력하세요", Toast.LENGTH_SHORT).show(); return; } List<Store> storeList = new ArrayList<>(); for(int i = 0; i < storeNameList.size(); i++){ Store(storeNameList.get(i).getText().toString(), storeAddressList.get(i).getText().toString()); Manager newManager = new Manager(null, id, password, storeList); Call<Manager> join = service.join(newManager); join.enqueue(new Callback<Manager>() { @Override public void onResponse(Call<Manager> call, Response<Manager> response) { if(response.isSuccessful()){ Toast.makeText(SignActivity.this, "회원가입 성공", Toast.LENGTH_SHORT).show(); finish(); } } }) }</pre>
	<p>[그림 8 회원가입 화면]</p> <p>[회원가입 실제 사용 코드]</p> <p>-> 회원가입을 하는 화면입니다. 아이디와 비밀번호, 관리하는 매장의 상세정보 (이름, 주소)를 입력해야 합니다. 플러스 버튼을 누르면 가게를 여러 개 등록할 수 있습니다.</p>

과제 내용



[그림 9 매장 선택 화면]

-> 등록된 매장을 선택할 수 있는 화면입니다.



[그림 10 기능 선택 화면]

-> 매장을 선택하면 사용할 수 있는 기능이 나옵니다. 가게 배치, 제품 등록, 제품 수정 및 삭제 기능이 있습니다.



[그림 11 전체 제품 리스트]

-> 제품 등록 버튼을 누르면 전체 제품 리스트가 나옵니다. 이 화면에서 등록할 제품을 선택합니다. 화면에서는 검색 기능을 사용할 수 있습니다.



[그림 12 제품 등록 화면]

-> 제품을 선택하여 등록하는 화면입니다. 이 화면에서는 제품의 이름, 이미지, 가격, 종류를 확인하고 등록할 수 있습니다.



[그림 13 가게 배치 화면]

-> 가게 배치 버튼을 누르면 나오는 화면입니다.

이 화면에서는 매대를 추가, 삭제할 수 있는 버튼과 추가한 매대에 제품을 진열할 수 있는 버튼이 있습니다. 매대 추가 버튼을 누르면 추가할 매대의 시작점과 끝점을 선택해주세요. 라는 문구가 나오고 매대를 추가할 수 있습니다. 매대 삭제 버튼을 누르면 삭제할 매대를 선택해주세요. 라는 문구가 나오고 매대를 삭제할 수 있습니다. 상품 진열 버튼을 누르면 진열할 매대를 선택해주세요. 라는 문구가 나오고 진열할 매대를 선택해 상품을 등록할 수 있습니다.

```
public void onClick(View v) {
    TableRow tableRow = new
    TableRow(getApplicationContext());
    TextView textView = new TextView(getApplicationContext());
    TableRow.LayoutParams layoutParams = new
    TableRow.LayoutParams(
        TableRow.LayoutParams.MATCH_PARENT,
        TableRow.LayoutParams.WRAP_CONTENT);
    layoutParams.gravity = Gravity.CENTER;
    textView.setTextSize(50);
    textView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(LocationDetailPage.this,
                ((TextView)v).getText().toString(), Toast.LENGTH_SHORT).show();
            String text = ((TextView)v).getText().toString();
            if(text.equals("+")){
                Intent intent1 = new Intent();
                addItem(v);
            }
        }
    });
}
```

[실제 사용 코드]



[그림 14 매대 추가 화면]

-> 매대를 추가한 화면입니다.

매대는 관리자가 원하는 위치에 추가할 수 있고 이동도 가능합니다. 제품 진열이 되어 있지 않은 매대에는 진열 X라는 문구가 생성됩니다.



[그림 15 상품 진열 화면]

-> 추가한 매대에 상품을 진열하는 화면입니다. 진열하고 싶은 상품들을 선택합니다.



[그림 16 매대에 상품을 진열한 화면]

-> 관리자가 매대를 추가하고 상품을 진열한 화면입니다.

GS25 천안대점 매장 매대에는 음료와 과자가 진열되어있는 모습을 확인할 수 있습니다.

해당 매대를 누르고 상품 진열 버튼을 누르면 선택한 매대에 있는 모든 제품을 확인할 수 있습니다.

