2023-1 AI 융합학부 캡스톤 디자인

건강을 위한 첫 걸음

당일일지 혈당관리앱

2023학년도 1학기 캡스톤 디자인 프로젝트 최종 보고서

AI 융합학부 당동당

김예린 김혜빈 우미경 이하늘

AI 융합학부 20192897

김혜빈

Contents

로그인 및 메인 화면


:D
아이디를 입력하세요
비밀번호를 입력하세요
로그인
회원가입


회원가입
아이디: 아이디 입력
패스워드: 패스워드 입력
중복 체크: 패스워드 재입력
닉네임: 설정할 닉네임 입력
나이: 나이 입력
성별: 여자 남자
키: 키 입력
몸무게: 몸무게 입력


전날
05월 28일
다음날
점심 (11:10)
식전 혈당
80
점심 (12:30)
식후 혈당
120
점심
불고기
당류: 7.8g
탄수화물: 7.9g
저녁
황태구이
당류: 14.2g
탄수화물: 26.9g
어제의 건강 현황
섭취한 총 당량: 35 mg
혈당 최고치: 170 mg/dl


마이페이지
딩동당님 반갑습니다!

혈당 등록 및 조회


11:07
전날
05월 28일
다음날
점심 (11:10)
식전 혈당
80
점심 (12:30)
식후 혈당
120
점심
불고기
당류: 7.8g
탄수화물: 7.9g
저녁
황태구이
당류: 14.2g
탄수화물: 26.9g
어제의 건강 현황
섭취한 총 당량: 35 mg
혈당 최고치: 170 mg/dl


11:07
혈당 등록
식단 등록


11:07
혈당 등록
입력하는 날짜/시간
날짜: 2023/05/28
수정
시간: 23:07
수정
아침 점심 저녁
식전/식후
식전 식후
혈당값
혈당 값을 입력해주세요.
등록


11:08
전날
05월 28일
다음날
오늘의 혈당량 변화
점심 (12:30)
식후 혈당
120
점심 (12:30)
식후 혈당
120
저녁 (17:20)
식전 혈당
90
저녁 (18:30)
식후 혈당
110

