Team 1

Vision Project

GD Door Guard

창민 : 정우 : 지환 : 정호

CONTENTS

01

프로젝트 개요

01

프로젝트 개요

개발 취지 / 목표

1인 가구 증가

주거 침입 범죄 증가

여성 대상 범죄 증가

개발 담당

서창민 Pro

- 사물 감지
- 이미지 전송(DataBase)
- QT UI / WEB / DB 서버 구현
- 각 모듈 코드 병합

김지환 Pro

- 모자이크 처리
- None 마스킹 처리 값 분리 저장 (수사기관 요청 시, 제출용)

박정우 Pro

- 이미지 파일 생성 감지
- 인증 토큰 절차 자동화
- Kakao API 인증 절차 단순화

최정호 Pro

- Kakao API 인증 process 구현
- 이미지 반복 생성 시 버그 개선

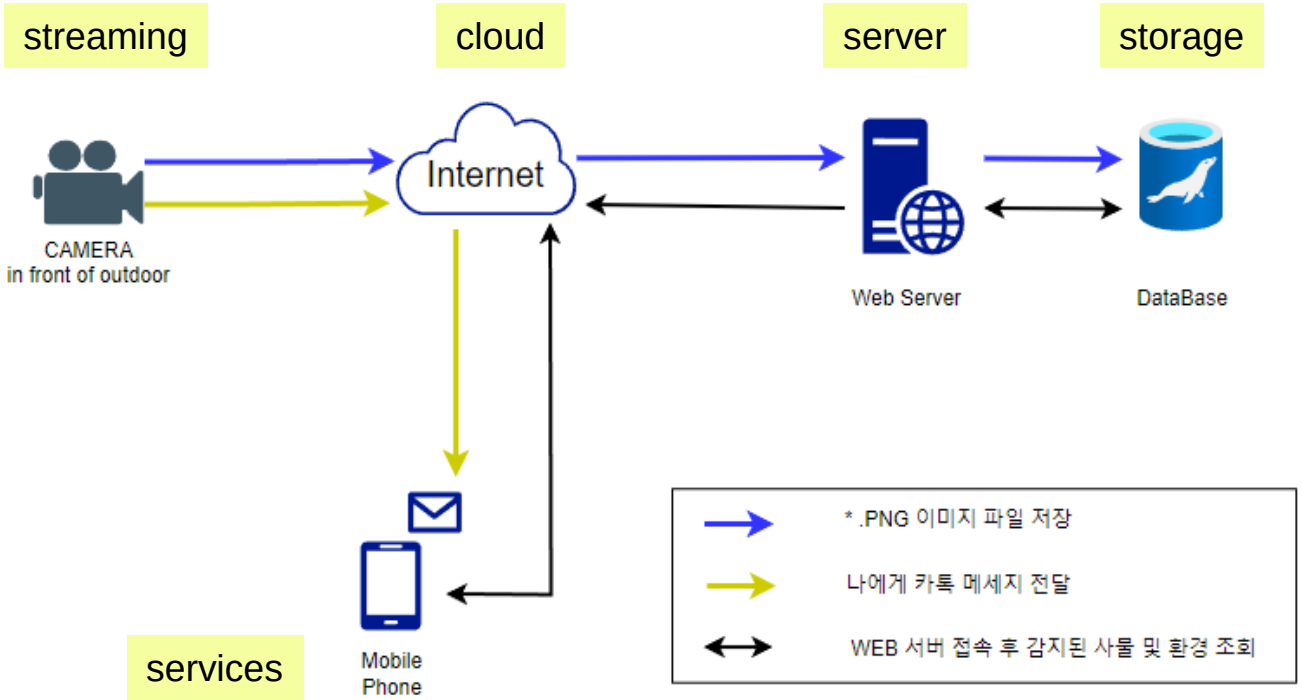
개발 일정(Work Breakdown Structure)

팀원	담당 내용	일정			
		09월 02일	09월 03일	09월 04일	09월 05일
서창민	WEB / DB 서버 구축				
	웹페이지 제작				
	Camera 사물 감지				
	이미지 저장 및 전송				
	QT UI 및 버튼 추가				
	코드 병합				
김지환	모자이크 처리				
	비 마스킹 처리 값 분리 저장				
박정우	이미지 파일 생성 감지				
	인증 토큰 절차 자동화				
	Kakao API 인증 절차 간소화				
최정호	Kakao API 인증 Process 구현				
	이미지 반복 생성 시, 버그 개선				

