**종합설계 프로젝트 수행 보고서**

|  |  |
| --- | --- |
| 프로젝트 명 | **“딥러닝 기반 식물 식별 서비스”** |
| 팀 번호 | S2-9 |
| 문서 제목 | **수행계획서( )**  **2차발표 중간보고서( )**  **3차발표 중간보고서( )**  **4차발표 중간보고서( )**  **최종결과보고서(o)** |

**2021.11.14**

**S2-9 팀원 : 배기선 (팀장)**

**주우재 (팀원)**

**지도교수 : 박정민 교수**

**지도교수 : 박정민 교수**

# 문서 수정 내역

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **작성일** | **대표작성자** | **버전(Revision)** | **수정내용** |  |
| 2020.11.11. | 배기선, 김지원, 주우재 | 1.0 | 1차 발표자료 작성 | 최초 작성 |
| 2020.11.12. | 배기선, 김지원, 주우재 | 1.1 | 1차 발표자료  내용 추가 | 개발 환경 변경 |
| 2020.12.28. | 김지원 | 2.0 | 1차 계획서 작성 | 최초 작성 |
| 2020.12.28. | 배기선, 김지원 | 2.1 | 1차 계획서  내용 추가 | 개발 환경 추가 |
| 2021.02.18. | 배기선, 주우재 | 3.0 | 2차 발표자료 작성 | 최초 작성 |
| 2021.02.19. | 김지원 | 3.0 | 수행 보고서 작성 | 최초 작성 |
| 2021.03.01. | 배기선, 김지원, 주우재 | 3.1 | 2차 발표자료  내용 추가 | 내용 추가 |
| 2021.05.07 | 배기선 | 4.0 | 수행 보고서 추가 작성 | 추가 작성 |
| 2021.11.08 | 배기선, 주우재 | 5.0 | 최종보고서 작성 | 내용 수정 및 추가 |
| 2021.11.14 | 배기선, 주우재 | 5.1 | 최종보고서 추가 작성 | 내용 추가 |

**이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계”교과목에서 프로젝트 “딥러닝 기반 식물 식별 서비스”를 수행하는 (S2-9, 배기선, 김지원, 주우재)들이 작성한 것으로 사용하기 위해서는 팀원들의 허락이 필요합니다.**

**목 차**

**Ⅰ. 서론**

1. 작품선정 배경 ································································································································· 4

2. 기존 시장/기술동향 분석 ············································································································ 5

3. 개발 목표 ········································································································································· 6

4. 팀 역할 분담 ··································································································································· 6

5. 개발 일정 ········································································································································· 6

6. 개발 환경 ········································································································································· 6

**Ⅱ. 본론**

1. 개발 내용 ········································································································································· 7

2. 문제 및 해결방안 ··························································································································· 7

3. 시험시나리오 & 시스템 구성도 ·································································································· 7

4. 상세 설계 ···································································································································· 8

5. Prototype 구현 ·························································································································· 17

6. 시험/ 테스트 결과 ····················································································································· 19

7. Coding & DEMO ······················································································································· 20

**Ⅲ. 결론**

1. 연구 결과 ······································································································································ 34

2. 작품제작 소요재료 목록 ············································································································· 34

참고문헌 ···················································································································································· 35

# Ⅰ. 서 론

## 작품 선정 배경

**-선물 받은 알 수 없는 식물의 정보 확인**

|  |  |
| --- | --- |
| * 집에서 미관 목적이나 공기 정화 등의 목적을 지닌 키우기 쉬운 식물을 선물 받았을 경우 식물의 이름을 알지 못 하는 상황에서 사용 * 선물 받은 식물의 정보가 기억나지 않는 상황에서 사용 | C:\Users\user\Desktop\commonJ2V7TTVQ.jpg |

**-길에서 발견한 식물의 정보 확인**

|  |  |
| --- | --- |
| * 길가에 피어 있는 이름을 알 수 없는 식물에 대한 정보를 알고 싶은 상황에서 사용 | https://search.pstatic.net/common/?src=http%3A%2F%2Fblogfiles.naver.net%2F20121219_111%2Ftpmav00_1355897358667NwBCR_JPEG%2FuT4tni_UGmHzzoiREzrSxQ.jpg&type=sc960_832 |

## 기존의 시장/기술동향 분석

1. LG V30S 씽큐

|  |  |
| --- | --- |
| * 카메라로 꽃을 비추면   수억 개의 데이터를 기반으로 한 인공지능이 피사체와 가장 비슷한 사물들의 이름을 나열 | https://search.pstatic.net/common/?src=http%3A%2F%2Fimgnews.naver.net%2Fimage%2F016%2F2018%2F02%2F26%2F20180226000011_1_20180226071651565.jpg&type=sc960_832 |

1. 네이버 이미지 인식 서비스 베타버전

|  |  |
| --- | --- |
| * 카메라로 사물을 찍으면 이미지 인식서비스가 이를 인지하여   쇼핑 또는 다양한 서비스로 연계 | https://search.pstatic.net/common/?src=http%3A%2F%2Fimgnews.naver.net%2Fimage%2F138%2F2017%2F07%2F11%2F0002052359_001_20170711103014835.jpg&type=sc960_832 |

1. 다음 어플리케이션 사물 검색

|  |  |
| --- | --- |
| * 사물 검색은 이용자가 도서, 영화 포스터, 주류 라벨 등을   카메라로 촬영하면 해당 사물을 인식해 바로 검색 결과로 제공 | https://search.pstatic.net/common/?src=http%3A%2F%2Fimgnews.naver.net%2Fimage%2F014%2F2011%2F08%2F11%2F0922389143_0.jpg&type=sc960_832 |

## 개발 목표

- 누구나 핸드폰 하나로 식물을 쉽게 찾을 수 있게 한다.

- 식물 관리에 필요한 물 주기, 식물에 맞는 영양제, 다양한 식물 정보 등과

같은 도움을 줄 수 있도록 한다.

