
Final Report

손동작 인식과 얼굴 인식으로 구성된 2차 문서 보안

Secom달콤 (4조)

2018112127 김기현

2020112087 우신영

2018112095 유병민

2018112115 이승택



목차

1. 배경	4
1.1 문제 정의	4
1.1.1 스마트폰 보안 문제	4
1.1.2 기존 보안 방법(패스워드)의 취약함	5
1.1.3 모바일 보안 시장	5
1.1.4 문서 보안 시장	6
1.2 해결 방안	7
1.2.1 디바이스 분리와 2단계 인증	7
1.2.2 모션 인식과 얼굴 인식	7
2. 구현	9
2.1 구현방법	9
2.1.1 손동작 인식	9
2.1.2 얼굴 인식	12
2.1.3 FrontEnd 1 – 손동작 인식 애니메이션 설정	16
2.1.4 BackEnd 1 – 데이터베이스 생성 및 서버 연결	18
2.1.5 FrontEnd 2 – 회원가입 및 로그인	23
2.1.6 BackEnd 2 – 회원가입 및 로그인	28
2.1.7 FrontEnd 3 – 문서 업로드 및 암호 설정	30
2.1.8 BackEnd 3 – 문서 업로드 및 암호 설정	34
2.1.9 FrontEnd 4 – 문서 확인 및 암호 해제	36
2.1.10 BackEnd 4 – 문서 확인 및 암호 해제	42
2.2 구현 도구	44
2.1.1. 개발 환경	오류! 책갈피가 정의되어 있지 않습니다.
2.1.2. 손동작 인식	오류! 책갈피가 정의되어 있지 않습니다.
2.1.3. 얼굴 인식	45
2.1.4. FrontEnd	46
2.1.5. BackEnd	47
3. 결과	48
3.1. 결과물 시연	48
3.2. 결과물 분석	48

3.2.1.	성능 평가	48
3.2.2.	설계와의 비교	57
3.3.	제약사항.....	59
3.3.1.	손동작 인식	59
3.3.2.	얼굴 인식	60
3.3.3.	FrontEnd	61
3.3.4.	BackEnd	61
4.	작업 진행 방법	62
4.1.	작업 분담 구조.....	62
4.1.1.	선형 책임 도표	62
4.2.	설계 일정 및 역할 분담	64
4.2.1.	간트 차트	64
5.	참고문헌	64

1. 배경

1.1 문제 정의

1.1.1 스마트폰 보안 문제

현대 사회를 살아가고 있는 이들 모두가 사용하고 있는 스마트폰에 대한 보안 위협이 더욱 거세질 것으로 전망된다. 최근 정부에서 디지털 신분증과 전자증명서 서비스를 확대하고 있으며 모바일에서 한글, 워드, PDF 등 파일 실행 문서를 열어보고 사용할 수 있는 개인 사용자와 개인 소유 스마트 기기들을 업무에 활용하는 BYOD(Bring Your Own Device) 정책을 채택하여, 직원 생산성과 효율성을 높이려는 기업의 행보는 해커의 이목을 이끌고 있다. Check Point Research에 따르면 모바일 장치에 대한 사이버 공격은 2018년부터 2019년 상반기까지 50%가 증가하였으며 Symantec에 따르면 모바일 장치 36개 중 1개에는 고위험 애플리케이션이 설치되어 있다고 한다. 한국인터넷진흥원(KISA)에 보고돼 분석 절차에 들어간 악성 앱 수는 2015년 1665건, 2016년 1635건, 2017년 3023건, 2018년 4039건으로 집계됐다.



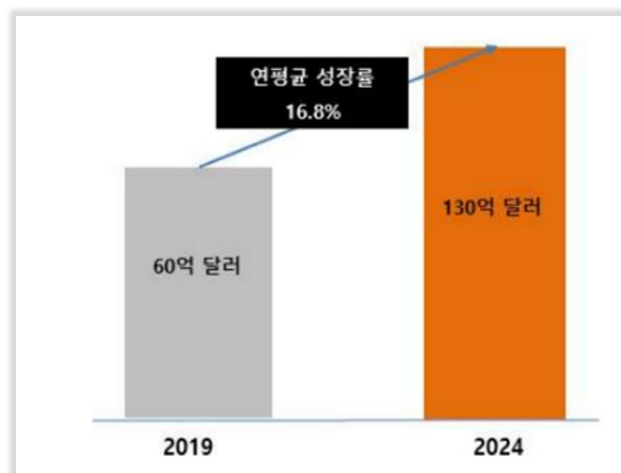
스마트폰 보안 문제 1: 급증하고 있는 스마트폰 악성 앱

또한 스마트폰은 수많은 연락처와 이메일 정보, 사진, 동영상 등 개인정보를 저장하고 있다. 카카오톡, 텔레그램, 인스타그램, 트위터 등 각종 SNS 활동을 모바일에서도 활용할 수 있는 만큼 계정탈취를 노린 모바일 보안위협도 역시 존재한다. 보안 전문 기업 이스트시큐리티는 2022년 주요 사이버 위협 동향 결산 및 회고 TOP 5와 2023년 사이버 위협 전망 TOP 5'에서 디지털 신분증 및 전자 문서 서비스의 보편화로 인한 개인정보 탈취 공격을 꼽은 바 있다.

1.1.2 기존 보안 방법(패스워드)의 취약함

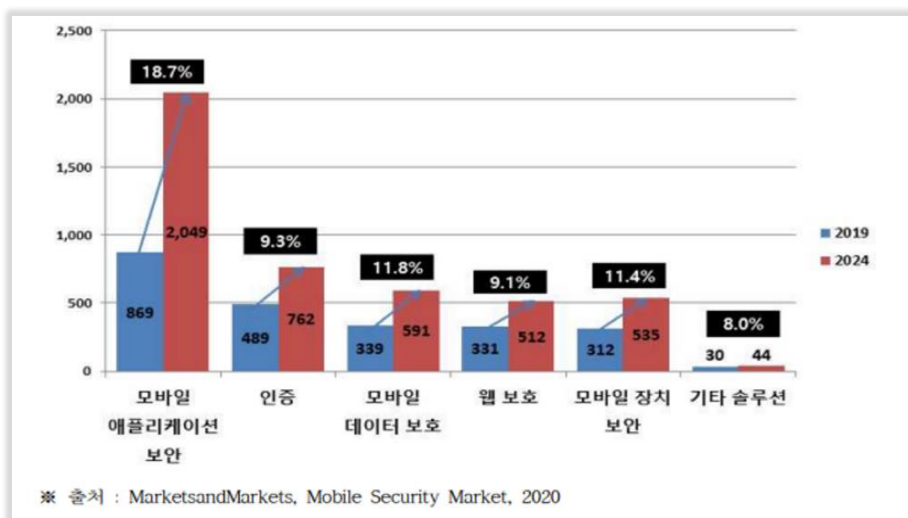
기존 비밀번호는 대·소문자, 숫자, 특수문자 등을 조합하여 사용하고 있다. 하지만 대부분의 사용자가 이용에 어려움을 느껴 단순한 문구 뒤에 숫자와 느낌표 정도만 추가하여 비밀번호를 만드는 현실이다. 다크웹 등에서는 유출된 수많은 ID와 비밀번호가 거래되고 있는데, 해커는 그렇게 입수한 ID와 비밀번호 조합으로 수많은 서비스에 로그인을 시도한다. 일명 '크리덴셜 스테핑' 공격이다. 사용자가 다른 서비스에서도 동일하거나 비슷한 ID와 비밀번호를 사용한다면, 한 서비스의 ID와 비밀번호만 노출되었을 뿐인데 다른 서비스들도 줄지어 해킹당할 수 있다.

1.1.3 모바일 보안 시장



모바일 보안 시장 1: 글로벌 모바일 보안 시장 규모 및 전망

전 세계 모바일 보안 시장은 연평균 성장률 16.8%로 증가했다. 2019년에 60억 달러였던 것에 비해 2024년에는 130억 달러에 이를 것으로 전망되고 있다.



모바일 보안 시장 2: 글로벌 모바일 보안 시장의 운영체제별 시장 규모 및 전망

모바일 보안 시장은 기업용 솔루션에 따라 모바일 애플리케이션 보안, 인증, 모바일 데이터 보호, 웹 보호, 모바일 장치 보안, 기타 솔루션 등으로 분류되며, 모바일 애플리케이션 보안은 2019년을 기준으로 가장 높은 36.7%의 점유율을 가지고 있었다. 또한, 인증이 20.6%, 모바일 데이터 보호가 14.3%, 웹 보호가 14.0%, 모바일 장치 보안이 13.2%, 기타 솔루션이 1.3%의 점유율을 가지고 있었다..

1.1.4 문서 보안 시장

개인 공간에서 업무를 보는 재택근무 혹은 원격근무를 적극적으로 시행하고 있는 만큼 기밀문서나 주요 문서에 대한 관리로 등장한 것이 DRM(Digital Rights Management, 문서암호화)과 DLP(Data Loss Prevention, 문서 유출 방지), 문서중앙화 등의 '문서보안 솔루션' 이다. 문서보안 솔루션의 개념은 업무를 위해 사용하는 문서에 대해 기업 내부에서만 열람이 가능하게 하며, 인가된 사용자만이 열고 수정할 수 있게 한다. 또한 파일이 외부로 유출된다고 하더라도 열리지 않게 하거나 문서를 암호화 시켜 내용을 볼 수 없게 한다.

점차 PC와 대등해지고 있는 하드웨어 성능을 보이는 스마트폰은 미래 스마트워크 환경에서 PC의 보조기기가 아니라 IT환경의 중심이 될 것이라는 전망을 보이고 있다. 그에 맞춰 업무용 PC에서 작업하던 보안문서를 스마트폰 내에서도 동일하게 열람, 편집, 저장이 가능한 기술들이 소개되고 있다.

하지만 스마트폰의 뛰어난 휴대성에 가려진 기기 도난과 분실이 정보유출로, 다양한 분야에서 모바일 문서보안 수요가 늘어날 것으로 보인다. 소프트캠프 모바일 문서 보안 솔루션의 경우에는, 모바일오피스 환경 구현 시 메일이나 시스템을 통해 공유되는 암호화된 사내보안문서를 사용자 스마트폰에서 직접 인증 받도록 한다. 인증된 사용자에게 한해서만 문서 열람 권한을 부여해 문서보안을 실행하는 것이다. 분실 시에도 인증 받지 않은 사용자는 데이터 열람이 불가능해 문서의 보안을 유지한다.

1.2 해결 방안

1.2.1 디바이스 분리와 2단계 인증

디바이스 분리의 장점

비밀번호 외에 추가적인 방법으로 본인을 인증하는 방식이 2단계 인증이라고 한다. 구글의 경우, 계정 로그인에 2단계 인증을 사용한다. 로그인을 시도할 경우, 사용자가 사전에 등록해둔 스마트폰이나 이메일로 인증번호를 발송하고, 그 인증번호를 입력해야 로그인을 할 수 있는 방식이다. 구글 계정 로그인의 사례처럼 컴퓨터와 스마트폰을 모두 사용한 보안의 경우, 단순히 비밀번호만을 사용한 보안보다 훨씬 강력한 보안을 제공한다. 최근에는 스마트폰 앱을 통해 지문이나 QR코드 촬영 등으로 본인임을 인증하는 방식까지 적용하고 있어 보다 더 간편하면서도 안전하게 로그인 하는 것이 가능하다.

2단계 보안의 필요성

사용자에게 서비스를 제공하는 업체가 데이터 유출을 방지하고 사용자 데이터를 보호하기 위한 모든 수단과 조치를 강구하고 있다. 하지만 데이터 유출 사고는 그래도 발생하고 있는 것이 오늘날이다. 개인정보를 보호하고, 잠재적인 보안 노출을 최대한 줄이기 위해 보안 단계를 추가하는 것은 사용자의 몫이다. 그 중 가장 효과적인 방법이 바로 '2단계 인증'이다.

2단계 인증의 대표적인 장점은, 해킹 시도에 대한 알림을 받을 수 있다는 점이다. 2단계 인증의 설정을 통해, 사용자 계정에 대한 허가 받지 않는 접근 시도 정황을 알 수 있다. 해커의 해킹 시도를 알 수 있다는 것 자체가 보안이 강하다는 것을 의미한다.

1.2.2 모션 인식과 얼굴 인식

앞서 언급한 비밀번호의 문제점을 해결하기 위해 비밀번호 대신 모션 인식 방법과 생체 기반 인증 방식을 고안했다.

모션 인식

손의 움직임만을 이용하여 잠금을 해제하는 보안장치를 만들어, 기존의 비밀번호를 사용한 보안의 단점을 보완하는 인증 방식을 구상했다.

얼굴 인식

현재의 얼굴 인식 기술은 정면 얼굴일 경우 95% 이상 인식이 가능하며, 높은 인식률을 기반으로 다양한 국가, 기관에서는 보안매체로 얼굴인식 기술이 활용되고 있다.

본 프로젝트에서는 보다 높은 보안을 제공하기 위하여, 2가지 단계의 인증 단계를 실행한다. 또한 각각의 보안은 다른 디바이스에서 실행하여, 디바이스 분리와 2단계 인증을 모두 사용하는 강력한 문서 보안 서비스를 제공한다.

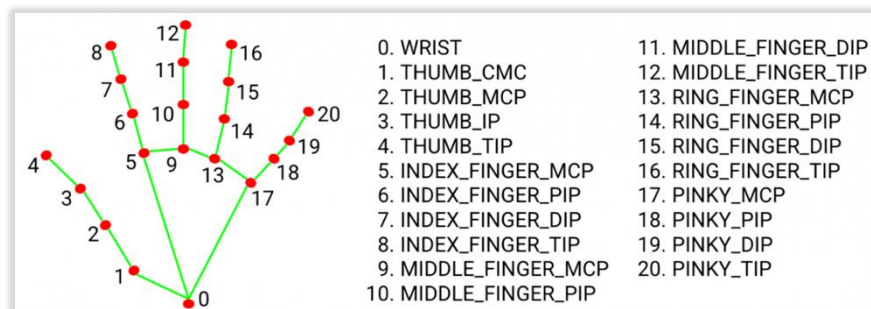
2. 구현

2.1 구현방법

2.1.1 손동작 인식

① 암호 손동작 설정

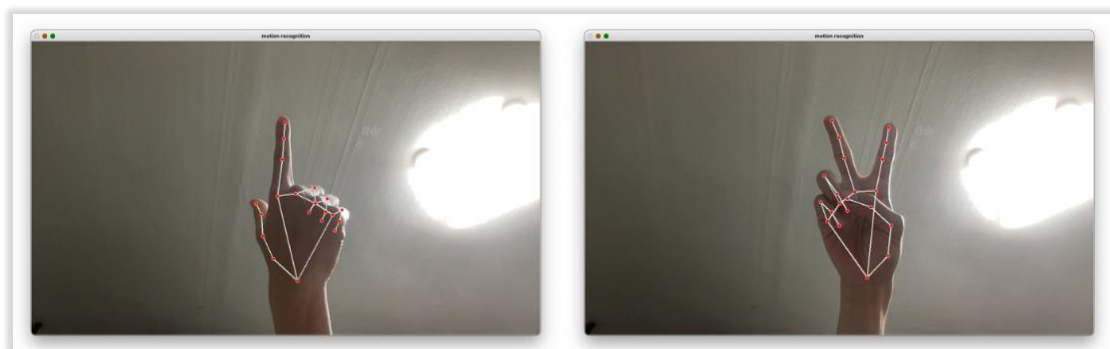
암호 손동작 설정을 하기 위해 웹캠 이미지를 일정 시간동안 실시간으로 받아온다. 그리고 받은 이미지에서 손을 인식해 정보를 계속해서 저장한다. 손의 flow를 추출해 저장하는 것이다.



암호 손동작 설정 1 : Mediapipe Hands가 예측하는 21가지 관절들

손을 인식할 때는 Mediapipe Hands 라이브러리를 사용한다. Mediapipe Hands는 구글에서 개발한 오픈소스 라이브러리이며, 높은 정확도로 손을 인식할 수 있어 손동작 분류 및 추적 애플리케이션에 많이 사용되고 있다.

Mediapipe Hands는 Convolutional Neural Network (CNN) 기반의 딥러닝 알고리즘을 사용해 손의 위치를 찾는다. 그리고 keypoint regression 모델을 적용해 손의 21가지 관절 위치를 예측한다. 관절들 간의 상대적인 위치를 예측하며, 이렇게 예측한 관절들의 위치를 이미지 평면 상의 좌표와 3차원 위치를 반환한다. 그리고 관절마다 visibility(가시성) 값을 가지며, 이미지에서 해당 관절이 얼마나 잘 보이는지 나타내는 지표이다. 가시성 값은 관절들의 신뢰도를 결정하는 지표이다. 이미지 상에서 잘 보이지 않는 관절의 경우, 해당 관절에 낮은 신뢰도를 부여하는 것이다. 3차원 위치 정보와 신뢰도를 고려하며 손동작 인식의 모든 과정을 진행한다.



암호 손동작 설정 2: 예측한 관절들을 표시하게 했을 때

암호 손동작을 설정하기 위해, 20초 동안 실시간 웹캠 이미지를 받아온다. 그리고 손 관절들의 3차원 위치와 visibility 데이터를 계속해서 배열에 저장한다.

20초가 지나면 웹캠 이미지를 더 이상 받아오지 않고, 그동안 저장한 데이터를 처리하기 시작한다. 3차원 위치와 visibility 정보를 저장한 numpy 형식의 시퀀스 데이터로 만든다. 한 순간의 손 모습이 아닌, 손의 움직임을 암호로 사용하기 위함이다. 따라서 시간의 흐름에 따른 손 동작의 데이터 정보를 담는다. 저장할 시퀀스의 길이를 90으로 설정하여 연속된 90개의 프레임 정보(1초당 30프레임을 가정 했을 때, 약 3초 정도의 프레임)를 담는다. 데이터를 저장하고, 데이터의 shape을 통해 시퀀스 개수가 충분하면 true값을, 충분하지 않다면 false를 서버로 전달한다.