식단 등록 및 조회

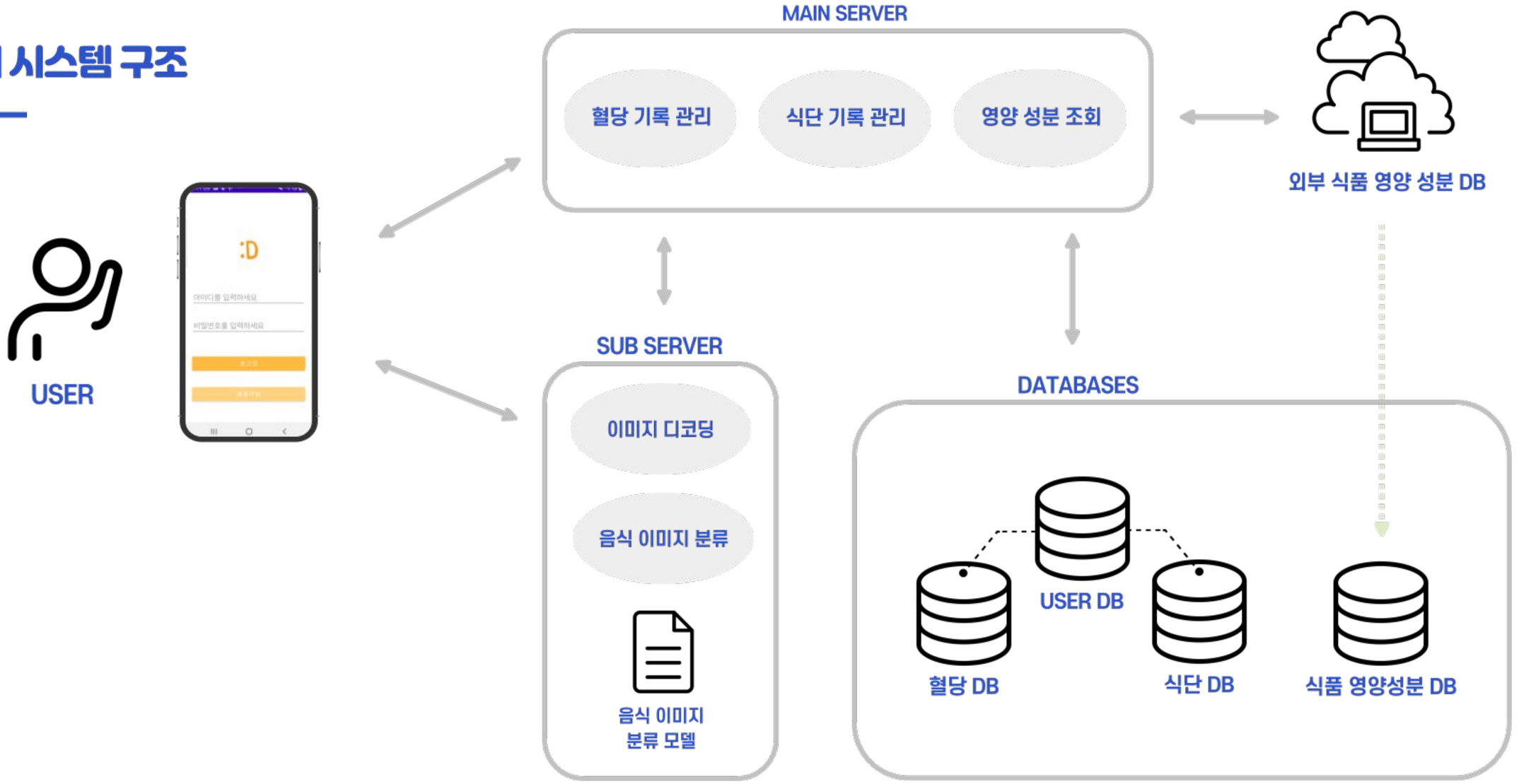

11:07
전날
05월 28일
다음날
점심 (11:10)
식전 혈당
80
점심 (12:30)
식후 혈당
120
점심
불고기
당류: 7.8g
탄수화물: 7.9g
저녁
황태구이
당류: 14.2g
탄수화물: 26.9g
어제의 건강 현황
섭취한 총 당량: 35 mg
혈당 최고치: 170 mg/dl


11:07
식단 등록
직접입력
사진찍기
갤러리
이미지전송

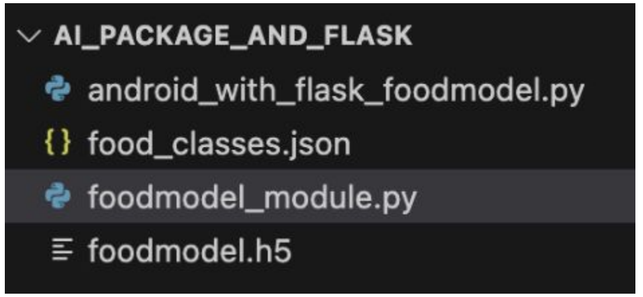

11:08
불고기
영양성분 조회
이름: 불고기
칼로리: 395.29
탄수화물: 7.8
단백질: 34.4
지방: 25.2
당류: 7.9
나트륨: 560.71
입력하는 날짜/시간
날짜: 2023/05/28
수정
시간: 23:07
수정
아침 점심 저녁
등록

