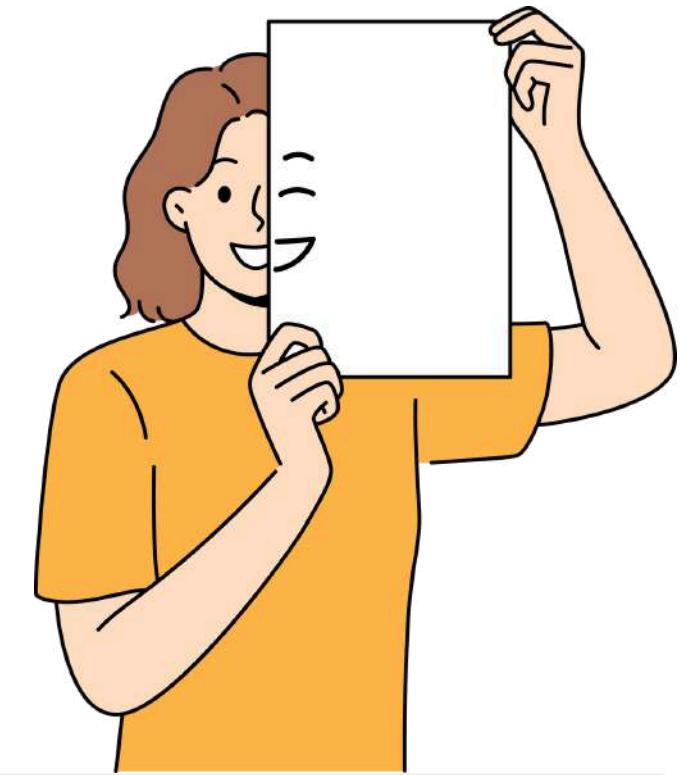


# AI 기반 이미지 처리 머신러닝&딥러닝 프로젝트

마스크 착용 안면 인식 판별



최준희, 방현, 배선화, 복권일,  
서명숙, 추성호, 안은영



# AI 기반 퍼스널컬러 진단 앱

1 About Project

2 Mask Detection Model

3 AI Personal Color Predict

4 BeautyGAN Model

5 Email Certification

6 JPA & Mybatis

7 Conclusion

트렌드모니터의 매거진

## ‘나’ 자신이 가장 중요한 시대, 스스로에게 관심 가지면서 개인의 ‘정체성’을 찾는 사람들



트렌드모니터



2020.11.19 ~ 08:08

2491

1

1

0

- 무엇보다 내가 중요한 시대, 전체 76.5% “평소 나에게 관심 기울여”
- 전체 66.4% “나 자신을 위해 사는 것이 인생에서 가장 중요”
- 반면 타인에 대한 관심은 점점 감소(13년 65.1%→20년 55.7%)
- 다만 타인의 시선을 의식하는 태도(17년 52%→20년 54.1%)는 여전
- 10명 중 8명 “남들에게 보이기 위한 행동을 해야만 하는 경우가 많다”
- 전체 63.5% “평소 나의 평판을 잘 관리하는 것은 나에게 중요해”

# What is Personal Color?

- ◎ 퍼스널컬러는 개인의 피부 톤, 머리 색, 눈 색 등을 고려 가장 잘 어울리는 색상을 찾아주는 개념.
- ◎ 보편적으로 크게 Warm과 Cool 그리고 봄, 여름, 가을, 겨울 사계절톤으로 구분.
- ◎ 자신을 더욱 돋보이게 하고 자신감을 향상.



# 주제 선정 이유

- 1 자기 자신에 대한 관심 증가
- 2 메이크업 트렌드와 개인 맞춤형 뷰티 솔루션에 대한 니즈 증가
- 3 퍼스널컬러 진단 비용의 부담 감소 및 편리하고 접근 가능한 서비스 제공의 필요성 증가
- 4 뷰티 솔루션과 AI 기술의 융합을 통한 확장성에 주목

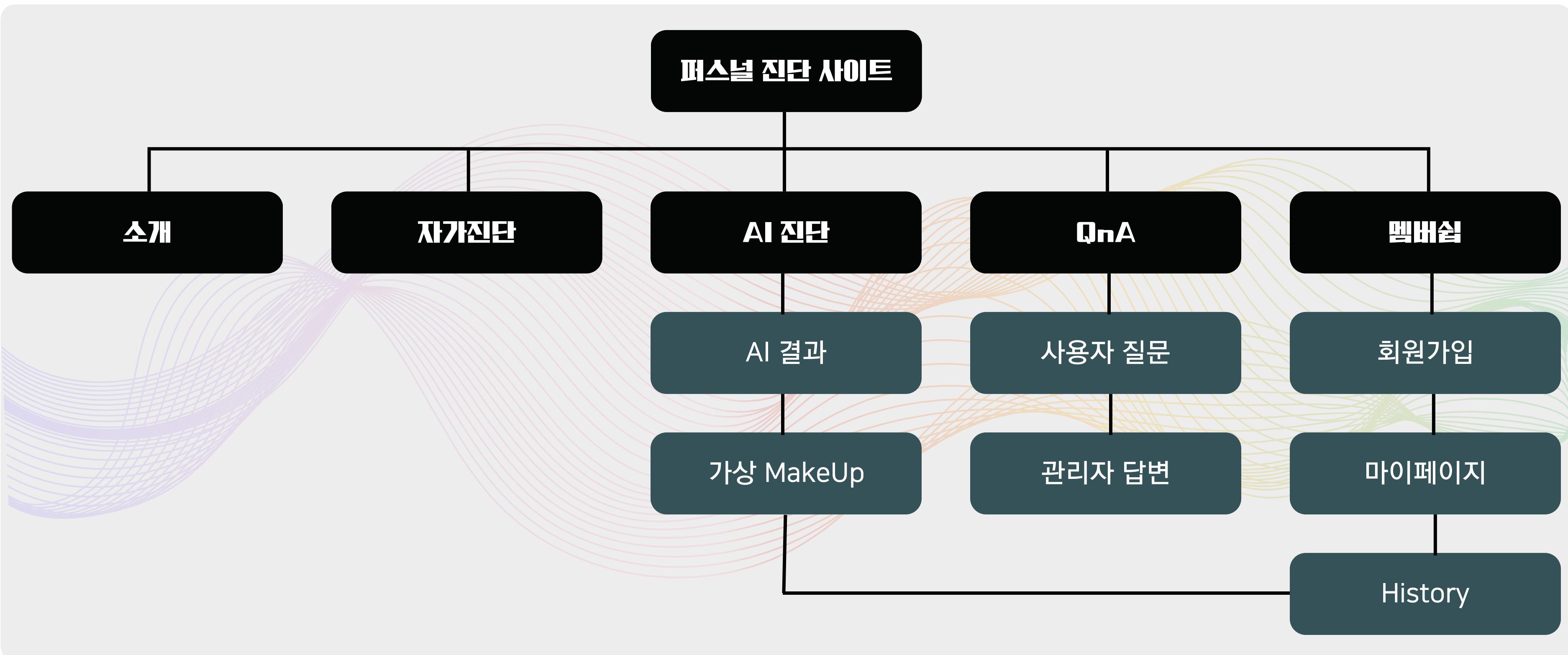


현대인들의 뷰티 니즈를 충족시키고, 편리하고 혁신적인 뷰티 솔루션을 제공

# 프로젝트 일정

sun	mon	tue	wen	thu	fri	sat
					06/28	
					프로젝트 시작	주제구상
			DB설계 및 서버구축			
주제구상 / 아이디어회의			UI 설계 및 제작			데이터 수집
			Spring-django-vue CORS통신 / back단 구축			
			데이터 수집 및 모델생성			
CORS통신 / back단 구축				07/18		
취합		PPT 제작 및 최종검수		마감		

# 홈페이지 도식화



## 팀원 소개



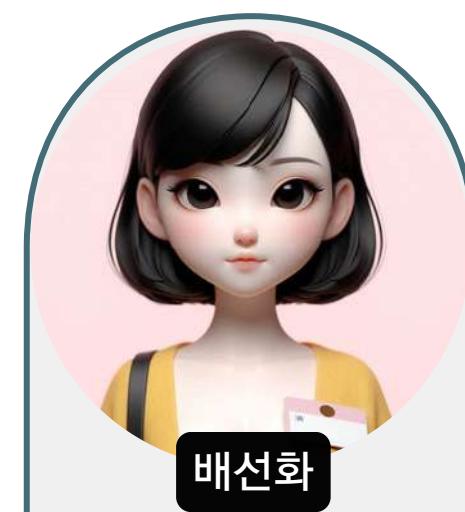
방 현

admin UI 제작  
DB 프로시저 정의 및 생성  
ERD 설계  
서버구조 구축  
admin 회원관리 구조 구축

JPA - Vue Axios CORS통신

테스트 더미 데이터 생성 로직 작성

ERD PPT제작



배선화

자가진단 및 Q&A UI 제작  
홈페이지 전체 디자인 검수  
데이터 크롤링 및 데이터 증강  
admin 회원관리 구조 구축

JPA - Vue Axios CORS통신

마스크인식 모델

마스크인식 PPT제작



복권일

메인 UI 제작  
Q&A 게시판 및 마이페이지 구조 구축  
전체 code 취합 및 코드검수  
홈페이지 대표 디자인 설계 및 구축

Mybatis - Vue CORS통신

가상메이크업 Beauty GAN모델

Beauty GAN PPT제작



최준희

프로젝트 총괄  
마이페이지 및 히스토리 UI 제작  
마이페이지 구조 구축

Mybatis - Vue CORS통신

가상메이크업 Beauty GAN모델

마이페이지, 히스토리 PPT제작



추성호

가상메이크업 UI 제작  
가상메이크업 구조 구축  
AI진단 결과 구조 구축  
django CORS통신

Mybatis - Vue CORS통신

가상메이크업 Beauty GAN모델

django PPT제작



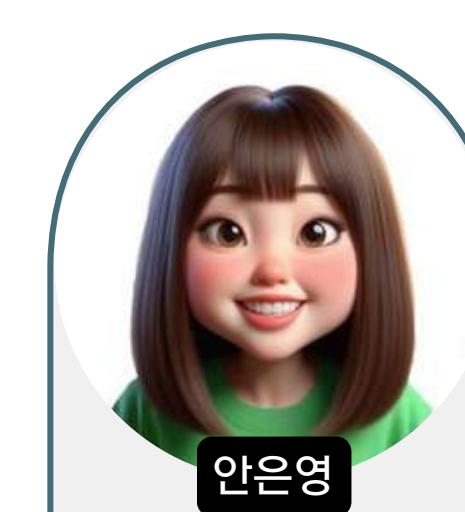
서명숙

AI진단 UI 제작  
결과확인 UI 제작  
Q&A UI 제작  
Q&A 게시판 구조 구축  
더미 데이터 제작

Mybatis - Vue CORS통신

게시판 CRUD PPT 제작

시연영상 촬영



안은영

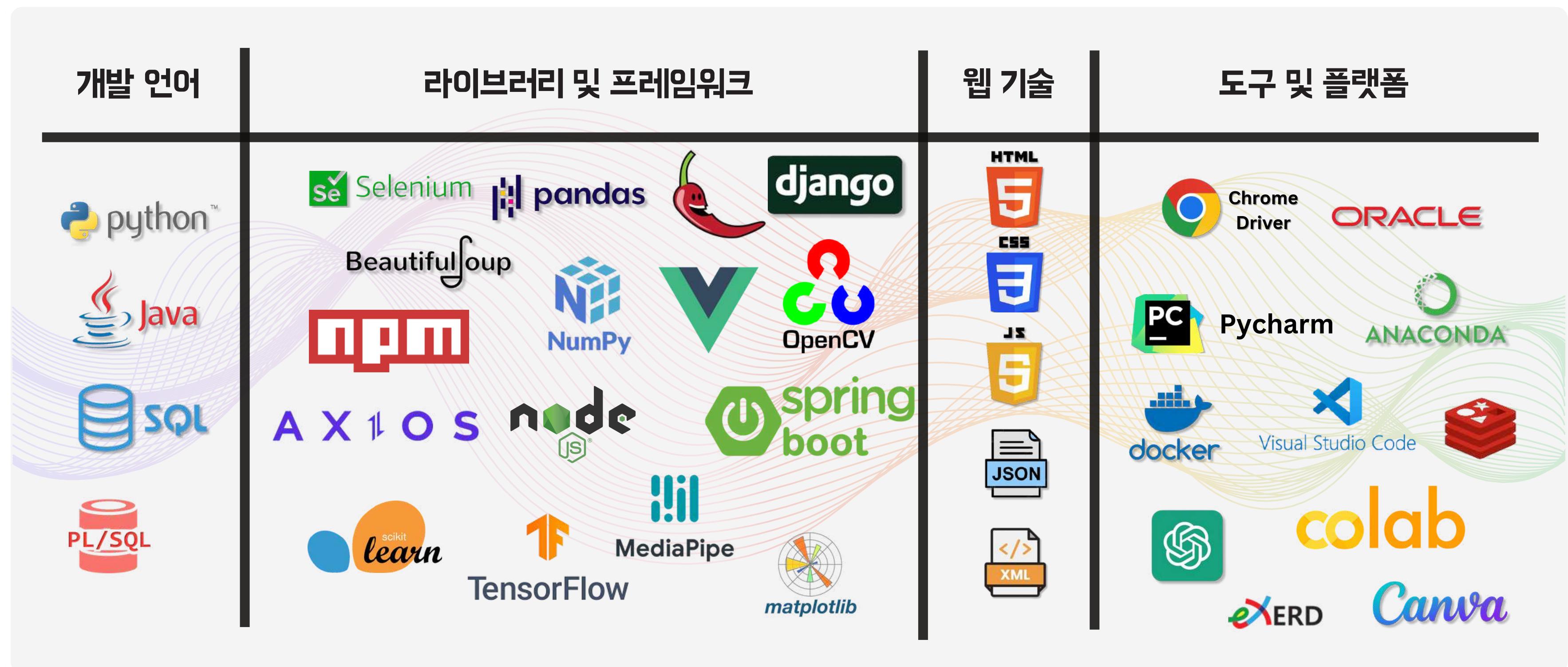
회원가입 UI 제작  
회원가입 및 히스토리 구조 구축  
Docker Redis 이메일인증 구현  
홈페이지 대표 디자인 설계 및 검수

Mybatis - Vue CORS통신

Redis 이메일인증 PPT제작

프로젝트 발표

# 개발환경



# 이미지 처리 기술 요소

## OpenCV

OpenCV는 컴퓨터의 눈이라 불리는 오픈소스 컴퓨터 비전을 목적으로 실시간 이미지 프로세싱에 중점을 둔 라이브러리

## Media-Pipe

구글에서 개발한 크로스 플랫폼 프레임워크 실시간으로 다양한 컴퓨터 비전 작업을 수행할 수 있도록 돋는 도구

## TensorFlow – Keras

딥러닝 프레임워크로, 이미지 분류, GAN, 객체 검출, 세그먼테이션을 통한 이미지 생성 및 변환 등에 사용.

## BeautyGAN

얼굴 이미지의 미용관련 스타일 전이를 위한 딥러닝 모델. 생성적 적대 신경망을 활용하여 사실적이고 자연스러운 효과생성

# Benchmarking

<b>UI</b>	<a href="https://matinkim.com/">https://matinkim.com/</a> ► 랜딩페이지 디자인 컨셉
<b>퍼스널컬러 진단서비스</b>	<a href="https://github.com/anveloper/YourSeasons">https://github.com/anveloper/YourSeasons</a> ► 서비스 프로세스
<b>BeautyGAN</b>	<a href="https://jonhyuk0922.tistory.com/103">https://jonhyuk0922.tistory.com/103</a> ► 서비스 구현 모델
<b>BeautyGan 논문</b>	<a href="https://medium.com/@sriskandaryan/facial-makeup-transfer-beautygan-d99389e1aae4">https://medium.com/@sriskandaryan/facial-makeup-transfer-beautygan-d99389e1aae4</a> ► 서비스 모델 구현
<b>마스크인식</b>	<a href="https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset">https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset</a> ► 서비스 모델 구현
<b>VGG16 논문</b>	<a href="https://neurohive.io/en/popular-networks/vgg16/">https://neurohive.io/en/popular-networks/vgg16/</a> ► 서비스 모델 구현

# Database ERD



# DB 설계

ERD

MEMBER_LOC					
	물리 이름*	도메인	데이터 타입	논리 이름*	널 허용
▶ LOCTNUM	일련번호	NUMBER(10)	광역시도번호	N·N	
● LOCNAME	지역명	VARCHAR2(20)	시도명	N·N	
● LATITUDE	도(°)	FLOAT(10)	위도	N·N	
● LONGITUDE	도(°)	FLOAT(10)	경도	N·N	

QnA					
	물리 이름*	도메인	데이터 타입	논리 이름*	널 허용
▶ QNUM	일련번호	NUMBER(10)	게시글번호	N·N	
▶ MNUM	일련번호	NUMBER(10)	회원번호	N·N	
● MNICK	닉네임	VARCHAR2(50)	닉네임	N·N	
● title	제목	VARCHAR2(40)	제목	N·N	
● content	내용	CLOB	내용	N·N	
● CDATE	날짜	DATE	작성날짜	N·N	
● HIT	조회수	NUMBER(10)	조회수	N·N	
● ANS	답변	CLOB	답변	NULL	
● ADATE	일시	DATE	답변일자	NULL	
● OPENYN	공개여부	NUMBER(10)	공개여부	N·N	

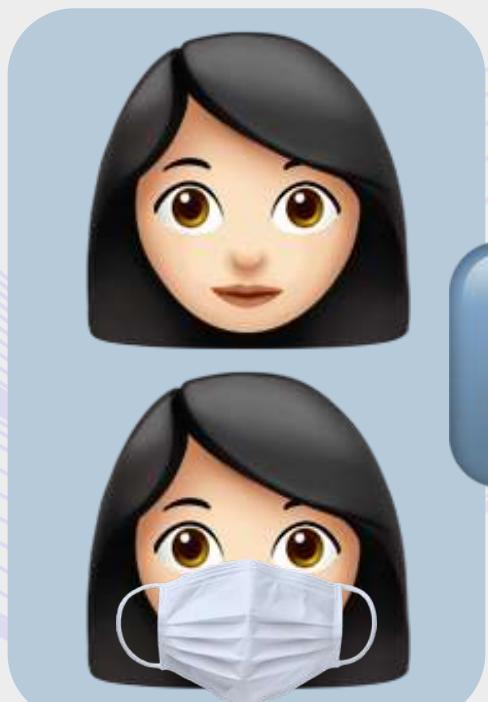
MEMBER					
	물리 이름*	도메인	데이터 타입	논리 이름*	널 허용
▶ MNUM	일련번호	NUMBER(10)	회원번호	N·N	
● MEMAIL	이메일	VARCHAR2(50)	이메일	N·N	
● MPWD	암호	VARCHAR2(50)	암호	N·N	
● MNICK	닉네임	VARCHAR2(50)	닉네임	N·N	
● MNAME	이름	VARCHAR2(50)	이름	N·N	
● MGENDER	성별	NUMBER(10)	성별	N·N	
● MBIRTH	생년월일	VARCHAR2(30)	생년월일	N·N	
● MDATE	일시	DATE	가입일	N·N	
● MIMGN	사진이름	VARCHAR2(50)	프로필사진이름	NULL	
▶ LOCNUM	일련번호	NUMBER(10)	광역시도번호	N·N	
● STATUS	상태	VARCHAR2(50)	상태	NULL	
● SEASON	계절ton	VARCHAR2(20)	계절ton	NULL	

AI_Res					
	물리 이름*	도메인	데이터 타입	논리 이름*	널 허용
▶ RNUM	일련번호	NUMBER(10)	A이진단번호	N·N	
▶ MNUM	일련번호	NUMBER(10)	회원번호	N·N	
● BEFIMGN	사진이름	VARCHAR2(50)	전이미지	N·N	
● AFIMGN	사진이름	VARCHAR2(50)	후이미지	NULL	
● RDATE	날짜	DATE	일시	N·N	
● SEASON	계절ton	VARCHAR2(20)	계절ton	N·N	

