온라인 시험을 위한 부정행위 방지 시스템의 설계 및 구현

임정민, 이강복, 하인혜

Design and implementation of cheating detection systems for online testing

Jungmin Lim, Gangbok Lee, Inhye ha

구체적 날짜에 따른 프로젝트 구현 계획

<table>
<tr><td></td><td colspan="3">내용</td></tr>
<tr><td>3월</td><td colspan="3">프로젝트 주제 선정</td></tr>
<tr><td>4월</td><td colspan="3">선행 연구 분석
프로젝트 설계</td></tr>
<tr><td>5월</td><td colspan="2">첫 화면(소개화면) 구현
상단 메뉴바와 사이드바 구현
로그인, 회원가입 구현</td><td>Data Preprocessing
(데이터 전처리)</td><td>Data Preprocessing
(데이터 전처리)</td></tr>
<tr><td rowspan="2">6월</td><td>Supervisor page</td><td>Candidate page</td><td rowspan="3">Model Training
(모델 설계 및 구축)</td><td rowspan="2">Verification Model 구축</td></tr>
<tr><td>출제 페이지 구현
질문게시판 구현
설정 구현</td><td>질문게시판 구현
결과게시판 구현
설정 구현</td></tr>
<tr><td>7월</td><td colspan="2">감독관과 응시자 상호 화면공유 구현
감독관과 응시자 상호 채팅 시스템 구현</td><td rowspan="2">Anti-spoofing Model 구축
OCR Engine 구현</td></tr>
<tr><td>8월</td><td colspan="2">TEST 화면 구현</td><td>모델 학습 및 성능 비교</td></tr>
<tr><td>9월</td><td colspan="2">웹 디자인 설정
Django 내장 DB에서 MongoDB로 연동</td><td>Gaze Estimation model
웹 서버에 적용</td><td>Verification Model
웹 서버에 적용</td></tr>
<tr><td>10월</td><td colspan="2">예외처리 및 디버깅
React+ Django 프로젝트 AWS로 배포</td><td>예외처리 및 전체적인 프로그램 TEST</td><td>예외처리 및 전체적인 프로그램 TEST</td></tr>
<tr><td>11월</td><td colspan="3">최종 결과물 제출</td></tr>
</table>

Ⅰ. 개발 환경

OS	Windows /Linux
Framework	React, Django, MongoDB, AWS,
IDE	Visual Studio Code, Pycharm Jupyter notebook
개발 언어	Python, Javascript

Ⅱ. 프로젝트 설계

1. 첫 소개화면 구현

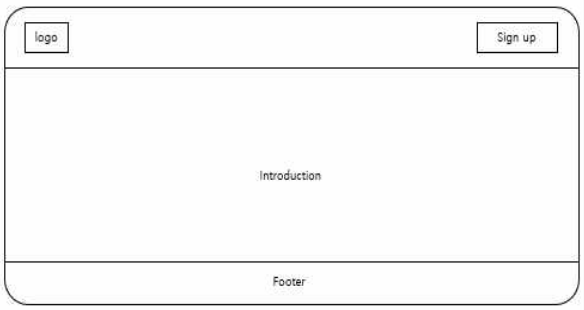

그림 1. 첫 화면 디자인 개요

첫 화면은 로고와 Cheating Detection을 소개하는 내용을 중앙에 넣고, 회원가입 혹은 로그인을 할 수 있는 버튼을 우측 상단에 집어넣는다. 로고를 클릭하게 되면 언제든지 첫 화면(로그인 중이라면 응시자, 감독관 첫 화면)으로 돌아갈 수 있게 한다. 이후 소개 글의 스크롤을 내리면 로고와 회원가입이 있는 Navigation Bar를 스크롤과 동시에 내려오게 하여 소개 글을 읽다가도 손쉽게 로그인 페이지로 가게 구현한다. Footer에는 feedback, contact us, 프로젝트 설계 등을 열람할 수 있는 링크를 넣는다. 그림1에서는 첫 화면에 대한 대략적인 디자인 개요를 나타내고 있다.

