

# 아이 파이

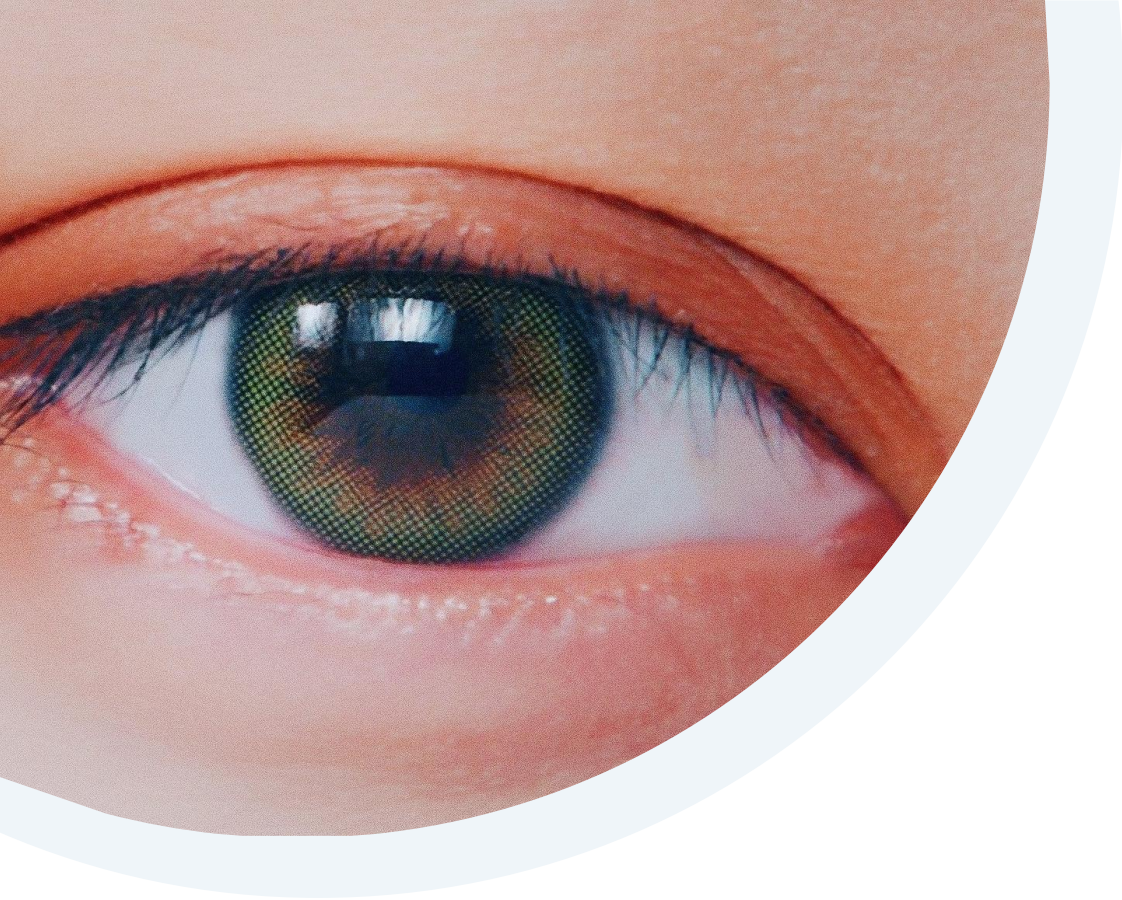
EYE-FI



- 비대면 수업 개선을 위한 시각 분석 솔루션 -

*Vision analysis solution for non-face-to-face class*



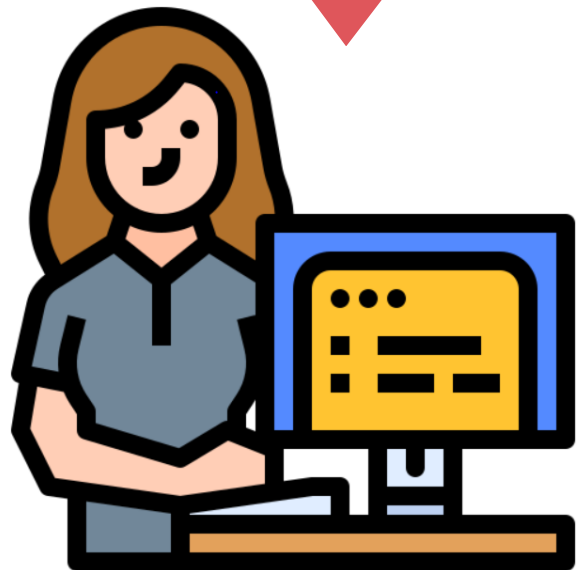


# CONTENTS

1. 팀원 소개 및 R&R
2. 배경 & 목표
3. 개발 과정
4. 결과물
5. 기대 효과 & 개선 방안

# 01

## 팀원 소개 및 R&R



송 정 현

Flask 서버 & Web 구현



서 준 교

팀장 역할 수행 및  
영상 처리 알고리즘 개발



김 림 매

데이터 분석 & PPT

02

## 배경 & 목표



# SUSTAINABLE DEVELOPMENT GOALS



## Quality Education

비대면 교육 서비스 개선



코로나19의 확산 이래 대학교에서  
비대면 수업이 많이 진행되고 있다.  
2020년 국내 203개 대학의 대학생  
2만 1784명을 대상으로 설문조사를  
실시한 결과  
설문에 참여한 **대학생의 82.0%**가  
온라인 수업의 **질이 떨어진다고**  
답변함

출처 - 국회입법조사처 보도자료



Eye Tracking 기술을 활용해  
학생들이 수업에 얼마나  
집중하는지에 대한 데이터를  
수집하고,  
강사에게 **학생들의 집중도**  
수치를 직관적으로 시각화하여  
**수업의 질을 높일 수 있는**  
**정보를 제공**하도록 한다.