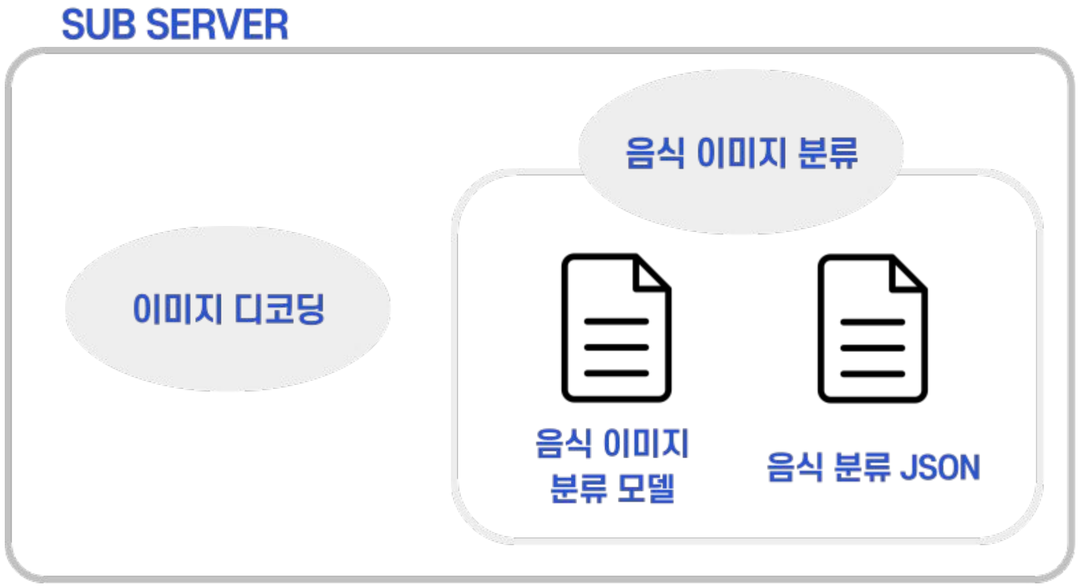

11:08
전날
05월 28일
다음날
점심 (12:30)
불고기
당류: 7.8g
탄수화물: 7.9g
저녁 (18:10)
황태구이
당류: 14.2g
탄수화물: 26.9g
오늘의 적정 당 섭취량 진행 현황

전체 시스템 구조


MAIN SERVER
혈당 기록 관리
식단 기록 관리
영양 성분 조회
외부 식품 영양 성분 DB
USER
:D
SUB SERVER
이미지 디코딩
음식 이미지 분류
음식 이미지
분류 모델
DATABASES
USER DB
혈당 DB
식단 DB
식품 영양성분 DB

플라스크 서버 - Sub server





이미지 디코딩 모듈

- 플라스크 서버 내 안드로이드 앱과의 이미지 송수신을 위한 모듈 내장
- Base64로 인코딩된 이미지를 받아 디코딩하여 PIL Image 객체로 메모리에 저장
- 모델에 활용하기 위해 numpy 배열로 변환

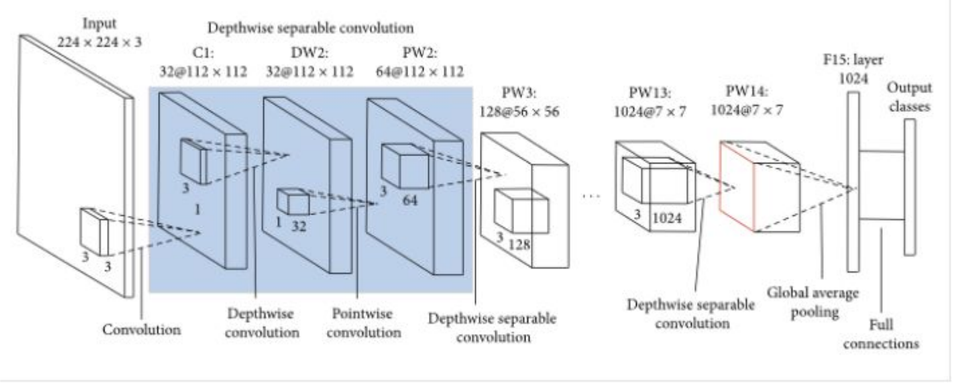
이미지 분류기 활용 모듈

- 플라스크 서버 내에서 학습된 음식 분류 모델을 활용할 수 있도록 하기 위함
- 모델 및 음식 분류 딕셔너리 생성 및 로드
- 이미지 전처리와 예측 결과 반환

학습된 모델 파일 / 음식 분류 JSON

- 플라스크 서버가 실행될 때 서버 메모리에 로드

한식 이미지 분류 모델 구조



기존 모델의 특성 추출 활용 + 새로운 완전 연결 층

- MobileNet 구조 중에서 특성 추출까지만 활용
- 한식 이미지 50종 분류에 맞춰 새로운 Dense 층 추가 구성

사용 가능한 모델 종류 : https://keras.io/api/applications/

```
base_model = MobileNet (
                weights='imagenet',
                include_top=False,
                input_shape=(224, 224, 3),
                pooling='avg',
            )
```