② 손동작 유사도 측정을 위한 모델 생성

암호 손동작 설정이 끝나면, 유사도 검사를 위한 모델을 생성하기 위한 학습을 진행한다. 이 단계는 데이터 불러오기, 데이터 전처리, 모델 구성, 모델 학습 단계로 나눌 수 있다.

- 데이터 불러오기

서버로부터 매개변수로 전달받은 문서 이름을 활용해, 손동작 암호 설정단계에서 생성한 시퀀스 데이터들을 불러온다. 불러온 시퀀스 데이터들을 하나의 데이터셋으로 만든다.

- 데이터 전처리

데이터셋을 x_data와 y_data로 나눈다. x_data에는 데이터셋의 첫번째 값인 라벨을 제외한 나머지 데이터를 저장하고, y_data에는 데이터셋의 첫번째 값인 라벨을 one-hot encoding한 결과를 저장한다.

One-hot encoding은 각각의 라벨을 모델이 이해할 수 있도록 고유한 벡터로 변환하는 기법이다. 예를 들어 'first', 'second', 'third'를 one-hot encoding하면, 'first'는 [1,0,0]으로, 'second'는 [0,1,0], 'third'는 [0,0,1]로 변환된다.

마지막으로, train_test_split 함수를 이용해 전체 데이터를 train set과 validation set으로 나눈다.

- 모델 구성

LSTM(Long Short-Term Memory) 기반의 신경망 모델을 구성한다. LSTM은 RNN(Recurrent Neural Network)의 한 종류로, 시간에 따른 의존성을 가진 데이터를 모델링하고 학습하는데 효과적이다. 본 프로젝트에서는 시간에 따라 연속적인 시퀀스 데이터를 다루기 때문에, LSTM을 활용한다.

LSTM 레이어와 Dense 레이어를 순차적으로 쌓아 Sequential 모델을 생성한다. Sequential 모델은 Keras에서 제공하는 모델 구성 방법 중 하나로, 각 레이어를 순차적으로 쌓아 구성하는 방식이다. 추가된 레이어가 이전 레이어의 출력값을 입력값으로 받는다.

Dense 레이어는 입력 레이어에서 들어온 값에 가중치를 적용한 결과값을 출력으로 내보내는 레이어로, 분류 문제에서 주로 사용된다.

- 모델 학습

Fit 함수를 이용해 앞서 구성한 모델을 학습시킨다. 콜 백 함수를 사용해 검증 데이터의 정확도를 모니터링하고, 가장 성능이 좋은 모델을 저장한다.

③ 손동작 유사도 측정

손동작 암호 설정 단계와 유사하게, 웹캠 이미지를 실시간으로 받아온다. 유저의 손동작을 인식하고, 앞서 학습한 모델을 사용해 암호로 설정한 손동작과의 유사도를 검사한다. 판단 로직은 다음과 같이 구현했다.

실시간으로 손 관절들의 좌표를 배열에 저장하며, 암호 손동작 설정 단계와 같이 90프레임 단위로 학습한 동작들과 유사한지 실시간으로 확인한다. 학습한 모델이 유저가 암호로 설정한 손동작 중 하나를 했다고 90% 이상의 confidence로 예측한 경우, predict_action 배열에 예측한 손동작의 이름(first, second, third 중 하나)을 넣는다. 이렇게 predict_action 배열의 최근 30개의 원소 중 25개의 원소가 같은 손동작일 경우, 해당 손동작을 했다고 판단하고 true 값을 서버에 전달한다. 30초 안에 해당 손동작을 했다고 판단하지 못하면 false값을 서버로 보낸다.

2.1.2 얼굴 인식

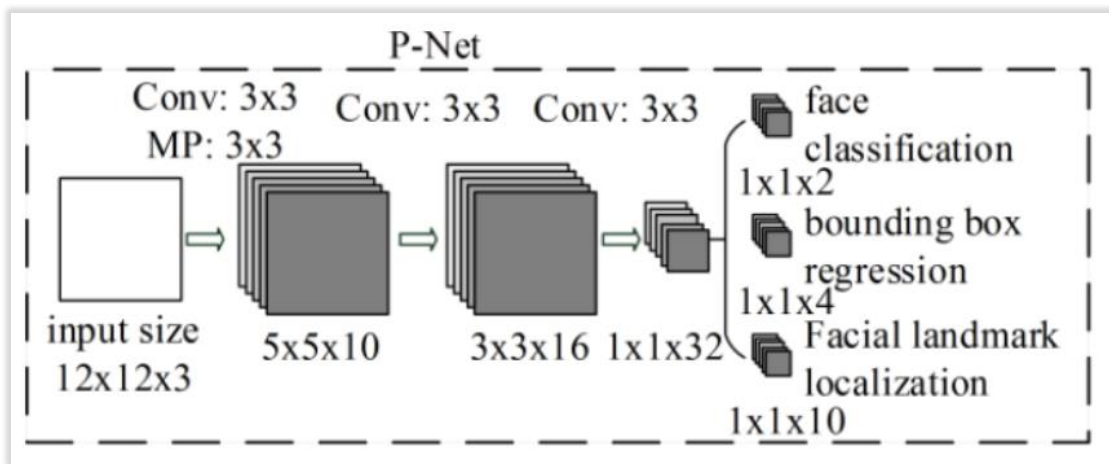
① 얼굴 인식 모델 구축

본 프로젝트의 얼굴 인식 모델은 CNN을 사용하였다. 얼굴 검출에 효과적인 MTCNN 알고리즘과 keras로 구현된 facnet 모델을 이용하여 얼굴 유사도를 판단하여 암호 해제 여부를 반환한다.

MTCNN(Multi-Task Cascaded Convolutional Networks)은 3개의 neural network로 이루어져 있는데, 각 net에서 진행되는 과정을 동시에 학습시키는 방식 (joint learning)을 사용하여, 다른 방법에 비해 정확도가 높다. keras의 CNN구조만을 사용하는 방법, vggface 혹은 siamese_model을 사용해 가중치 파일을 학습하는 방법, dlib라이브러리를 활용한 방법 등 다양한 방법으로 얼굴 인식을 구현할 수 있지만, 본 프로젝트에서는 그 중 가장 높은 정확도를 보인 MTCNN 모델을 선택했다.

MTCNN은 P-net, R-net, O-net의 3개로 분할된 CNN모델을 기반으로 한다. 각 net에서 face classification과 bounding box regression, face landmark localization 과정을 진행하면서 동시에 학습시키는 join learning 방식을 사용한다.

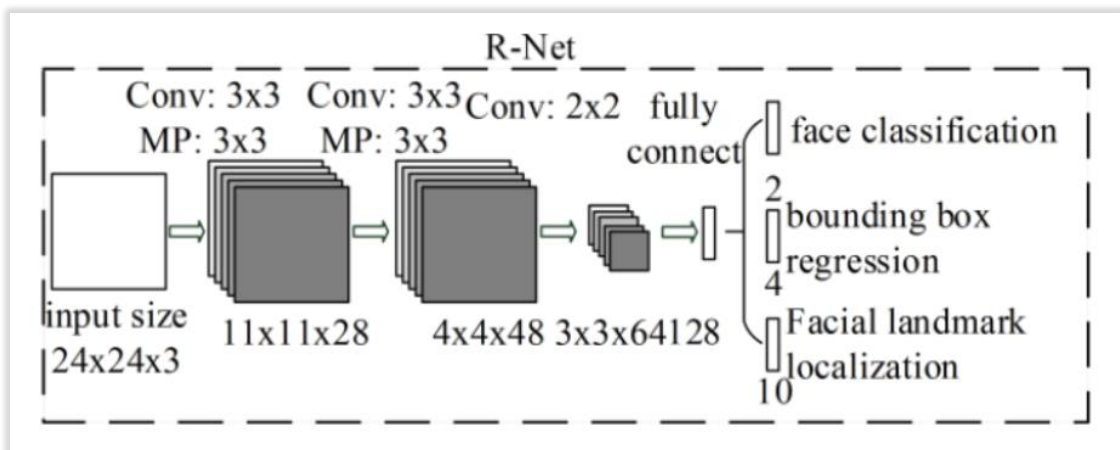
- P-Net(Proposal Network)



얼굴 인식 모델 구축 1 : P-Net

첫번째 P-Net에서는 이미지에서 얼굴을 찾아내는데 중점을 둔다. CNN의 일반적인 구조와 다르게 convolution layer로만 이루어진 것이 특징이다. P-Net에서는 초기 얼굴 후보를 생성하는 역할을 한다. 입력 이미지를 받으면 여러 scale로 변환한 후, 얼굴 유무를 예측한다. 크기가 다른 얼굴을 탐지할 수 있는 특징을 갖고 있으며, NMS(Non-Maximum Suppression) 알고리즘으로 정확도가 높은 후보만 남도록 추려낸다. 모든 초기 얼굴 후보들에 대해 bounding box regression, face landmark localization 값을 다음 단계로 보낸다.

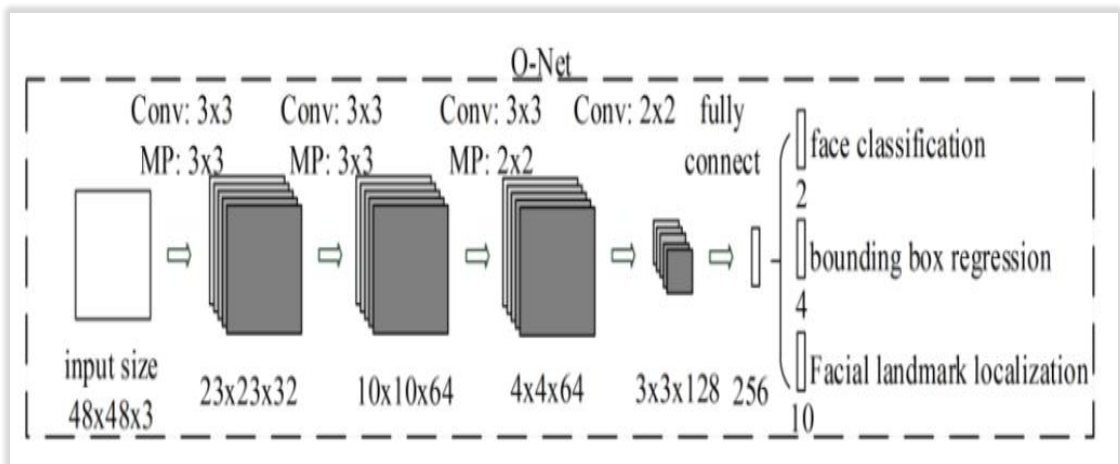
- R-Net(Refine NetWork)



얼굴 인식 모델 구축 2 : R-Net

R-Net은 P-Net에서 추려낸 초기 후보들 중 추려내는데 중점을 둔 단계이다. 구조는 P-Net과 비슷하지만, fully connected layer가 추가된다. 추가된 fully connected layer로 인해 좀 더 정확한 값을 추려낼 수 있다. P-Net에서와 마찬가지로 추려낸 얼굴 후보들의 face classification, bounding box regression, face landmark localization 값을 다음 단계로 보낸다.

- Net(Output Network)

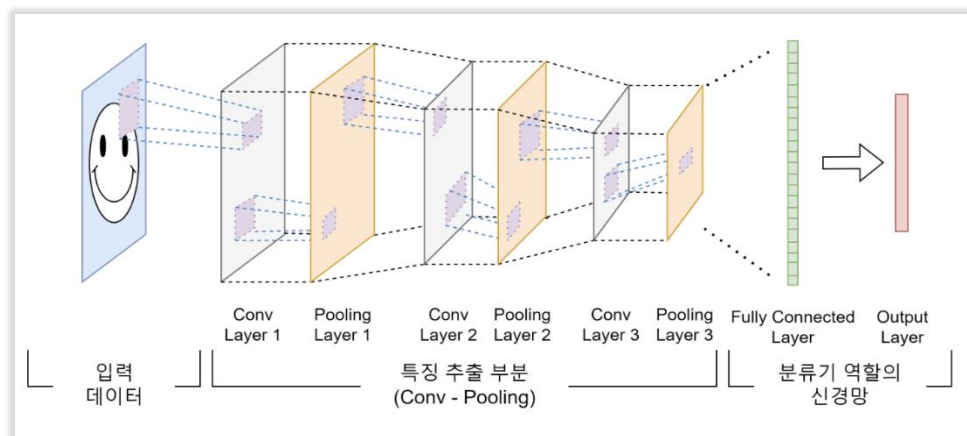


얼굴 인식 모델 구축 3 : O-Net

마지막 O-Net은 얼굴 후보를 최종적으로 검증하고 각 영역에서 face landmark를 찾아내는데 중점을 둔다. R-net에서 생성된 얼굴 후보를 입력으로 받아 얼굴 유무를 분류하고, bounding box의 정확도를 향상시키며 얼굴 특징 벡터를 추출한다. 얼굴 후보를 최종적으로 확정하는 단계이다.

MTCNN으로 얼굴 영역을 검출해내면, Keras의 Facenet 모델을 사용하여 얼굴 이미지의 특징을 검출해낸다. Facenet 모델은 얼굴 인식과 얼굴 임베딩을 위한 딥러닝 기반 알고리즘으로, 얼굴 이미지를 입력으로 받아 임베딩 벡터를 생성한다. 크게 세 가지 주요 구성 요소로 구성된다.

- CNN



얼굴 인식 모델 구축 4: CNN

얼굴 이미지에서 특징을 추출하기 위한 아키텍처로 사용한다. 여러 개의 Convolution 및 Pooling layer가 겹쳐져 있는 구조를 활용해 이미지를 입력 받아 얼굴 feature를 추출하는데 사용한다. Convolution layer에서는 입력으로 받은 이미지를 합성곱을 통해 결과 값을 산출해, 이미지의 크기를 설정해준다. Pooling Layer에서는 Convolution Layer에서 뽑아낸 결과값들을 특정 크기로 잘라내어 조절해주는 역할을 한다.

- Triplet Loss

얼굴 간의 유사성을 학습하기 위한 것으로, 얼굴 이미지의 얼굴 특징 벡터 간의 거리를 최소화하고 동일한 사람의 얼굴 특징 벡터는 가까워지도록 학습한다. 이를 통해 얼굴 특징 벡터의 유사성을 보장하고 얼굴 간의 거리를 구분한다.

- Embedding Vector

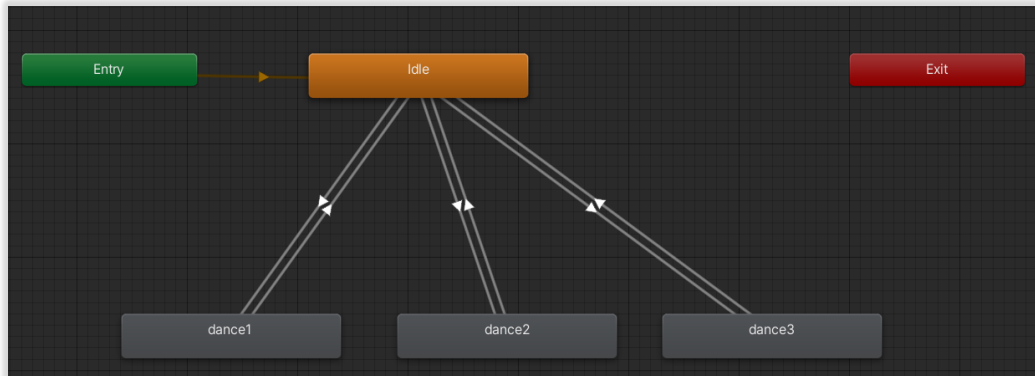
CNN을 통해 추출된 특징을 사용하여 얼굴을 임베딩하는 고차원 벡터를 생성한다. 이 벡터는 얼굴을 고유하게 나타내고, 얼굴 간의 유사성 및 차이를 계산하는데 사용된다.

② 얼굴 이미지 동일 판단

동일 여부를 판단할 두 사진을 각각 RGB 형식으로 변환하여 불러온다. 두 이미지를 각각 numpy 배열로 변환하여 변수에 저장한다. Numpy 배열에 저장된 값들을 모두 MTCNN 모델에 적용하여 각 이미지에서의 얼굴을 감지한다. 얼굴을 감지한 후, 각각 해당 얼굴 영역에서 얼굴의 위치와 크기를 불러오고, 추출하여 변수에 저장한다. facenet 모델을 사용하여 각 얼굴의 특징 벡터를 생성하고, 특징 벡터 간의 거리를 계산하여, 유사도를 판단한다. 유사도 similarity는 $1/(1 + \text{벡터 간의 거리})$ 로 계산한다. 일반적으로 벡터 간의 유클리드 거리가 작을수록 유사도가 높아지고, 두 개체의 거리가 0에 가까울수록 완전히 일치한다고 볼 수 있다. 해당 유사도를 임계값과 비교해 동일 인물인지 판단한다. 임계값이 높으면, 더 엄격한 일치 조건이 필요하여 인식이 떨어질 수 있고, 반대로 임계값이 낮으면 유연한 일치 조건이 적용되어 더 많은 얼굴을 동일하게 간주할 수 있다. 해당 모델에서는 임계값을 0.8로 설정했다. 임계값이 0.8보다 낮은 경우 동일 인물로 판단하는 것이다.

