실시간 시선 추적 기반의 시간 관리 플랫폼 개발

팀명: 뉴뉴뉴갈매기

201924480 박준형

201924431 김상유

201924530 이상준

지도교수: 전상률

제출일: 2024년 10월 15일

부산대학교

목차

1 서론 **1**

1.1 연구 배경 및 기존 문제점 1

1.2 연구 목표 1

2 연구 배경 **2**

2.1 EyeTracker 2

2.1.1 WebCam Eyetracker 2

2.1.2 Eyetracker 제약사항 2

2.2 GPT API 3

3 연구 내용 **4**

3.1 아이트래커 비교 4

3.2 WebGazer.js 4

3.3 Webgazer 구현 5

3.3.1 구현 이유 5

3.3.2 학습 과정 5

3.3.3 작동 원리 6

3.3.4 노이즈 필터링 7

3.4 JWT 8

3.5 스프링 시큐리티를 통한 사용자 인증, 인가 과정 8

3.5.1 사용자의 로그인 과정 9

3.5.2 스프링 시큐리티 필터 동작 원리 9

3.6 배포 자동화 9

3.7 도커 11

3.8 MS Azure를 통한 MySQL DB 서버 배포 11

3.8.1 MS Azure 채택 이유 11

3.8.2 Azure Database for MySQL 소개 12

3.8.3 DB 배포 방법 12
3.9 프로젝트 기능설명 13
3.9.1 팔로우, 팔로잉 기능 13
3.9.2 랭킹 top 10 13
3.9.3 Todo 13
3.9.4 Screen time, Category 13
3.9.5 공부 시간 관리 시스템 14

4 연구 결과 분석 및 평가 16
4.1 개발일정 16
4.2 구성원 역할 분담 17
4.3 API 명세서 17
4.4 ERD 18
4.5 Github 18

5 결론 및 향후 연구 방향 19
5.1 결론 및 기대효과 19
5.2 향후 연구 방향 19

제1장 서론

1.1 연구 배경 및 기존 문제점

최근 공부에 집중해야 하는 학생들은 유튜브, 틱톡 등의 디지털 매체로 쉽게 산만해지며 공부에 집중하기 어려워하는 문제가 두드러지고 있다. 기존의 공부 타이머는 강제성 부족으로 인해 효과가 떨어지는 문제점이 존재한다. 예시로 2018년 11월 11일 (주) 팔로에서 출시된 "열정품은타이머"는 자신의 공부시간을 측정하고 이를 서로 공유함으로써 공부하고 있는 사람들을 보고 자극을 받아 공부할 수 있지만, 집계되는 시간동안 사용자가 실제로 공부에 집중하고 있는지 여부 판단은 실행하고 있지 않기 때문에 타이머를 켜놓고 자거나 놀며 공부시간을 말도 안 되게 늘린다는 단점이 존재한다. 이에 컴퓨터 비전 기술의 시선 추적 기술(eye tracking)을 활용하여 사용자의 시선을 파악하고, 이를 통해 현재 공부에 집중하고 있는지를 판단하여 타이머를 진행함으로써 사용자의 효율적인 공부 집중을 유도하고 동기부여를 촉진할 수 있도록 돕기 위해 해당 프로젝트를 제안한다.

1.2 연구 목표

본 연구의 목표는 웹캠을 활용한 실시간 시선 추적 기술(EyeTracker)을 개발하고, 이를 통해 사용자가 화면에서 주목하는 지점을 정확히 파악하는 것이다. 먼저, 시선 추적 시스템의 성능을 개선하여 사용자 시선의 움직임을 정밀하게 분석할 수 있도록 한다. 이어서, 시선 데이터를 활용해 사용자가 화면에서 보고 있는 콘텐츠가 학습과 관련된지 여부를 자동으로 판별하는 모델을 개발한다. 마지막으로, 이러한 기능을 통합한 웹 서비스를 구축하여 사용자들이 쉽게 접근하고 사용할 수 있도록 하며, 나아가 사용자 개개인의 학습 습관과 성취도를 분석하여 맞춤형 학습 피드백을 제공하는 시스템을 구현하는 것을 최종 목표로 한다.

제2장 연구 배경

2.1 EyeTracker

2.1.1 WebCam Eyetracker

하드웨어 기반 아이트래커는 다양한 조명 조건에서도 눈의 움직임을 포착할 수 있도록 설계된 전용 카메라와 센서를 사용한다. 이 장비는 적외선 다이오드를 활용하여 각막 반사를 캡처하며, 머리 움직임과 눈 영역의 다양한 생리적 변화를 정밀하게 측정할 수 있다. 이를 통해 높은 정확도와 신뢰성을 제공하지만, 고가의 하드웨어가 필요하고 사용 환경에 제약이 있을 수 있다. 반면, 웹캠 기반 시선 추적은 일반적으로 가시광선 스펙트럼의 빛을 감지하는 표준 카메라를 이용한 비디오 기반 기술이다. 적외선 다이오드 없이 사용자의 얼굴 및 눈동자 움직임을 분석하여 시선을 추적하며, 각막 반사가 아닌 영상 처리 기법을 통해 시선 데이터를 얻는다. 본 연구에서는 접근성과 비용 효율성을 고려하여 웹캠 기반 시선 추적을 선택하였다. 웹캠은 대부분의 컴퓨터에 내장되어 있어 추가 장비가 필요하지 않으며, 하드웨어에 의존하지 않는다는 장점이 있다. 이를 통해 누구나 쉽게 사용할 수 있는 시선 추적 시스템을 구현할 수 있으며, 사용자 접근성을 크게 향상시킬 수 있다.