과제 내용



[그림 17 제품 수정 및 삭제 화면]

-> 제품 수정 및 삭제 버튼을 누르면 매장 내에 등록된 제품 리스트 화면이 출력됩니다. 삭제와 수정하고 싶은 제품을 누르면 해당 화면이 출력되고 이 화면에서는 해당 제품의 재고를 수정, 제품을 삭제할 수 있습니다.

```
private void modifyItem(){
    String quantity = itemQuantity.getText().toString();
    nowItem.setQuantity(Integer.parseInt(quantity));
    Call<Item> modifyQuantity = service.modifyItemQuantity(nowItem);
    modifyQuantity.enqueue(new Callback<Item>() {
        @Override
        public void onResponse(Call<Item> call, Response<Item> response) {
            if(response.isSuccessful()){
                Log.e("modify item error", response.errorBody().toString());
                Toast.makeText(ItemModifyDeleteActivity.this, "물건 수정 예러",
                    Toast.LENGTH_SHORT).show();
            }
            Toast.makeText(ItemModifyDeleteActivity.this, "재고 수정이 완료되었습니다.",
                Toast.LENGTH_SHORT).show();
        }
    });
    private void deleteItem(){
        Call<Item> deleteItem = service.deleteItem(nowItem.getId());
        deleteItem.enqueue(new Callback<Item>() {
            @Override
            public void onResponse(Call<Item> call, Response<Item> response) {
                if(response.isSuccessful()){
                    Log.e("delete item error", response.errorBody().toString());
                    Toast.makeText(ItemModifyDeleteActivity.this, "물건 삭제 예러",
                        Toast.LENGTH_SHORT).show();
                }
                Toast.makeText(ItemModifyDeleteActivity.this, "물건 삭제 완료!",
                    Toast.LENGTH_SHORT).show();
                finish();
            }
        });
    }
}
```

[실제 사용 코드]

```
private void getAllCenterItem(){
    Call<List<CenterItemResponse>> getAllItem =
    service.findAllCenterItem();
    getAllItem.enqueue(new Callback<List<CenterItemResponse>>() {
        @Override
        public void onResponse(Call<List<CenterItemResponse>> call,
            Response<List<CenterItemResponse>> response) {
                if(response.isSuccessful()){
                    Log.e("getAllItem Error", response.errorBody().toString());
                    Toast.makeText(ManagerSearchCenterItemActivity.this, "아이템
                    리스트 가져오기 예러", Toast.LENGTH_SHORT).show();
                    return;
                }
                Toast.makeText(ManagerSearchCenterItemActivity.this, "리스트
                    업 성공!", Toast.LENGTH_SHORT).show();
                centerItemList = response.body().stream()
                    .map(ci ->new CenterItem(ci.getId(), ci.getName(),
                    ci.getType(), ci.getPrice(), ci.getImage(), ci.getBarcode()))
                    .collect(Collectors.toList());
                setItemRecyclerView();
            }
        });
}
```

[제품 등록하는 코드]

과제 내용

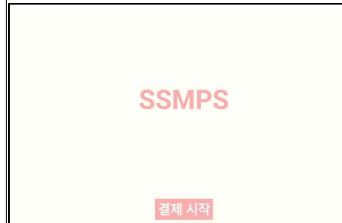
```
private void searchItem(){
    String itemName = itemNameInput.getText().toString();
    Log.e("itemName", itemName);
    Call<List<CenterItemResponse>> findItemByName =
    service.findCenterItemByName(itemName);
    findItemByName.enqueue(new Callback<List<CenterItemResponse>>()
    {
        @Override
        public void onResponse(Call<List<CenterItemResponse>> call,
            Response<List<CenterItemResponse>> response) {
                if(response.isSuccessful()){
                    Log.e("get search item", response.errorBody().toString());
                    Toast.makeText(ManagerSearchCenterItemActivity.this, "아이템
                    검색 예러", Toast.LENGTH_SHORT).show();
                    return;
                }
                Toast.makeText(ManagerSearchCenterItemActivity.this, "아이템
                    검색 성공", Toast.LENGTH_SHORT).show();
                centerItemList = response.body().stream()
                    .map(ci ->new CenterItem(ci.getId(), ci.getName(),
                    ci.getType(), ci.getPrice(), ci.getImage(), ci.getBarcode()))
                    .collect(Collectors.toList());
                setItemRecyclerView();
            }
        });
}
```

[제품 검색하는 코드]

과제 내용

- 앱 개발:

[결제 앱]



[그림 1 결제 앱의 초기화면]

```
( 결제 시작 버튼 클릭 시 리스너를 통한 화면 이동 코드 )
startButton.setOnClickListener(view ->new Thread() -> {
    new Thread(new ServerListener(receiver)).start();
    Intent intent =new Intent(getApplicationContext(), start.class);
    startActivity(intent);
}).start()
```

- Receiver를 이용해서 Jetson Nano 카메라에서 사람이 인식될 경우 결제 화면으로 자동으로 이동하는 신호를 받을 경우, 화면이 그림3으로 이동할 수 있도록 하였습니다.
- 또는 사람 인식이 실패했을 경우, 결제 시작 버튼을 사용자가 임의로 누르게 되어도 그림3 화면으로 이동할 수 있습니다.