## 팀 역할 분담

|  |  |  |
| --- | --- | --- |
| **번호** | **성명** | **주요 역할** |
| 1 | 배기선 | 안드로이드 서버 통신, Json 데이터 파싱, 즐겨찾기 구현 |
| 2 | 주우재 | Tensorflow 모델 구현, TF.Lite 모델 적용, 이미지 변환, 안드로이드 서버 통신, 최근조회 구현 |

## 개발 일정



## 개발 환경

- 개발 언어 : Python, Java

- 사용 라이브러리 : Tensorflow 2.0

- 개발 환경 : 파이참, 아나콘다, 주피터, Android Studio

- DB : 직접 수작업으로 입력 또는 공공데이터 포탈

# Ⅱ. 본 론

## **개발 내용**

- 사진을 찍어서 식물을 식별

- 식별된 식물의 정보 제공

- 식물의 이름으로 검색

- 관리자 웹페이지로 식물 정확도를 확인

- 관리자 웹페이지로 사용자들의 앱 만족도 확인

## **문제 및 해결방안**

가. 사진을 찍어서 식물을 식별

→ Python 기반의 Tensorflow 활용

나. 식별된 식물의 정보 제공

→ 수작업으로 입력하거나 공공데이터 포탈의 데이터베이스 활용

다. 식물의 이름으로 검색

→ 수작업으로 입력하거나 공공데이터 포탈의 데이터베이스 활용

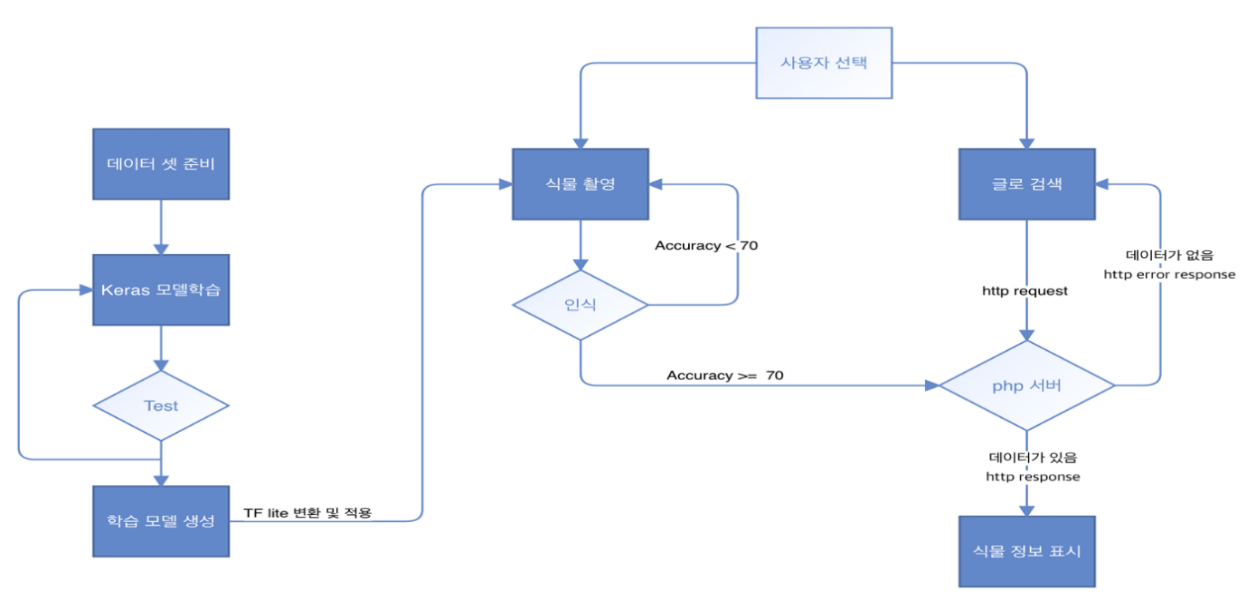
라. 관리자 웹페이지로 식물 정확도를 확인

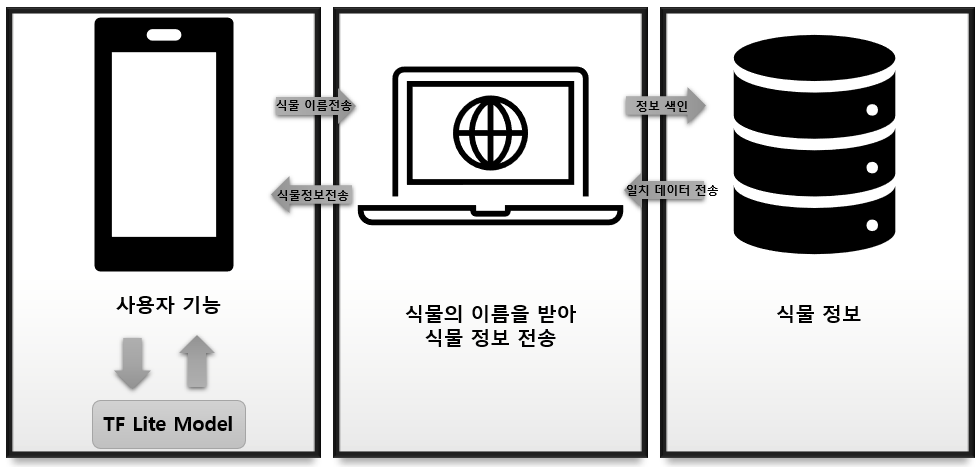
→ 이클립스, 톰캣

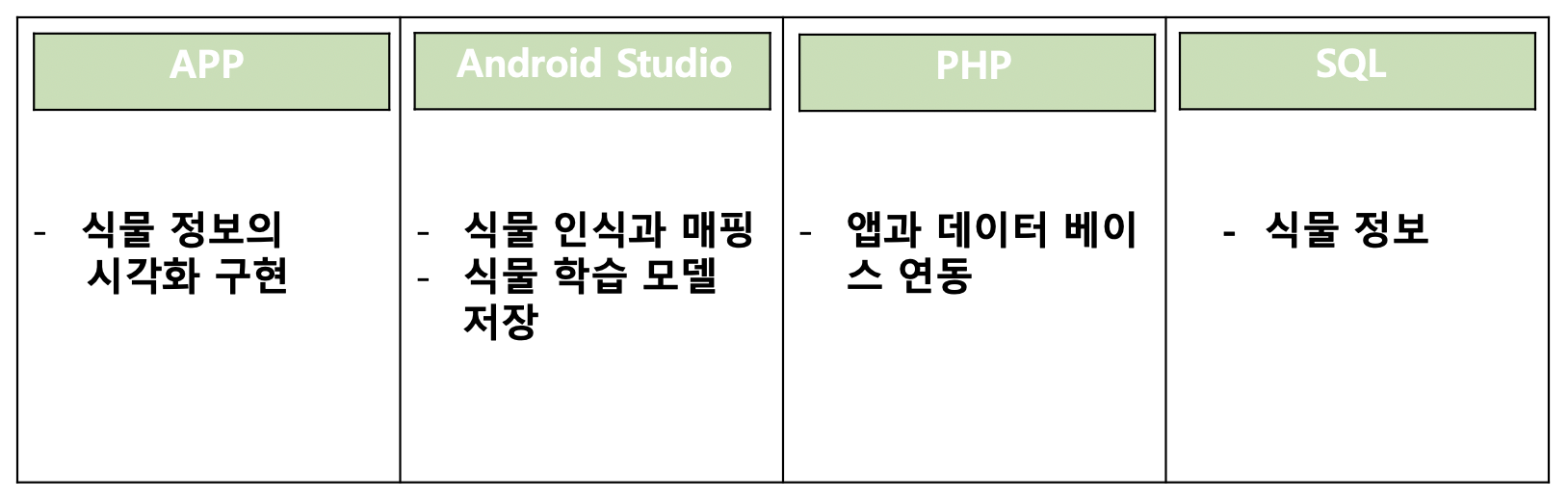
1. 관리자 웹페이지로 사용자들의 앱 만족도 확인

→ 이클립스, 톰캣

## **시험 시나리오 & 시스템 구성도**







## **상세 설계**

데이터베이스  
1) Flower\_DB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | name | feature | management | notice |

꽃의 정보를 저장하는 db이다. Id를 부여하고 식물의 이름, 특징, 관리방법, 기타 알림 사항 등을 알려준다.  
2) Favorite\_db

|  |  |  |
| --- | --- | --- |
| email | plant\_id | plant\_name |

즐겨찾기한 식물을 저장하는 db이다. 로그인한 유저의 email, 식물 이름, 식물 id를 저장한다.  
3) User\_DB

|  |  |  |
| --- | --- | --- |
| uid | uemail | upassword |

User의 정보를 담는 db이다. Uid, 유저 이메일, 유저 비밀번호 등을 저장한다.

● 서버 php

1. Psearch.php  
   검색하는 식물의 이름을 받아 db에서 해당하는 식물의 정보를 출력
2. loadDB.php  
   카메라로 찍은 식물의 이름을 받아 db에서 해당하는 식물 정보 출력
3. Bregister.php  
   로그인한 이메일을 받고 즐겨찾기 할 식물의 이름을 받아서 즐겨찾기에 삽입
4. Bdelete.php  
   즐겨찾기 란에서 삭제할 식물의 이름을 받아 해당 식물 삭제
5. 로그인, 최근조회  
   Firebase를 활용하여 로그인과 최근조회 구현

App

1) Login

|  |
| --- |
| Login 모듈 |
| **1.기능** |
| 로그인 기능 |
| **2.다루는 정보** |
| Firebase를 통하여 userid, userpw 입력 후 로그인 |

1) Register

|  |
| --- |
| Register 모듈 |
| **1.기능** |
| 회원가입 기능 |
| **2.다루는 정보** |
| Firebase를 통하여 useremail, userpw, usercheckpw 등을 입력 후 회원가입 |

|  |  |
| --- | --- |
| onSupportNavigateUp | |
| **형식** | Public Boolean onSupportNavigeteUp |
| **반환 값** | 0,1 |
| **설명** | 회원가입에서 로그인으로 뒤로가기 |

2) Recent Check

|  |
| --- |
| Recent Check 모듈 |
| **1.기능** |
| 최근조회 기능 |
| **2.다루는 정보** |