2.1.3 FrontEnd 1 – 손동작 인식 애니메이션 설정

① Animator 생성

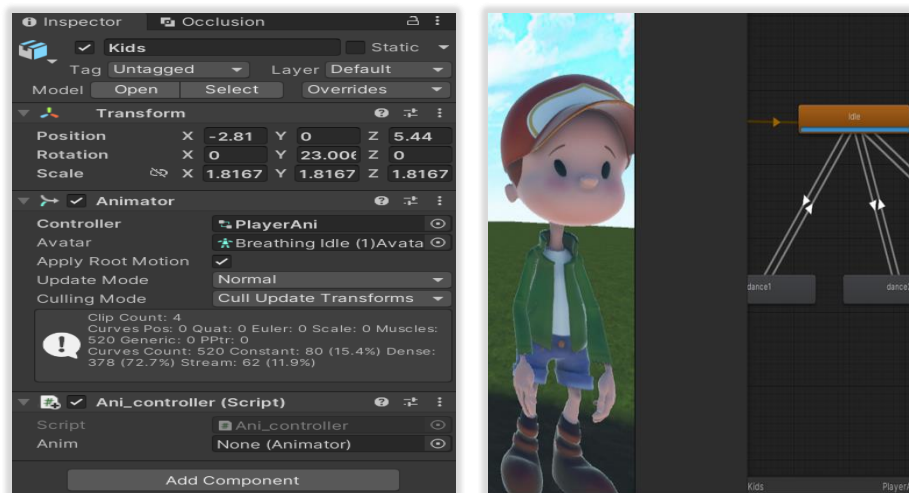


Animator 생성 1 : Animator를 통한 Animation Control

유니티의 Animator는 게임 오브젝트에 애니메이션을 적용하고, 상호작용 가능한 애니메이션을 만들 수 있는 도구이다. Animator는 상태 머신(State Machine)으로 구성되어 있으며, 애니메이션을 조작하는 데 필요한 모든 기능을 제공한다.

해당 Animator는 'PlayAni'의 명칭으로 생성된 Animator이다. Animator를 통해 저장된 애니메이션들에 대한 Parameters를 설정할 수 있으며 Parameters의 종류는 Float, Int, Bool, Trigger로 구성되어 있다. Parameters를 애니메이션 사이의 노드에 저장하면 다음 애니메이션으로 넘어가는 기준을 제시해준다. 본 프로젝트에선 모든 Animation의 Parameter를 Trigger로 설정하였으며 Trigger의 특징에 의해 return 값을 따로 설정하지 않아도 기존 형태인 애니메이션으로 돌아온다.

② Animator 적용

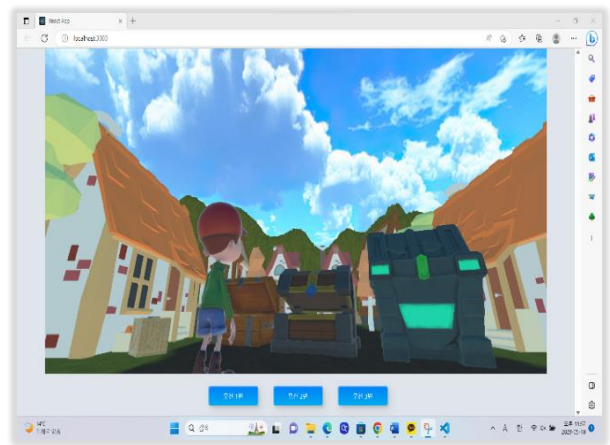
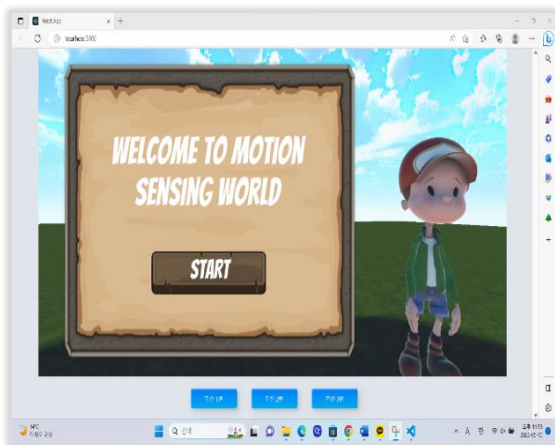


Animator 적용 1 : Animator 적용

위 사진은 해당 Animator를 3D Object에 적용한 모습이고, 3D 캐릭터를 구성하고 있는 상태창을 확인할 수 있다. Animator의 Controller는 'PlayerAni' , Avatar는 3D Object 객체를 설정한 것을 확인할 수 있다. 또한 애니메이션이 실행되면, 사진처럼 실행되는 애니메이션의 진행 여부를 파란색 게이지를 통해 확인할 수 있다.

③ 애니메이션을 Web 환경에서 열기

WebGL을 사용하여 React에서 Unity 함수를 제어하기 위해, Unity의 sendMessage 함수를 사용한다. sendMessage는 Unity 씬의 여러 오브젝트 간에 메시지를 전송할 수 있는 Unity의 기본 제공 함수이다. sendMessage를 사용하면, 스크립트 외부에서 Unity 스크립트의 함수를 호출할 수 있다. 이는 React에서 Unity 함수를 제어하기 위해 필요한 작업이다.



애니메이션을 Web 환경에서 열기 1 : WebGL을 통한 웹에서 유니티 구현

sendMessage를 통하여 웹페이지에서 유니티 애니메이션을 구현한 모습이다. 사용자는 제공되는 웹캠의 화면을 통해 본인의 손동작을 확인할 수 있다. 동시에 유니티 애니메이션이 제공되어 사용자에게 가이드를 제공하며, 사용자는 지루함을 느끼지 않고 시각적으로 과정을 살펴볼 수 있다. 유니티 애니메이션은 암호 손동작이 저장되었을 때를 변수로 하여 사용자에게 다음 준비 단계를 안내하는 역할을 수행하고 있다. 다음 준비 단계는 손동작 암호 설정에 성공하여 서버에 저장되었을 때와 아닐 때를 나누어 구분한다.

2.1.4 BackEnd 1 – 데이터베이스 생성 및 서버 연결

① 데이터베이스 생성

본 프로젝트에서 사용할 Schema인 capstonedesign_db를 MySQL에 생성한다. 해당 Schema에 프로젝트에서 사용할 데이터를 저장하는 테이블을 추가한다.

account table

account table은 사용자가 본 어플리케이션을 이용하기 위해 로그인을 할 때 사용하는 정보를 저장한다. 각각의 Column에 저장하는 정보는 다음과 같다.

- id

사용자를 고유하게 식별하기 위한 기본 키다. 이 값은 자동으로 증가하는 "auto_increment"로 설정되어 있어, 사용자가 추가될 때마다 자동으로 증가한다.

- account_password

사용자가 회원가입시에 입력한 비밀번호 정보이다. 해당 값은 서버에서 암호화한 뒤, 암호화한 값으로 저장된다.

- account_name

사용자가 회원가입시에 입력한 이름 정보이다.

- account_created_at

사용자가 회원가입을 완료한 시간에 대한 정보이다.

- account_email

사용자가 회원가입시에 입력한 이메일 정보이다. 로그인은 해당 이메일 정보와 비밀번호를 확인하여 여부를 판단한다.

- account_role

사용자가 일반 사용자인지, 관리자인지 구분하기 위한 정보이다.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	account_password	varchar(255)	NO		NULL	
	account_name	varchar(255)	NO		NULL	
	account_created_at	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
	account_email	varchar(255)	NO		NULL	
	account_role	varchar(45)	YES		NULL	

데이터베이스 생성 1: 생성된 account table

document table

document table은 사용자가 업로드한 문서에 대한 정보를 저장한다. 각각의 Column에 저장하는 정보는 다음과 같다.

- id

Primary Key로 문서를 구분하기 위한 정보이다. 이 값은 자동으로 증가하는 “auto_increment”로 설정되어 문서가 추가될 때마다 자동으로 증가한다.

- document_name

문서가 서버에 저장되는 이름이다. 같은 이름으로 업로드 시에 중복되는 것을 방지하기 위해 임의로 설정한 문자열을 사용한다.

- document_url

문서가 서버에 저장되는 경로이다.

- document_created_at

문서가 업로드 된 시간에 대한 정보이다.

- document_original_name

문서의 기존 이름으로, 사용자에게 문서를 제공할 때 기존에 업로드했던 이름대로 제공하기 위한 정보이다.

- account_id_document

Foreign Key로 문서를 어떤 사용자가 업로드했는지 구분하는 정보

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	document_name	varchar(255)	NO		NULL	
	document_url	varchar(255)	NO		NULL	
	document_created_at	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
	document_original_name	varchar(45)	NO		NULL	
	account_id_document	int	NO	MUL	NULL	

데이터베이스 생성 2: 생성된 document table

motion_password table

motion_password 은 사용자가 각 문서에 대해 설정한 손동작 암호 정보를 저장한다. 각각의 Column에 저장하는 정보는 다음과 같다.

- **id**

Primary Key로 손동작 암호를 구분하기 위한 정보이다. 이 값은 자동으로 증가하는 “auto_increment”로 설정되어 손동작 암호가 추가될 때마다 자동으로 증가한다.

- **motion_password_url**

사용자가 설정한 손동작을 학습한 모델이 저장된 경로다.

- **account_id_motion_password**

Foreign Key로 문서를 어떤 사용자가 업로드했는지 구분하는 정보이다.

- **document_id_motion_password**

Foreign Key로 손동작 암호를 어떤 문서에 대하여 설정하였는지 구분하는 정보이다.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	motion_password_url	varchar(255)	NO		NULL	
	account_id_motion_password	int	NO		NULL	
	document_id_motion_password	int	NO	MUL	NULL	

데이터베이스 생성 3 : 생성된 motion_password table

image_password table

image_password table은 사용자가 각 문서에 대해 설정한 얼굴 암호 정보를 저장한다. 각각의 Column에 저장하는 정보는 다음과 같다.

- Id

Primary Key로 얼굴 암호를 구분하기 위한 정보이다. 이 값은 자동으로 증가하는 “auto_increment”로 설정되어 얼굴 암호가 추가될 때마다 자동으로 증가한다.

- image_password_url

사용자가 업로드한 얼굴 암호(사진)를 서버에 저장한 경로이다.

- account_id_image_password

Foreign Key로 문서를 어떤 사용자가 업로드했는지 구분하는 정보이다.

- document_id_image_password

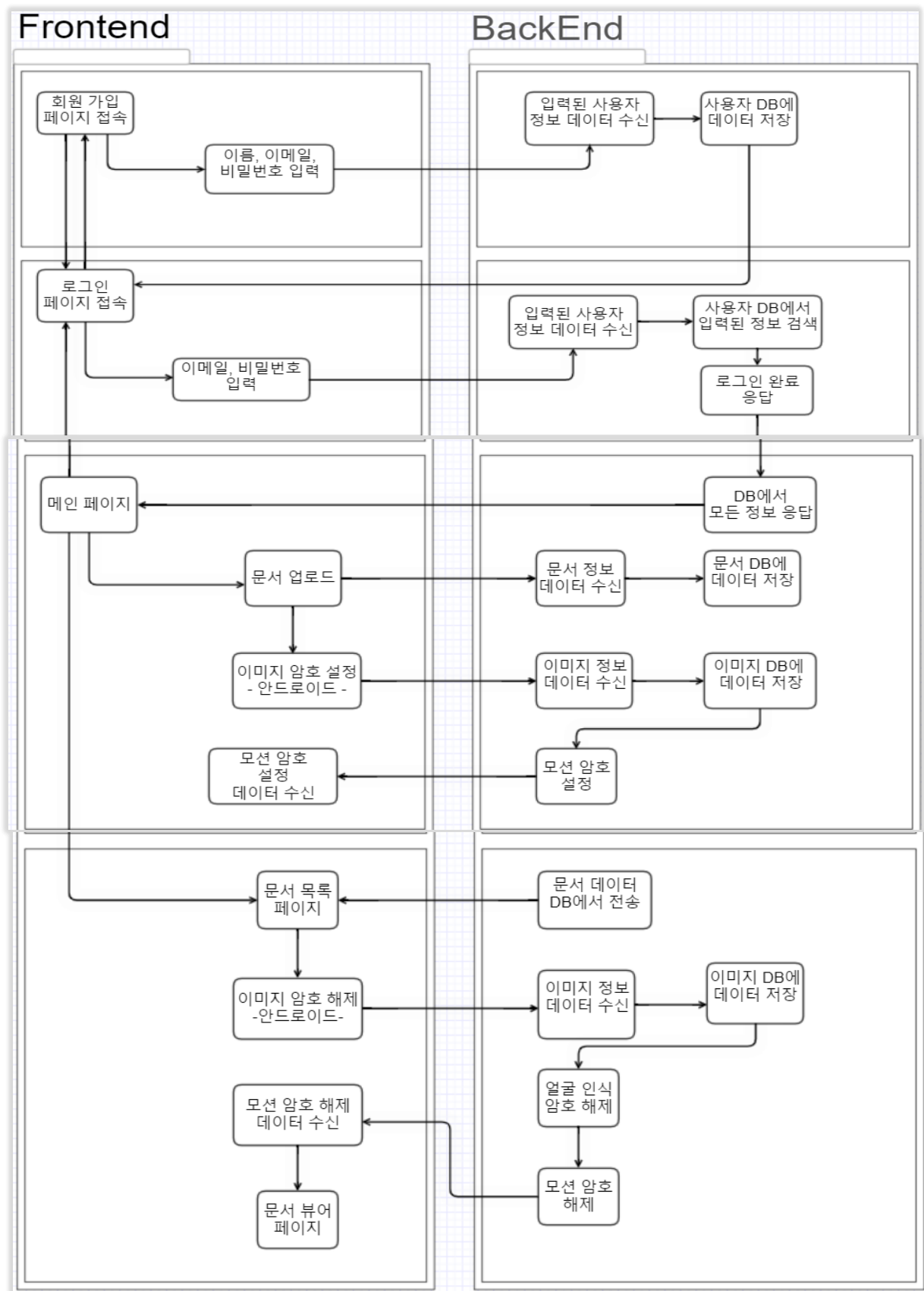
Foreign Key로 얼굴 암호를 어떤 문서에 대하여 설정하였는지 구분하는 정보이다.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	image_password_url	varchar(255)	NO		NULL	
	account_id_image_password	int	NO		NULL	
	document_id_image_password	int	NO	MUL	NULL	

데이터베이스 생성 4 : 생성된 image_password table

② Spring서버와 MySQL 데이터베이스 연결

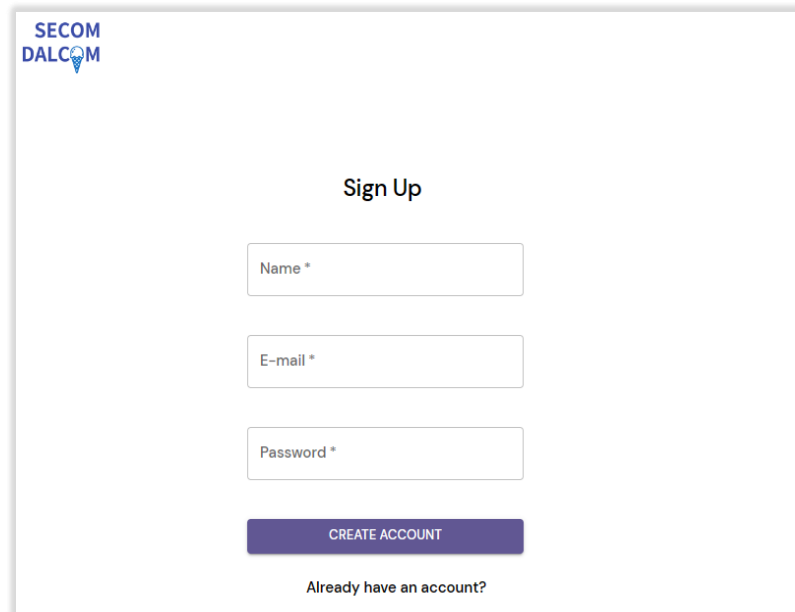
위의 내용처럼 생성한 데이터베이스를 spring 서버와 연결하여 서버에서 데이터베이스에 데이터를 저장하고, 조회할 수 있도록 한다. MySQL과 spring 서버를 연결하기 위해 mySQL에 대한 의존성을 추가한다.



구현 방법 1: 전체 Flow Chart

2.1.5 FrontEnd 2 – 회원가입 및 로그인

① Web 회원가입

A screenshot of a web registration form titled 'Sign Up' for 'SECOM DALCOM'. The form is centered on a white background. It features three input fields: 'Name *', 'E-mail *', and 'Password *', each with a light gray border. Below these fields is a purple button with the text 'CREATE ACCOUNT' in white. At the bottom of the form, there is a link that says 'Already have an account?'. The top left corner of the form area displays the 'SECOM DALCOM' logo.

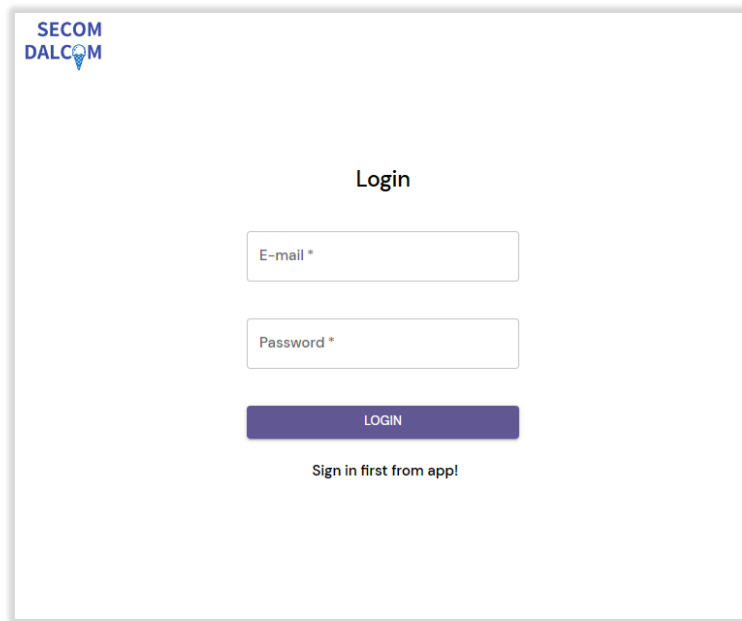
Web 회원가입 1 : 회원가입 UI

회원가입 페이지를 통해 회원가입을 할 경우 이름, 이메일, 비밀번호를 입력 받는다. 입력 받은 3가지 정보를 Backend 서버로 보낸다. 이 과정에서, Backend와의 CORS(Cross Origin Resource Sharing)라는 교차 출처 리소스 공유 에러가 발생할 수 있다. 따라서 React의 Proxy 설정을 통해, 다른 출처의 선택한 자원에서 접근할 수 있는 권한을 부여해 해당 에러를 해결한다.