상품명	가격	수량	총 가격	
코카콜라	2000	8	16000	1개
맛있는물	3000	9	27000	1개
배달료	4000	12	48000	1개
				총 금액: 91000

[그림 2 결제 상품 및 금액 표시 화면]

```
( 결제시 네트워크 통신으로 정보를 전달하는 코드 )
new Thread() -> {
    Socket client;
    try {
        client =new Socket();
        InetSocketAddress ip =new InetSocketAddress(ip, port);
        client.connect(ip);
        sender = client.getOutputStream();
        receiver = client.getInputStream();
        JSONObject jsonObj =new JSONObject();
        for (int i =0; i < items.size(); i++) {
            jsonObj.put(items.get(i).getText(), items.get(i).getPrice()+"-"+items.get(i).getCnt());
        }
        System.out.println(jsonObj);
        String msg = jsonObj.toString();
        byte[] data = msg.getBytes();
        ByteBuffer b = ByteBuffer.allocate(4);
        b.order(ByteOrder.LITTLE_ENDIAN);
        b.putInt(data.length);
        sender.write(b.array(), 0, 4);
        sender.write(data);
    } catch (Throwable e) {
        e.printStackTrace();
    }
}.start();
```

해당 코드는 Jetson Nano 와 Socket 통신을 이용하여 통신하는 코드입니다.

Socket을 생성하여, sender 와 receiver를 지정하고, 입력받은 items의 값을 JSON 형태로 변환하여 Spring boot 서버로 Json 형태로 전달하기 위함입니다.

그리고 JSON을 toString()을 이용하여 문자열 형태로 변환 후, 해당 JSON의 길이를 먼저 4byte로 전송한 이후, byte 배열인 data를 전송합니다.

Socket의 Serve가 되는 곳에서 해당 Message 의 Length를 알 수 있기 때문에 data를 필요한 크기만 정확히 받을 수 있습니다.

과제 내용

```
( 결제시 입력받은 정보로 결제 앱에 대해서 )
public void addItem(String text, int price, int cnt) {
    for (ListItem item : items) {
        if (item.getText().equals(text)) {
            item.setCnt(item.getCnt() + cnt);
            return;
        }
    }
    items.add(new ListItem(text, price, cnt));
}
```

- Jetson Nano에서 카메라를 통해 입력받은 상품 정보를 Jetson Nano에서 Android 결제 앱으로 전송받아, 전송 받은 정보를 addItem을 이용하여 결제 화면에 띄우는 형태입니다.
- 결제 시, addItem에 추가되어있던 상품들의 정보를 Jetson Nano로 전송하여, DB에 있는 재고의 값을 수정합니다

- 상품 인식 AI: Jetson nano의 카메라를 통해서 결제할 상품을 인식합니다.

1. 모델선택

모델을 선택할 때 중요하게 생각한 부분은 정확하면서도 실시간 처리가 가능해야 했습니다. 여러 모델 중에서 YOLO가 제일 적합하다고 판단하였으며, YOLO 버전 중에 제일 best인 yolov7을 선택하였습니다.

2. 데이터셋

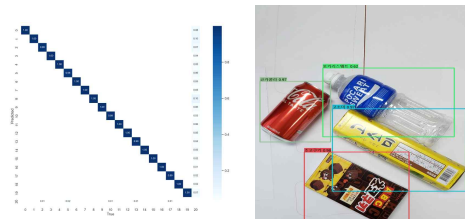
- 2.1. ai hub [상품이미지](#)에서 총 20개의 상품을 선택하여 학습을 진행했습니다.
- 2.2. 이미지의 크기가 2988x2988로 커서 yolo 모델을 사용할 때 가장 적합하다는 이미지 사이즈인 416x416으로 변경하여 진행했습니다.
- 2.3. 기존의 라벨링, 구성되어있는 것이 PascalVOC 데이터 세트의 형식이라 YOLO 형식으로 변환하였습니다.
- 2.4. 기존에 있던 상품 데이터 셋으로는 학습이 덜 되어서 image augmentation을 하였고, 데이터셋은 train:8, validation:1, test:1로 나누어서 진행하였습니다.

3. 학습

- 3.1. yolov7 모델에 custom한 데이터를 사용하기 위한 각종 yaml 파일을 변경하였습니다.
- 3.2. 학습을 진행하기 전에 좋은 하이퍼 파라미터를 찾는 방법인 ‘—evolve’ 인자값을 사용하여 먼저 간단한 학습을 진행 후 만들어진 최적의 하이퍼 파라미터로 세팅하였습니다.
- 3.3. 학습을 진행할 때는 미리 만든 416x416 이미지를 사용하였고, test/detect를 진행할 때는 640x640 사이즈로 바꾸기 위해 ‘--img 640’ 인자값을 두어서 진행하니 더 좋은 결과가 나타났습니다.