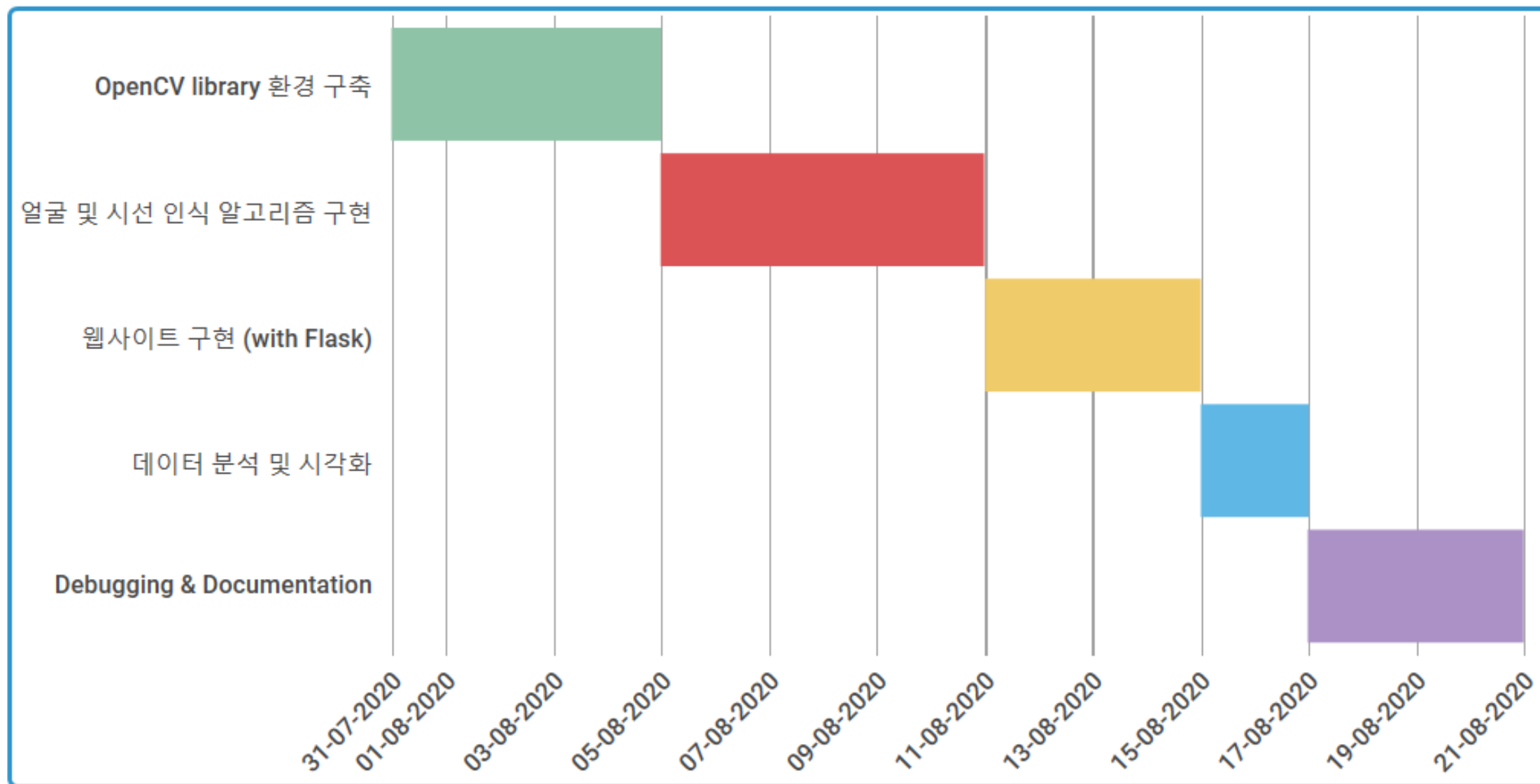
# 03

## 개발 과정

◆ 개발 일정

◆ 협업 과정

# 개발 일정





# 협업 과정



GitHub

Joonkkyo issue #2 solved 11c1d9e 3 days ago 5 commits		
.idea	initial commit	14 days ago
templates	issue #2 solved	3 days ago
README.md	Create README.md	14 days ago
haarcascade_frontalface_default.xml	initial commit	14 days ago
list_file.txt	issue #2 solved	3 days ago
main.py	issue #2 solved	3 days ago
server.py	issue #2 solved	3 days ago
shape_predictor_68_face_landmarks.dat	initial commit	14 days ago