이후에 Axios통신을 사용한다. Post를 통해 데이터에 접근하고, Get을 통해 데이터를 조회가 가능하게 한다. 회원가입 기능에서는 Post를 통해 사용자의 이름(name), 이메일 주소(email), 비밀번호(password)의 정보를 Backend에 전달하고 결과를 응답 받는다. Backend에서 보내는 회원가입의 성공여부를 변수로 하여 이미 가입이 된 회원과 그 경우가 아닌 신규회원의 여부를 판단한다. 이미 가입된 회원일 경우, 경고창을 통해 사용자에게 현재 상태를 보여준다.

로그인 페이지에서 Sign in first from app!라는 문구를 통해 회원가입 창으로 이동할 수 있으며, 회원가입 화면에서는 'Already have an account?' 문구를 통해 다시 로그인 창으로 원활하게 이동할 수 있다. MUI 'Typography'의 'onClick' 함수를 통해 'useNavigate'로 설정한 페이지의 주소로 전환이 가능하게 한다.

② Web 로그인

The image shows a web login interface. At the top left is the logo for 'SECOM DALCOM'. In the center, the word 'Login' is displayed. Below it are two input fields: 'E-mail *' and 'Password *'. Under the password field is a purple button labeled 'LOGIN'. At the bottom, there is a link that says 'Sign in first from app!'.

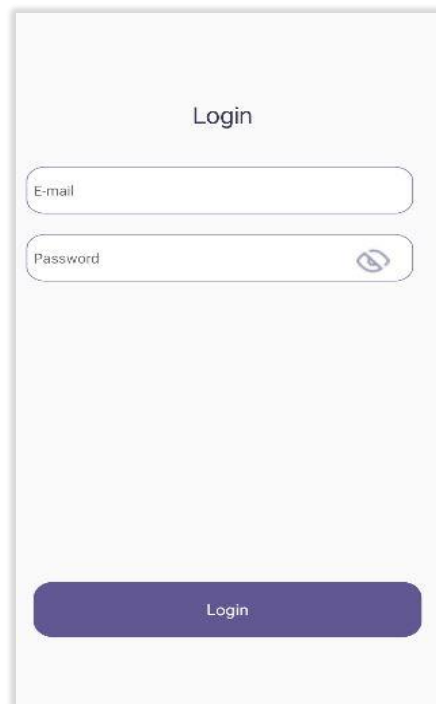
Web 로그인 1: 로그인 UI

회원가입을 하였다면, 로그인 페이지를 통해 로그인을 할 수 있다. 사용자가 이메일, 비밀번호를 입력하면, Backend로 그 정보를 보낸다. 회원가입과 동일한 과정으로 Proxy 설정을 통해 CORS문제를 해결하고, Axios 통신을 사용해 서버와의 통신을 구성한다.

Post를 통해 사용자의 이메일 주소(email)와 비밀번호(password)의 정보를 Backend로 보내고, JWT 토큰을 응답 받는다. 이후 Backend에서 보내는 로그인의 성공여부를 변수로 하여 가입이 된 회원의 여부를 판단한다. 저장된 회원의 정보가 아닐 경우 경고창을 통해 사용자에게 올바른 정보가 아니라는 것을 알리고, 페이지를 새로고침 하여 사용자가 다시 로그인을 시도할 수 있게 한다.

React에서의 webStorage의 한 종류인 sessionStorage의 기능을 통해 현재 로그인 상태를 확인한다. 로그인 상태인 경우에만 가입된 유저를 대상으로 한 기능들을 사용할 수 있다. Backend에서 token을 받아 webStorage에 저장한다. 또한 sessionStorage는 브라우저 창이 닫히면, 세션이 종료와 동시에 storage에 저장되었던 데이터도 소멸된다.

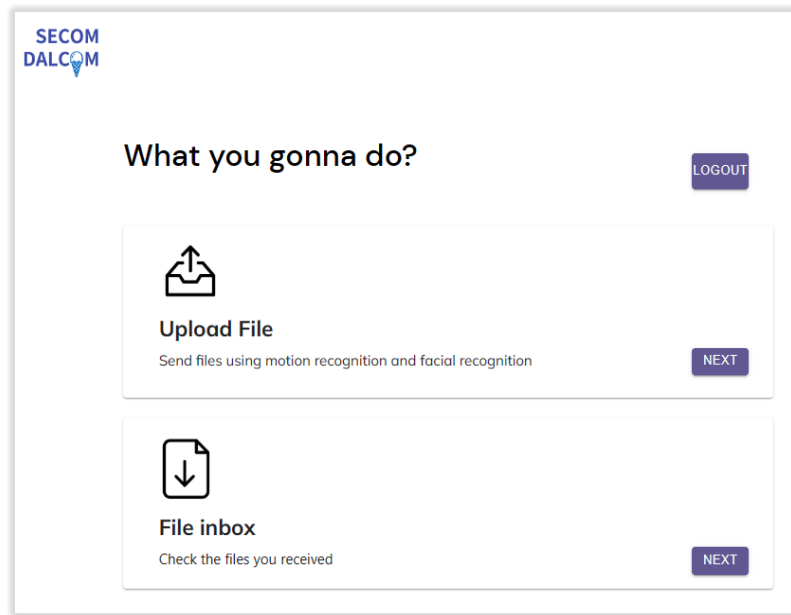
③ App 로그인

A mobile app login screen mockup. At the top, the word "Login" is centered. Below it are two input fields: "E-mail" and "Password". The "Password" field has a small eye icon on the right side. At the bottom, there is a large, rounded purple button with the word "Login" in white text.

App 로그인 1: 로그인 UI

회원가입을 완료한 사용자는 APP의 로그인 페이지에서 로그인을 할 수 있다. 생성한 계정을 입력하고, login 버튼을 클릭하면 Backend에서는 해당 계정이 사용자인지 확인한다. RESTful API와 통신하기 위해 사용되는 네트워크 라이브러리인 Retrofit2를 안드로이드에서 사용한다. API 인터페이스를 정의하여 통신하려는 Backend의 RESTful API 메서드를 정의하고, Retrofit2의 요청 메소드(Post)를 지정한다. 그 이후 Retrofit2를 사용하기 위해 API를 요청하고, 이를 받을 데이터 모델(Email, Password)을 의미하는 클래스로 구성된 Retrofit2 인스턴스를 생성한다. 이후 생성한 API 인터페이스의 메서드를 호출하여 Backend로부터 데이터를 요청한다. 회원이면 로그인에 성공하여 메인 화면으로 넘어간다.

④ Web 메인 화면

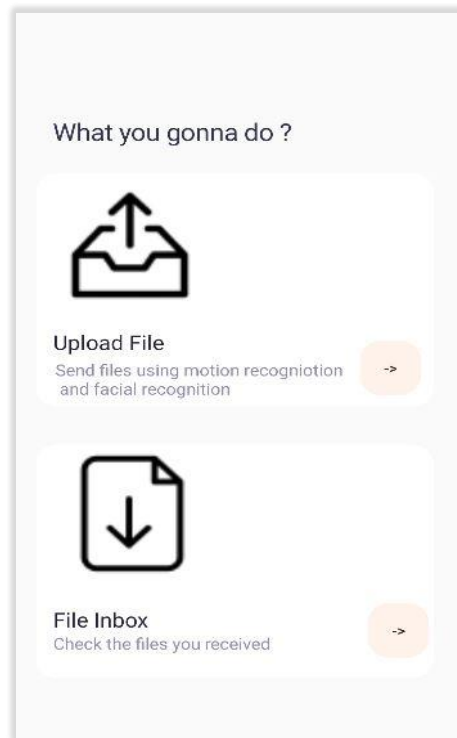


Web 메인화면 1 : Web 메인화면 UI

메인 화면은 사용자가 어플리케이션을 실행했을 때 처음 보게 되는 화면으로, 직관적이고 편리한 정보를 제공하여야 한다. 사용자는 메인화면에서 Upload File 또는 File Inbox 중 원하는 서비스를 이용할 수 있다. 원하는 서비스의 버튼을 누르면, Button 의 onClick 함수를 통해 사용자가 원하는 서비스로 연결한다.

sessionStorage의 기능을 활용하여 로그아웃 기능도 구현할 수 있다. 사용자가 로그아웃 버튼을 누르면, sessionStorage에 저장된 현재 id 값을 삭제하면서 다시 login 페이지로 이동한다. 로그아웃 상태로 돌아가면, 가입된 유저를 대상으로 한 기능들은 사용할 수 없게 된다.

⑤ App 메인 화면



App 메인 화면 1 : App 메인 화면 UI

App에서도 마찬가지로 로그인에 성공하면, 메인 화면에 접속된다. 메인 화면에는 파일을 업로드할 수 있는 기능과 업로드 한 파일을 볼 수 있는 기능으로 구성된다. Button을 누르면, onclick 함수를 사용해 해당 페이지로 이동하게 했다.

2.1.6 BackEnd 2 – 회원가입 및 로그인

① 회원가입

사용자가 회원가입 시 입력한 정보인 이름, 이메일, 비밀번호에 대한 정보를 Frontend에서 받아온다. 이때 비밀번호는 Spring Security가 제공하는 암호화 클래스를 이용하여 암호화한다. 이름, 이메일, 암호화된 비밀번호 정보와 회원가입 당시의 시간, 사용자 권한에 대한 정보를 서버에서 설정하여 database에 저장한다.

만약 사용자가 입력한 이메일이 이미 데이터베이스에 존재할 경우, 회원가입이 불가능하다.

이때 사용자는 손동작 인식과 얼굴 인식을 기반으로 한 문서 보안 서비스를 이용할 일반 사용자를 위한 회원가입이다.

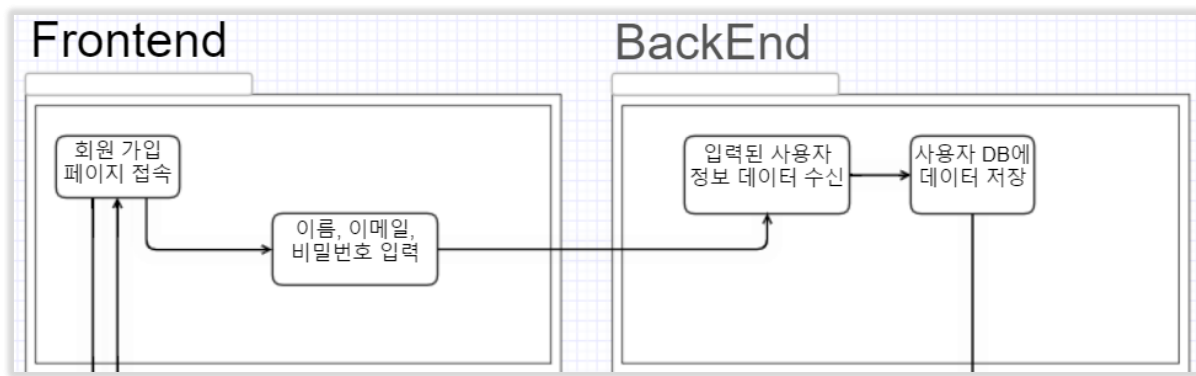
	id	account_password	account_name	account_created_at	account_email	account_role
▶	1	\$2a\$10\$.xfrM5pKgV22TqF7Ra4A6.BcTIEaNSIR...	우신영	2023-05-08 19:01:27	tsdud8205@naver.com	ROLE_USER
	2	\$2a\$10\$dsfdqb5CCaV0Pfs0JlsEKusCzBpfCsa.p...	우신영	2023-05-16 18:40:39	2020112087@dgu.ac.kr	ROLE_USER
•	NULL	NULL	NULL	NULL	NULL	NULL

회원가입 1 – database에 저장된 회원 정보

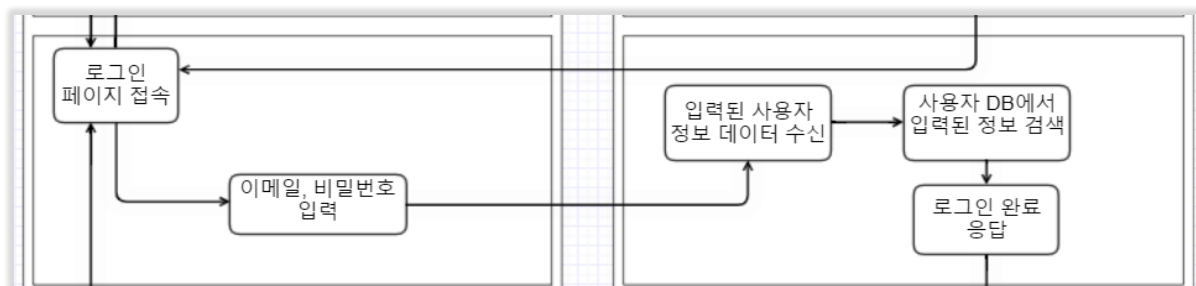
위의 사진을 보면 사용자의 암호화된 비밀번호, 이름, 이메일, 회원가입 시간 그리고 사용자 권한에 대한 정보가 database서버로 전송되어 저장되는 것을 볼 수 있다.

② 로그인

회원가입을 완료한 사용자가 로그인을 시도하면, 로그인 시 입력한 이메일, 비밀번호를 수신하여 account table에 해당 회원이 존재하는지 조회하여 성공 여부와 사용자에게 대한 jwt 토큰을 Frontend에 전달한다.



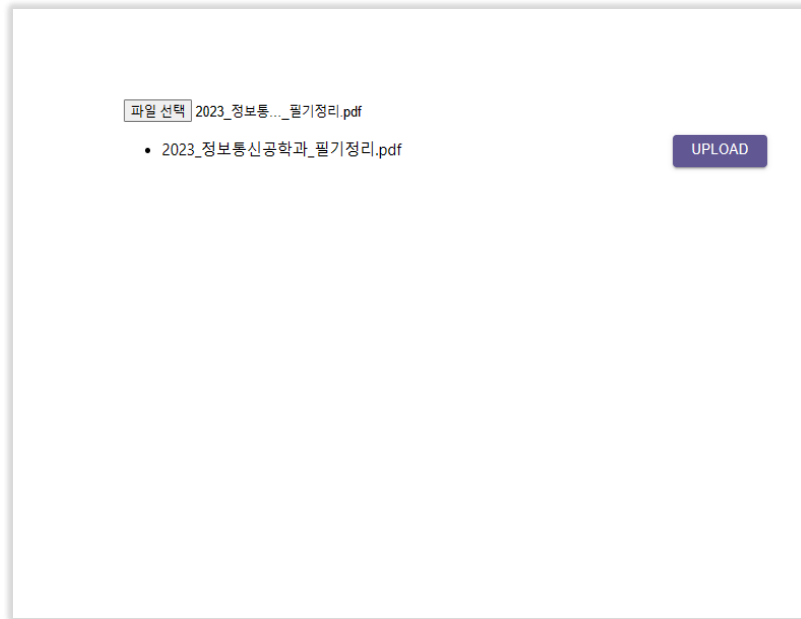
구현 방법 2 : 회원가입 Flow Chart



구현 방법 3 - 로그인 Flow Chart

2.1.7 FrontEnd 3 – 문서 업로드 및 암호 설정

① Web 문서 업로드

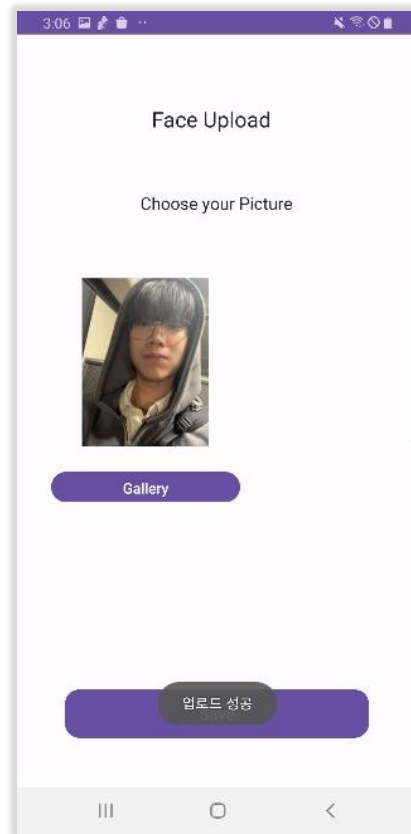


Web 문서 업로드 1 - 문서 업로드

사용자가 웹 어플리케이션에서 문서 파일을 선택하고 FormData를 활용하여 해당 파일을 준비한다. React에서는 `<input type="file">`을 사용하여 파일을 선택할 수 있다. 사용자가 업로드한 파일은 JavaScript의 FormData 객체를 사용하여 형식에 맞게 처리된다. FormData는 선택한 파일의 키-값 쌍을 포함하고, 이를 서버로 전송할 준비를 한다.

React에서는 Axios 라이브러리를 사용하여 HTTP 요청을 처리한다. 선택한 문서 파일을 FormData 객체와 함께 POST 요청으로 Backend에 전송한다. 이때, data 속성에 FormData 객체를 설정하여 파일을 첨부한다. 이후, Backend의 엔드포인트 URL로 POST 요청을 보내고 성공 여부와 문서 ID가 응답 받는다.

② APP 이미지 업로드



APP 이미지 업로드 1 - APP 이미지 업로드

1 차 보안인 얼굴 인식 암호를 사용하기 위해, 사용자가 본인 얼굴이 포함된 사진을 Backend 에 업로드 하는 과정이다. 버튼을 통해 사용자의 기기에 갤러리를 열어 사진을 선택한다. 선택된 사진을 어플리케이션 화면에서 보여주고 이를 Backend 에 업로드 한다. 이때도 로그인과 마찬가지로 Retrofit2 를 사용하여 통신이 이루어진다. 이미지 업로드를 위해 Retrofit2 에서는 @Multipart 어노테이션을 사용하여 멀티파트 요청을 설정한다. 이를 통해 일반적인 폼 데이터와 함께 이미지 파일을 Backend 에 보내고, 통신의 결과와 문서 ID 를 응답 받는다.

