TEAM PROJECT

AI 융합 리액트 기반

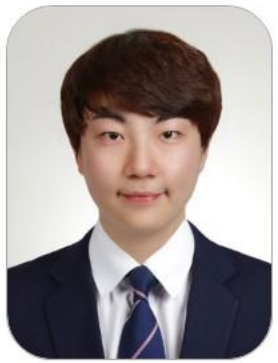
대화형 면접 평가 서비스

목차

01

팀원 소개

조흥재 팀장



- 데이터베이스 전체 설계 및 통합 관리
- Git 리포지토리 관리 및 협업 환경 설정
- Spring Boot 기반 프로젝트 설정 및 서버 운영 관리
- React와 Next.js 기반 프로젝트 설정 및 서버 운영 관리
- Toss Payments API를 활용한 결제 및 환불 시스템 구현
- 이력서를 기반으로 면접 질문을 생성하는 백엔드 기능 구현
- 면접 결과 조회를 위한 백엔드 기능 처리
- Google SMTP와 CoolSMS를 통한 이메일 및 휴대폰 인증 구현
- 마이페이지 정보 조회 및 수정 백엔드 기능 구현
- Spring Security를 활용한 인증/인가 처리 및 JWT 토큰 기반 보안 구현
- OAuth 2.0을 이용한 소셜 로그인 기능 구현 및 처리

김지선 부팀장

- Figma를 활용한 메인화면, 헤더, 푸터, 로그인, 이력서 첨삭, 적성탐색, 상세검색, 요금제 화면, 자유 게시판, 공지사항 화면을 html 코드 베이스로 디자인
- React 와 MUI를 활용해 메인화면 UI/UX를 구현
- React 와 MUI를 활용해 헤더와 푸터 UI/UX를 구현
- React 와 MUI를 활용해 적성탐색 UI/UX를 구현
- React 와 MUI를 활용해 요금제화면 UI/UX를 구현
- 커리어넷 API를 활용한 적성탐색 기능 구현

추인철 부팀장

- React와 MUI를 활용해 상세검색 및 적성탐색 UI/UX를 구현
- 이용권 구매 UI/UX를 디자인 및 구현하여 사용자 경험 최적화
- 네이버 API를 이용해 메인 페이지 뉴스 구현
- 웹 크롤링을 이용해 상세검색 뉴스 구현
- 카카오 지도와 로컬 API를 이용해 상세검색 기능을 구현
- 사용자 검색 기록 및 회사 찜 목록을 백엔드에서 DB에 저장
- 커리어넷 API를 활용한 적성탐색 기능 구현 및 PDF로 결과 저장
- PDF 내용을 Chat-GPT API로 분석 후 프론트엔드로 전달 구현
- 공공데이터포털 API를 활용해 기업 정보를 수집 및 검색된 결과를 지도에 시각화하는 기능 구현

최일락 팀원

- React와 MUI를 활용해 챗봇 및 AI 면접 UI/UX 구현
- OpenAI API의 파인튜닝을 활용한 자연어 처리 및 응답 생성 기능 구현
- 실시간 대화형 사용자 인터페이스와 음성 인식 및 TTS 기능 통합
- 뮤직 플레이어 및 이모지 선택기 기능 구현
- GCS를 활용한 비디오 파일의 업로드 및 다운로드 관리 시스템 구축
- OpenAI API를 통한 이력서 기반 맞춤형 면접 질문 생성
- 모의 면접과 실전 면접 모드를 구분하여 개인화된 면접 경험 제공
- FastAPI 서버 연동으로 얼굴 인식 및 감정 분석 기능 통합
- OpenCV를 이용한 Head-eye 움직임 분석 알고리즘 통합
- asyncio를 활용한 비동기 작업 큐로 비디오 분석 프로세스 효율 관리
- Claude 및 Clova API를 활용한 AI 기반 면접 결과 분석 및 피드백 제공
- Clova Speech Recognition API로 음성-텍스트 변환(STT) 기능 구현

윤지은 팀원

- React와 MUI를 활용해 마이페이지 전체 UI/UX 설계 및 구현
- Python / React 기반으로 MQTT 메시징 프로토콜을 활용하여 회원 간 실시간 채팅 기능 구현 (1:1, 단체)
- 채팅 관련 UI/UX 설계 및 구현
- Mosquitto를 MQTT Broker로 설정하여 안정적인 데이터 통신을 지원
- 채팅 프론트엔드 및 백엔드 구현 및 연결
- MQTT 를 이용하여 채팅과 게시판의 실시간 알림 기능 프론트엔드 및 백엔드 구현 및 연결
- FastAPI 서버를 연동하여 데이터 관리

최미지 팀원

- MobX를 사용한 관리자 권한 검증 및 비관리자 페이지 접근 차단 기능 및 관리자 메인 페이지 UI/UX 설계 및 구현
- 공지사항 CRUD 및 공지사항 관리 페이지 UI/UX 설계 및 구현
- 비밀번호로 보호된 Q&A 조회 및 접근 제어, Q&A CRUD, Q&A 페이지, 관리자 문의관리 페이지 UI/UX 설계 및 구현
- 사용자 정보 조회, 수정, 비활성화 및 활성화, 탈퇴 기능 구현 및 관리자 회원관리 페이지 UI/UX 설계 및 구현
- 신고된 게시글/댓글 조회, 상태 업데이트, 영구 삭제 및 복구 기능 구현 및 관리자 신고관리 UI/UX 설계 및 구현
- 관리자 결제관리 페이지 UI/UX 설계 및 구현
- 삭제된 게시글/댓글 조회, 영구 삭제, 복구 기능 및 관리자 삭제관리 페이지 UI/UX 설계 및 구현

윤성빈 팀원

- 리액트 기반 이력서 등록 및 관리 페이지 UI 설계 및 구현
- 사용자 이력서 등록 및 삭제 백엔드 기능 구현 및 DB 설계
- 한스펠(hanspell by 부산대학교) api 사용 이력서 맞춤법 검사 기능 구현
- gpt-3.5-turbo 모델 사용 이력서 AI 첨삭 기능 구현
- 자바스크립트 html2canvas , jspdf 라이브러리 사용 이력서 등록 페이지 캡쳐 및 pdf 변환 기능 구현
- 메인 페이지 (초기화면) UI 아이디어 제공 및 구현

김세종 팀원

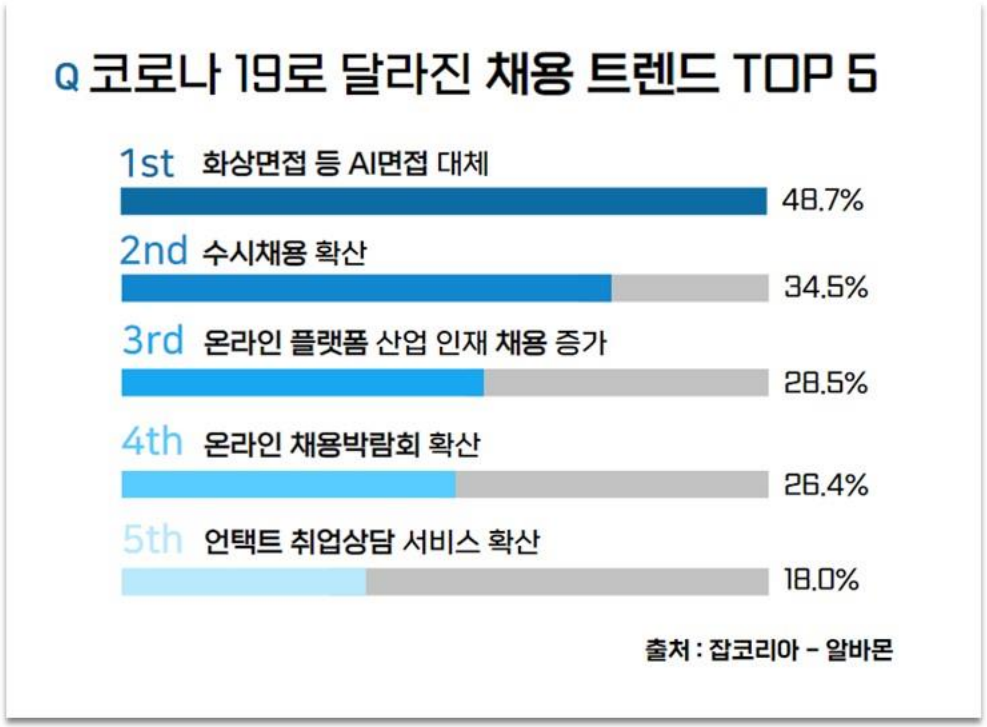
- 전체 게시판 CRUD 백엔드 구현
- 조회수, 댓글 기능 백엔드 구현
- 파일 업로드 기능 백엔드 구현
- 사용자 공지사항 UI 및 프론트엔드 구현
- 게시판 UI 구현

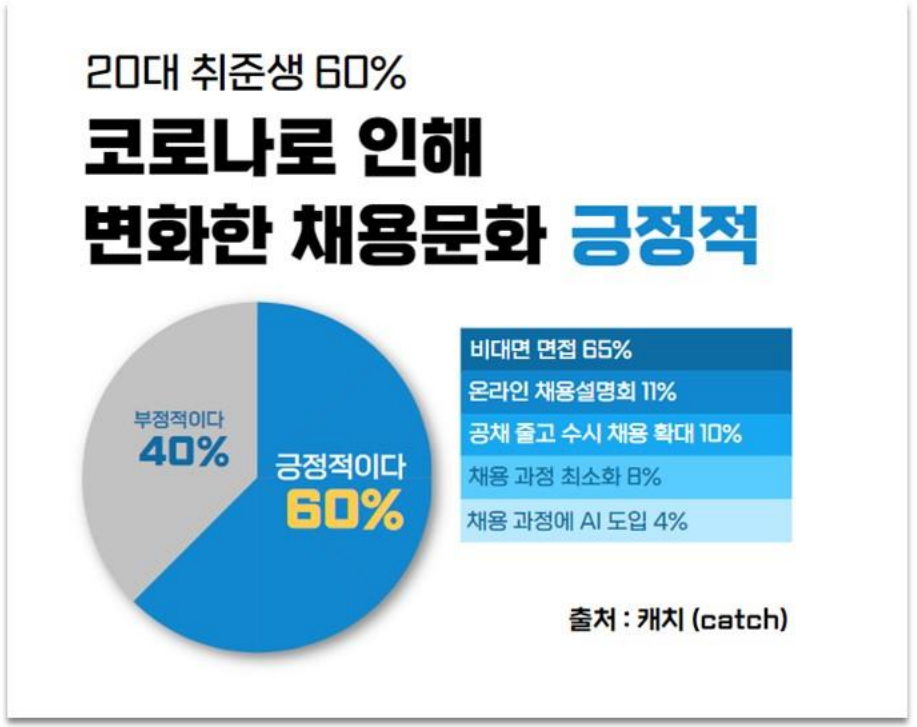
02

개발 배경

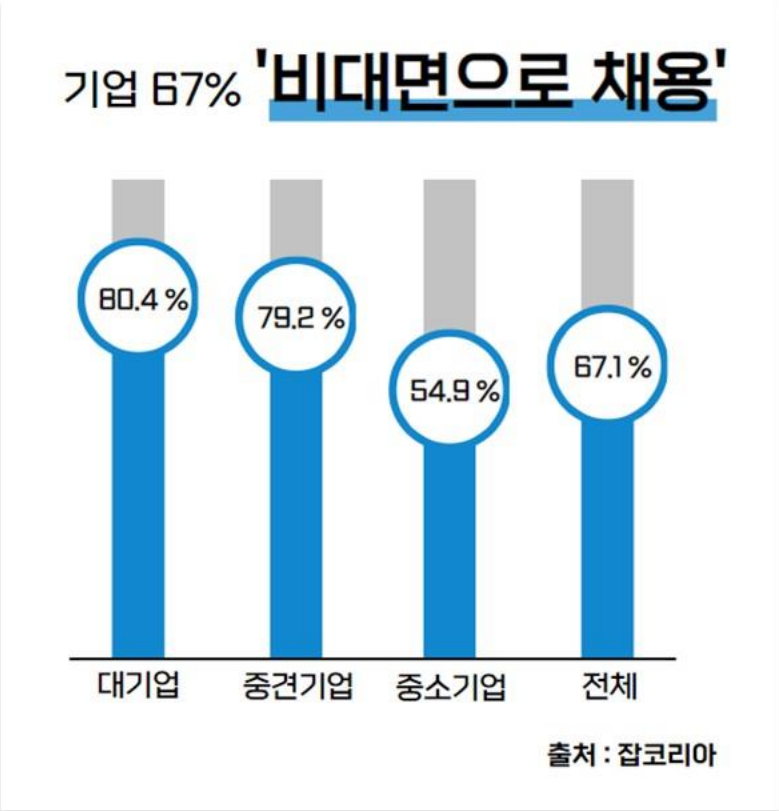
코로나 19 이후 채용 트렌드의 변화

“ 비대면 면접, 채용의 새로운 표준으로 자리잡다! ”





비대면 면접 도입 현황 및 증가 추세



“ 구직자에게 필수가 되어버린 비대면 면접 준비 ”

프라임경제

올해 대기업 82.7%, 언택트 채용 도입한다

[프라임경제] 코로나19가 불러온 언택트 시대를 맞아 각 기업들은 비대용 채용에 적극적인 분위기다. 인크루트가 비대면 알바채용 바로면접 알바콜과...

AI타임스

잡코리아 "AI 활용하면 합격률 높아져...매월 140% 증가"

특히 기업-인재 추천 및 매칭 정확도와 속도를 높인 원픽에서 가장 성과가 두드러진다. 공고 등록과 헤드헌팅을 결합한 형태의 서비스로, AI를 활용해...

구직자들의 준비 부족, 비대면 면접의 큰 장애물

구직자 10명 중 6명,

언택트 면접 부담!

출처 : 사람인

“정보 부족과 불안감, 막막한 취업 준비… 필요한건?”

쿠키뉴스

“취준생 10명 중 6명, 소극적 구직자…취업 환경 어두워”

대졸 채용시장의 한파가 지속되며 취업을 준비하는 청년들의 기대도 낮아졌다. 취업준비생 10명 중 6명은 '소극적 구직자'로 조사됐다.

공생공사닷컴

MZ구직자 10명 중 9명 “취업 준비하면서 막막하다”

MZ세대 구직자 10명 가운데 9명은 취업준비를 하면서 막막해한 것으로 나타났다. 이런 막막함은 자신의 스펙이 남들보다 부족하다고 느껴지거나 취업...

AI 면접의 도입과 확산

“ 찬반 논란 속 확산되는 AI 채용 ”

동아일보

AI 면접, 600여 기업 채용 과정에 도입… 화상인터뷰로 표정-억양 분석해 점수화

'AI 역량검사' 체험해 보니 비대면 채용 확산속 활용 기업 늘어 응시자 성향에 지적 역량까지 파악 일부선 탈락 자료로… 영향력 커져.