< Code sharing >

0 Open 3 Closed

concentration이 가끔 떨어지지 않는 버그 발생 bug

#3 by Joonkkyo was closed 18 hours ago

main.py 데이터 flask 서버와 연동 enhancement

#2 by Joonkkyo was closed yesterday

환경마다 eye gaze ratio가 변화하는 것을 확인

#1 by Joonkkyo was closed 19 hours ago

< Issue Tracking >

### ◆ 시스템 구조

### ◆ 세부 기능

- 얼굴 인식 알고리즘
- 시선 인식 알고리즘
- 집중도 계산
- 데이터 분석
- Flask 서버 연동 및 시각화

# 시스템 구조



Eye tracking

python™

Dlib

+

OpenCV



Web



Flask

+

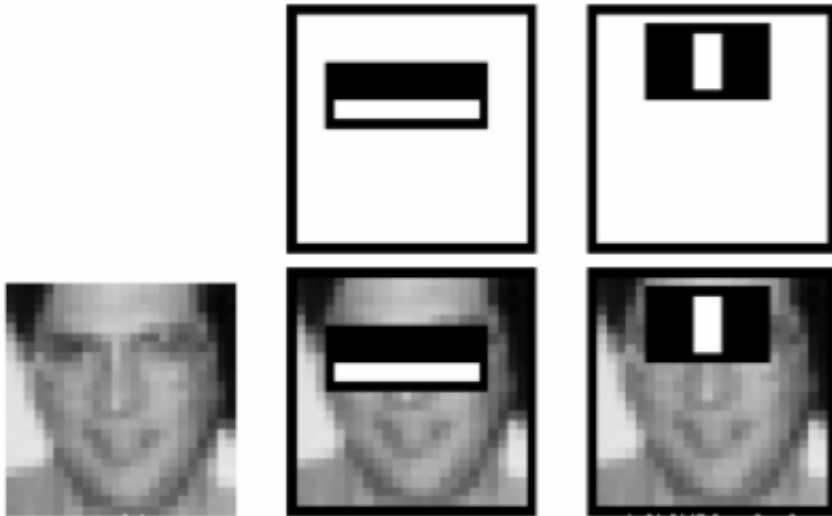


matplotlib

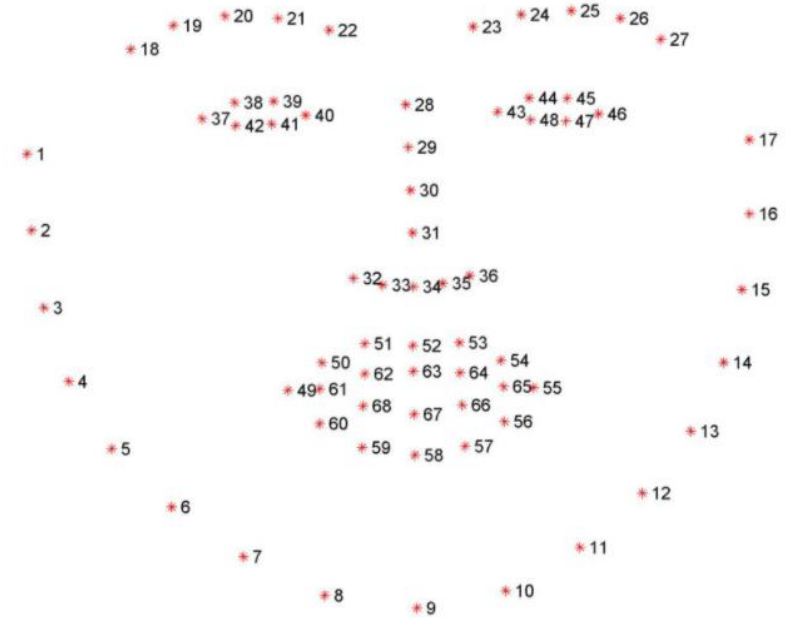
# 세부 기능



이미지 처리 및 기계 학습, 얼굴 인식에 특화된  
C++ 기반 고성능 라이브러리



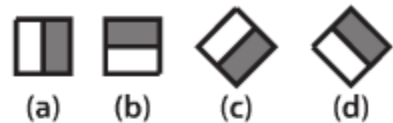
Haar Cascade를 이용한 얼굴 검출



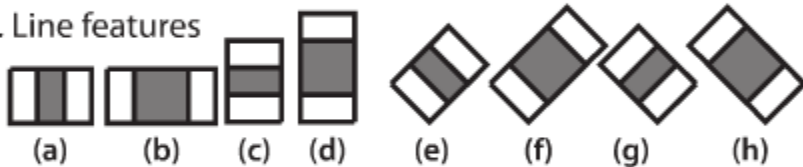
Face Landmark Detector를 이용한 Indexing

# 얼굴 인식 알고리즘 (Haar Cascade)

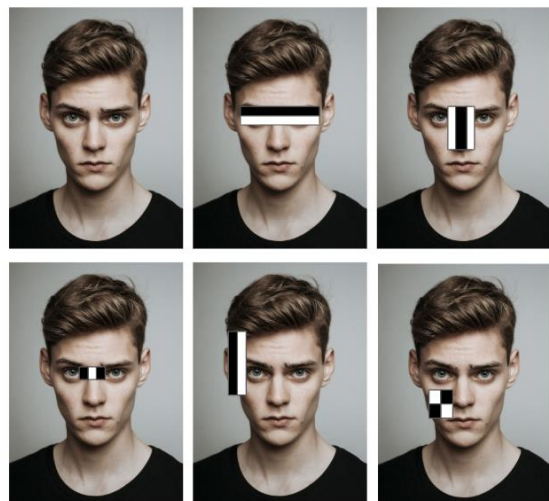
## 1. Edge features



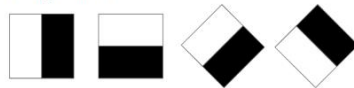
## 2. Line features



## 3. Center-surround features



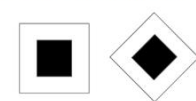
