

# 프로젝트 개인 보고서

Animated Facial Expression Conference APP

---



연세대학교  
YONSEI UNIVERSITY

수업	응용프로그래밍	교수님	이상훈 교수님
이름	정래현	전공	전기전자공학부
학번	2020142218	제출일	2021-12-13

# 1. 프로젝트 개요

## 1. 프로젝트 주제

코로나 바이러스로 인한 비대면 수업이, 사생활 침해와 집중력 저하를 발생시키는 문제상황을 인식하였다. 이를 해결하고자 얼굴 인식을 이용해 얼굴 표정을 실시간으로 animated 된 캐릭터로 변환하여 나타내어, 교수님과 학생들이 서로 원활한 소통을 하며 강의에 참여할 수 있도록 교육용 화상회의 안드로이드 App을 개발하기로 하였다.

## 2. 프로젝트 구성

얼굴의 위치와 표정을 인식하는 method는 python 오픈소스 OpenCV, dlib을 사용하였다. 이를 통해 얼굴의 68개 점의 랜드마크 좌표를 구할 수 있다. 이 랜드마크 데이터를 이용하여, animated 된 캐릭터를 만드는 과정은 Unity를 사용하였다. 사용자가 이용할 App을 만드는 과정은 안드로이드 스튜디오를 이용하였다. 파이썬 얼굴인식, 안드로이드, 유니티를 연결할 통신 방법으로는 파이썬의 Flask 서버를 이용하였다. 서버에 데이터를 저장하는 방법으로는 딕셔너리 형태로 데이터를 저장하는 MongoDB를 데이터베이스로 이용하였다.

## 3. 프로젝트 역할 분담

프로젝트 역할 분담은 박인호 팀원이 안드로이드 APP 파트 개발, 문예나 팀원이 Unity animate 파트 개발 그리고 본인이 서버 파트 개발을 하였다.

## 2. 프로젝트 시나리오

프로젝트 시나리오는 다음과 같다.

1. 안드로이드에서 강의실 번호와 비밀번호, 유저 정보를 이용해 생성 요청을 보낸다.
2. 서버에서 요청을 받아 첫 번째 DB에 강의실을 생성한다.
3. 안드로이드에서 사진을 촬영하고, 강의실 번호, 유저 정보와 함께 서버로 보낸다.
4. 서버에서 사진의 랜드마크를 추출하고, 유니티로 보낼 변수를 계산하여 두 번째 DB에 저장한다.
5. 유니티는 서버로 랜드마크 변수와 유저정보를 요청한다.
6. 유니티는 전달받은 랜드마크 변수로 모델링을 하여 유저정보와 함께 이미지를 서버로 보낸다.
7. 서버는 유저정보와 이미지를 전달받아 세 번째 DB에 저장한다.
8. 안드로이드는 서버로 모델링 이미지를 요청한다.
9. 서버는 안드로이드로 요청한 이미지를 전송한다.
10. 안드로이드는 서버에서 받은 모델링 이미지를 출력한다.

위의 2~9가 반복되며 진행된다.

### 3. 프로젝트 구현 파트 코드 및 상세 설명

#### 1. 프로젝트 구현 파트 개요

프로젝트의 서버 파트를 담당하였다. 서버는 안드로이드, 유니티와 통신하고, 데이터를 DB에 저장한다. DB는 pymongo를 이용해 MongoDB를 이용하였다. Flask의 Routing을 이용하여, 안드로이드와 유니티가 요청을 보낼 URL을 만들고, 요청에 따른 작업을 수행하는 함수들을 작성하였다. 작성한 라우팅 URL들과 대략적인 URL별 기능은 다음과 같다.

```
# 서버 접속 테스트용 처음화면 http://animated-facial-expression-conference.shop:5000/
@app.route('/')
def hello():...

# Android: 안드로이드에서 방 생성 (POST) API
@app.route('/open-new-room', methods=['POST'])
def open_new_room():...

# Android : 안드로이드에서 방 참여 (POST) API
@app.route('/enter-room', methods=['POST'])
def enter_room():...

# Android : 안드로이드에서 사진 전송받아 랜드마크로 변환 (POST) API
@app.route('/image-landmark', methods=['POST'])
def image_landmark():...

# Unity : 유니티로 랜드마크 전송 (GET) API
@app.route('/landmark-return', methods=['GET'])
def landmark_return():...

# Unity : 유니티에서 변환된 애니메이션 받아 저장 (POST) API
@app.route('/animated-image', methods=['POST'])
def animated_image():...

# Android: 안드로이드로 유니티를 통해 Animated 된 사진을 전송 (POST) API
@app.route('/image-return', methods=['POST'])
def image_return():...
```