# Mask Detection model



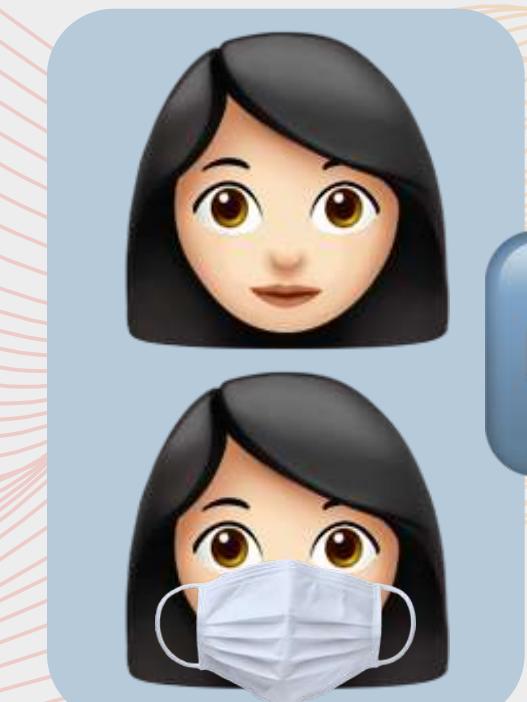
# 식별



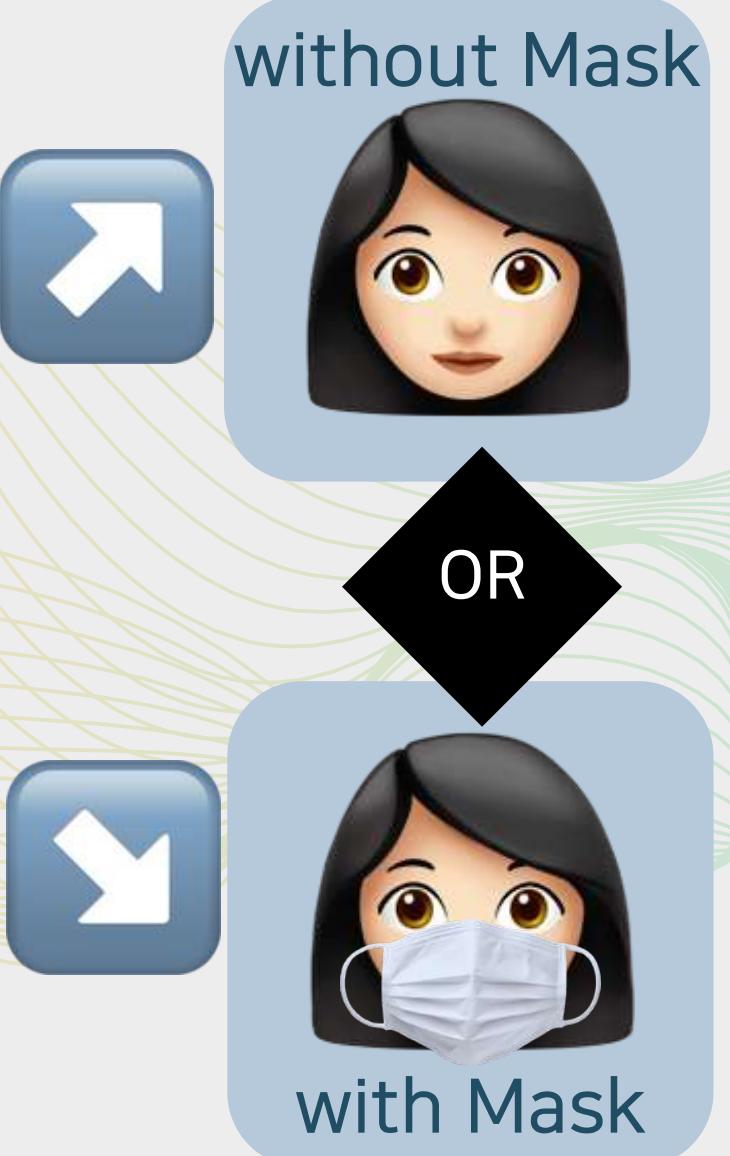
얼굴  
인식  
모델



# 검증



얼굴  
인식  
모델

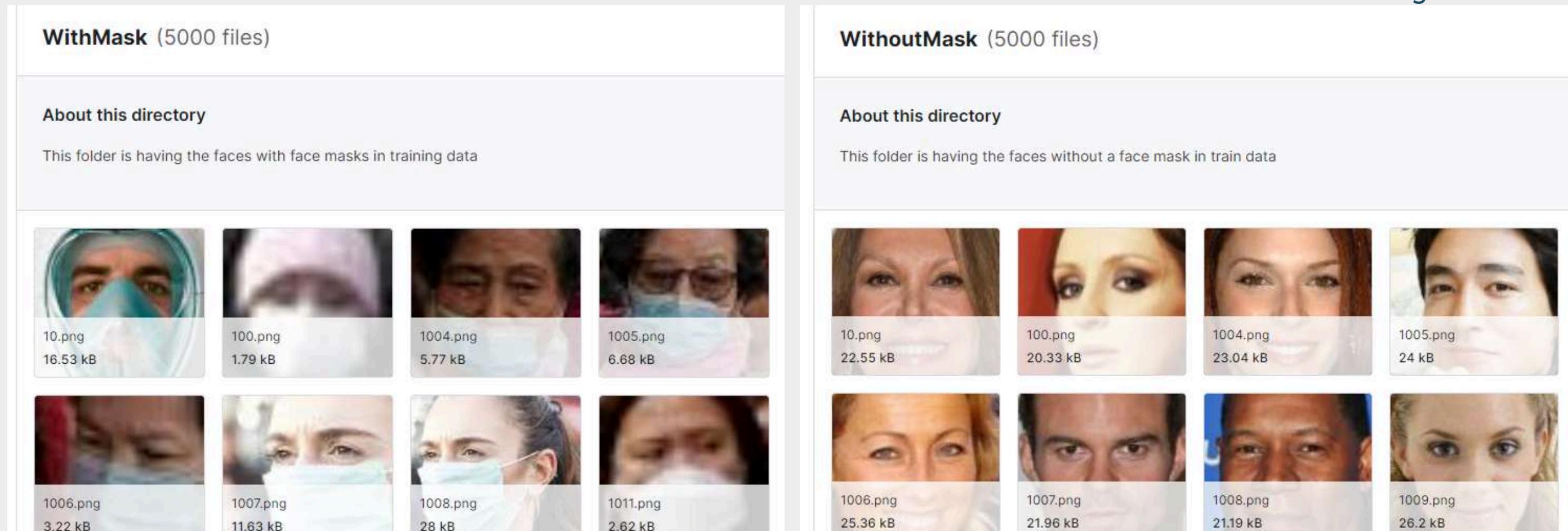


마스크 인식 알고리즘

## 데이터셋 : 총 11,822장의 이미지

- 훈련 데이터 : 총 10,000장의 이미지 (마스크 착용 이미지 : 5,000장, 마스크 미착용 이미지 : 5,000장)
- 검증 데이터 : 총 830장의 이미지 (마스크 착용 이미지 : 415장, 마스크 미착용 이미지 : 415장)
- 테스트 데이터 : 총 992장의 이미지 (마스크 착용 이미지 : 496장, 마스크 미착용 이미지 : 496장)

Images Dataset



- 데이터 출처(kaggle) : Face Mask Detection ~12K Images Dataset  
(<https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset>)

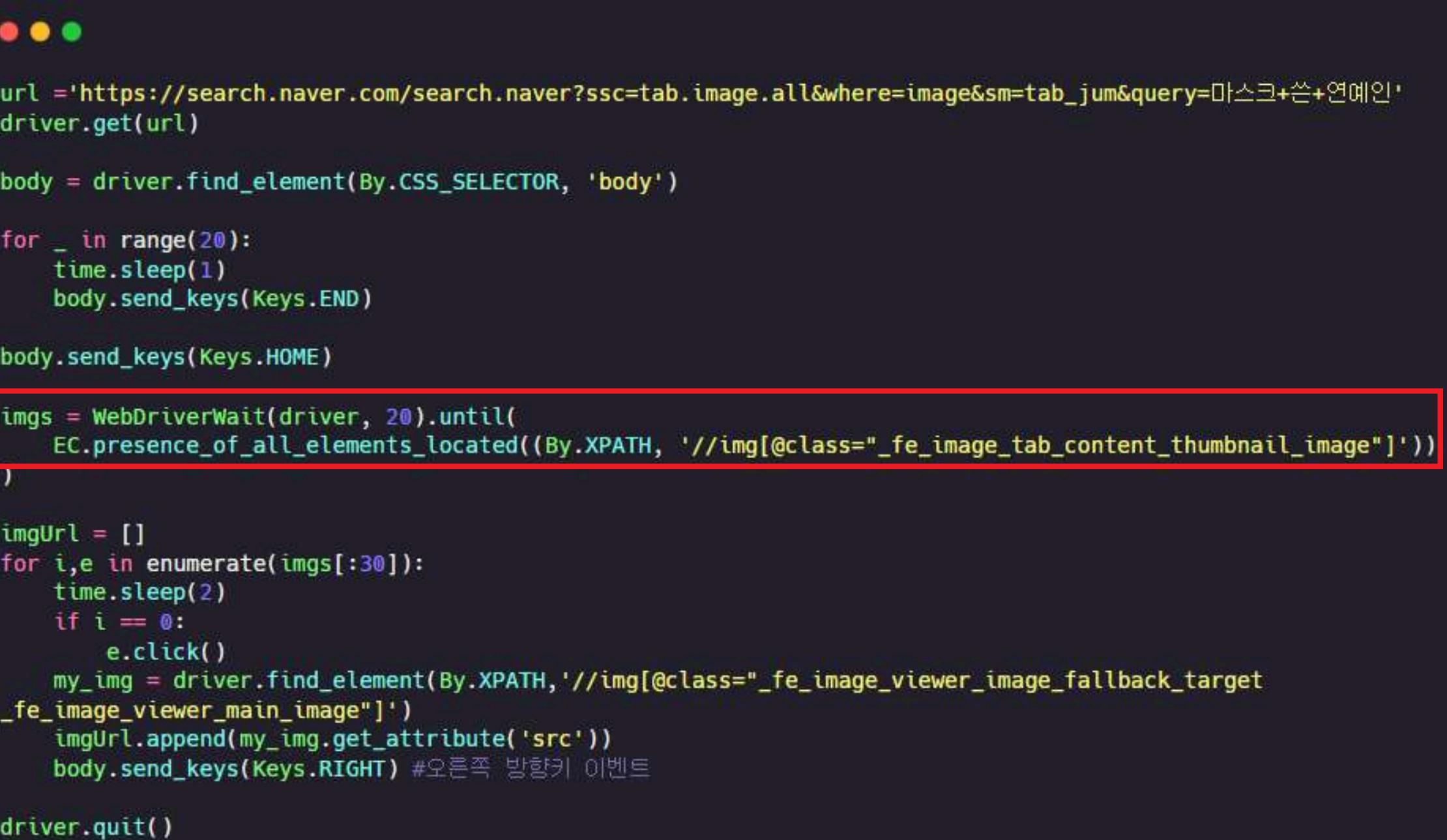
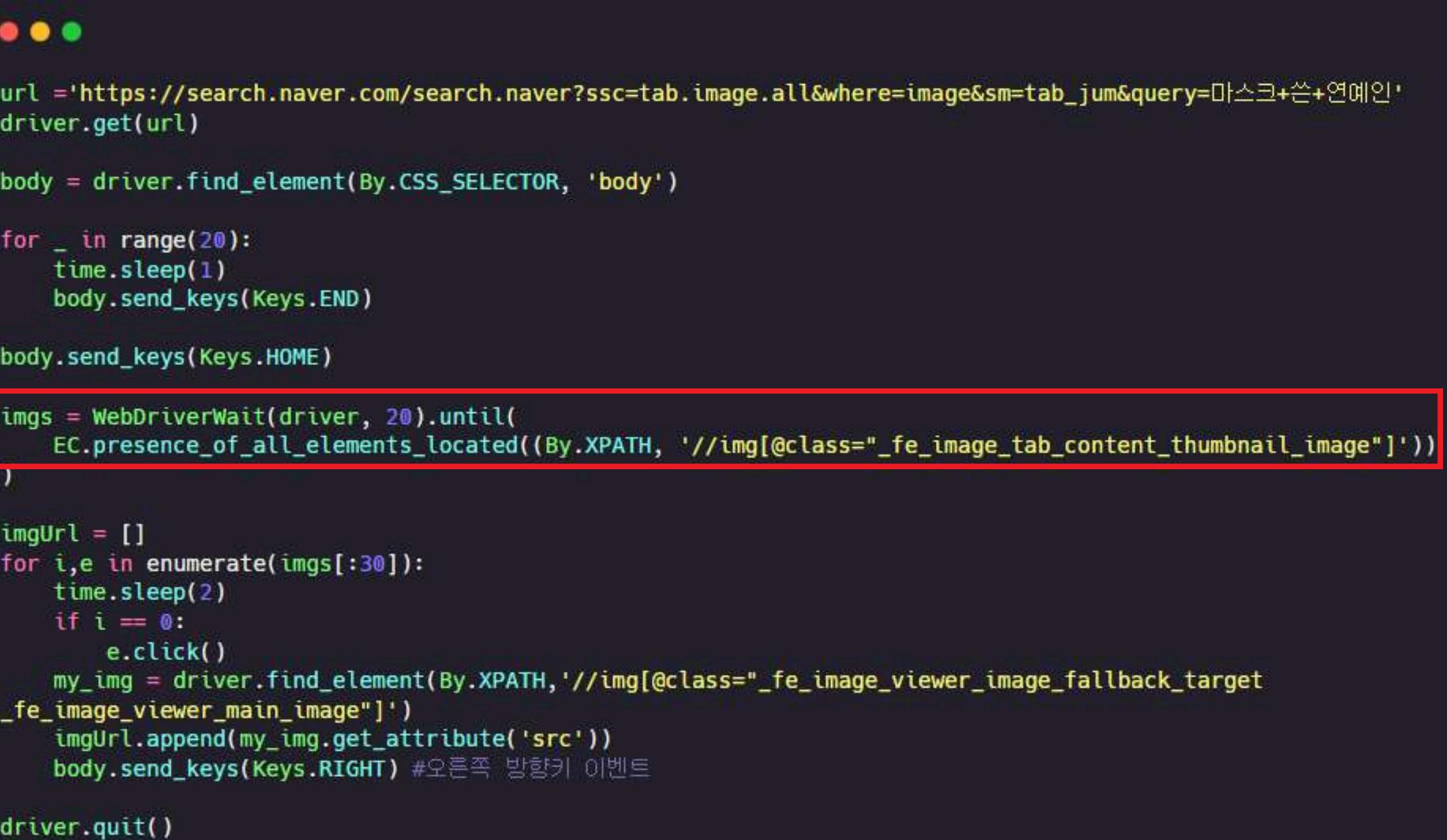
## 데이터셋 : 총 11,822장의 이미지

```
url ='https://search.naver.com/search.naver?ssc=tab.image.all&where=image&sm=tab_jum&query=마스크+쓴+연예인'
driver.get(url)

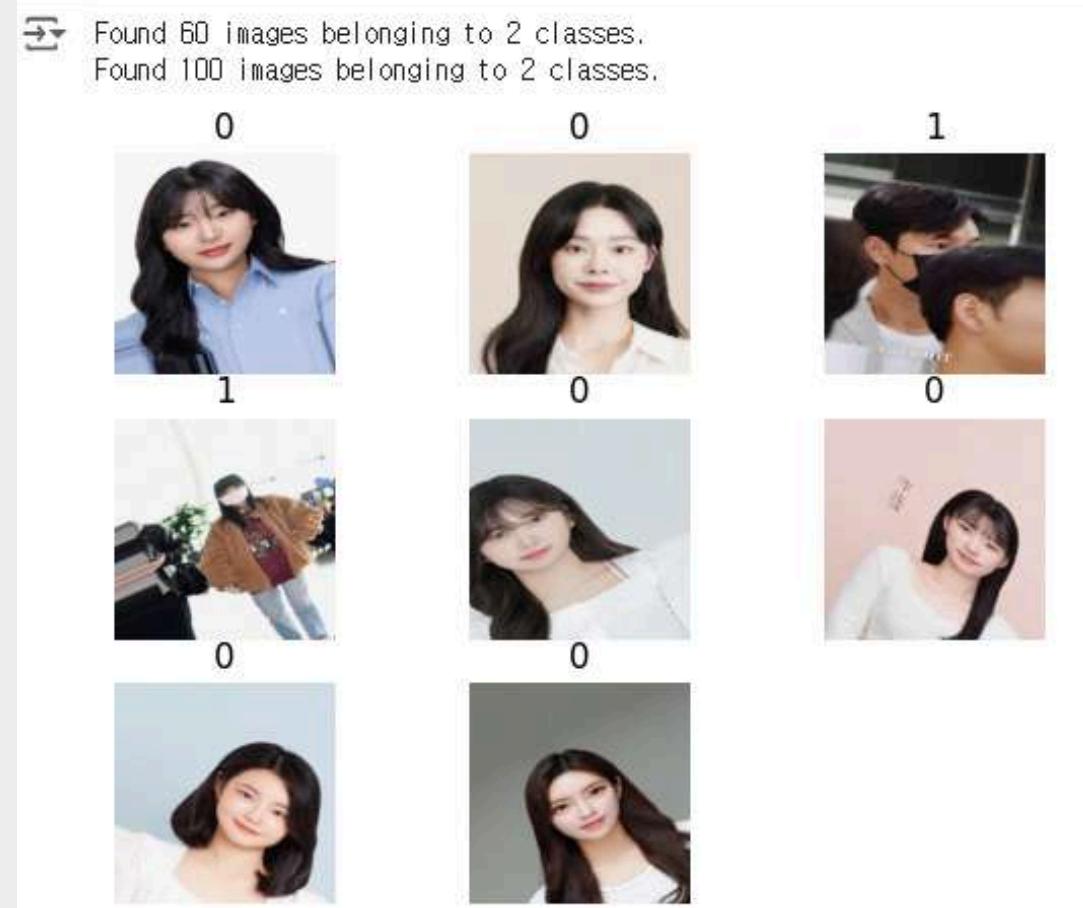
body = driver.find_element(By.CSS_SELECTOR, 'body')

for _ in range(20):
    time.sleep(1)
    body.send_keys(Keys.END)

body.send_keys(Keys.HOME)

```



웹 크롤링으로 이미지 데이터 증강

## Mask Detection

- 얼굴 인식 모델의 정확도는 사진 속 얼굴 위치와 촬영 각도의 다양성에 영향을 받음
- 이를 해결하기 위해 이미지 정규화와 다양한 데이터 증강 기법을 적용하여 전처리 과정을 선행함

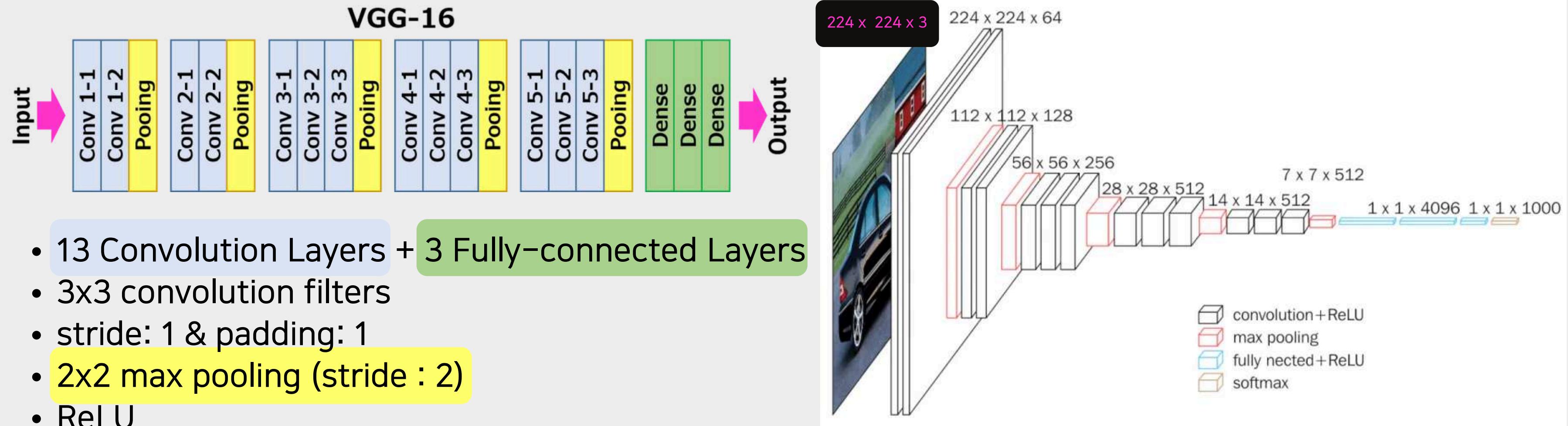


이미지 전처리 과정

## 모델 선정 : VGGNet-16

- ImageNet Challenge에서 Top-5 테스트 정확도를 92.7% 달성
  - AlexNet 테스트 오차율 (16.4%)
  - VGGNet-16 테스트 오차율 (7.3%)
- 딥러닝 기반 컴퓨터 비전 모델의 시대를 열었던 AlexNet(2012)의 8-layers 모델보다  
깊이가 2배 이상 깊은 네트워크의 학습에 성공한 신경망 모델

VGG-16 Architecture



## 훈련 데이터 : 데이터 증강 기법 적용

```
# 훈련 데이터 처리
training_datagen = ImageDataGenerator(
    rescale = 1./255, # 이미지를 0에서 1 사이로 조정
    rotation_range = 10, # 이미지를 무작위로 회전 (0-10도)
    width_shift_range = 0.2, # 좌우 이동 범위
    height_shift_range = 0.2, # 상하 이동 범위
    shear_range = 0.2, # 전단 변환 범위
    zoom_range = 0.2, # 이미지를 무작위로 확대/축소 범위 (0.8-1.2 배율)
    brightness_range = (0.5, 1.3), # 밝기 조절
    horizontal_flip = True, # 수평 뒤집기
    fill_mode ='nearest' # 빈 공간을 근처에 있는 픽셀로 채우기
)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    batch_size = batch_size, # 한 번에 반환할 이미지의 개수
    target_size = (224, 224), # 이미지 리사이징 (너비, 높이)
    class_mode = 'categorical', # # 다중 클래스 분류 : 'categorical'
    shuffle = True,
    seed=10
)
```