2.1.2 Eyetracker 제약사항

1. **웹캠 해상도에 따른 눈 영역 픽셀 수 부족 문제**
   최신 웹캠은 높은 해상도를 지원하는 경우가 대부분이지만 대기 시간과 프레임의 균형을 이루어야 한다. 해상도가 낮으면 실시간성을 보장할 수 있지만 눈 영역을 나타내는 픽셀 수가 적어지는 문제가 발생한다. 픽셀 수가 적으면 시선 추적에 있어서 필요한 정보가 부족하고 정확도와 신뢰성이 낮아진다는 문제가 발생한다.

2. **어두운 조명 조건에서의 정확도 문제**
   웹캠 기반 시선 추적은 전용 하드웨어 장치와 다르게 적외선을 사용하지 않고 가시광선 영역의 데이터만으로 시적 추적을 진행한다. 그러므로 주변 조명 조건에 따라 데이터가 변경되는 문제가 발생한다. 일반적으로 대비가 낮을 경우 얼굴 배경에 대한 눈 움직임을

감지하는 정도가 약해 정확도가 낮아질 수 있다.

3. **안경, 얼굴 각도, 움직임 등 외부 조건에 따른 정확도 문제**
   사용자의 안경 등의 착용 물품, 카메라 각도, 시선 추적 도중 머리의 움직임 또한 얼굴 영역으로부터 눈 영역을 추출하는 모델의 성능을 떨어뜨릴 수 있는 주요 원인으로 볼 수 있다. 이러한 외부 환경이 적은 이상적인 조건에서 모델의 예측이 정확하게 수행된다는 한계점이 존재한다.

2.2 GPT API

GPT API 개요

1. GPT (Generative Pre-trained Transformer) API는 OpenAI에서 제공하는 인공 지능 기반의 텍스트 생성 및 처리 서비스이다. 이 API를 통해 개발자들은 GPT 모델을 활용하여 다양한 언어 처리 작업을 수행할 수 있다. GPT 모델은 대규모의 언어 데이터를 사전 학습하여 자연스러운 언어 이해 및 생성 능력을 갖추고 있으며, 이를 API 형태로 제공함으로써 사용자가 쉽게 접근하고 활용할 수 있도록 한다.

2. 텍스트 생성: 사용자가 입력한 텍스트에 기반하여 관련성 높고 자연스러운 텍스트를 생성한다. 이를 통해 기사 작성, 스토리텔링, 자동 답변 생성 등에 활용할 수 있다.

3. 언어 이해: 질문에 대한 답변을 제공하거나 텍스트 요약, 감정 분석 등의 작업을 수행한다.

4. 번역: 다양한 언어 간의 텍스트 번역을 지원한다.

5. 코드 생성 및 수정: 프로그래밍 언어에 대한 이해를 바탕으로 코드 조각을 생성하거나 기존 코드를 수정하는 데 도움을 준다.

제3장 연구 내용

3.1 아이트래커 비교

표 3.1: 아이트래커 비교표

제품명	개발 언어	성능	가격	주요 특징
GazePointer	C#	중간(기본적 시선 추적, 경미한 지연)	무료	웹캠 기반, 데스크톱 앱
WebGazer.js	JavaScript	높음(실시간, 최소 지연)	무료(오픈소스)	웹 브라우저 호환, API 활용
OpenGaze	C++	중간(실시간, 시스템 의존적)	무료(오픈소스)	컴퓨터 비전 알고리즘, 데스크톱 앱
Cambridge Face-Tracker	C++	높음(실시간, 낮은 지연)	무료	고성능 얼굴 추적, 데스크톱 앱
iTracker	Python	가변적(시스템 의존적)	무료(오픈소스)	딥러닝 기반, 별도 웹 통합 필요

3.2 WebGazer.js

WebGazer 개요

WebGazer는 일반적인 웹캠을 사용하여 페이지에서 웹 방문자의 시선 위치를 실시간으로 유추하는 시선 추적 라이브러리이다. 여기에 포함된 시선 추적 모델은 웹 방문자가 웹 페이지와 상호작용하는 것을 보고 자체적으로 보정하고 눈의 특징과 화면의 위치 간의 매핑을 훈련한다. 이는 별도의 복잡한 교정 과정 없이도 시간이 지남에 따라 추적 정확도를 자동으로 개선 한다. WebGazer는 전적으로 JavaScript로 작성되었으며 모든 웹사이트에 통합할 수 있다. 이러한 용이한 통합성은 개발 과정을 간소화하고 시간을 절약해준다. WebGazer.js는 전적으로 클라이언트 브라우저에서 실행되므로 비디오 데이터를 서버로 전송할 필요가 없다.

3.3 Webgazer 구현

3.3.1 구현 이유

본 프로젝트는 WebGazer.js의 접근 방식에서 영감을 받아 자체적인 웹 기반 시선 추적 시스템을 구현했다. WebGazer의 핵심 개념인 사용자 상호작용 기반 자체 교정 기능과 순수 JavaScript 구현 방식을 참고하되, 우리 프로젝트의 특수한 요구사항에 맞춰 최적화된 솔루션을 개발본 프로젝트는 WebGazer.js의 접근 방식에서 영감을 받아 자체적인 웹 기반 시선 추적 시스템을 구현했다.

구체적으로, 다음과 같은 핵심 요소들을 직접 구현한 것이다.

- 사용자의 클릭 및 마우스 움직임 데이터를 수집하고 분석하는 모듈
- 수집된 데이터를 바탕으로 시선 위치를 예측하는 알고리즘
- 실시간으로 정확도를 개선하는 자체 교정 시스템

이러한 직접 구현 방식을 통해 WebGazer의 장점을 살리면서도, 프로젝트의 특정 요구사항에 더욱 부합하는 맞춤형 솔루션을 제공할 수 있게 되었다. 이는 단순히 기존 라이브러리를 사용하는 것보다 훨씬 더 유연하고 확장 가능한 시스템을 구축할 수 있게 해준 것이며, 향후 지속적인 개선과 최적화가 가능한 기반을 마련했다.