| 사용자가 카메라, 검색으로 식물을 인식 할 경우 해당 정보에서 이름을 리스트로 저장 |
| **3.수행 정보** |
| 식물 인식할 때마다 |

|  |  |
| --- | --- |
| onQueryTextSubmit | |
| **형식** | Public boolean onQueryTextSubmit |
| **반환 값** | 0,1 |
| **설명** | 식물 검색어를 보냄 |

|  |  |
| --- | --- |
| Menu Event | |
| **menurecent** | 최근조회 메뉴 관련 이벤트처리 |
| **menucamera** | 카메라 메뉴 관련 이벤트 처리 |
| **menugallery** | 앨범 메뉴 관련 이벤트 처리 |
| **menubookmark** | 즐겨찾기 메뉴 관련 이벤트처리 |

3) PlantInfo

|  |
| --- |
| PlantInfo 모듈 |
| **1.기능** |
| 식물정보 보여주기 기능 및 사용자와 서버 간의 통신 |
| **2.다루는 정보** |
| 유를 통해 받은 식물의 이름, 키우는 방법, 특징, 기타  알림 등을 알려줌 |

|  |  |
| --- | --- |
| plant | |
| **형식** | Public void plant( final String name) |
| **반환 값** | 없음 |
| **설명** | 서버와의 연결 또한 검색한 이름, 찍은 사진의 이름을 서버로 전송 |

|  |  |
| --- | --- |
| showJSONResult | |
| **형식** | Public void showJSONList( string response) |
| **반환 값** | Json result |
| **설명** | post형식으로 서버로부터 정보를 Object로 파싱하고 원하는 위치에 정보를 놓음 |

4) User, Classifier

|  |
| --- |
| User 모듈 |
| **1.기능** |
| 유저 정보를 담아두는 기능 |
| **2.다루는 정보** |
| 즐겨찾기를 위한 유저 이메일, 이름 정보를 담아둠 |

|  |
| --- |
| Classifier 모듈 |
| **1.기능** |
| 분류한 식물의 정보를 담는 기능 |
| **2.다루는 정보** |
| 식물 id, 이름, 정확도 등의 정보를 담아둠 |

5) Bookmark

|  |
| --- |
| Bookmark 모듈 |
| **1.기능** |
| 관심있는 식물 정보를 저장하고 담아두는 기능 |
| **2.다루는 정보** |
| 즐겨찾기를 추가하며, 삭제할 수 있고 즐겨찾기에는 식물 이름이 담겨있음 즐겨찾기의 식물 이름을 클릭하면 해당 식물 페이지로 넘어가고 즐겨찾기의 식물 이름을 꾹 누르면 삭제할 수 있음 |

|  |  |
| --- | --- |
| bplant | |
| **형식** | private void bplant( String fname, String femail) |
| **반환 값** | 없음, 이름 이메일 서버 전송 |
| **설명** | 식물 정보란에서 즐겨찾기 모양을 클릭하면 서버와 연결되며 이름과 이메일이 서버로 전송되고 즐겨찾기 등록 |

|  |  |
| --- | --- |
| ldelete | |
| **형식** | private void ldelete( string fname) |
| **반환 값** | 없음, 해당 이름삭제 |
| **설명** | 서버와 연결되며 해당 이름의 식물을 즐겨찾기에서 제거함 |

|  |  |
| --- | --- |
| plant | |
| **형식** | private void plant( final String femail) |
| **반환 값** | Json result |
| **설명** | 서버와의 연결하며 해당 이메일의 db에서 즐겨찾기 목록을 보여줌 |

|  |  |
| --- | --- |
| Click Event | |
| **listView.setOnItem**  **ClickListener** | 즐겨찾기 리스트에서  클릭 시 발생하는 이벤트 |
| **설명** | 즐겨찾기에서 클릭시 해당 식물 정보란으로 이동 |
| **listView.setOnItem**  **LongClickListener** | 즐겨찾기 리스트에서 오래 클릭 시 발생하는 이벤트 |
| **설명** | 즐겨찾기에서 길게 클릭시 해당 식물 db에서 삭제 |