## 검증 및 테스트 데이터 : 단순 정규화 적용

```
# 검증 데이터 처리 (이미지를 0에서 1 사이로 조정 (정규화))
validation_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    batch_size = batch_size, # 한 번에 반환할 이미지의 개수
    target_size = (224, 224), # 이미지 리사이징 (너비, 높이)
    class_mode = 'categorical', # 다중 클래스 분류 : 'categorical'
    shuffle = False
)

# 테스트 데이터 처리 (이미지를 0에서 1 사이로 조정 (정규화))
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    TEST_DIR,
    batch_size = batch_size, # 한 번에 반환할 이미지의 개수
    target_size = (224, 224), # 이미지 리사이징 (너비, 높이)
    class_mode = 'categorical', # 다중 클래스 분류 : 'categorical'
    shuffle = False
)
```

- 이미지에 다양한 변형을 적용 (회전, 이동, 전단, 확대/축소, 밝기 조절, 수평 뒤집기)
- 이미지를 동일한 형태와 크기로 변경 (정규화 및 VGG-16 모델의 사이즈로 설정  
`target_size = (224, 224)`)

## VGG16 모델 불러오기

```
# VGG16 모델 불러오기
vgg_base = VGG16(weights = 'imagenet',
                   include_top = False,
                   input_shape = (224, 224, 3))

# VGG16의 가중치 설정
vgg_base.trainable = False
```

- weights = 'imagenet' **사전 학습된 가중치 사용**
- include\_top = False **최상위 레이어 포함하지 않음**
- trainable = True = False **가중치가 업데이트 되지 않도록 설정**  
(추가 레이어에 대해서만 그래디언트를 계산)

## 전체 모델 구성

```
model = Sequential([
    vgg_base, # VGG16 기본 모델
    GlobalAveragePooling2D(), # 전역 평균 풀링
    Dropout(.25), # 드롭아웃 추가
    BatchNormalization(), # 배치 정규화
    Dense(64, activation = 'relu'), # 완전 연결 레이어
    BatchNormalization(), # 배치 정규화
    Dropout(.5), # 드롭아웃 추가
    Dense(2, activation = 'softmax') # 출력 레이어
])
```

- GlobalAveragePooling2D **전역 평균 풀링 레이어**
- Dropout(.25), Dropout(.5) **드롭아웃 레이어**  
(25%, 50% 비율)
- BatchNormalization **배치 정규화 레이어**
- Dense(64, activation = 'relu') **완전 연결 레이어**  
(64개 뉴런, ReLU 활성화 함수)
- Dense(2, activation = 'softmax') **출력 레이어**  
(2개 클래스, 소프트맥스 활성화 함수)

# Mask Detection

## 모델 컴파일 설정

```
# 학습률 설정  
initial_learning_rate = 0.001  
  
# Adam 옵티마이저 인스턴스 생성  
optimizer = Adam(learning_rate = initial_learning_rate)  
  
# 모델 컴파일 설정  
model.compile(optimizer = optimizer,  
              loss = 'categorical_crossentropy',  
              metrics = ['accuracy'])
```

- learning\_rate = 0.001

학습률 설정 (초기 학습률 0.001)

- optimizer = Adam(learning\_rate = initial\_learning\_rate)

optimizer = optimizer

Adam 옵티마이저 생성하여 컴파일시 사용하도록 설정

- loss = 'categorical\_crossentropy'

손실함수 설정

- metrics = ['accuracy']

평가지표 설정 (정확도)

## 모델 구조 확인

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688 입력층
global_average_pooling2d ( GlobalAveragePooling2D )	(None, 512)	0
dropout (Dropout)	(None, 512)	0
batch_normalization (Batch Normalization)	(None, 512)	2048 은닉층
dense (Dense)	(None, 64)	32832
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130 출력층

```
Total params: 14749954 (56.27 MB)  
Trainable params: 34114 (133.26 KB)  
Non-trainable params: 14715840 (56.14 MB)
```

학습 가능 파라미터

### 콜백 함수 정의 1 : 훈련 중 가장 좋은 성능의 모델 가중치를 저장

```
# ModelCheckpoint 콜백 설정
checkpoint = ModelCheckpoint(
    model_path,
    monitor = 'val_accuracy',
    verbose = 1,
    save_best_only = True,
    mode = 'max'
)
```

- monitor = 'val\_accuracy' 검증 정확도 모니터링
- verbose = 1 저장 과정 상세 메시지 출력
- save\_best\_only = True 최적의 검증 성능을 가진 모델만 저장
- mode = 'max' 최대 정확도일 때 저장

### 콜백 함수 정의 2 : 훈련 중 검증 성능이 향상되지 않을 때 훈련을 조기에 중단

```
# EarlyStopping 콜백 설정
early_stopping = EarlyStopping(
    monitor = 'val_accuracy',
    min_delta = 0,
    patience = 3,
    verbose = 1,
    restore_best_weights = True
)
```

- verbose = 1 중단 메시지 출력
- min\_delta = 0 성능 변화가 0보다 크면 개선으로 간주
- patience = 3 3번의 epoch 동안 성능 향상이 없을 시 중단
- restore\_best\_weights = True 훈련 중 가장 좋은 성능의 모델의  
가중치를 ModelCheckpoint로 저장하고 메모리에 유지

# Mask Detection

## 모델 훈련 및 저장

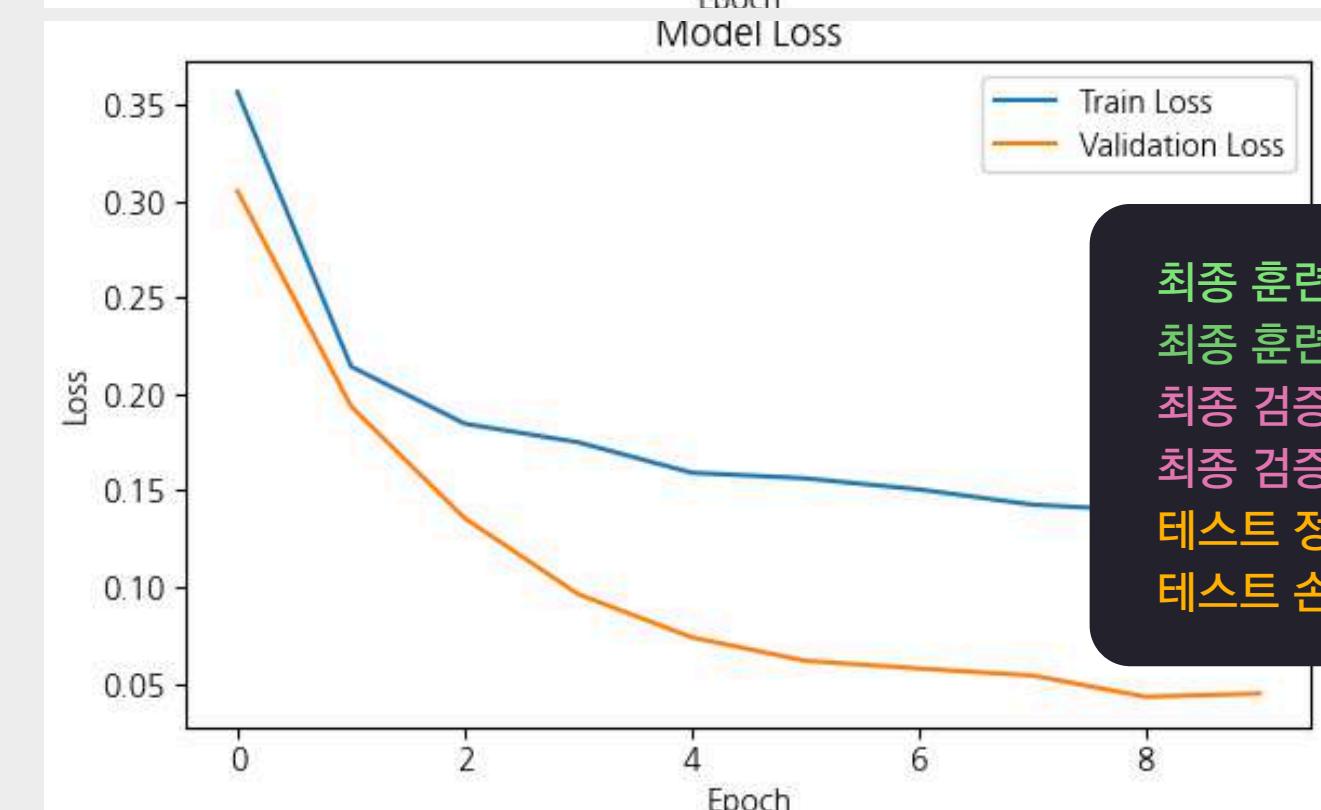
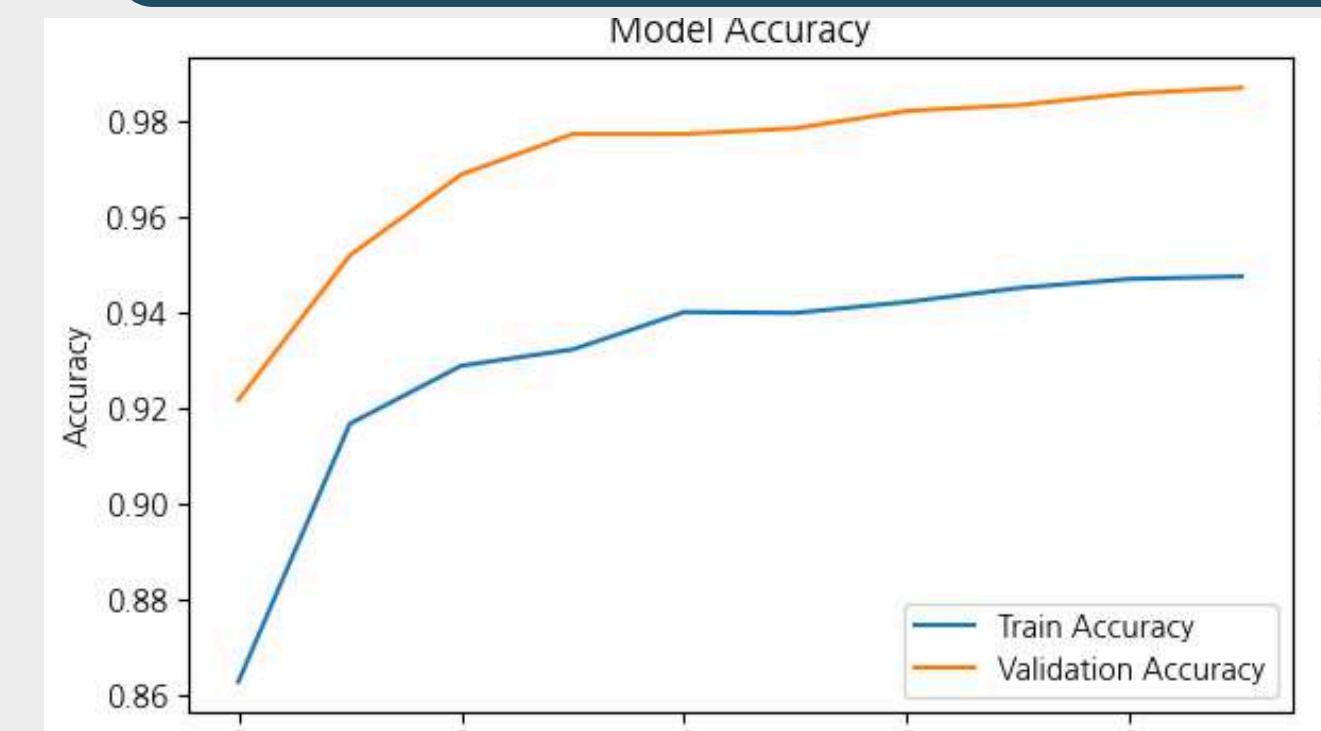
```
# 모델 훈련
history = model.fit(
    train_generator,
    epochs=10, # 에폭 수
    validation_data=validation_generator, # 검증 데이터
    verbose=1,
    callbacks=[checkpoint, early_stopping] # 콜백 함수
)
```

### 훈련 로그 확인

```
Epoch 1/10
79/79 [=====] - ETA: 0s - loss: 0.3566 - accuracy: 0.8629
Epoch 1: val_accuracy improved from -inf to 0.92169, saving model to
/content/drive/MyDrive/Colab Notebooks/Project(Face-Mask-
Detection)/models/vgg16_Face_model.h5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning:
You are saving your model as an HDF5 file via `model.save()`. This file format is
considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
    saving_api.save_model(
79/79 [=====] - 189s 2s/step - loss: 0.3566 - accuracy: 0.8629
- val_loss: 0.3052 - val_accuracy: 0.9217

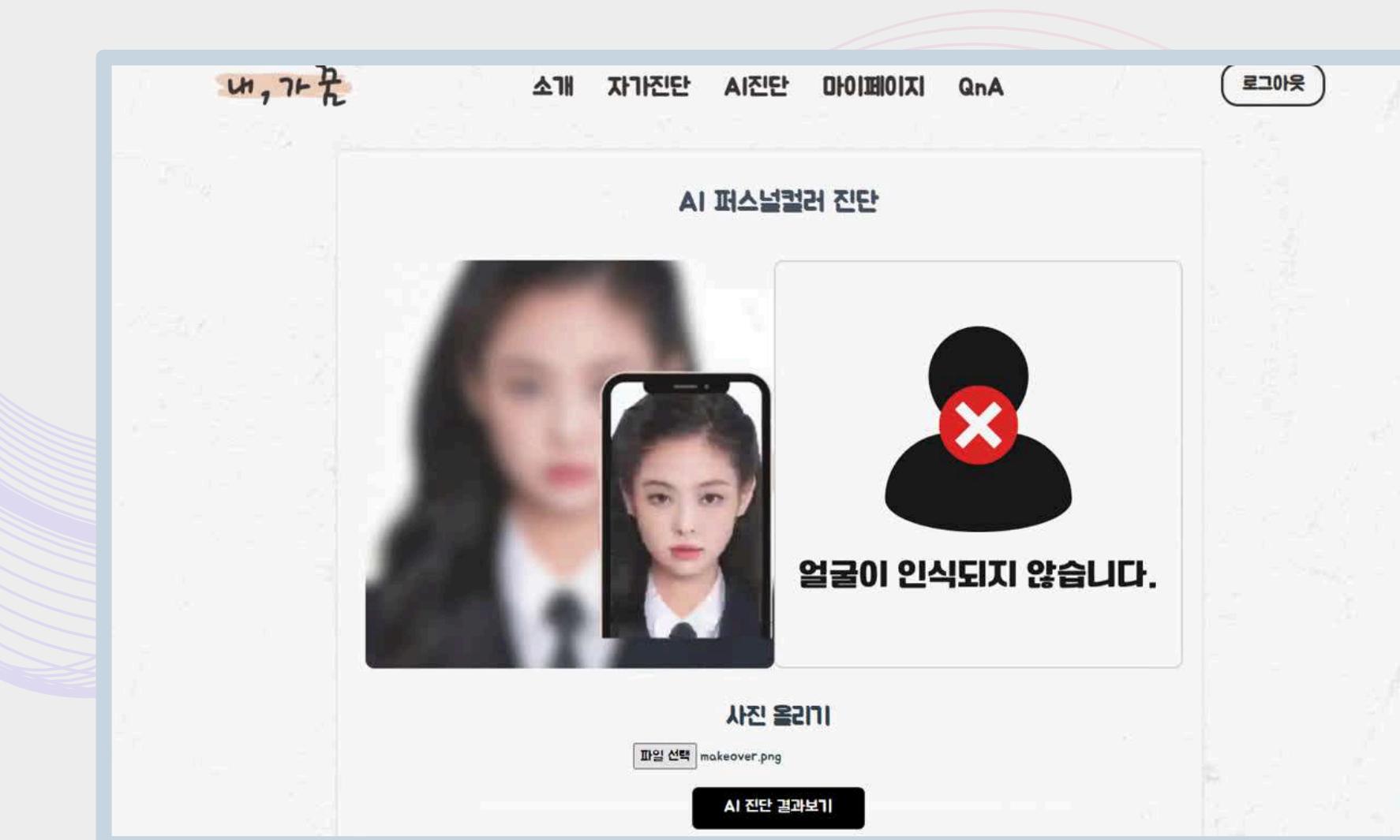
Epoch 10: val_accuracy improved from 0.98554 to 0.98675, saving model to
/content/drive/MyDrive/Colab Notebooks/Project(Face-Mask-
Detection)/models/vgg16_Face_model.h5
79/79 [=====] - 155s 2s/step - loss: 0.1388 - accuracy: 0.9474
- val_loss: 0.0448 - val_accuracy: 0.9867
```

## 모델 검증



최종 훈련 정확도 : 94.74%  
최종 훈련 손실 : 0.1388  
최종 검증 정확도 : 98.67%  
최종 검증 손실 : 0.0448  
테스트 정확도 : 98.39%  
테스트 손실 : 0.0386

## Image Upload



1 사람이 아닌 이미지

얼굴이 인식되지 않았습니다

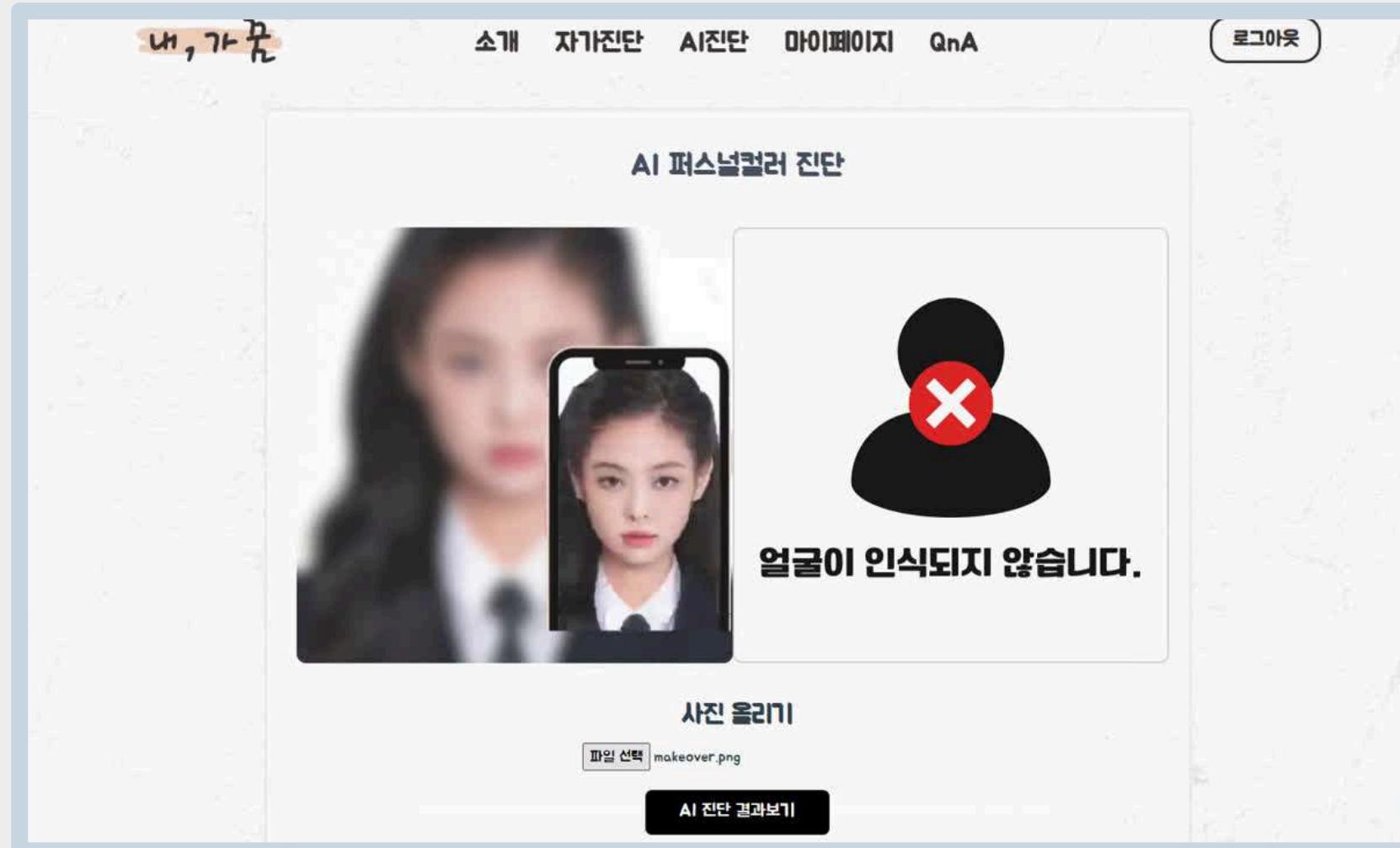
2 마스크를 쓴 이미지

마스크를 벗어주세요

3 마스크를 쓰지 않은 사람

이미지 정상 업로드

## Image Upload



```
<input type="file" accept="image/*" @change="uploadImage">
```

file이 @change되면 uploadImage 함수가 실행

```
uploadImage(event) {  
    const file = event.target.files[0];  
    const reader = new FileReader();  
    reader.readAsDataURL(file);  
  
    // FormData 생성 및 파일 추가  
    const formData = new FormData();  
    formData.append('imgfile', file);  
  
    // 서버로 이미지 전송  
    axios.post(`django 서버 주소/detect_mask`, formData)  
    .then((response) => {  
        const imageUrl = `data:image/jpeg;base64,${response.data.image}`;  
        this.uploadedImageUrl = imageUrl;  
    })  
}
```