3.3.2 학습 과정

동공 위치와 눈 특징을 화면 좌표에 맞추기 위해서는 2차원 벡터(동공 위치)와 120차원 (눈 특징)를 각각 화면상의 시건 좌표에 매핑해야 한다. 카메라와 화면에 대한 머리의 3D 위치와 회전에 따라 달라지기 때문에, 이럴 정확하게 처리하려면 세심한 보정과 많은 연산이 필요하다. 대신, WebGazer에서는 일반적으로 발생하는 사용자 상호작용을 통해 지속적인 자가 보정에 의존한다. Gaze and cursor alignment in web search[1]에 따르면 사용자가 클릭하는 순간, 시선과 커서 간의 거리가 평균적으로 74픽셀이라는 것을 보여주었다. 이 거리는 작업에 따라 달라질 수 있지만, 시선과 커서가 클릭 시 완벽하게 정렬된다고 단순화하여 분석한다. 이 가정은 일반적으로 충분히 포괄적이어서 다양한 환경과 작업에서 사용할 수 있도록 한다. 우리는 클릭과 커서 움직임에 초점을 맞추지만 다른 유형의 상호작용으로 확장할 수 있다.

3.3.3 작동 원리

얼굴 랜드마크 좌표 획득(tfjs의 Face-Landmark 모델 사용)

첫 번째 단계는 TensorFlow.js의 face-landmark 모델을 사용하여 사용자의 얼굴에서 주요 랜드마크의 좌표를 추출하는 것이다. Face-Landmark 모델은 딥러닝 기반의 얼굴 인식 모델로, 웹캠으로 들어오는 이미지 프레임에서 얼굴의 특징을 실시간을 감지한다. 이 모델은 눈 위치를 추출하는 데 필수적인 정보를 제공한다.

JavaScript 환경에서 활용성이 높은 TFJS[1] 모델로 선정했다.

눈 이미지 추출

랜드마크 좌표가 확보되면, 해당 좌표 중 눈 부분에 해당하는 영역을 잘라내어 이미지를 가져온다. 눈 영역의 이미지는 시선 추적을 위한 입력으로 사용될 것입니다. 눈을 구성하는 랜드마크 좌표를 이용하여 눈 영역의 바운딩 박스를 얻고 이미지를 추출한다.

눈 이미지 전처리

추출된 눈 이미지는 선형회귀를 입력으로 사용하기 위해 10x6 크기로 리사이징한다. 리사이징된 이미지는 그레이스케일과 히스토그램 평활화를 거쳐 노이즈를 제거하고 환경의 영향을 적게 받아 강건성을 뛰게 한다. 이 후 이미지르 1차원 벡터로 변환하여 선형회귀의 입력으로 사용한다.

선형회귀 학습

웹캠을 통해 수집된 눈 이미지를 X 변수로 사용하고, 사용자의 클릭 위치를 Y변수로 사용한다. 이때 선형회귀 알고리즘을 사용하여 X와 Y의 관계를 학습한다.

선형회귀는 X와 Y 사이의 선형적 관계를 찾는 알고리즘으로, 눈의 이미지 벡터로부터 사용자가 실제로 모니터에서 바라보는 위치를 예측한다. 이 과정에서 사용자의 시선 데이터를 기반으로 회귀 계수가 학습되며, 이는 나중에 실시간 시선 추적에 사용된다.

딥러닝이 아닌 선형회귀를 사용한 이유

1. **실시간 데이터 반영**

   선형회귀는 적은 연산량으로 인해 실시간 데이터를 즉시 반영할 수 있다. 사용자 상호작용을 통해 지속적인 자가 보정에 의존하는 시스템 특성 상 새로운 데이터가 추가될

[1] https://github.com/tensorflow/tfjs-models/tree/master/face-landmarks-detection#keypoints

때마다 빠르게 모델을 업데이트할 수 있어야 한다. 딥러닝의 경우 연산량이 많아 실시간성이 저하될 수 있고 사용자에게 부정적인 경험을 줄 수 있다.

2. **적은 데이터**
선형회귀는 단순한 수학적 모델을 사용하여 눈 이미지 벡터와 시선 좌표 간의 관계를 단순히 추정하는데 적합하며, 복잡한 데이터 전처리나 대규모 학습이 필요하지 않다. 따라서 비교적 적은 양의 데이터로도 만족스러운 성능을 보일 수 있다.

3.3.4 노이즈 필터링

윈도우 슬라이싱

본 연구에서는 시선 추적 데이터의 예측 정확성을 더욱 향상시키기 위해 불확실성을 고려한 윈도우 슬라이싱(Window Slicing) 기법을 적용하였다. 이 기법은 최근의 예측 결과를 기반으로 최종 예측 값을 도출하여, 일관된 시선 추적 경로를 제공하는 데 중점을 두었다. 윈도우 슬라이싱 기법은 특정 시점에서의 가장 최근 예측값 4개를 선택하여 이들의 평균치를 계산하는 방식으로 작동한다. 이를 통해 개별 예측값의 변동성을 줄이고, 더욱 안정적인 시선 좌표를 생성할 수 있었다. 이러한 접근 방식은 다음과 같은 이점을 가진다.

- **변동성 완화**: 최근 예측값의 평균을 사용함으로써, 일시적인 노이즈나 외부 요인으로 인한 급격한 변동이 최종 예측에 미치는 영향을 최소화할 수 있다. 이는 시선 추적 결과의 신뢰성을 높이는 데 기여한다.
- **시간적 연속성 유지**: 윈도우 슬라이싱 기법을 통해 최근 예측값의 집합을 사용하므로, 시선 이동의 시간적 연속성을 자연스럽게 반영할 수 있다. 이는 사용자가 화면을 주시하는 흐름을 더 원활하게 표현할 수 있도록 도와준다.