### • Edge Features



### • Line Features



### • Center-surround Features



### • Four-rectangle Features



- 긍정 이미지, 부정 이미지를 통해 얼굴의 특징을 추출할 수 있는 다양한 분류기(classifier) 학습
- 특징값 = 검정색 사각형 영역의 픽셀값의 합 - 하얀색 사각형에서의 픽셀값의 합
- 얼굴의 특징이 학습된 분류기를 영상에 적용, 단계적으로 적용 하면서 실패한 영역은 제외
- 모든 단계를 통과한 윈도우가 최종적으로 얼굴 영역이 됨



# 시선 인식 알고리즘

## 눈 영역 추출



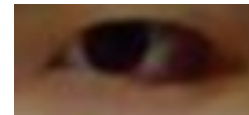
Face Landmark Detector

```
gaze_ratio_left_eye = get_gaze_ratio([36, 37, 38, 39, 40, 41], landmarks)
gaze_ratio_right_eye = get_gaze_ratio([42, 43, 44, 45, 46, 47], landmarks)
```

## 이진화



```
gray_eye = eye[min_y: max_y, min_x: max_x]
_, threshold_eye = cv2.threshold(gray_eye, 30, 255, cv2.THRESH_BINARY)
```



## 안구 영역 분할 및 흰자 비율 계산

```
left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]
left_side_white = cv2.countNonZero(left_side_threshold)

right_side_threshold = threshold_eye[0: height, int(width / 2): width]
right_side_white = cv2.countNonZero(right_side_threshold)
```