02

시스템 아키텍처

02 시스템 아키텍처



개발 환경

✓ 운영체제

✓ 개발환경/도구

✓ 개발언어

✓ 개발 제약사항

03

핵심 기능 소개

3-1 사물 감지
이미지 전송
WEB / DB 구축
3-2 마스킹 처리
3-3 신규 이미지 생성 감지
내게 카톡 전송

03

핵심 기능 소개

MediaPipe

비디오 데이터 등

- 실시간 비디오 스트림
- 얼굴 감지/처리 최적화
- 다양한 내장 솔루션

적은 리소스 사용

- Dlib,PyTorch 대비
- 모바일 장치 최적화

공식 문서 풍부한 예제

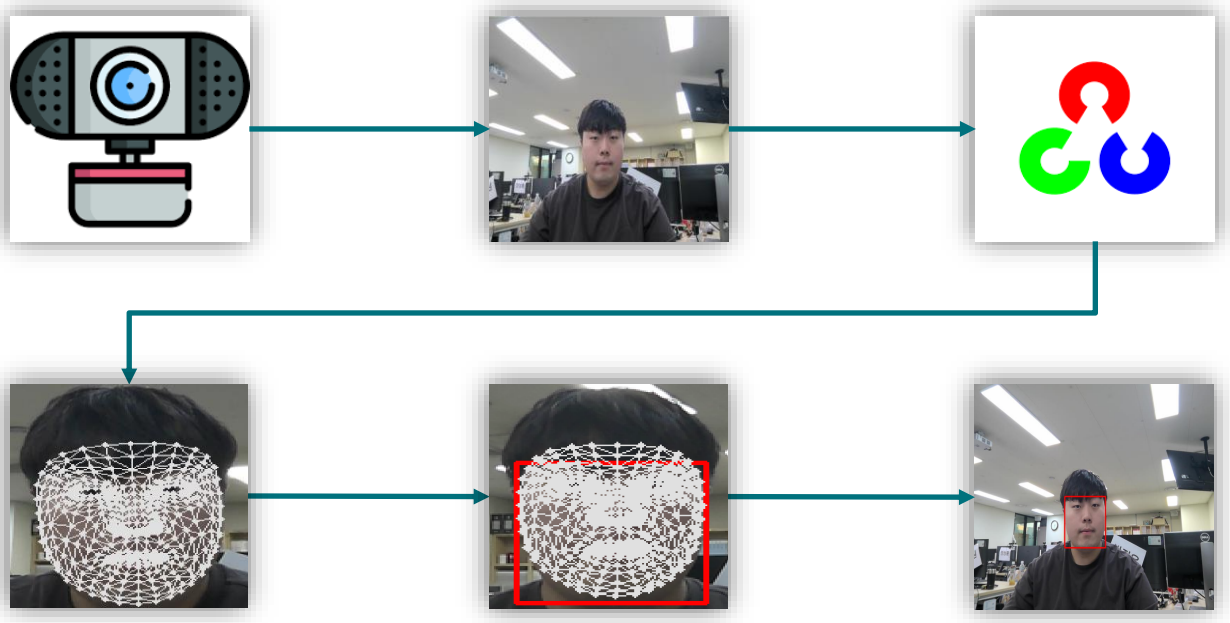
- 기능별 모듈 세분화
- **Colab** 테스트 환경 제공

03

핵심 기능 소개

코드 설명

- 순서도



얼굴 감지 코드

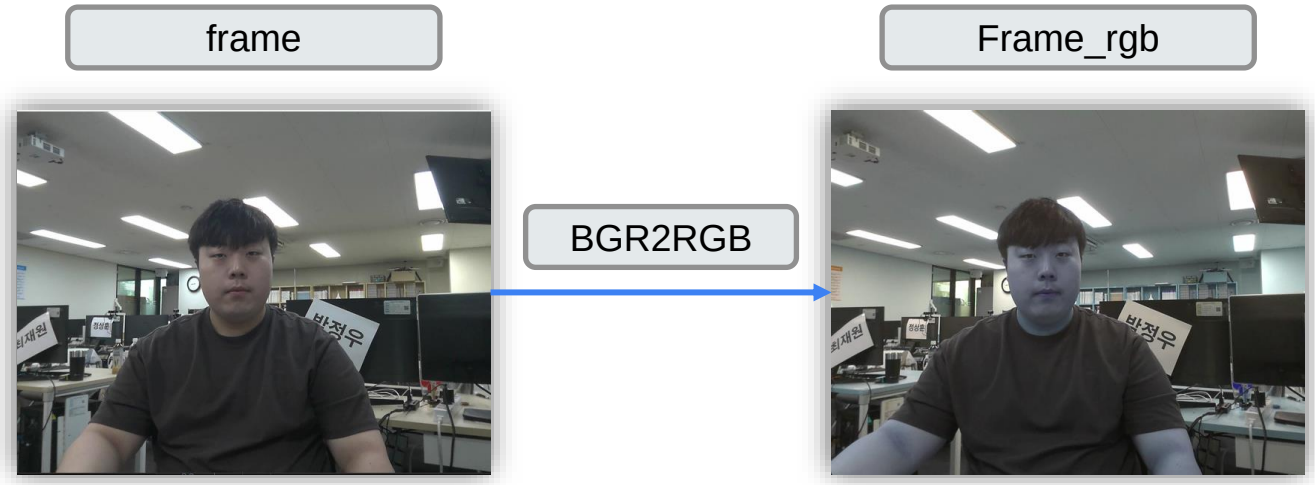
```
mp_face_detection = mp.solutions.face_detection
face_detection = mp_face_detection.FaceDetection(min_detection_confidence=0.2)

def detect_faces(frame):
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = face_detection.process(frame_rgb)
```

mp_face_detection	얼굴 감지 모듈 호출
face_detection	얼굴 감지 모델 초기화, 신뢰도 = 0.2
frame_rgb	BGR 이미지를 RGB로 변환
results	RGB 변환 프레임에서 얼굴 감지 결과 저장

03

핵심 기능 소개

얼굴 감지 코드



03

핵심 기능 소개

얼굴 감지 코드

```
if results.detections:
    ih, iw, _ = frame.shape
    for detection in results.detections:
        bboxC = detection.location_data.relative_bounding_box
        x, y, w, h = int(bboxC.xmin * iw), int(bboxC.ymin * ih),
        int(bboxC.width * iw), int(bboxC.height * ih)

        # 얼굴 주변에 빨간 사각형 그리기
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

return results.detections
```

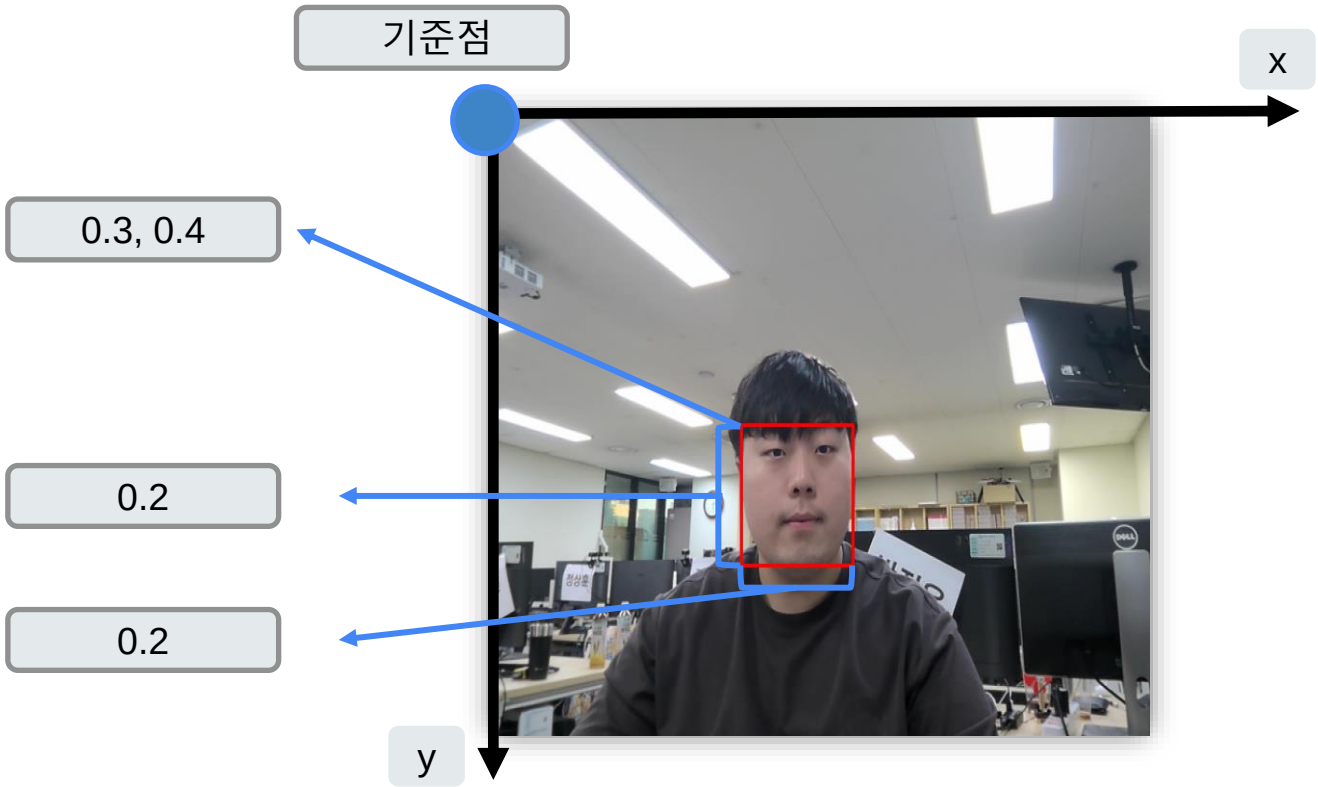
bboxC — 얼굴의 상대적 경계 상자 정보 저장 (0~1)