칼만 필터

본 연구에서는 웹캠 기반 시선 추적 시스템의 정확성을 향상시키기 위해 칼만 필터(Kalman Filter)를 적용하였다. 시선 추적 기술은 사용자가 화면에서 어디를 주시하고 있는지를 실시간으로 분석하는데, 이 과정에서 발생하는 작은 노이즈나 순간적인 변동이 시각적으로 불안정한 추적 경로를 만들 수 있다. 이를 해결하기 위해 칼만 필터를 적용하여 시선 좌표를 부드럽고 안정적으로 연결되도록 하였다.

칼만 필터는 현재 시점에서의 시선 좌표를 예측하고, 실제 측정된 시선 데이터와 비교하여 이를 업데이트하는 방식으로 동작한다. 이 과정에서 발생할 수 있는 노이즈나 오류를 최소화하면서, 시선 이동 경로를 자연스럽게 추적할 수 있도록 보정한다.

필터는 두 가지 주요 단계로 구성된다.

- **예측 단계**
  이전 시점의 시선 좌표를 바탕으로, 사용자가 화면에서 다음에 주시할 위치를 예측한다. 이 단계는 시선의 연속성을 유지하고 급격한 변화가 발생하지 않도록 도와준다.

- **업데이트 단계**
  실제 측정된 시선 좌표와 예측값을 비교한 후, 두 값의 차이를 기반으로 현재 시선 좌표를 조정한다. 이때 노이즈를 고려하여 추적 경로의 정확도를 높인다.

이를 통해, 시선 추적 경로는 화면 상에서 부드럽고 자연스럽게 표현되며, 실시간 데이터의 불규칙성을 효과적으로 완화할 수 있다. 특히, 사용자가 빠르게 시선을 이동하거나 조명 조건이 변하는 등의 상황에서도 시선 추적 결과가 안정적으로 화면에 표시되도록 구현할 수 있었다. 칼만 필터는 이러한 시선 추적 기술의 안정화에 중요한 역할을 하였으며, 결과적으로 시각적으로 일관성 있고 신뢰성 있는 추적 데이터를 제공할 수 있었다.

3.4 JWT

SeagullsRoom은 단순히 학습 시간을 측정하는 도구를 넘어, GPT API를 활용해 사용자의 학습 습관을 전면적으로 개선하고 최적화하는 종합적인 학습 관리 플랫폼으로 자리매김할 것으로 기대된다. 향후 사용자 피드백을 바탕으로 지속적인 기능 개선과 확장을 통해, 더욱 효과적이고 개인화된 학습 경험을 제공할 수 있을 것이다.

채택 이유

- **무상태성(stateless)**: 서버 측에서 별도의 세션 저장소를 유지할 필요가 없어 확장성이 우수하다.
- **보안성**: 암호화된 서명을 통해 데이터 무결성을 보장하며, 토큰 탈취 시에도 민감한 정보 노출을 최소화할 수 있다.
- **효율성**: 인증 정보가 토큰에 포함되어 있어 데이터베이스 조회 횟수를 줄일 수 있다.
- **유연성**: 다양한 환경(웹, 모바일 등)에서 일관된 인증 체계를 유지할 수 있다.
- **크로스 도메인 인증**: 서로 다른 도메인 간에도 토큰 기반의 인증이 가능하다.

3.5 스프링 시큐리티를 통한 사용자 인증, 인가 과정

본 프로젝트에서는 사용자 인증 및 인가 과정을 안전하고 효율적으로 관리하기 위해 스프링 시큐리티 프레임워크를 채택하였다. 스프링 시큐리티는 스프링 기반 애플리케이션의

보안을 담당하는 강력한 인증과 접근 제어 프레임워크이다.

3.5.1 사용자의 로그인 과정

1. 사용자의 요청으로 Login 요청을 받는다.
2. UsernamePasswordAuthenticationFilter가 ID, Password를 꺼내서 Authentication Manager에게 넘겨준다.
3. Authentication Manager은 DB로 회원 정보를 가져와서 검증을 한다.
4. 검증이 확인되면 succussfulAuth가 JWTUtil을 통해 JWT를 생성해 사용자에게 넘겨준다.

3.5.2 스프링 시큐리티 필터 동작 원리

스프링 시큐리티는 클라이언트의 요청이 여러 개의 필터를 거쳐 DispatcherServlet(Controller) 으로 향하는 중간 필터에서 요청을 가로챈 후 검증(인증/인가)을 진행한다.

사용자의 요청은 다음과 같은 과정으로 진행된다.

클라이언트 요청 → 서블릿 컨테이너 → 서블릿 필터 → 서블릿(컨트롤러)

3.6 배포 자동화

본 프로젝트에서는 효율적이고 안정적인 배포 프로세스를 구축하기 위해 Git Action과 Docker를 활용한 배포 자동화 파이프라인을 구현하였다. 이를 통해 개발부터 운영 환경까지의 일관된 배포 과정을 확립하고, 수작업으로 인한 오류(human error)를 최소화하였다.

배포 자동화 개요

배포 자동화는 소프트웨어의 빌드, 테스트, 배포 과정을 자동화하여 개발 생산성을 향상시키고 배포의 안정성을 높이는 기술이다. 본 프로젝트에서는 Git Action을 CI/CD(지속적 통합/지속적 배포) 도구로, Docker를 컨테이너화 플랫폼으로 선택하여 자동화 파이프라인을 구축하였다.

Git Action 활용

Git Action은 GitHub에서 제공하는 워크플로우 자동화 도구로, 다음과 같은 작업을 수행한다.