{ 'Content-Type': 'multipart/form-data' }  
formData의 default Content-Type은 multipart/form-data 이미지를 담아서 장고 서버로 전송

- 이미지 파일을 업로드 하면 얼굴 인식, 마스크 착용 여부를 안내
- 이미지 파일 선택 시 uploadImage() 실행 → django서버로 이미지 전송

## 사전 학습된 모델 로드 및 이미지를 읽고 마스크 인식

```

model = tf.keras.models.load_model('/content/drive/MyDrive/MyProject/model/vgg16_Face_model.h5')

# 이미지 파일 읽기
image = cv2.imread('/content/drive/MyDrive/MyProject/D...')

# 이미지를 RGB 포맷으로 변환하여 얼굴 감지 모듈에 입력
res = face_detection.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

# 감지된 얼굴이 없을 경우
if image is None:
    print("Error: 이미지 로드 실패")
else:
    input_shape = (224, 224, 3)
    VGG-16 모델의 target size와 동일하게 맞춤

    resized_image = cv2.resize(face_image, (input_shape[1], input_shape[0]))
    preprocessed_image = resized_image.astype('float32') / 255.0
    preprocessed_image = np.expand_dims(preprocessed_image, axis=0)

    # 예측 수행
    predictions = model.predict(preprocessed_image)

    # 예측 결과 해석
    class_index = np.argmax(predictions, axis=1)[0]

    if class_index == 1:
        print("얼굴 영역에서 마스크 미착용이 감지되었습니다.")
    else:
        print("얼굴 영역에서 마스크 착용이 감지되었습니다.")

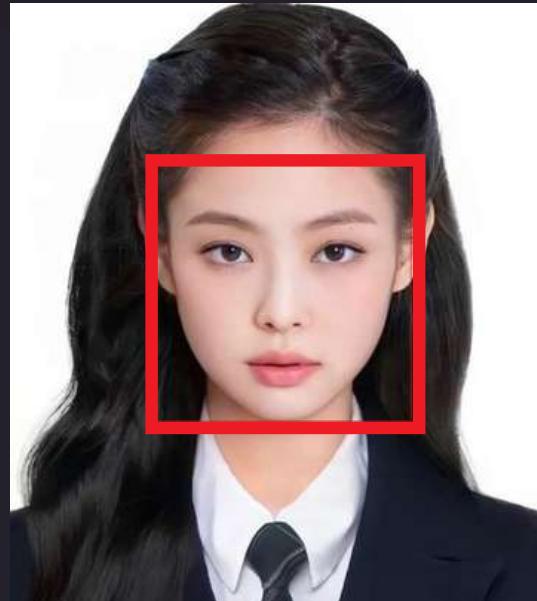
```

cv2.COLOR\_BGR2RGB  
cvtColor를 이용하여 RGB로 변환

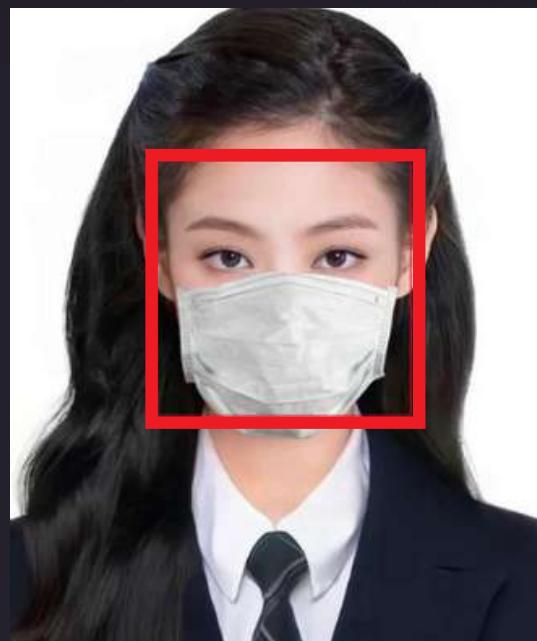
input\_shape = (224, 224, 3)  
VGG-16 모델의 target size와 동일하게 맞춤

np.argmax = (predictions, axis = 1)[0]  
가장 높은 확률을 가진 클래스의 인덱스 가져옴 (마스크 착용 : 1, 미착용 : 0)

1/1 [=====] - 0s 150ms/step  
얼굴 영역에서 마스크 미착용이 감지되었습니다.



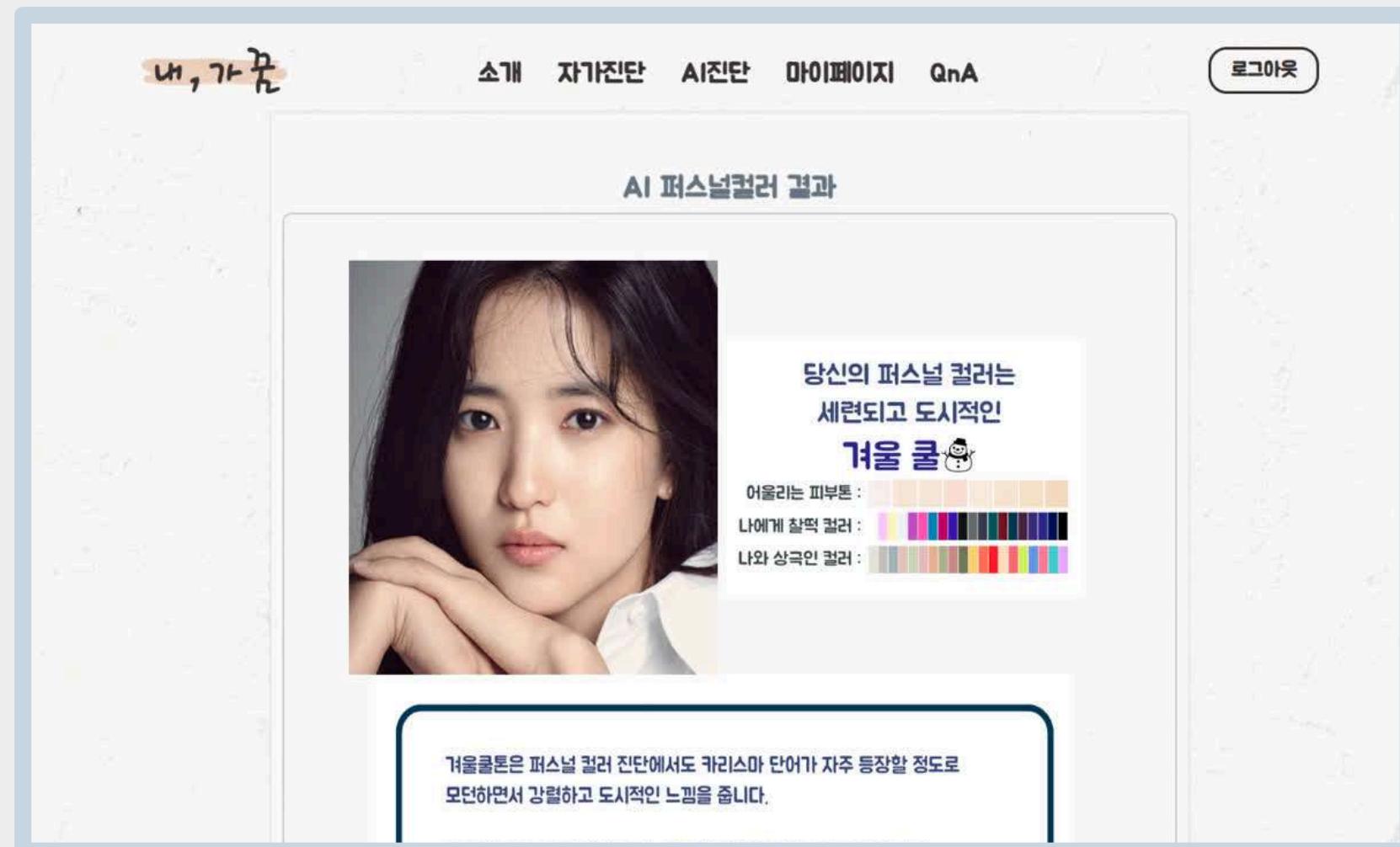
1/1 [=====] - 0s 152ms/step  
얼굴 영역에서 마스크 착용이 감지되었습니다.



# AI Personal Color Predict



## AI Personal Color Result



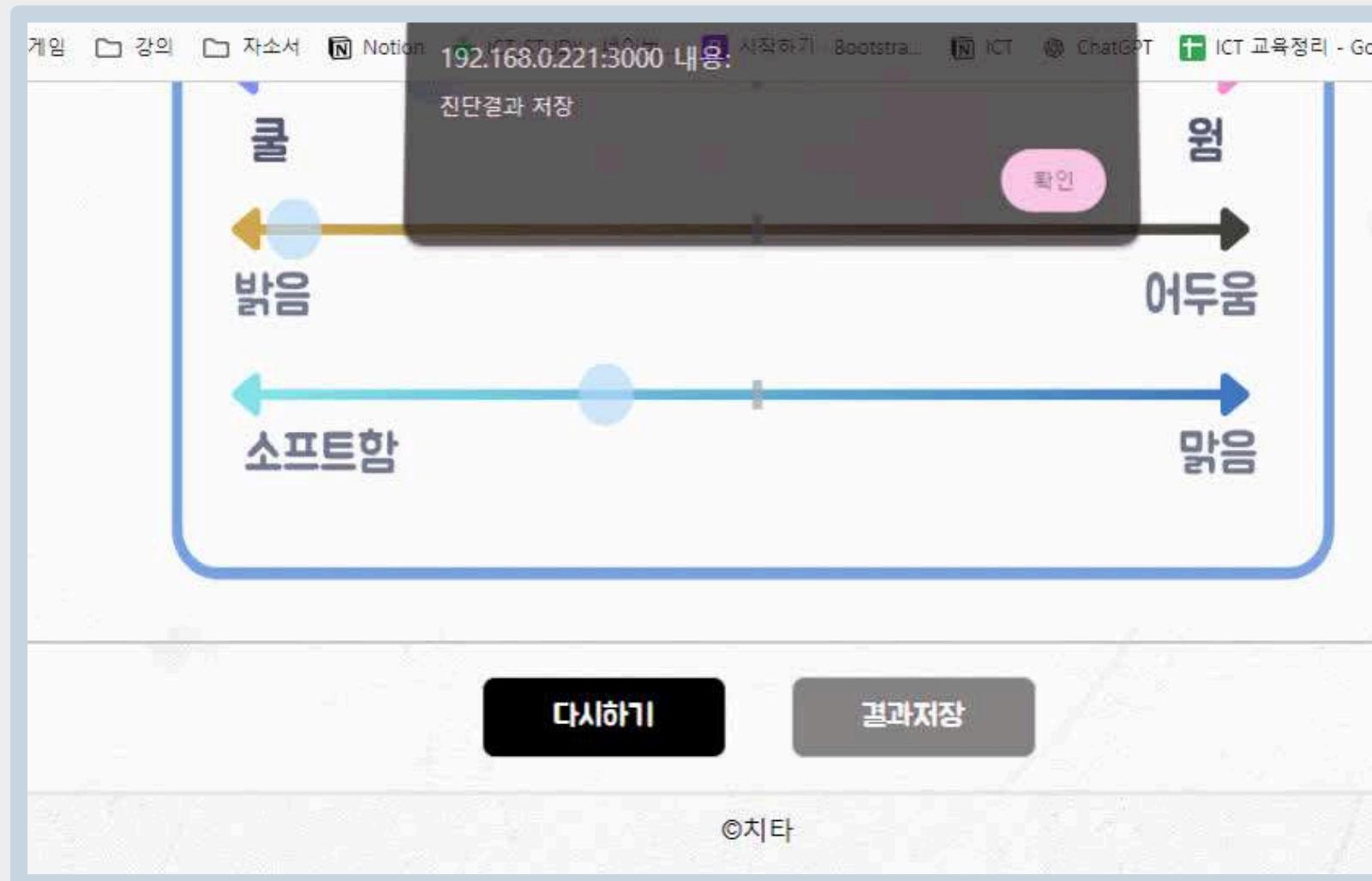
```
localStorage.setItem(key, value); // 로컬 스토리지에 저장  
localStorage.getItem(key); // 데이터 조회  
localStorage.removeItem(key); // 키에 해당되는 데이터 삭제  
localStorage.clear(); // 저장소 데이터 전체 삭제
```

```
reader.onload = (e) => {  
    localStorage.setItem('befimg', e.target.result);  
    localStorage.setItem('befimgn', file.name);  
};  
  
axios.get(`django 서버 주소}/seasontone`)  
.then((response) => {  
    console.log(response.data.season)  
    localStorage.setItem('season', response.data.season);  
  
    this.$router.push('/AIResult');  
})
```

- localStorage를 사용하여 데이터를 임시로 저장하고, 결과페이지로 이동
- 업로드한 이미지에 맞는 퍼스널컬러를 보여줌

## AI Personal Color Prediction

### AI Personal Color Save



```
<button @click="saveData">결과저장</button>
```

base64로 인코딩 된 이미지 데이터  
blob 형식으로 변환 후  
formData에 저장

```
saveData() {  
  alert("진단결과 저장");  
  const formData = new FormData();  
  formData.append('imgfile', this.dataURItoBlob(this.befimg), this.befimgn);  
  axios.post(`${django 서버 주소}/cuttingImage`, formData, {  
    headers: { 'Content-Type': 'multipart/form-data' },  
    responseType: 'blob' // blob으로 응답받기 위해 설정  
  })  
  .then(response => {  
    const url = URL.createObjectURL(response.data);  
    this.croppedImgUrl = url;  
    const formData = new FormData();  
    formData.append('croppedImg', response.data, this.befimgn);  
    axios.post(`${spring boot 서버주소}/ai/croppedImgSave`, formData)  
    .then(response => {  
      this.befimgn = response.data;  
      const jsonData = {  
        mnum: this.mnum,  
        befimgn: this.befimgn,  
        afimgn: 'defaultImg.jpeg',  
        season: localStorage.getItem('season')  
      };  
      axios.post(`${spring boot 서버 주소}/ai/saveTest`, jsonData)  
      .then(response => {  
        this.showMakeupButton = true;  
      })  
    })  
  })  
}
```

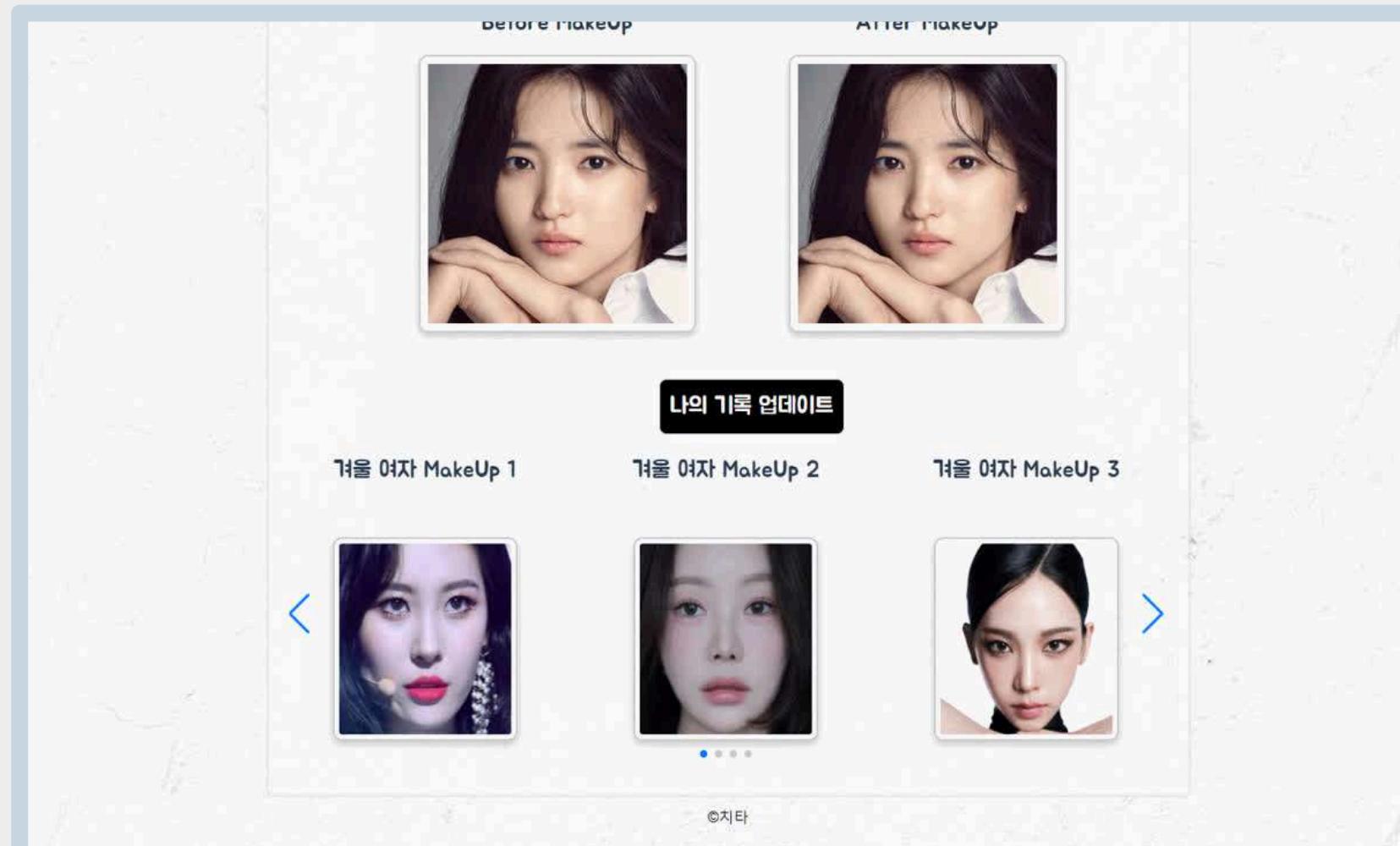
업로드 한 이미지의 얼굴부분만  
자른 후 장고서버에서 반환받음

반환받은 이미지 formData에 저장 후  
spring boot 서버로 전송해서  
vue 경로에 이미지 저장

AI 진단결과 Insert,  
Member의 season정보 Update

- 진단결과 저장을 누름 → 가상 메이크업 체험 버튼 활성화
- 얼굴 이미지 저장, 진단결과 Insert, 회원의 퍼스널컬러 정보Update

## AI Personal Color

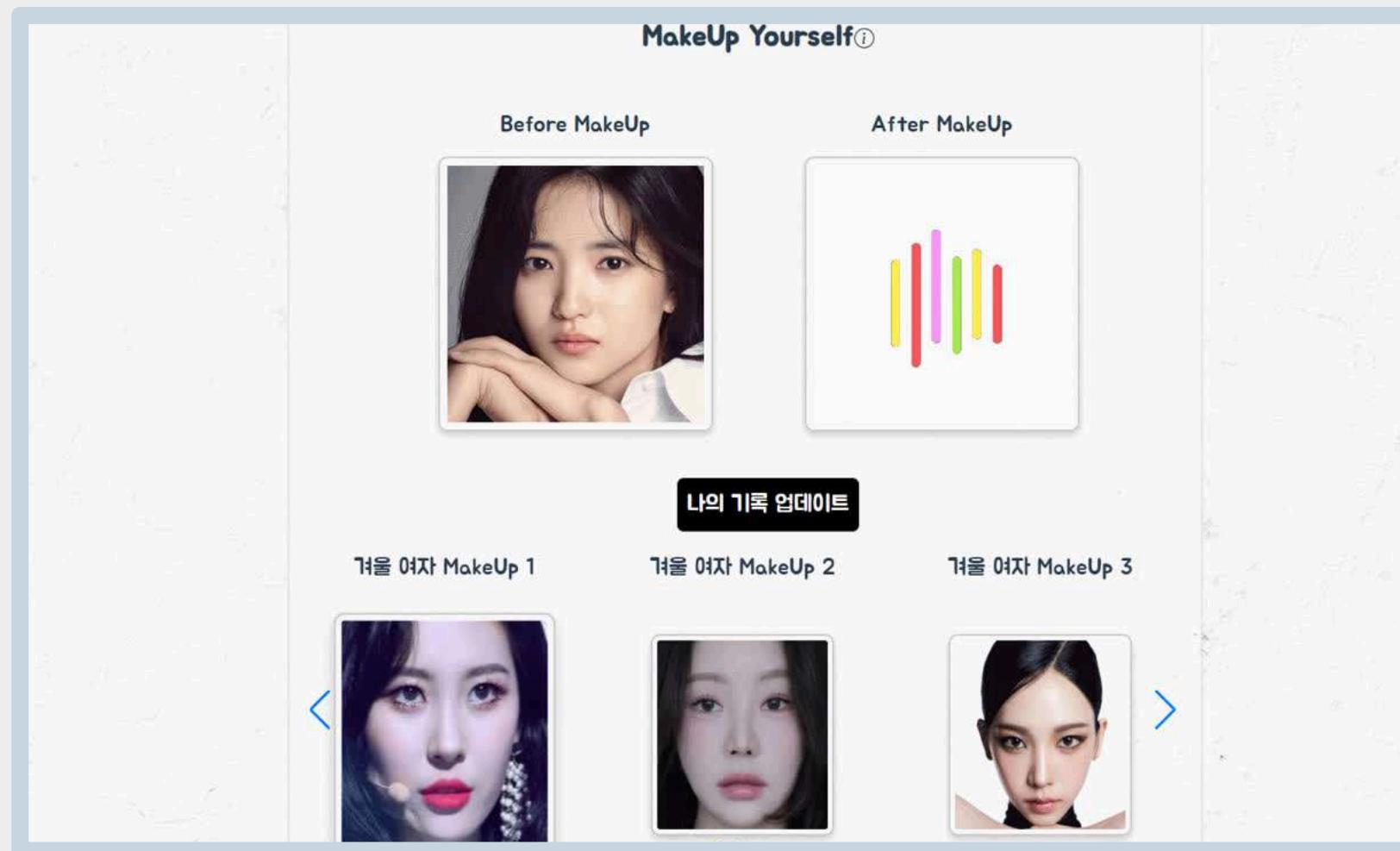


```
data() {
  return {
    mnum: '세션에 저장된 값',
    mgender: '',
    season: localStorage.getItem('season'),
    befimgn: localStorage.getItem('befimgn'),
  }
}
created() {
  axios.get(`boot서버주소}/members/detail?mnum=${this.mnum}`)
  .then((response) => {
    this.mgender = response.data.mgender
  })
}
saveData() {
  const returnrnum = axios.post(`boot서버주소}/ai/saveTest`, jsonData)
  this.rnum = returnrnum
}

this.$router.push({
  name:'MakeUp',
  params:{befimgn: this.befimgn,
          mnum:this.mnum,
          mgender:this.mgender,
          season:this.season,
          rnum:this.rnum}});
}
```