x, y, w, h — 상자의 좌표, 길이의 절대적 값 저장

03

핵심 기능 소개

얼굴 감지 코드

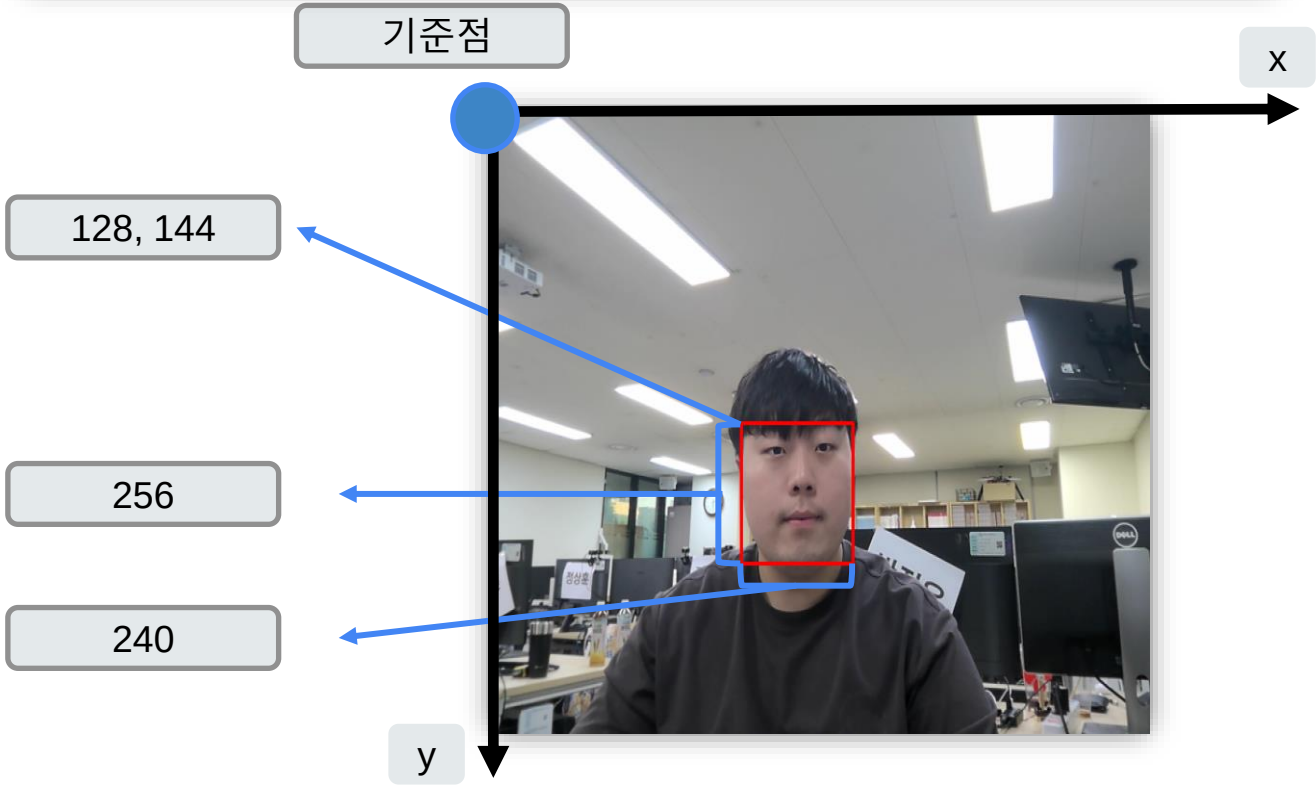


03

핵심 기능 소개

얼굴 감지 코드

```
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
```



03
핵심 기능 소개

이미지 저장

imwrite

이미지를 직접 파일로 저장

imencode

이미지를 메모리 버퍼에 저장

```
if last_saved_time is None or current_time - last_saved_time >= capture_interval:
    timestamp = datetime.now().strftime('%Y_%m_%d_%H_%M_%S')
    filename = f'{timestamp}.png'
    file_path = os.path.join(base_output_dir, filename)

    cv2.imwrite(file_path, frame)
    print(f'Successfully saved {filename} to local folder.')

    _, buffer = cv2.imencode('.png', frame)
    image_data = buffer.tobytes()
    save_image_to_db(filename, image_data)

    last_saved_time = current_time  # 마지막 저장 시간 업데이트
```

로컬에 바로 저장

인코딩 이미지를 바이트로 변환

데이터 DB로 전송

03

핵심 기능 소개

이미지 전송

capture_images DB 에 업로드시간, 파일 이름, 이미지 전송

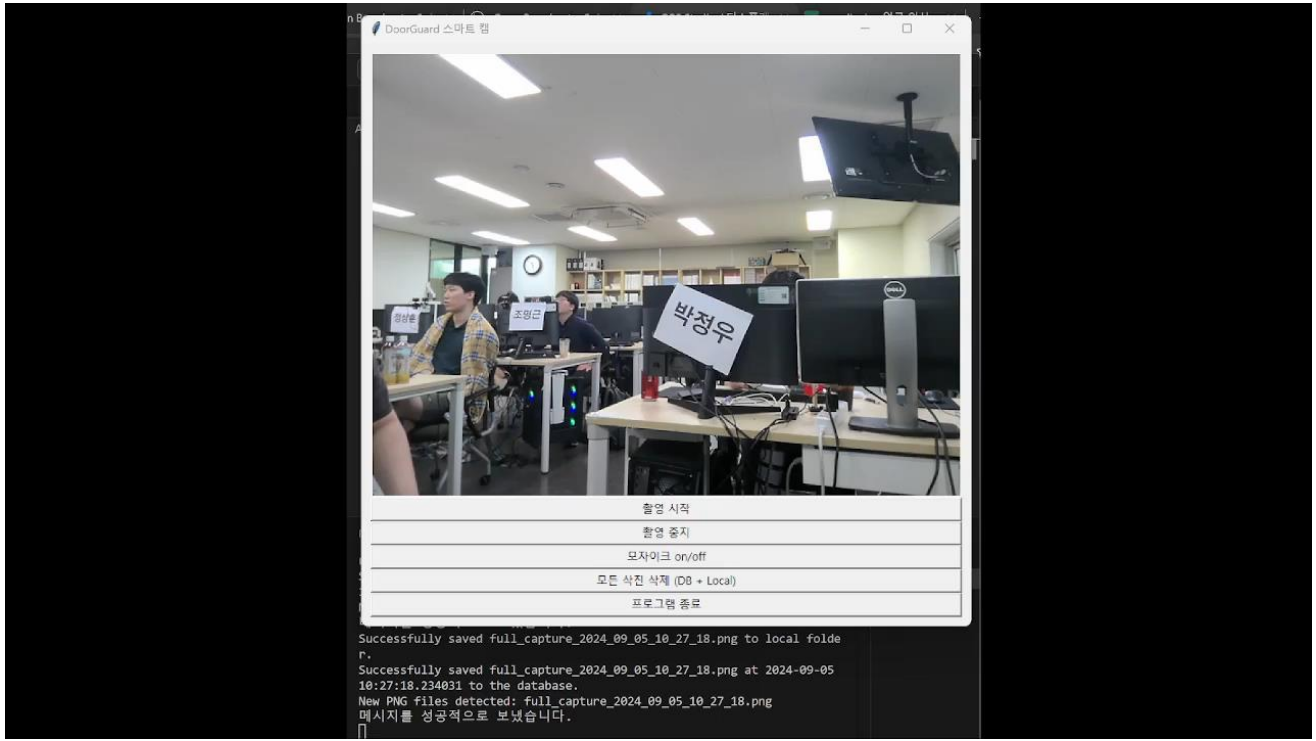
전송 결과를 저장 (commit)

```
def save_image_to_db(filename, image_data):
    """ 이미지 데이터베이스에 저장 """
    upload_time = datetime.now()
    try:
        cursor.execute("INSERT INTO captured_images (upload_time, filename, image
        conn.commit()
        print(f'Successfully saved {filename} at {upload_time} to the database.')
    except mysql.connector.Error as err:
        print(f'Failed to save image to database: {err}')
```