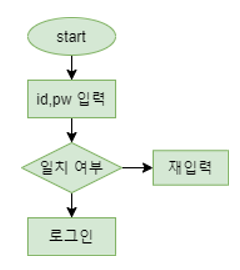
6) Camera

|  |
| --- |
| Camera 모듈 |
| **1.기능** |
| 카메라로 식물 사진을 찍고 식별하는 기능 |
| **2.다루는 정보** |
| 안드로이드에 저장된 Tensorflow lite를 이용하여 식물을 식별하고 식별된 식물을 보여줌 |

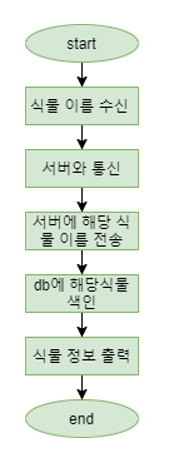
|  |  |
| --- | --- |
| initTensorflowAndLoalModel | |
| **형식** | private void initTensorflowAndLoadM() |
| **반환 값** | 없음, 초기화 |
| **설명** | 텐서플로우를 초기화하는 메소드 |

|  |  |
| --- | --- |
| onImage | |
| **형식** | Public void onImage(  CameraKitImage cameraKitImage) |
| **반환 값** | 식별된 식물 이름, 정확도 |
| **설명** | 카메라로 찍은 이미지를 텐서플로에 보내서 이미지를 인식하고 식물이름, 정확도를 보여줌 |

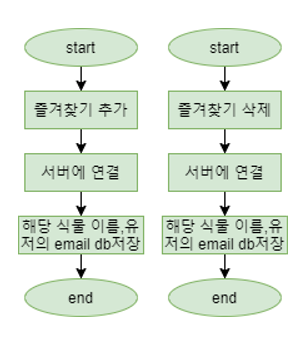
|  |  |
| --- | --- |
| OnActivityResult | |
| **형식** | proteccted void onAactivityResult (  int requestcode, int resultcode, Intent data) |
| **반환 값** | ArrayList로 리턴 |
| **설명** | 이미지피커에서 선택된 이미지를 텐서플로우에 전송 및 해당 식물이름을 다른 액티비티로 전송 |

1. **로그인**

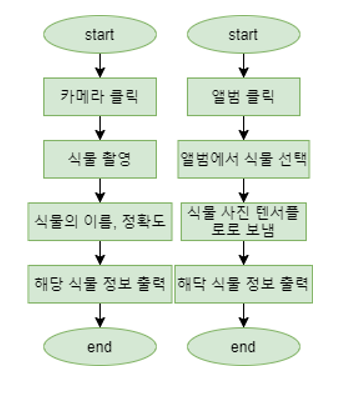
로그인 버튼을 누르면 Firebase의 db회원정보를 확인하고 해당 이메일이 존재하고 비밀번호가 일치하면 접속, 만약 틀리거나 계정이 존재하지 않는다면 해당 메시지가 출력된다.

1. **식물정보**

검색어를 통한 식물 이름, 사진을 찍어서 텐서플로우로 식별된 식물의 이름 등을 통해 서버와 db로식물의 정보를 가져온다. 이때 다른 사용자가 활용할 수 있게 하기위해 외부포트 80번을 개방하고 서버컴퓨터의 ip는 본래 공유기의 ip를 입력한다. 만약 식별된 식물이 없다면 해당 사진은 없다고 나온다.

1. **즐겨찾기**

식물 정보 액티비티에서 해당 식물이 마음에 든다면 즐겨찾기를 추가할 수 있다 이때 유저의 email과 식물의 이름을 db로보내어 저장하며 사용자의 즐겨찾기 란에서 확인이 가능하다.  
삭제의 경우는 즐겨찾기 란에서 가능한데 식물의 이름을 db로보내고 해당 이름의 식물을 삭제하면 사용자의 즐겨찾기란에서 바로 삭제된다.

1. **카메라**

카메라 촬영을 클릭하면 카메라가

켜진다. 그 후 식물을 찍으면 식물의

사진을 224x224 크기로 맞추어

텐서플로로 전송한다. 텐서플로

라이트는 그 사진으로 학습된 데이터를

바탕으로 식물을 판단하고 그

해당결과를 보내준다. 정화도와 이름을

보내주는데 식물의 이름을 식물 정보로

보내어 식물 정보를 나타나게 해준다.

앨범을 선택하면 앨범이 켜지고 카메라

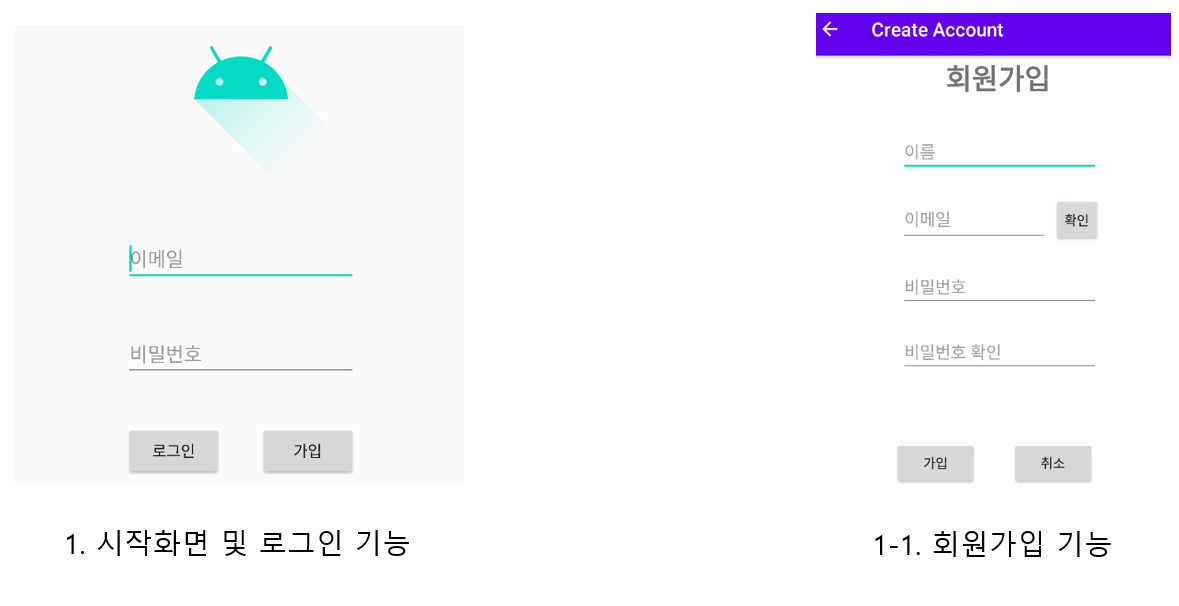
촬영과 비슷한 루틴으로 작동된다.