MobileNet

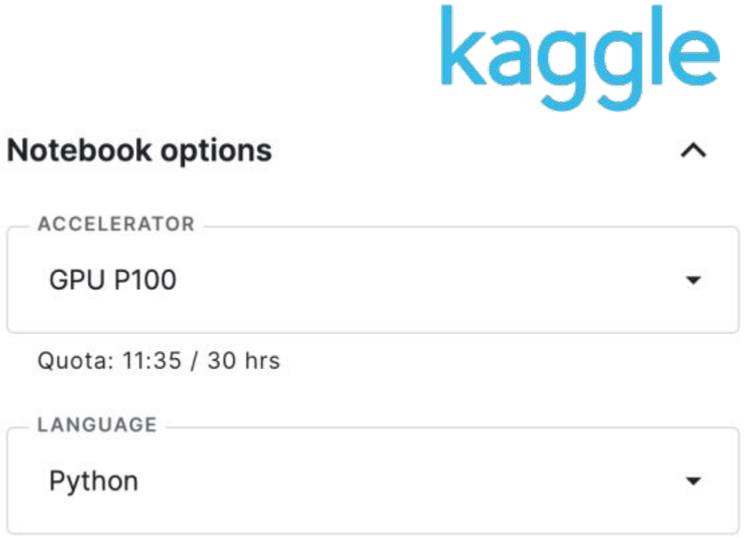
- imagenet 가중치로 초기 가중치 셋팅
- GlobalAveragePooling2D 층을 output으로 설정하여 특징 추출까지만 활용

```
x = base_model.output
x = Dense(1024, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.2)(x)
output = Dense(food_classes, activation='softmax')(x)
```

완전 연결 층

- 다중 클래스 분류를 위해 softmax 활용
- 일반화 성능 향상을 위해 BatchNormalization 층과 Dropout 층 추가 구성

한식 이미지 분류 모델 학습



Kaggle kernel을 활용한 학습

- Google Colab과 Kaggle kernel 모두 사용
- GPU P100을 사용하여 학습 진행
- gdown 라이브러리를 통해 이미지셋 다운로드

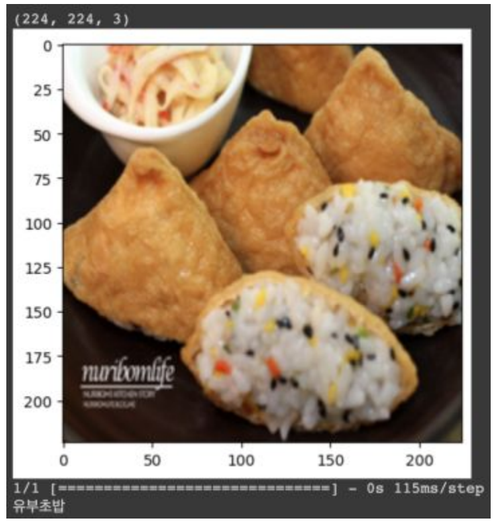
```
# 모델 컴파일
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Epoch 16/20

1172/1172 [==============================] - 141s 120ms/step - loss: 0.1459 - accuracy: 0.9533 - val_loss:
1.2533 - val_accuracy: 0.7525

Epoch 17/20

1172/1172 [==============================] - 142s 121ms/step - loss: 0.1369 - accuracy: 0.9559 - val_loss:
1.2370 - val_accuracy: 0.7359

Epoch 18/20

1172/1172 [==============================] - 144s 123ms/step - loss: 0.1318 - accuracy: 0.9574 - val_loss:
1.2986 - val_accuracy: 0.7408

Epoch 19/20

1172/1172 [==============================] - 139s 118ms/step - loss: 0.1142 - accuracy: 0.9625 - val_loss:
1.4751 - val_accuracy: 0.7241
```

모델 컴파일 및 학습

- 최적화 알고리즘 : Adam
- 손실 함수 : categorical_crossentropy
- Batch size : 32
- Epoch : 20