03

핵심 기능 소개

이미지 전송

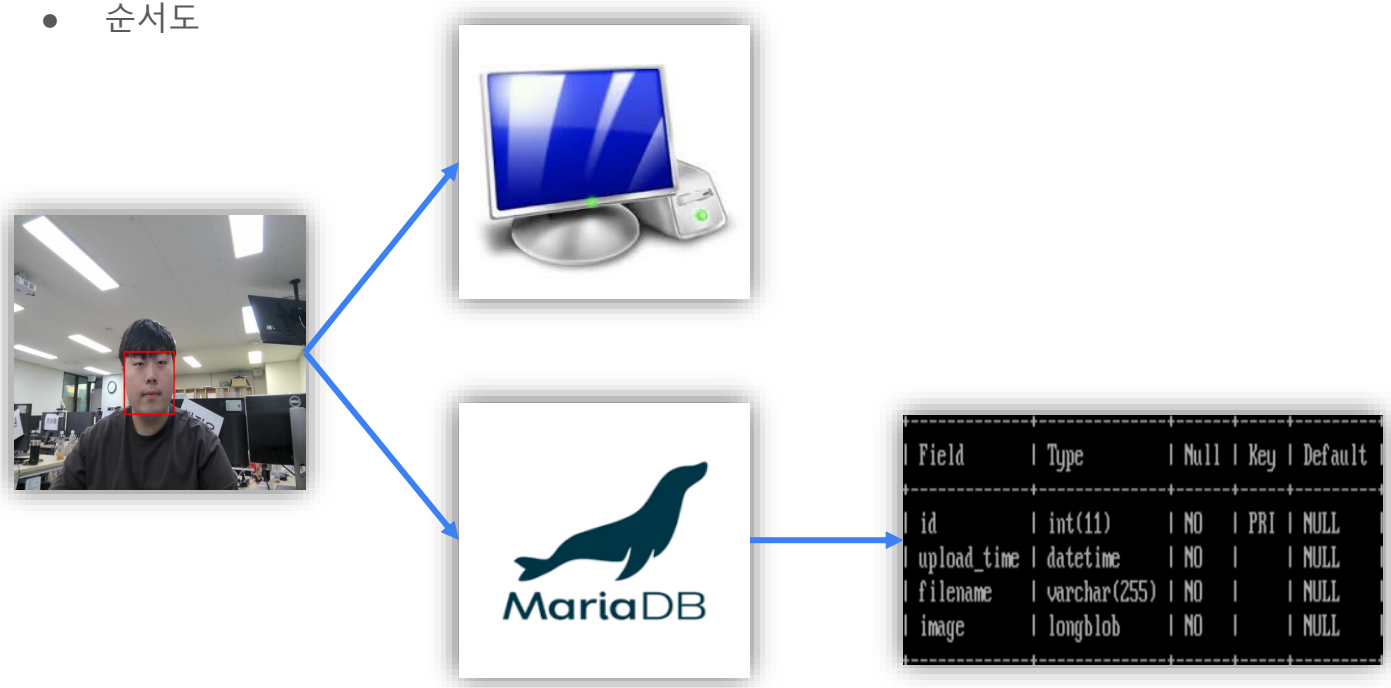


03

핵심 기능 소개

코드 설명

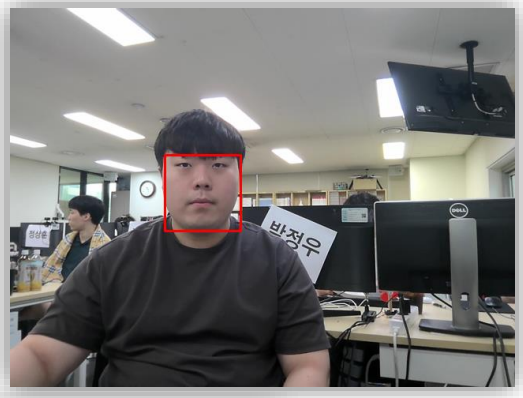
- 순서도

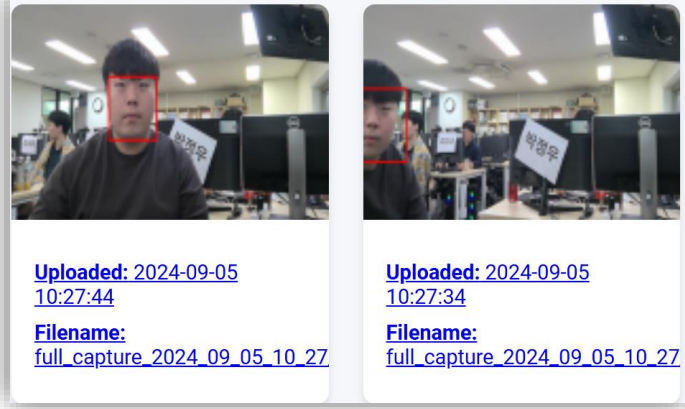


03

핵심 기능 소개

이미지 전송

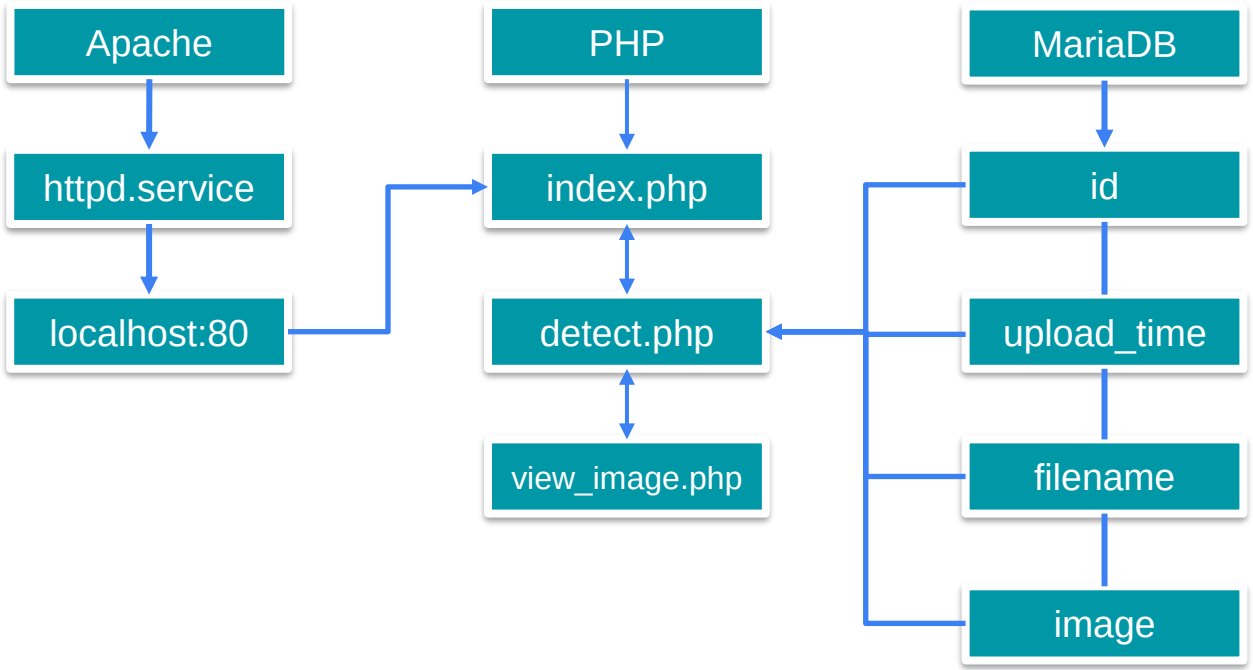




03

핵심 기능 소개

APM 서버



03

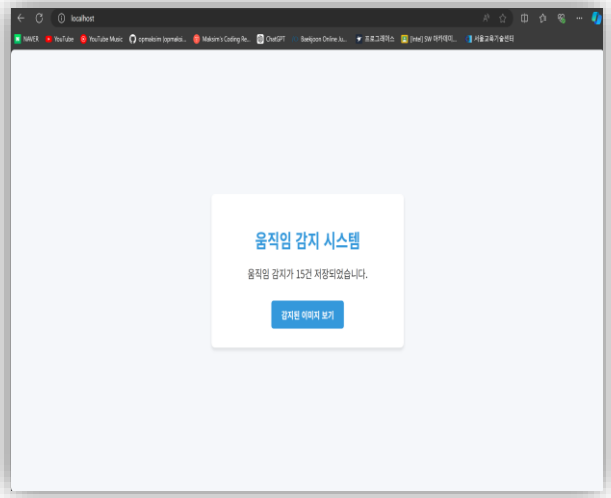
핵심 기능 소개

Index.php

- DB테이블에 저장된 데이터 값의 개수를 불러옴
- 개수에 따라 메인 페이지의 건수가 변동됨

```
// 저장된 이미지 수 조회
$sql = "SELECT COUNT(*) as count FROM captured_images";
$result = $conn->query($sql);
$row = $result->fetch_assoc();
$image_count = $row['count'];
```

```
<body>
    <div class="container">
        <h1>움직임 감지 시스템</h1>
        <p>움직임 감지가 <?php echo $image_count; ?>건 저장되었습니다.</p>
        <a href="detect.php" class="btn">감지된 이미지 보기</a>
    </div>
```