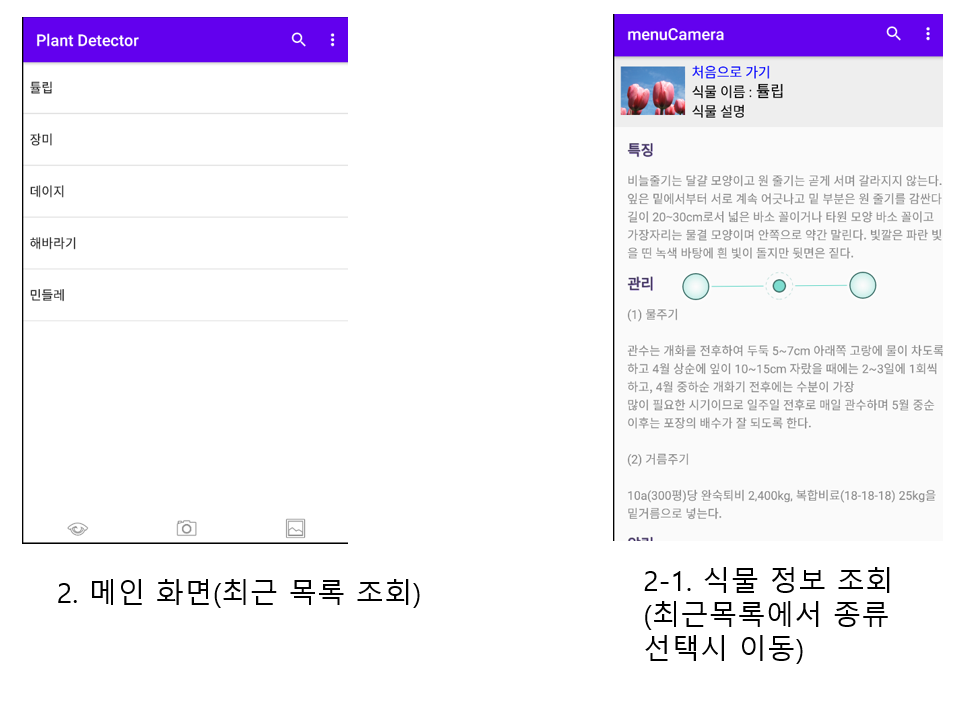
1. 텍스트, 시계이(가) 표시된 사진

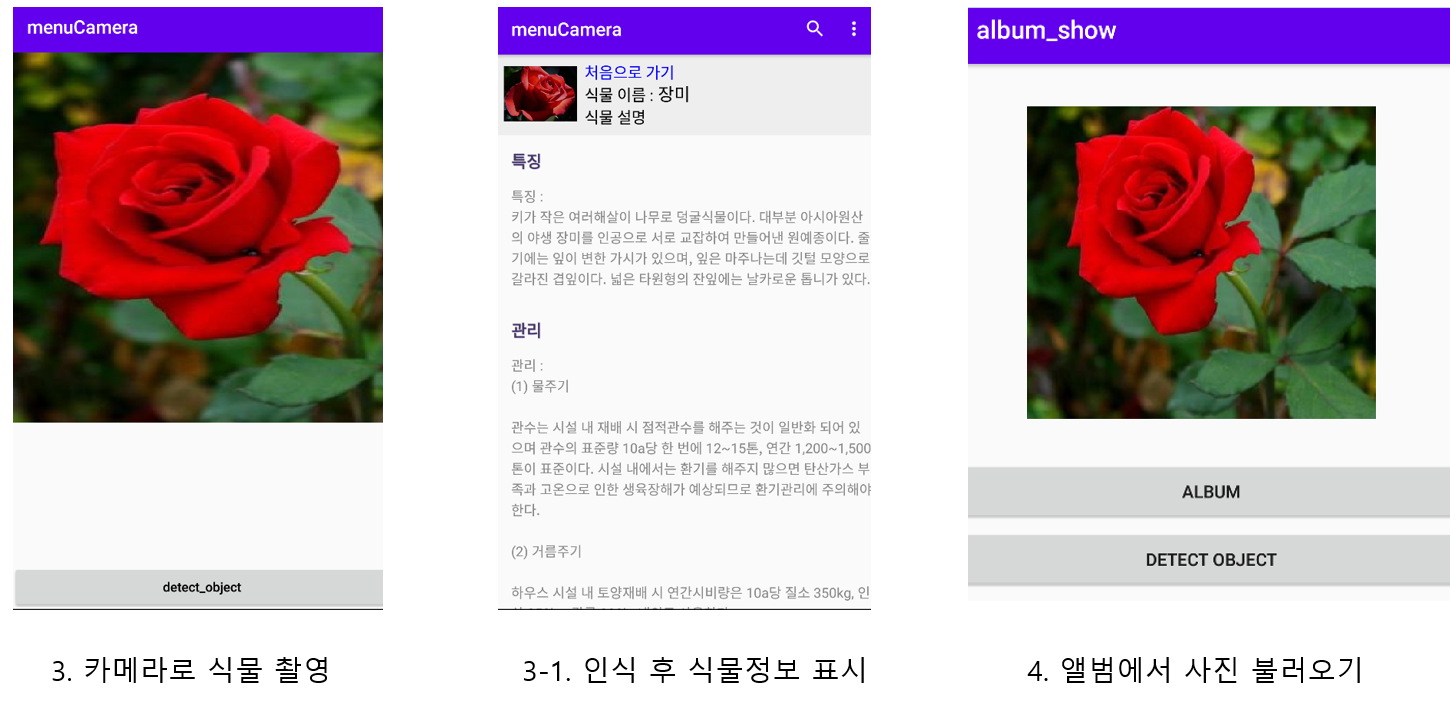
   자동 생성된 설명**최근조회**

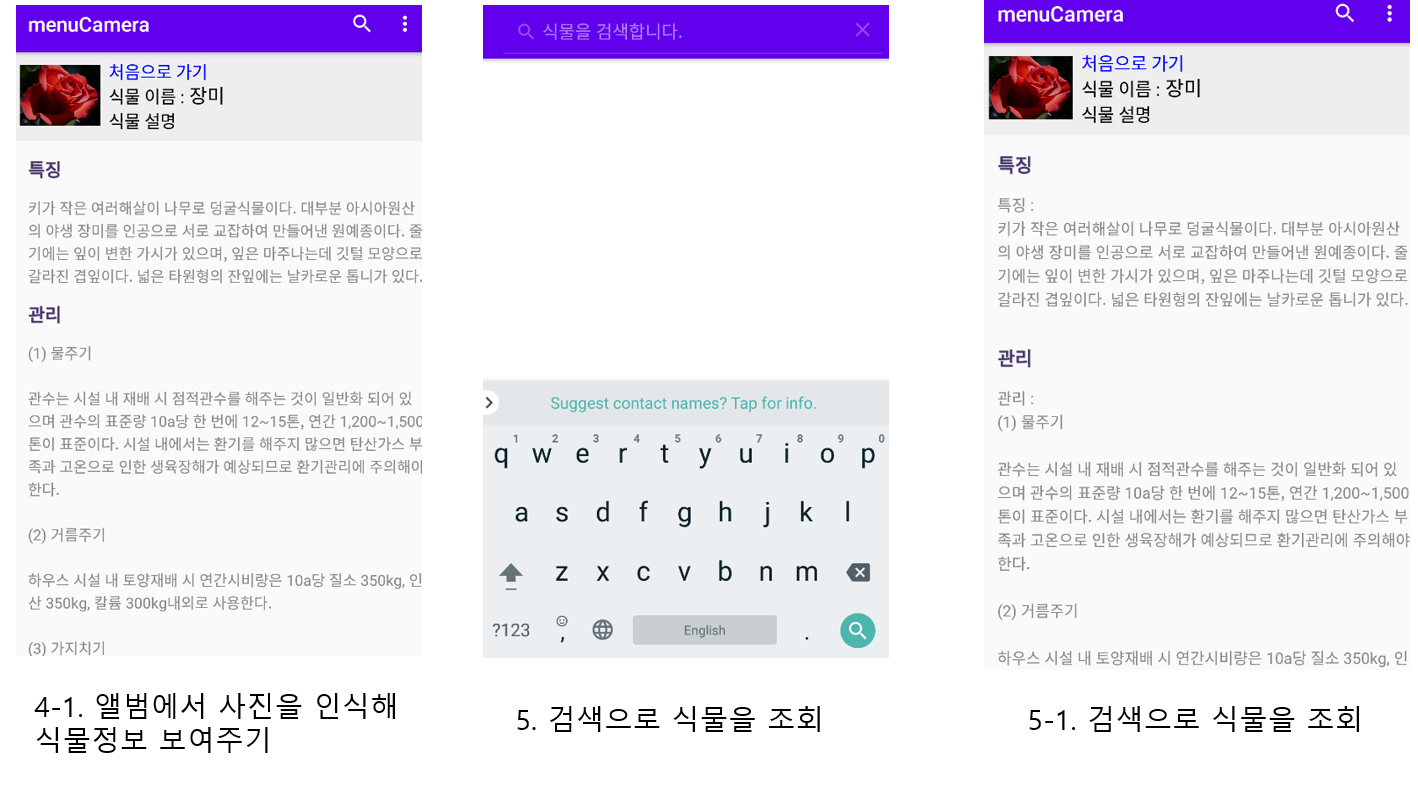
카메라로 식물을 촬영하거나, 앨범으로 식물을 선택하거나, 검색창에서 식물을 검색하면 해당 정보들은 최근조회 목록에 저장된다. 그리고 최근조회목록을 클릭하면 손쉽게 다시 해당식물로 찾아갈 수 있다.