- params로 넘겨줄 데이터들을 localStorage, 데이터베이스에서 가져옴
- 가상 메이크업 체험 버튼을 누르면 이동

### AI BeautyGAN Model



```
<img @click="afterMakeup(getMakeupImage(index))">
```

```
afterMakeup(targetImage) {
  const beforeImage = this.loadImageAsBase64(this.beforeImage);
  const targetImage = this.loadImageAsBase64(this.targetImage);
  this.afterImage = this.loadingImage;

  axios.post(` ${django서버주소}/PersonalPredict` , {
    before_image: beforeImage,
    makeup_image: targetImage
  });
  .then(response => {
    const imageData = response.data.image_base64;
    this.afterImage = 'data:image/jpeg;base64,' + imageData;
  })
}
```

- 하단 이미지 클릭 시 Before MakUp 이미지에 메이크업 적용
- 하단 이미지와 Before MakeUp 이미지를 django서버로 전송 후 BeautyGAN 모델 사용

## 사전 학습된 모델 로드 및 이미지를 읽고 마스크 인식

```

# MediaPipe setup
mp_face_detection = mp.solutions.face_detection
mp_face_mesh = mp.solutions.face_mesh

before_makeup_img = cv2.imread("/content/drive/MyDrive/MyProject/Data/woman_spring_2.jpeg")
before_makeup_img_rgb = cv2.cvtColor(before_makeup_img, cv2.COLOR_BGR2RGB)
before_makeup_faces = align_faces(before_makeup_img_rgb)

target_makeup_img = cv2.imread("/content/drive/MyDrive/MyProject/Data/man_spring_1.jpeg")
target_makeup_img_rgb = cv2.cvtColor(target_makeup_img, cv2.COLOR_BGR2RGB)
target_makeup_faces = align_faces(target_makeup_img_rgb)

if img1_faces and img2_faces:
    before_img = before_makeup_faces[0]
    target_img = target_makeup_faces[0]

    X_img = preprocess(before_img)
    X_img = np.expand_dims(X_img, axis=0) # (256, 256, 3) -> (1, 256, 256, 3)

    Y_img = preprocess(target_img)
    Y_img = np.expand_dims(Y_img, axis=0) # batch dimension

    output = sess.run(Xs, feed_dict={X: X_img,
                                     Y: Y_img})
    output_img = postprocess(output[0])

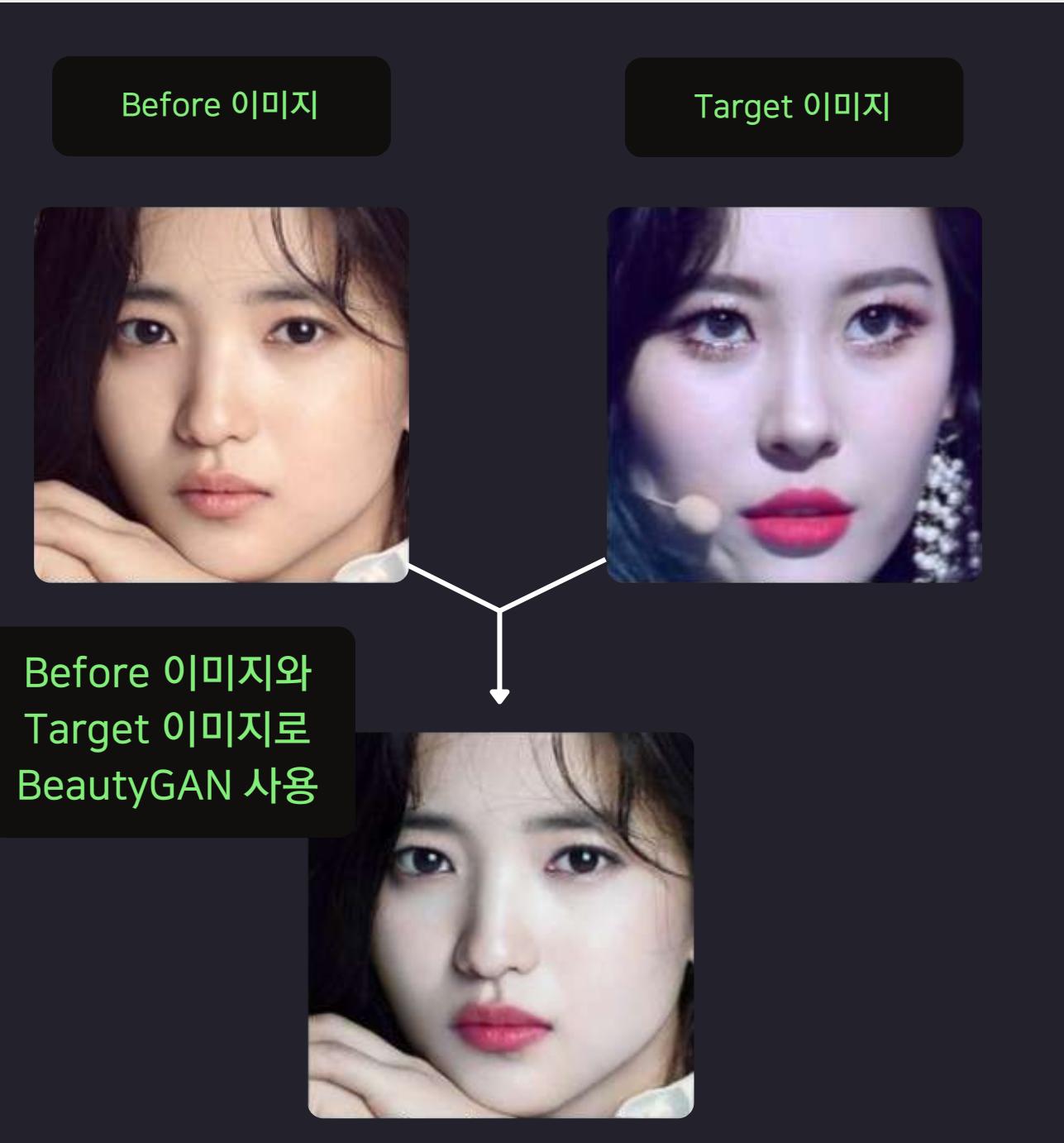
```

cv2.COLOR\_BGR2RGB  
cvtColor를 이용하여 두 이미지 모두 RGB로 변환

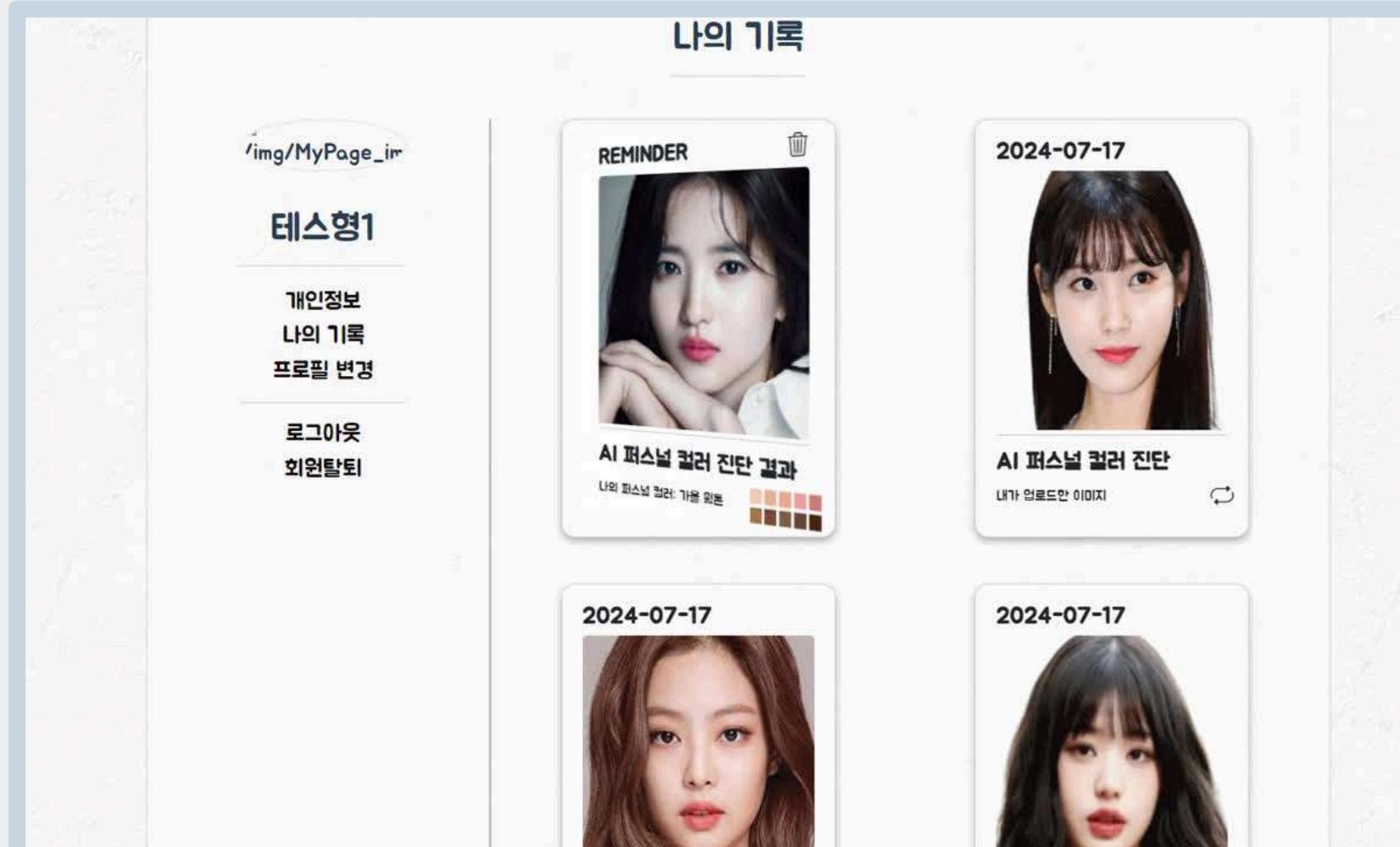
이미지 전처리

모델 실행

이미지 후처리



## AI BeautyGAN Model



```
<button type="button" @click="historyUpdate">나의 기록 업데이트</button>

historyUpdate() {
  const afterImage = this.loadImageAsBase64(this.afterImage);
  const formData1 = new FormData();
  formData1.append('afterimage', this.dataURIToBlob(afterImage), this.afimgn);

  const response = axios.post(`django서버주소}/base64toFile`, formData1, {
    headers: { 'Content-Type': 'multipart/form-data' },
    responseType: 'blob'
  });

  const formData2 = new FormData();
  formData2.append('imgfile', response.data, this.django);
  axios.post(`boot서버주소}/ai/after_imageSave`, formData2);

  const formData3 = new FormData();
  formData3.append('afimgn', this.afimgn);
  formData3.append('rnum', this.rnum);

  axios.post(`boot서버주소}/ai/updateHistory`, formData3);
  alert('업데이트 완료!');
  // History.vue로 라우팅
  this.$router.push({ name: 'History', params: { mnum: this.mnum } });
}
```

base64로 인코딩 된 이미지를 파일형식으로 바꾸기 위해 django로 이미지 전송

vue 경로에 MakeUp 이미지 저장

나의 기록 업데이트

- 가상 메이크업을 충분히 진행한 후 나의 기록 업데이트 클릭
- 해당 퍼스널컬러 진단결과에 가상 메이크업이 저장된 이미지가 업데이트됨

# BeautyGAN model



# BeautyGAN

Facial Makeup Transfer Demystified: BeautyGAN

원본 이미지



타겟 이미지

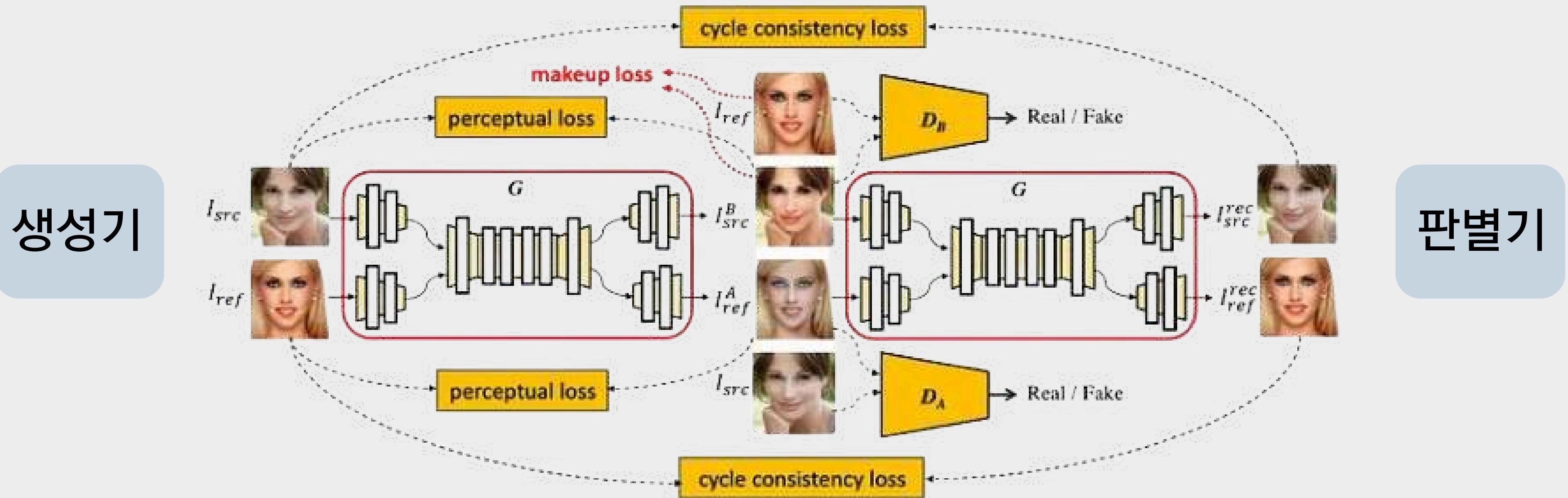


메이크업 이미지



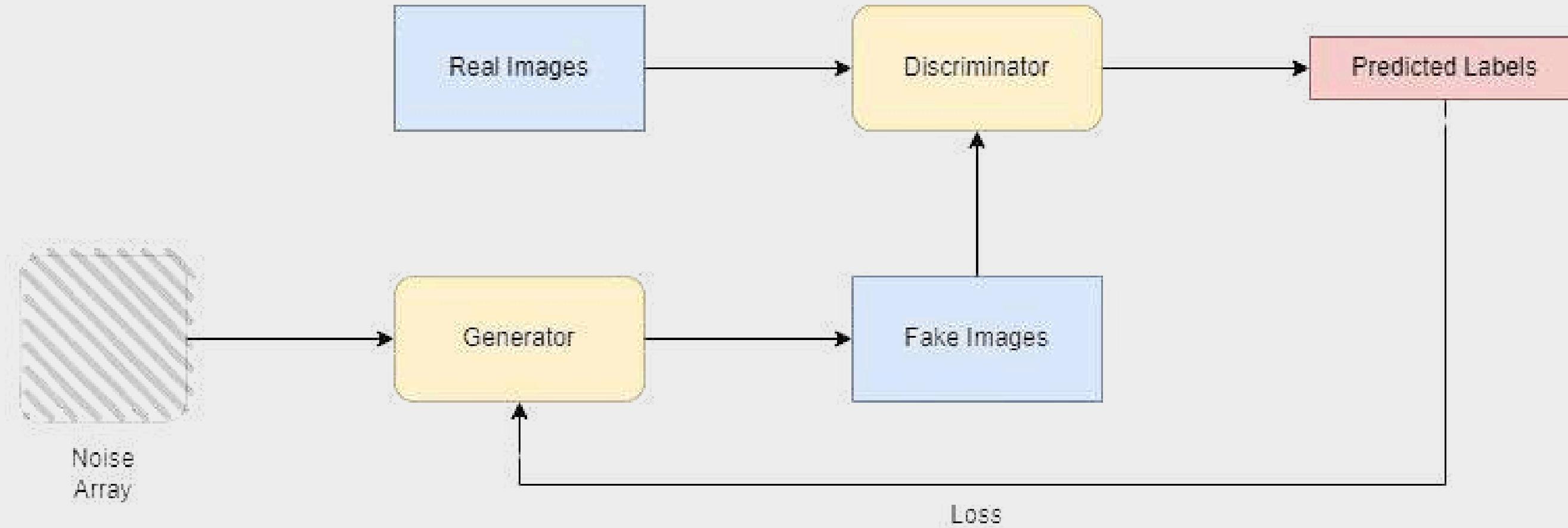
- 딥러닝 기반의 이미지 변환 모델
- 사람의 얼굴 이미지에 다양한 메이크업 스타일을 자동으로 적용하는 기술

# Network Architecture



- 생성기 (Generator) : 두 개의 입력 가지가 존재하며 중간에서 결합하여 잔차 블록에 입력
- 판별기 (Discriminator) : 두 개의 판별기가 동일한 아키텍처를 공유

## Loss Function

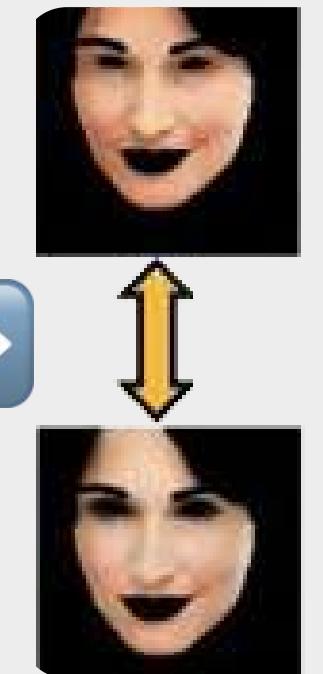
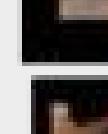
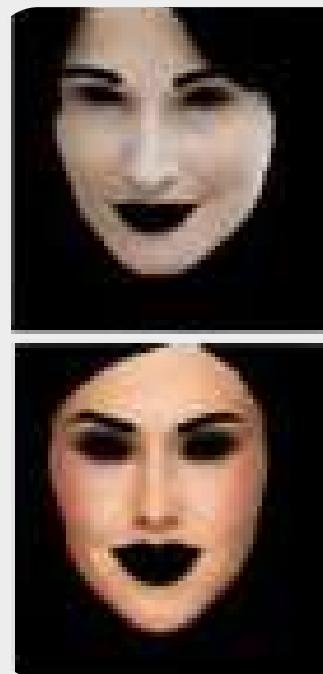


- 적대적 손실 (Adversarial Loss) : 시각적으로 만족스러운 이미지를 생성
- 지각적 손실 (Perceptual Loss) : 얼굴의 정체성과 구조를 유지
- 사이클 일관성 손실 (Cycle Consistency Loss) : 배경 정보를 유지합니다.
- 메이크업 손실 (MakeUp Loss) : 픽셀 수준의 히스토그램 손실을 통해 메이크업 스타일을 유지