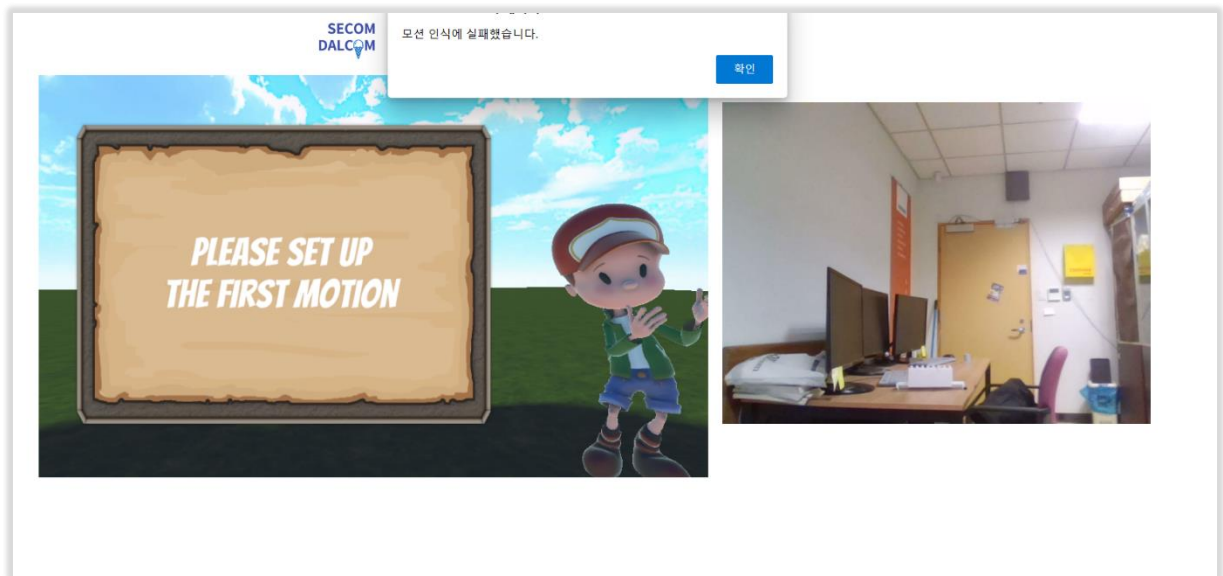
③ APP에서 Web과 통신 (모션 설정)



APP에서 Web과 통신 1 - APP에서 Web과 통신

이미지 업로드에 성공하면, 페이지 이동 후 Web을 호출한다. 이때 Backend가 어플리케이션과 Web 사이를 중계하여 2차 보안인 손동작 인식 암호를 Web 환경에서 설정하도록 한다. APP에서는 Retrofit2를 사용하여 Backend에 요청을 보내고, Backend에서는 해당 요청을 노트북의 웹으로 중계하는 구조로 구현하였다. 이 과정을 통해 안드로이드 어플리케이션과 노트북의 웹 간의 통신을 효과적으로 관리할 수 있다.

④ Web 유니티 모션 설정



Web 유니티 모션 설정 1- 유니티 모션설정

React의 Axios 통신을 통하여 모션설정에 대한 결과를 Backend의 URL로 받게 되면, 해당 결과를 유니티 애니메이션을 통해 사용자에게 보여준다. 모션 설정이 원활하게 이루어지면, React의 useEffect 기능을 통해 변수를 감지하고, 다음 단계 애니메이션으로 진행한다. 최종적으로는 모든 암호 손동작 설정이 완료되어 메인화면으로 이동한다. 암호 손동작을 설정하는 과정에서 손동작 정보가 부족한 이유 등으로 암호 손동작이 설정되지 않을 수 있다. 이런 경우 경고 창 기능을 통해 페이지 이동이 이루어지고, 해당 페이지에서 다시 암호 손동작을 설정할 수 있도록 했다

2.1.8 BackEnd 3 – 문서 업로드 및 암호 설정

① 문서 업로드

사용자가 문서를 업로드하고, 해당 문서에 대한 이름을 입력하면 문서 파일을 Frontend에서 받아온다. 문서 파일은 로컬 서버에 저장된다. 이때 파일의 이름이 중복되어 저장이 불가능한 경우가 발생하는 것을 방지하기 위해, 파일명을 임의의 문자열로 바꾸어 저장한다. 이후 저장한 문서 파일에 대한 이름(임의의 문자열), 문서의 본래 이름, 문서 업로드 당시의 시간, 문서가 서버에 저장된 경로와 사용자의 id를 database에 저장한다.

id	document_name	document_url	document_created_at	document_original_name	account_id_document
3	94043a1a-3e1e-430f-9445-ef969db662f6.docx	C:\Users\woosy\CapstoneDesign\보고서.docx	2023-05-15 19:56:24	보고서.docx	1
4	da0ffa0-ad7b-4546-9b67-11a1c2eb8b22.docx	C:\Users\woosy\CapstoneDesign\da0ffa0-ad7...	2023-05-15 20:08:43	보고서.docx	1
5	6dd76ec2-0ccb-4da7-ba03-890ee410eae8.docx	C:\Users\woosy\CapstoneDesign\documents\6...	2023-05-15 22:23:00	Progress Report 3.docx	1
6	46f0285c-365e-41b6-8d5e-6fe26fae84bd.png	C:\Users\woosy\CapstoneDesign\documents\4...	2023-05-16 16:05:57	스크린샷 2023-03-06 234511.png	1

문서 업로드 1 : database에 저장된 문서 정보

② 문서에 대한 얼굴 인식 암호 설정

사용자는 얼굴 이미지를 업로드 함으로써 문서에 대한 얼굴 인식 암호를 설정할 수 있다. 얼굴 인식 암호의 설정과 해제는 모바일 기기에서 진행한다. 모바일 기기의 갤러리에서 사용자가 얼굴 이미지를 선택하여 업로드하면, 해당 이미지 파일은 로컬 서버에 저장된다. 이때 파일의 이름이 중복되어 저장이 불가능한 경우가 발생하는 것을 방지하기 위해, 임의의 문자열로 파일명이 바뀌어 저장한다. 이후 문서가 서버에 저장된 경로, 사용자의 id, 문서의 id가 database에 저장된다.

id	image_password_url	account_id_image_password	document_id_image_password
1	C:\Users\woosy\CapstoneDesign\images\8b68de97-f4b5-4a97-96dd-9a907b485c...	2	28
2	C:\Users\woosy\CapstoneDesign\images\628e1806-0965-44fc-ac7f-799e3235d6d...	1	29
4	C:\Users\woosy\CapstoneDesign\images\b1608b62-f9d9-4fbb-bcdc-609c01a851fc...	1	30
5	C:\Users\woosy\CapstoneDesign\images\d30f25f5-2a92-4cea-8114-8e74ce3983b...	1	31

문서에 대한 얼굴 인식 암호 설정 1 : database에 저장된 얼굴 인식 암호 정보

③ 문서에 대한 손동작 인식 암호 설정

a. 손동작 인식 dataset 생성

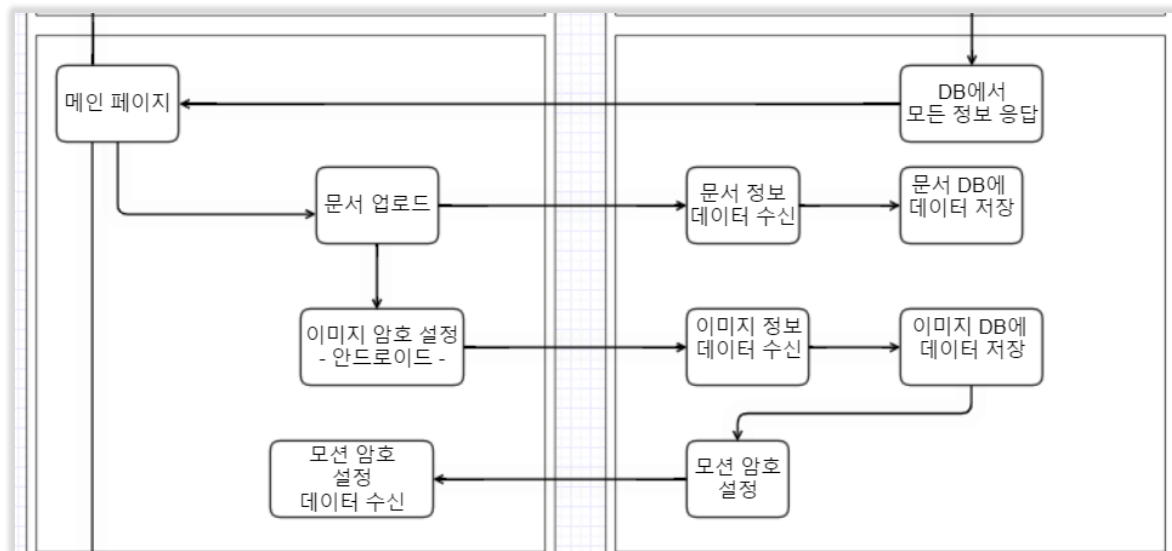
본 프로젝트에서는 Spring 프레임워크로 서버를 구성하고, 손동작 인식을 위한 코드는 Python으로 구현된다. 따라서 Java 언어로 Python 파일을 실행하는 코드가 필요하다. 손동작 인식 암호 설정 페이지로 들어오면, 컨트롤러에서 Java의 ProcessBuilder 라이브러리를 사용하여 Python 파일인 *setMotion.py*를 실행한다. *setMotion.py*파일을 build할 때, 문서 이름과 손동작의 순서를 파라미터로 전송하게 했는데, 이는 Python 파일에서 수집한 데이터들을 파일로 저장할 때 파일명으로 사용된다.

b. 손동작 인식 model 생성

3가지의 손동작에 대한 dataset이 생성되면, 해당 dataset을 이용하여 *trainMotion.py* 파일에서 손동작 인식 model을 생성한다. *trainMotion.py* 파일을 build할 때는 문서의 이름을 파라미터로 전송한다. 문서에 대한 손동작 dataset들은 문서의 이름과 동작의 순서를 조합해 저장 되어있고, 모델 생성 과정에서 그 데이터들에 접근하기 위함이다. 실행이 완료되면, 문서에 대한 model이 생성되고, 서버에 저장된다. 이후, model의 저장경로와, 사용자의 id, 문서의 id가 database에 저장된다.

	id	motion_password_url	account_id_motion_password	document_id_motion_password
▶	1	C:/Users/woosy/CapstoneDesign/models/model...	1	26
	2	C:/Users/woosy/CapstoneDesign/models/model...	1	26
*	NULL	NULL	NULL	NULL

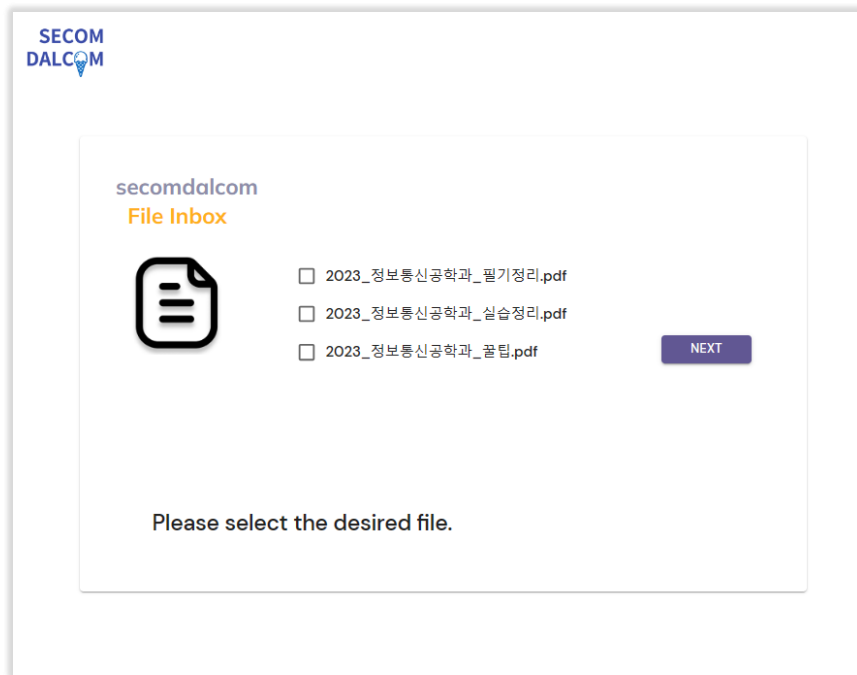
BackEnd3 2 – database에 저장된 손동작 인식 암호 정보



구현 방법 4: 문서 업로드 및 암호 설정 Flow Chart

2.1.9 FrontEnd 4 – 문서 확인 및 암호 해제

① Web 문서리스트

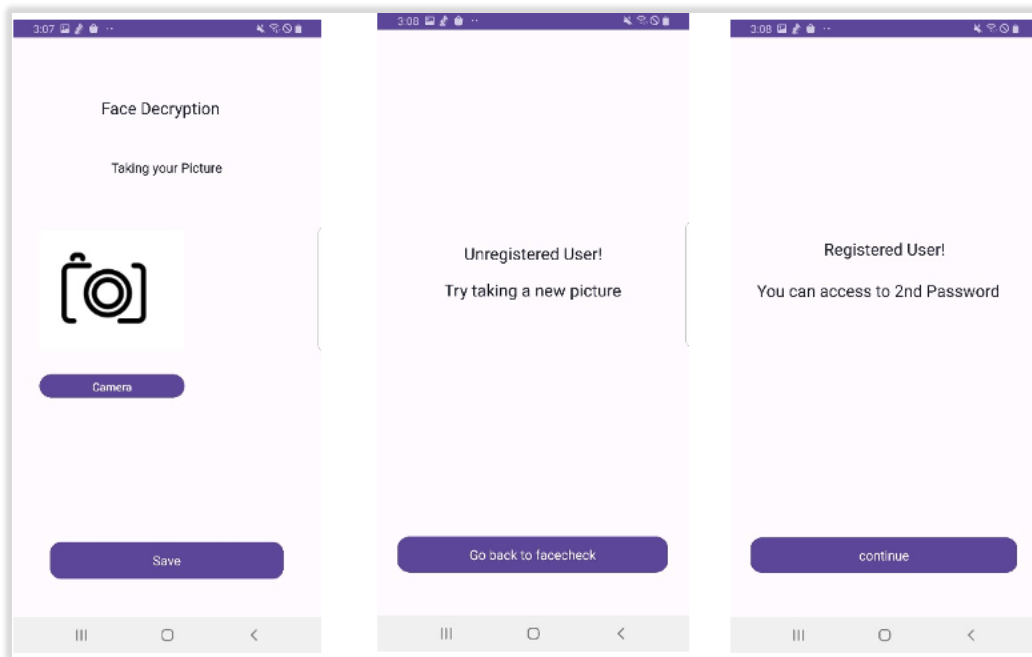


Web 문서리스트 1 : Web 문서리스트

문서 목록은 사용자가 이전에 업로드한 파일을 열람할 수 있는 기능이다. React 의 Axios 통신을 통하여 Backend 로부터 문서 데이터 조회하여, 문서 목록을 나타내는 JSON 형태의 데이터가 응답 된다. 이를 통해 사용자가 업로드했던 문서들을 리스트의 형태로 확인할 수 있다.

체크박스 기능을 활용하여 원하는 문서를 선택 혹은 해제할 수 있고, 사용자가 선택한 문서들의 리스트를 보여주어 사용자가 확인할 수 있는 파일들의 목록을 시각화 한다. 위 과정을 통해 사용자가 원하는 문서에 접근할 수 있게 하였다.

② APP 사진 촬영을 통한 본인인증



APP 사진 촬영을 통한 본인인증 1: APP 사진 촬영을 통한 본인인증

이미지 업로드와 유사한 과정을 가진다. 이미지 업로드 과정에서는 보유한 갤러리에서 사용자의 사진을 업로드했지만, 이 과정에서는 카메라 어플리케이션을 통해 사용자 본인 얼굴을 촬영한다. 이 또한 마찬가지로 Retrofit2 를 사용하여 통신이 이루어진다. 이미지 업로드를 위해 Retrofit2 에서는 @Multipart 어노테이션을 사용하여 멀티파트 요청을 설정한다. 이를 통해 일반적인 폼 데이터와 함께 이미지 파일을 Backend 에 보내지고, 통신의 결과와 문서 ID 를 응답 받는다.

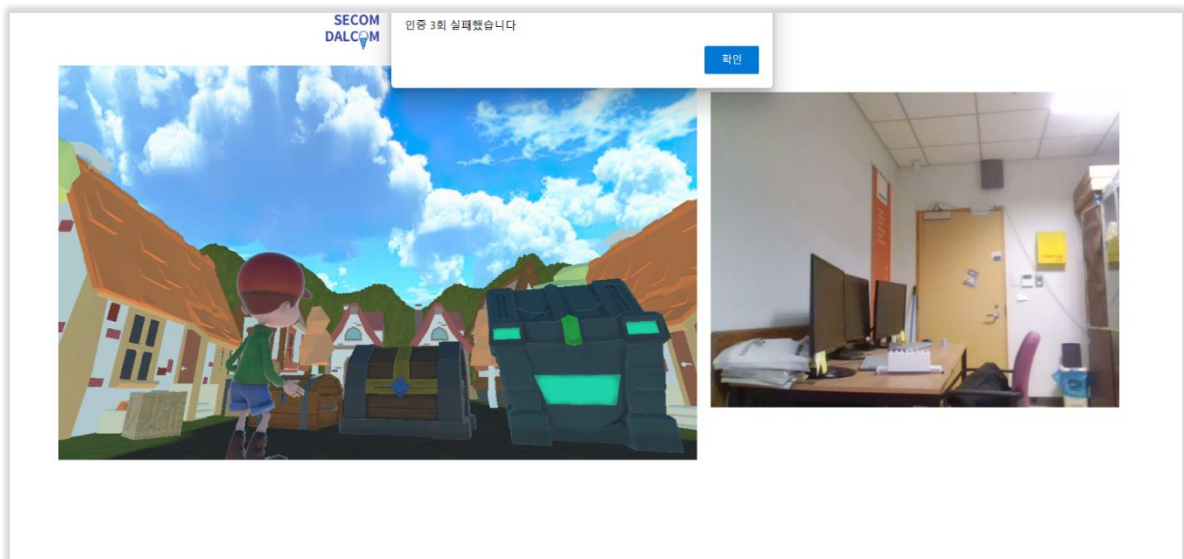
③ App에서 Web 통신(모션 해제)



App에서 Web 통신 1 : App에서 Web 통신

Backend 에서 보낸 1 차 보안의 해제의 여부를 받아 해제가 되면, 페이지 이동 후 Web 에서 손동작 인식 암호를 해제할 수 있는 환경이 열리게 된다. 이때도 Backend 가 어플리케이션과 Web 사이를 중계하여 2 차 보안 해제 과정을 진행할 수 있게끔 한다. 어플리케이션에선 Retrofit2 를 사용하여 Backend 에 요청을 보내고, Backend 에서는 해당 요청을 노트북의 웹으로 중계하는 구조로 구현하였다. 이 과정을 통해 안드로이드 어플리케이션과 노트북의 웹 간의 통신을 효과적으로 관리할 수 있다.