서버 URL별 기능

## 2. 개발 환경 설정

파이썬의 Flask 서버를 개발할 때, dilb와 같은 패키지를 설치하는데 pip install의 경우 오류가 발생하였다. Anaconda를 이용하여 virtual env를 만들었고, conda install을 사용하여 해결하였다. python version 3.9를 이용하여 만들었고, 설치한 패키지들은 다음과 같다.

```
(animated_conference_server) C:\Users\W정래현>conda list
# packages in environment at C:\ProgramData\Anaconda3\envs\animated_conference_server:
#
# Name                                Version                                Build                                Channel
ca-certificates                      2021.10.8                             h5b45459_0                          conda-forge
certifi                              2021.10.8                             py39hcbf5309_1                      conda-forge
click                                 8.0.3                                 py39hcbf5309_1                      conda-forge
colorama                             0.4.4                                 pyh9f0ad1d_0                        conda-forge
dataclasses                          0.8                                   pyhc8e2a94_3                        conda-forge
dlib                                 19.22.0                               py39hf8509d4_0                      conda-forge
face_recognition                     1.3.0                                 pyhd3deb0d_2                        conda-forge
face_recognition_models              0.3.0                                 pyh9f0ad1d_0                        conda-forge
flask                                 2.0.2                                 pyhd8ed1ab_0                        conda-forge
freelut                               3.2.1                                 h0e60522_2                          conda-forge
freetype                             2.10.4                               h546665d_1                          conda-forge
icu                                  68.2                                 h0e60522_0                          conda-forge
intel-openmp                         2021.4.0                             h57928b3_3556                       conda-forge
itsdangerous                         2.0.1                                 pyhd8ed1ab_0                        conda-forge
jasper                               2.0.33                               h77af90b_0                          conda-forge
jbig                                 2.1                                  h8d14728_2003                       conda-forge
jinja2                               3.0.3                                 pyhd8ed1ab_0                        conda-forge
jpeg                                 9d                                   h8ffe710_0                          conda-forge
lcms2                               2.12                                 h2a16943_0                          conda-forge
lerc                                 3.0                                  h0e60522_0                          conda-forge
libblas                             3.9.0                               12_win64_mkl                       conda-forge
libcblas                            3.9.0                               12_win64_mkl                       conda-forge
libclang                             11.1.0                             default_h5c34c98_1                 conda-forge
libdeflate                           1.8                                 h8ffe710_0                          conda-forge
liblapack                           3.9.0                               12_win64_mkl                       conda-forge
liblapacke                           3.9.0                               12_win64_mkl                       conda-forge
libopencv                            4.5.3                               py39h4b6fd43_5                     conda-forge
libpng                               1.6.37                              h1d00b33_2                          conda-forge
libprotobuf                          3.18.1                              h7755175_0                          conda-forge
libtiff                              4.3.0                              hd413186_2                          conda-forge
libwebp-base                         1.2.1                              h8ffe710_0                          conda-forge
libzlib                              1.2.11                             h8ffe710_1013                       conda-forge
lz4-c                                1.9.3                              h8ffe710_1                          conda-forge
markupsafe                           2.0.1                              py39hb82d6ee_1                     conda-forge
mkl                                  2021.4.0                             h0e2418a_729                       conda-forge
numpy                                1.21.4                              py39h6635163_0                     conda-forge
olefile                              0.46                               pyh9f0ad1d_1                       conda-forge
opencv                              4.5.3                              py39hcbf5309_5                     conda-forge
opencv-python-headless               4.5.4.60                           py39_0                              fastai
openjpeg                             2.4.0                              hb211442_1                          conda-forge
openssl                              1.1.11                              h8ffe710_0                          conda-forge
pillow                               8.4.0                              py39h916092e_0                     conda-forge
pip                                  21.2.4                              py39haa95532_0                     conda-forge
py-opencv                            4.5.3                              py39h832f523_5                     conda-forge
pymongo                              4.0                                py39h415ef7b_0                     conda-forge
python                               3.9.7                              h6244533_1                          conda-forge
python_abi                           3.9                                 2_cp39                              conda-forge
qt                                    5.12.9                              h5909a2a_4                          conda-forge
setuptools                           58.0.4                              py39haa95532_0                     conda-forge
sqlite                                3.36.0                              h2bbff1b_0                          conda-forge
tbb                                  2021.4.0                             h2d74725_1                          conda-forge
tk                                    8.6.11                              h8ffe710_1                          conda-forge
tzdata                               2021e                               hda174b7_0                          conda-forge
vc                                    14.2                                h21ff451_1                          conda-forge
vs2015_runtime                       14.27.29016                         h5e58377_2                          conda-forge
werkzeug                             2.0.1                              pyhd8ed1ab_0                        conda-forge
wheel                                 0.37.0                              pyhd3eb1b0_1                       conda-forge
wincertstore                         0.2                                py39haa95532_2                     conda-forge
xz                                    5.2.5                              h62dcd97_1                          conda-forge
zlib                                 1.2.11                              h8ffe710_1013                       conda-forge
zstd                                  1.5.0                              h6255e5f_0                          conda-forge
```

### 3. 서버 배포

서버는 노트북을 이용하였다. 서버가 안드로이드 App과 Unity와 통신이 외부에서도 가능하도록, 노트북의 방화벽 포트를 열고, 사용하는 인터넷 공유기에서는 포트포워딩을 설정하였다. 그리고 가비아에서 도메인 <http://animated-facial-expression-conference.shop> 을 구입하여 노트북의 ip를 도메인에 연결하였다. 그리고 가비아 네임서버를 통해 도메인을 배포하였다.