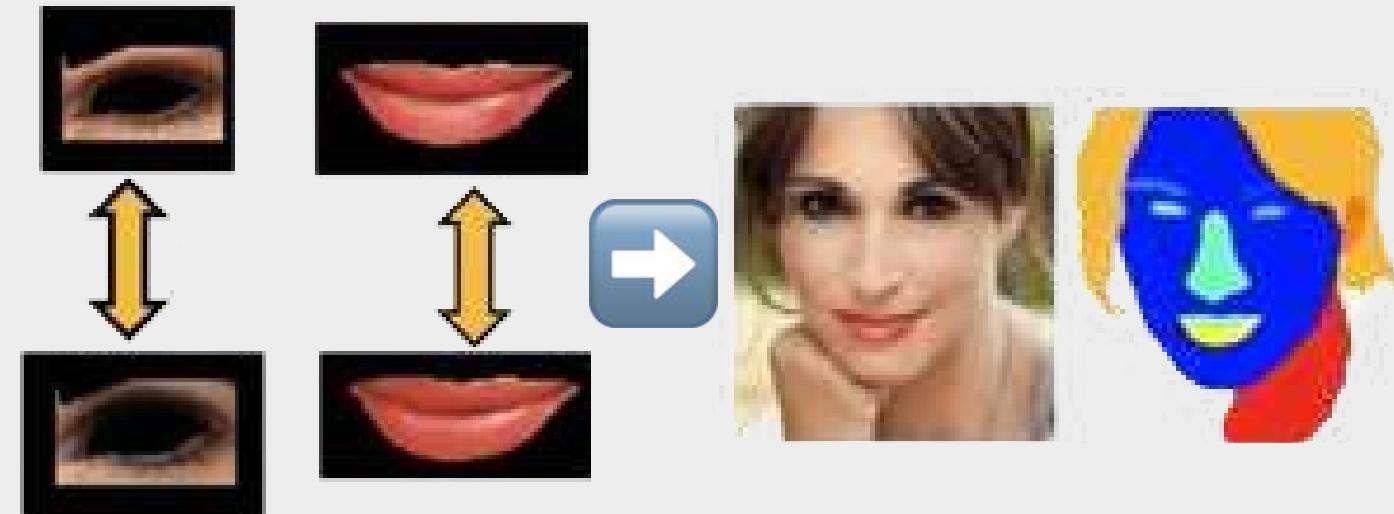
## MakeUp Loss

$$L_{makeup} = \lambda_f L_{face} + \lambda_s L_{shadow} + \lambda_l L_{lips}$$

히스토그램 매칭



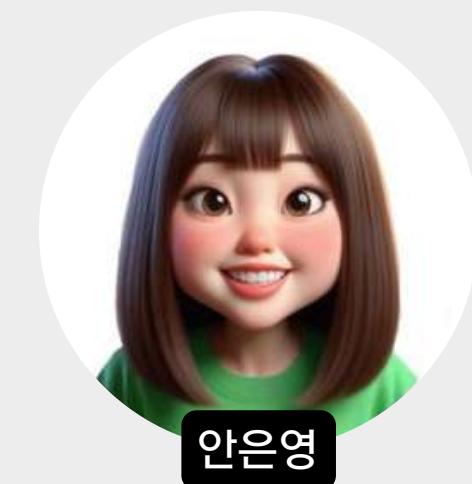
MakeUp Loss



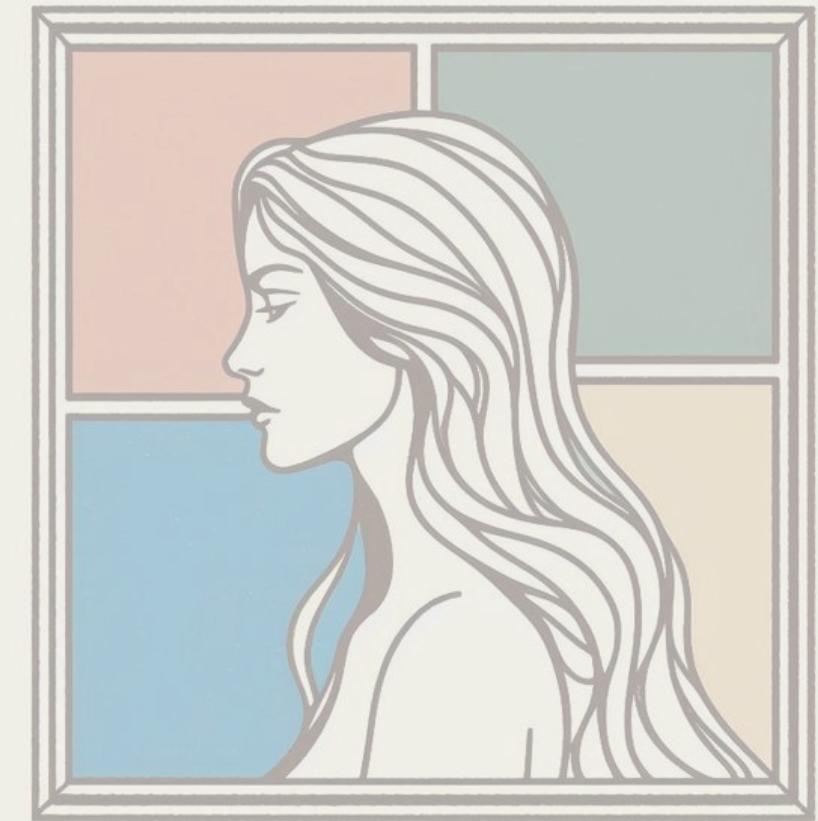
- 원본 이미지와 참조 이미지의 색상 분포를 맞추기 위해 히스토그램 매칭을 사용
- 얼굴, 눈, 입술 부위에 대한 개별적인 히스토그램 손실 적용
- 각 부위별 매칭된 이미지와 원본 이미지간의 MSE손실을 계산하여 최적화

# 이데일 인증

## -Mybatis



안은영



# Redis

## Redis란?

Redis는 일종의 캐시 시스템으로서 영속성, 다양한 데이터 구조와 같은 부가적인 기능을 함께 지원  
-> 모든 데이터를 메모리에 저장하고 조회(인메모리 데이터베이스)  
-> 고성능 키-값 저장소로서 문자열, 리스트, 해시, 셋 형식 등의 데이터를 지원하는 NoSQL

## 특징

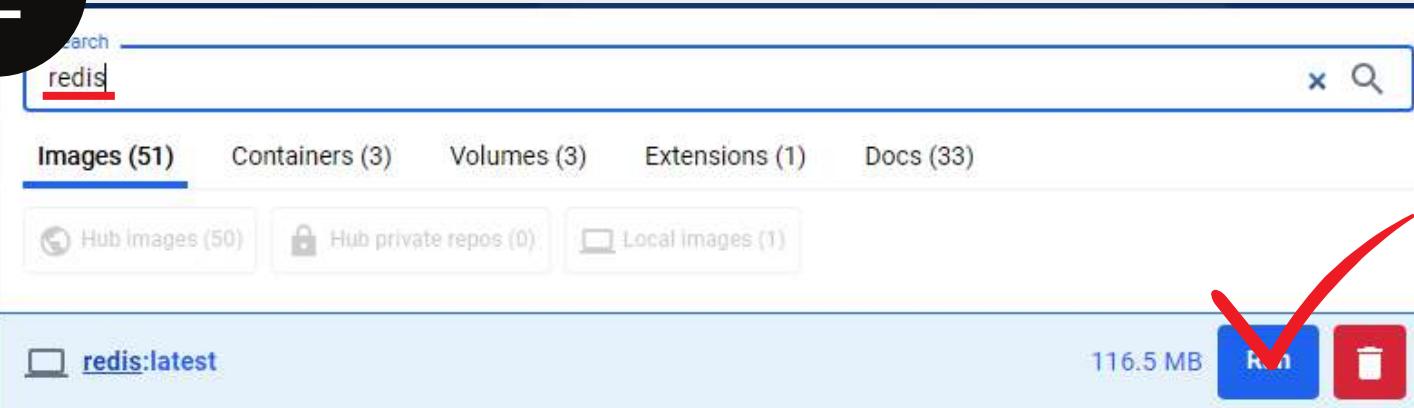
- 캐싱이 빠름
- 실시간 어플리케이션에 대한 처리도 가능
- 영속성을 지원하는 인메모리 데이터 저장소
- 읽기 성능 증대를 위한 서버 측 복제를 지원
- 쓰기 성능 증대를 위한 클라이언트 측 샤딩(Sharding) 지원
- 문자열, 리스트, 해시, 셋, 정렬된 셋과 같은 다양한 데이터형을 지원.  
->메모리 저장소임에도 불구하고 많은 데이터형을 지원하므로 다양한 기능을 구현

## 환경설정

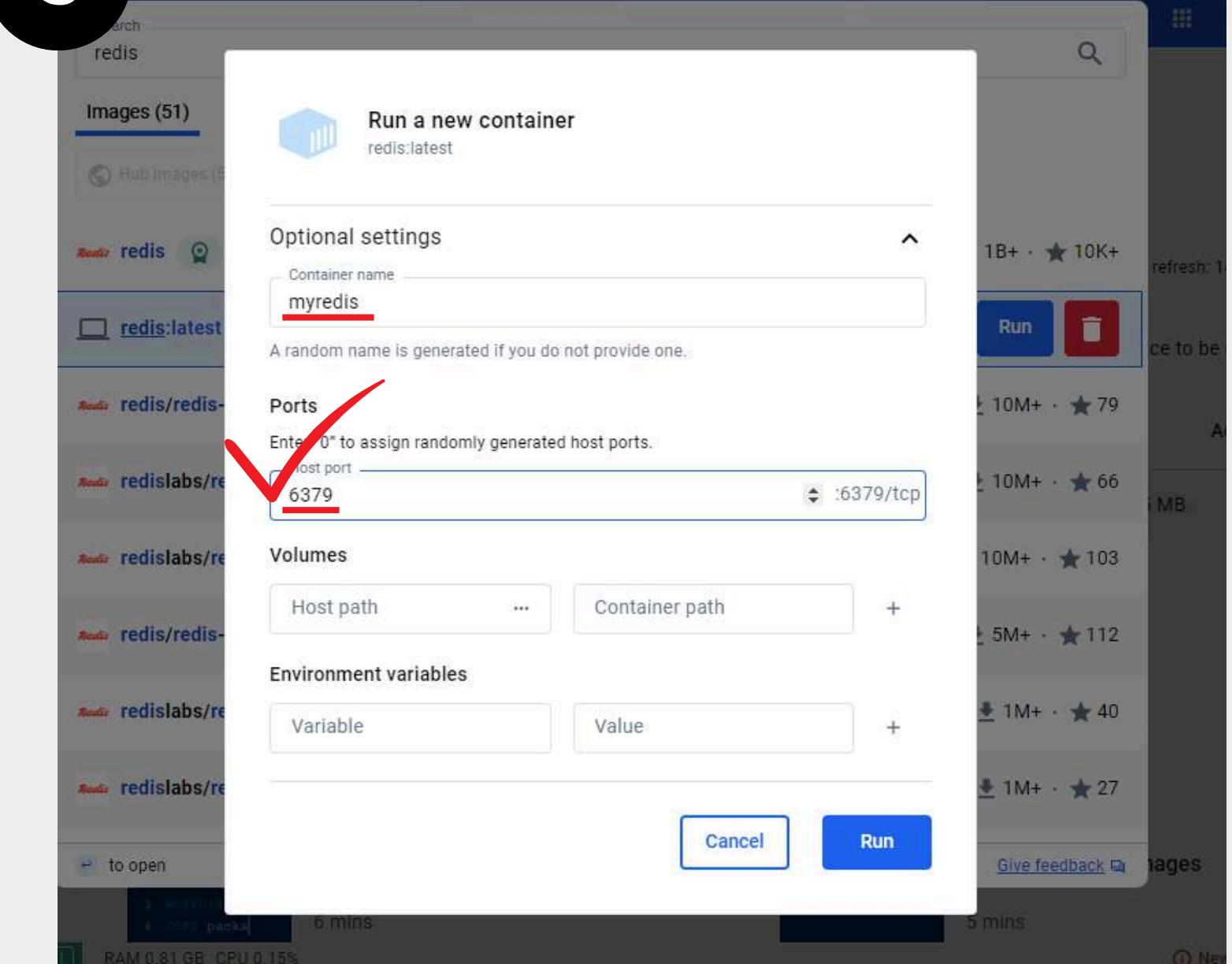
### 1 docker desktop 설치



### 2 docker desktop 실행 후 Redis 검색 및 설치



### 3



## Spring Boot 설정

build.gradle에 의존성 추가

### build.gradle

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-mail'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis:3.0.4'
}
```

application.properties에 redis 및 이메일 기본설정 추가

### application.properties

```
spring.mail.host=smtp.naver.com
spring.mail.port=465
spring.mail.username=your mail
spring.mail.password=your password
spring.mail.default-encoding=utf-8
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.data.redis.host=localhost
spring.data.redis.port=6379
Redis 를 활용한 회원 메일 인증 6
redis.property.host = localhost
spring.output.ansi.enabled=always
server-port-url = http://192.168.0.xx:3000/
server-port-local-url = http://localhost:3000/
```

### application.properties

#### ✓ 메일 설정

spring.mail.host: 이메일 서버의 호스트 주소를 설정  
spring.mail.port: 이메일 서버의 포트 번호를 설정  
spring.mail.username: 이메일 서버에 로그인할 때 사용할 사용자 이름  
spring.mail.password: 이메일 서버에 로그인할 때 사용할 비밀번호  
spring.mail.default-encoding: 이메일의 기본 인코딩을 설정  
spring.mail.properties.mail.smtp.auth: SMTP 인증을 사용할지 여부  
spring.mail.properties.mail.smtp.starttls.enable: STARTTLS 사용 여부

#### ✓ Redis 설정

spring.data.redis.host: Redis 서버의 호스트 주소를 설정  
spring.data.redis.port: Redis 서버의 포트 번호를 설정

## Vue

### emailCheck()

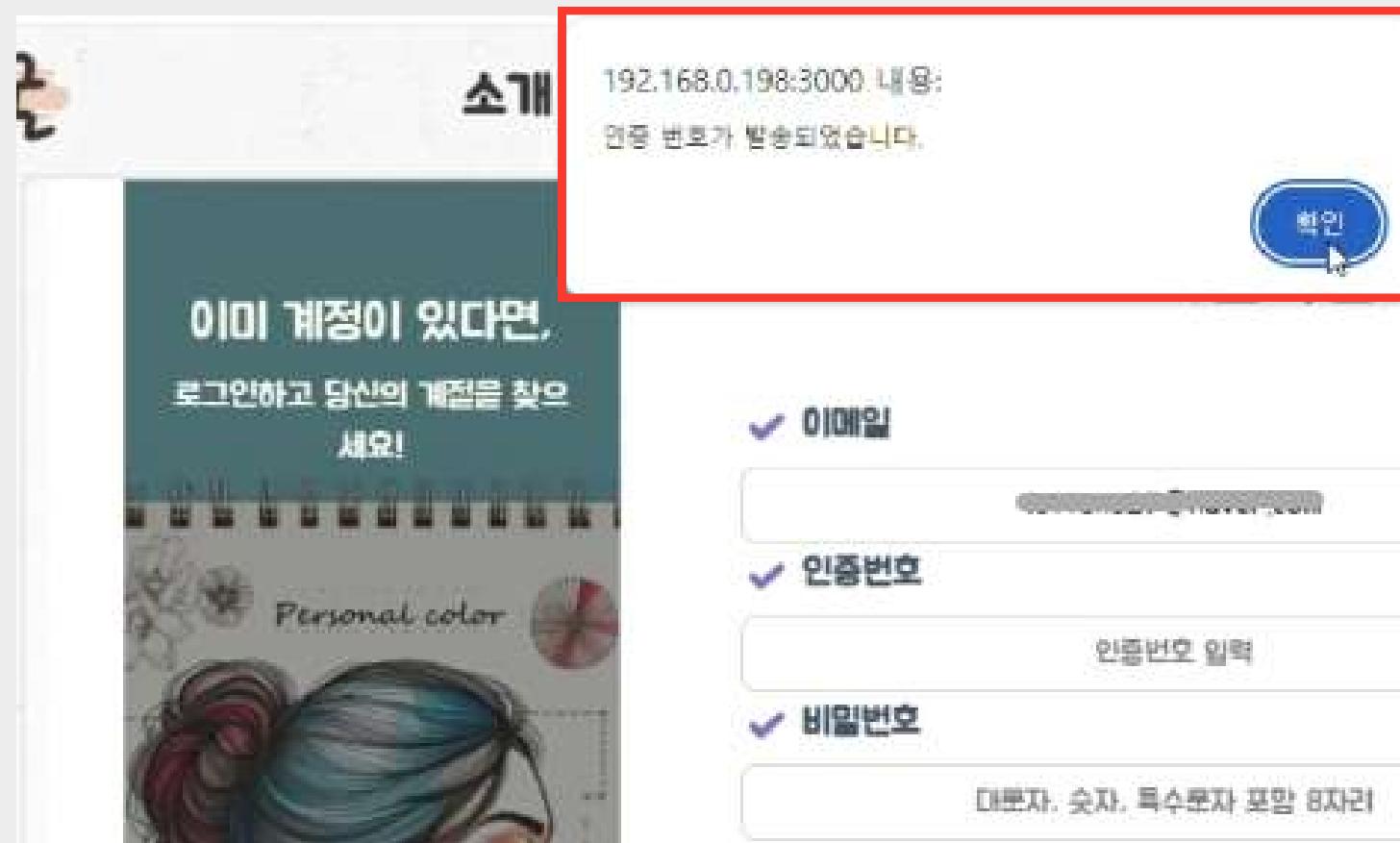
-> 이메일 중복체크 및 인증번호 발송

### submit signup()

-> signup submit 시

1. 이메일 인증 확인

2. form 제출



```
methods: {
  emailCheck() {
    const memail = document.getElementById("memail").value;
    axios .post(`process.env.VUE_APP_BACK_END_URL/api/auth/emailCheck`, {memail: memail})
      .then((res) => {
        if (res.data == 0) {
          alert("인증 번호가 발송되었습니다.");
          document.getElementById("emailCheck-msg").style.display = "none";
        } else {
          // <<중복 시 알림메세지 표시 로직 구현>>
        }
      });
    submitForm(formType) {
      if (formType === 'login') { // <<로그인 로직 구현>>
      } else if (formType === 'signup') {
        // <<회원가입 폼 제출 전 간단한 폼 체크로직 구현>>
        axios.post(`process.env.VUE_APP_BACK_END_URL/api/auth/emailCheck/certification`, {
          memail: this.memail,
          code: this.code, // 입력된 인증번호
        })
        .then((res) => {
          console.log(res);
          if (res.data === 1) {
            const formData = new FormData();
            formData.append("memail", this.memail);
            ...
            axios.post(`process.env.VUE_APP_BACK_END_URL/members/signup`, formData)
            .then((res) => {
              if (res.status === 200) {
                alert("회원가입이 완료되었습니다.");
                this.$router.push("/Login");
              } else {
                alert("회원가입 중 문제가 발생했습니다.");
              }
            })
            .else {
              // <<인증번호 불일치 처리 로직 구현>>
            }
          }
        })
      }
    }
  }
}
```

## Email check



```
@Setter  
@Getter  
@Alias("cvo")  
public class MailCheckVO {  
    private String memail;  
    private String code;  
}
```

이메일 중복체크를 위한 email 주소와 code를 받은 VO 생성



```
<mapper namespace="kr.co.ict2.ngg.dao.EmailCertificationDAO">  
    <select id="countByEmail" parameterType="String" resultType="int">  
        select count(*) from member where memail = #{memail}  
    </select>  
</mapper>
```

email count를 위한 mapper 작성



```
@Mapper  
public interface EmailCertificationDAO {  
    int countByEmail(String memail);  
}
```

mapper를 연결할 DAO



DAO 인터페이스를 구체화한 서비스 생성

```
@Service  
public class EmailSenderService {  
    @Autowired  
    private EmailCertificationDAO emailCertificationDAO;  
    public int duplicateEmail(String memail) {  
        long checkEmail = emailCertificationDAO.countByEmail(memail);  
        return checkEmail > 0 ? 1 : 0;  
    }  
}
```



버튼 클릭 시 변수 memail을 받아 email 체크 서비스 로직을 호출할 컨트롤러 생성

```
@RestController  
@RequestMapping("/api/auth")  
public class MailCertificationController {  
    @Autowired  
    private EmailSenderService emailSenderService;  
    @PostMapping("/emailCheck")  
    public int sendEmail(@RequestBody MailCheckVO memail) {  
        int checkEmail = emailSenderService.duplicateEmail(memail.getMemail());  
        if (checkEmail == 0) {  
            emailSenderService.sendEmail(memail.getMemail());  
            return 0;  
        } else {  
            return 1; 이메일 중복확인 후  
        }  
    }  
}
```

emailSenderService를 통해 인증메일 발송

## Email check



소개 자기진단 AI진단 마이페이지 QnA

회원가입

✓ 이메일 lottone21@naver.com

✓ 인증번호 c1EjvK

인증

```
public boolean isVerify(String memail, String authCode) {  
    if (certificationNumberDAO.hasKey(memail) && certificationNumberDAO  
        .getCertificationNumber(memail).equals(authCode)) {  
        certificationNumberDAO.deleteCertificationNumber(memail);  
        return true;  
    } else {  
        return false;  
    }  
}
```

이메일과 인증 코드가 일치하는지 확인

hasKey를 통해 이메일에 대한 인증 코드가 존재하는지 확인

getCertificationNumber를 통해 저장된 인증 코드와 일치하는지 확인

인증 코드가 일치하면 인증 코드를 삭제하고 true를 반환합니다.

인증 코드가 일치하지 않으면 false를 반환

```
@PostMapping("/emailCheck/certification")  
public boolean verifyCertificationNumber(@RequestBody MailCheckVO vo) {  
    return emailSenderService.isVerify(vo.getMemail(), vo.getCode());  
}
```

## Email send