파이낸셜뉴스 · 11일

SK C&C, 하반기 공채 시작…전 과정에 'AI 채용 에이전트' 도입

[서울=뉴시스]송혜리 기자 = SK C&C는 오는 19일까지 하반기 신입 인재 채용 서류 접수를 진행한다고 10일 밝혔다. 특히 이 회사는 채용 전 과정에 인공지능(AI) 채용 에이전트…

SK 주식회사 C&C

“ AI 면접, 기업 채용의 새로운 기준 ”

Q 채용 과정의 AI의 도입, 어떻게 생각하십니까

찬성 60%

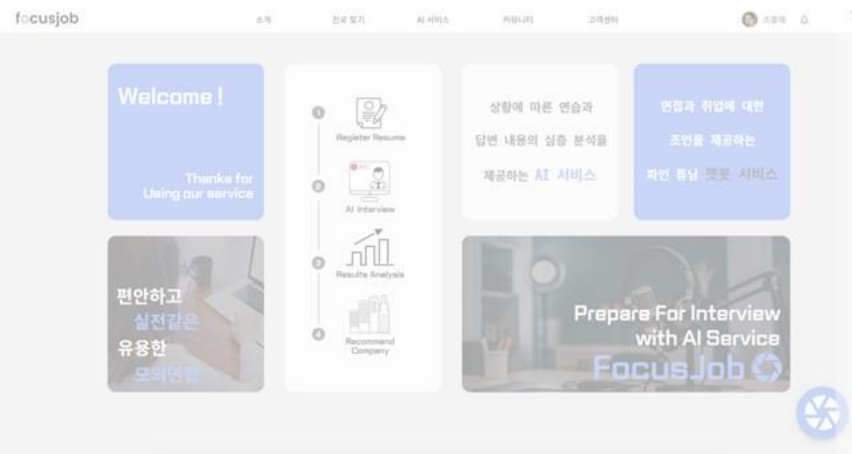
온라인 플랫폼 산업 인재 채용 증가	33%
채용비리 방지 가능성	26%
시간•장소 구애 받지 않음	26%

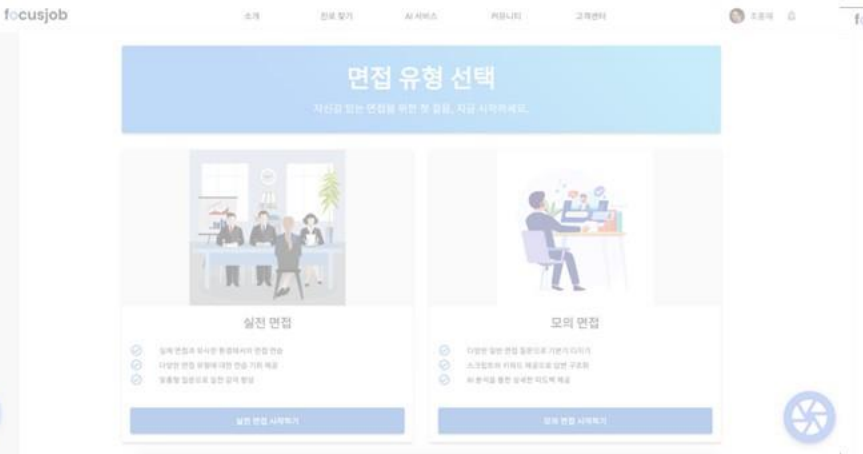
반대 40%

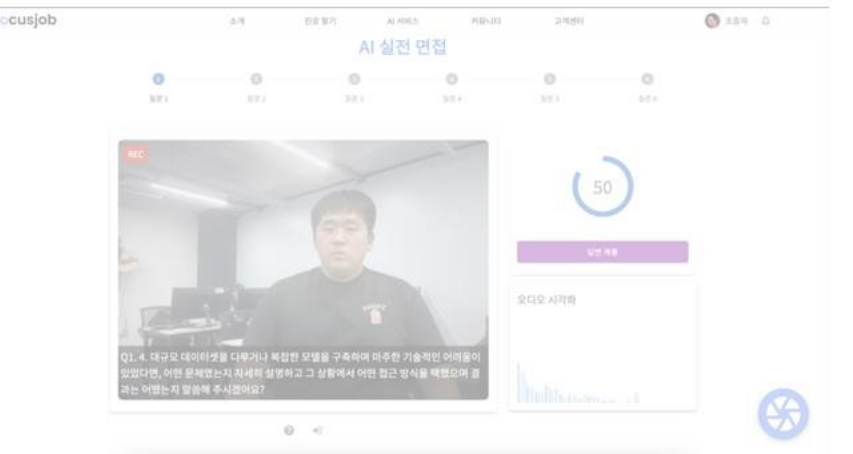
기술력의 한계	33%
평가기준 획일화 우려	26%
AI에게 평가 받는 데 거부감	26%

출처 : 인크루트

"포커스잡, 구직자들의 어려움을 해결하기 위해 개발되었습니다."







focusjob

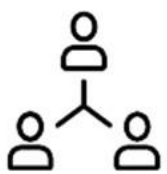
"취업을 준비하는 구직자들을 위한
맞춤형 AI 면접 연습 플랫폼으로,
사용자들에게 실전과 같은 면접 경험과
다양한 연습 기능을 제공합니다."

focusjob의 차별점 !

개인 이력서와 연계된 심층적인 면접 질문 및 시나리오 제공

AI 기술을 활용한 맞춤형 비대면 면접 연습

AI 분석을 통한 실시간 피드백 및 면접 결과 평가 시스템



구직자 커뮤니티와의 연계를 통한 면접 준비 및 경험 공유

03

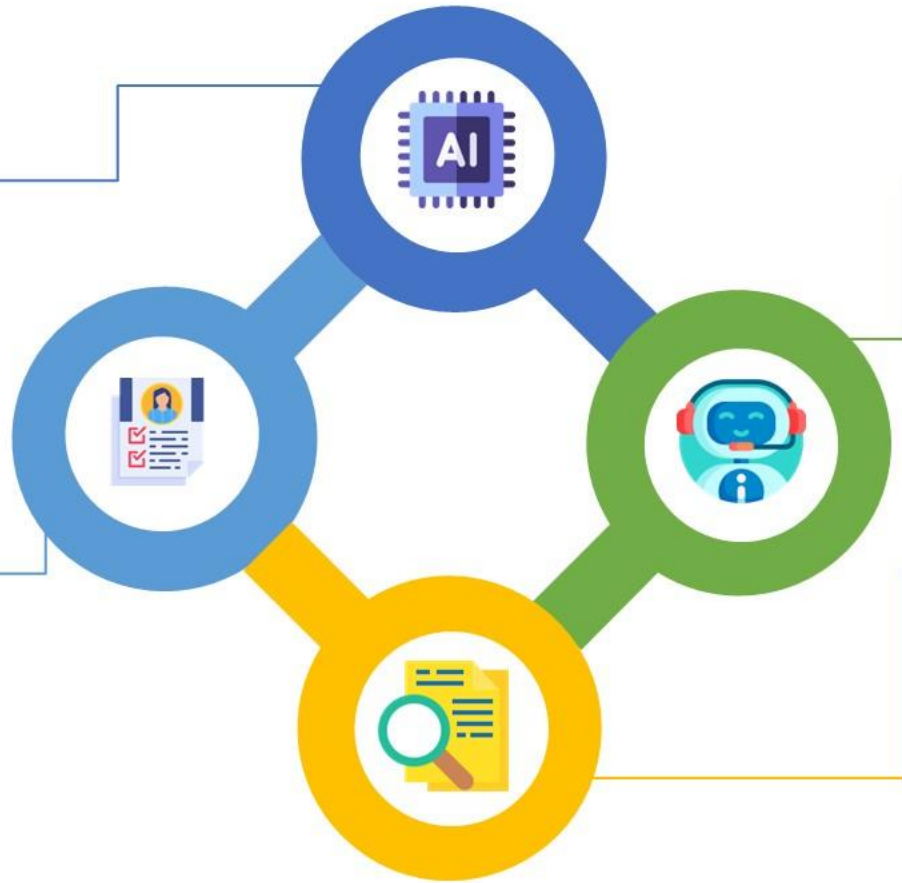
프로젝트 주요 기능

AI 면접

AI 기반 면접 질문 생성 및
맞춤형 피드백 제공

이력서 첨삭

기업 맞춤형 이력서 등록 및
AI 기반 이력서 첨삭



챗봇

실시간 대화 기능을 갖춘 지능형
챗봇 시스템을 구축

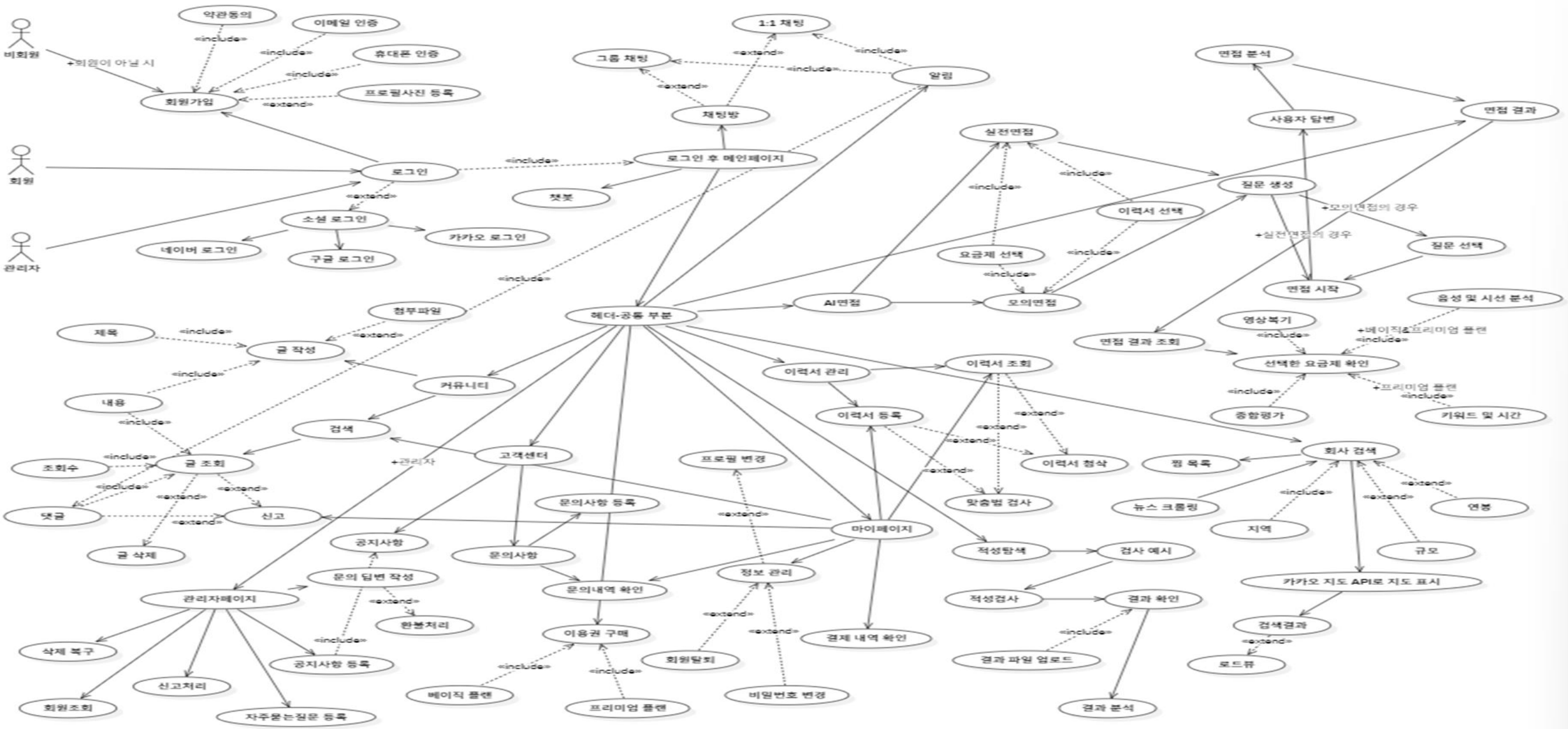
적성 탐색

커리어넷 API를 통해 개인
맞춤 직업 추천

04

프로젝트 설계

유스케이스 다이어그램

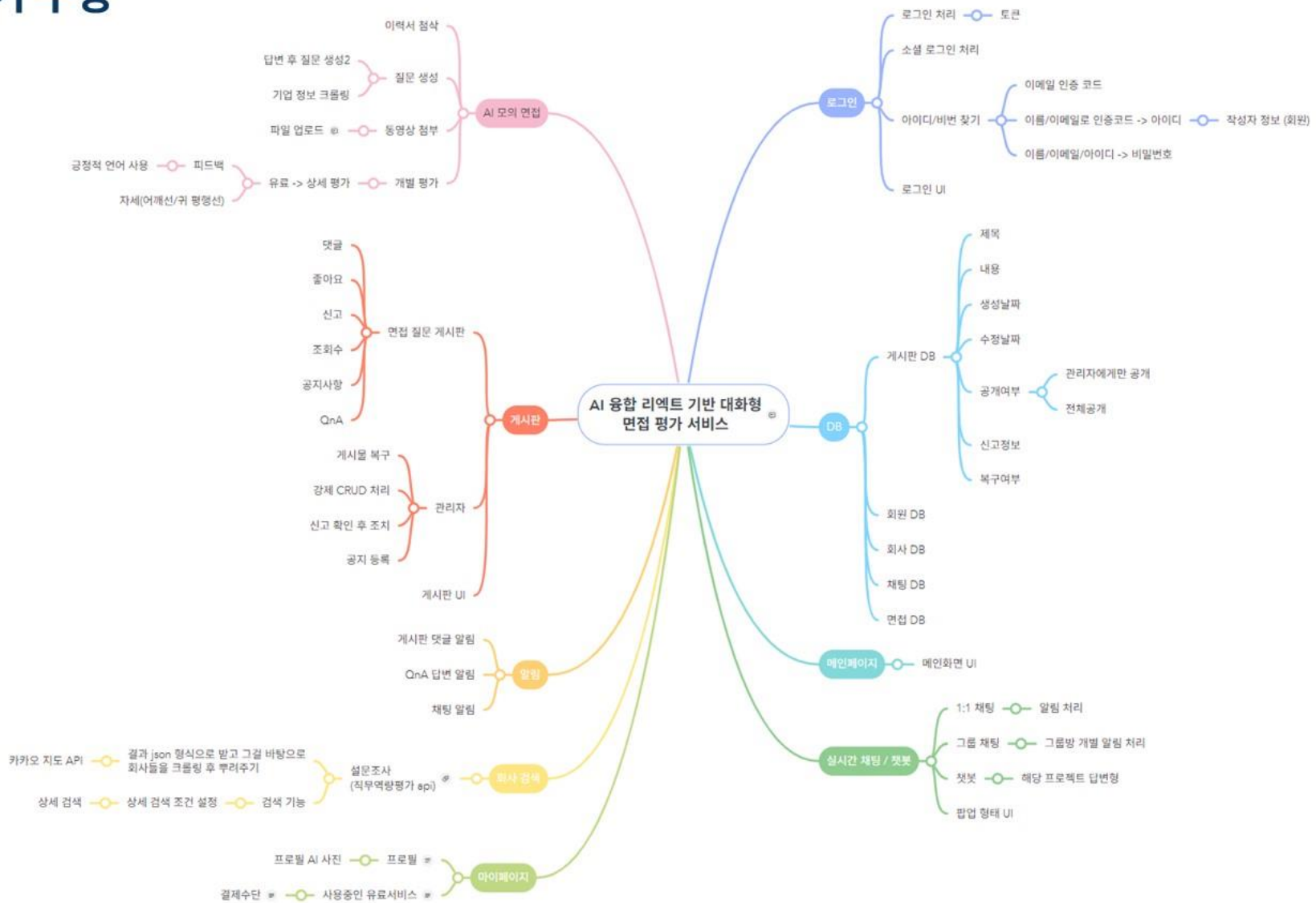

비회원
+회원이 아닐 시
약관동의
이메일 인증
휴대폰 인증
프로필사진 등록
회원가입
회원
관리자
로그인
소셜 로그인
네이버 로그인
구글 로그인
카카오 로그인
로그인 후 메인페이지
챗봇
1:1 채팅
그룹 채팅
채팅방
알림
헤더-공통 부분
첨부파일
제목
글 작성
내용
커뮤니티
검색
글 조회
조회수
댓글
글 삭제
신고
고객센터
+관리자
공지사항
문의 답변 작성
관리자페이지
삭제 복구
신고처리
회원조회
자주묻는질문 등록
공지사항 등록
완료처리
문의사항 등록
문의사항
문의내역 확인
이용권 구매
베이직 플랜
프리미엄 플랜
회원탈퇴
프로필 변경
정보 관리
비밀번호 변경
결제 내역 확인
AI면접
모의면접
실전면접
요금제 선택
이력서 선택
이력서 관리
이력서 조회
이력서 등록
이력서 첨삭
맞춤법 검사
마이페이지
적성탐색
검사 예시
적성검사
결과 확인
결과 파일 업로드
결과 분석
면접 분석
사용자 답변
면접 결과
질문 생성
+모의면접의 경우
+실전면접의 경우
질문 선택
면접 시작
면접 결과 조회
영상복기
음성 및 시선 분석
+베이직&프리미엄 플랜
선택한 요금제 확인
종합평가
+프리미엄 플랜
키워드 및 시간
회사 검색
찜 목록
뉴스 크롤링
지역
연봉
규모
카카오 지도 API로 지도 표시
검색결과
로드뷰
include
extend