2. 모바일 화면(Sidebar) 구현

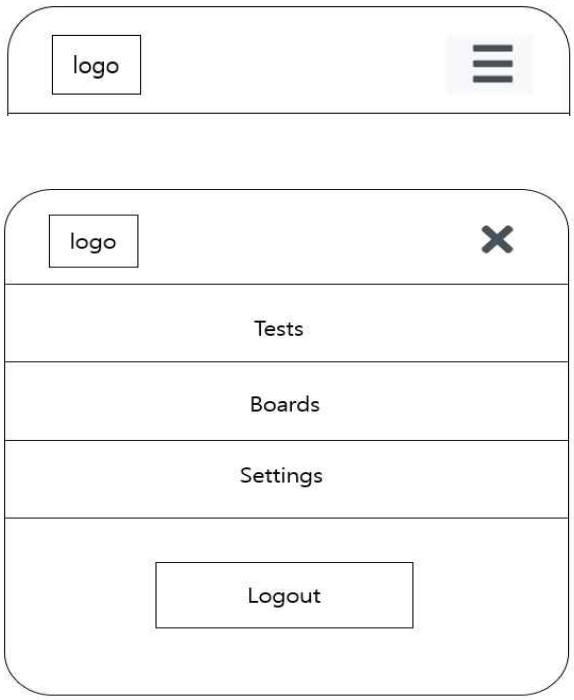

그림2. 모바일 화면에서 웹사이트에 접속하였을 때

우리는 모바일 환경, 모바일이 아니더라도 화면이 작은 Client가 이 웹페이지에 접근하는 것에 대한 호환성을 늘려야 한다고 생각한다. 쉽게 얘기해, 모바일 환경에서 버튼을 누르고 이동하는 데 있어서 오류가 생기면 Client가 불편함을 느낀다고 판단하였다. 이 웹사이트는 모바일 환경을 고려하고 만든 사이트는 아니므로 Application으로 제작하지는 않지만 최소한의 호환성을 보안하려고 한다.

그림2는 모바일 Application이 아닌 모바일 Chrome에서 접속하였을 때 화면을 예시로 나타낸 것이다. 다음과 같은 방식은 React Hooks + CSS media queries를 이용하여 max-width를 설정하여 구현할 수 있다.

CSS에서 @media screen and (max-width: (value))를 사용하여 Html에서 <link> 요소에 특성이 조건에 맞을 때 기존의 css 대신 @media screen css를 가져온다. 따라서 적절한 max-width 값을 설정하여 모바일 환경에서도 클릭에 불편함이 없게 하였다.