## 시선의 위치를 파악할 수 있는 변수 'gaze\_ratio' 계산

```
if left_side_white == 0:
    gaze_ratio = 1
elif right_side_white == 0:
    gaze_ratio = 5
else:
    gaze_ratio = left_side_white / right_side_white
return gaze_ratio
```



# 집중도 계산

```


if time_flag and time_count == 21:
    face_frequency.append(len(faces))
    time_count = 0
    if face_frequency[-1] == 0:
        concentration -= 1.75
        if concentration <= 0:
            concentration = 0
    else:
        concentration += 0.5
        if concentration >= 100:
            concentration = 100
    con_list.append(concentration)
if 50 <= concentration <= 100:
    if concentration == 100:
        concentration = 100
    red = (100 - concentration) * 5
    green = 128 + (100 - concentration)
elif concentration < 50:
    if concentration == 0:
        concentration = 0
    red = 250
    green = concentration * 3

```

```

if gaze_ratio <= 0.9:
    cv2.putText(frame, "RIGHT", (50, 100), font, 2, (0, 0, 255), 3)
    if time_count2 == 21:
        concentration -= 0.1
        time_count2 = 0
elif 0.9 < gaze_ratio < 2.0:
    cv2.putText(frame, "CENTER", (50, 100), font, 2, (0, 0, 255), 3)
    if time_count2 == 21:
        concentration += 0.3
        time_count2 = 0
else:
    cv2.putText(frame, "LEFT", (50, 100), font, 2, (0, 0, 255), 3)
    if time_count2 == 21:
        concentration -= 0.1
        time_count2 = 0

```

- 개발 환경의 frame에 맞춰 1초 단위로 집중도를 계산하도록 구현
- 얼굴이 검출되지 않았을 때 집중도의 하락폭을 가장 크게 하였음
- 얼굴이 검출되었더라도 시선이 오른쪽, 왼쪽을 향하는 경우 집중도 하락
- 집중도 수치를 색깔별로 시각화 
- 계산된 집중도는 리스트에 초단위로 저장

# 데이터 분석

## - 평균 집중도

리스트에 집중도를 초단위로 저장한 후 계산

```
def cal_con_average(c_list):
    sum = 0
    for i in c_list:
        sum += i
    average = sum / len(c_list)
    return average
```

## - 집중력이 가장 크게 떨어진 구간

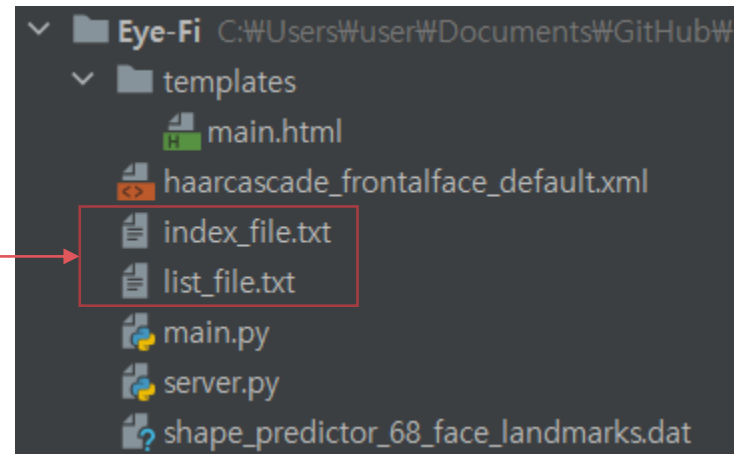
얼굴, 시선을 판단하여 집중력이  
연속으로 감소한 가장 긴 구간을  
계산하고 리스트에 저장

```
# calculate worst concentration period
for i in range(len(face_frequency) - 1):
    if face_frequency[i] == 0 and face_frequency[i + 1] == 0:
        if first_flag:
            worst_time_first = i
            first_flag = 0
        worst_cnt += 1
    else:
        if worst_max < worst_cnt:
            worst_max = worst_cnt
            worst_time_first_final = worst_time_first
            worst_time_second = i
        first_flag = 1
        worst_cnt = 0
```

# Flask 서버 연동 및 시각화

## - 프로세스간 데이터 통신

독립된 프로세스 사이의 데이터 통신을 위해  
파일 입출력 시스템을 이용하여 데이터 시각화에  
필요한 데이터를 txt파일로 저장



```
with open('list_file.txt', 'w') as f:
    for con in con_list:
        f.write("%s\n" % con)

with open('index_file.txt', 'w') as f:
    for index in index_list:
        f.write("%s\n" % index)
```