디렉토리 구조도 및 역할 분담

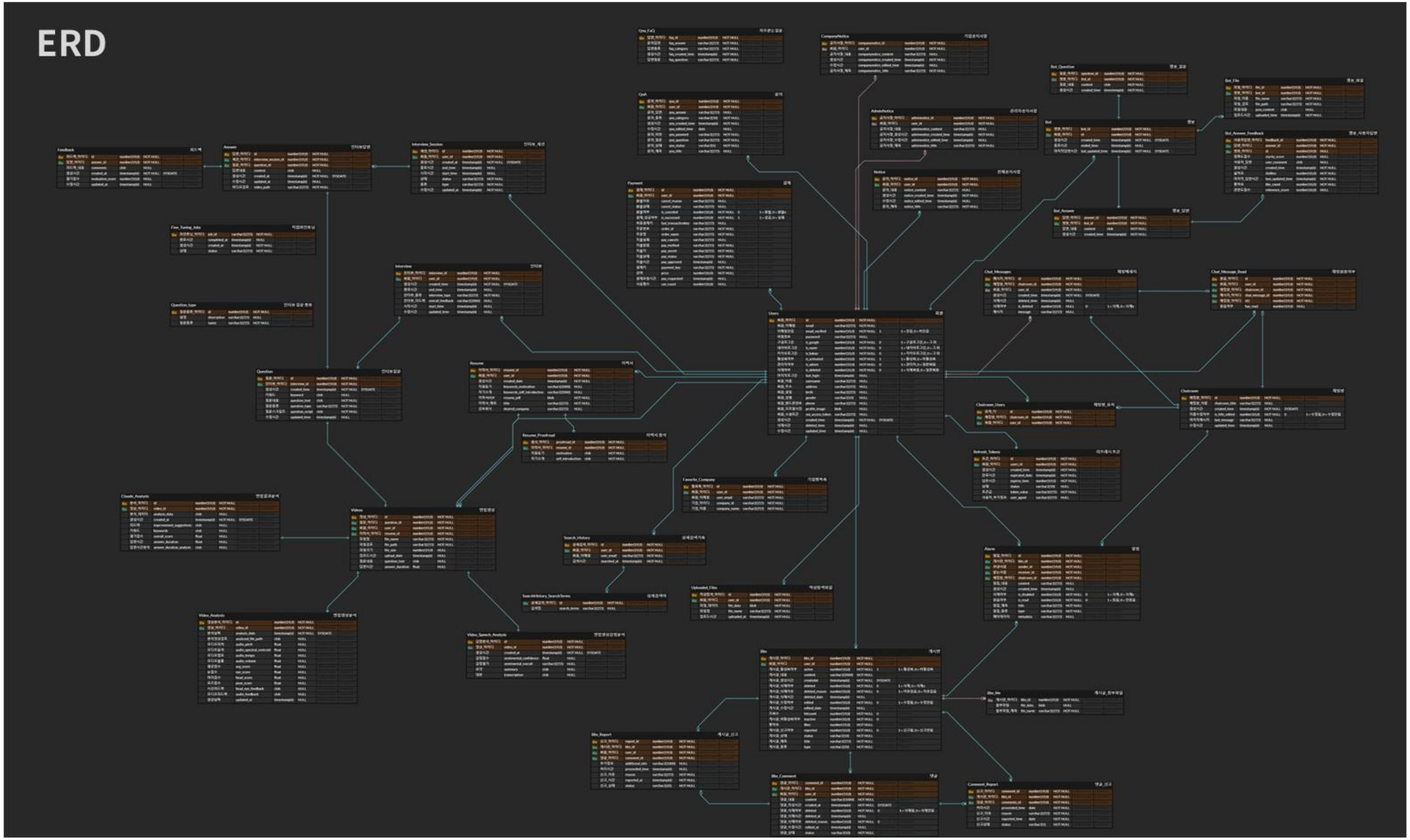
디렉토리 구조도 및 팀원 역할 분담표

팀명	2팀		팀장	조용재	팀원	김지선, 윤지은, 윤성번, 최일락, 최미지, 추인철, 김세종		
과정명	한국 ICT 고기 AI 융합 리액트 기반 대화형 면접 평가 서비스(2024년 6월 17일 ~ 2024년 9월 20일)							
구분	대분류	중분류	소분류	폴더명	화면내용	주담당팀원	보조팀원	비고
FRONT END단	메인화면	메인화면 페이지		/	메인화면	김지선		
	로그인	로그인	로그인 페이지	/auth	로그인 UI	조용재		
			로그인	/auth	일반 사용자 로그인 처리	조용재		JWTToken
		일반 회원가입	회원가입 페이지	/auth/register	회원가입 UI	조용재		
			회원가입	/auth/register	회원가입 처리 및 유효성 검사	조용재		
		소셜 로그인	소셜 로그인	/auth/sregister	소셜 로그인 API 처리	조용재		카카오, 구글, 네이버 API
		아이디/비번 찾기	아이디/비번 찾기 페이지	/auth/find	아이디/비밀번호 찾기 UI	조용재		
			아이디 찾기	/auth/find	아이디 찾기 처리(이름, 이메일 입력)	조용재		
			비밀번호 찾기	/auth/find	비밀번호 찾기 처리(이름, 아이디, 비밀번호)	조용재		
			이메일 인증 코드	/auth/find	입력한 이메일로 인증코드 처리	조용재		구글 SMTP
	마이페이지	마이페이지		/mypage	마이페이지 UI	윤지은		
		회원정보	면접 결과 정보	/mypage	면접 결과 정보 보여주기	윤지은		
			설문 결과 정보	/mypage	설문 결과 정보 보여주기	윤지은		
			작성글 및 댓글	/mypage	작성글 및 댓글 확인	윤지은		
			QnA 및 신고 내역	/mypage	문의사항(QnA) 및 신고 내역 보여주기	윤지은		
			결제정보	/mypage	결제정보 확인	윤지은		
			차단한 사용자 정보	/mypage	차단한 사용자 목록 보여주기	윤지은		
			회원정보 수정 페이지	/mypage	회원정보 수정 UI	윤지은		
			회원정보 수정	/mypage	회원정보 수정 및 유효성 검사 처리	윤지은		
		프로필사진 설정	프로필 사진 수정 팝업	/mypage	프로필 사진 수정 팝업 UI	윤지은		
			프로필 사진 수정	/mypage	프로필 사진 AI 혹은 파일 수정 처리	윤지은		DALL-E
	직무/회사 검색	사용자 설문조사	사용자 설문조사 페이지	/survey	설문조사 UI	김지선	추인철	
			사용자 설문조사 입력	/survey	사용자 설문조사 답변 처리	김지선	추인철	
		설문조사 AI 평가 결과	평가 결과 페이지	/survey/result	설문조사 결과UI	김지선	추인철	
			맞춤 회사 추천	/survey/result	설문조사 결과 AI 처리	김지선	추인철	데이터공공포털API
		상세 검색	상세검색 페이지	/search	상세검색 UI	추인철		
			상세검색 지도	/search	상세검색 지도 화면	추인철		카카오지도 API
			상세검색 검색조건	/search	상세검색 조건 선택 처리	추인철		
			상세검색 결과 링크	/search	상세검색 결과 회사 링크 연결 처리	추인철		데이터 크롤링
	실시간 채팅	팝업		/chat	채팅 팝업UI	윤지은		
		1:1 채팅	1:1 채팅 팝업	/chat	1:1 채팅 UI	윤지은		
			1:1 채팅	/chat	MQTT을 활용한 1:1 채팅 처리	윤지은		MQTT
		그룹 채팅	그룹 채팅 팝업	/chat	그룹 채팅UI	윤지은		
			그룹 채팅	/chat	MQTT을 활용한 그룹 채팅 처리	윤지은		MQTT
	챗봇	챗봇 팝업		/bot	챗봇 팝업 UI	최일락		
		챗봇 답변		/bot	챗봇 답변 처리	최일락		
	이력서 첨삭	이력서 첨삭	이력서 입력UI	/resume	이력서 입력 폼은 등록 페이지 UI	윤성번		
			이력서 입력 처리	/resume	이력서 입력 처리	윤성번		
			이력서 AI 첨삭 처리	/resume	이력서 AI 학습 후 첨삭 처리	윤성번		chat gpt API
		이력서 결과	이력서 결과 페이지UI	/resume/result	이력서 결과 페이지 UI	윤성번		
			이력서 결과 AI 처리	/resume/result	이력서 검사 결과 AI 처리	윤성번		
	면접 (알락, 지선, 성번, 세종)	AI 모의 면접	AI 모의 면접 페이지	/interview	모의 면접 UI	최일락		
			AI 모의 면접 가이드	/interview	면접 셋팅 가이드(카메라, 마이크 설정 등)	최일락		webRTC
			AI 모의 면접 관리자 질문	/interview	사용자의 답변 인식 후 관리자 답변 처리	최일락		
			AI 모의 면접 AI 질문	/interview	사용자의 답변 인식 후 AI 답변 처리	최일락		chat gpt API
			AI 모의 면접	/interview	AI 모의 면접 처리	최일락		
		AI 모의 면접 결과	AI 모의 면접 결과 페이지	/interview/result	모의 면접 결과 UI	최일락		
			AI 모의 면접 속약 결과	/interview/result	점수 형태로 AI 결과 처리	최일락	조용재	chat gpt API
			AI 모의 면접 상세 결과	/interview/result	결제 시스템을 통해 페이지 확장 및 결과 처리	최일락	조용재	로스페이먼츠 API, chat gpt API
			AI 모의 면접 피드백	/interview/feedback	상세 결과를 통한 AI 피드백 처리	최일락	조용재	chat gpt API
	게시판	게시판 페이지		/bbs	게시판 UI	김세종		
		공지사항		/bbs	공지사항 처리	김세종		
		QnA		/bbs	QnA 처리	김세종		
	기타	실시간 알림	1:1 채팅 알림	/chat	1:1 채팅 실시간 알림 처리(메인화면과 연동)	윤지은		
			그룹 채팅 알림	/chat	그룹 채팅 실시간 알림 처리(메인화면과 연동)	윤지은		
			QnA 답변 알림	/bbs	관리자 QnA 답변 실시간 알림 처리(메인화면과 연동)	윤지은		
BACK END단	메인화면	메인화면 페이지		/	메인화면	김지선		
	로그인	로그인 페이지		/auth	로그인 UI	조용재		
		관리자 로그인		/auth	관리자 로그인 처리	조용재		로그인 화면은 사용자랑 같음
	관리자페이지	관리자페이지		/admin/mypage	마이페이지 UI	최미지		
		회원정보 수정	관리자정보 수정 페이지	/admin/mypage	관리자정보 수정 UI	최미지		
			관리자정보 수정	/admin/mypage	관리자정보 수정 처리	최미지		
			일반 사용자 목록	/admin/mypage	일반 사용자 목록 확인 처리	최미지		
			일반 사용자 관리	/admin/mypage	일반 사용자 일부, 수정, 추가 등 처리	최미지		
		프로필사진 설정	프로필 사진 수정 팝업	/admin/mypage	프로필 사진 수정 팝업 UI	최미지		
			프로필 사진 수정	/admin/mypage	프로필 사진 AI 혹은 파일 수정 처리	최미지		DALL-E
	직무/회사 검색	설문조사 관리 페이지		/admin/survey	설문조사 관리 페이지 UI	김지선	추인철	
		설문조사 사용자 통계		/admin/survey	표, 그래프를 활용한 설문조사 사용자 통계 처리	김지선	추인철	
		설문조사 질문 설정		/admin/survey	설문조사 질문 수정 처리	김지선	추인철	
	챗봇	챗봇 답변 관리		/admin/chat	챗봇 답변 수정 처리	최일락		
	이력서 첨삭	이력서 통계	이력서 통계 페이지	/admin/resume	이력서 통계 페이지 UI	윤성번		
			이력서 통계 처리	/admin/resume	이력서 통계 처리	윤성번		
	면접	면접 관리 페이지	면접 관리 페이지	/admin/interview	면접 관리 처리 페이지 UI	최일락		
			면접 셋팅 가이드 작성	/admin/interview	면접 셋팅 가이드 처리(카메라, 마이크 설정 등)	최일락		webRTC
		AI 모의 면접	AI 모의 면접 질문 설정	/admin/interview	모의 면접 질문 작성 및 수정 처리	최일락		chat gpt API
			AI 모의 면접 평가 기준 설정	/admin/interview	AI의 평가기준 설정 처리	최일락	조용재	chat gpt API
			AI 모의 면접 통계	/admin/interview	실제 사용자들의 면접 통계 처리	최일락	조용재	
	게시판	게시판 관리 페이지		/admin/bbs	게시판 글 수정, 삭제 등 처리	김세종		
		공지사항 작성		/admin/bbs	공지사항 등록, 수정, 삭제 등 처리	김세종		
		QnA 답변		/admin/bbs	사용자 QnA 답변 처리	김세종		
	기타	실시간 알림	1:1 채팅 알림	/chat	1:1 채팅 실시간 알림 처리(메인화면과 연동)	윤지은		
			그룹 채팅 알림	/chat	그룹 채팅 실시간 알림 처리(메인화면과 연동)	윤지은		
			QnA 등록 알림	/bbs	사용자 QnA 질문 실시간 알림 처리(메인화면과 연동)	윤지은		

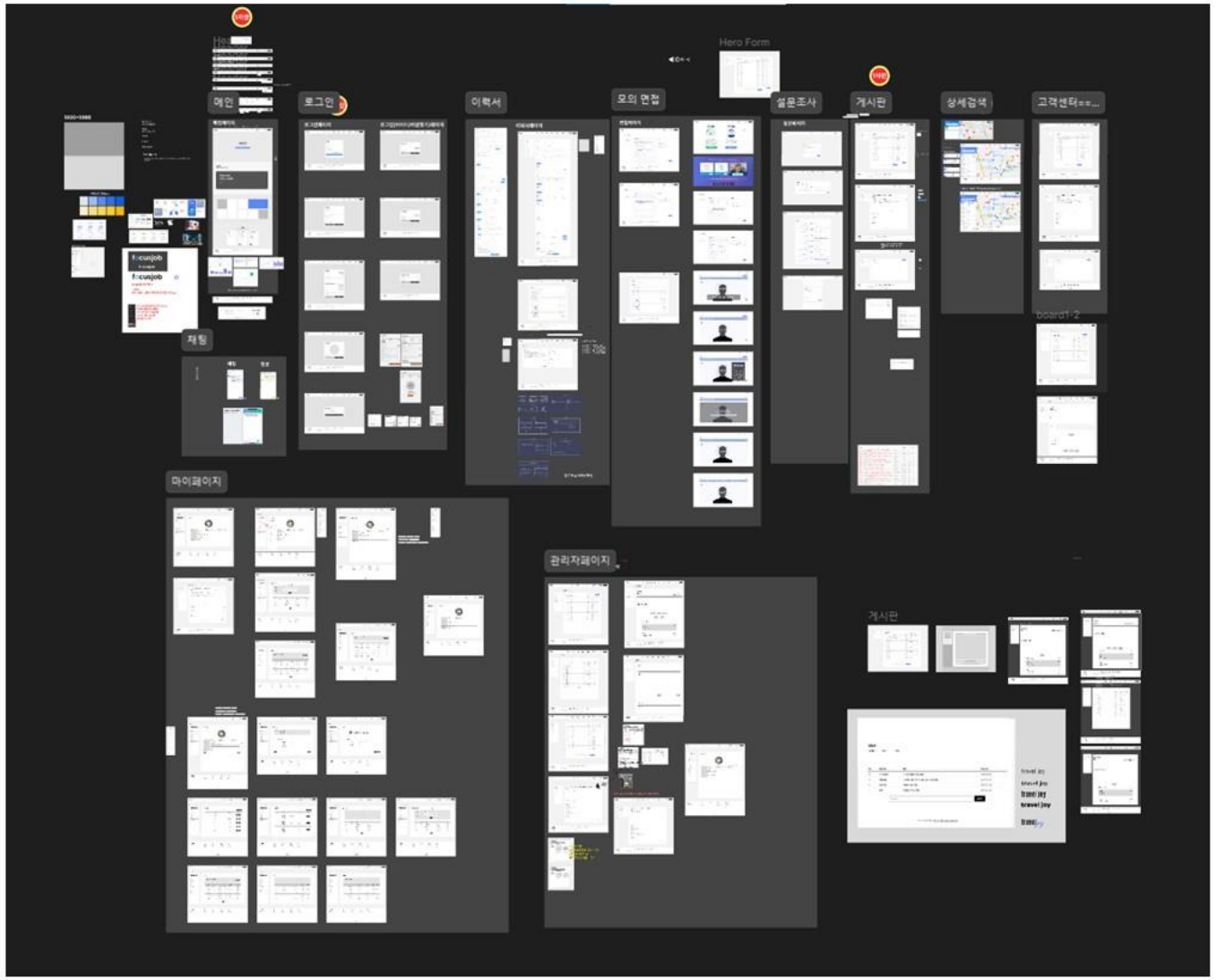
마인드맵_초기 구상


AI 융합 리엑트 기반 대화형 면접 평가 서비스
AI 모의 면접
이력서 첨삭
질문 생성
답변 후 질문 생성2
기업 정보 크롤링
파일 업로드
동영상 첨부
개별 평가
유료 -> 상세 평가
긍정적 언어 사용
피드백
자세(어깨선/귀 평행선)
게시판
면접 질문 게시판
댓글
좋아요
신고
조회수
공지사항
QnA
관리자
게시물 복구
강제 CRUD 처리
신고 확인 후 조치
공지 등록
게시판 UI
알림
게시판 댓글 알림
QnA 답변 알림
채팅 알림
회사 검색
설문조사
(직무역량평가 api)
카카오 지도 API
결과 json 형식으로 받고 그걸 바탕으로 회사들을 크롤링 후 뿌려주기
상세 검색
상세 검색 조건 설정
검색 기능
마이페이지
프로필
프로필 AI 사진
결제수단
사용중인 유료서비스
로그인
로그인 처리
토큰
소셜 로그인 처리
아이디/비번 찾기
이메일 인증 코드
이름/이메일로 인증코드 -> 아이디
작성자 정보 (회원)
이름/이메일/아이디 -> 비밀번호
로그인 UI
DB
게시판 DB
제목
내용
생성날짜
수정날짜
공개여부
관리자에게만 공개
전체공개
신고정보
복구여부
회원 DB
회사 DB
채팅 DB
면접 DB
메인페이지
메인화면 UI
실시간 채팅 / 챗봇
1:1 채팅
알림 처리
그룹 채팅
그룹방 개별 알림 처리
챗봇
해당 프로젝트 답변형
팝업 형태 UI