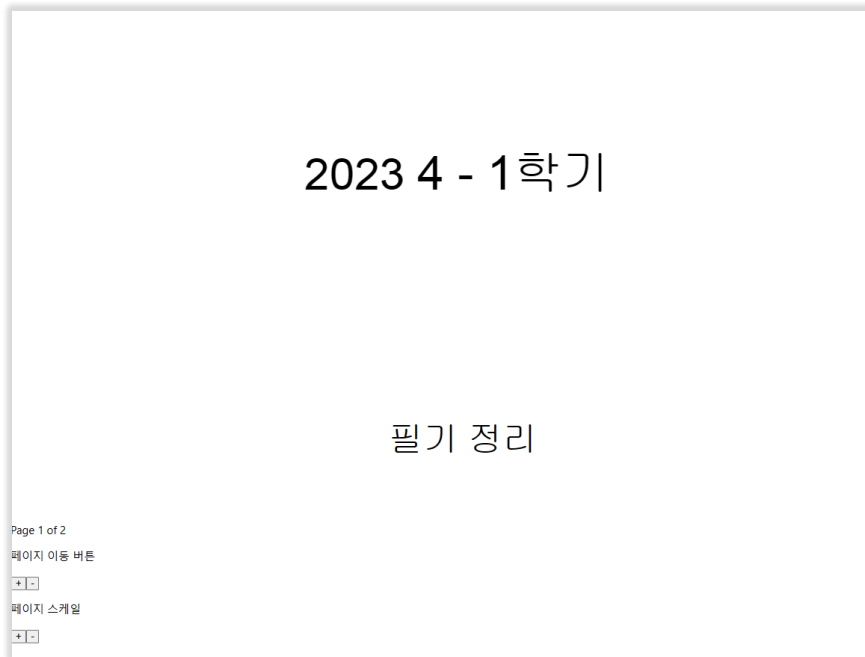
④ Web 유니티 모션 해제



Web 유니티 모션 해제 1: Web 유니티 모션 해제

React의 Axios 통신을 통하여 Backend로부터 손동작 암호 해제에 대한 결과를 응답받고, 그 결과에 맞는 유니티 애니메이션을 실행하여 사용자에게 해당 과정의 결과를 보여주게 된다. 모션 해제가 원활하게 이루어지면, useEffect 기능을 활용하여 변수에 대한 변화를 감지한다. 그 변수의 값을 통해 상자가 순차적으로 열리는 애니메이션을 실행하고 문서에 접근할 수 있게 된다. 하지만 해제 과정 중에서 해제가 이루어지지 않으면, 경고창을 통해 손동작 암호 해제 실패에 대한 메시지를 사용자에게 전달하고 페이지를 이동시켜 다시 손동작 암호를 해제하게끔 한다. 3회 모션 인식 실패 시, 다시 모션을 해제하는 것이 아닌 메인으로 이동하는 것으로 보안성을 높였다.

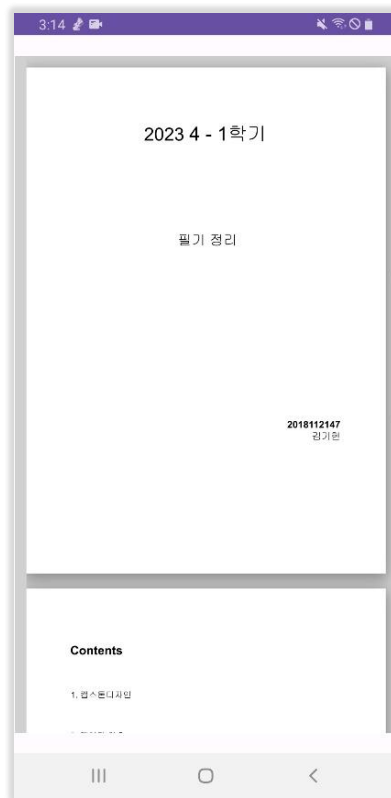
⑤ Web 문서 뷰어



Web 문서 뷰어 1 : Web 문서뷰어

사용자가 원하는 파일을 불러오기 위해 React의 Axios 통신을 하는데, Backend의 URL을 통해 문서에 대한 데이터에 접근한다. 그 전에 pdf.js 라이브러리를 통해 웹페이지에서 PDF 파일을 렌더링하거나, 처리할 때 필요한 워커 스크립트를 먼저 불러온다. Backend와의 통신이 이루어지면 응답의 데이터 타입을 Blob으로 설정하고, Backend로부터 받은 응답 데이터로 Blob 객체를 생성한다. 생성된 Blob 객체를 사용하여 이름이 "file.pdf"이고 MIME 타입(파일의 종류를 나타내는 식별자)이 "application/pdf"인 파일 객체를 생성한다. 생성된 파일 객체를 사용하여 파일 URL을 생성하고 해당 URL은 React의 useState를 사용하여 상태를 관리한다. 이를 통해 파일 URL을 컴포넌트에서 사용하거나 렌더링 할 수 있다.

App 문서뷰어



App 문서뷰어 1 : App 문서뷰어

2 차 보안인 모션 암호를 모두 해제하면 URL 을 통해 문서에 대한 데이터에 접근한다. 응답 데이터 타입을 Blob 으로 설정하여 사용자가 원하는 파일을 응답 받도록 하고, 어플리케이션에서 해당 문서를 확인할 수 있다. 이때 안드로이드의 WebView 와 구글 Docs 주소를 통해 PDF 파일을 확인할 수 있다. 안드로이드 WebView 는 웹 콘텐츠를 어플리케이션 내에서 표시하는 데 사용되는 뷰이다. WebView 를 사용하여 구글 Docs 주소를 통해 안드로이드 어플리케이션에서 PDF 파일을 웹 형태로 표시할 수 있다. 이때 사용자는 PDF 파일을 스크롤하고 확대/축소할 수 있다.

2.1.10 BackEnd 4 – 문서 확인 및 암호 해제

① 문서 목록 확인

문서 목록 확인 페이지에서는 사용자가 업로드한 문서의 목록을 확인할 수 있다. 사용자가 업로드한 문서에 대한 목록을 database에서 조회하여 문서의 목록을 FrontEnd에 전달한다.

② 문서에 대한 얼굴 인식 암호 해제

a. 암호 해제를 위한 Python 파일 실행

사용자는 업로드한 문서에 대한 얼굴 인식 암호 해제를 할 때, 직접 얼굴의 사진을 찍어 서버에 전송한다. 문서에 대한 얼굴 인식 암호를 해제하는 페이지로 들어오면 컨트롤러에서 Java의 ProcessBuilder 라이브러리를 사용하여 *imagePasswordUnlock.py* 라는 Python 파일을 실행한다. 컨트롤러에서 *imagePasswordUnlock.py* 파일을 실행할 때, FrontEnd에서 전송된 카메라로 찍은 얼굴 이미지와 database에서 해당 문서에 설정된 얼굴 인식 암호 이미지를 조회하여 두 개의 이미지 파일에 대한 경로를 파라미터로 전송한다. 이는 Python 파일에서 두 개의 이미지 파일에 대한 경로를 수집하고, 얼굴 인식 암호 해제에 대한 여부를 반환하기 위함이다.

b. 암호 해제 여부 전달

imagePasswordUnlock.py 파일을 실행하면 얼굴 인식 암호의 해제 여부가 출력되고, 출력 정보를 읽어와 암호 해제에 대한 여부를 FrontEnd에 전송한다.

③ 문서에 대한 손동작 인식 암호 해제

a. 암호 해제를 위한 Python 파일 실행

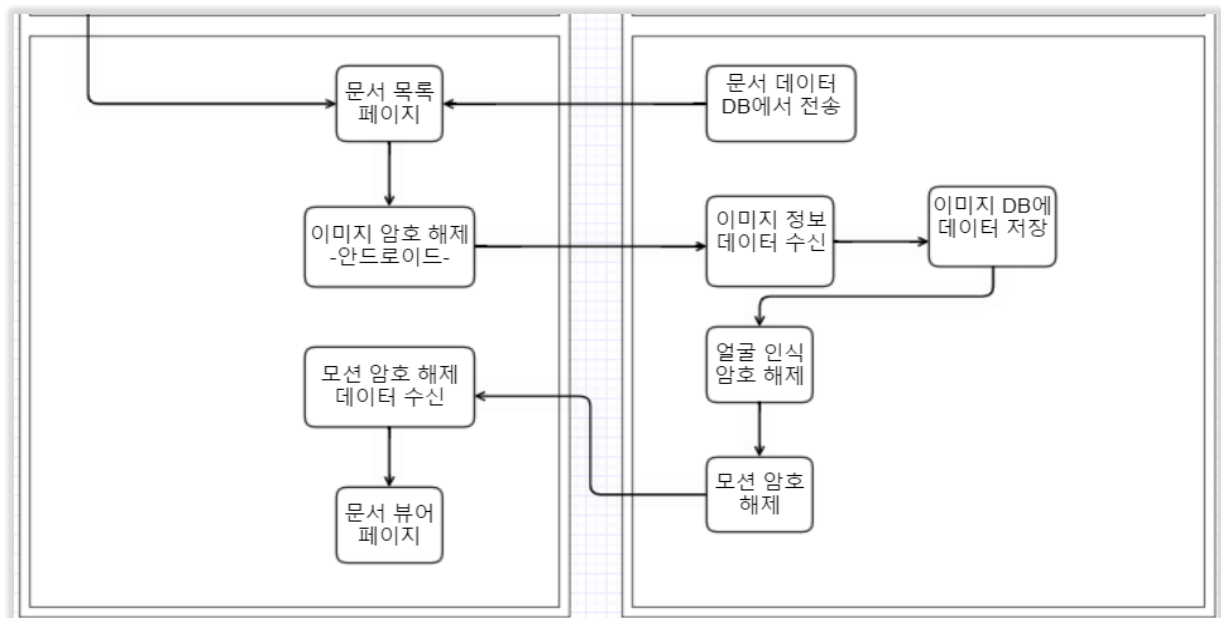
문서에 대한 손동작 인식 암호를 해제하는 페이지로 들어오면 컨트롤러에서 Java의 ProcessBuilder 라이브러리를 사용하여 *testMotion.py* 라는 Python 파일을 실행한다. 컨트롤러에서 *testMotion.py* 파일을 실행할 때, database에서 해당 문서에 설정된 손동작 인식 암호의 model 저장 경로와, 손동작의 순서를 파라미터로 전송한다. 이는 Python 파일에서 문서에 설정된 손동작 암호의 model 경로를 수집하여 model을 가져오고, 순서에 맞는 손동작인지에 대한 여부를 반환하기 위함이다.

b. 암호 해제 여부 전달

testMotion.py 파일을 실행하면 손동작 인식 암호의 해제 여부가 출력되고, 출력 정보를 읽어와 암호 해제에 대한 여부를 FrontEnd에 전송한다.

④ 문서 확인

얼굴 인식 암호가 모두 해제되어 문서 뷰어 페이지로 들어오면, database에서 해당 문서에 대한 저장 경로를 조회하여 문서 파일을 가져온다. 문서 파일은 blob형태로 변환하여 Web FrontEnd와 APP FrontEnd에 전달한다.



구현 방법 5: 문서 확인 및 암호 해제 Flow Chart

2.1 구현 도구

2.1.1 손동작 인식

Visual Studio Code에서 Python 언어를 사용해 구현했다. 주로 사용한 라이브러리들은 다음과 같다.

① Mediapipe Hands

Mediapipe 라이브러리에서 제공하는 손 인식 모델이다. 실시간 비디오 스트림에서 손의 위치, 손가락의 각도, 손가락의 특징점 등을 추정하여 손 모양과 동작을 감지할 수 있다. 손을 감지하기 위해 손 모양의 특징을 학습한 신경망을 기반으로 하며, 손의 키포인트를 추정하기 위해 관절과 손가락의 위치를 예측하는 신경망을 사용한다. 본 프로젝트의 손동작 인식 파트에서는 암호 설정 및 암호 해제 과정에서 사용자의 손동작 정보를 추출하는 과정에서 쓰였다.

② Opencv

오픈소스 컴퓨터 비전 라이브러리로, 이미지 및 비디오 처리와 관련된 다양한 기능을 제공한다. 컴퓨터 비전 작업에 필요한 이미지 및 비디오 입력, 출력, 변환, 필터링, 특징 검출, 객체 추적 등의 기능을 제공하며, 카메라 캡처, 동영상 재생, 기하학적 변환 등과 같은 비디오 관련 작업에도 사용된다. 본 프로젝트의 손동작 인식 파트에서는 웹캠 이미지를 실시간으로 받아오는 과정에서 쓰였다.

③ Tensorflow

TensorFlow는 구글에서 개발한 딥러닝과 기계 학습을 위한 오픈소스 라이브러리이다. 다양한 딥러닝 모델을 구축하고 학습시키는 데 사용되며, 분산 컴퓨팅, GPU 가속, 자동 미분 등의 기능을 제공한다. 본 프로젝트의 손동작 인식 파트에서는 유사도 검사를 위한 모델을 학습시키는 과정에서 쓰였다.

④ Keras

Keras는 딥러닝 모델을 구축하고 학습시키기 위한 고수준 신경망 API이다. TensorFlow, Theano, CNTK 등의 백엔드 엔진을 사용하여 딥러닝 모델을 구성하고 학습시킬 수 있으며, 사용하기 쉬운 API를 제공하여 딥러닝 모델의 설계와 실험을 단순화하고, 빠른 프로토타이핑과 실험을 가능하게 한다. 다양한 딥러닝 모델 아키텍처와 레이어를 지원하며, 이미지 처리, 자연어 처리, 음성 인식 등 다양한 분야에서 주로 사용된다. 본 프로젝트의 손동작 인식 파트에서는 유사도 검사를 위한 모델을 학습시키는 과정에서 쓰였다.

⑤ Scikit-learn

Scikit-learn은 파이썬에서 사용할 수 있는 머신러닝 라이브러리이다. 다양한 머신러닝 알고리즘, 데이터 전처리 기능, 모델 평가 도구, 차원 축소 기법 등을 포함하고 있으며, 주로 분류, 회귀, 클러스터링, 차원 축소, 모델 선택 등 다양한 머신러닝 작업을 지원한다. 데이터 분석 및 예측 모델링에서 사용되는 라이브러리이다. 본 프로젝트의 손동작 인식 파트에서는 유사도 검사를 위한 모델을 학습시키는 과정에서 쓰였다.

2.1.1. 얼굴 인식.

Visual Studio Code 에서 Python 언어를 사용해 구현했다. 주로 사용한 라이브러리들은 다음과 같다.

① Python

언어는 python 을 사용하고, 개발 환경으로 visual studio code 를 사용한다. Python 은 다양한 분야에서 사용되는 프로그래밍 언어로, 여러 장점을 지닌다. 쉬운 문법과 가독성을 지니며, 풍부한 라이브러리와 프레임워크를 갖추고 있어 높은 생산성을 갖는다. 본 프로젝트에서의 얼굴 인식에 필요한 라이브러리를 다양하게 갖추고 있어 해당 언어를 택한다. Visual studio code 는 microsoft 에서 통합 개발 환경이며, 가볍고 빠른 성능을 지니고, 다양한 언어를 지니고 있어 다른 여러 부분과 확장성이 용이하여 선택한다.

- Keras_facenet

keras_facenet 은 Keras 로 구현된 FaceNet 얼굴 인식 모델을 사용할 수 있게 해주는 패키지이다. FaceNet 은 얼굴 이미지에서 고정 크기의 임베딩 벡터를 추출하는 딥 러닝 모델이다.

- MTCNN

MTCNN(Multi-Task Cascaded Convolutional Networks)은 얼굴 감지를 수행하는 딥 러닝 기반의 알고리즘이다. MTCNN 은 얼굴 영역을 찾기 위해 다단계 탐지 절차를 사용한다.

- PIL

PIL(Python Imaging Library)은 이미지 처리 작업을 수행하기 위한 파이썬 라이브러리이다. PIL 을 사용하여 이미지를 로드하고, RGB 형식으로 변환하여 처리한다.

- NUMPY

numpy 는 파이썬에서 수치 연산을 위한 핵심 패키지이다. numpy 를 사용하여 이미지를 numpy 배열로 변환하고, 얼굴 영역을 추출한 후에도 numpy 배열로 처리한다.

2.1.2. FrontEnd

Visul Studio Code에서 JavaScript언어를 사용해 구현했다. 주로 사용한 라이브러리들은 다음과 같다.

① React

React는 페이스북에서 개발된 사용자 인터페이스(UI) 라이브러리로, 웹 애플리케이션의 구축 및 유지 관리를 위한 강력한 도구이다.

특징으로는 React는 가상 돔(Virtual DOM)을 사용하여 효율적인 UI 업데이트를 가능하게 한다. 가상 돔은 실제 돔(Document Object Model)과 동기화되어 있으며, UI의 변경 사항을 실제 돔에 직접 반영하는 대신, 가상 돔을 통해 변경 사항을 비교하고 최소한의 돔 조작만 수행한다. 이를 통해 성능 향상과 빠른 렌더링을 실현할 수 있다.