< 파일에 쓰기 (main.py) >

```
# index_file 내용 가져오기
with open('index_file.txt', 'r') as f:
    index_file = open('index_file.txt', 'r').read().split('\n')

for i in range(len(index_file) - 1):
    index_file[i] = int(index_file[i])
index_file.pop() # 공백 제거
index_list = index_file
```

< 파일 내용 읽기 (server.py) >

```
<html lang="en">
<head>
<link href="https://fonts.googleapis.com/css2?family=Jua&display=swap" rel="stylesheet">
<title>집중도 분석 결과</title>
<meta charset="UTF-8">
<style>
h1 {
font-family: 'Jua', sans-serif;
font-size: 60px;
}
h2 {
font-family: 'Jua', sans-serif;
font-size: 40px;
}
</style>
</head>
<body>
<center>
<h1>집중도 분석 결과</h1>
<h2>서론과 학생의 평균 집중도 : {{ avg_concentration }} %
<emoj style="font-size:50px">{{ emoji }}</emoj>
</h2>
<br/>
<h2>핵심 학생의 집중력이 가장 낮았던 시점은 {{ start }}초부터 {{ end }}초입니다.</h2>
</center>
</body>
</html>
```



+

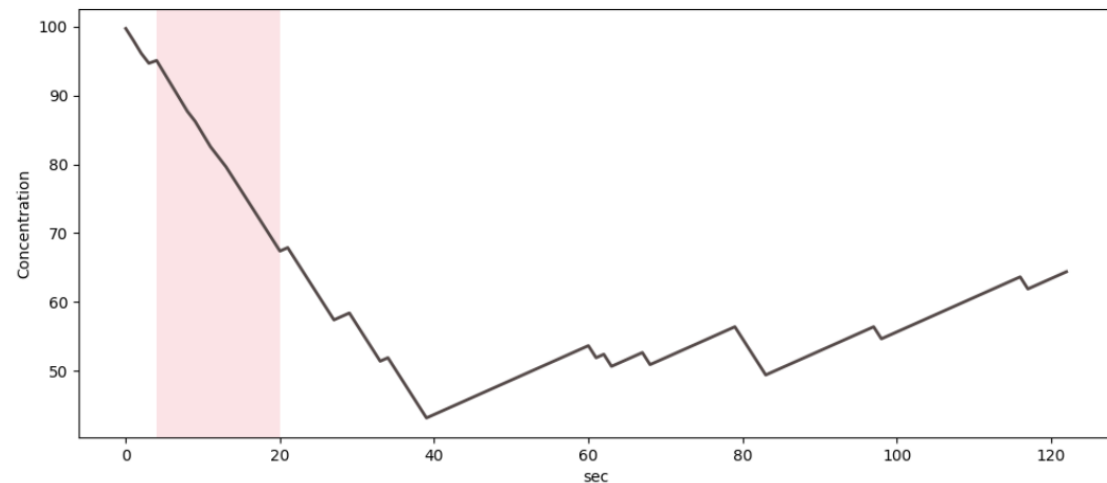


\_\_\_\_\_  
\_\_\_\_\_



## 집중도 분석 결과

서준교 학생의 평균 집중도 : 59.4077 % 🙄



**해당 학생의 집중력이 가장 낮았던 시점은 4초부터 20초입니다.**



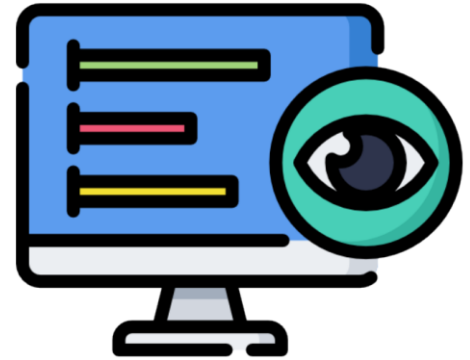
# 05

## 기대 효과 & 개선 방안

- ◆ 기대 효과
- ◆ 개선 방안

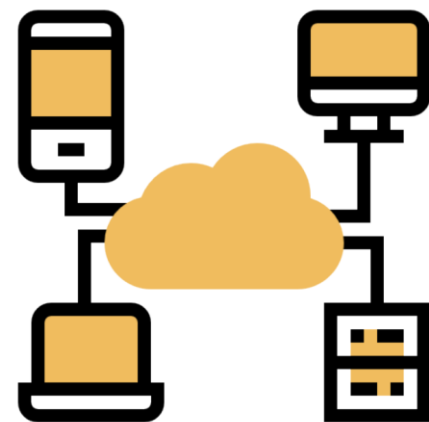
## 기대 효과

- 강사가 학생들의 집중도를 한 눈에 파악할 수 있기  
때문에 원격 환경에서도 학생 지도를 용이하게 할 수 있다.
- 집중도가 가장 크게 하락한 부분을 강사가 파악할 수 있도록  
하여 해당 시간대의 강의 주제에 대한 자료 보충 및 쉬는 시간  
조율 등 수업의 질을 높이는 피드백 효과를 기대할 수 있다.