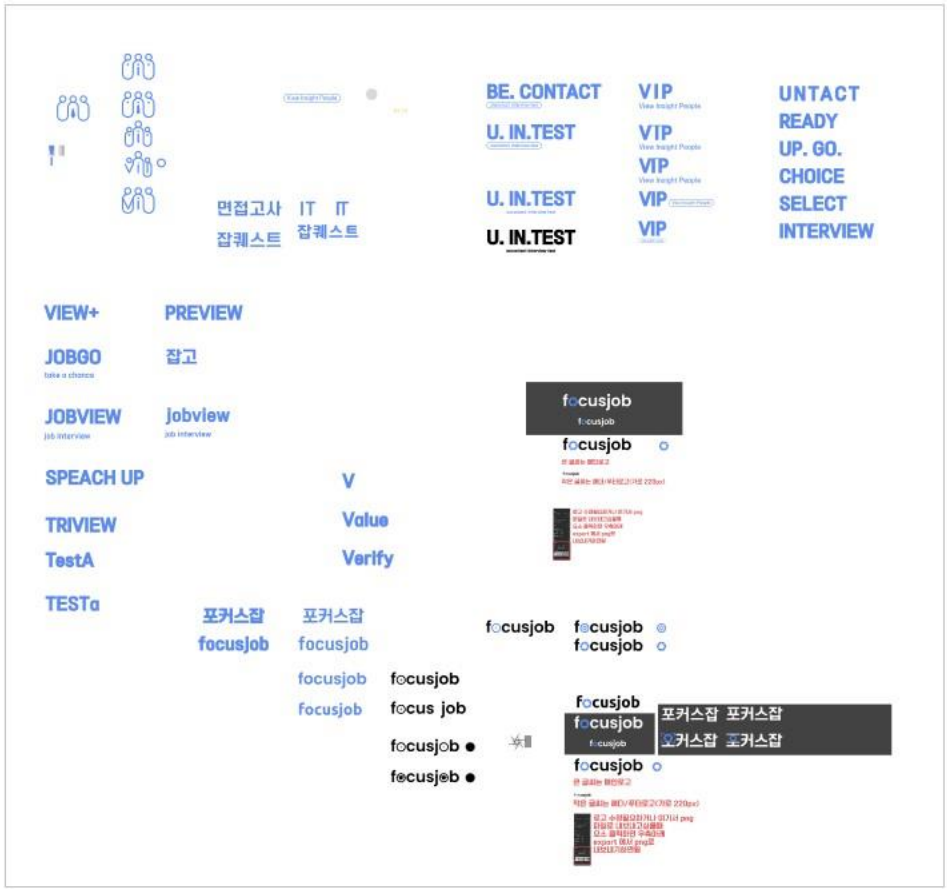
ERD


Qna_Cat
CompanyNotice
Bot_Question
Bot_File
Qna
AdminNotice
Bot
Bot_Answer_Feedback
FreeBoard
Answer
Interview_Session
Payment
Notice
Bot_Answer
Free_Tuning_Jobs
Interview
Question_type
Chat_Messages
Chat_Message_Read
Question
Resume
Users
Chatroom
Chatroom_Users
Resume_Proofread
Refresh_Tokens
Favorite_Company
Create_Analysis
Video
Search_History
Alarm
Unloaded_Files
Video_Analysis
Video_Speech_Analysis
Bbs
Bbs_File
Bbs_Report
Bbs_Comment
Comment_Report

UI 설계


Hero Form
메인
로그인
이력서
모의 면접
설문조사
게시판
상세검색
고객센터==...
채팅
마이페이지
관리자페이지
게시판

Figma


면접고사 IT IT
잡퀘스트 잡퀘스트
BE. CONTACT
U. IN.TEST
U. IN.TEST
U. IN.TEST
VIP
VIP
VIP
VIP
VIP
UNTACT
READY
UP. GO.
CHOICE
SELECT
INTERVIEW
VIEW+
PREVIEW
JOBGO
잡고
JOBVIEW
Jobview
SPEACH UP
V
TRIVIEW
Value
TestA
Verify
TESTa
포커스잡 포커스잡
focusjob focusjob
focusjob focusjob
focusjob focus job
focusjob
focusjob
focusjob
focusjob
focusjob
focusjob
focusjob
포커스잡 포커스잡
오커스잡 포커스잡
focusjob

기획설계

slack
Figma

형상관리

git
GitHub

DB

ORACLE

AI

MediaPipe
XGBoost
Pydantic
PyTorch
OpenCV
Keras
scikit-learn
AI Hub
colab
Joblib

Front-End

HTML
JS
CSS
NEXT.JS
node.js
Bootstrap
jQuery
AXIOS
Recharts
AJAX

Back-End

Java
spring boot
Spring Security
JWT
Maven
Apache Tomcat
{JSON}
JPA
Selenium
wordcloud
Flask
JACKSON
Beautiful Soup
Pandas
OAUTH 2
NumPy

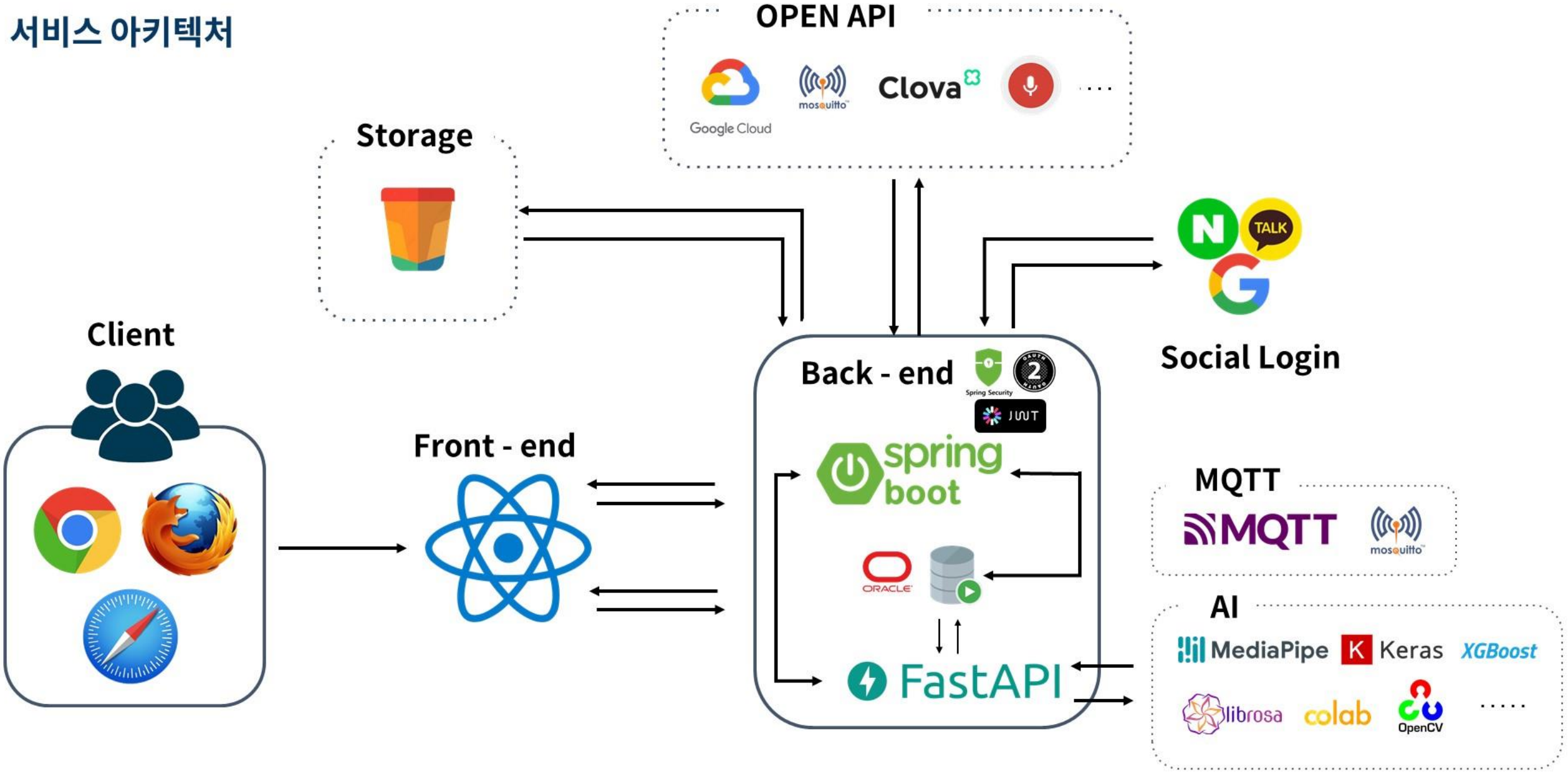
API

MQTT
Clova
kakaomap Maps API
coolsms
mosquitto
FastAPI
Google Cloud
TALK
toss payments
DATA
Swagger

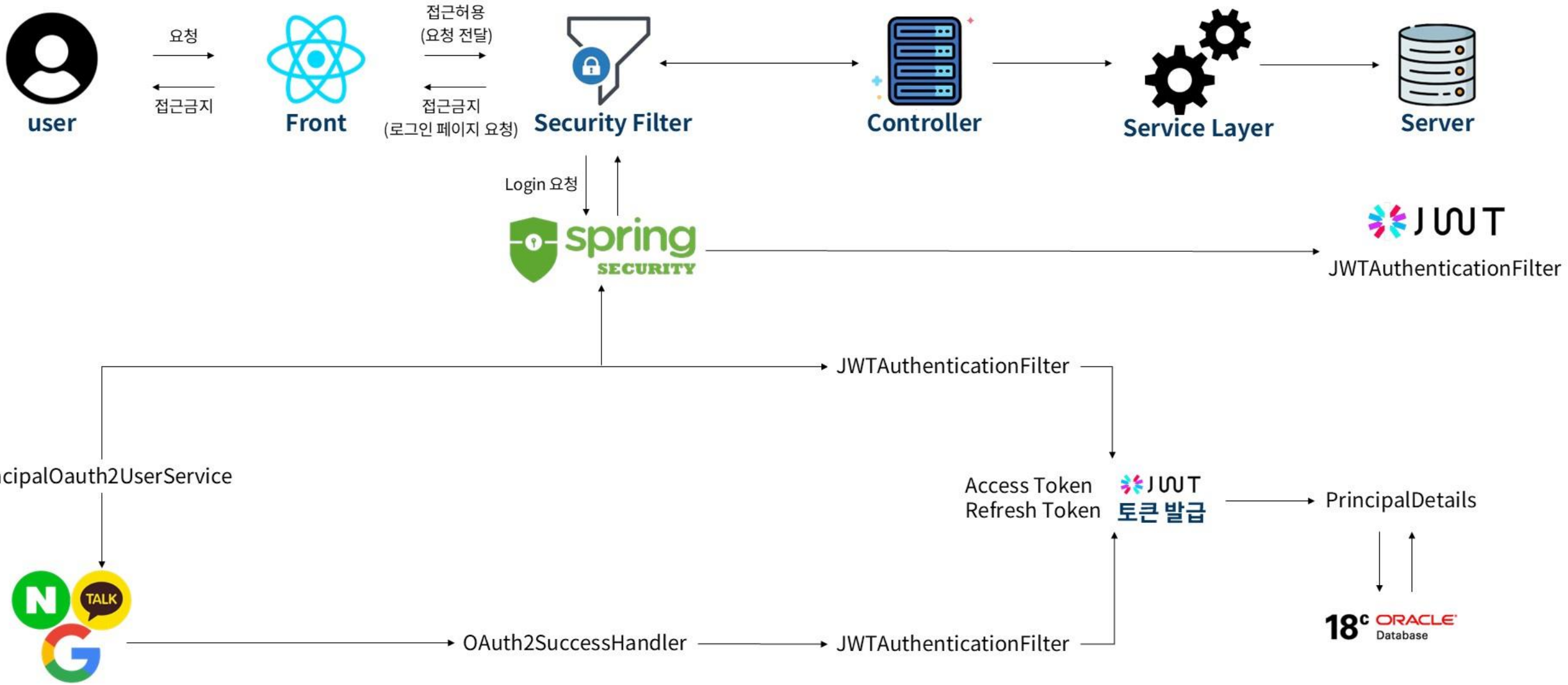
TOOL

eclipse
PC
POSTMAN
StarUML
ERD
jsPDF
matplotlib
seaborn
MUI
SciPy
ERMASTER

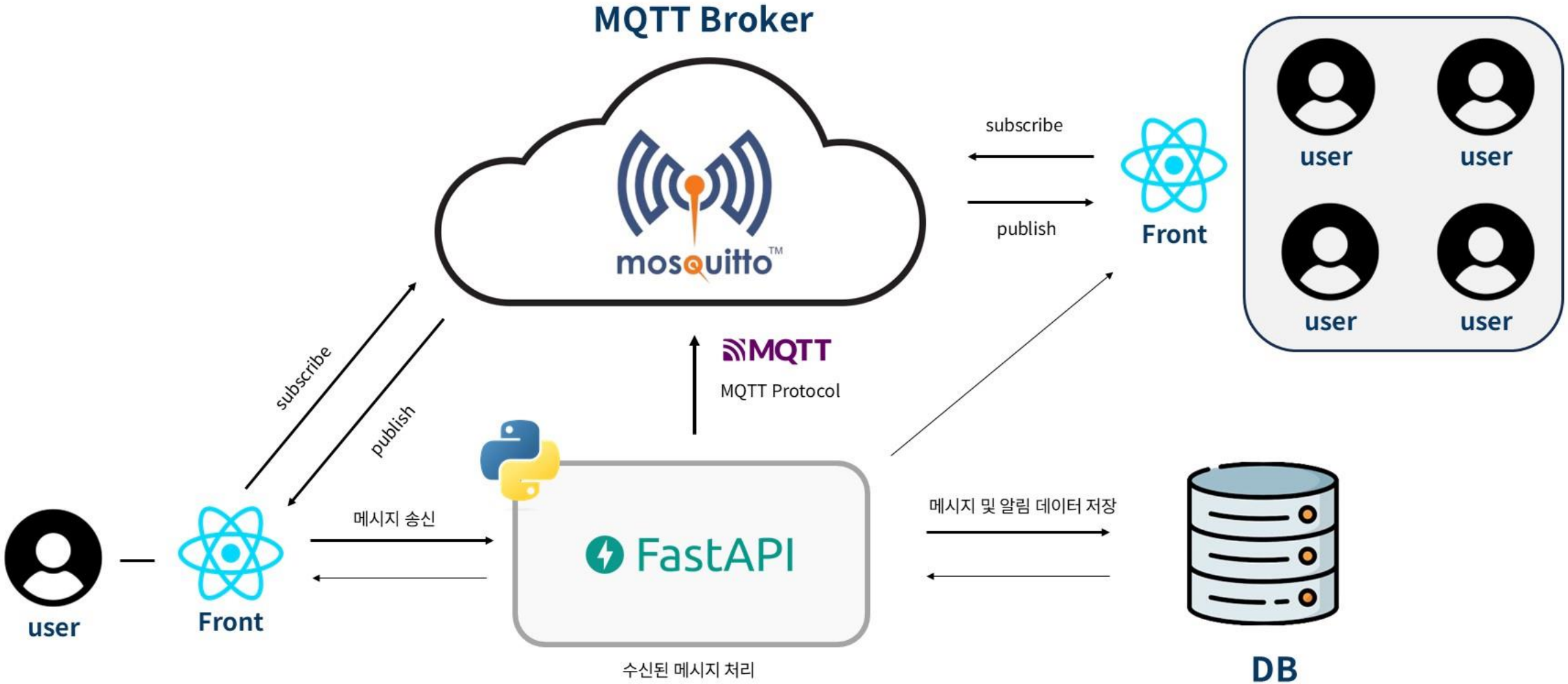
서비스 아키텍처


OPEN API
Google Cloud
mosquitto
Clova
Storage
Social Login
Client
Front - end
Back - end
Spring Security
JWT
spring boot
ORACLE
FastAPI
MQTT
MQTT
mosquitto
AI
MediaPipe
Keras
XGBoost
librosa
colab
OpenCV