3. 상단 메뉴바 구현

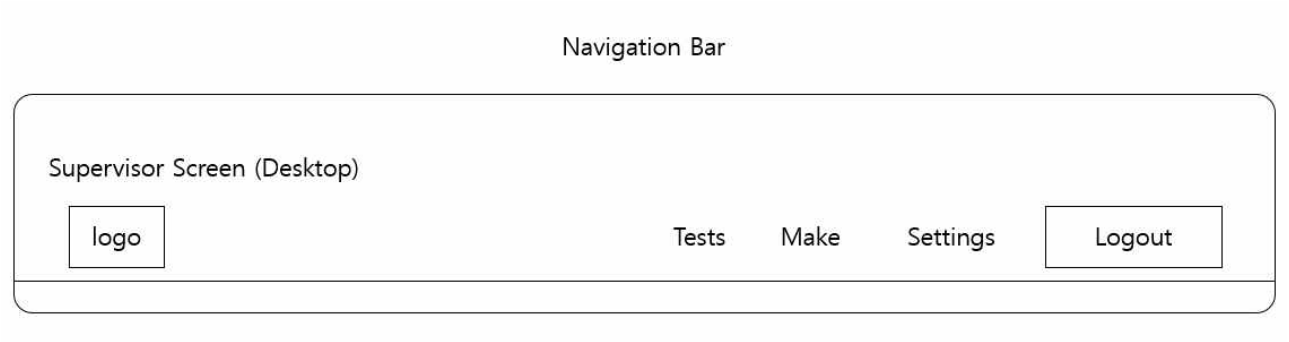

그림3. 관리자로 로그인 하였을 때 나오는 상단 메뉴바

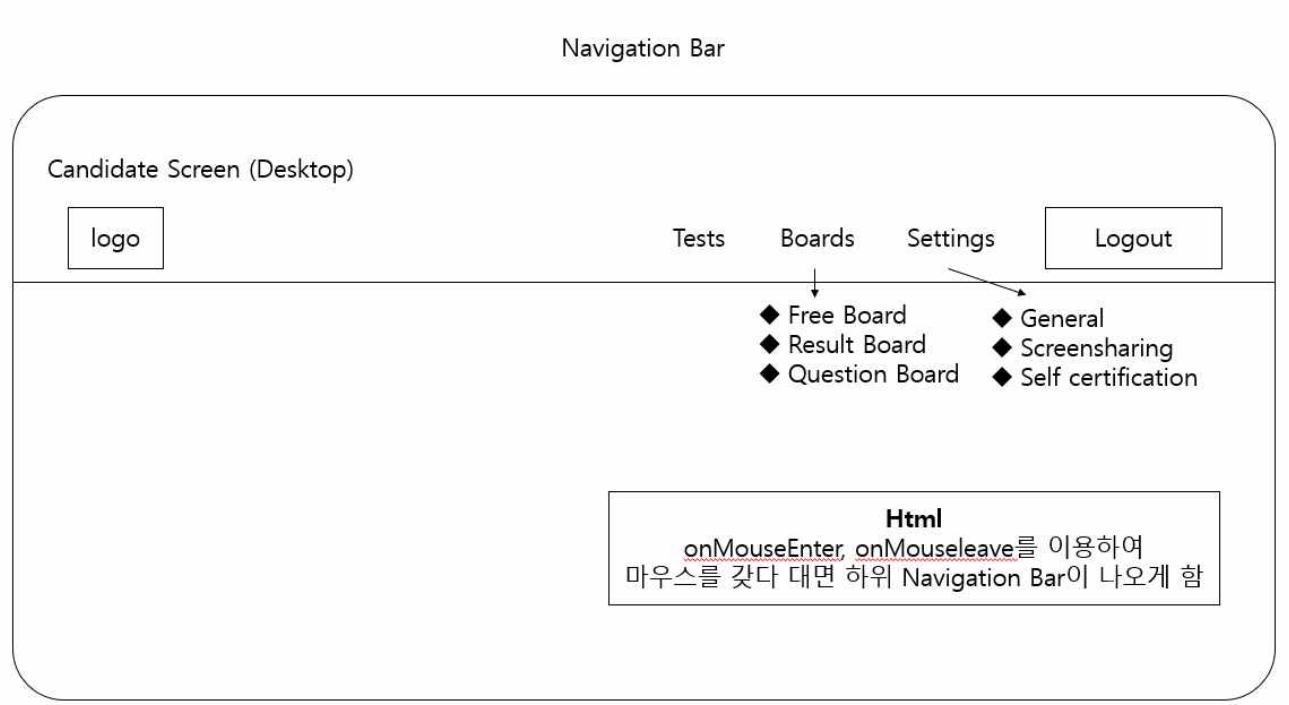

그림4. 응시자로 로그인 하였을 때 나오는 상단 메뉴바

관리자의 화면과 시험 응시자의 화면을 각각 따로 구현하여 각자 자신의 목적에 맞게 사용할 수 있게 만들고, 웹 페이지의 직관성을 높여서 관리자가 응시자의 화면으로 가거나 응시자가 관리자의 화면으로 가는 것을 방지한다.

주목해야 할 점은, 그림 4의 Html의 onMouseEnter, onMouseleave와 React Hooks를 이용하여 Boards와 Settings의 구체적인 Navigation을 마우스를 갖다 댔을 때 작동하고 평소에는 보이지 않게 설정하는 것이다. 그리고 다음의 메뉴바를 React HashRouter의 상단에 위치시켜 어느 화면을 가더라도 상단 메뉴바가 나타나게 구현한다.

마찬가지로 로고를 클릭하면 응시자의 경우 응시자의 첫 화면으로 가게 되고, 관리자의 경우 관리자의 첫 화면으로 가게 된다. 로그아웃 버튼을 누르면 첫 소개화면으로 이동한다.

4. 로그인, 회원가입 구현

로그인 화면과 회원가입에서 짚어야 할 것은 DataBase 연동, DB로 데이터를 보내는 방법, 토큰화이다.

첫 번째로 DataBase 연동을 하기 위해 Django의 내장된 DB인 SQLite3와 연동한다. 연동하기 위해 Django의 Admin 페이지에 User Model을 만들고, rest_framework를 설치하여 REST API로 React와 연동한다. React에서는 fetch로 JSON파일을 받아와서 확인한다.

DB로 데이터 보내는 방법은 마찬가지로 REST API를 POST 방식으로 Django 서버에 요청하면 Django DB에 저장이 되는 형식이다.