**gabia. (주)가비아**

### 도메인 등록 확인증

Certificate of Domain Proxy Registration

도메인명	ANIMATED-FACIAL-EXPRESSION-CONFERENCE.SHOP	
소유자 정보	소유자명	정래현 (Jaong Faahyeon)
	주소	[REDACTED]
	전화번호	[REDACTED]
	이메일	[REDACTED]
관리자 정보	관리자명	정래현 (Jaong Faahyeon)
	주소	[REDACTED]
	전화번호	[REDACTED]
	이메일	[REDACTED]
네임서버 정보	1차	211.234.118.50 ns.gabia.co.kr
	2차	121.78.117.39 ns1.gabia.co.kr
등록일/만료일	등록일	2021년 12월 04일
	만료일	2022년 12월 09일

상기와 같이 당사에 도메인 등록되었음을 확인합니다.

발행일 2021년 12월 13일

(주)가비아 대표이사 김흥국

이 확인증은 (주)가비아를 통해 도메인을 등록하였음을 확인하는 문서로, 법적 효력은 없습니다.

한국인터넷진흥원 (KISA) | 한국인터넷진흥원 | 한국인터넷진흥원 | 한국인터넷진흥원 | 한국인터넷진흥원

## 4. 코드 상세 설명

서버 파일인 app.py에 대한 설명이다. 코드 중 #디버그 라고 적혀있는 부분은 서버가 정상적으로 작동하는지를 확인하기 위해 작성한 코드로, 주석처리하거나 지워도 문제없이 돌아간다.

### 1. 사용한 package, Flask/MongoDB 설정, dlib face detector/landmark predictor 설정

```
1 from flask import Flask, request
2 from pymongo import MongoClient
3 import numpy as np
4 import time
5 import dlib
6 import cv2
7 import base64
8
9 app = Flask(__name__)
10
11 # MongoDB DB이름 : DB_animated_conference_server
12 client = MongoClient('localhost', 27017)
13 db = client.DB_animated_conference_server
14
15 # face_detector와 landmark predictor 정의
16 detector = dlib.get_frontal_face_detector()
17 predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

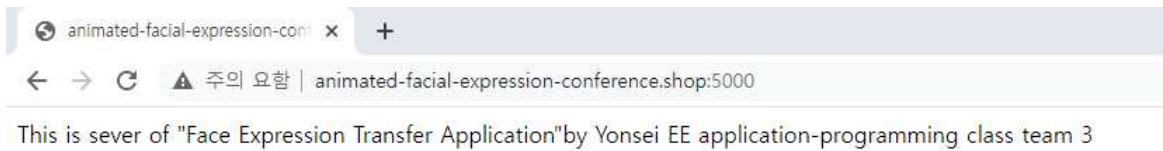
위와 같이 package를 import 하였고, PyMongo를 이용해 DB\_animated\_conference\_server라는 이름의 DB를 설정하였다. 얼굴 인식에 사용할 dlib face detector와 landmark predictor를 정의하였다. shape\_predictor\_68\_face\_landmarks.dat 파일은 68개의 얼굴 랜드마크를 따는데 필요한 파일로, app.py와 같은 디렉토리에 있다.

### 2. 서버 접속 테스트용 처음화면

```
19 # 서버 접속 테스트용 처음화면 http://animated-facial-expression-conference.shop:5000/
20 @app.route('/')
21 def hello():
22     return f'This is sever of \'Face Expression Transfer Application\' \
23         \'by Yonsei EE application-programming class team 3 \'
```

URL: <http://animated-facial-expression-conference.shop:5000/>

외부에서 서버에 접속할 때 서버가 돌아가고 있는 상태인지 확인하기 위해 만들어놓은 함수이다. 서버는 5000번 포트를 이용하여 구동되므로 URL에 :5000을 추가로 적어야 접속할 수 있다. 정상적으로 동작하면 외부에서 Chrome으로 접속했을때 다음과 같이 보인다.



### 3. Android: 안드로이드에서 방 생성(POST) API