5. 프로토타입 구현









1. 시작화면

어플리케이션 실행시 로그인창이 메인화면이며 회원 가입이 가능

2. 메인화면

최근조회, 카메라 인식, 앨범 인식, 검색으로 조회

3. 최근조회(하단 눈 모양의 버튼)

본인이 최근에 검색했던 또는 촬영했던 식물 이름이 표기

4. 카메라(하단 카메라 버튼)

스마트폰 카메라를 이용하여 식물을 촬영하고 해당 식물조회

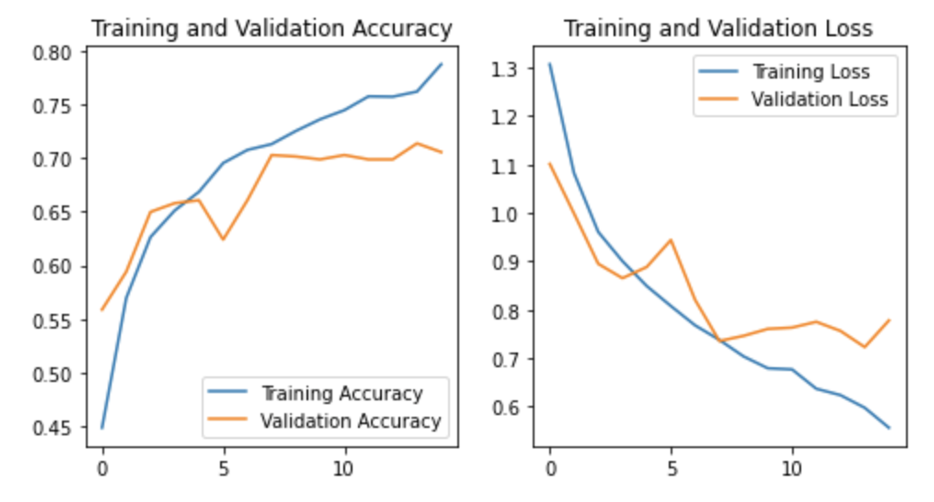
5. 앨범(하단 앨범 버튼)