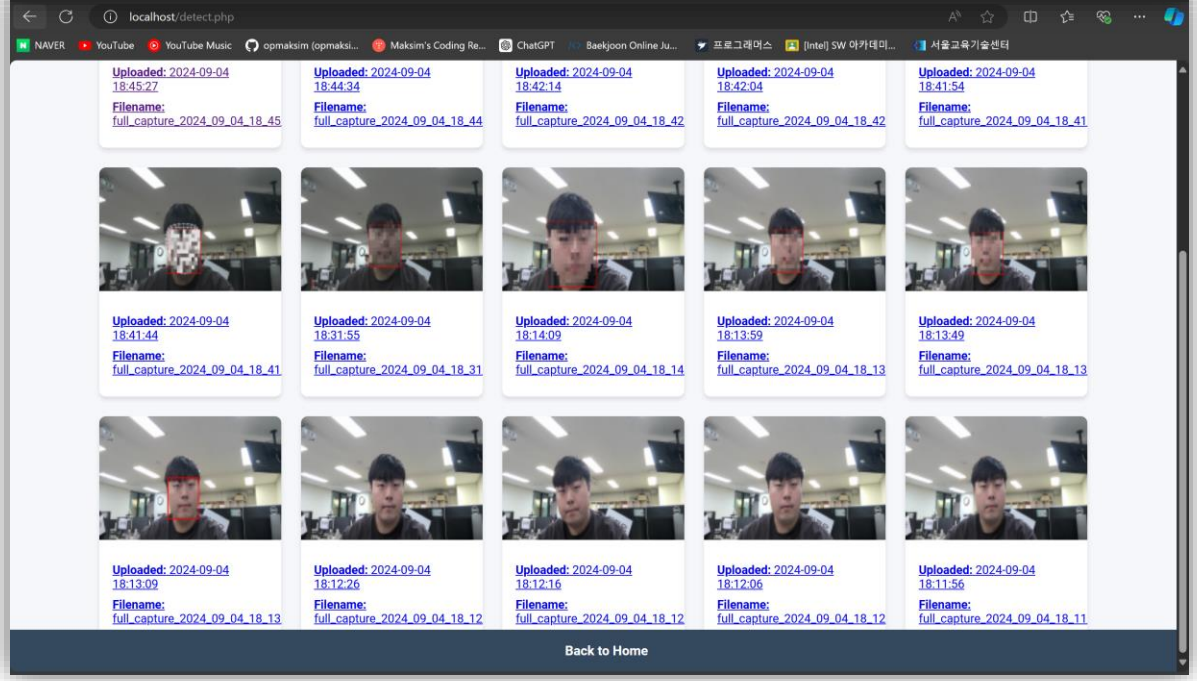
03

핵심 기능 소개

detect.php

- view_image.php 내용을 불러와 사진 보여줌
- DB에 있는 id, upload_time, filename만 불러옴

```
<a href="view_image.php?id=<?php echo $row['id']; ?>">
    <img src="view_image.php?id=<?php echo $row['id']; ?>&thumbnail=true" alt="Thumbnail" loading="lazy">
    <div class="info">
        <p><strong>Uploaded:</strong> <?php echo htmlspecialchars($row['upload_time']); ?></p>
        <p><strong>Filename:</strong> <?php echo htmlspecialchars($row['filename']); ?></p>
```



03

핵심 기능 소개

view_image.php

- 바이트 형태로 저장된 이미지 데이터를 인코딩하여 사진으로 보여줌
  □base64_encode….

```
<h1>View Image</h1>
<img src="<?php echo base64_encode($image_dat
<div class="info">
    <p><strong>Uploaded:</strong> <?php echo htmlspecialchars($uploa
    <p><strong>Filename:</strong> <?php echo htmlspecialchars($filenam
```

```
if ($row = $result->fetch_assoc()) {
    $upload_time = $row['upload_time'];
    $filename = $row['filename'];
    $image_data = $row['image'];
```

03

핵심 기능 소개

주요 코드

- 라이브러리 임포트, Mediapipe 초기화

```
import cv2
import mediapipe as mp
import time
import os
from datetime import datetime

# MediaPipe 초기화
mp_face_detection = mp.solutions.face_detection
face_detection = mp_face_detection.FaceDetection(min_detection_confidence=0.2)
```

cv2	OpenCV 라이브러리, 이미지 및 비디오 처리
mediapipe	머신러닝 기반의 얼굴 인식 기능을 제공
time	시간 관련 작업을 위한 라이브러리
os	파일 및 디렉토리 작업을 위한 라이브러리
datatime	현재 날짜와 시간을 처리하기위한 라이브러리

03

핵심 기능 소개

Mediapipe

- 얼굴 감지와 관련된 기능 제공 3rdParty Library
- 고급 머신러닝 모델로 얼굴을 정확히 감지 및 강인한 조명 화각에 얼굴 인식 우수
- 다양한 개발 환경에서 사용, 저 해상도 웹캠 환경에서 효율적으로 동작

03

핵심 기능 소개

주요 코드

- 모자이크 적용 함수

```python
def apply_mosaic(face_img, mosaic_size=10):
    """ 얼굴 영역에 모자이크 효과 적용 """
    (h, w) = face_img.shape[:2]
    small = cv2.resize(face_img, (w // mosaic_size, h // mosaic_size), interpolation=cv2.INTER_LINEAR)
    mosaic = cv2.resize(small, (w, h), interpolation=cv2.INTER_NEAREST)
    return mosaic
```

apply_mosaic — 얼굴 영역에 모자이크 효과를 적용하는 함수

cv2.resize — 이미지 크기를 조정하여 모자이크 효과 생성
먼저 축소한 후 확대하여 모자이크 효과 생성

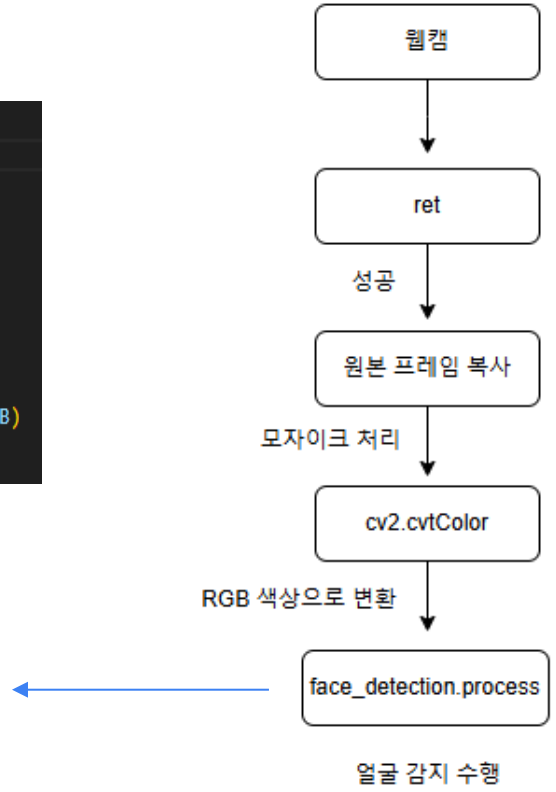
03

핵심 기능 소개

주요 코드

- 비디오 캡처 및 처리 루프

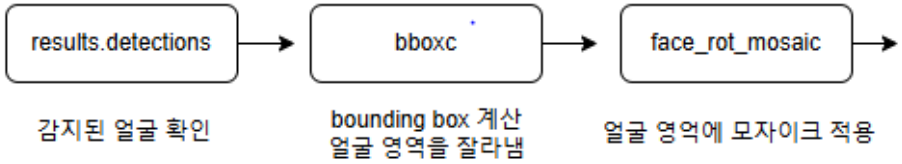
```
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # 모자이크 처리를 위한 복사본 생성
    frame_with_mosaic = frame.copy()

    # 이미지 전처리
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = face_detection.process(frame_rgb)
```