```
25 # Android: 안드로이드에서 방 생성 (POST) API
26 @app.route('/open-new-room', methods=['POST'])
27 def open_new_room():
28     # 안드로이드에서 방번호, 방비밀번호, user_key를 입력받아 서버로 전송
29     room_num_receive = int(request.json['room_num'])
30     room_password_receive = int(request.json['room_password'])
31     user_key_receive = request.json['user_key']
32
33     # 디버그
34     print(f"{user_key_receive} Opened [ room num : {room_num_receive}, password : {room_password_receive} ]")
35
36     # DB에 있는 데이터인지 확인
37     if db.user_room_list.find_one({'room_num': room_num_receive}) != None:
38         # DB에 이미 있는 방번호인 경우
39
40         # 디버그
41         print("open-new-room : Already existing room number")
42
43         # 이미 있는 방번호라는 메시지 return
44         return {"msg": 'Already existing room number'}
45     # DB에 없을 경우
46     else:
47         # 서버에 저장할 내용
48         user_room_list = {
49             'room_num': room_num_receive,
50             'room_password': room_password_receive,
51             'user_key': user_key_receive,
52             'time': time.strftime('%Y-%m-%d %H:%M:%S')
53         }
54         # 서버에 저장
55         db.user_room_list.insert_one(user_room_list)
56         return {"msg": 'New room created'}
```

안드로이드에서 방을 생성하는 함수이다. 안드로이드에서 방번호, 방비밀번호, user\_key를 입력받아 POST 방식으로 서버로 전송한다. 서버에서는 방번호가 이미 존재하는지 확인하고 존재하지 않을 경우 새 방을 생성하여 유저와 방 정보를 저장하는 DB인 user\_room\_list에 저장한다.



#### 4. Android : 안드로이드에서 방 참여(POST) API

```
58 # Android : 안드로이드에서 방 참여 (POST) API
59 @app.route('/enter-room', methods=['POST'])
60 def enter_room():
61     # 안드로이드에서 방번호, 비밀번호번호, user_key를 입력받아 서버로 전송
62     room_num_receive = int(request.json['room_num'])
63     room_password_receive = int(request.json['room_password'])
64     user_key_receive = request.json['user_key']
65
66     # DB에 방이 있는지 확인
67     if db.user_room_list.find_one({'room_num': room_num_receive}) != None:
68         # 비밀번호가 맞는지 확인
69         if db.user_room_list.find_one({'room_num': room_num_receive})['room_password'] == room_password_receive:
70             # 비밀번호가 맞는 경우
71             # 서버에 저장할 내용
72             user_room_list = {
73                 'room_num': room_num_receive,
74                 'room_password': room_password_receive,
75                 'user_key': user_key_receive,
76                 'time': time.strftime('%Y-%m-%d %H:%M:%S')
77             }
78             # 서버에 저장
79             db.user_room_list.insert_one(user_room_list)
80
81             # 디버그
82             print(f"{user_key_receive} Entered [ room num : {room_num_receive}, password : {room_password_receive} ]")
83
84             # DB에 정보 추가되었을때 방에 성공적으로 입장했다는 메시지 return
85             return {"msg": 'Room enter success'}
86         # 비밀번호가 다른 경우
87         else:
88             # 디버그
89             print("enter-room : Wrong password")
90
91             # 비밀번호가 다를 때 비밀번호가 다르다는 메시지 return
92             return {"msg": 'Wrong Password'}
93     # DB에 방이 없는 경우
94
95     else:
96         # 디버그
97         print("enter-room : No room number")
98
99         # 방이 없을때 입력한 방이 없다는 메시지 return
100         return {"msg": 'No room number'}
```

안드로이드에서 방을 생성하는 함수이다. 안드로이드에서 방번호, 비밀번호번호, user\_key를 입력받아 POST 방식으로 서버로 전송한다. 서버에서는 방번호가 이미 존재하는지 확인하고 존재하지 않을 경우 새 방을 생성하여 유저와 방 정보를 저장하는 DB인 user\_room\_list에 저장한다.

## 5. Android : 안드로이드에서 사진 전송받아 랜드마크로 변환(POST) API

```
101 # Android : 안드로이드에서 사진 전송받아 랜드마크로 변환 (POST) API
102 @app.route('/image-landmark', methods=['POST'])
103 def image_landmark():
104     # 안드로이드에서 방번호, user_key, base64 형식의 이미지를 전송
105     room_num_receive = int(request.json['room_num'])
106     user_key_receive = request.json['user_key']
107     image_receive = request.json['b6pimg']
108
109     # 디버그
110     print(f"from Android, image sent : {room_num_receive}, {user_key_receive}")
111
112     # ----- 랜드마크 따는 파트 -----
113     # face_detector와 landmark predictor 정의
114     global detector
115     global predictor
116
117     # base64 이미지 read
118     try:
119         decoded_data = base64.b64decode(image_receive)
120         np_data = np.frombuffer(decoded_data, np.uint8)
121         img = cv2.imdecode(np_data, cv2.IMREAD_UNCHANGED)
122     except:
123         # 디버그
124         print('Image read failed')
125
126         return {"msg": 'Image read failed'}
127
128     # resize
129     r = 600. / img.shape[1]
130     dim = (600, int(img.shape[0] * r))
131     resized = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
132
133     # 얼굴 detection
134     try:
135         # 얼굴이 있는 사각형 detect
136         rect = detector(resized, 1)[0]
137
138         # 얼굴 랜드마크 68개 점 detect
139         shape = predictor(resized, rect)
140
141         # 점들을 리스트로 저장
142         landmarks = []
143         for j in range(68):
144             x, y = shape.part(j).x, shape.part(j).y
145             landmarks.append([x, y])
146             # # 디버그 : facial landmark를 빨간색 점으로 찍어서 표현
147             # cv2.circle(resized, (x, y), 1, (0, 0, 255), -1)
148     # 얼굴 인식에 실패할 경우
149     except:
150         # 디버그 print
151         print('Face read failed')
152         return {"msg": 'Face read failed'}
```