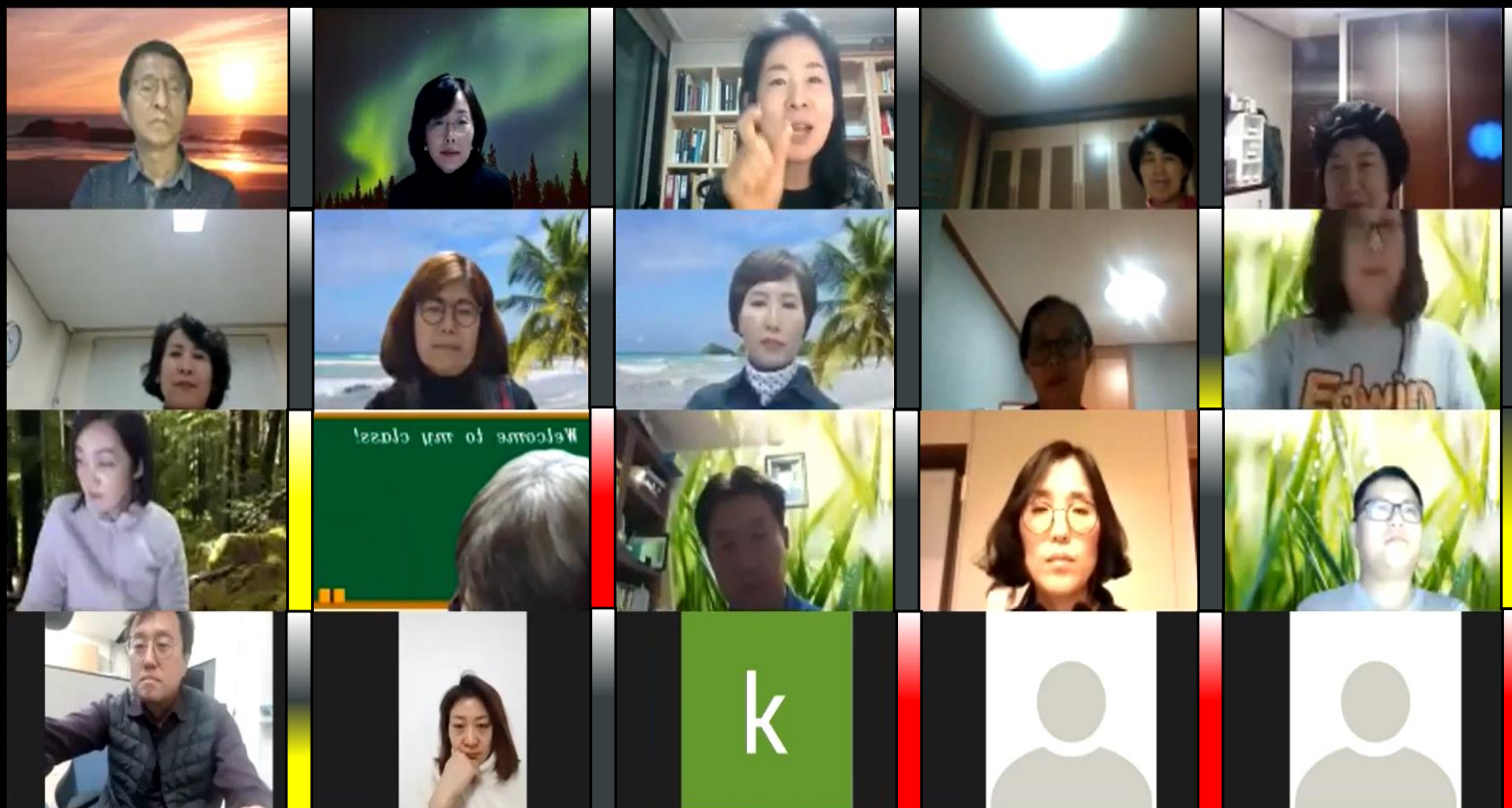
## 개선 방안

- 본 프로젝트는 개인의 데이터로 한정되었지만, 다중 접속이 가능한 서버를 구현하여 대량의 데이터 수집이 가능해지면 인공지능을 통해 더 폭넓은 분석이 가능해진다.