토큰화는 Django에서 authtoken을 이용하여 token으로 변환하고 실제로는 token이 같은지

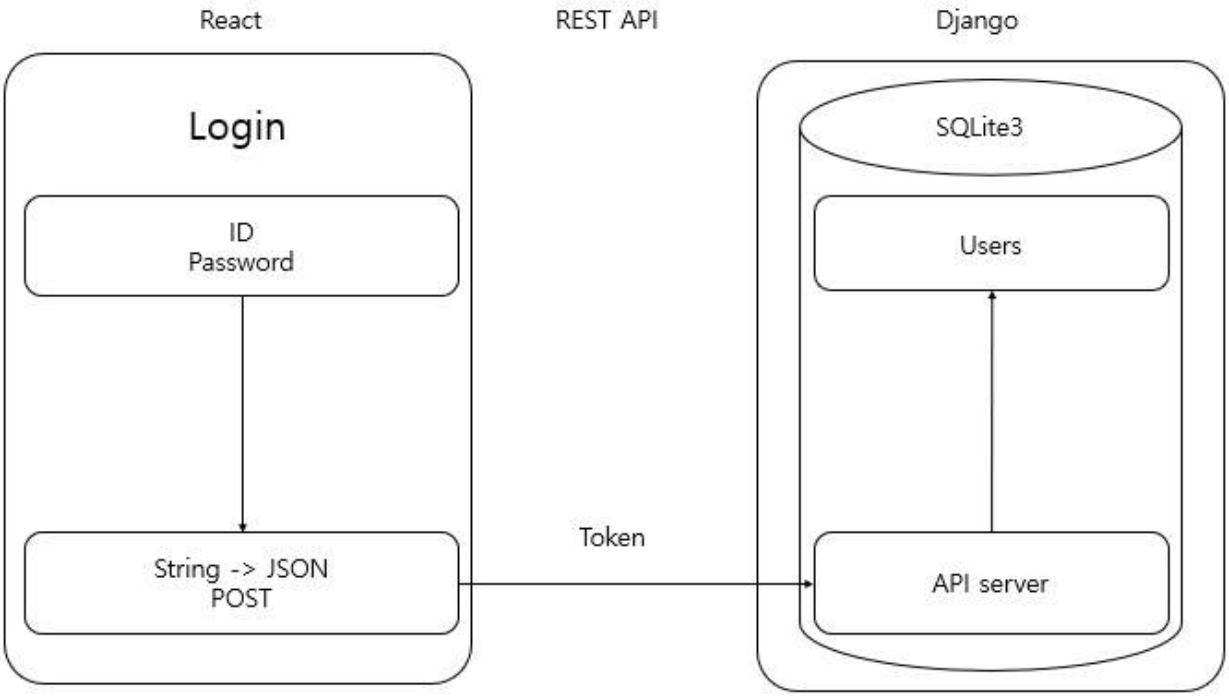

그림 5. Django와 React를 REST API로 연동하는 방법

확인하는 형태로 진행한다.

그림5에서는 React와 Django를 REST API를 활용하여 연동하는 방법을 도식화하여 나타내고 있다.

5. 게시판 구현

게시판도 로그인, 회원가입과 마찬가지로 REST API를 이용하여 Django와 소통하는 방식을 이용하지만 Token을 사용하지는 않는다. Password와 같이 보안상 문제되는 것이 비교적 적기 때문이다. 게시판에서는 많은 DB의 Seed들을 모두 출력하기 때문에 이것들을 10개, 혹은 15개 정도의 숫자로 분할하고 정복할 필요가 있다. 예를 들어 게시판의 게시물이 n개라고 하고 m개씩 분할한다고 하면 page의 개수를 n/m+1형태로 생성하고 첫 번째 페이지는 Previous를 제외하고 마지막 페이지는 Next를 제외한 뒤 Next와 Previous 버튼을 클릭하면 적절한 m개의 DB를 GET 방식으로 RESTAPI로 호출한다. 이후 'react-router-dom'의 <Link>를 통하여 게시물을 클릭하면 그 안의 content를 출력하게 한다. 결과 게시판의 경우 학생이 자신만의 점수를 확인해야 하므로 암호화를 통해 타인의 점수를 볼 수 없게 구현한다.