```

158 # 유니티로 전송할 변수들 계산
159 # 왼쪽 눈 변수
160 eye_left = (((Landmarks[43][0]-Landmarks[47][0])**2+(Landmarks[43][1]-Landmarks[47][1])**2)**(1/2)+
161             ((Landmarks[44][0]-Landmarks[46][0])**2+(Landmarks[44][1]-Landmarks[46][1])**2)**(1/2))\
162             /2/(((Landmarks[42][0]-Landmarks[45][0])**2+(Landmarks[42][1]-Landmarks[45][1])**2)**(1/2))
163 # 오른쪽 눈 변수
164 eye_right = (((Landmarks[38][0]-Landmarks[40][0])**2+(Landmarks[38][1]-Landmarks[40][1])**2)**(1/2)+
165              ((Landmarks[37][0]-Landmarks[41][0])**2+(Landmarks[37][1]-Landmarks[41][1])**2)**(1/2))\
166              /2/(((Landmarks[36][0]-Landmarks[39][0])**2+(Landmarks[36][1]-Landmarks[39][1])**2)**(1/2))
167 # 입 벌어진 정도 변수
168 mouth_openclose = (((Landmarks[50][0]-Landmarks[58][0])**2+(Landmarks[50][1]-Landmarks[58][1])**2)**(1/2)+
169                    ((Landmarks[51][0]-Landmarks[57][0])**2+(Landmarks[51][1]-Landmarks[57][1])**2)**(1/2)+
170                    ((Landmarks[52][0]-Landmarks[56][0])**2+(Landmarks[52][1]-Landmarks[56][1])**2)**(1/2))\
171                    /3/(((Landmarks[48][0]-Landmarks[54][0])**2+(Landmarks[48][1]-Landmarks[54][1])**2)**(1/2))
172 # 입 왼쪽 기울기 변수
173 mouth_inclination_left = abs((Landmarks[54][1]-(Landmarks[62][1]+Landmarks[66][1])/2)/(Landmarks[54][0]-(Landmarks[62][0]+Landmarks[66][0])/2))
174 # 입 오른쪽 기울기 변수
175 mouth_inclination_right = abs((Landmarks[48][1]-(Landmarks[62][1]+Landmarks[66][1])/2)/(Landmarks[48][0]-(Landmarks[62][0]+Landmarks[66][0])/2))

177 # ----- 입력받은 변수로 db 업데이트 -----
178 # 유저를 DB에 입력받은 번호의 방에 유저 키 정보가 있을 때
179 if db.user_room_list.find_one({'room_num': room_num_receive, 'user_key': user_key_receive}) != None:
180     # 랜드마크 DB에 입력받은 번호의 방에 유저 키 있을 때 - 기존 정보 변경
181     if db.landmarks_list.find_one({'room_num': room_num_receive, 'user_key': user_key_receive}) != None:
182         # id로 찾아 서버 데이터 업데이트
183         data_id = db.landmarks_list.find_one({'room_num': room_num_receive, 'user_key': user_key_receive})['_id']
184         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'eye_left': eye_left}})
185         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'eye_right': eye_right}})
186         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'mouth_openclose': mouth_openclose}})
187         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'mouth_inclination_left': mouth_inclination_left}})
188         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'mouth_inclination_right': mouth_inclination_right}})
189         db.landmarks_list.update_one({'_id': data_id}, {'$set': {'time': time.strftime('%Y-%m-%d %H:%M:%S')}})
190         print('Landmark updated success')
191         return {'msg': 'Landmark updated success'}
192
193 # 입력받은 번호의 방에 유저 키 없을 때 - 정보 새로 추가
194 else:
195     # 서버에 저장할 내용
196     room_landmark = {
197         'room_num': room_num_receive,
198         'user_key': user_key_receive,
199         'eye_left': float(eye_left),
200         'eye_right': float(eye_right),
201         'mouth_openclose': float(mouth_openclose),
202         'mouth_inclination_left': float(mouth_inclination_left),
203         'mouth_inclination_right': float(mouth_inclination_right),
204         'time': time.strftime('%Y-%m-%d %H:%M:%S')
205     }
206     # 서버에 저장
207     db.landmarks_list.insert_one(room_landmark)
208     # 디버그
209     print('from Android, Landmark inserted success')
210     return {'msg': 'Landmark inserted success'}
211
212 # DB에 입력받은 번호와 유저 정보가 없을 때
213 else:
214     return {'msg': 'No room number or No user key'}

```

안드로이드에서 받은 사진에서 얼굴 랜드마크를 따는 파트이다. 앞서 선언한 face detector와 landmark predictor로 얼굴을 인식한다. 그럼 얼굴을 따라 68개의 face landmark가 생긴다. 이 랜드마크들 간의 거리를 이용하여, Unity에서 표정을 표현하는데 필요한 변수들을 계산하여 저장한다. 사진에서 얼굴을 인식하지 못할 경우 ERROR가 발생하는데, 이를 try except문으로 잡아 얼굴을 감지할 수 없었다는 메시지를 반환한다. 이후 입력받은 방 번호와 유저 키가 DB user\_room\_list에 있는지 확인한다. 있을 경우, 방 번호와 유저 키가 랜드마크를 저장하는 DB인 landmarks\_list에 있는지 확인한다. 없을 경우 처음 보낸 사진이므로, 랜드마크를 DB에 새로 추가한다. 있을 경우 기존의 DB에서 랜드마크 정보와 시간 정보를 업데이트 한다.

## 6. Unity : 유니티로 랜드마크 전송(GET) API