- 코드 변경 감지: main 브랜치에 push 또는 pull request 이벤트 발생 시 자동으로 워크플로우 실행
- 자동 빌드 및 테스트: 프로젝트 빌드 및 단위 테스트 실행
- Docker 이미지 생성: 애플리케이션의 Docker 이미지 자동 생성
- 이미지 푸시: 생성된 이미지를 Docker Hub 또는 프라이빗 레지스트리에 푸시

Docker 컨테이너화

Docker를 사용하여 애플리케이션을 컨테이너화함으로써 다음과 같은 이점을 얻었다.

- 환경 일관성: 개발, 테스트, 운영 환경 간의 일관성 유지
- 배포 용이성: 컨테이너 이미지를 통한 간편한 배포 및 롤백
- 자원 효율성: 가볍고 빠른 컨테이너 실행으로 자원 활용 최적화
- 확장성: 필요에 따라 컨테이너 인스턴스 쉽게 확장 가능

자동화 파이프라인 흐름

1. 코드 푸시: 개발자가 GitHub 레포지토리의 main 브랜치에 코드 푸시
2. 워크플로우 트리거: Git Action 워크플로우 자동 실행
3. 빌드 및 테스트: 프로젝트 빌드 및 자동화된 테스트 수행
4. Docker 이미지 생성: 성공적인 빌드 후 Docker 이미지 생성
5. 이미지 푸시: 생성된 이미지를 레지스트리에 푸시
6. 배포: 대상 서버에 새 이미지를 사용하여 애플리케이션 자동 업데이트

이러한 배포 자동화 파이프라인의 구축으로 본 프로젝트는 더욱 효율적이고 안정적인 개발 및 운영 프로세스를 확립하였다. 이는 서비스의 품질 향상과 향후 프로젝트의 확장성과 유지보수성을 크게 개선하였다.

1. Pull Request가 main branch에 merge가되면 Github Actions 스크립트가 수행된다.
2. Github Actions이 Docker 기반으로 Docker Image를 만든다
3. 만들어진 Docker Image를 docker hub에 로그인후 push한다.

4. Github Actions 스크립트로 EC2에접근하고, push한 Docker Image를 pull 하여실행한다.

5. 배포 완료

3.7 도커

본 프로젝트에서는 배포 및 운영 환경의 효율성, 일관성, 유연성을 극대화하기 위해 도커(Docker)를 채택하였다. 도커는 애플리케이션을 컨테이너화하여 개발, 배포, 실행하는 오픈소스 플랫폼으로, 현대 소프트웨어 개발 및 운영에 있어 핵심적인 기술이다.

도커의 도입은 본 프로젝트의 개발 및 운영 프로세스를 현대화하고, 애플리케이션의 안정성과 확장성을 크게 향상시켰다. 이는 결과적으로 서비스 품질 향상과 유지보수 비용 절감으로 이어졌다.

```
FROM openjdk:17-jdk
ARG JAR_FILE=SeagullsRoom/build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=docker", "-jar", "app.jar"]
```

그림 3.1: 작성한 Dockerfile

3.8 MS Azure를 통한 MySQL DB 서버 배포

이번 프로젝트에서는 효율적인 데이터베이스 관리와 클라우드 환경에서의 원활한 개발 진행을 위해 MS Azure를 활용하여 MySQL DB 서버를 배포하였다. 클라우드 기술은 현대 소프트웨어 개발의 핵심적인 인프라로 자리잡고 있으며, 이를 통해 서버 관리 부담을 줄이고, 확장성과 가용성을 확보하는 것은 프로젝트의 성공을 위한 중요한 요소 중 하나이다. 또한, 실제 클라우드 환경에서의 배포 경험을 통해 실무 역량을 강화하였다.

3.8.1 MS Azure 채택 이유

1. **클라우드 환경 경험**: 클라우드 환경에서 배포를 진행함으로써 실무적인 경험을 쌓기 위함이다.

2. **학생 지원 기능**: MS Azure 학생 구독을 통해 $100.00의 크레딧을 지원받을 수 있었다.

3. **편리한 관리 기능**: Azure Database for MySQL은 관리형 서비스로 제공되기 때문에 사용자가 직접 서버를 설치하고 관리할 필요가 없다는 장점이 있다.

4. **높은 확장성 및 가용성**: MS Azure는 글로벌 클라우드 인프라를 통해 자동 확장 기능을 제공하며, 데이터베이스 트래픽에 맞춰 성능을 유연하게 조정할 수 있어 높은 확장성과 가용성을 보장한다.

5. **보안 및 데이터 보호**: Azure는 강력한 보안 기능을 제공하며, 데이터 암호화, 접근 제어, 네트워크 방화벽 설정 등을 통해 중요한 데이터를 안전하게 보호할 수 있다.

6. **지역화된 데이터 센터 지원**: Azure는 글로벌 서비스이지만, 지역별 데이터 센터를 통해 가까운 위치에서 서비스를 제공함으로써 네트워크 지연을 최소화하고 더 빠른 응답 속도를 제공한다. 이번 과제에서는 Korea Central 서버를 사용하여 배포하였다.

3.8.2 Azure Database for MySQL 소개

Azure Database for MySQL은 MS Azure에서 제공하는 완전 관리형 MySQL 데이터베이스 서비스이다. 사용자는 MySQL 서버를 직접 관리할 필요 없이, 클라우드에서 안정적이고 확장 가능한 MySQL 환경을 사용할 수 있다. 이 서비스는 자동 백업, 고가용성, 자동 스케일링 등의 기능을 통해 성능과 안정성을 보장한다.

3.8.3 DB 배포 방법

1. **Azure Portal 접속**: Microsoft Azure 계정으로 로그인 후, Azure 포털에 접속한다.

2. **리소스 생성**: '리소스 만들기'를 선택하고, 'Azure Database for MySQL'을 검색하여 데이터베이스 서버를 생성한다.