6. Test 화면 구현

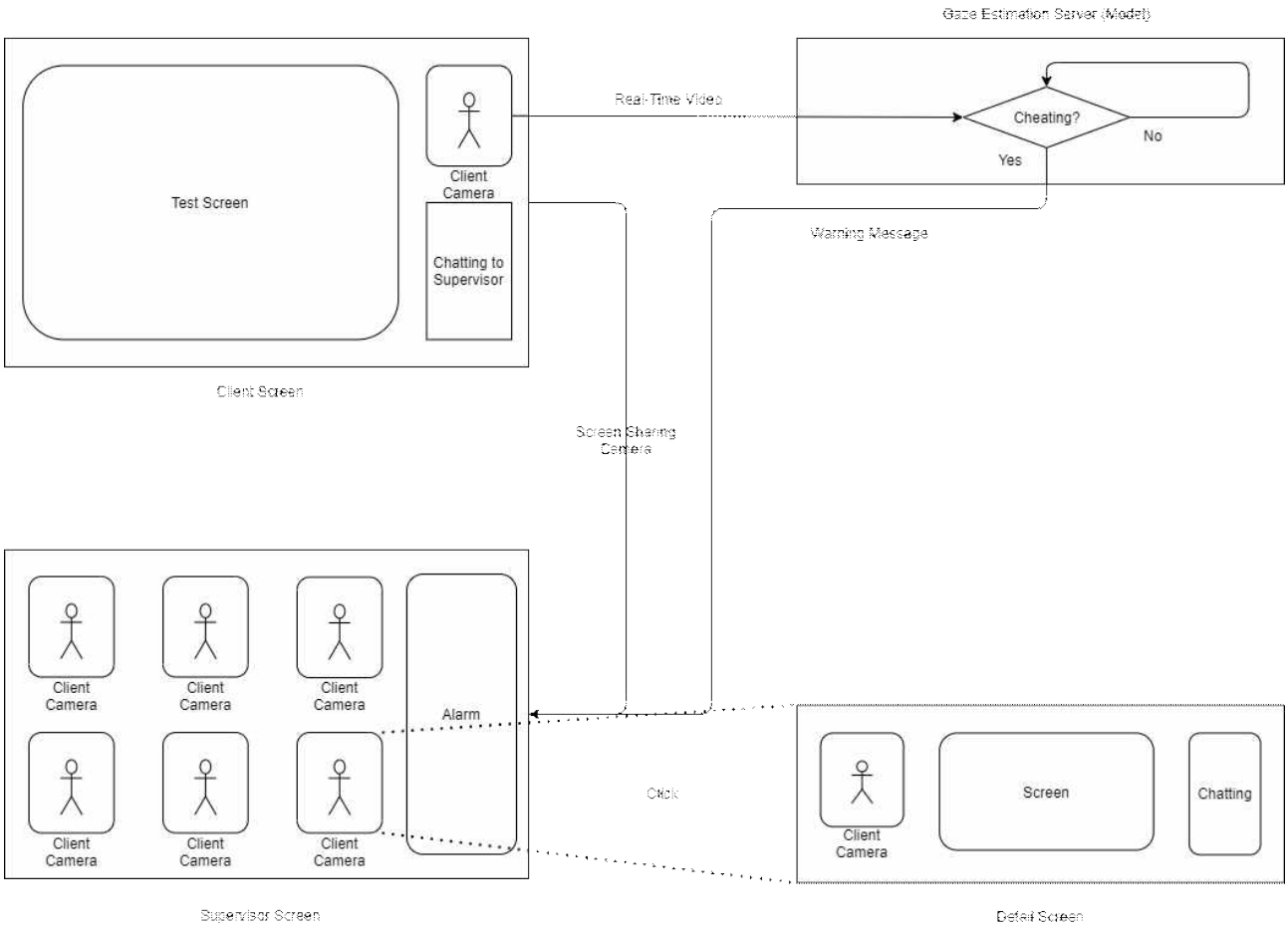

그림6. Test 화면 디자인 및 설계

우리는 Test 화면을 구현하기 위해 응시자의 화면과 시험 감독관의 화면을 두 개 만들어야 할 필요가 있다. 따라서 감독관은 여러 명의 응시자를 동시에 볼 수 있게 하고, 응시자는 자신의 화면과 시험 문제지를 볼 수 있게 설계하였다. 화면공유와 채팅은 WebRTC로 해결하고, Eye-Tracking 문제는 Gaze Estimation Model로 해결하려고 한다.

추가로, 위의 Gaze Estimation Model은 시험을 보는 도중 응시자의 카메라를 통해 시선 추적을 실시한다. 그 다음 시선 추적을 통해 응시자의 시선이 화면의 밖으로 갈 때 응시자에게 경고를 보내는 방식으로 진행되고, 일정 경고 회수가 초과 될 시에 관리자에게 알림이 가는 형태로 설계되었다.

① ScreenSharing & Chatting

WebRTC를 이용하여 기본적인 화상/음성/메세지/데이터 통신을 하기 위해서는 보안상의 이유로 Extension App를 이용해야 한다.
이 Extension App에 대해서 조금 더 정확히 설명하자면, 사실 App 자체로 화면 Stream을 얻고 구현하는 것은 아니다. 단순히 Extension API를 호출하여 로컬 PC의 Screen Stream ID를 가져오고 실제 웹 페이지와 통신하여 이 ID를 전달하는 역할이 전부다. 다음은 웹 페이지와 Extension App이 통신하는 구조를 설명하고 있다.