한식 이미지 분류 모델 테스트 및 성능 확인

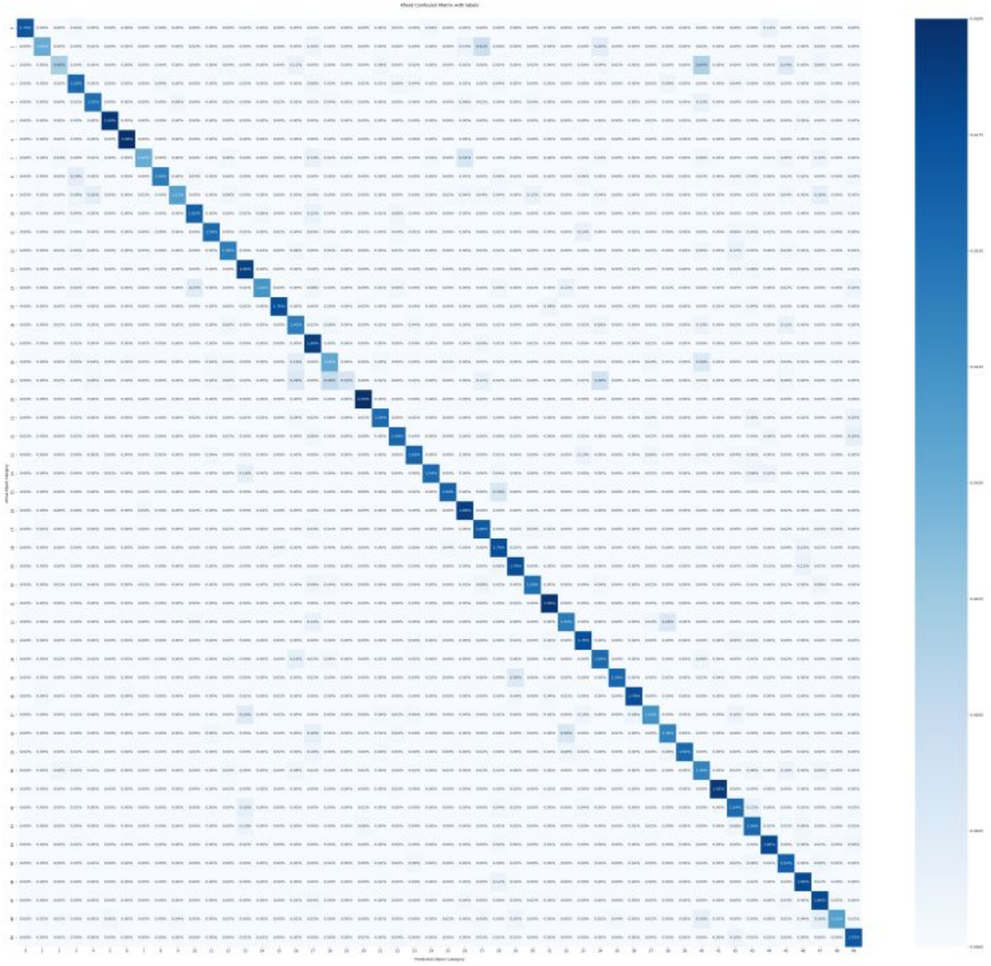


Classification Report:

	precision	recall	f1-score	support
고사리나물무침	0.98	0.85	0.91	100
달걀국	0.91	0.51	0.65	100
닭개장	0.74	0.34	0.47	100
삼겹살구이	0.70	0.78	0.74	100
소고기무국	0.81	0.76	0.78	100
숙주나물무침	0.96	0.97	0.97	100
시금치나물무침	0.95	0.99	0.97	100
시래기된장국	0.91	0.51	0.65	100
훈제오리	0.95	0.71	0.81	100
갈비탕	0.92	0.56	0.70	100
갈치구이	0.81	0.81	0.81	100
감자조림	0.87	0.77	0.81	100
감자탕	0.73	0.70	0.71	100
건새우볶음	0.68	0.94	0.79	100
고등어구이	0.90	0.61	0.73	100
김치볶음밥	0.90	0.88	0.89	100
김치찌개	0.59	0.70	0.64	100
누룽지	0.58	0.94	0.72	100
동태찌개	0.57	0.53	0.55	100
된장찌개	0.94	0.16	0.27	100
두부김치	0.90	1.00	0.95	100
두부조림	0.80	0.77	0.79	100
떡갈비	0.80	0.79	0.79	100

```
[ ]  # 테스트 이미지셋을 통한 손실값, 예측값 확인
     loss, acc = model.evaluate(test_generator, verbose=1)
     print('모델 loss: {:5f}'.format(loss))
     print('모델 accuracy: {:5.2f}%'.format(100*acc))

     157/157 [==============================] - 176s 1s/step - loss: 1.1865 - accuracy: 0.76
     모델 loss: 1.186464
     모델 accuracy: 76.92%
```