3. **설정**: 데이터베이스 서버의 지역, 가격 책정 계층, 버전 등의 설정을 진행한 후, 관리자 사용자 이름 및 암호를 설정한다.

4. **네트워크 설정**: 데이터베이스 접근을 위한 방화벽 규칙을 설정하여 IP 주소를 허용한다.

5. **배포 완료 및 연결**: 서버가 배포된 후, 제공된 연결 정보를 이용해 MySQL 클라이언트를 통해 데이터베이스에 연결할 수 있다.

6. **데이터베이스 생성 및 관리**: MySQL Workbench 등을 이용해 데이터베이스와 테이블을 생성하고, 데이터를 관리할 수 있다.

3.9 프로젝트 기능설명

3.9.1 팔로우, 팔로잉 기능

사용자가 팔로우/팔로잉 기능을 사용할 수 있도록 구현했다. Friend 테이블을 통해 사용자와 친구 간의 관계를 관리하며, 사용자(User)와 친구(Friend) 간의 1:N 관계를 기반으로 친구 추가 기능을 구현했다. 이 기능을 통해 사용자는 다른 사용자를 친구로 추가할 수 있으며, 중복 추가를 방지하고 자기 자신을 친구로 추가하는 것을 방지하는 유효성 검사가 포함되어 있다.

3.9.2 랭킹 top 10

당일의 전체 이용자의 공부 시간을 기준으로 상위 10명의 랭킹을 보여주는 기능이다.

이를 구현하기 위해, 현재 날짜의 공부 데이터를 조회한 후, 사용자 ID별로 그룹화하여 여러 공부 기록을 합산한다. 그런 다음, 총 공부 시간을 기준으로 내림차순 정렬하고, Pageable을 사용하여 상위 10명의 데이터를 조회한다.

3.9.3 Todo

Todo 기능은 사용자가 개인적인 할 일 목록을 관리할 수 있게 해주는 시스템이다. 사용자는 제목과 설명을 포함한 새로운 Todo 항목을 생성할 수 있으며, 이는 데이터베이스에 저장된다. 각 Todo 항목은 완료 여부를 표시할 수 있어, 사용자가 자신의 진행 상황을 추적할 수 있다.

사용자는 자신의 Todo 목록을 조회, 수정, 삭제할 수 있다. 조회 기능은 사용자의 모든 Todo 항목을 나열하며, 수정 기능을 통해 제목, 설명, 완료 상태를 변경할 수 있다. 삭제 기능은 더 이상 필요하지 않은 Todo 항목을 제거한다.

이 시스템은 사용자 인증을 통해 개인정보를 보호하며, 각 사용자는 자신의 Todo 항목만을 관리할 수 있다. 또한, 입력 데이터의 유효성 검사를 통해 부적절한 데이터의 입력을 방지하고 있다. 이를 통해 사용자는 자신의 학습 계획을 체계적으로 관리하고 추적할 수 있다.

3.9.4 Screen time, Category

Screen time & Category 기능은 사용자의 학습 활동을 더욱 세밀하게 분석하고 관리할 수 있게 해주는 시스템이다. 이 기능은 사용자가 정의한 카테고리를 기반으로 화면 활동을 추적하고, 그 결과를 시각적으로 제공한다.

사용자는 웹 인터페이스를 통해 자신만의 고유한 카테고리를 정의하고 추가할 수 있다. 이렇게 추가된 카테고리 정보는 브라우저의 localStorage에 저장되어 클라이언트 측에서 지속적으로 관리된다. 공부 세션이 시작될 때, 이 저장된 카테고리 정보는 자동으로 ChatGPTRequestDto에 포함된다.

구체적으로, 화면 캡처가 이루어질 때마다 localStorage에서 사용자 정의 카테고리 목록을 가져와 서버로 전송되는 요청에 포함시킨다. 이 과정에서 기본 카테고리(예: "GAME", "SNS", "OTHER")와 사용자 정의 카테고리가 함께 ChatGPT에 전달된다.

ChatGPT는 이렇게 확장된 카테고리 목록을 바탕으로 화면 활동을 분석한다. 사용자 정의 카테고리가 포함됨으로써, AI는 각 사용자의 고유한 학습 환경과 활동 패턴을 더욱 정확하게 이해하고 분류할 수 있게 된다. 예를 들어, 한 사용자가 "온라인 강의"라는 카테고리를 추가했다면, ChatGPT는 이를 인식하고 관련 화면 활동을 해당 카테고리로 분류할 수 있다.

이러한 맞춤형 접근 방식을 통해, 시스템은 각 사용자의 독특한 학습 습관과 환경을 고려한 더욱 정확하고 개인화된 공부 집중 확인 서비스를 제공할 수 있다. 사용자는 자신의 학습 활동이 더 정확하게 분류되고 추적되는 것을 경험하게 되며, 이는 결과적으로 더 효과적인 학습 관리와 개선으로 이어질 수 있다.

화면 활동은 설정된 시간마다 캡처되어 ChatGPT에 의해 분석되며, 그 결과는 카테고리별로 집계된다. 이 데이터는 서버에 저장되어 누적되며, 사용자는 웹 인터페이스를 통해 자신의 스크린타임 데이터를 시각적으로 확인할 수 있다.

프론트엔드에서는 도넛 차트를 활용하여 각 카테고리별 시간 사용 비율을 직관적으로 표시한다. 이를 통해 사용자는 자신의 학습 패턴과 시간 사용 현황을 한눈에 파악할 수 있다.

또한, 이 기능은 사용자의 학습 행동을 모니터링하고 피드백을 제공하는 데 활용될 수 있다. 예를 들어, 특정 카테고리에 과도한 시간을 사용하고 있다면 경고를 줄 수 있고, 학습에 집중하지 못하는 패턴이 발견되면 조언을 제공할 수 있다.