또한 컴포넌트 기반 프레임워크의 특징을 지닌다. React는 컴포넌트 기반 아키텍처를 사용하여 재사용 가능하고 모듈화된 UI 요소를 만들 수 있다. 각 컴포넌트는 독립적으로 동작하며, 필요한 경우 다른 컴포넌트와 조합하여 복잡한 UI를 구성할 수 있다. 이를 통해 코드의 가독성과 유지 보수성이 향상되고 개발 생산성이 높아진다.

마지막으로 React는 단방향 데이터 흐름을 따르는 아키텍처인 Flux나 Redux와 함께 사용된다. 이는 상태(State)를 컴포넌트 계층 구조를 따라 단방향으로 전달하고 업데이트하는 방식이다. 상태의 변경은 순수 함수인 리듀서(Reducer)를 통해 이루어지며, 이를 통해 애플리케이션의 상태 관리가 용이해진다.

Android Studio에서 Java언어를 사용하여 구현하였다

② Android Studio

Android Studio는 구글에서 제공하는 공식적인 안드로이드 애플리케이션 개발 도구이다. 안드로이드 스튜디오는 안드로이드 애플리케이션을 개발, 디자인, 테스트, 배포하기 위한 통합 개발 환경(IDE)이다.

특징으로 Android Studio는 안드로이드 프로젝트의 구조와 파일을 효율적으로 관리할 수 있도록 도와준다. 프로젝트 템플릿을 제공하며, Gradle을 사용하여 프로젝트의 종속성을 관리할 수

있다.

또한 Android Studio는 사용자 인터페이스(UI)를 디자인하기 위한 레이아웃 에디터를 제공한다. 시각적인 편집 환경에서 XML 기반의 UI 레이아웃을 작성하고, 미리보기 기능을 통해 실시간으로 디자인을 확인할 수 있다.

마지막으로 Android Studio는 안드로이드 디바이스 에뮬레이터를 제공하여 애플리케이션을 테스트할 수 있다. 또한, 실제 안드로이드 기기를 연결하여 테스트 및 디버깅을 할 수 있다.

2.1.3. BackEnd

IntelliJ에서 Java언어를 통해 구현했다. 사용한 프레임 워크는 다음과 같다.

① Spring

Spring framework는 Java 플랫폼에서 사용할 수 있는 오픈소스 애플리케이션 프레임워크이다. 엔터프라이즈급 애플리케이션을 개발하기 위한 다양한 기능을 제공하며, 경량화된 솔루션으로 알려져 있다. Spring framework는 자바 객체를 관리하기 위해 경량 컨테이너를 사용한다. 이 컨테이너는 객체의 라이프사이클을 관리하고 필요한 객체를 제공한다. 이는 제어의 역전 (IOC) 개념에 기반을 두고 있으며, 개발자가 객체 간의 의존성을 직접 관리하지 않고 스프링 컨테이너가 자동으로 의존성을 주입해준다.

Spring framework의 주요 특징 중 하나는 POJO (Plain Old Java Object)이다. POJO는 특별한 제약 없이 일반적인 자바 객체를 의미한다. 이로 인해 객체의 의존성이 간소화되고 유연성과 테스트, 유지보수의 용이성이 향상된다. 또한, Spring framework는 AOP (관점 지향 프로그래밍)를 지원하여 핵심 로직과 공통 기능을 분리할 수 있다. AOP를 활용하면 중복 코드를 줄이고 공통 기능을 모듈화하여 유지보수성을 향상시킬 수 있다.

MySQL Workbench에서 구현했다. 사용한 Tool은 다음과 같다.

② MySQL

MySQL은 오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)이다. 다양한 응용 프로그램을 위한 데이터 저장 및 관리를 제공한다. MySQL은 사용하기 쉽고 안정적이며 많은 개발자들과 조직에서 널리 사용되고 있다. MySQL은 SQL(Structured Query Language)을 사용하여 데이터베이스 관련 작업을 수행한다. SQL을 통해 데이터베이스에 데이터를 삽입, 조회, 수정, 삭제할 수 있으며, 복잡한 데이터 관련 작업도 처리할 수 있다.

MySQL은 다양한 운영체제에서 사용할 수 있으며, 확장성이 우수하고 높은 성능을 제공한다. 또한, 표준 SQL을 따르는 기능 외에도 트랜잭션 처리, 복제, 보안 등 다양한 고급 기능을 제공한다. 또한, MySQL Workbench를 통해 시각적인 방식으로 데이터베이스를 관리할 수 있다. MySQL Workbench는 데이터베이스 설계, 쿼리 작성, 성능 모니터링 등 다양한 기능을 제공하여 개발자가 효율적으로 MySQL 데이터베이스를 관리할 수 있도록 도와준다.

3. 결과

3.1. 결과물 시연

결과물 시연은 다음 유튜브 링크에서 확인할 수 있다.

https://www.youtube.com/watch?v=JsMg4bn_jKs&feature=youtu.be

3.2. 결과물 분석

3.2.1. 성능 평가

① 손동작인식

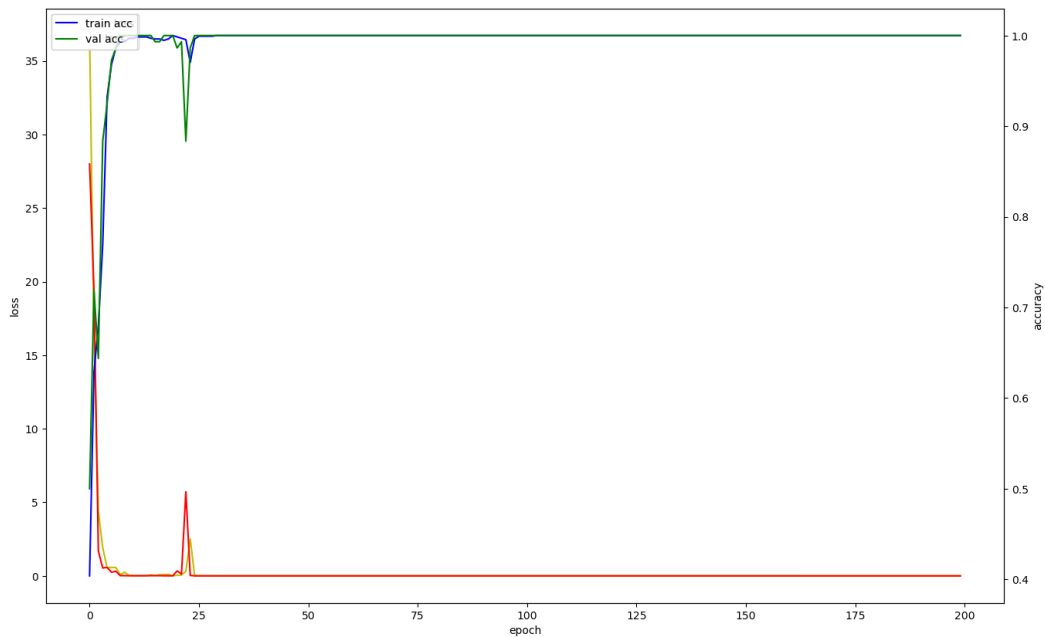


그림 2 : 모델의 학습 완료 그래프

위 그래프는 모델의 학습 과정에서의 손실(loss)과 정확도(accuracy)를 나타낸 그래프이다. train loss validation loss가 각각 노랑색과 빨강색으로 나타나고, train accuracy와 validation accuracy가 각각 파랑색과 초록색으로 나타난다. 학습이 진행될수록 정확도는 올라가고 손실은 낮아지는 것을 확인할 수 있다.

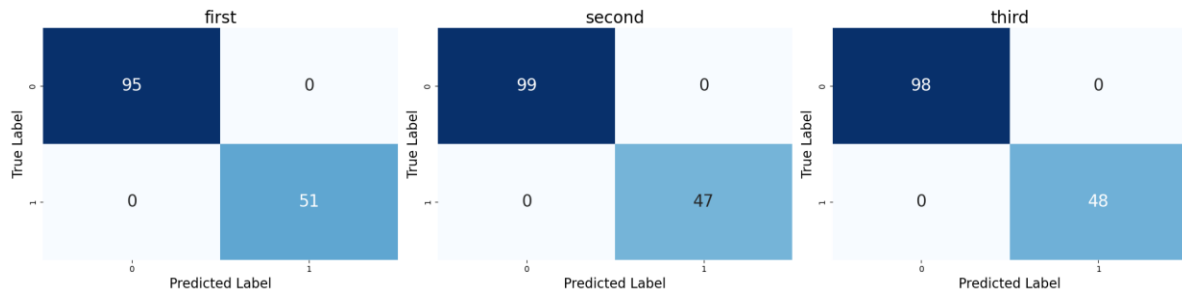


그림 3: 모델의 혼동행렬

모델의 성능을 평가하기 위해 혼동행렬을 계산하였다. 혼동행렬은 실제 라벨과 예측한 라벨이 얼마나 일치하는지를 나타내는 행렬이다. FP(False Positive, 실제로는 Negative인데 Positive로 예측한 경우), FN(False Negative, 실제로는 Positive인데 Negative로 예측한 경우), TP(True Positive, Positive를 Positive로 예측한 경우), TN(True Negative, Negative를 Negative로 예측한 경우)값들로 모델의 성능을 계산할 수 있다.

First 클래스의 경우, TP가 95, FN이 0, FP가 0, TN이 51이다. Second 클래스의 경우, TP가 99, FN이 0, FP가 0, TN이 47이다. Third 클래스의 경우, TP가 98, FN이 0, FP가 0, TN이 48이다. 따라서, 위 모델은 모든 클래스에서 FN 및 FP 값이 0이며 높은 정확도를 가지고 있음을 알 수 있다.

학습한 동작들 사이에서 혼동이 일어날 수 있는지에 대한 성능은 혼동행렬을 통해 그렇지 않다는 것을 알 수 있지만, 학습하지 않은 동작들을 학습한 동작 중 하나로 인식하는지에 대한 실험 또한 필요했다. 따라서, 1분동안 학습하지 않은 손동작들을 무작위로 계속 하고, 학습한 동작 중 하나로 인식하는 경우가 있는지를 실험했다. 실험을 10번 진행했지만, 학습한 동작들 중 하나로 인식한 경우는 없었다.

② 얼굴 인식

- 동일한 이미지에 대해 같은 결과값을 내는가?

해당 모델을 사용할 때, 2개의 입력 이미지를 받아, 유사도를 측정하는데, 같은 입력값에 대하여 얼마나 일관된 결과를 산출하는지 확인한다.

다음은 테스트에 사용된 두 이미지이며, 동일 인물이다. 모델 또한 동일 인물로 판단하였다. 동일한 이미지에 대해 같은 결과값을 내는지 확인하기 위해 50번 테스트를 진행하였고, 모두 같은 결과를 나타냈다.



얼굴 인식 1: 테스트한 인물 사진 1

```
C:\Users\Ki Hyeon\Desktop\파이썬\pythonworkspace>C:/Python310/python/ffinal.py
1/1 [=====] - 1s 519ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 259ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 70ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 76ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 89ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 81ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 64ms/step 동일한 인물로 판단됩니다.
2/2 [=====] - 0s 12ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 373ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 80ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 46ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 66ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 71ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 63ms/step 인물 일치율: 65.15 %
1/1 [=====] - 0s 70ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 59ms/step 인물 일치율: 65.15 %
2/2 [=====] - 0s 13ms/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 166ms/step 인물 일치율: 65.15 %
1/1 [=====] - 5s 5s/step 동일한 인물로 판단됩니다.
1/1 [=====] - 0s 142ms/step 인물 일치율: 65.15 %
인물 일치율: 65.15 %
동일한 인물로 판단됩니다.
```

얼굴 인식 2: 테스트 결과 1

- **짧은 인물들을 잘 구분해주는가?**

눈으로 보기에 비슷하다고 생각되는 인물들의 이미지도 잘 구분하는지 확인했다. 다음 두 이미지는 짧은 두 명의 연예인의 이미지이다. 육안으로 보기에도 굉장히 비슷한 모습을 보인다. 모델은 다른 인물로 정확히 판단했다.



얼굴 인식 3: 테스트한 인물 사진 2

```
C:\Users\Ki Hyeon\Desktop\파이썬\pythonworkspace>C:/Python310/python.exe "c:/Users/Ki Hyeon/Desktop/파이썬/pythonworkspace/ffinal.py"
1/1 [=====] - 1s 566ms/step
1/1 [=====] - 0s 307ms/step
1/1 [=====] - 0s 87ms/step
1/1 [=====] - 0s 92ms/step
1/1 [=====] - 0s 65ms/step
1/1 [=====] - 0s 75ms/step
1/1 [=====] - 0s 87ms/step
1/1 [=====] - 0s 88ms/step
1/1 [=====] - 0s 86ms/step
3/3 [=====] - 0s 15ms/step
1/1 [=====] - 0s 419ms/step
1/1 [=====] - 0s 121ms/step
1/1 [=====] - 0s 107ms/step
1/1 [=====] - 0s 73ms/step
1/1 [=====] - 0s 108ms/step
1/1 [=====] - 0s 100ms/step
1/1 [=====] - 0s 93ms/step
1/1 [=====] - 0s 75ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 92ms/step
3/3 [=====] - 0s 17ms/step
1/1 [=====] - 0s 101ms/step
1/1 [=====] - 6s 6s/step
1/1 [=====] - 0s 201ms/step
인물 일치율: 48.16 %
다른 인물로 판단됩니다.
```

얼굴 인식 4: 테스트 결과 2

- 사진의 각도가 다른 동일 인물을 동일 인물로 판단하는가?

다음 두 이미지는 얼굴의 각도가 다른 동일 인물의 모습이다. 해당 이미지들에 대해서도 동일 인물이라고 판단하는지 확인했다. 모델은 동일 인물로 판단했다.



얼굴 인식 5: 테스트한 인물 사진 3

```
C:\Users\Ki Hyeon\Desktop\파이썬\pythonworkspace>C:/Python310/python.exe "c:/Users/Ki Hyeon/Desktop/파이썬/pythonworksp
ace/ffinal.py"
1/1 [=====] - 2s 2s/step
1/1 [=====] - 1s 884ms/step
1/1 [=====] - 0s 421ms/step
1/1 [=====] - 0s 243ms/step
1/1 [=====] - 0s 167ms/step
1/1 [=====] - 0s 125ms/step
1/1 [=====] - 0s 92ms/step
1/1 [=====] - 0s 92ms/step
1/1 [=====] - 0s 75ms/step
1/1 [=====] - 0s 76ms/step
1/1 [=====] - 0s 79ms/step
1/1 [=====] - 0s 69ms/step
1/1 [=====] - 0s 66ms/step
1/1 [=====] - 0s 85ms/step
10/10 [=====] - 1s 22ms/step
1/1 [=====] - 0s 394ms/step
1/1 [=====] - 0s 77ms/step
1/1 [=====] - 0s 77ms/step
1/1 [=====] - 0s 73ms/step
1/1 [=====] - 0s 72ms/step
1/1 [=====] - 0s 70ms/step
1/1 [=====] - 0s 107ms/step
1/1 [=====] - 0s 67ms/step
2/2 [=====] - 0s 10ms/step
1/1 [=====] - 0s 84ms/step
1/1 [=====] - 7s 7s/step
1/1 [=====] - 0s 215ms/step
인물 일치율: 58.88 %
동일한 인물로 판단됩니다.
```

얼굴 인식 6: 테스트 결과 3

- 악세사리의 착용 유무가 바뀌어도 동일 인물로 판단하는가?

다음 두 이미지는 안경을 쓰고 벗은 동일 인물의 모습들이다. 안경과 같은 악세사리의 유무가 동일 인물 판단에 영향을 주는지 확인했다. 모델은 동일 인물로 판단했다.



얼굴 인식 7: 테스트한 인물 사진 4

```
C:\Users\Ki Hyeon\Desktop\파이썬\pythonworkspace>C:/Python310/python.exe "c:/Users/Ki Hyeon/Desktop/파이썬/pythonworkspace/ffinal.py"
1/1 [=====] - 1s 783ms/step
1/1 [=====] - 0s 401ms/step
1/1 [=====] - 0s 189ms/step
1/1 [=====] - 0s 103ms/step
1/1 [=====] - 0s 60ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 45ms/step
1/1 [=====] - 0s 37ms/step
1/1 [=====] - 0s 36ms/step
1/1 [=====] - 0s 44ms/step
1/1 [=====] - 0s 29ms/step
1/1 [=====] - 0s 23ms/step
1/1 [=====] - 0s 24ms/step
1/1 [=====] - 0s 23ms/step
7/7 [=====] - 0s 8ms/step
1/1 [=====] - 0s 138ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 24ms/step
1/1 [=====] - 0s 51ms/step
1/1 [=====] - 0s 37ms/step
1/1 [=====] - 0s 26ms/step
1/1 [=====] - 0s 25ms/step
1/1 [=====] - 0s 23ms/step
2/2 [=====] - 0s 4ms/step
1/1 [=====] - 0s 31ms/step
1/1 [=====] - 2s 2s/step
1/1 [=====] - 0s 68ms/step
인물 일치율: 61.67 %
동일한 인물로 판단됩니다.
```

얼굴 인식 8: 테스트 결과 4

③ FrontEnd & BackEnd

구현한 기능이 정상적으로 작동하는지 확인하기 위해 아래 테스트 케이스들을 진행했다. 7개의 테스트 케이스 모두 정상적으로 작동하였다.