주요 코드

- 얼굴 감지 및 모자이크 적용

```
if results.detections:
    current_time = time.time()

    for detection in results.detections:
        bboxC = detection.location_data.relative_bounding_box
        ih, iw, _ = frame.shape
        x, y, w, h = int(bboxC.xmin * iw), int(bboxC.ymin * ih), int(bboxC.width * iw), int(bboxC.height * ih)
        face_roi = frame[y:y+h, x:x+w]

        if face_roi.size > 0:
            face_roi_mosaic = apply_mosaic(face_roi, mosaic_size=10)
            frame_with_mosaic[y:y+h, x:x+w] = face_roi_mosaic
```



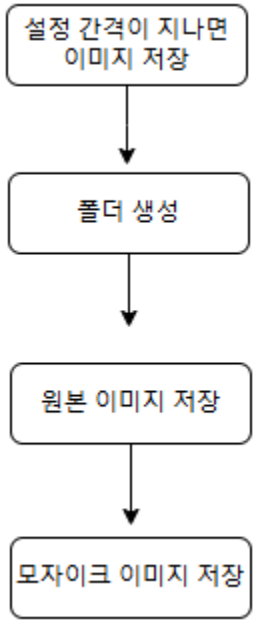
03

핵심 기능 소개

주요 코드

- 이미지 저장

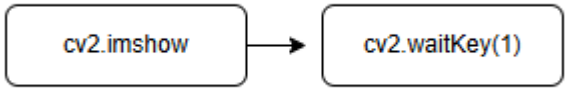
```python
# 사람이 처음 감지되었거나, 마지막 저장 후 10초가 지난 경우 이미지 저장
if current_time - last_saved_time >= capture_interval:
    timestamp = datetime.now().strftime('%Y_%m_%d_%H_%M_%S')
    folder_path = os.path.join(base_output_dir, timestamp)

    # 폴더가 없으면 생성
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    # 저장 경로 설정
    original_file_path = os.path.join(folder_path, f'original_capture_{timestamp}.png')
    mosaic_file_path = os.path.join(folder_path, f'mosaic_capture_{timestamp}.png')

    try:
        # 원본 이미지 저장
        cv2.imwrite(original_file_path, frame)
        print(f"Original image saved to {original_file_path}")

        # 모자이크가 적용된 이미지 저장
        cv2.imwrite(mosaic_file_path, frame_with_mosaic)
        print(f"Mosaic image saved to {mosaic_file_path}")

        last_saved_time = current_time  # 마지막 저장 시간 업데이트
    except Exception as e:
        print(f"Failed to save image: {e}")
```



주요 코드

- 비디오 출력 및 종료 조건

```
# 화면에 모자이크 처리된 비디오 표시
cv2.imshow('Mosaic Face', frame_with_mosaic)

# 종료 조건 (ESC 키를 누르면 종료)
if cv2.waitKey(1) & 0xFF == 27:
    break
```



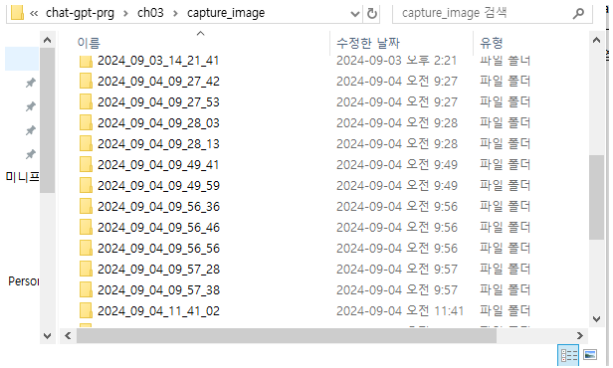
화면에 나타나는 모습

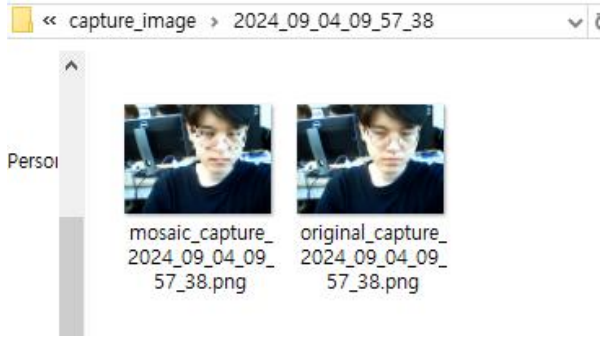
03

핵심 기능 소개

저장과 결과물

- 날짜와 시간별 폴더로 모자이크 사진과 원본 사진 저장





03

핵심 기능 소개

주요 코드

- 생성되는 파일을 저장하여 파일이 생성될 때마다 메시지 전송
- 기존의 파일과 생성된 파일의 값을 비교하여 그 값의 차이로 메시지 전송 여부 판단

```python
global existing_files
global current_files
directory_to_watch = r"D:\Study\Academy\Project_DoorGuard\capture_image"

# 기존의 파일 저장
def save_current_status():
    global existing_files
    existing_files = {f for f in os.listdir(directory_to_watch) if f.endswith('.png')}

# 새로운 파일 저장
def check_new_png():
    global existing_files
    global current_files
    current_files = {f for f in os.listdir(directory_to_watch) if f.endswith('.png')}

    # 새로 생긴 파일 탐지
    # 기존에 파일과 현재 파일의 값을 비교
    # 두 개의 파일이 동일한 파일이면 차이가 0
    # 동일하지 않은 파일이면 0이 아닌 수
    new_files = current_files - existing_files

    if new_files:
        print(f"New PNG files detected: {', '.join(new_files)}")
        import API.set_json
        #  새로운 파일이 탐지된 경우, 메시지 전송
        subprocess.run(['python', 'API/send_message.py'], check=False)
        existing_files = current_files
```

- **save_current_status():** 현재 파일 값을 저장함으로써, 이후엔 기존의 파일 값으로 남음
- **check_new_png():** 새로운 파일 값을 저장하고, 그 값을 근거로 기존의 파일값과 비교하여 파일 생성 여부를 판단
- **existing_files:** 기존에 저장된 파일 값
- **current_files:** 현재 생성된 파일 값
- **new_files:** 새로운 파일 판단 여부 값