이러한 상세한 데이터 추적과 분석은 사용자가 자신의 학습 습관을 개선하고 더 효율적인 시간 관리를 할 수 있도록 돕는다. 결과적으로 이 기능은 사용자의 학습 효율성을 높이고, 더 나은 학습 결과를 얻는 데 기여할 수 있다.

3.9.5 공부 시간 관리 시스템

1. **공부 시작 기능**

   공부 시작 기능은 사용자가 학습 세션을 시작할 때 활성화된다. 이 기능은 다음과 같은 프로세스로 구현되어 있다.

   (a) 사용자가 웹 인터페이스의 '공부시작' 버튼을 클릭한다.

(b) 시스템은 사용자가 정의한 카테고리가 최소 3개 이상인지 확인한다.

(c) 화면 공유 권한을 요청하고 획득한다.

(d) 눈 추적(eye tracking) 시스템을 초기화하고 시작한다.

(e) 타이머를 시작하여 공부 시간 측정을 시작한다.

(f) 주기적으로(5초마다) 화면을 캡처하고 AI 분석을 위해 서버로 전송한다.

2. **공부 정지(일시 정지) 기능**

공부 정지 기능은 사용자가 잠시 학습을 중단할 때 사용된다.

(a) 사용자가 '일시정지' 버튼을 클릭하거나, AI가 학습 외 활동을 감지했을 때 자동으로 활성화된다. 특히, 학습 외 활동 감지 시 Web API를 활용해 알림 이벤트를 전송한다.

(b) 타이머를 정지하고 현재까지의 공부 시간을 저장한다.

(c) 화면 캡처 및 분석 프로세스를 일시 중지한다.

(d) 사용자가 '재시작' 버튼을 클릭하면 중단된 지점부터 다시 시작한다.

3. **공부 저장(종료) 기능**

공부 저장 기능은 학습 세션을 완전히 종료하고 데이터를 저장할 때 사용된다.

(a) 사용자가 '종료' 버튼을 클릭한다.

(b) 타이머를 정지하고 최종 공부 시간을 계산한다.

(c) 눈 추적 시스템과 화면 공유를 종료한다.

(d) 학습 시간 데이터를 서버로 전송하여 저장한다

(e) 카테고리별 스크린타임 데이터를 서버로 전송하여 저장한다.

(f) localStorage에 저장된 임시 데이터를 초기화한다.

(g) 사용자 인터페이스를 초기 상태로 리셋한다.

이러한 기능들은 'StudyService'와 'StudyController'를 통해 서버 측에서 처리되며, 클라이언트 측에서는 JavaScript를 통해 구현되어 있다. 이 시스템은 사용자의 학습 패턴을 정확히 추적하고, 효과적인 시간 관리를 지원하며, ChatGPT API 기반 분석을 통해 학습 효율성을 향상시키는 데 중점을 두고 있다.

제4장 연구 결과 분석 및 평가

4.1 개발일정

표 4.1에 나타낸 개발 일정표는 프로젝트의 전체적인 진행 과정을 월별로 나타낸 것이다. 기획 단계부터 최종 배포 및 수정에 이르기까지의 모든 주요 개발 단계가 시각적으로 표현되어 있다. 이를 통해 각 개발 구분별 작업의 시작과 종료 시점, 그리고 여러 작업이 동시에 진행되는 구간을 명확히 파악할 수 있다.

특히 이 표는 인공지능 모델 개발, DB 설계, REST API 설계, 애플리케이션 개발 등 핵심적인 개발 단계들이 어떻게 병렬적으로 진행되는지를 효과적으로 보여주고 있다. 이는 프로젝트 관리 측면에서 매우 중요한 정보로, 각 팀원들이 자신의 역할과 책임, 그리고 전체 프로젝트 내에서의 위치를 정확히 이해하는 데 도움을 준다.

표 4.1: 개발 일정표

개발구분	세부항목	5	6	7	8	9	10
기획	주제 선정 및 고도화	○					
	사전 조사	○					
인공지능 모델 개발	Eyetracking 구현	○	○				
	GPT API 연동		○	○			
	GPT 파인튜닝		○	○			
DB 설계	DB 설계 및 구축		○	○			
REST API 설계	DB와 모델 연동			○			
	기능 개발 및 배포			○	○		
애플리케이션 개발	REST API 연동		○	○	○	○	
	기능 개발 및 배포		○	○	○	○	
배포 및 수정	보완사항 수정 및 배포					○	○

4.2 구성원 역할 분담

구성원별 역할 분담은 표 4.2와 같다.

표 4.2: 역할 분담

학번	성명	역할
201924480	박준형	JWT 활용한 회원가입, 로그인 기능 GPT 4o API 로직 구현 공부시간 top 10조회 기능 팔로잉 기능 챗봇 프론트 UI 도커, AWS를 활용한 배포 자동화
201924431	김상유	로그인 페이지 공부 타이머 GPT Function Call 아이트래커 구현 마이페이지 구현 웹페이지 디자인
201924530	이상준	study 도메인 friend 도메인 todo 도메인 알림 기능 API Azure Database for MySQL 서버 배포 및 연동 스크린타임 API 명세서 작성

4.3 API 명세서

API 명세서는 웹 서비스의 API 구조와 사용법을 상세히 기술한 문서이다. 이는 개발자들이 API를 이해하고 올바르게 사용할 수 있도록 돕는 중요한 도구이다. Swagger는 API 설계, 문서화, 테스트를 위한 오픈소스 도구 세트로, RESTful API를 시각적으로 표현하고 관리할 수 있게 해준다. 본 프로젝트에서 Swagger를 채택한 이유는 자동화된 문서 생성, 직관적인 UI를 통한 API 테스트 기능, 그리고 개발 과정에서의 효율성 향상 때문이다. Swagger를 사용함으로써 API 문서를 항상 최신 상태로 유지하고, 프론트엔드와 백엔드 개발자 간의 원활한 협업을 도모할 수 있다.