4. 결과

아래의 test confusion matrix 이미지를 보면 각각의 클래스가 background 부분에서 fn, fp가 발생하는 것을 볼 수 있습니다. 해당하는 문제를 해결하기 위해서 background 이미지를 따로 만들어 학습하는 것을 진행하였지만 똑같은 결과가 나타나서 해당 결과도 나쁘지 않다고 생각해 마무리 지었습니다.



[그림 1 상품 인식 test confusion matrix] [그림 2 상품 인식 detect 이미지]

또한, 학습한 데이터가 대부분 흰색 배경에 noise 없는 데이터라 흰색의 배경일 때 잘 학습이 되었습니다. background 색깔에 영향을 안 받기 위해서 OpenCV를 이용해 detect 하려는 이미지의 배경을 하얗게 바꿔보기도 했습니다.



[그림 3 영상 처리한 detect 이미지]

아쉽고 알게 된 점: 다양한 데이터 세트 형식(PascalVOC, yolo, CreateML)의 라벨링 차이점을 자세히 알게 되었습니다. 아쉬웠던 점은 코랩 지원이 안되어 컴퓨팅 자원이 부족하다 보니까 많은 학습과 다양한 모델을 사용해보지 못해 아쉬움이 남습니다.

- **얼굴 인식 AI(이후에 yolo 모델이 리소스를 많이 차지해서 Cascade로 변경):** Jetson nano의 카메라를 통해서 얼굴을 인식합니다.

1. 모델선택

상품 인식과 동일한 이유로 yolov7으로 진행했습니다.

2. 데이터셋

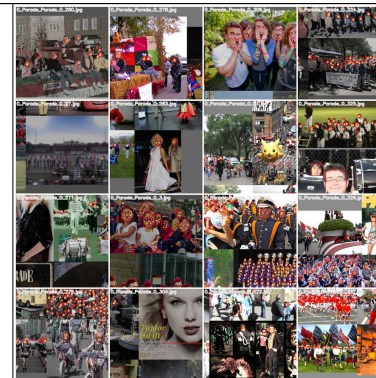
[widerface](#) 사이트 이미지를 사용하여 이미지 사이즈 416x416으로 학습을 진행하였습니다.

3. 학습

상품 인식과 마찬가지로 각종 yaml 파일을 변경하였고, 학습을 진행할 때는 416x416 이미지, test/detect를 진행할 때는 '--img 640' 인자값을 두어서 학습하였습니다.

4. 결과

아래의 test 이미지와 같이 얼굴을 잘 인식하는 것을 볼 수 있습니다.



[그림 1 얼굴 인식 test 이미지]

상품 및 제품화 가능성:

이 시스템은 무인 매장, 편의점 등 다양한 상업적인 환경에서 제품화 가능성이 있습니다.

시스템의 인식 기능과 데이터베이스 기능은 다양한 분야에서도 적용 가능하며, 다양한 응용 분야에 활용될 수 있습니다.

기대효과:

매장 관리의 효율성 향상:

실시간 재고 및 위치 확인 시스템은 매장 관리를 효율적으로 만들어 줄 것입니다. 이는 재고 부족 및 재배치에 따른 시간과 노력을 줄여주며, 매장 운영 비용을 절감하는 데 도움이 될 것으로 예상 됩니다.

고객 경험의 향상:

빠르고 편리한 결제 과정과 실시간 재고 확인을 통해 고객에게 다양한 정보를 제공하게 됩니다.

이는 고객 만족도를 향상시킬 수 있습니다.


무인 매장의 운영 효율성 향상:

이 시스템은 무인 매장에서 특히 유용할 것입니다.

매장 관리자는 상품의 재고 및 위치를 실시간으로 관리하고, 고객은 상품을 쉽게 찾고 결제할 수 있게 됩니다.

이를 통해 매장의 운영에 효율성이 향상될 수 있습니다.

결과물의 활용방안 및 기대효과

수행 방법	구분	성명	과제 참여 내용(역할)
	팀장	조은	총괄 및 하드웨어
	팀원	권민우	서버 개발 (Spring boot), DB 설계 및 관리(MySQL)
	팀원	진민주	인공지능(facial recognition system), 결제화면 디자인
	팀원	한기수	안드로이드 개발(FrontEnd)
	팀원		
	팀원		
결과물			
	활동사진		활동사진
			