```
216 # Unity : 유니티로 랜드마크 전송 (GET) API
217 @app.route('/landmark-return', methods=['GET'])
218 def landmark_return():
219     # Unity 에서 방 번호와, 0, 1, 2, 3 ... 의 숫자로 한명씩 랜드마크를 요청
220     room_num_receive = int(request.args.get('room_num'))
221     user_num_receive = int(request.args.get('user_num'))
222
223     # 요청받은 인자로 방에 있는 유저들의 랜드마크 변수를 리스트에 담음
224     room_member_list = list(db.landmarks_list.find({'room_num': int(room_num_receive)}))
225
226     # 디버그
227     print(f"from Unity, landmark requested: {room_num_receive}, {user_num_receive}")
228
229     # 리스트[0], 리스트[1], [2], ... 의 원소들에서 값을 json 형식으로 return
230     try:
231         return {
232             'user_key': room_member_list[user_num_receive]['user_key'],
233             'eye_left': room_member_list[user_num_receive]['eye_left'],
234             'eye_right': room_member_list[user_num_receive]['eye_right'],
235             'mouth_openclose': room_member_list[user_num_receive]['mouth_openclose'],
236             'mouth_inclination_left': room_member_list[user_num_receive]['mouth_inclination_left'],
237             'mouth_inclination_right': room_member_list[user_num_receive]['mouth_inclination_right'],
238             'Valid': "True"
239         }
240     # 리스트에 원소가 더 없을 경우, Valid: False 반환으로 유니티에 리스트의 끝을 알림
241     except:
242         return {
243             'user_key': 0,
244             'eye_left': 0,
245             'eye_right': 0,
246             'mouth_openclose': 0,
247             'mouth_inclination_left': 0,
248             'mouth_inclination_right': 0,
249             'Valid': "False"
250         }
```

유니티로 랜드마크를 이용하여 계산한 변수들을 전송하는 함수이다. 방 번호와, 0, 1, 2, 3 ... 의 숫자를 증가시키며 한명씩 랜드마크 변수들을 요청한다. 유니티에서 전달받은 방번호에 있는 유저들의 랜드마크 변수들을 리스트에 담고, 숫자가 증가함에 따라 한명씩 리스트 값들을 전달한다. try except문을 이용하여 리스트의 끝일 경우 Valid: False 반환으로 유니티에 리스트의 끝을 알린다.

## 7. Unity : 유니티에서 변환된 애니메이션 받아 저장 (POST) API

```
252 # Unity : 유니티에서 변환된 애니메이션 받아 저장 (POST) API
253 @app.route('/animated-image', methods=['POST'])
254 def animated_image():
255     # Unity 에서 방 번호와 user_key, 이미지를 서버로 전송
256     room_num_receive = int(request.form['room_num'])
257     image_receive = request.form['image']
258     user_key_receive = request.form['user_key']
259
260     # 디버그
261     print(f"from Unity, animated-image sent : {room_num_receive} {user_key_receive}")
262
263     # 이미지 DB에 입력받은 번호의 방에 유저 키 있을 때 - 기존 정보 변경
264     if db.animated_images.find_one({'room_num': room_num_receive, 'user_key': user_key_receive}) != None:
265         data_id = db.animated_images.find_one({'room_num': room_num_receive, 'user_key': user_key_receive})['_id']
266         db.animated_images.update_one({'_id': data_id}, {'$set': {'image': image_receive}})
267         db.animated_images.update_one({'_id': data_id}, {'$set': {'time': time.strftime('%y-%m-%d %H:%M:%S')}})
268         print('Animated-image updated success')
269         return {"msg": 'Animated-image updated success'}
270
271     # 입력받은 번호의 방과 유저 키 없을 때 - 정보 새로 추가
272     else:
273         # 유니티에서 받은 방번호, 이미지, 유저 키
274         animated_image_list = {
275             'room_num': room_num_receive,
276             'image': image_receive,
277             'user_key': user_key_receive,
278             'time': time.strftime('%y-%m-%d %H:%M:%S')}
279
280         db.animated_images.insert_one(animated_image_list)
281         print('Animated-image inserted success')
282         return {"msg": 'Animated-image inserted success'}
```

유니티에서 변환된 애니메이션 이미지를 받아 저장하는 함수이다. Unity에서 방 번호와 user\_key, 이미지를 전송받는다. 애니메이션 이미지를 저장할 animated\_images DB에 image 정보가 있는지 확인하고, 없을 경우 새로 DB에 정보를 추가하고, 있을 경우 DB에 있는 기존 정보를 업데이트한다.



## 8. Android: 안드로이드로 유니티를 통해 Animated 된 사진을 전송(POST) API