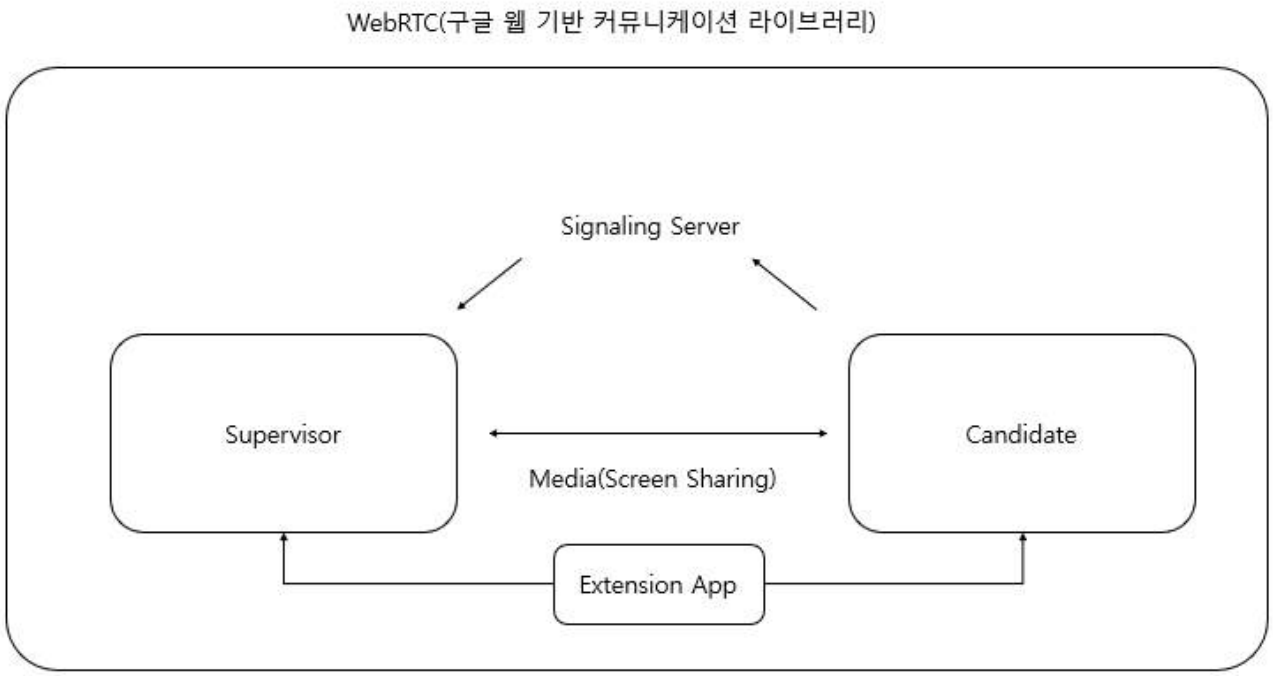

그림7. WebRTC를 이용한 ScreenSharing & Chatting

- Background Script

1) Content Script를 주입한다. 주입하는 방식은 아래 두가지가 있다.
2) manifest에 등록하여 주입
3) chrome.tabs.executeScript API를 통하여 주입
4) Content Script와 통신할 수 있도록 port를 통한 메시지 송/수신을 준비한다.
5) 메시지가 수신되면 chrome.desktopCapture.chooseDesktopMedia API를 호출한다.

- Content Script

1) Background Script와 통신할 수 있도록 port를 통한 메시지 송/수신을 준비한다.
2) 웹페이지와는 window.postMessage API를 통해 통신한다.

- 웹페이지 Script

1) 웹페이지는 Content Script에게 Screen Stream ID를 받으면 getUserMedia API를 통해 실제 Stream 객체를 가져오고, 이 stream 객체를 상대 피어에게 전달하거나, 현재 페이지를 그대로 보여줄 수 있다.