사용자의 갤러리를 통해 식물을 조회

시험 테스트 결과







PlantInfo2.java

public class PlantInfo2 extends Camera\_activity {  
  
 String tcname;  
 ImageView imageview;  
  
 private FirebaseDatabase firebaseDatabase = FirebaseDatabase.*getInstance*();  
 private DatabaseReference databaseReference = firebaseDatabase.getReference();  
 FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*plant\_info*);  
 TextView first = findViewById(R.id.*first*);  
 imageview = findViewById(R.id.*image*);  
  
 //식물이름 데이터 받아오기  
 Intent intent = getIntent();  
 tcname = intent.getStringExtra("value2");  
 plant(tcname);  
 databaseReference.child("Users").child(user.getUid()).child("recent").push().setValue(tcname);  
 }  
  
 public boolean onCreateOptionsMenu(Menu menu) {  
 MenuInflater inflater = getMenuInflater();  
 inflater.inflate(R.menu.*menu\_option*, menu);  
 return true;  
 }  
  
 public void onClick(View v){  
 Intent intent = new Intent(PlantInfo2.this, MainActivity.class);  
 switch(v.getId()){  
 case R.id.*first*:  
 startActivity(intent);  
 break;  
 default:  
 break;  
 }  
 }  
  
 public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.*report*:  
 Toast.*makeText*(this, "보고", Toast.*LENGTH\_SHORT*).show();  
 return true;  
 case R.id.*recent*:  
 return true;  
 default:  
 return super.onOptionsItemSelected(item);  
 }  
 }  
  
 private void plant(final String name) {  
 // 1. RequestQueue 생성 및 초기화  
 RequestQueue requestQueue = Volley.*newRequestQueue*(this);  
 //String url = "http://192.168.121.46/plantimage.php";  
 //String url = "http://115.143.180.87:8000/plantimage.php";  
 String url = "http://52.79.250.50/plantimage.php";  
 // 2. Request Obejct인 StringRequest 생성  
  
 FirebaseStorage storage = FirebaseStorage.*getInstance*();  
 StorageReference storageReference = storage.getReference();  
 StorageReference pathReference = storageReference.child("flower\_image");  
  
 System.*out*.println(name);  
  
 String presult = name + ".jpg";  
  
  
  
 System.*out*.println(presult);  
  
 String image\_path = "flower\_image/"+presult ; //이미지 경로 지정  
  
 if (pathReference == null) {  
 Toast.*makeText*(PlantInfo2.this, "저장소에 사진이 없습니다." ,Toast.*LENGTH\_SHORT*).show();  
 } else {  
 StorageReference submitProfile = storageReference.child(image\_path); //여기에 삽입  
 submitProfile.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {  
 @Override  
 public void onSuccess(Uri uri) {  
 //System.out.println("msg2 pathReference : "+"flower\_image/"+presult);  
 Glide.*with*(PlantInfo2.this).load(uri).into(imageview);  
  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
  
 }  
 });  
 }  
  
  
 StringRequest request = new StringRequest(Request.Method.*POST*, url,  
 new Response.Listener<String>() {  
 @Override  
 public void onResponse(String response) {  
 System.*out*.println("222222");  
 Log.*d*("result", "[" + response + "]"); // 서버와의 통신 결과 확인 목적  
 showJSONList(response);  
 }  
 },  
 new Response.ErrorListener() {  
 @Override  
 public void onErrorResponse(VolleyError error) {  
 Log.*d*("error", "[" + error.getMessage() + "]");  
 }  
 }) {  
 @Override  
 protected Map<String, String> getParams() throws AuthFailureError {  
 Map<String, String> params = new HashMap<>();  
 params.put("name", name);  
 return params;  
 }  
 };  
  
 // 3) 생성한 StringRequest를 RequestQueue에 추가  
 requestQueue.add(request);  
 }  
  
 private void showJSONList(String response) {  
 final TextView tfeature = findViewById(R.id.*tv\_feature*);  
 final TextView tname = findViewById(R.id.*tv\_name*);  
 final TextView tmanagement = findViewById(R.id.*tv\_management*);  
 final TextView tnotice = findViewById(R.id.*tv\_notice*);  
 final ImageView timage = findViewById(R.id.*image*);  
 // 서버 정보를 파싱하기 위한 변수 선언  
 try {  
 JSONArray jsonArray = new JSONArray(response.toString());  
  
 for (int i = 0; i < jsonArray.length(); i++) {  
 JSONObject jsonObject = jsonArray.getJSONObject(i);  
  
 String name = jsonObject.getString("name");  
 tname.setText(name);  
 String feature = jsonObject.getString("feature");  
 tfeature.setText("특징 : "+"\n"+feature+"\n");  
 String management = jsonObject.getString("mangement");  
 tmanagement.setText("관리 : "+"\n"+management+"\n");  
 String notice = jsonObject.getString("notice");  
 tnotice.setText("병 : "+"\n"+notice+"\n");  
 String image = jsonObject.getString("image");  
 int pi = getResources().getIdentifier(image,"drawable",getPackageName());  
 timage.setImageResource(pi);  
 }  
 }catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
}

Camera\_activity.java

public class Camera\_activity extends AppCompatActivity {  
  
 private static final String *MODEL\_PATH* = "model.tflite";  
 private static final boolean *QUANT* = true;  
 private static final String *LABEL\_PATH* = "labels.txt";  
 private static final int *INPUT\_SIZE* = 224;  
  
 private Classifier classifier;  
  
 private Executor executor = Executors.*newSingleThreadExecutor*();  
  
 private Button btnDetectObject;  
 private ImageView imageViewResult;  
 private CameraView cameraView;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*camera\_activity*);  
 setTitle("menuCamera");  
  
  
 cameraView = findViewById(R.id.*cameraView*);  
  
  
 btnDetectObject = findViewById(R.id.*btnDetectObject*);  
  
 cameraView.addCameraKitListener(new CameraKitEventListener() {  
 @Override  
 public void onEvent(CameraKitEvent cameraKitEvent) {  
 }  
  
 @Override  
 public void onError(CameraKitError cameraKitError) {  
 }  
  
 @Override  
 public void onImage(CameraKitImage cameraKitImage) {  
  
 Bitmap bitmap = cameraKitImage.getBitmap();  
  
 bitmap = Bitmap.*createScaledBitmap*(bitmap, *INPUT\_SIZE*, *INPUT\_SIZE*, false);  
  
 final List<Classifier.Recognition> results= classifier.recognizeImage(bitmap);  
  