```
284 # Android: 안드로이드로 유니티를 통해 Animated 된 사진을 전송 (POST) API
285 @app.route('/image-return', methods=['POST'])
286 def image_return():
287     # 안드로이드는 방 번호를 요청
288     room_num_receive = int(request.json['room_num'])
289
290     # DB에 입력받은 번호의 방이 있을 때
291     if db.animated_images.find_one({'room_num': room_num_receive}) != None:
292         room_image_list = list(db.animated_images.find({'room_num': room_num_receive}))
293         # 안드로이드로 전송할 return_json 구성
294         # return_json = {
295         #     "msg": 'True',
296         #     "count": 방에 있는 사람 수,
297         #     "0": {"userkey": 유저키, "img": 이미지},
298         #     "1": {"userkey": 유저키, "img": 이미지},
299         #     "2": {"userkey": 유저키, "img": 이미지},
300         #     ... (방에 있는 사람 수 만큼)
301         # }
302         return_json = {}
303         return_json["msg"] = 'True'
304         return_json["count"] = len(room_image_list)
305         return_json["0"] = room_image_list[0]['image']
306         count = 0
307         for i in room_image_list:
308             return_json[str(count)] = {
309                 "userkey": i['user_key'],
310                 "img": i['image']
311             }
312             count = int(count)
313             count += 1
314
315         # 디버그
316         print("To Android, Image return success")
317
318         # 앱으로 전송
319         return return_json
320     # DB에 입력받은 번호의 방이 없을 때
321     else:
322         print('image-return : No room number')
323         return {"msg": 'No room number'}
```

안드로이드로 Animated 된 사진을 전송하는 함수이다. 안드로이드에서는 방번호를 요청한다. 서버에서는 animated\_images DB에서 방 번호에 맞는 정보들을 찾아 { "msg": 'True', "count": 방에 있는 사람 수, "0": {"userkey": 유저키, "img": 이미지}, "1": {"userkey": 유저키, "img": 이미지}, "2": {"userkey": 유저키, "img": 이미지} ... (방에 있는 사람 수 만큼) } 와 같은 json 형식으로 안드로이드로 사진 정보를 보낸다.

## 9. 서버 구동

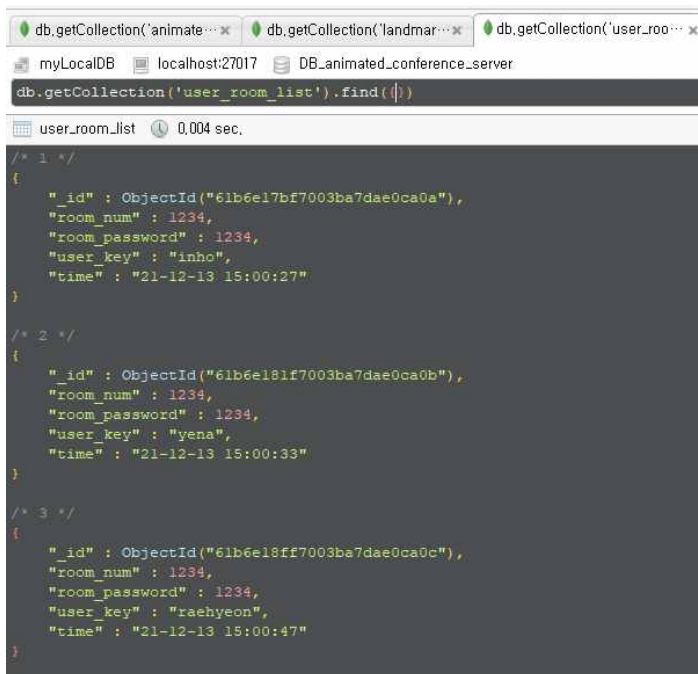
```
325 | # -----  
326 | # 서버 구동  
327 | ▶ if __name__ == '__main__':  
328 |     app.run('0.0.0.0', port=5000, debug=True)
```

Flask 서버를 구동시키는 함수이다.

## 5. DB 데이터

user\_room\_list, landmarks\_list, animated\_images 세 개의 DB를 사용하였는데, 각각의 DB에 데이터가 어떤 식으로 들어가는지, MongoDB를 데이터를 볼 수 있는 Robo3T 라는 프로그램을 이용하여 확인해 보았다.

### 1. user\_room\_list



```
db.getCollection('animate...') db.getCollection('landmar...') db.getCollection('user_roo...')
myLocalDB localhost:27017 DB_animated_conference_server
db.getCollection('user_room_list').find({})

user_room_list 0.004 sec.