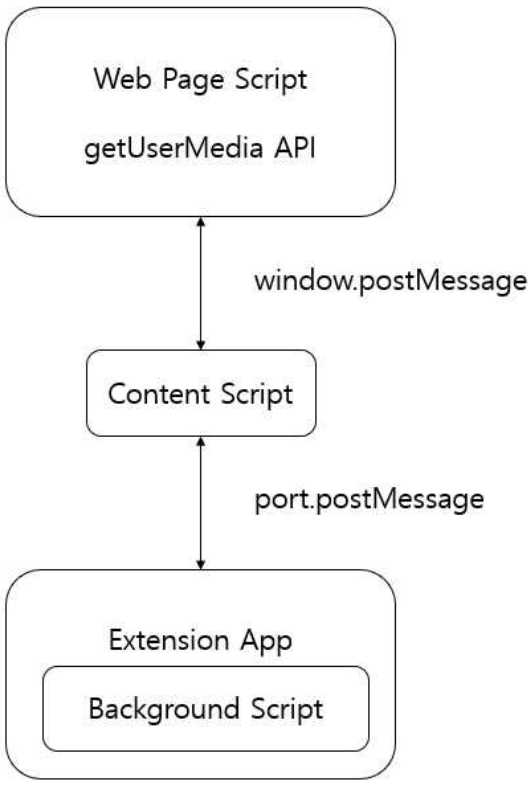

그림8. 웹페이지와 Extension App 통신 방법

② Eye-Tracking

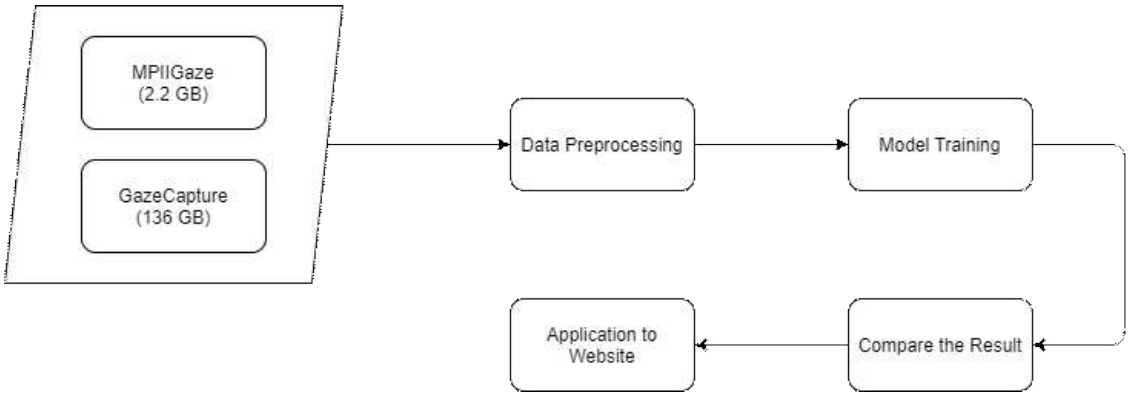

그림9. Model Design 개요

1) Data Preprocessing

현재 준비된 Gaze Estimation 관련 데이터셋은 MPIIGaze(약 2.2GB)와 GazeCapture(약 136GB)이다. 데이터셋은 실제 사람의 눈이 바라보고 있는 이미지와, 카메라 교정에 대한 정보, 그리고 사람이 실제로 바라보는 곳의 위치, 사람의 눈의 위치 등의 데이터를 담고 있다. 데이터 전처리를 통하여, 이미지 데이터들을 하나의 모양으로 통일을 시키고, 사람의 눈의 위치와 사람이 눈이 바라보고 있는 위치의 데이터를 이용하여 실제 사람이 바라 보고 있는 target의 direction을 구한다.

2) Model Training

기존에 존재하는 모델이든, 새로운 모델을 만들어 보든 모델을 만들어서 Train 시킨다.
모델 설계에 사용될 Framework는 Pytorch를 통해 구현을 할 예정이다. 시선 추적 모델에 사용할 후보들로는 논문에 있었던 FAZE, GazeNet, Fast-CRNN, Vanilla CNN 등 여러가지 모델을 구현해 볼 예정이다.
기존 논문에서는 Linux 환경에서 작동을 하였기 때문에 기존 코드를 분석하여 Windows 환경에서 작동할 수 있게 수정하는 작업이 필요할 것 같다. 또는 가상환경이 아닌 실제 데스크톱에 리눅스 환경을 설정하는 방법도 있다. (그래픽카드 사용이 불가피하기 때문에 가상환경으로는 한계가 존재한다.)
그리고 구현된 모델에 이전에 전처리한 데이터를 통해 훈련시킨다.

3) Compare the Results

사전 훈련된 모델들을 test data를 통해서 혹은 직접 real-time에 적용해보고 성능을 비교한다.

4) Application to Our Website

웹 서버와 연계하여 시선 추적을 할 수 있도록 한다.