```
public void createAuthCode() {  
    int length = 6;  
    StringBuilder authCode = new StringBuilder();  
    Random random = new Random();  
    for (int i = 0; i < length; i++) {  
        int type = random.nextInt(3);  
        switch (type) {  
            case 0:  
                authCode.append(random.nextInt(10));  
                break;  
            case 1:  
                authCode.append((char) (random.nextInt(26) + 65));  
                break;  
            case 2:  
                authCode.append((char) (random.nextInt(26) + 97));  
                break;  
        }  
    }  
    this.authCode = authCode.toString();  
}
```

숫자(0-9), 대문자(A-Z), 소문자(a-z)로 이루어진 임의의 6자리 인증코드 생성

```
public void sendEmail(String toEmail) {  
    createAuthCode();  
    MimeMessage message = mailSender.createMimeMessage();  
    try {  
        MimeMessageHelper helper = new MimeMessageHelper(message, true);  
        helper.setFrom("메일주소 기재");  
        helper.setTo(toEmail);  
        helper.setSubject("이메일 제목 기재");  
        String body = "발송될 Email의 내용 기재";  
        helper.setText(body, true);  
        mailSender.send(message);  
        certificationNumberDAO.saveCertificationNumber(toEmail, authCode);  
    } catch (MessagingException e) {  
        throw new RuntimeException(e);  
    }  
}
```

createAuthCode를 호출하여 새로운 인증 코드를 생성  
MimeMessage 객체를 생성  
MimeMessageHelper로 이메일 발신자, 수신자, 제목, 본문을 설정,  
mailSender를 통해 이메일을 전송

# **Admin**

**- JPA**

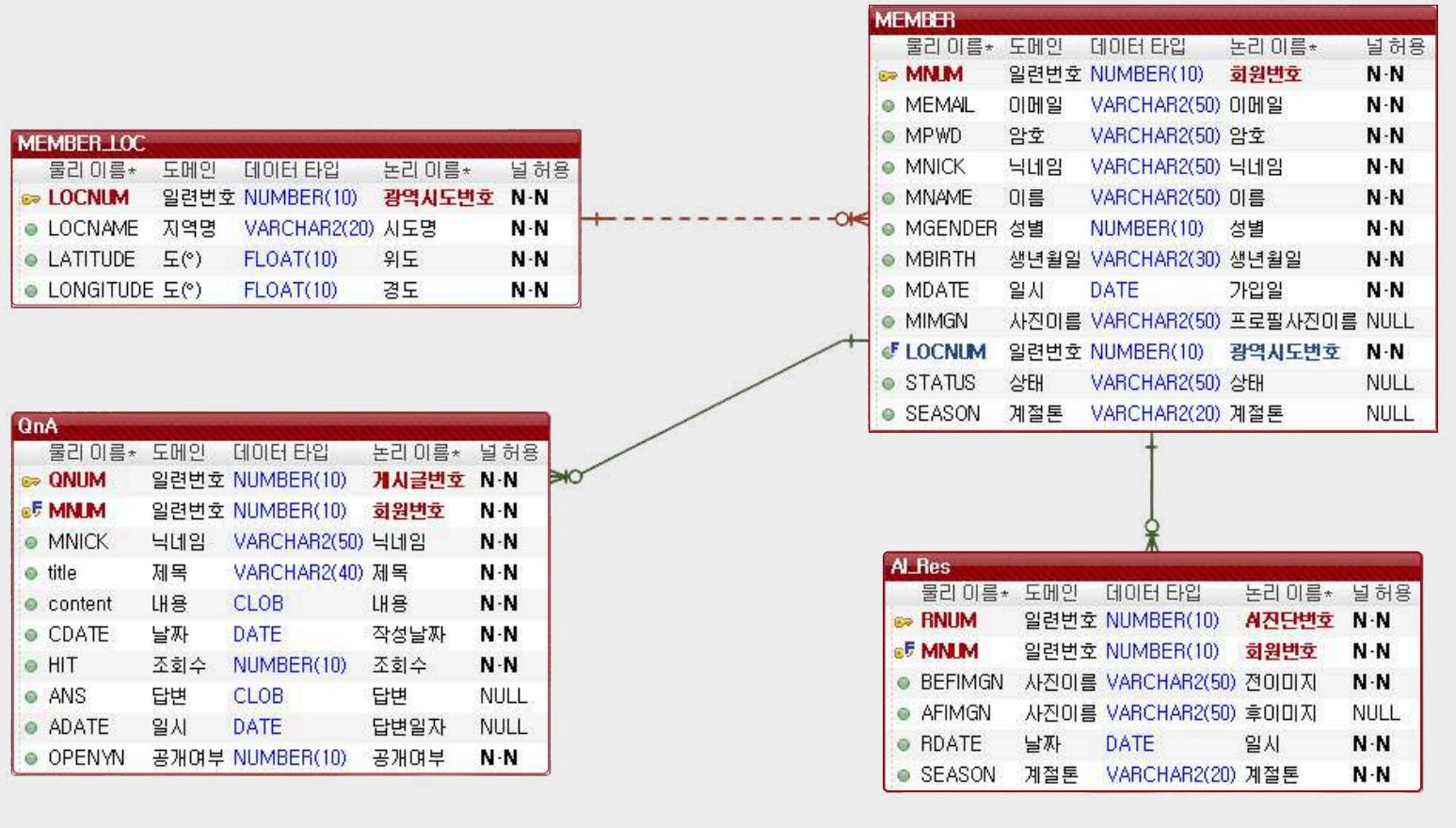


방현



# 테이블 생성

DB 구조



MEMBER Entity

```

@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "seq_member")
@SequenceGenerator(name = "seq_member", sequenceName = "seq_member", allocationSize = 1)
private Long mnum; // 회원번호

@Column(length = 50, nullable = false, unique = true)
private String memail; // 이메일(id)

@Column(length = 50, nullable = false)
private String mpwd; // 비밀번호

// ... 생략

@Column(length = 50, nullable = true)
private String status; // 회원상태

@OneToMany(mappedBy = "member", cascade = CascadeType.ALL, orphanRemoval = true, fetch =
FetchType.LAZY)
@JsonIgnore
private List<QnA> qnaList = new ArrayList<>();

@JsonIgnore
@OneToMany(mappedBy = "member", cascade = CascadeType.ALL, orphanRemoval = true, fetch =
FetchType.LAZY)
private List<AI_RES> aiResList = new ArrayList<>();

```

- JPA는 스프링 서버를 실행하면 스프링에서 설정한 Entity로 테이블 생성 및 설정되기 때문에 DB 구조와 동일하게 설정
- 다른 테이블 또한 동일하게 생성

## 관리자 (JPA) - QnA



- 현재 문의 상황을 알기 위해서 QnA Count 쿼리를 작성
- Front로 전달할 데이터 형태와 맞게 DTO 준비

## 관리자 (JPA) - QnA

### SQL PROCEDURE

```
CREATE OR REPLACE PROCEDURE NGG.GETQNAREPORT (
    searchValue IN VARCHAR2,
    searchType IN NUMBER,
    qna_cursor OUT SYS_REFCURSOR
) AS
BEGIN
    IF searchType = 1 THEN
        OPEN qna_cursor FOR
        SELECT q.qnum, q.mnick, q.title, q.hit,
               CASE WHEN q.ans IS NULL THEN '문의 대기' ELSE '문의 완료' END AS
               status,q.cdate
        FROM QnA q WHERE q.title LIKE '%' || searchValue || '%';
    ELSIF searchType = 2 THEN
        OPEN qna_cursor FOR
        SELECT q.qnum, q.mnick, q.title, q.hit,
               CASE WHEN q.ans IS NULL THEN '문의 대기' ELSE '문의 완료' END AS
               status,q.cdate
        FROM QnA q WHERE q.mnick LIKE '%' || searchValue || '%';
    ELSIF searchType = 3 THEN
        OPEN qna_cursor FOR
        SELECT q.qnum, q.mnick, q.title, q.hit,
               CASE WHEN q.ans IS NULL THEN '문의 대기' ELSE '문의 완료' END AS
               status,q.cdate
        FROM QnA q WHERE q.content LIKE '%' || searchValue || '%';
    ELSE
        OPEN qna_cursor FOR
        SELECT q.qnum, q.mnick, q.title, q.hit,
               CASE WHEN q.ans IS NULL THEN '문의 대기' ELSE '문의 완료' END AS
               status,q.cdate
        FROM QnA q;
    END IF;
END;
```

INPUT 데이터

### Front

QnA		오늘 문의 상황	완료 문의	비완료 문의	
번호	제목	닉네임	등록일	조회수	상태
39	회원님의 QnA	회원	2024-05-07	582	문의 완료
69	회원님의 QnA	회원	2024-05-22	461	문의 완료
78	회원님의 QnA	회원	2024-04-22	1166	문의 완료
122	회원님의 QnA	회원	2024-04-24	783	문의 완료
128	회원님의 QnA	회원	2024-05-04	531	문의 완료
141	회원님의 QnA	회원	2024-07-10	450	문의 완료
160	회원님의 QnA	회원	2024-05-12	908	문의 대기
170	회원님의 QnA	회원	2024-05-19	172	문의 완료
176	회원님의 QnA	회원	2024-07-08	566	문의 대기
196	회원님의 QnA	회원	2024-05-25	1169	문의 완료

- 검색어에 따른 데이터값을 위해 프로시저를 생성
- 검색 탑입과 검색 내용을 입력 파라미터(INPUT) 설정

### QnAStatusDTO

```
public class QnAStatusDTO {  
    // list에서 받아올 데이터형태  
    private Integer qnum;  
    private String mnick;  
    private String title;  
    private Integer hit;  
  
    // '문의 대기' 또는 '문의 완료'  
    private String status;  
  
    private Date cdate;  
}
```

### QnA Entity

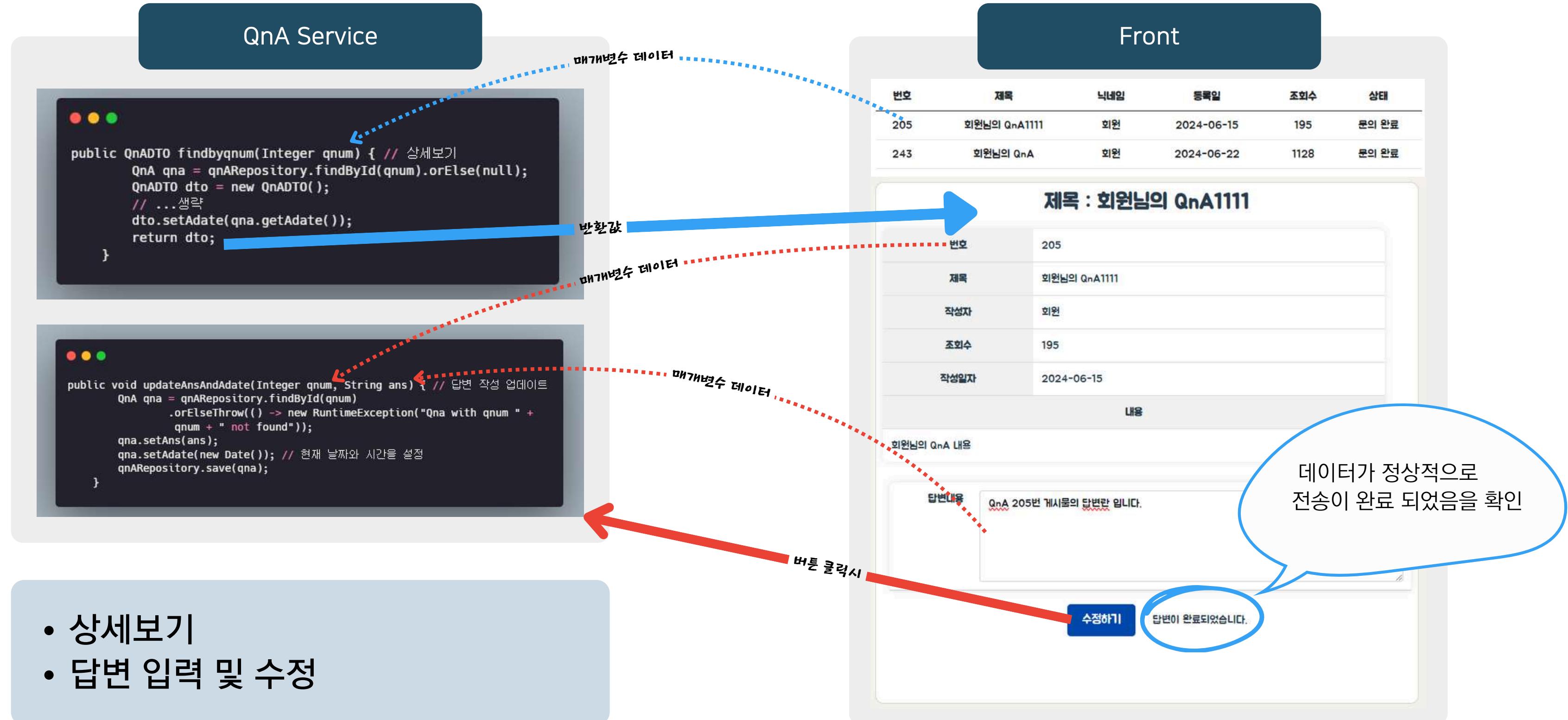
```
@Entity  
@SqlResultSetMapping(name = "QnAStatusDTOMapping", classes = { // 검색 프로시저  
    @ConstructorResult(targetClass = QnAStatusDTO.class, columns = {  
        @ColumnResult(name = "qnum", type = Integer.class),  
        @ColumnResult(name = "mnick", type = String.class),  
        @ColumnResult(name = "title", type = String.class),  
        @ColumnResult(name = "hit", type = Integer.class),  
        @ColumnResult(name = "status", type = String.class),  
        @ColumnResult(name = "cdate", type = Date.class)  
    })  })  
public class QnA {  
    // 기존 Entity테이블  
}
```

- **프로시저를 호출하기 위해서 SqlResultSetMapping를 지정**
- **Result를 저장하기 위한 QnAStatusDTO를 준비**



- createStoredProcedureQuery
  - 실행할 테이블 및 프로시저 설정
- registerStoredProcedureParameter
  - 프로시저에 입력할 파라미터 및 OUTPUT 커서 설정
- setParameter
  - 입력값 설정
- query.execute()
  - 저장 프로시저 실행
- query.getResultList()
  - 프로시저 실행결과 반환

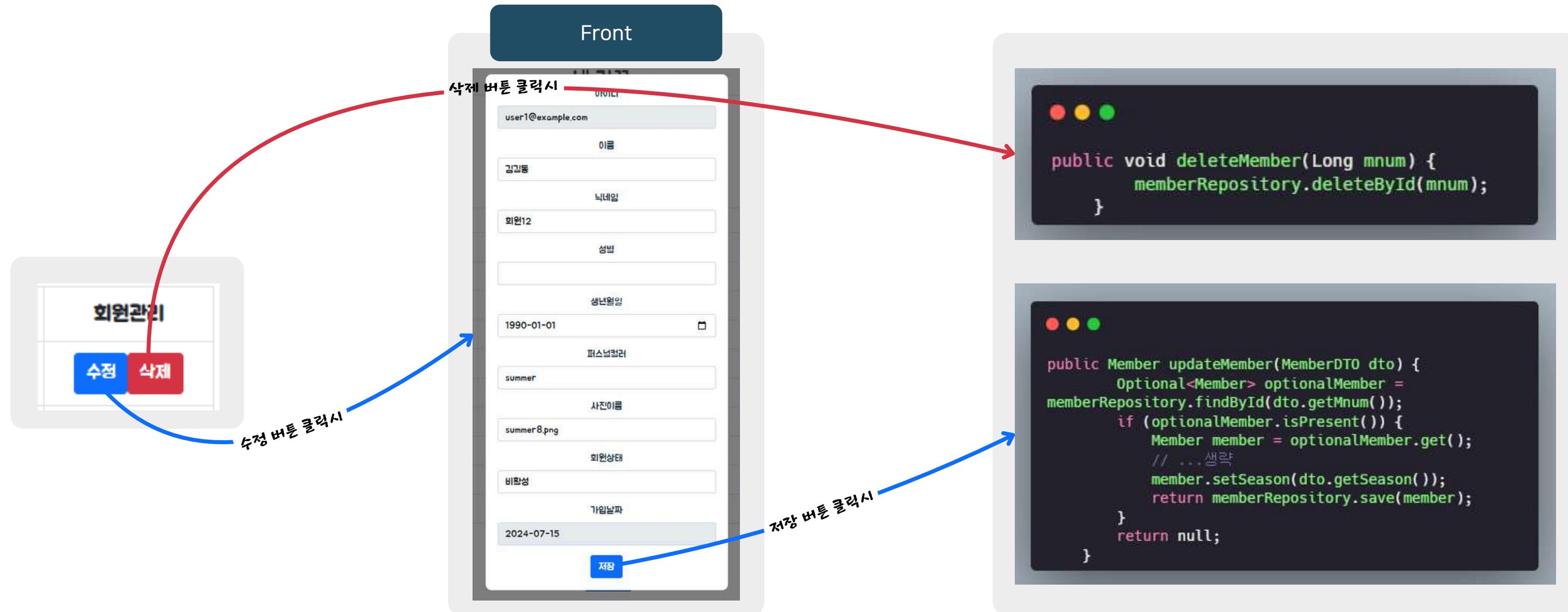
## 관리자 (JPA) - QnA



## 관리자 (JPA) - 회원관리



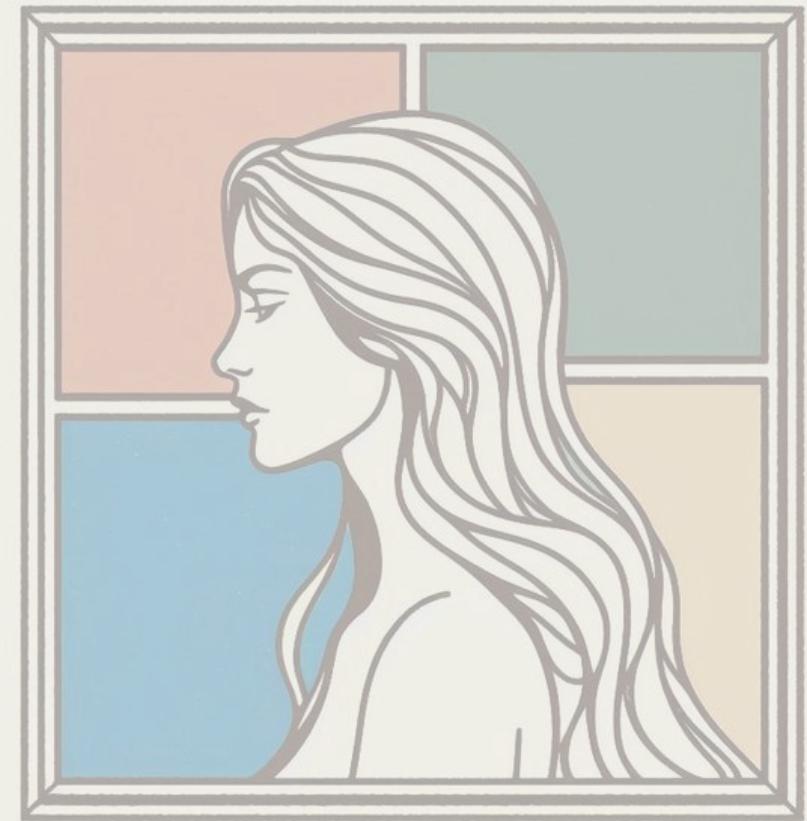
- QnA와 동일하게 검색 프로시저 생성



- 회원 수정 및 삭제

# Mypage

## - Mybatis



# 마이 페이지(컨트롤러)

The screenshot displays a user profile page with various sections and navigation links, each linked to specific Java code snippets.

**Profile Picture:** A circular profile picture of a person with long dark hair.

**회원ID:** 회원12

**나의 기록:** This link is highlighted with a red oval and an arrow points from it to the `@GetMapping("/myhistory")` code snippet.

**프로필 변경:**

**로그아웃:**

**회원탈퇴:** This link is highlighted with a red oval and an arrow points from it to the `@GetMapping("/delAccount")` code snippet.

**마이 페이지:** This link is highlighted with a red oval and an arrow points from it to the `@GetMapping("/detail")` code snippet.

**User Profile Details:**

- 김길동 | 김길동
- user1@example.com | 이메일
- \*\*\*\*\* | 비밀번호
- 0 | 성별
- 1990년 1월 1일 | 생년월일
- 경상도 | 주소
- spring | 키워드
- 2024년 7월 15일 | 등록일

**Buttons:**

- 수정 (Edit) button

**Java Code Snippets:**

- `@GetMapping("/detail")`  
public MemberVO detail(@RequestParam("mnum") int num)
- `@PostMapping("/seasonupdate")`  
public void seasonupdate(@RequestBody MemberVO vo)
- `@GetMapping("/myhistory")`  
public List<AIResVO> myhistory(@RequestParam("mnum") int mnum)
- `@GetMapping("/delAccount")`  
public void delAccount(@RequestParam int mnum)

# 마이 페이지(back)

## 마이 페이지