/* 1 */
{
  "_id" : ObjectId("61b6e17bf7003ba7dae0ca0a"),
  "room_num" : 1234,
  "room_password" : 1234,
  "user_key" : "inho",
  "time" : "21-12-13 15:00:27"
}

/* 2 */
{
  "_id" : ObjectId("61b6e181f7003ba7dae0ca0b"),
  "room_num" : 1234,
  "room_password" : 1234,
  "user_key" : "yena",
  "time" : "21-12-13 15:00:33"
}

/* 3 */
{
  "_id" : ObjectId("61b6e18ff7003ba7dae0ca0c"),
  "room_num" : 1234,
  "room_password" : 1234,
  "user_key" : "raehyeon",
  "time" : "21-12-13 15:00:47"
}
```



## 2. landmarks\_list

```
db.getCollection('animate...x db.getCollection('landmar...x db.getCollection('user_roo...x
myLocalDB localhost:27017 DB_animated_conference_server
db.getCollection('landmarks_list').find({})

landmarks_list 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("61b6e1b8f7003ba7dae0ca0f"),
  "room_num" : 1234,
  "user_key" : "raehyeon",
  "eye_left" : 0.329460377622765,
  "eye_right" : 0.337118152367076,
  "mouth_openclose" : 0.466837940142696,
  "mouth_inclination_left" : 0.0467289719626168,
  "mouth_inclination_right" : 0.029126213592233,
  "time" : "21-12-13 15:06:47"
}

/* 2 */
{
  "_id" : ObjectId("61b6e1baf7003ba7dae0ca11"),
  "room_num" : 1234,
  "user_key" : "inho",
  "eye_left" : 0.227780675635309,
  "eye_right" : 0.20682825947917,
  "mouth_openclose" : 0.334152905705322,
  "mouth_inclination_left" : 0.138297872340426,
  "mouth_inclination_right" : 0.05,
  "time" : "21-12-13 15:06:52"
}

/* 3 */
{
  "_id" : ObjectId("61b6e1caf7003ba7dae0ca14"),
  "room_num" : 1234,
  "user_key" : "yena",
  "eye_left" : 0.427399023496108,
  "eye_right" : 0.416286333576893,
  "mouth_openclose" : 0.486152458360508,
  "mouth_inclination_left" : 0.0526315789473684,
  "mouth_inclination_right" : 0.0666666666666667,
  "time" : "21-12-13 15:06:55"
}
```

## 3. animaged\_images

```
db.getCollection('animate...x db.getCollection('landmar...x db.getCollection('user_roo...x
myLocalDB localhost:27017 DB_animated_conference_server
db.getCollection('animaged_images').find({})

animaged_images 0.018 sec.

/* 1 */
{
  "_id" : ObjectId("61b6e1b9f7003ba7dae0ca10"),
  "room_num" : 1234,
  "image" : "iVBORw0KGgoAAAANSUHEUgAABL4AAAL8CAIAAAABhwOsGAAAgAE1EQVR4AWTdy2YkSZKlZS/dI7OggT4YlthgAezw/s+E0+IurKy",
  "user_key" : "raehyeon",
  "time" : "21-12-13 17:36:18"
}

/* 2 */
{
  "_id" : ObjectId("61b6e1bbf7003ba7dae0ca12"),
  "room_num" : 1234,
  "image" : "iVBORw0KGgoAAAANSUHEUgAABL4AAAL8CAIAAAABhwOsGAAAgAE1EQVR4AWTdy2YkSZKlZS/dI7OggT4YlthgAezw/s+E0+IurKy",
  "user_key" : "inho",
  "time" : "21-12-13 17:36:17"
}

/* 3 */
{
  "_id" : ObjectId("61b6e1cbf7003ba7dae0ca15"),
  "room_num" : 1234,
  "image" : "iVBORw0KGgoAAAANSUHEUgAABL4AAAL8CAIAAAABhwOsGAAAgAE1EQVR4AWTdy2YkSZKlZS/dI7OggT4YlthgAezw/s+E0+IurKy",
  "user_key" : "yena",
  "time" : "21-12-13 17:36:18"
}
```

## 6. 한계점 / 개선해야 할 점

실시간 음성 통신, 말풍선을 이용한 채팅, 유저가 캐릭터를 선택하고 여러 액세서리를 착용하는 기능, 강의실 로그아웃 기능 등이 아직 구현되지 않았고, 디버그 모드로 서버를 구동하여, 불필요한 코드들을 제거하면 통신 속도가 더 빨라질 것으로 예상된다.