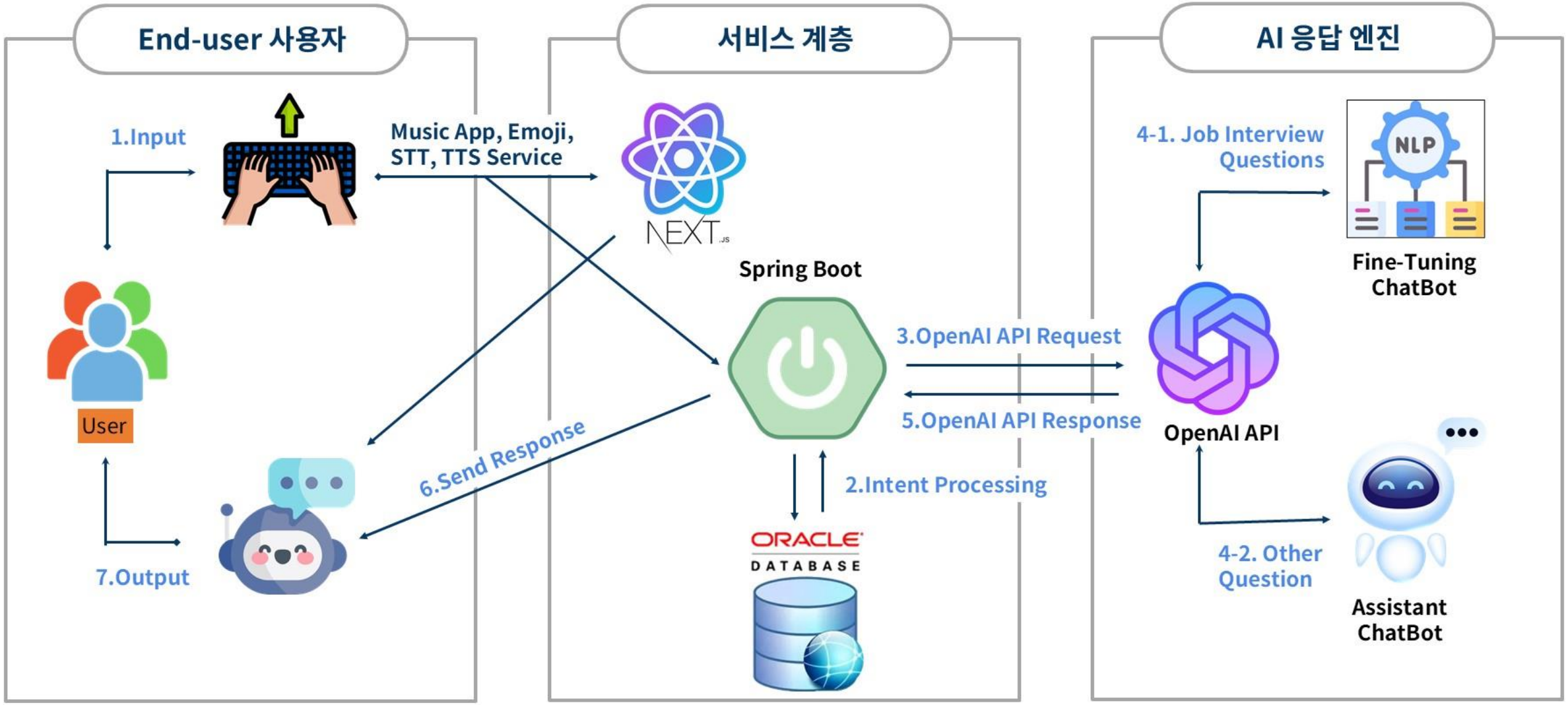
Security & JWT



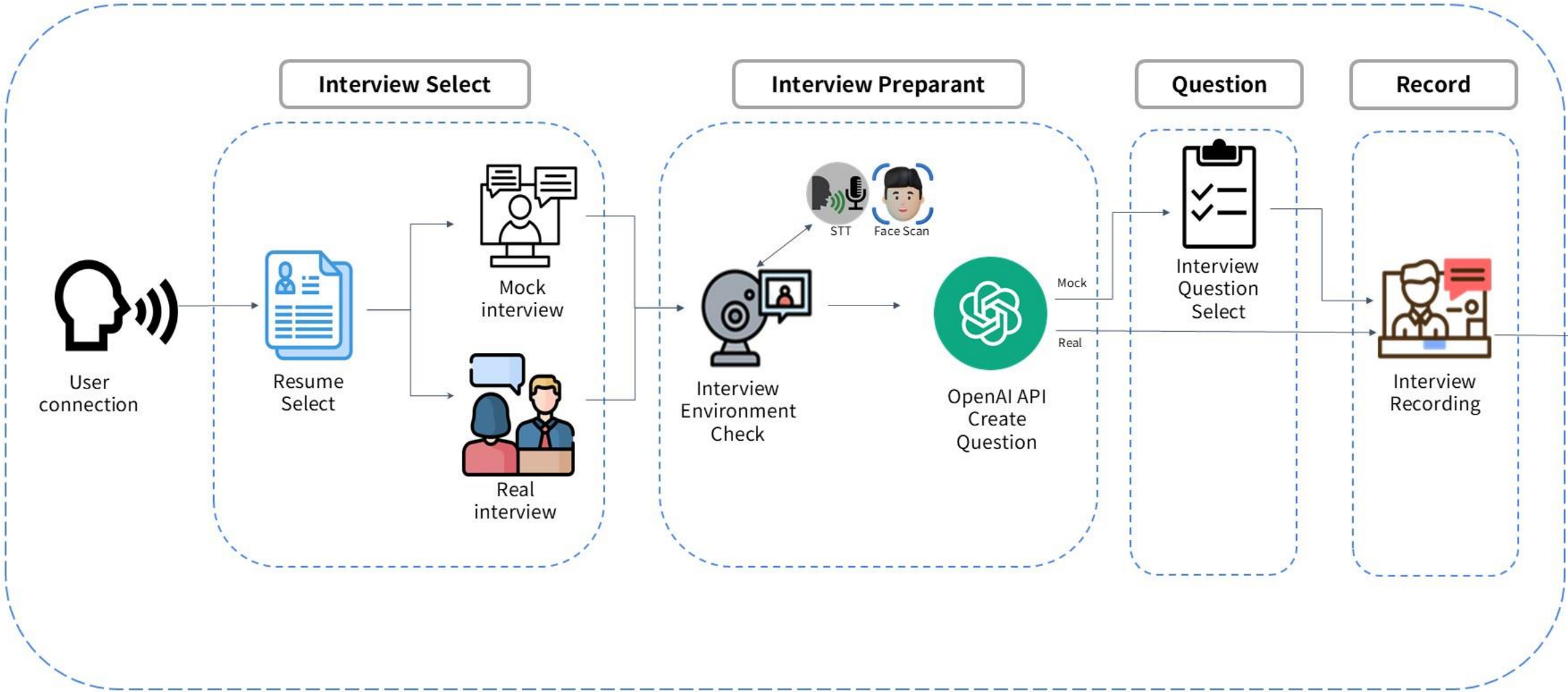
실시간 채팅 및 알림 (MQTT)



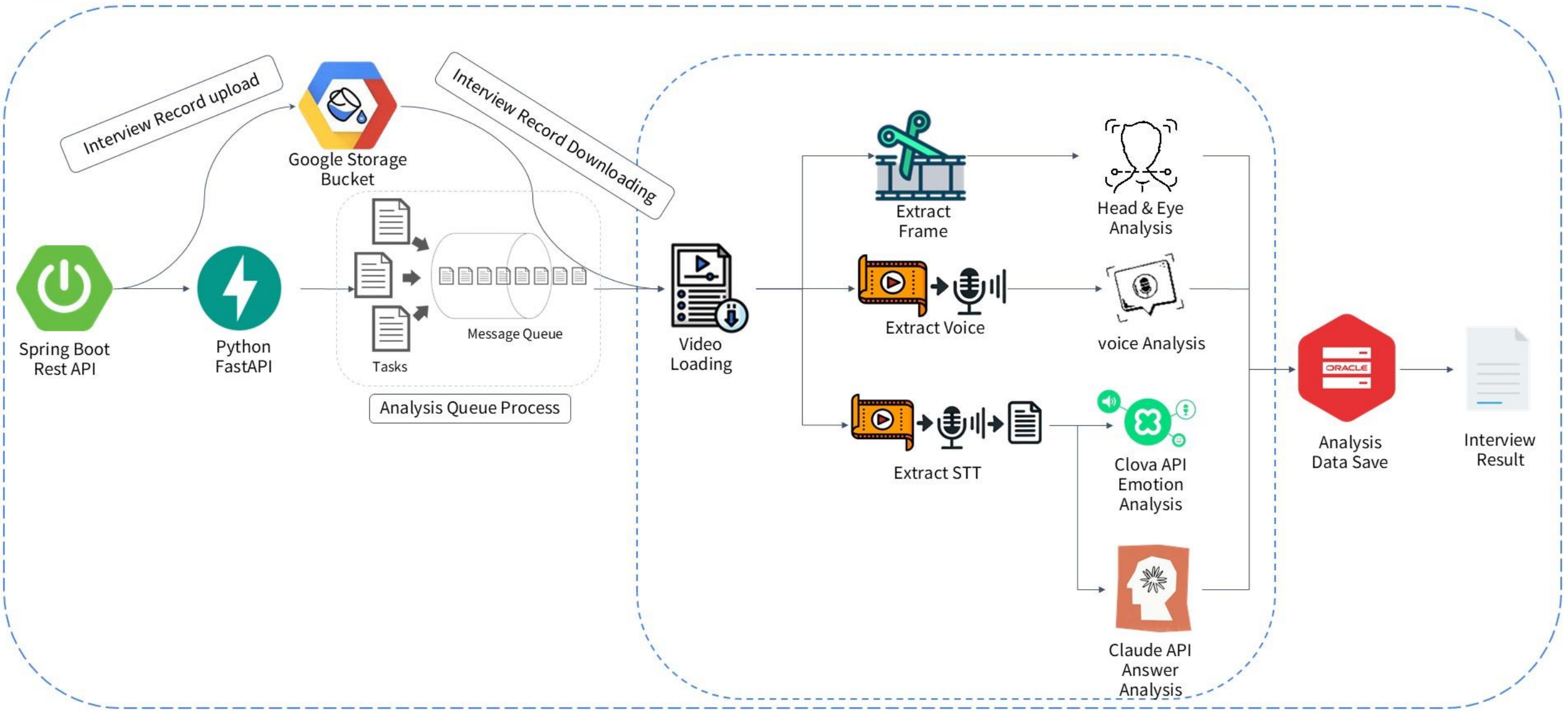
AI 챗봇


End-user 사용자
서비스 계층
AI 응답 엔진
1.Input
Music App, Emoji,
STT, TTS Service
NEXT.JS
Spring Boot
User
3.OpenAI API Request
5.OpenAI API Response
6.Send Response
2.Intent Processing
7.Output
ORACLE
DATABASE
4-1. Job Interview
Questions
NLP
Fine-Tuning
ChatBot
OpenAI API
4-2. Other
Question
Assistant
ChatBot

AI 면접


Interview Select
Interview Preparant
Question
Record
User connection
Resume Select
Mock interview
Real interview
STT
Face Scan
Interview Environment Check
OpenAI API Create Question
Mock
Real
Interview Question Select
Interview Recording

AI 면접


Interview Record upload
Google Storage Bucket
Interview Record Downloading
Spring Boot Rest API
Python FastAPI
Tasks
Message Queue
Analysis Queue Process
Video Loading
Extract Frame
Head & Eye Analysis
Extract Voice
voice Analysis
Extract STT
Clova API Emotion Analysis
Claude API Answer Analysis
Analysis Data Save
Interview Result

05

주요 소스코드

로그인 및 토큰

```java
// 로그인 성공 시 실행되는 메소드입니다. 인증된 사용자 정보를 바탕으로 JWT를 생성하고, 이를 응답 헤더에 추가합니다.
@Override
protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain ch
    // 인증 객체에서 CustomUserDetails를 추출합니다.
    MyUserDetails customUserDetails = (MyUserDetails) authentication.getPrincipal();

    // CustomUserDetails에서 사용자 정보를 추출합니다.
    String username = customUserDetails.getUsername();
    String name = customUserDetails.getName();
    String address = customUserDetails.getAddress();
    String phone = customUserDetails.getPhone();
    String birth = customUserDetails.getBirth();
    String gender = customUserDetails.getGender();
    Long id = customUserDetails.getId();

    // 사용자 이름을 사용하여 JWT를 생성합니다.
    String access  = jwtUtil.generateToken(username,"access",accessExpiredMs);
    String refresh  = jwtUtil.generateToken(username,"refresh",refreshExpiredMs);
    Optional<User> userOptional = userService.findByEmail(username);
    System.out.println(userOptional);
    if(userOptional.isPresent()){
        User user = userOptional.get();
        RefreshToken refreshToken = RefreshToken.builder()
                .status("activated")
                .userAgent(request.getHeader("User-Agent"))
                .user(user)
                .tokenValue(refresh)
                .expiresTime(refreshExpiredMs)
                .build();

        refreshService.save(refreshToken);
    } else {
        // 사용자 정보를 찾을 수 없는 경우에 대한 처리
        System.out.println("User not found: " + username);
    }

    // 응답 헤더에 JWT를 'Bearer' 토큰으로 추가합니다.
    response.addHeader("Authorization", "Bearer " + access);
    System.out.println(access);
    // HTTP에 Content-Type 설정을 추가합니다.
    response.setContentType("application/json;charset=UTF-8");
    // 클라이언트가 Authorization 헤더를 읽을 수 있도록, 해당 헤더를 노출시킵니다.
    response.setHeader("Access-Control-Expose-Headers", "Authorization");

    // 여기서 부터 사용자 정보를 응답 바디에 추가하는 코드입니다.
    // 사용자의 권한이나 추가 정보를 JSON 형태로 변환하여 응답 바디에 포함시킬 수 있습니다.
    boolean isAdmin = customUserDetails.getAuthorities().contains(new SimpleGrantedAuthority("ROLE_ADMIN"));
    Map<String, Object> responseBody = new HashMap<>();

    responseBody.put("id", id);
    responseBody.put("gender", gender);
    responseBody.put("birth", birth);
    responseBody.put("phone", phone);
    responseBody.put("address", address);
    responseBody.put("name", name);
    responseBody.put("username", username);
    responseBody.put("isAdmin", isAdmin);
    responseBody.put("refresh",refresh);
    responseBody.put("status", 200);
    // ObjectMapper를 사용하여 Map을 JSON 문자열로 변환합니다.
    String responseBodyJson = new ObjectMapper().writeValueAsString(responseBody);

    // 응답 컨텐츠 타입을 설정합니다.
    response.setContentType("application/json");

    // 응답 바디에 JSON 문자열을 작성합니다.
    response.getWriter().write(responseBodyJson);
    response.getWriter().flush();
}
```

로그인 성공 메소드

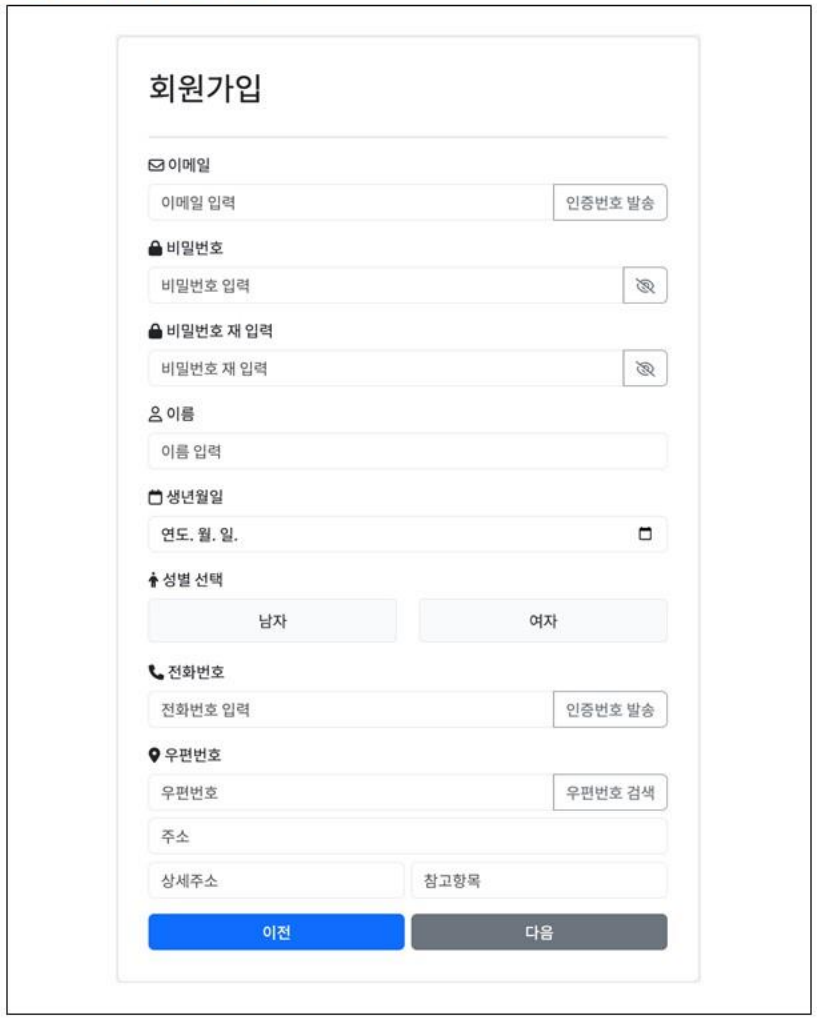
```java
// 사용자의 이메일을 기반으로 JWT를 생성합니다.
public String generateToken(String userEmail,String category, Long expiredMs) {
    // UserRepository를 사용해 사용자 정보를 조회합니다.
    Optional<User> user = userRepository.findByEmail(userEmail);

    // 사용자 정보가 없는 경우, UsernameNotFoundException을 발생시킵니다.
    if (user.isEmpty()) {
        throw new UsernameNotFoundException("User with email " + userEmail + " not found");
    }

    // 사용자의 관리자 여부를 확인합니다.
    boolean isAdmin = user.get().getIsAdmin();

    // JWT를 생성합니다. 여기서는 사용자 이메일을 주체(subject)로, 관리자 여부를 클레임으로 추가합니다.
    return Jwts.builder()
            .setSubject(userEmail)
            .claim("admin", isAdmin) // "admin" 클레임에 관리자 여부를 설정합니다.
            .claim("category",category)
            .setExpiration(new Date(System.currentTimeMillis() + expiredMs)) // 만료 시간을 설정합
            .signWith(secretKey, SignatureAlgorithm.HS256) // 비밀키와 HS256 알고리즘으로 JWT를 서명
            .compact(); // JWT 문자열을 생성합니다.
}
```