03

핵심 기능 소개

주요 코드

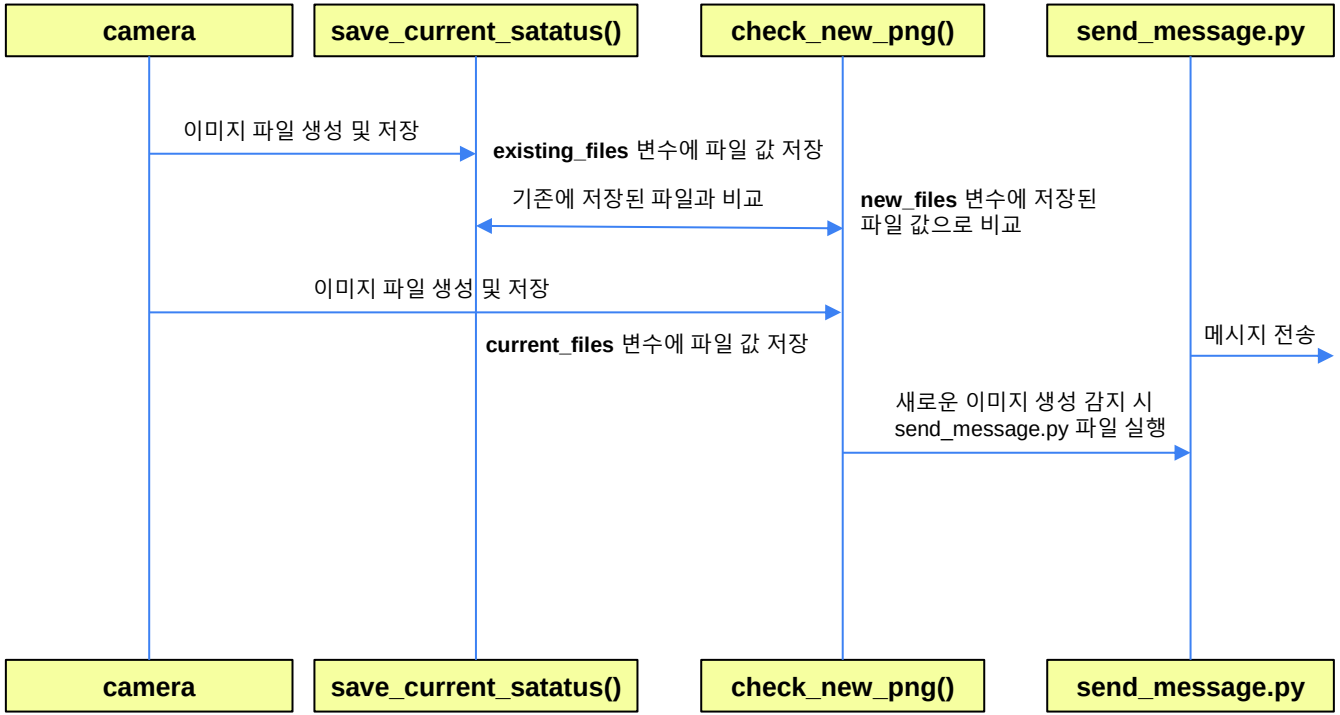
```python
global existing_files
global current_files
directory_to_watch = r"D:\Study\Academy\Project_DoorGuard\capture_image"

# 기존의 파일 저장
def save_current_status():
    global existing_files
    existing_files = {f for f in os.listdir(directory_to_watch) if f.endswith('.png')}

# 새로운 파일 저장
def check_new_png():
    global existing_files
    global current_files
    current_files = {f for f in os.listdir(directory_to_watch) if f.endswith('.png')}

    # 새로 생긴 파일 탐지
    # 기존에 파일과 현재 파일의 값을 비교
    # 두 개의 파일이 동일한 파일이면 차이가 0
    # 동일하지 않은 파일이면 0이 아닌 수
    new_files = current_files - existing_files

    if new_files:
        print(f"New PNG files detected: {', '.join(new_files)}")
        import API.set_json
        #  새로운 파일이 탐지된 경우, 메시지 전송
        subprocess.run(['python', 'API/send_message.py'], check=False)
        existing_files = current_files
```

- subprocess: 파이썬 스크립트에서 다른 프로세스를 실행하고 출력 결과를 가져올 수 있게 해주는 라이브러리
- 카카오톡 메시지 전송을 기능을 수행하는 소스 코드인 API 모듈의 send_message.py 코드를 subprocess 라이브러리의 run() 함수를 통해 실행

03

핵심 기능 소개

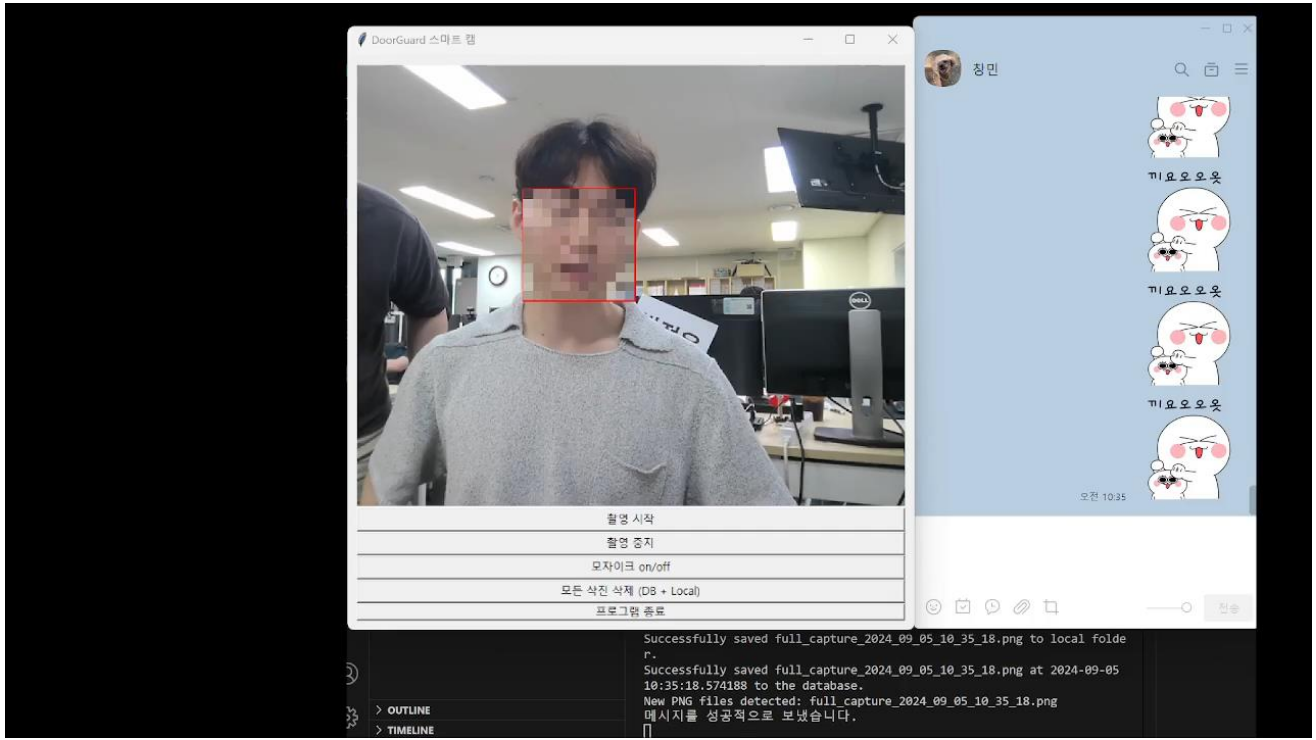
호출 흐름도



03

핵심 기능 소개

시연 영상



주요 코드

- 허가된 토큰을 발급 받아 메시지를 전송하기 위해 필요한 JSON파일 저장
- 토큰 인증을 위한 OAuth URL, Rest API 키 및 토큰 발급을 위해 리다이렉트되는 URL 을 데이터 값으로 입력하여 JSON 파일 생성

권한 부여 승인 코드 방식
(Authorization Code Grant)

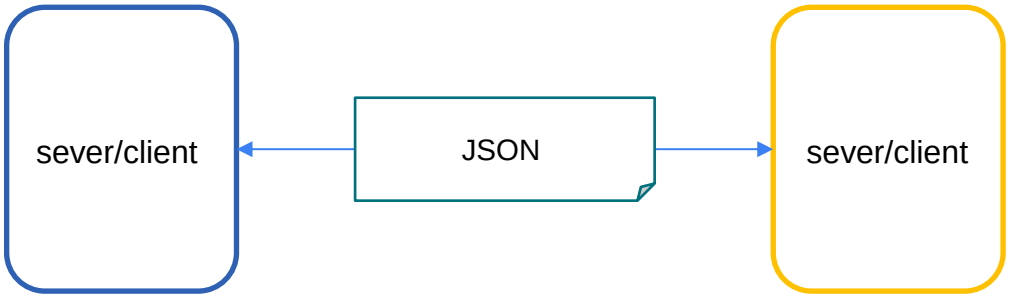
- **url, rest_api_key, redirect_uri 변수:** Access Token을 발급받기 위한 요청 파라미터 데이터
- **data 변수:** Access Token을 발급받기 위한 JSON타입의 데이터 포맷(파라미터)
- **response 변수:** 요청 후 받은 승인된 응답 데이터 (토큰 값이 포함된 JSON 타입의 데이터)

```python
import requests

# Read the data from tmp_data.txt file
with open('API/code_data.txt', 'r') as f:
    authorize_code = f.read()

# Now tmp2 contains the data from tmp1 (i.e., 'abc')
print(authorize_code)  # This will print 'abc'

url = 'https://kauth.kakao.com/oauth/token'
rest_api_key = 'c91a351774d89d3a1d7ed1077eb6bef0'
redirect_uri = 'https://example.com/oauth'

data = {
    'grant_type':'authorization_code',
    'client_id':rest_api_key,
    'redirect_uri':redirect_uri,
    'code': authorize_code,
    }

response = requests.post(url, data=data)
tokens = response.json()
#print(tokens)

# json 저장
import json
#1.
with open(r"D:\Study\Academy\Project_DoorGuard\API\kakao_code.json","w") as fp:
    json.dump(tokens, fp)
```