Confusion Matrix

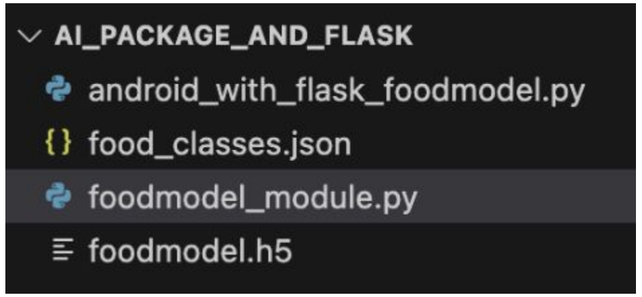


모델 및 분류 정보 저장

```
# 학습된 모델 h5 파일로 저장
modelFile = f'MobileNet-32-20-{acc}.h5' # modelName-batch-epochs-accuracy
model.save(models_path+modelFile)
```

```
[ ]  ##### 음식 레이블 json 형식으로 저장
     import json

     # 음식 레이블 저장
     class_dict = test_generator.class_indices # dict 형태 : {음식 종류 : 종류 index}
     print('음식 종류 :' ,list(class_dict.keys())) # 음식 이름 확인
     print('클래스 개수 :', len(class_dict.keys()))

     # JSON 형식으로 저장 - {index : 음식종류} 형태
     jsonfile_name = "food_classes.json"
     rev_class_dict = {v : k for k, v in class_dict.items()}
     with open(jsonfile_name, "w", encoding="utf-8") as json_file:
         json.dump(rev_class_dict, json_file, ensure_ascii=False)
```

```
<class 'dict'>
dict_keys(['고사리나물무침', '달걀국', '닭개장', '삼겹살구이', '소고기무국', '숙주나물무침', '시금치나물무침',
클래스 개수 :  50
```



학습된 모델 파일 : foodmodel.h5
음식 분류 json 파일 : food_classes.json

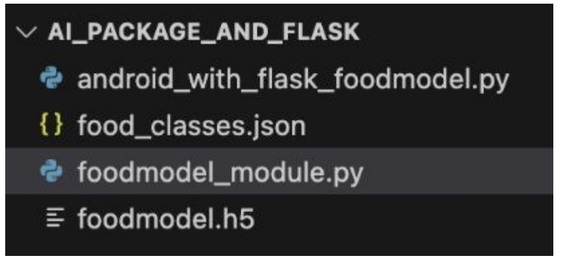
이미지 분류기 활용 모듈

```python
######### 모델 예측 관련 함수 ###########
# Pre : 모델 및 음식 분류 json 파일 load 필요

### 모델 예측 결과 ###
# Input : 클래스 분류 json 파일, 저장된 모델, 음식 이미지 배열
# Output : 음식 종류 이름
def predict_one_food(foodDict, model, foodImage) :
    foodImage = np.expand_dims(foodImage, axis=0) # 입력 레이어 형식 맞춤
    foodImage = foodImage / 255. # 이미지 데이터 정규화
    foodIdx = np.argmax(model.predict(foodImage), axis=1) # 예측 -> index
    foodName = foodDict[str(foodIdx[0])] # index -> 음식 이름
    return foodName
```