  
 //textViewResult.setText(results.toString());  
  
 String cname = results.toString();  
 String result = cname.substring(1, cname.length()-1);  
 System.*out*.println(result);  
 System.*out*.println(cname);  
 Intent intent = new Intent(Camera\_activity.this, PlantInfo2.class);  
 intent.putExtra("value2",result);  
 startActivity(intent);  
  
 }  
  
 @Override  
 public void onVideo(CameraKitVideo cameraKitVideo) {  
  
 }  
 });  
  
  
 btnDetectObject.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 cameraView.captureImage();  
  
 }  
 });  
  
 initTensorFlowAndLoadModel();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 cameraView.start();  
 }  
  
 @Override  
 protected void onPause() {  
 cameraView.stop();  
 super.onPause();  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 executor.execute(new Runnable() {  
 @Override  
 public void run() {  
 classifier.close();  
 }  
 });  
 }  
  
 private void initTensorFlowAndLoadModel() {  
 executor.execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 classifier = TensorFlowImageClassifier.*create*(  
 getAssets(),  
 *MODEL\_PATH*,  
 *LABEL\_PATH*,  
 *INPUT\_SIZE*,  
 *QUANT*);  
 makeButtonVisible();  
 } catch (final Exception e) {  
 throw new RuntimeException("Error initializing TensorFlow!", e);  
 }  
 }  
 });  
 }  
  
 private void makeButtonVisible() {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 btnDetectObject.setVisibility(View.*VISIBLE*);  
 }  
 });  
 }  
}

Classifier.java

public interface Classifier{  
  
 class Recognition {  
 */\*\*  
 \* A unique identifier for what has been recognized. Specific to the class, not the instance of  
 \* the object.  
 \*/* private final String id;  
  
 */\*\*  
 \* Display name for the recognition.  
 \*/* private final String title;  
  
 */\*\*  
 \* Whether or not the model features quantized or float weights.  
 \*/* private final boolean quant;  
  
 */\*\*  
 \* A sortable score for how good the recognition is relative to others. Higher should be better.  
 \*/* private final Float confidence;  
  
 public Recognition(  
 final String id, final String title, final Float confidence, final boolean quant) {  
 this.id = id;  
 this.title = title;  
 this.confidence = confidence;  
 this.quant = quant;  
  
 }  
  
  
 public String getId() {  
 return id;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public Float getConfidence() {  
 return confidence;  
 }  
  
  
 @Override  
 public String toString() {  
 String resultString = "";  
 /\*  
  
 if (title != null) {  
  
 resultString += title + " ";  
 }  
  
 return resultString.trim();  
 }  
 }  
  
  
 List<Recognition> recognizeImage(Bitmap bitmap);  
  
 void close();  
}

Model.py

<라이브러리>

텍스트이(가) 표시된 사진

자동 생성된 설명

<데이터 셋 경로지정>



<Train , Validation Set >

텍스트이(가) 표시된 사진

자동 생성된 설명

<모델 성능 향상을 위한 이미지 전처리>

텍스트이(가) 표시된 사진

자동 생성된 설명

<모델 생성 및 훈련>

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

<.h5 -> .tflie 확장자로 변환>

텍스트이(가) 표시된 사진

자동 생성된 설명

# Ⅲ. 결 론

**1. 연구결과** 본 연구는 등산 시, 혹은 선물 받은 식물들을 쉽게 인식하고 정보를 찾는데 도움을 주고자 하였다. CNN모델에서의 식물의 인식률을 높이기 위해 Tensorflow에서 제공하는 AUTO TUNE을 사용함으로써 기존 모델 대비 정확도를 향상 시켰다. 또한 검색기능을 통합하여 다른 어플리케이션(인터넷 등)의 사용 없이도 볼 수 있으며 기존에 검색했던 식물들을 데이터 베이스에 저장함으로써 사용자가 다시 조회할 수 있는 편의성을 제공한다.

현재 구현한 것에 더 나아가 지금보다 더 좋은 성능을 내기 위해 이미지 인식 정확도가 높은 Vit-G/14 또는 실시간으로 Object Detection이 가능한 YOLO 모델을 사용하여 더 많은 종에 대한 인식도와 속도를 개선할 수 있다. 추가적으로 소비자가 많이 조회하며 관심있어 하는 식물을 분석하여 해당 식물에 필요한 악세사리 - 화분, 영양제, 흙 등을 추천함으로써 전문가의 도움 없이도 손쉽게 구매하고 케어 할 수 있는 환경을 기대할 수 있다.

**2. 제품 제작 소요 제품 목록**

|  |  |  |
| --- | --- | --- |
| **Software Development Enviroment** | | |
| Server | AWS(Apache Tomcat) | Tomcat ver 8.5.2 |
| DB | MySQL, Firebase | ver 8.0.1, ver 21.0.1 |
| Android IDE | Android Studio | ver 4.1.3 |
| Deep Learning | Tensorflow | ver 2.0 |

* **참고문헌**

**[1] 책,**

**혼자 공부하는 머신러닝+딥러닝**

**밑바닥부터 시작하는 딥러닝**

**[2] Tensorflow,** [**https://www.tensorflow.org/api\_docs/python/tf/data/Dataset?hl=ko#cache**](https://www.tensorflow.org/api_docs/python/tf/data/Dataset?hl=ko#cache)

[**https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/api\_docs/python/**](https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/api_docs/python/)

[**https://www.tensorflow.org/lite/convert?hl=ko**](https://www.tensorflow.org/lite/convert?hl=ko)

[**https://www.tensorflow.org/lite/examples/image\_classification/overview?hl=ko**](https://www.tensorflow.org/lite/examples/image_classification/overview?hl=ko)

**[3] 블로그,**

[**https://tykimos.github.io/2017/03/25/Dataset\_and\_Fit\_Talk/**](https://tykimos.github.io/2017/03/25/Dataset_and_Fit_Talk/)

[**https://eveningdev.tistory.com/25**](https://eveningdev.tistory.com/25)

[**https://call203.tistory.com/28**](https://call203.tistory.com/28)

**[4] 유튜브,**

[**https://www.youtube.com/channel/UCOAyyrvi7tnCAz7RhH98QCQ**](https://www.youtube.com/channel/UCOAyyrvi7tnCAz7RhH98QCQ)

**[5] 논문**

[**https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf**](https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)

[**https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A**](https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A)

[**https://arxiv.org/pdf/1803.08375.pdf**](https://arxiv.org/pdf/1803.08375.pdf)