데이터 포맷

JSON 데이터 포맷: 크로스프랫폼(애플리케이션 및 플랫폼, OS등 환경에 구애받지 않고) 기반 네트워크 통신 장점과 런타임 메모리에 저장된 데이터로 마크업 타입의 텍스트로 변환 (데이터 직렬화)하여 통신하기 위함

- XML 보다 적은 메타데이터와 높은 파싱 기능성으로 현재 가장 널리 쓰이고 있는 데이터 포맷임



데이터 포맷

XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

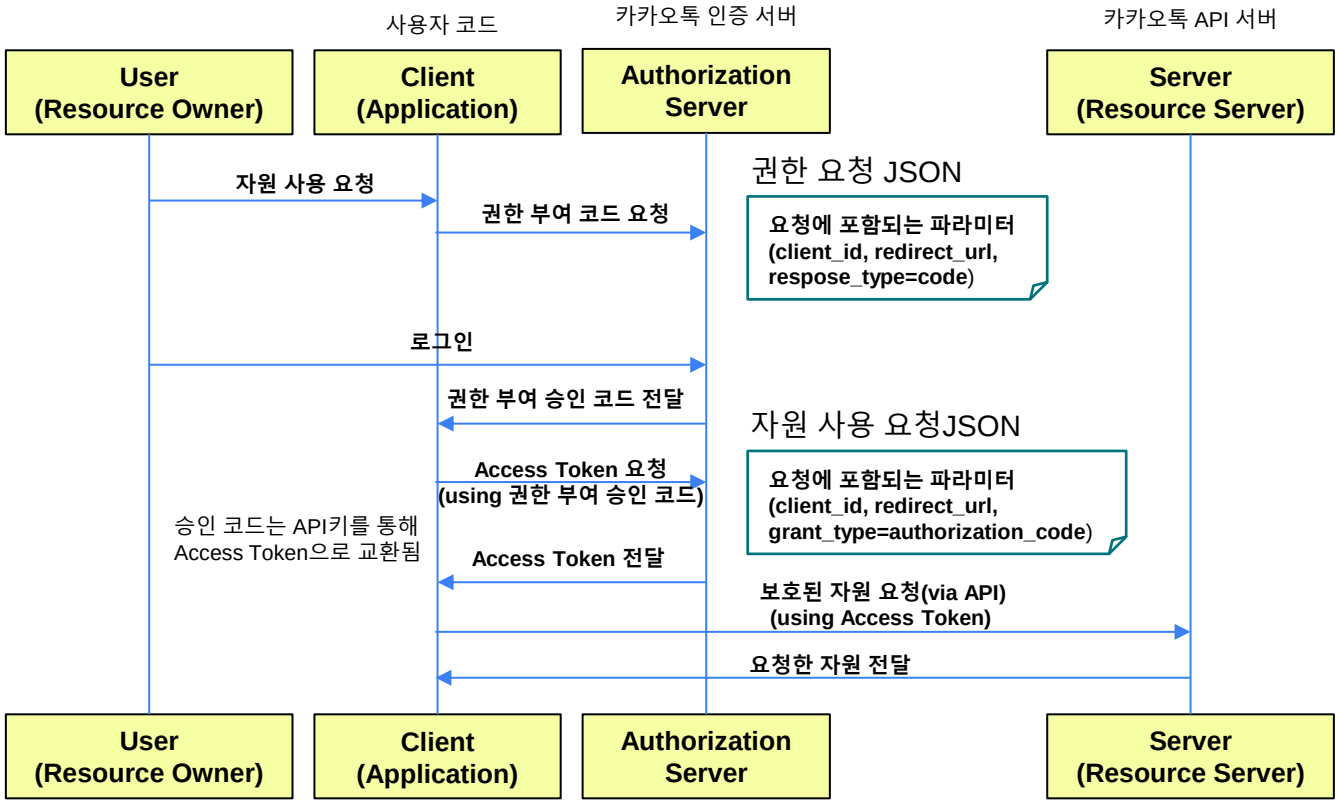
```
{  "empinfo" :
    {
        "employees" :  [
        {
            "name" : "James Kirk",
            "age" : 40,
        },
        {
            "name" : "Jean-Luc Picard",
            "age" : 45,
        },
        {
            "name" : "Wesley Crusher",
            "age" : 27,
        }
                        ]
    }
}
```

출처: https://medium.com/@oazzat19/what-is-the-difference-between-html-vs-xml-vs-json-254864972bbb

03

핵심 기능 소개

API 호출 흐름도



03

핵심 기능 소개

코드 설명

- 생성한 JSON 파일을 기반으로 메시지 전송

```
with open(r"D:\Study\Academy\Project_DoorGuard\API\kakao_code.json","r") as fp:
    tokens = json.load(fp)


url="https://kapi.kakao.com/v2/api/talk/memo/default/send"

# kapi.kakao.com/v2/api/talk/memo/default/send

headers={
    "Authorization" : "Bearer " + tokens["access_token"]
}

data={
    "template_object": json.dumps({
        "object_type":"text",
        "text":"사람이 감지되었습니다. 지금 바로 확인하세요! http://localhost/detect.php",
        "link":{
            "web_url":"http://localhost/detect.php"
        }
    })
}

response = requests.post(url, headers=headers, data=data)
response.status_code
if response.json().get('result_code') == 0:
    print('메시지를 성공적으로 보냈습니다.')
else:
    print('메시지를 성공적으로 보내지 못했습니다. 오류메시지 : ' + str(response.json()))
```

- **url 변수:** 메시지 전송을 위한 카카오톡 API URL
- **header 변수:** API로의 자원 사용(메시지 전송) 요청에 필요한 JSON 파일 헤더 데이터
- **data 변수:** API로의 자원 사용 요청에 필요한 JSON 파일 바디 데이터
- **response 변수:** API로의 자원 사용 요청에 대한 응답 데이터

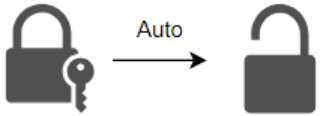
04

개선 프로세스

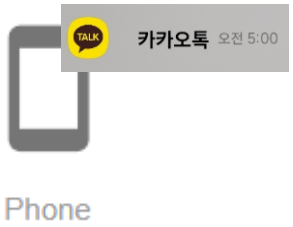
4-1 발전/확장 가능성

4-2 느낀 점

개선 아이디어



로그인 과정
자동화



상단 푸시 기능

감지 상황 추론
문장 생성 제공

04

개선 프로세스

느낀 점

프로그래밍 문법 중요성

- 풍부한 예제 이해
- 코드 이식/병합 수월

Why How 질문 중요

- XML vs JSON
- 기능 동작 원리
- 응용력 확장

팀플레이 & 결과

- 진행률 향상
- 시간대비 아웃풋 ↑

Ex. 버그 개선 소요 시간

Ex. 목표한 결과물 도출

Team !

Thank You!