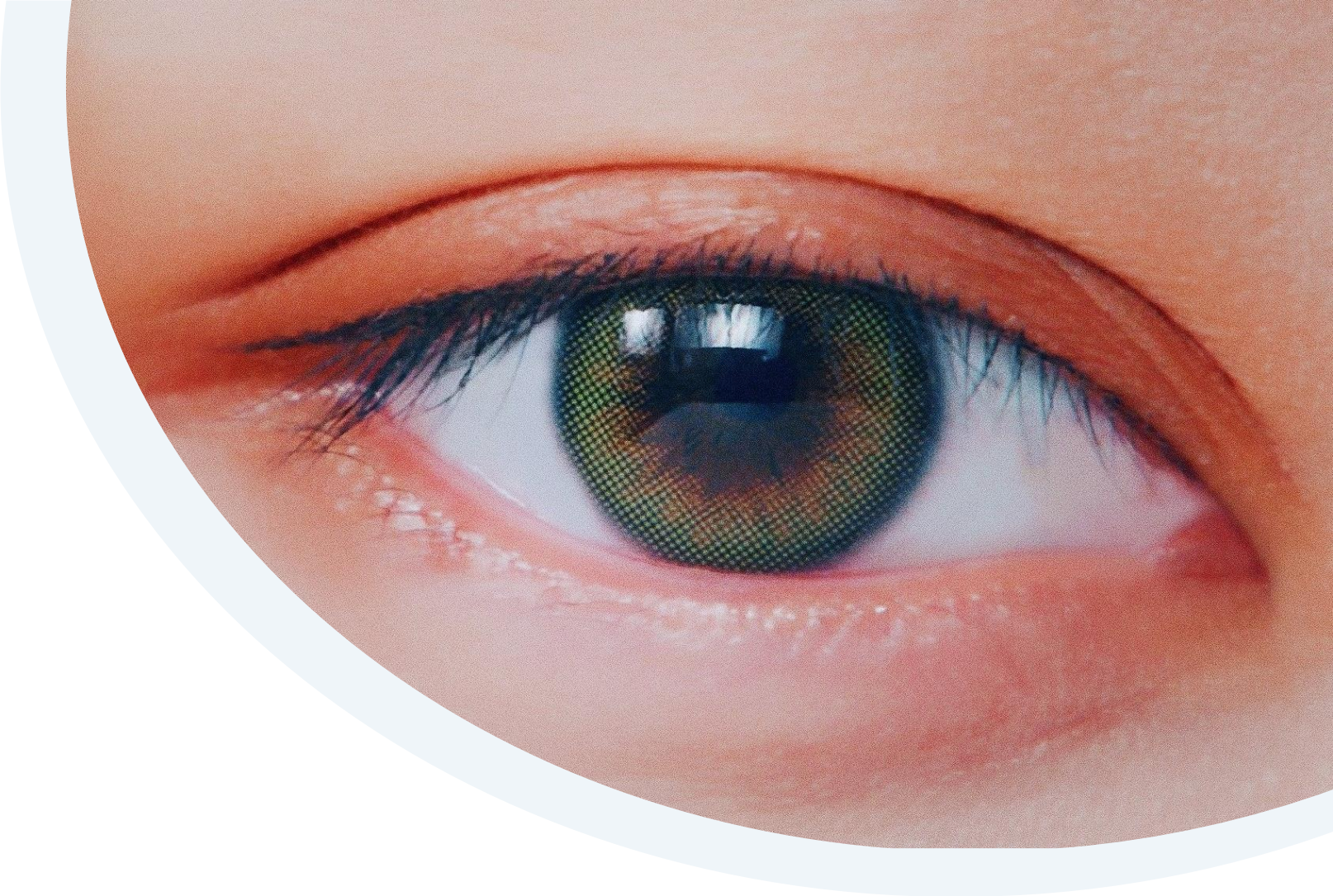


- 조명, 눈의 모양 등 여러 요인에 따라 인식률이 변화하기 때문에 외부 환경의 영향을 최소화할 수 있는 robust한 알고리즘으로 개선되면 신뢰성 있는 데이터 확보가 가능하다.









감사합니다

Thanks for watching.