(1) Test_Case1

항목	내용
Test Case Name	회원가입 확인
Test Case ID	TC-user-01
Test Object	Computer(Laptop), back-end, database
Test Case Description	회원가입 시 입력한 정보들이 제대로 저장
Test Step	1. Computer(Laptop)을 통해서 웹에 접속하여 회원가입 2. 이름, 이메일, 비밀번호 등 사용자의 정보 입력 3. database상에 유저 정보가 제대로 입력되었는지 확인
E.R(expected result)	database 상에 제대로 입력된 유저정보

(2) Test_Case2

항목	내용
Test Case Name	웹에서 파일 업로드 확인
Test Case ID	TC-upload-01
Test Object	Computer(Laptop), back-end
Test Case Description	Computer(Laptop)를 통해 파일을 업로드
Test Step	1. 웹에 접속하여 Computer(Laptop)에 저장 되어있는 파일을 업로드 한다. 2. 업로드 한 파일이 back-end server로 제대로 전달되는지 확인
E.R(expected result)	업로드 한 파일이 back-end server로 전달

(3) Test_Case3

항목	내용
Test Case Name	앱에서 갤러리를 통한 사진 업로드 확인
Test Case ID	TC-upload-02
Test Object	smart phone, back-end
Test Case Description	Smart Phone 갤러리를 통해 이미지가 제대로 업로드 되어야 한다.
Test Step	1. 앱에 접속하여 Smart Phone의 갤러리를 통해서 사진을 고른다. 2. 고른 사진이 back-end server로 제대로 전달되는지 확인
E.R(expected result)	갤러리에서 고른 사진이 back-end server로 전달

(4) Test_Case4

항목	내용
Test Case Name	손동작 인식 암호 Dataset 구성 확인
Test Case ID	TC-motion-01
Test Object	Computer(Laptop), motion model, back-end
Test Case Description	손동작 인식 암호가 motion model을 통하여 dataset 구성이 제대로 되는지 확인
Test Step	1. 손동작 인식 암호 dataset구성 확인 2. 손동작을 웹캠에 비추어 dataset 생성 3. dataset이 서버에 저장되는지 확인
E.R(expected result)	Dataset이 back-end server에 저장되는 지 확인

(5) Test_Case5

항목	내용
Test Case Name	손동작 인식 암호 모델 생성 확인
Test Case ID	TC-motion2-02
Test Object	Computer(Laptop), motion model, back-end
Test Case Description	손동작 인식 암호가 motion model을 통하여 인식이 제대로 되는 지 확인
Test Step	1. 손동작 인식 암호 모델 생성 파일 실행 2. 모델이 서버에 저장되는지 확인
E.R(expected result)	Dataset이 back-end server에 저장되는 지 확인

(6) Test_Case6

항목	내용
Test Case Name	얼굴 인식 암호 해제 여부
Test Case ID	TC-face_solve-01
Test Object	smart phone, face model, back-end
Test Case Description	얼굴 인식 암호가 face model을 통하여 인식을 제대로 되는지 확인
Test Step	1. smart phone을 통해 사진 업로드 2. 얼굴 인식 암호 해제 model을 통한 결과 확인 3. 결과가 앱 상에서 잘 표시되는지 확인
E.R(expected result)	결과가 back-end server에 저장되는 지 확인

(7) Test_Case7

항목	내용
Test Case Name	손동작 인식 암호 해제 여부
Test Case ID	TC- motion_solve -01
Test Object	Computer(Laptop), smart phone, motion model, back-end, database
Test Case Description	손동작 인식 암호가 motion model을 통하여 인식을 제대로 되는지 확인
Test Step	1. 손동작 인식 암호 해제 파일 실행 2. 손동작을 웹캠에 비추어 암호 해제 결과 확인 3. 결과가 웹과 앱상에서 잘 표시되는지 확인
E.R(expected result)	결과가 back-end server에 저장되는 지 확인

3.2.2. 설계와의 비교

① 손동작 인식

암호 설정 시간을 7초 정도로 설계했었지만, 데이터가 부족해 정확도가 떨어진다고 판단했다. 따라서 암호 설정 시간을 20초로 늘려 데이터를 늘려주었고, 정확도를 향상 시킬 수 있었다.

또한 판단 로직도 바뀌었다. 90% 이상의 유사도를 보인 동작들을 배열에 넣고 5개의 동작이 연속으로 배열에 들어오면 해당 동작을 했다고 판단하도록 설계했었다. 그러나 구현을 하고 더 높은 정확도가 필요하다고 판단하고, 연속되는 30개의 동작들 중 25개가 한 동작일 경우에 해당 동작을 했다고 판단하도록 수정하였다.

학습 과정 및 다른 부분들은 설계대로 진행하였다.

② 얼굴 인식

본 프로젝트 설계 당시 얼굴 인식 모델은 opencv를 통해 이미지를 읽고, LBPHFaceRecognizer라는 메소드를 통해 얼굴 유사도를 측정하도록 설계하였다. opencv는 영상 및 이미지 처리에 용이한 대표적인 라이브러리로서 본 프로젝트에서 필요한 동일 인물 판단에도 사용하면 좋은 결과를 얻을 수 있을 것이라고 생각하였다. 그러나 구현을 시작하여 이미지를 입력 받아 유사도를 측정하여 동일 인물을 판단할 때, 정확도가 조금 떨어지는 모습을 보였다. 동일한 입력 값에 대해서도 종종 다른 결과를 반환하고, 다른 인물에 대해 동일 인물이라고 판단하는 결과를 나타내는 경우가 있어, 모델을 변경하였다. 변경된 모델은 CNN딥러닝 구조를 이용한 MTCNN 모델을 적용해, 얼굴 감지를, keras의 facnet 모델을 적용해 얼굴 특징 검출을 진행하여 유사도를 측정하여 동일 인물을 판단하게 된다.

③ FrontEnd

설계문서대로 구현하였다.

④ BackEnd

서비스에서 주고받는 파일들을 AWS S3에 저장하는 것으로 계획했지만, 변경되었다.

- 성능 및 지연 시간

AWS S3와 같은 클라우드 스토리지는 인터넷 연결을 통해 액세스되므로 데이터 전송과 응답 시간에 영향을 받을 수 있다. 이는 프로젝트의 성능 요구 사항과 관련하여 지연 시간이나 대역폭 제한 등의 문제를 야기할 수 있다. 따라서, 데이터를 로컬 환경에 저장하여 더 빠른 성능을 확보하였다.

- 비용 관리 및 예산 제약

AWS S3 클라우드 서비스는 일반적으로 사용량에 따라 비용이 발생한다. 비용을 절감하고 예산을 효율적으로 관리하기 위해 데이터를 로컬 환경에 저장하는 것으로 선택하였다.

3.3. 제약사항

3.3.1. 손동작 인식

① 컴퓨터의 성능이 충분하지 않을 경우

암호 손동작 설정, 손동작 유사도 검사 과정에서 높은 정확도를 얻기 위해 시퀀스 길이를 90으로 설정했다. 즉 90프레임 단위로 데이터를 처리하도록 했다. 시퀀스 길이를 높이면 유사도 검사에 대한 정확도를 올릴 수 있지만, 메모리에 무리가 갈 수도 있다. 실행환경에 따라 조정이 필요할 수도 있다.

또한 1초당 30프레임을 읽어 들일 수 있다는 가정 하에 3초 단위로 데이터를 처리하기 위해 시퀀스 길이를 90으로 설정했다. 그러나 컴퓨터의 성능에 따라 1초당 30프레임이 아닌, 그보다 훨씬 적거나 많은 프레임을 읽어 들일 수 있다. 이러한 경우, 목표로 하는 3초가 아닌 더 짧거나 긴 단위로 데이터를 처리하게 되어 유사도 검사에 대한 정확도에 영향을 줄 수 있다.

② 암호로 설정한 손동작들이 비슷할 경우

손동작 유사도 측정 단계는 예측한 손동작들을 배열에 계속해서 넣고, 일정 비율 이상이 같은 손동작이면 최종적으로 그 손동작을 했다고 판단한다. 예를 들어 first 손동작과 third 손동작이 비슷한 손동작을 포함하고 있다면, 그 부분 때문에 두 손동작이 교차로 배열(판단 로직의 predict_action 배열)에 들어가 어떤 손동작을 했는지 판단하지 못할 수도 있다. 혹은 둘 중 한 손동작만 배열에 들어가 의도와 다른 손동작이 인식되거나 다른 손동작으로 인식해 암호를 해제하지 못할 수도 있다. 따라서 암호 손동작을 설정하는 유저에게도 최대한 다른 손동작들과 최대한 겹치지 않는 손동작을 하도록 유도해야한다.

③ 느리게 변화하는 손동작을 암호 손동작으로 설정하는 경우

본 프로젝트에서는 사진처럼 특정 순간의 손의 정보로 판단 하는게 아닌, 일정 시간동안 손이 어떻게 변화하는지를 고려한다. 이를 위해 시간의 흐름에 따른 순서를 고려해 데이터를 처리한다. 그러나 암호 손동작을 설정하는 과정에서, 데이터를 처리하는 단위와 비슷하거나 더 긴 시간동안 손동작의 변화가 없는 경우, 이 때 처리한 데이터들이 유사도 검사에 영향을 줄 수 있다. 손동작의 움직임들을 고려한 유사도 판단을 의도했지만, 움직이지 않았을 때의 데이터들 때문에 의도와는 다르게 해당 동작을 했다고 판단해버릴 수도 있다.

④ 손의 관절들이 웹캠에서 보이지 않을 경우

웹캠에서 보이는 손의 관절들은 해당 좌표를 저장하지만, 보이지 않는 손의 관절들은 예측값을 저장한다. 보이고, 보이지 않는 정도로 가중치를 주긴 하지만, 해당 손동작 정보에 대한 신뢰도가 떨어지는 것은 사실이다. 해당 손동작으로 학습을 진행하고, 손동작 유사도 검사 단계에서도 웹캠에서 보이지 않는 손의 관절의 경우 예측값을 사용하기 때문에 유사도 검사가 원활하게 이루어지지 않을 수 있다. 따라서 암호 손동작을 설정하는 유저에게 최대한 손바닥 혹은 손등이 잘 보이는 모션을 하도록 유도해, 손의 관절들을 잘 파악할 수 있도록 해야한다.

⑤ 웹캠 이미지에 다른 손이 등장할 경우

본 프로젝트의 손동작 인식에는 한 손만을 사용한다. 따라서 한 손만의 데이터를 처리하는데, 손동작 암호 설정 또는 해제 과정에서 한 손이 아닌 두 손이 보여져 의도하지 않은 손이 인식되는 경우가 있을 수 있다. 이러한 경우, 의도하지 않은 손동작 데이터가 처리되어, 유사도 검사에 대한 정확도에 영향을 줄 수 있다.

3.3.2. 얼굴 인식

① 여러 인물이 인식되는 사진일 경우

본 프로젝트에서 얼굴 인식은 기본적으로 한 이미지에서 한 인물이 있는 경우로 가정한다. 기본적으로 문서 보안이라는 주제와 맞게, 인식을 시도할 때 인물이 혼자 있다는 가정하에 진행하기 때문이다. 따라서, 여러 인물이 있는 사진의 경우 인식이 불가능하여 재시도를 해야한다.

② 화질이 낮은 혹은 흔들리는 사진일 경우

얼굴 인식은 딥러닝 모델을 사용하여 이미지에서 얼굴 특징 벡터를 추출해 해당 값으로 유사도를 측정한다. 사진이 흔들려 얼굴 영역을 감지하지 못하거나, 얼굴 영역을 감지해내더라도, 기존과 다르게 판단한다. 화질이 현저히 낮은 경우에도 마찬가지로 영역 감지를 아예 못하거나, 감지하더라도 다르게 판단할 수 있다. 이러한 경우 인식 재시도를 해야한다.

3.3.3. FrontEnd

① Fronted와 Backend가 동일한 네트워크에 접속한 것이 아닌 경우

React와 Spring 서버 간의 통신을 위해서는 동일한 네트워크에 있어야 한다. 이것은 React 앱이 웹 브라우저에서 실행되는 클라이언트 측 코드이고, Spring 서버가 Backend에서 실행되는 서버 측 코드이기 때문이다. React는 브라우저에서 실행되기 때문에 클라이언트는 서버에 HTTP 요청을 보낼 수 있어야 한다. 따라서 React 앱과 Spring 서버는 동일한 네트워크에 있어야 서로 통신할 수 있다.

마찬가지로, 안드로이드 앱과 Spring 서버 간의 통신을 위해서도 동일한 네트워크에 있어야 한다. 안드로이드 앱은 모바일 기기에서 실행되는 클라이언트 앱이며, Spring 서버는 서버 측에서 실행되는 Backend이다. 안드로이드 앱은 모바일 기기에서 HTTP 요청을 서버로 보내야 하므로 네트워크 연결이 필요하다. 안드로이드 앱과 Spring 서버는 동일한 네트워크에 있어야 서로 통신할 수 있다.

② 파일 형식 제한

업로드하는 문서 파일은 PDF 형식으로 제한된다. 다른 파일 형식을 사용하여 전송할 수 없으며, 서버 및 클라이언트 양측에서 PDF 파일 형식을 지원하고 있다. 파일을 Blob 형태로 전송하기 때문에, 클라이언트 측에서 파일을 Blob 형식으로 변환하여 전송하며 서버에서도 동일하게 Blob으로 처리한다.

3.3.4. BackEnd

① 확장성 및 용량 관리

AWS S3이 아닌 로컬 스토리지를 선택하면서, 스토리지에 대한 용량 한계 및 확장성에 대한 제약이 생긴다. 사용자가 증가하여 파일의 수가 증가할수록 추가적인 스토리지 장치 및 용량 관리 계획이 필요하다.

② Fronted와 Backend가 동일한 네트워크에 접속한 것이 아닌 경우

앞서 Frontend에서 설명한 사항과 같은 제약사항이므로 생략한다.

4. 작업 진행 방법

4.1. 작업 분담 구조

4.1.1. 선형 책임 도표

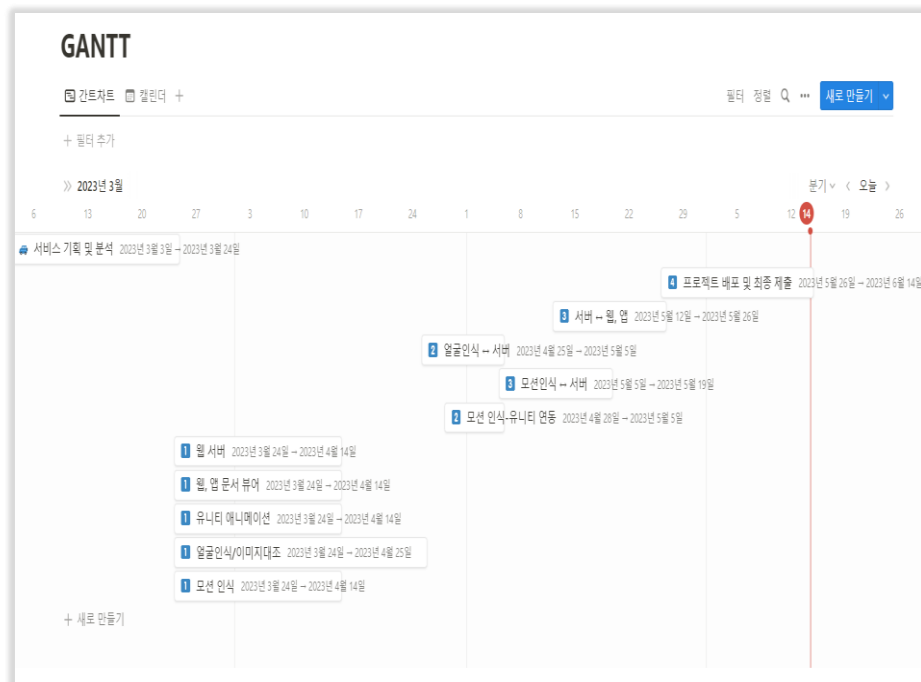
선형 책임 도표	팀원 1 (김기현)	팀원 2 (우신영)	팀원 3 (유병민)	팀원 4 (이승택)
1.0 주제 아이디어 생성	◎	◎	◎	◎
1.1 주제 선정	◎	◎	◎	◎
1.2 최종 주제 결정	◎	◎	◎	◎
2.0 프로젝트 문제 정의	◎	◎	◎	◎
2.1 프로젝트 목적 설정	◎	◎	◎	◎
3.0 관련 기술 조사	○	○	◎	◎
3.0.1 관련 기술 동향 파악	○	○	◎	◎
3.0.2 관련 기술의 수요 및 전망 파악	○	○	◎	◎
3.1 관련 제품 시장 조사	○	○	◎	◎
4.0 구현 가능성 분석	◎	○	○	◎
4.1 아이디어 타당성 분석	◎	◎	◎	◎
4.2 아이디어 최종 결정	◎	◎	◎	◎
5.0 아이디어 기반 요구사항 분석	◎	◎	◎	◎
5.0.1 기능 요구사항 정의		◎	◎	
5.0.2 성능 규격 조건 설정	◎			◎
5.1 요구사항 세부사항 결정		◎	◎	
6.0 동작 환경 분석	◎			◎
6.1 개발 환경 분석	◎	◎	◎	◎
6.2 개발 비용 분석	◎			◎
6.3 사회적 측면 분석		◎	◎	
7.0 평가 항목 설정 및 평가	◎	◎	◎	◎
7.1 대안 최종 선택	◎	◎	◎	◎
8.0 설계 목표 설정			◎	

8.1 설계 및 도면 작성		◎	◎	
8.2.1 손동작 인식 설계 및 도면 작성				◎
8.2.2 이미지 인식 설계 및 도면 작성	◎			
8.3 테스트 케이스 작성		◎	◎	
9.0 일정 요약		◎		
9.1 설계 문서 작성	◎	◎	◎	◎
9.2 구현 방법 제안	◎	◎	◎	◎
11.0 구현	◎	◎	◎	◎
11.0.1 손동작 인식 파트 구현			○	◎
11.0.2 이미지 인식 구현	◎		○	
12.1 테스트 케이스 기반 테스트		◎	◎	
12.2 결함 수정 및 회귀 테스트	◎			◎
13.0 프로젝트 결과 작성	◎	◎	◎	◎
13.1 최종 보고서 작성	○	○	○	◎
14.0 진행 상황 모니터링				◎

* ◎: 주 책임자 ○: 보조 책임자

4.2. 설계 일정 및 역할 분담

4.2.1. 간트 차트



간트 차트 1: 간트 차트

5. 참고문헌

- <https://docs.spring.io/spring-framework/reference/>
- https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
- <https://www.tensorflow.org/?hl=ko>
- <https://ko.legacy.reactjs.org>