토큰 생성 메소드

```java
// 로그인 실패 시 실행되는 메소드입니다. 실패한 경우, HTTP 상태 코드 401을 반환합니다.
@Override
protected void unsuccessfulAuthentication(HttpServletRequest request, HttpServletResponse response, Authenticat

    // failed 객체로부터 최종 원인 예외를 찾습니다.
    Throwable rootCause = failed;
    while (rootCause.getCause() != null && rootCause.getCause() != rootCause) {
        rootCause = rootCause.getCause();
    }

    // rootCause를 기반으로 오류 메시지를 설정합니다.
    String message;
    if (rootCause instanceof UsernameNotFoundException) {
        message = "존재하지 않는 이메일입니다.";
    } else if (rootCause instanceof BadCredentialsException) {
        message = "잘못된 비밀번호입니다.";
    } else if (rootCause instanceof DisabledException) {
        message = "계정이 비활성화되었습니다.";
    } else if (rootCause instanceof LockedException) {
        message = "계정이 잠겨 있습니다.";
    } else {
        // 다른 예외들을 처리
        message = "인증에 실패했습니다.";
    }
    System.out.println(message);
    // 응답 데이터를 준비합니다.
    Map<String, Object> responseData = new HashMap<>();
    response.setContentType("application/json;charset=UTF-8");
    response.setCharacterEncoding("UTF-8");
    response.setStatus(HttpStatus.BAD_REQUEST.value());
    responseData.put("timestamp", LocalDateTime.now().toString());
    responseData.put("status", HttpStatus.BAD_REQUEST.value());
    responseData.put("error", "Unauthorized");
    responseData.put("message", message);
    responseData.put("path", request.getRequestURI());

    // 응답을 보냅니다.
    try {
        String jsonResponse = new ObjectMapper().writeValueAsString(responseData);
        response.getWriter().write(jsonResponse);
        response.getWriter().flush();
    } catch (IOException ignored) {
    }
}
```

로그인 실패 메소드

회원가입 화면



소셜 로그인

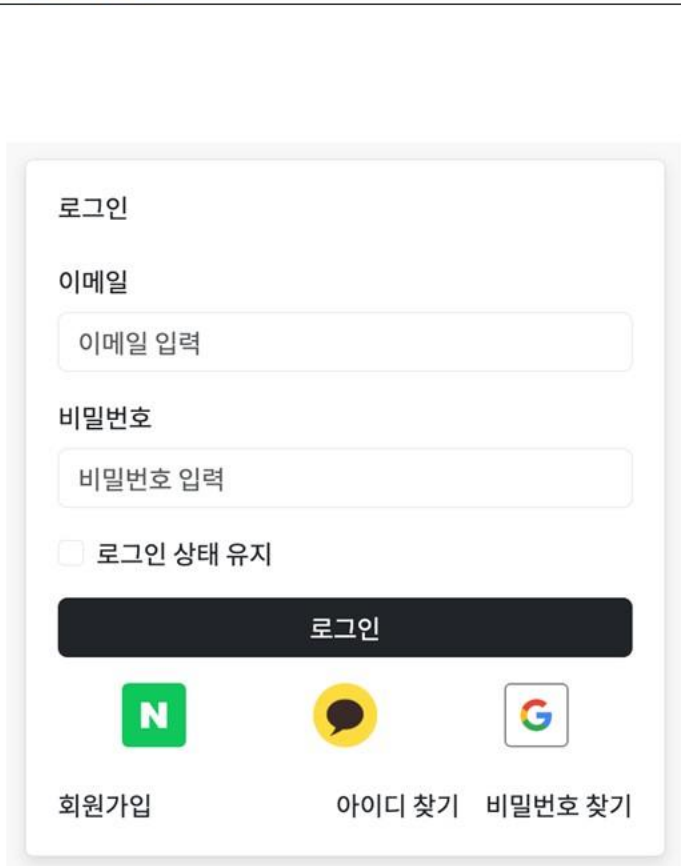
리프레시 토큰 생성 및 처리 메소드

```
@PostMapping("/reissue") // POST 요청을 '/reissue' 경로로 매핑합니다.
public ResponseEntity<?> reissue(HttpServletRequest request, HttpServletResponse response) {
    // HTTP 요청에서 'Authorization' 헤더를 통해 리프레시 토큰을 받아옵니다.
    String refresh = request.getHeader("Authorization");
    System.out.println("refresh: "+refresh);
    if (refresh == null || !refresh.startsWith("Bearer ")) { // 토큰이 없거나 Bearer 타입이 아니면 에
        return new ResponseEntity<>("refresh token null", HttpStatus.BAD_REQUEST);
    }

    String token = refresh.substring("Bearer ".length()); // 실제 토큰 값을 추출합니다.
    System.out.println("token: "+token);

    // 토큰 만료 여부 검사
    try {
        if (jwtUtil.isTokenExpired(token)) {
            // 리프레시 토큰이 만료되면 데이터베이스에서 삭제합니다.
            refreshService.deleteByRefresh(token);
            return new ResponseEntity<>("refresh token expired", HttpStatus.BAD_REQUEST);
        }
    } catch (ExpiredJwtException e) {
        // 리프레시 토큰이 만료되면 데이터베이스에서 삭제합니다.
        refreshService.deleteByRefresh(token);
        return new ResponseEntity<>("refresh token expired", HttpStatus.BAD_REQUEST);
    }

    // 리프레시 토큰이 맞는지 카테고리로 확인합니다.
    String category = jwtUtil.getCategoryFromToken(token);
    System.out.println("category: "+category);
    if (!category.equals("refresh")) {
        return new ResponseEntity<>("invalid refresh token", HttpStatus.BAD_REQUEST);
    }

    // 토큰에서 사용자 이메일을 추출합니다.
    String username = jwtUtil.getUserEmailFromToken(token);

    // 사용자 정보 조회
    Optional<User> userOptional = userService.findByEmail(username);
    if (userOptional.isEmpty()) {
        return new ResponseEntity<>("user not found", HttpStatus.NOT_FOUND);
    }

    // 리프레시 토큰이 유효한지 확인합니다.
    Optional<RefreshToken> refreshTokenOptional = refreshService.findByTokenValue(token);
    if (refreshTokenOptional.isEmpty() || !refreshTokenOptional.get().getUser().equals(userOpt
        return new ResponseEntity<>("refresh token not found or does not match", HttpStatus.BA
    }

    // 리프레시 토큰의 상태 검증
    RefreshToken refreshToken = refreshTokenOptional.get();
    if (!refreshToken.getStatus().equals("activated")) {
        return new ResponseEntity<>("refresh token invalid or expired", HttpStatus.BAD_REQUEST
    }

    // 새로운 액세스 토큰 생성
    // 액세스 토큰의 유효 시간 (밀리초 단위)
    Long accessExpiredMs = 86400000L;
    String access = jwtUtil.generateToken(username, "access", accessExpiredMs);

    // 응답에 새로운 액세스 토큰 추가
    response.addHeader("Authorization", "Bearer " + access);

    // 클라이언트가 Authorization 헤더를 읽을 수 있도록 설정
    response.setHeader("Access-Control-Expose-Headers", "Authorization");

    // 성공적으로 새 토큰을 발급받았을 때의 응답
    return new ResponseEntity<>(HttpStatus.OK);
}
```

소셜 로그인 메소드(예시: Google)

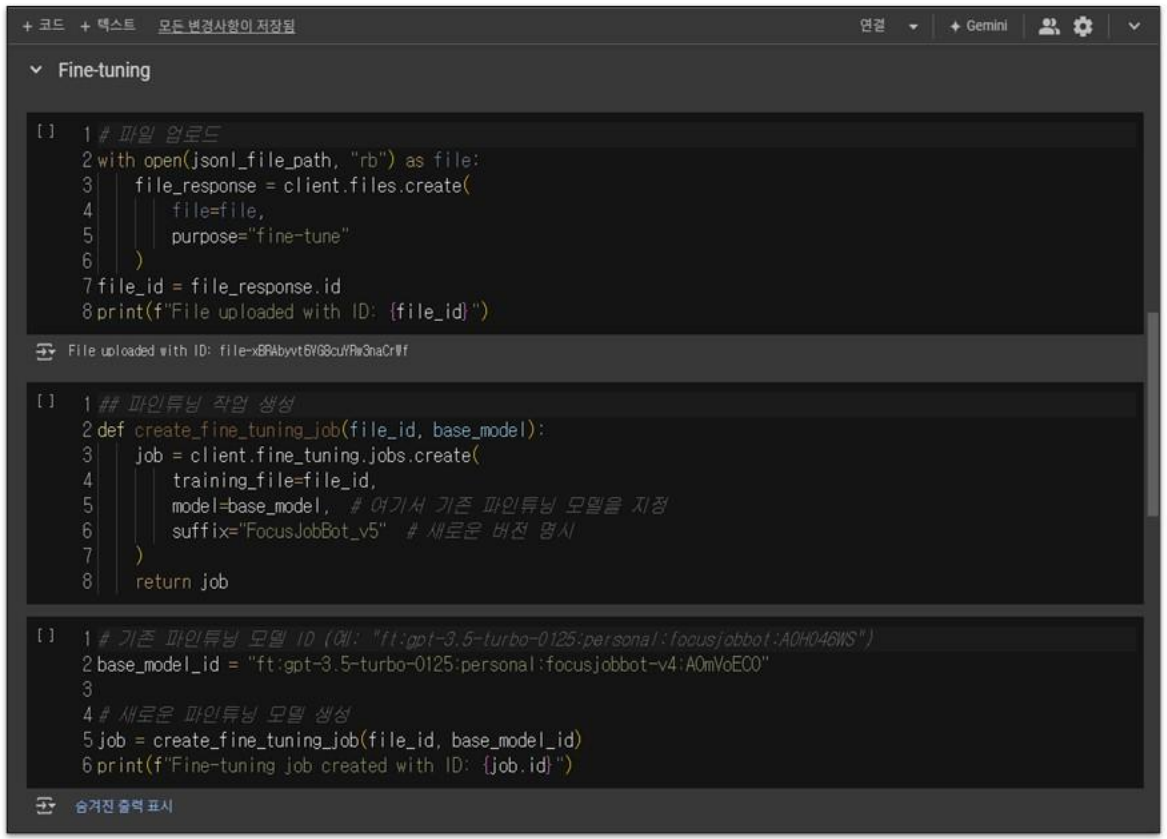
```
@GetMapping("/google/callback")
public void googleLogin(@RequestParam String code, HttpServletResponse response) throws IOException, JSONException {
    log.info("code = {}", code);

    // 액세스 토큰을 요청하기 위한 URL 및 헤더 설정
    String tokenUrl = "https://oauth2.googleapis.com/token";
    HttpHeaders tokenHeaders = new HttpHeaders();
    tokenHeaders.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
    String tokenRequestBody = "grant_type=authorization_code"
            + "&client_id=" + googleClientId
            + "&client_secret=" + googleClientSecret
            + "&redirect_uri=" + googleRedirectUri
            + "&code=" + code;

    // 토큰 요청
    HttpEntity<String> tokenRequestEntity = new HttpEntity<>(tokenRequestBody, tokenHeaders);
    RestTemplate restTemplate = new RestTemplate();
    ResponseEntity<String> tokenResponse = restTemplate.exchange(tokenUrl, HttpMethod.POST, tokenRequestEntity, String.
    log.info("token response = {}", tokenResponse.getBody());

    JSONObject tokenJson = new JSONObject(tokenResponse.getBody());
    String accessToken = tokenJson.getString("access_token");
    log.info("accessToken = {}", accessToken);

    // 사용자 정보를 요청하기 위한 URL 및 헤더 설정
    String userInfoUrl = "https://www.googleapis.com/userinfo/v2/me";
    HttpHeaders userInfoHeaders = new HttpHeaders();
    userInfoHeaders.set("Authorization", "Bearer " + accessToken);

    // 사용자 정보 요청
    HttpEntity<String> userInfoRequestEntity = new HttpEntity<>(userInfoHeaders);
    ResponseEntity<String> userInfoResponse = restTemplate.exchange(userInfoUrl, HttpMethod.GET, userInfoRequestEntity,
    log.info("user info response = {}", userInfoResponse.getBody());

    JSONObject userJson = new JSONObject(userInfoResponse.getBody());

    //Email 처리
    String email = userJson.has("email") ?
            userJson.getString("email") : "이메일 정보가 없습니다.";

    //username 처리
    String name = userJson.has("name") ?
            userJson.getString("name") : "이름 정보가 없습니다.";

    // Profile Image 처리
    String profileImageUrl = userJson.has("picture")?
            userJson.getString("picture") : "프로필 정보가 없습니다.";
    byte[] profileImage = null;
    try (InputStream in = new URL(profileImageUrl).openStream()) {
        profileImage = in.readAllBytes();
    } catch (IOException e) {
        log.error("Failed to download profile image", e);
    }

    Optional<User> optionalUser = userRepository.findByEmail(email);

    if (optionalUser.isPresent()) {
        User user = optionalUser.get();
        user.setLastLogin(LocalDateTime.now());
        user.setSnsAccessToken(accessToken);
        user.setIsNaver(0);
        user.setIsGoogle(1);
        user.setIsKakao(0);
        user.setProfileImage(profileImage);
        userRepository.save(user);

        // JWT 토큰 발급
        Long accessExpiredMs = 86400000L;
        String accessTokenJwt = jwtUtil.generateToken(email, "access", accessExpiredMs);
        Long refreshExpiredMs = 86400000L;
        String refreshTokenJwt = jwtUtil.generateToken(email, "refresh", refreshExpiredMs);

        RefreshToken refreshToken = RefreshToken.builder()
                .status("activated")
                .userAgent(response.getHeader("User-Agent"))
                .user(user)
                .tokenValue(refreshTokenJwt)
                .expiresTime(refreshExpiredMs)
                .build();

        refreshService.save(refreshToken);

        // 로그인 성공 후 URL에 토큰 정보 포함
        Optional<User> optinalUserForId = userRepository.findByEmailAndIsGoogle(email, 1);
        Long id = optinalUserForId.get().getId();
        String encodedUsername = URLEncoder.encode(name, StandardCharsets.UTF_8.toString());

        String redirectUrl = String.format("http://localhost:3000/auth/successSocialLogin?access=%s&refresh=%s&isAdmin=
                accessTokenJwt, refreshTokenJwt, optinalUserForId.get().getIsAdmin(), id, encodedUsername, email, profil

        response.sendRedirect(redirectUrl);
        log.info("로그인 성공: {}", email);
```

로그인 화면

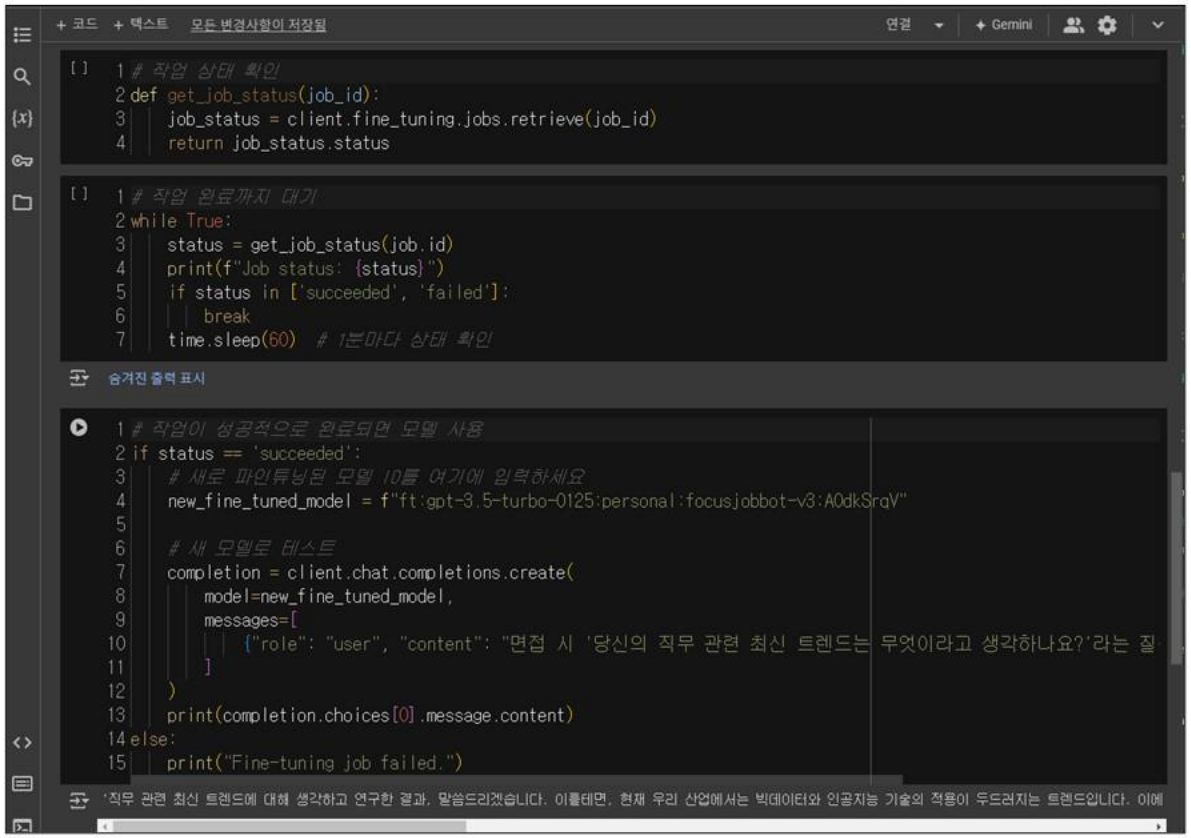


챗봇 Fine-tuning

OpenAI Fine-tuning 코드



```
Fine-tuning

[ ]
# 파일 업로드
with open(jsonl_file_path, "rb") as file:
    file_response = client.files.create(
        file=file,
        purpose="fine-tune"
    )
file_id = file_response.id
print(f"File uploaded with ID: {file_id}")

File uploaded with ID: file-xBRAbyvt6YG8cuYRw3naCrWf

[ ]
## 파인튜닝 작업 생성
def create_fine_tuning_job(file_id, base_model):
    job = client.fine_tuning.jobs.create(
        training_file=file_id,
        model=base_model,  # 여기서 기존 파인튜닝 모델을 지정
        suffix="FocusJobBot_v5"  # 새로운 버전 명시
    )
    return job

[ ]
# 기존 파인튜닝 모델 ID (예: "ft:gpt-3.5-turbo-0125:personal:focusjobbot:AOH046WS")
base_model_id = "ft:gpt-3.5-turbo-0125:personal:focusjobbot-v4:AOmVoECO"

# 새로운 파인튜닝 모델 생성
job = create_fine_tuning_job(file_id, base_model_id)
print(f"Fine-tuning job created with ID: {job.id}")

숨겨진 출력 표시
```



```
[ ]
# 작업 상태 확인
def get_job_status(job_id):
    job_status = client.fine_tuning.jobs.retrieve(job_id)
    return job_status.status

[ ]
# 작업 완료까지 대기
while True:
    status = get_job_status(job.id)
    print(f"Job status: {status}")
    if status in ['succeeded', 'failed']:
        break
    time.sleep(60)  # 1분마다 상태 확인

숨겨진 출력 표시

# 작업이 성공적으로 완료되면 모델 사용
if status == 'succeeded':
    # 새로 파인튜닝된 모델 ID를 여기에 입력하세요
    new_fine_tuned_model = f"ft:gpt-3.5-turbo-0125:personal:focusjobbot-v3:AOdkSrqV"

    # 새 모델로 테스트
    completion = client.chat.completions.create(
        model=new_fine_tuned_model,
        messages=[
            {"role": "user", "content": "면접 시 '당신의 직무 관련 최신 트렌드는 무엇이라고 생각하나요?'라는 질
        ]
    )
    print(completion.choices[0].message.content)
else:
    print("Fine-tuning job failed.")

'직무 관련 최신 트렌드에 대해 생각하고 연구한 결과, 말씀드리겠습니다. 이를테면, 현재 우리 산업에서는 빅데이터와 인공지능 기술의 적용이 두드러지는 트렌드입니다. 이에
```