7. 본인 인증 화면 구현

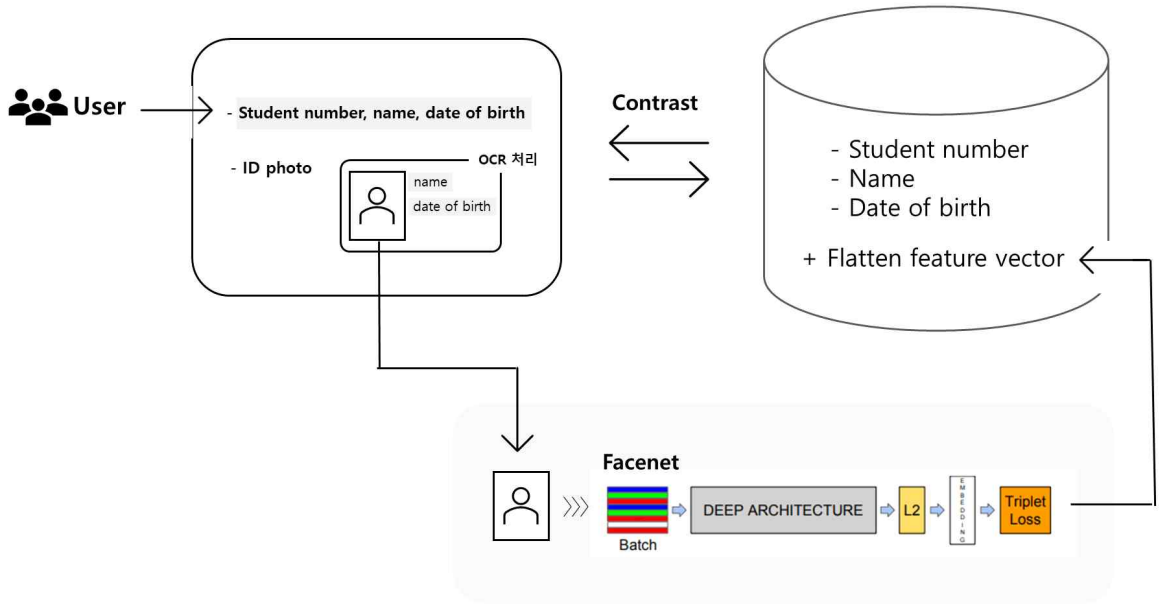

그림10. Pre-information registration

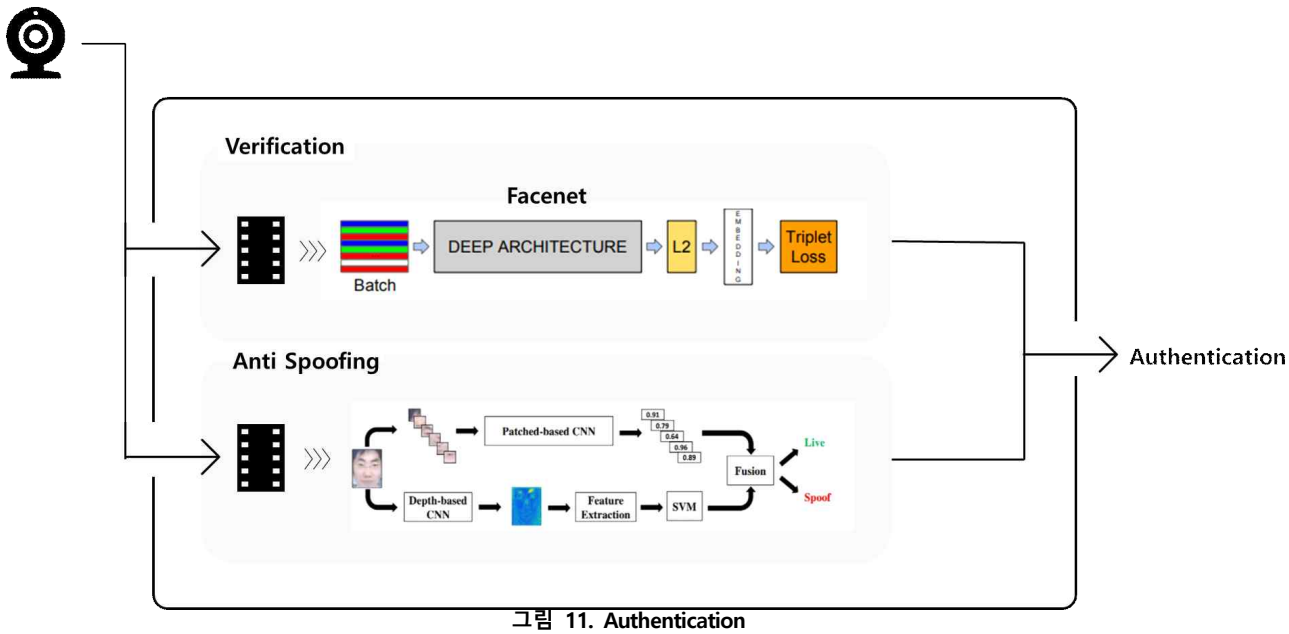

그림 11. Authentication

Fucntion List

1) Pre-information registration

본인 인증 화면에서는 응시자의 신분증을 촬영하여 그것에 기입 되어 있는 응시자의 이름과 생년월일을 사용자가 입력한 DB의 이름, 학번, 생년월일을 비교한다. 신분증은 OCR 알고리즘으로 처리하여서 문자를 추출하고 이것을 비교한 후 다르다면 사전 정보를 입력 불가능하게 구현한다.

그 후 신분증 사진을 pretrained network에 넣어서 Flatten Feature Vector를 추출한 후 DB에 저장하는 형식이다.

2) Authentication

- verification

시험 응시 전 웹캠에 비춘 얼굴의 feature vector과 DB의 feature vector를 Verification 한다.

- anti spoofing

웹캠에 비춘 얼굴을 Patched-based-CNN, Depth-based-CNN으로 이루어진 anti-spoofing network에 넣어 실제 얼굴인지 아닌지 확인

위의 두 가지 중 하나라도 불만족할 시 감독관에게 알림이 가서 직접 확인을 요청한다.

Simulation

1) Data Set 준비

기존 얼굴 데이터셋 : 서양인 위주의 데이터 셋
-> 한국 대학생의 얼굴 인식이 중심이므로 동양인 얼굴 인식 데이터 셋으로 가중치 훈련 필요 (한국인 안면 이미지 AI 데이터)

2) alignment를 위한 feature detector 성능 평가

OpenCV의 haar cascade, SSD, Dlib HoG, MTCNN Detector 중 가장 detection, alignment 성능이 좋은 Fearture Detector 선택

3) verification 알고리즘 학습 및 Threshold 값 설정

하이퍼파라미터를 바꿔가면서 Loss를 최소화하는 weights를 찾고 적정 threashold 값을 찾음

4) anti spoofing

Patch and Depth-Based CNNs, Cascade Face Detector Based on R-CNN and Retinex LBP, FeatherNets 등 anti spoofing network들 중 가벼우면서 성능이 좋은 network를 찾아 적용할 예정