<이미지 분류기 활용 모듈> foodmodel_module.py 중 일부


```
{"0": "고사리나물무침", "1": "달걀국", "2": "닭개장", "3": "삼겹살구이", "4": "소고기무국",
"5": "숙주나물무침", "6": "시금치나물무침", "7": "시래기된장국", "8": "훈제오리", "9": "갈비
탕", "10": "갈치구이", "11": "감자조림", "12": "감자탕", "13": "건새우볶음", "14": "고등어구
이", "15": "김치볶음밥", "16": "김치찌개", "17": "누룽지", "18": "동태찌개", "19": "된장찌
개", "20": "두부김치", "21": "두부조림", "22": "떡갈비", "23": "떡볶이", "24": "멸치볶음",
"25": "물냉면", "26": "미역국", "27": "북엇국", "28": "비빔냉면", "29": "비빔밥", "30":
"삼계탕", "31": "새우볶음밥", "32": "새우튀김", "33": "소세지볶음", "34": "순두부찌개",
"35": "알밥", "36": "애호박볶음", "37": "어묵볶음", "38": "오징어튀김", "39": "유부초밥",
"40": "육개장", "41": "잡곡밥", "42": "제육볶음", "43": "주꾸미볶음", "44": "짜장면",
"45": "짬뽕", "46": "쫄면", "47": "칼국수", "48": "콩나물국", "49": "황태구이"}
```


<음식 분류 JSON> food_classes.json



전역 변수 설정

- 모델 파일과 음식 분류 JSON 파일 경로를 설정하는 함수
- 모델 및 음식 분류를 메모리에 로드하기 전 미리 설정해야 함

모델 및 음식 분류 로드

- 플라스크 서버 내에서 학습된 음식 분류 모델을 활용할 수 있도록 하기 위함
- 모델 및 음식 분류 딕셔너리 load
- 이미지 전처리와 예측 결과 반환

예측 결과 반환

- 플라스크 서버가 실행될 때 서버 메모리에 load

최종 테스트 진행 환경

REST api

내부 네트워크 IP

Amazon EC2

Local PC

MAIN SERVER

spring

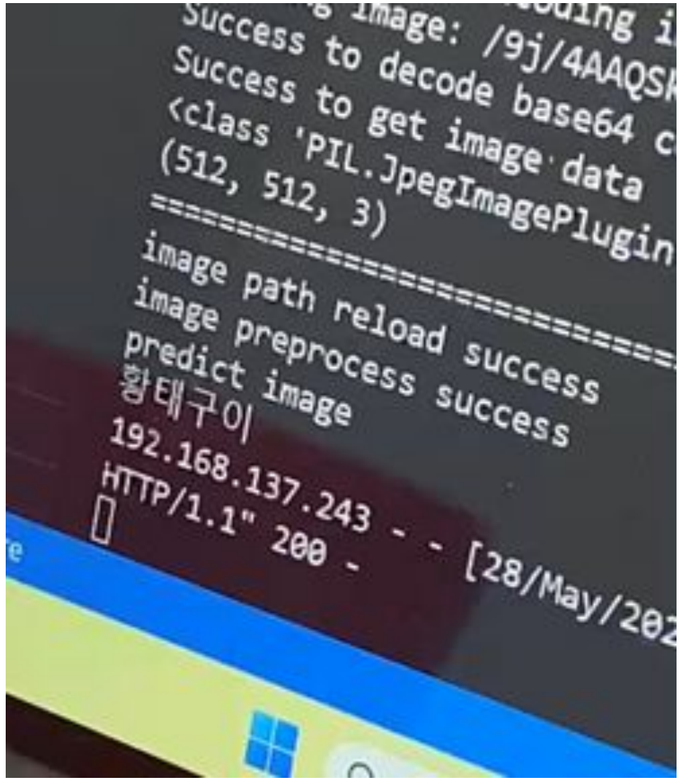
Spring Boot

SUB SERVER

Flask
web development,
one drop at a time

AI

이미지 등록 및 예측 시연 영상



출처 자료

아이콘 활용 : https://www.iconfinder.com/

음식 이미지 : <ai hub 한식 이미지 데이터> https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=79

spring framework icon : https://ko.m.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:Spring_Framework_Logo_2018.svg

Flask framework icon :
https://ko.m.wikipedia.org/wiki/%ED%94%8C%EB%9D%BC%EC%8A%A4%ED%81%AC_(%EC%9B%B9_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC)

감사합니다