챗봇 STT-TTS, 뮤직 플레이어

이모티콘, STT-TTS 코드

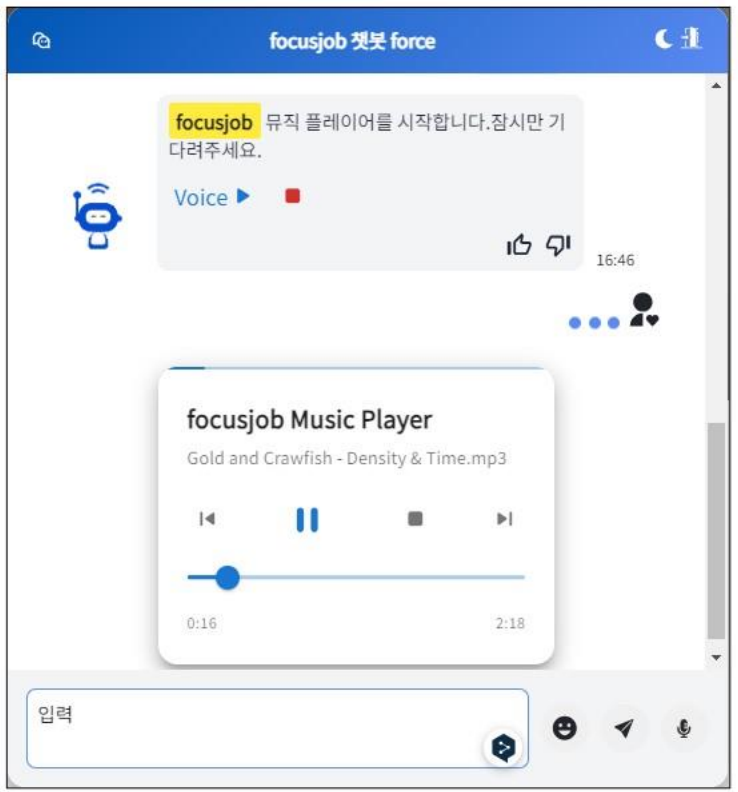
```
<div className={styles.inputArea}>
  <textarea
    value={inputMessage}
    onChange={handleInputChange}
    onKeyPress={handleKeyPress}
    placeholder="궁금한 사항이 있으면 여기에 입력해주세요.."
  />
  <div className={styles.buttonGroup}>
    <button onClick={toggleEmojiPicker}>
      <EmojiEmotionsIcon />
    </button>
    <button onClick={handleSendMessage}>
      <BsSendFill />
    </button>
    <button onClick={toggleListening}>
      {isListening ? <FaMicrophoneSlash /> : <FaMicrophoneLines />}
    </button>
  </div>
  {showEmojiPicker && (
    <div ref={emojiPickerRef}>
      <EmojiPicker onEmojiClick={onEmojiClick} />
    </div>
  )}
</div>
```

```
<Card>
  <LinearProgress ... /> {/* 재생 진행 상태 */}
  <CardContent>
    {/* ... */}
    <Box> {/* 컨트롤 버튼 */}
      <IconButton onClick={handlePrevious}>
        <SkipPreviousRoundedIcon />
      </IconButton>
      <IconButton onClick={isPlaying ? handlePause : handlePlay}>
        {isPlaying ? <PauseRoundedIcon /> : <PlayArrowRoundedIcon />}
      </IconButton>
      <IconButton onClick={handleStop}>
        <StopRoundedIcon />
      </IconButton>
      <IconButton onClick={handleNext}>
        <SkipNextRoundedIcon />
      </IconButton>
    </Box>
    <Slider ... /> {/* 재생 위치 조절 */}
    {/* ... */}
  </CardContent>
</Card>
```

뮤직 플레이어 코드

```
useEffect(() => {
  // ... 텍스트 변경 시 SpeechSynthesisUtterance 객체 생성 및 이벤트 처리
}, [text]);

const handlePlay = () => {
  // ... 음성 재생 또는 재개
};

const handlePause = () => {
  // ... 음성 일시 정지
};

const handleStop = () => {
  // ... 음성 중지 및 초기화
};

return (
  <Box>
    <Tooltip title={isPlaying ? "Pause" : "Speak"}>
      <IconButton onClick={isPlaying ? handlePause : handlePlay}>
        {/* ... 아이콘 표시 */}
      </IconButton>
    </Tooltip>
    <Tooltip title="Stop">
      <IconButton onClick={handleStop}>
        {/* ... 아이콘 표시 */}
      </IconButton>
    </Tooltip>
  </Box>
);
```

챗봇 화면



MQTT 실시간 채팅

```
useEffect(() => {
    if (!client) {
        const mqttClient = mqttStore.mqttClient;
        mqttClient.on('connect', () => {
            console.log('Connected to MQTT broker');
            setIsConnected(true);
        });
```

React에서 채팅방 토픽 **Subscribe**

```
const sendMessage = async () => {
    const userIdsForAlarm = await getUserIdsForChatAlarm();

    if (inputMessage.trim()) {
        const messageData = {
            text: inputMessage,
            chatroomId: currentChatRoomIdRef.current,
            sender: userStore.username,
            senderId: userStore.id.toString(),
            timestamp: new Date().toISOString(),
            type: 'chat',
            userIds: userIdsForAlarm
        };
        try {
            await axios.post('http://192.168.0.137:8000/sendMessage', messageData);
            setInputMessage('');
        } catch (error) {
            console.error("메시지 전송 중 오류 발생:", error);
        }
    }
};
```

채팅방에 입력한 데이터 FastAPI 서버로 전송

```
@app.post("/sendMessage")
async def send_message(message: ChatMessage):
    message_id = db.getMaxMessageId(conn, table_name: 'chat_messages') + 1
    alarm_id = db.getMaxMessageId(conn, table_name: 'alarm') + 1

    # MQTT로 메시지 publish
    mqtt_topic = f"mqtt/chat/{message.chatroomId}"
    mqtt_message = json.dumps({**message.dict(), 'messageId': message_id, 'alarmId': alarm_id})

    # publish
    mqtt_client.publish(mqtt_topic, mqtt_message)
```

```
# DB에 메시지 저장
isinserted = db.saveMessage(conn, dict: {
    'id': message_id,
    'topic': mqtt_topic,
    'text': message.text,
    'senderId': message.senderId
})
```

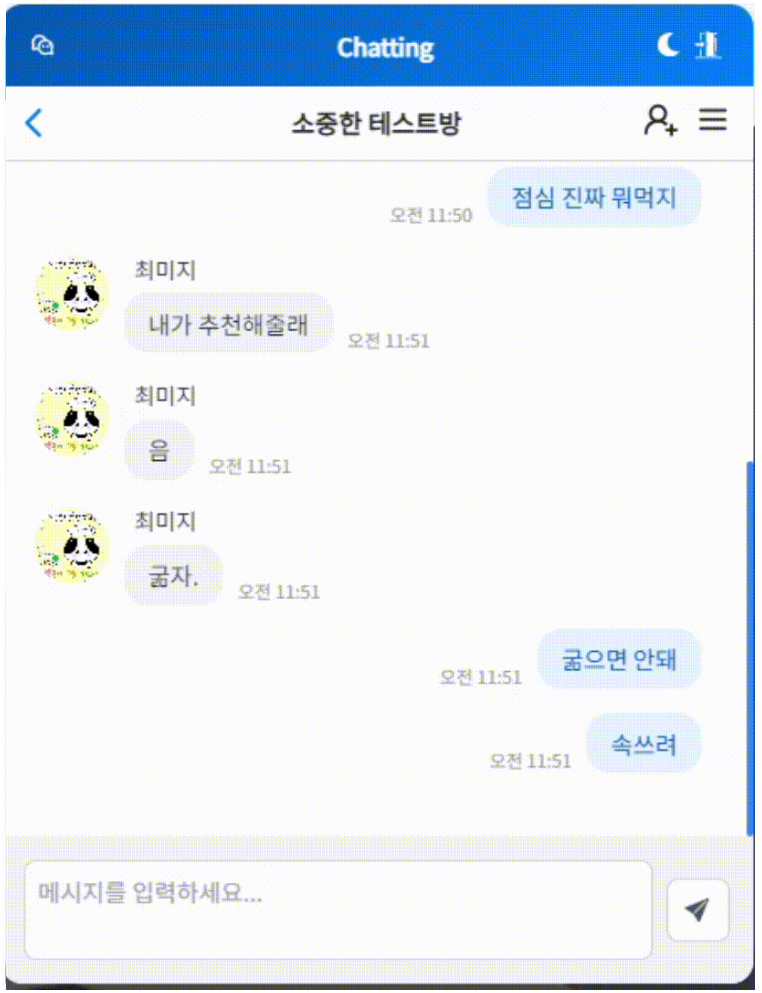
```
return {"status": "Message sent", "db_inserted": isinserted}
```

FastAPI 서버로 받은 데이터 MQTT로 **Publish** & DB에 데이터 저장

MQTT 실시간 채팅

```
mqttClient.on('message', (topic, message) => {
    if (topic.startsWith('mqtt/chat/')) {
        console.log('Received message:', message.toString());
        const receivedMessage = JSON.parse(message);

        const lastM = `${receivedMessage.sender} : ${receivedMessage.text}`;
        const lastMTopic = topic.split('/').pop();

        setChatRoomList(prevChatRoomList =>
            prevChatRoomList.map(room =>
                room.id == lastMTopic ? { ...room, lastMessage: lastM, updatedTime: new Date().toISOString() } :
            )
        );
        if (receivedMessage && receivedMessage.chatroomId == currentChatRoomIdRef.current) {
            setMessages((prevMessages) => [...prevMessages, receivedMessage]);
        }
    }
    if (topic.startsWith('mqtt/member/')) {
        console.log('topic.startsWith(mqtt/member/');
        const roomId = JSON.parse(message).chatroomId;
        if(roomId == currentChatRoomIdRef.current){
            readChatAlarmInChatroom(roomId);
        }
        getChatAlarm();
    }
});
```

Subscribe한 토픽으로 Publish된 메시지 수신



실시간 채팅 화면

MQTT 실시간 채팅

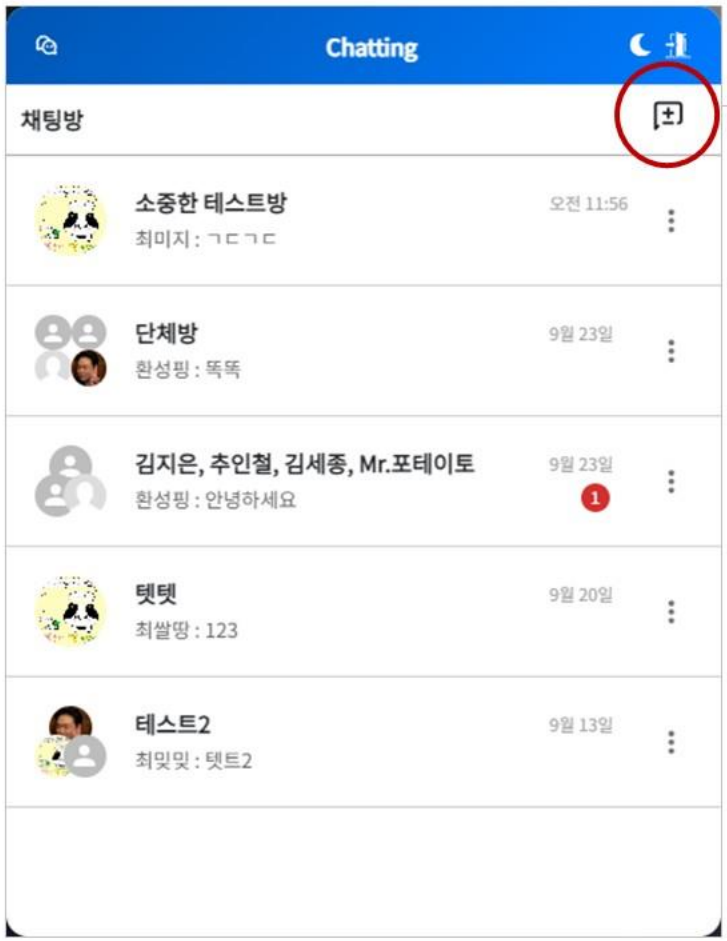
```
const getChatroomList = async () => {
    try {
        const userId = userStore.id;
        const response = await axios.post('http://localhost:8080/api/chat/userChatrooms',
            userId,
            { headers: { 'Content-Type': 'application/json' } }
        );
        setChatRoomList(response.data);
    } catch (error) {
        console.error('Error fetching chat rooms:', error);
    }
};
```

로그인한 사용자가 참여 중인 채팅방 목록 출력

```
const exitChatroom = async () => {
    try {
        await axios.delete('http://localhost:8080/api/chat/exitChatroom', {
            params: {
                currentChatRoomId: currentChatRoomId,
                userId: userStore.id
            }
        });
        getChatroomList();

        client.unsubscribe(currentChatRoomId);
        console.log(`Unsubscribed from topic: ${currentChatRoomId}`);

    } catch (error) {
        console.error('채팅방 나가기 중 에러 발생:', error);
    }
};
```

채팅방 나가기 _ 구독한 채팅방 토픽 Unsubscribe

채팅방 목록



채팅방 생성
& 대화상대 초대

채팅 & 게시글 알림

```
mqttStore.setMqttClient(mqttClient);

mqttClient.on('connect', () => {
  console.log('Connected to MQTT broker');
  mqttClient.subscribe(`mqtt/member/${userStore.id}`);
});
```

로그인 여부 판단 후 로그인한 유저의 토픽 구독 (header.jsx)

```
for user_id in message.userIds:
    mqtt_alarm_topic = f"mqtt/member/{user_id}"  # 각 user_id에 대해 토픽 생성
    mqtt_client.publish(mqtt_alarm_topic, mqtt_message)  # 해당 토픽으로 메시지 발행

    alarm_isinserted = db.saveAlarm(conn, dict_: {
        'id': db.getMaxMessageId(conn, table_name: 'alarm') + 1,
        'sender': message.sender,
        'senderId': message.senderId,
        'chatroomId': message.chatroomId,
        'text': message.text,
        'title': 'id값으로 제목 갖고오기',
        'type': message.type,
        'contentId': message_id,
        'receiverId': user_id
    })
```