- 모든 엔드포인트는 인증된 사용자만 접근 가능한 것으로 가정함(Principal 파라미터가 있는 경우).
- 요청 및 응답 본문의 상세 구조는 각 DTO 클래스에 정의되어 있음.

- 오류 응답은 별도로 명시되어 있지 않지만, 적절한 HTTP 상태 코드와 함께 반환될 것임.
- `/api/v1/users` 엔드포인트는 새 사용자 등록 후 로그인 페이지로 리다이렉트함.

4.4 ERD

그림 4.1에 나타낸 ERD는 우리 시스템의 데이터베이스 구조를 시각적으로 표현한 것이다. 주요 엔티티와 그들 간의 관계를 보여주며, 시스템의 데이터 모델을 명확하게 정의하고 있다.

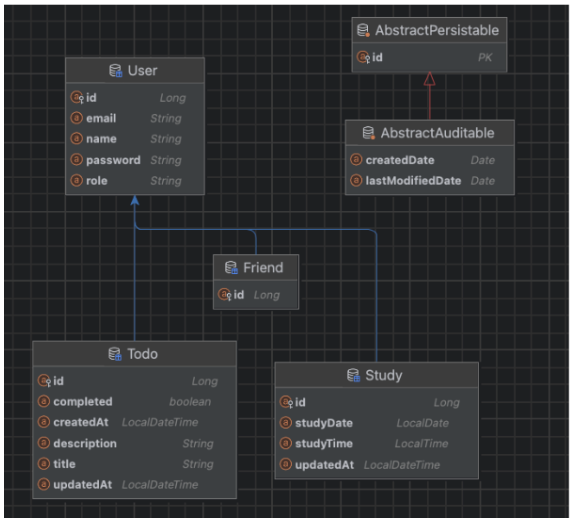

그림 4.1: 데이터베이스 ERD

4.5 Github

본 프로젝트의 전체 소스 코드와 관련 문서는 GitHub 레포지토리에서 확인할 수 있다. 해당 레포지토리는 프로젝트의 모든 구성 요소를 포함하고 있으며, 적극적인 PR 및 Issue를 활용해 지속적인 개발과 협업을 위한 플랫폼으로 활용되고 있다.

레포지토리 URL: `https://github.com/new3seagull/SeagullsRoom`

이 GitHub 레포지토리를 통해 우리 프로젝트의 기술적 구현 세부 사항을 공유하고, 커뮤니티와의 협업을 통해 지속적인 개선과 발전을 도모하고자 한다.

제5장 결론 및 향후 연구 방향

5.1 결론 및 기대효과

본 프로젝트를 통해 실시간 시선 추적 기반의 시간 관리 플랫폼 SeagullsRoom을 성공적으로 개발하였다. Eyetracker 와 GPT API 를 활용한 이 플랫폼은 사용자들이 공부 외에도 다양한 활동을 진행하며 시간을 효율적으로 관리하는 경험을 제공할 것으로 기대된다. 주요 기대 효과는 다음과 같다.

SeagullsRoom은 단순히 학습 시간을 측정하는 도구를 넘어, GPT API를 활용해 사용자의 학습 습관을 전면적으로 개선하고 최적화하는 종합적인 학습 관리 플랫폼으로 자리매김할 것으로 기대된다. 향후 사용자 피드백을 바탕으로 지속적인 기능 개선과 확장을 통해, 더욱 효과적이고 개인화된 학습 경험을 제공할 수 있을 것이다.

5.2 향후 연구 방향

SeagullsRoom 프로젝트의 구현을 바탕으로, 다음과 같은 향후 연구 방향을 제안한다. 이러한 연구 방향은 플랫폼의 기능을 확장하고 사용자 경험을 더욱 개선하는 것을 목표로 한다.

이러한 발전을 통해 SeagullsRoom은 사용자에게 더욱 풍부하고 효과적인 학습 경험을 제공할 것이다. 단순히 공부 시간을 관리하는 도구를 넘어, 사용자의 전반적인 학습 여정을 지원하고 최적화하는 종합적인 교육 플랫폼으로 진화할 것이다. 이는 궁극적으로 개인의 학습 성과 향상뿐만 아니라, 교육 시스템 전반에 긍정적인 변화를 가져올 수 있는 잠재력을 가지고 있다.

향후 이러한 연구 방향을 실현해 나가면서, 사용자 피드백과 교육 현장의 요구사항을 지속적으로 반영하여 플랫폼을 개선해 나갈 것이다. 이를 통해 SeagullsRoom은 끊임없이 진화하는 학습 환경에서 사용자들에게 가장 효과적이고 혁신적인 학습 도구가 될 것으로 기대한다.

참고문헌

[1] Jeff Huang, Ryen White, and Georg Buscher. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the sigCHI conference on human factors in computing systems*, pages 1341–1350, 2012.

[2] Broadcom Inc. Spring security 6.5.0. Online Document. Checked on 2024-10-08.

[3] Docker Inc. Docker documentation. Online Document. Checked on 2024-10-08.

[4] Ole Baunbæk Jensen. Webcam eye tracking vs. an eye tracker [pros & cons]. Online Document, 2022. Checked on 2024-10-08.

[5] Okta. Introduction to JSON web tokens. Online Document. Checked on 2024-10-08.

[6] Alexandra Papoutsaki, James Laskey, and Jeff Huang. Searchgazer: Webcam eye tracking for remote studies of web search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*, pages 17–26, 2017.

[7] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. WebGazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3839—-3845, 2016.

[8] Brown WebGazer Team. WebGazer GitHub repository. Online Document, 2016. Checked on 2024-10-08.