```
public MemberVO mydetail(int mnum); // 마이페이지  
  
public MemberVO detail(int mnum) {  
    return memberDao.mydetail(mnum);  
}  
  
<select id="mydetail" resultType="mvo" parameterType="int">  
    select * from member where mnum = #{mnum}  
</select>
```

## 나의 기록

```
public List<AIResVO> myhistory (int mnum); // 마이 히스토리  
  
public List<AIResVO> myhistory(int mnum){  
    return memberDao.myhistory(mnum);  
}  
  
<select id="myhistory" resultType="aivo" parameterType="int">  
    select * from ai_res where mnum = #{mnum} order by rdate DESC  
</select>
```

## 컬러 업데이트

```
public void seasonupdate(MemberVO vo); //컬러 업데이트  
  
public void seasonupdate(MemberVO vo) {  
    memberDao.seasonupdate(vo);  
}  
  
<update id="seasonupdate" parameterType="mvo">  
    update member set season=#{season} where mnum=#{mnum}  
</update>
```

## 회원탈퇴

```
public void delAccount(int mnum); //회원탈퇴  
  
public void delAccount(int mnum){  
    memberDao.delAccount(mnum);  
}  
  
<delete id="delAccount" parameterType="int">  
    delete from member where mnum = #{mnum}  
</delete>
```

# 정보 수정

**내 정보 수정하기**

이름	김길동
닉네임	회원12
비밀번호	.....
이메일	user1@example.com
성별	여
생일	1990-01-01
지역	경상도

**변경하기**

```
@GetMapping("/myidchk/{mnick}")
public int myidchk(@PathVariable("mnick") String mnick)
```

## 닉네임 중복체크

```
public int myidcheck(String mnick); // 닉네임 중복체크

public int myidchk(String mnick) {
    return memberDao.myidcheck(mnick);
}
```

```
<select id="myidcheck" parameterType="String" resultType="int">
    select count(*) cnt from member where mnick=#{mnick}
</select>
```

```
@PostMapping("/myupdate")
public void myupdate(MemberVO vo)
```

## 정보 수정

```
public void myupdate(MemberVO vo); // 내 정보 수정하기

public void myupdate(MemberVO vo) {
    memberDao.myupdate(vo);
}
```

```
<update id="myupdate" parameterType="mvo">
    update member set mnick=#{mnick}, mpwd=#{mpwd}, locnum=#{locnum} where mnum=#{mnum}
</update>
```

# 정보 수정

## locnum 맵핑

```
locnumMapping: {  
    1: "서울특별시",  
    2: "경기도",  
    3: "강원도",  
    4: "충청도",  
    5: "전라도",  
    6: "경상도",  
    7: "제주도",  
    default: "기타",  
}  
  
getLocationName(locnum) {  
    return this.locnumMapping[locnum] || this.locnumMapping.default;  
}
```

## Date형식 일자

```
changeDate(dateStr){  
    const [day, month, year] = dateStr.split('-');  
    if (year > 24){  
        return `19${year}년 ${month} ${day}일`;  
    }  
    else{  
        return `20${year}년 ${month} ${day}일`;  
    }  
}  
  
formattedDate() {  
    if (!this.detail.mbirth) return '';  
    return this.detail.mbirth.split('T')[0];  
}
```

# 프로필 사진 변경



```
@PostMapping("/myprofileupdate")
public ResponseEntity<?> myprofileupdate(@RequestParam("file") MultipartFile mf,
MemberVO vo, @RequestParam("mnum") int mnum, HttpServletRequest request) {
    if (mf == null || mf.isEmpty())
```

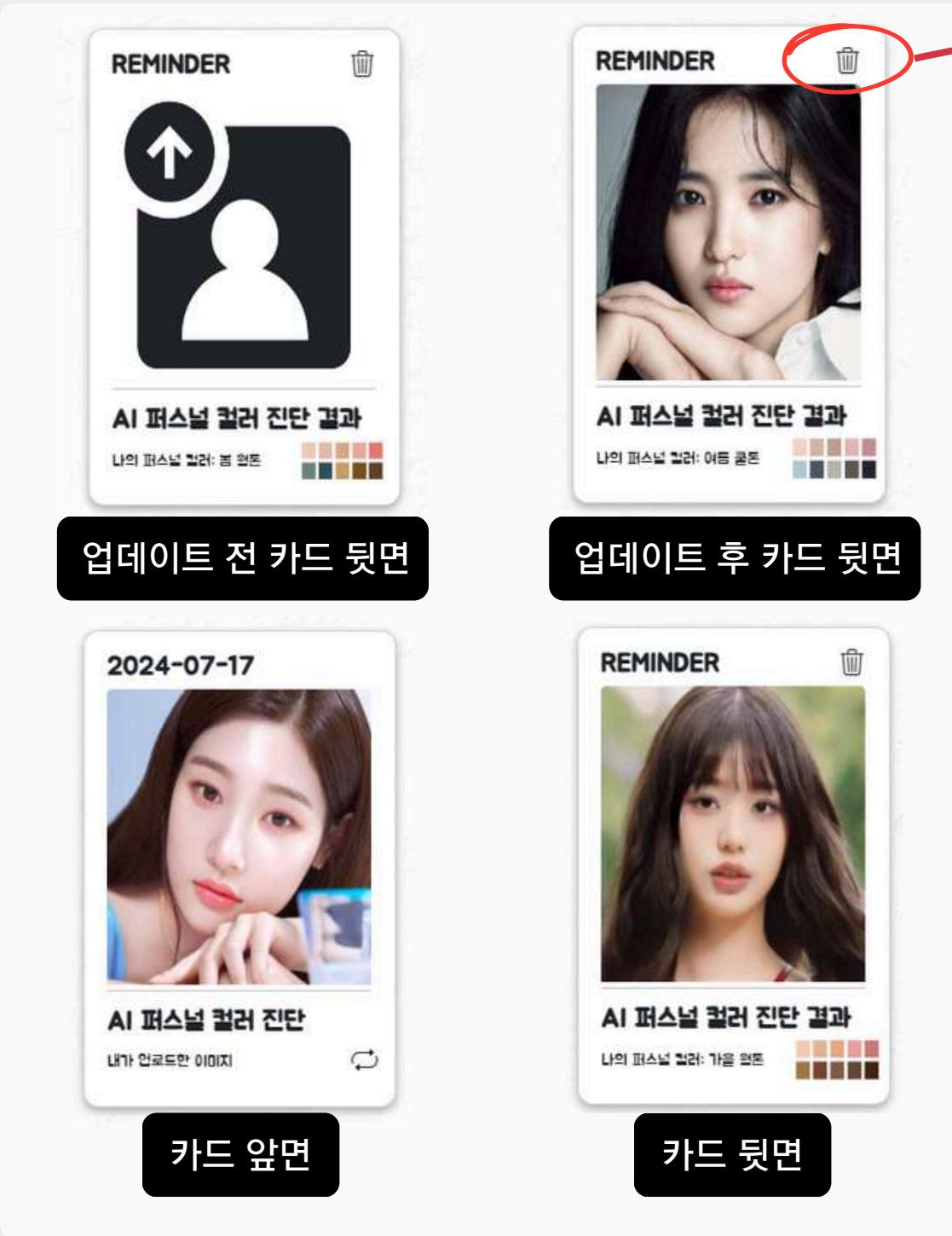
프로필 사진 변경

```
public void myprofileupdate(MemberVO vo); // 프로필 사진 변경
```

```
public void myprofileupdate(MemberVO vo) {
    memberDao.myprofileupdate(vo);
}
```

```
<update id="myprofileupdate" parameterType="mvo">
    update member set mimgn = #{mimgn} where mnum=#{mnum}
</update>
```

# 나의 기록



```
@GetMapping("/delHistory")
public void delHistory(@RequestParam int rnum)
```

나의 기록 삭제

```
public void delHistory(int rnum); //나의 기록 삭제
```

```
public void delHistory(int rnum){
    memberDao.delHistory(rnum);
}
```

```
<delete id="delHistory" parameterType="int">
    delete from ai_res where rnum = #{rnum}
</delete>
```

# 나의 기록

2024-07-17



AI 퍼스널 컬러 진단

내가 업로드한 이미지

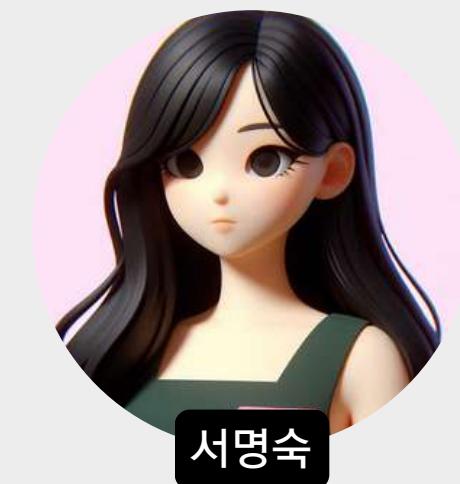


카드 플립

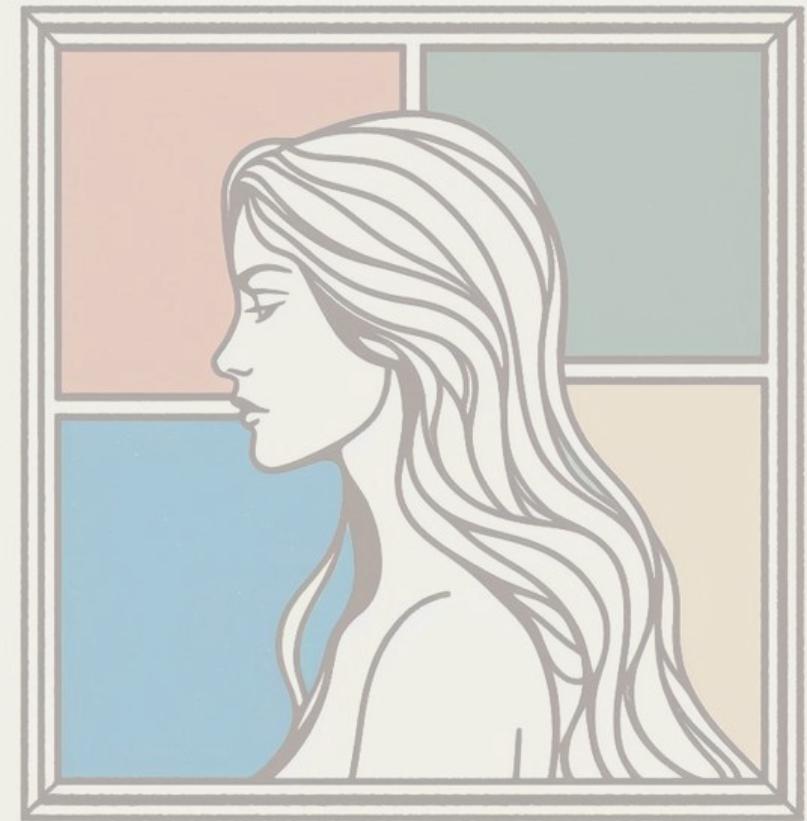
```
flipCard(index) {
    const actualIndex = (this.currentPage - 1) * this.itemsPerPage + index;
    this.cards[actualIndex].isFlipped = !this.cards[actualIndex].isFlipped;
    this.rnum = this.cards[actualIndex].rnum;
    this.befimgn = this.cards[actualIndex].befimgn;
}
```

# Q&A

## -Mybatis



서명숙



## QNA 게시판

The screenshot shows a Q&A board interface with the following elements:

- Header:** "QNA 게시판" (QNA Board) centered at the top.
- Search Bar:** A search bar with placeholder text "Search.." and two "검색" (Search) buttons (one white, one black).
- Table Headers:** "번호" (Number), "제목" (Title), "작성자" (Writer), "조회수" (View Count), "작성일자" (Creation Date), and "답변일자" (Response Date).
- Data Rows:** Five rows of Q&A items, each with a link labeled "회원님의 QnA".

번호	제목	작성자	조회수	작성일자	답변일자
279	<a href="#">회원님의 QnA</a>	회원	642	2024.04.21	2024.04.21
271	<a href="#">회원님의 QnA</a>	회원	792	2024.04.24	2024.04.24
243	<a href="#">회원님의 QnA</a>	회원	1128	2024.06.22	2024.06.22
205	<a href="#">회원님의 QnA1111</a>	회원	194	2024.06.15	
196	<a href="#">회원님의 QnA</a>	회원	1169	2024.05.25	2024.07.16
- Pagination:** A navigation bar with "Previous" and "Next" buttons, and page numbers 1, 2, 3, 4. The number 1 is highlighted in blue.
- Bottom Button:** A blue button labeled "글쓰기" (Write Article).

### QnA 게시판 목록 기능

#### 1 검색 기능

게시판 목록에서 제목, 작성자, 내용을 기준으로 검색할 수 있는 기능을 제공하여 사용자의 접근성을 높임

#### 2 상세페이지 조회 및 관리

각 게시물의 상세 정보를 제공하며, 조회, 수정, 삭제 등의 작업을 사용자가 수행할 수 있도록 함

#### 3 페이징 처리

게시판에서 페이지 간의 전환을 가능하게 하여 이전 페이지, 다음 페이지, 특정 페이지로의 이동을 지원

# QNA 게시판

번호	제목	작성자	조회수	작성일자	답변일자
361	<a href="#">QnA 게시판 이용방법 문의</a>	회원12	0	2024.07.17	

## 1 게시판 목록 조회

- 검색 조건 입력 후, 검색 버튼을 클릭하여 QnA 게시판 목록을 조회
- 스프링 부트와 CORS통신

```
submitForm() {
    const formData = new FormData();
    formData.append('searchType', this.searchType);
    formData.append('searchValue', this.searchValue);
    formData.append('cPage', this.cPage);

    axios.post(`${process.env.VUE_APP_BACK_END_URL}/qna/qnaList`, formData)
        .then((resp) => {
            console.log('after /qna/qnaList')
            console.log(resp.data)
            this.listData = resp.data
        })
}
```

```
public class QnaController {

    @Autowired
    private QnaService qnaService;

    @RequestMapping("/qnaList")
    public List<QnaVO> qnaList(QnaVO vo, Model model,
        @RequestParam Map<String, String> paramMap) {
        System.out.println("searchType:" + paramMap.get("searchType"));
        System.out.println("searchValue:" + paramMap.get("searchValue"));
        return qnaService.qnaList(paramMap);
    }
}
```

# QNA 게시판

```
● ● ●  
QnA 게시물의 식별자(ID)  
  
goToDetail(qnum) {  
  this.$router.push(`/QNAdetail?qnum=${qnum}`);  
},  
  
totalPages() {  
  return Math.ceil(this.listData.length / this.itemsPerPage);  
},  
paginatedList() { // 여기 이름을 변경  
  const start = (this.currentPage - 1) * this.itemsPerPage;  
  const end = start + this.itemsPerPage;  
  return this.listData.slice(start, end); // 실제 데이터를 slice  
},  
pageNumbers() {  
  const startPage = Math.floor((this.currentPage - 1) / this.pageBlockSize) * this.pageBlockSize +  
  const endPage = Math.min(startPage + this.pageBlockSize - 1, this.totalPages);  
  const pages = [];  
  for (let i = startPage; i <= endPage; i++) {  
    pages.push(i);  
  }  
  return pages;  
},  
  
changePage(page) {  
  if (page > 0 && page <= this.totalPages) {  
    this.currentPage = page;  
  }  
},
```

## 2 게시판 상세페이지 조회

- 특정 QnA 게시물의 상세 페이지로 이동
- 게시물의 세부 내용을 확인하고 필요한 경우 내용을 수정하거나 삭제 처리 시 사용

## 3 페이징 처리

- 페이지 번호를 클릭하여 해당 페이지로 이동

# QNA 게시판 글쓰기

**QNA 게시판 글쓰기**

제목	QnA 게시판 이용방법 문의
작성자	회원12
내용	내가 꿈 AI 전단 방법 문의드립니다.
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개
<button>등록</button>	

```
fetchData() {
    axios.get(`process.env.VUE_APP_BACK_END_URL}/members/detail?mnum=1`)
        .then((resp) => {
            console.log('멤버 정보 가져오기 성공');
        })
}

submitForm() {
    this.formData.append("title", this.title);
    this.formData.append("content", this.content);
    this.formData.append("mnick", this.mnick);
    this.formData.append("openyn", this.openyn);
    this.formData.append("mnum", this.mnum);
    axios.post(`process.env.VUE_APP_BACK_END_URL}/qna/qnaWrite`, this.formData)
        .then((resp)=>{
            console.log(resp.data)
            this.list = resp.data
            alert("정상 등록")
            this.$router.push('/QNA')
        })
}
```

- 작성자의 상세 정보를 불러오는 기능을 구현
- 스프링 부트와 CORS 통신을 통한 데이터 저장
- 정상적으로 저장된 후 게시판 목록 페이지로 이동하여 입력된 내용을 확인

# QNA 게시판 상세보기

## 운영자 답변이 없을 경우

**QNA 게시판 상세보기**

번호	361
제목	QnA 게시판 이용방법 문의
작성자	회원12
조회수	24
작성일자	2024년 7월 17일 11시 11분
내용	내가 꿈 AI 진단 방법 문의드립니다. 내용 1차 수정
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개

[목록으로 돌아가기](#) [수정](#) [삭제](#)

```
<div class="button-group">
  <router-link to="/QNA" class="back-button">목록으로 돌아가기</router-link>
  <router-link v-if="board.ans === null" :to="`/QNAUpdate?qnum=${board.qnum}`" class="update-button">
    <span>수정</span>
  </router-link>
  <button type="button" class="delete-button" @click.prevent="deleteQna">삭제</button>
</div>

deleteQna() {
  if(!confirm("정말 삭제하시겠습니까?")){
    return;
  }
  axios.get(`${process.env.VUE_APP_BACK_END_URL}/qna/qnaDelete?qnum=${this.board.qnum}`)
    .then((resp) => {
      console.log(resp.data);
      alert("삭제 정상 완료");
      this.$router.push('/QNA');
    })
    .catch((err) => {
      console.error('에러 발생:', err);
    });
},
```

- 게시물 상세 조회 및 운영자의 답변이 없을 경우, 수정 버튼 활성화
- 수정 버튼 클릭시 해당 게시물의 수정 페이지로 이동
- 스프링 부트와 CORS통신하여 삭제 요청을 처리, 삭제 후 게시판 목록 페이지로 이동

# QNA 게시판 상세보기

## 운영자 답변이 있을 경우

QNA 게시판 상세보기

번호	279
제목	회원님의 QnA
작성자	회원
조회수	627
작성일자	2024년 4월 21일 0시 0분
내용	회원님의 QnA 내용
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개

[목록으로 돌아가기](#) [삭제](#)

작성자 문의에 답변

답변일자	2024년 4월 21일 0시 0분
답변내용	1000자만 대답입니다.

```
<!-- 운영자 답변 색션 -->
<div v-if="board.ans" class="admin-write">
  <h2><b>작성자 문의 답변</b></h2>
  <table class="admin-write-table">
    <tbody>
      <tr>
        <th><label for="adate" class="form-label">답변일자</label></th>
        <td>{{ formatDate(board.adate) }}</td>
      </tr>
      <tr>
        <th><label for="ans" class="form-label">답변내용</label></th>
        <td><textarea class="form-control" id="ans" rows="5" v-model="board.ans"></textarea></td>
      </tr>
    </tbody>
  </table>
</div>

<div class="button-group">
  <a href="/QNA" class="back-button">목록으로 돌아가기</a>
  <a v-if="board.ans === null" :to="`/QNAUpdate?qnum=${board.qnum}`" class="update-button">
    <span>수정</span>
  </a>
  <button type="button" class="delete-button" @click.prevent="deleteQna">삭제</button>
</div>
```

- 게시물 상세 조회 및 운영자의 답변이 있을 경우 답변 내용 조회
- 운영자의 답변이 있을 경우 수정 버튼 비활성화
- 스프링 부트와 CORS통신하여 삭제 요청을 처리,  
삭제 후 게시판 목록 페이지로 이동

## QNA 게시판 수정

**QNA 게시판 수정**

제목	QnA 게시판 이용방법 문의
작성자	회원12
내용	내가 끓 AI 진단 방법 문의드립니다. 내용 1차 수정
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개
<b>저장</b>	

```
fetchData(qnum) {
  axios.get(`${process.env.VUE_APP_BACK_END_URL}/qna/qnaDetail?qnum=${qnum}`)
    .then((resp) => {
      this.board = resp.data;
    })
    .catch((err) => {
      console.error('에러 발생', err);
    });
},
```

```
submitForm() {
  axios.post(`${process.env.VUE_APP_BACK_END_URL}/qna/qnaUpdate`, this.board)
    .then((resp) => {
      console.log("서버 응답:", resp.data);
      if (resp.status === 200) {
        alert("정상 수정");
        this.$router.push('/QNA');
      } else {
        alert("수정 실패");
        console.error("수정 실패:", resp.data);
      }
    });
},
```

- 게시물 수정 및 업데이트 내용을 스프링 부트에 CORS 통신을 통해 전송
- 수정 작업이 성공적으로 완료되면 QnA 게시판 목록 페이지로 이동



# 감사합니다!

- 프로젝트명 : AI 기반 이미지 처리 머신러닝&딥러닝 프로젝트  
- 마스크 착용 안면 인식 판별
- 프로젝트 기간 : 2024.06.28 ~ 2024.07.18
- 팀원 : 방현, 배선화, 복권일, 서명숙, 안은영, 최준희, 추성호