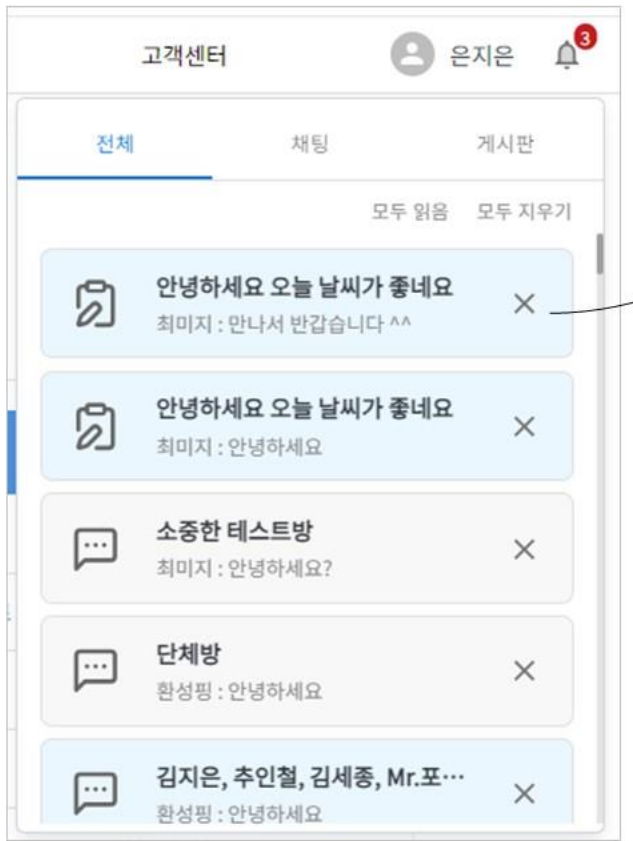
채팅 메시지를 받는 유저들의 토픽으로 알림 Publish

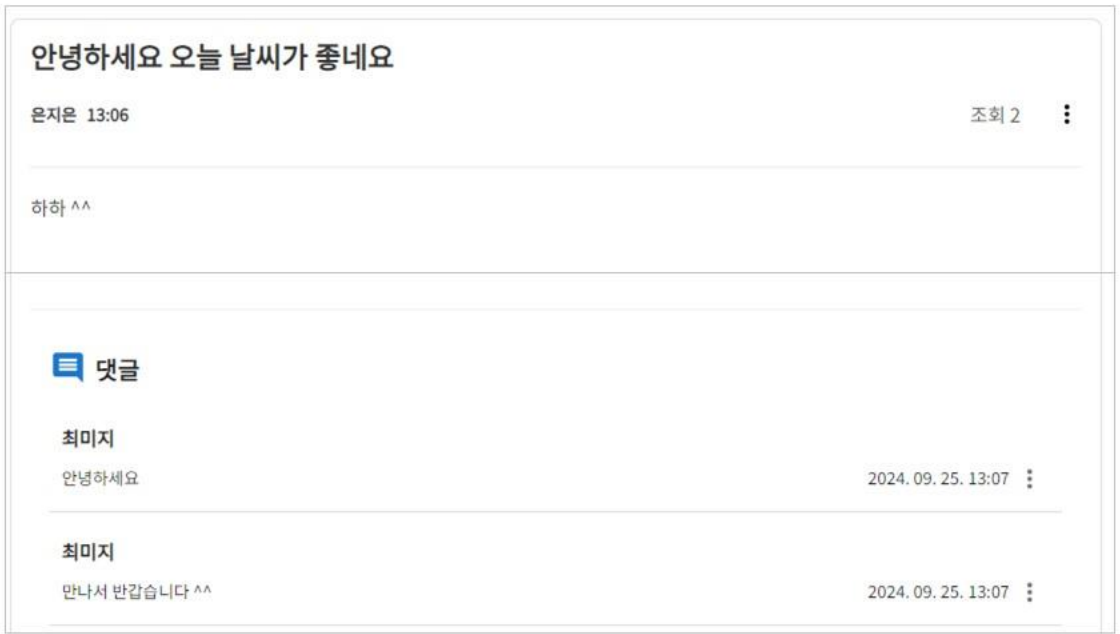
```
@app.post("/sendComment")
async def send_comment(message: CommentMessage):

    print('sendCommnet에서 message 출력: ', message)

    alarm_id = db.getMaxMessageId(conn, table_name: 'alarm') + 1

    # MQTT로 메시지 publish
    mqtt_topic = f"mqtt/member/{message.receiverId}"
    # mqtt_message = json.dumps(message.dict())
    mqtt_message = json.dumps({**message.dict(), 'alarmId': alarm_id})

    print('sendCommnet에서 mqtt_message 출력: ', mqtt_message)

    # publish
    mqtt_client.publish(mqtt_topic, mqtt_message)

    if(message.type == 'bbs'):
        alarm_isinserted = db.saveCommentAlarm(conn, dict_: {
            'id': db.getMaxMessageId(conn, table_name: 'alarm') + 1,
            'bbsId': message.bbsId,
            'title': message.bbsTitle,
            'text': message.text,
            'sender': message.sender,
            'senderId': message.senderId,
            'receiverId': message.receiverId,
            'contentId': message.contentId,
            'type': message.type
        })

    return {"status": "Comment Alarm sent"}
```

게시글 댓글 알림을 게시글 작성자토픽으로 Publish

채팅 & 게시글 알림

헤더의 알림창





알림 클릭 시 해당 게시글로 이동

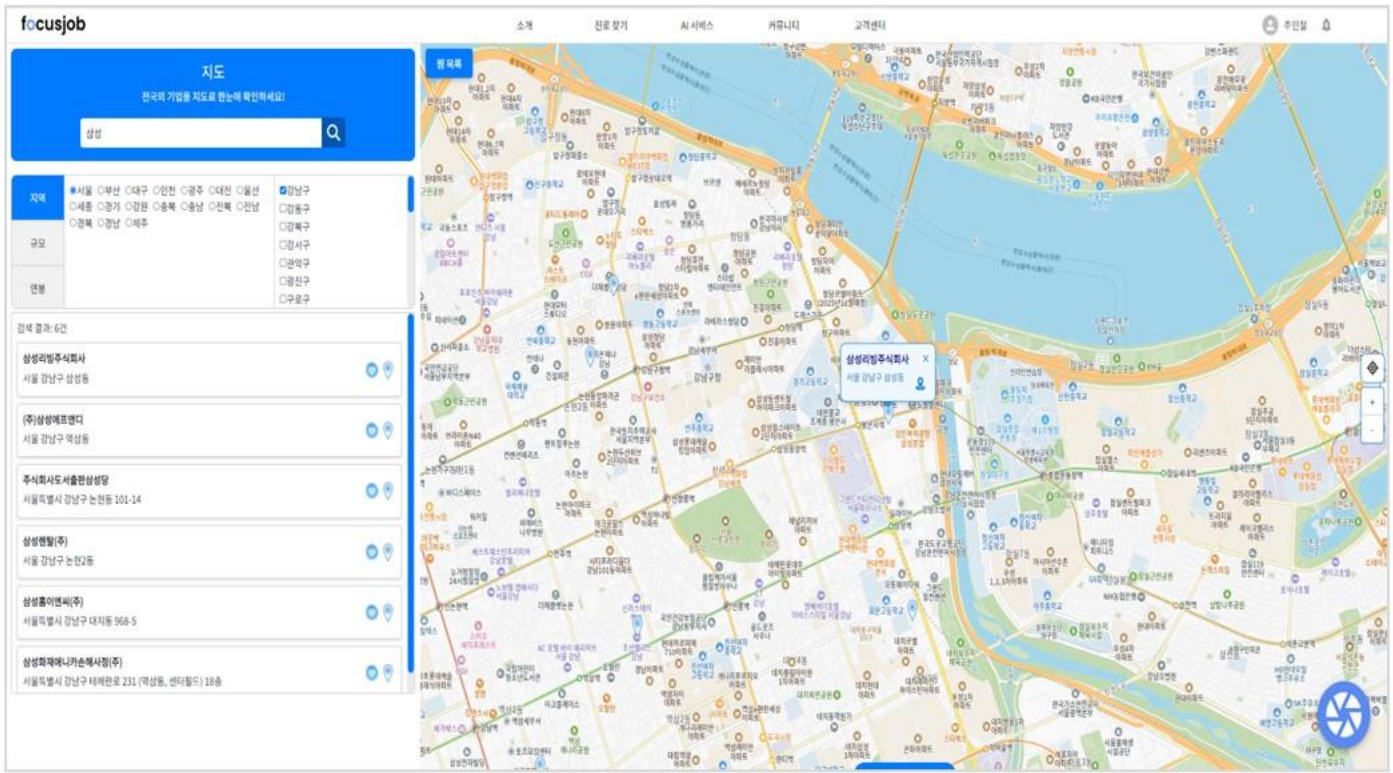
회사 검색 (1)

공공API 데이터 요청 및 필터링

```
export const fetchCorpInfo = async (params) => {
    try {
        // 요청 URL을 로그로 확인
        const requestUrl = `http://localhost:8080/api/search/corpInfo?${params.toString()}`;
        console.log('API 요청 URL:', requestUrl);

        // API 요청을 보내기 전 파라미터 확인
        console.log('요청 파라미터:', params);

        // API 요청
        const response = await fetch(requestUrl);
        console.log('API 요청 응답 상태:', response.status);

        // 응답이 정상인지 확인
        if (!response.ok) {
            throw new Error(`Network response was not ok: ${response.statusText}`);
        }

        // 응답 데이터를 JSON으로 변환
        const data = await response.json();
        console.log('API 응답 데이터:', data);
```

```
const Filter = ({ onFilterChange, handleSearch, setSearchInputFocus, searchInputRef, setCorpNm , setSearchTriggered}) => {
    const [selectedCategory, setSelectedCategory] = useState('지역');
    const [selectedRegion, setSelectedRegion] = useState('서울');
    const [selectedDistricts, setSelectedDistricts] = useState([]);
    const [selectedSizes, setSelectedSizes] = useState([]);
    const [selectedSalaries, setSelectedSalaries] = useState([]);

    useEffect(() => {

        handleFilterChange();
    }, [selectedRegion, selectedDistricts, selectedSizes, selectedSalaries]);

    const handleFilterChange = () => {
        const filterData = {
            selectedRegion,
            selectedDistricts,
            selectedSizes,
            selectedSalaries,
        };
        onFilterChange(filterData);
    };

    const handleCheckboxChange = (setter, value) => {
        setter(prev => prev.includes(value) ? prev.filter(v => v !== value) : [...prev, value]);
        // setSearchInputFocus();
    };

    const handleDistrictCheckboxChange = (e) => {
        const { value, checked } = e.target;
        if (checked && selectedDdistricts.length >= 30) {
            return;
        }
        handleCheckboxChange(setSelectedDistricts, value);
    };

    const handleKeyPress = (e) => {
        if (e.key === 'Enter') {
            e.preventDefault();
            handleSearch();
        }
    };
```

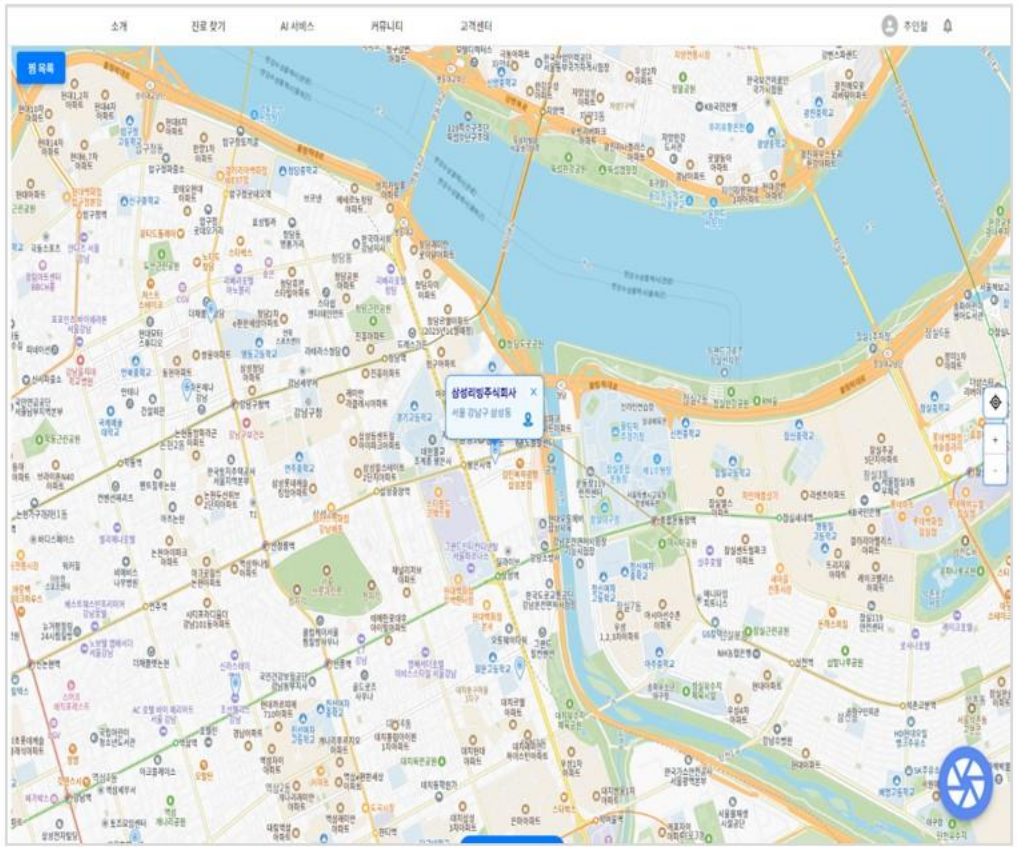
회사 검색 화면



기업리스트 필터 조건 설정

회사 검색 (2)

회사 검색 화면



카카오API 응답 처리 및 에러 핸들링

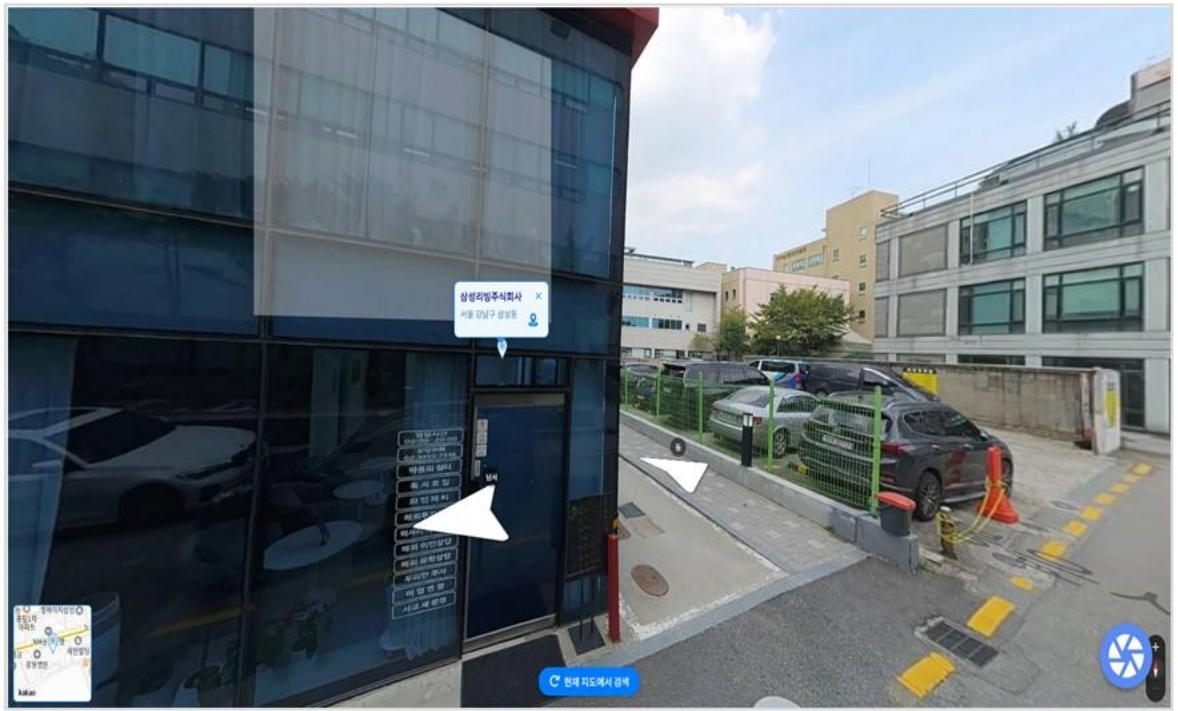
```
import React, { useEffect, useRef, useState } from 'react';
import currentLocationStyles from '@/styles/search/currentLocationBtn.module.css'; // CSS Modules 방식으로 import
import roadviewStyles from '@/styles/search/roadviewAndOverlay.module.css'; // CSS Modules 방식으로 import

import CurrentLocationButton from './currentLocationButton';
import ZoomControl from "./zoomControl"; // zoomControl 컴포넌트 import

const UserLocationMap = ({ companies = [], setMapBounds, searchTriggered, setFilteredCompanies, setLoading, selectedCompany }) => {
    const mapRef = useRef(null);
    const clustererRef = useRef(null);
    const markersRef = useRef([]);
    const customOverlaysRef = useRef([]);
    const currentOverlayRef = useRef(null);
    const currentMarkerRef = useRef(null);
    const roadviewCloseBtnRef = useRef(null);
    const [isRoadviewVisible, setIsRoadviewVisible] = useState(false);
    const [isMapInitialized, setIsMapInitialized] = useState(false);

    useEffect(() => {
        // 외부가 작업을 수행하는 함수 정의
        const loadKakaoMapScript = async () => {
            try {
                const existingScript = document.querySelector('script[src*="dapi.kakao.com"]');
                if (existingScript) {
                    console.log('Kakao Map script already loaded');
                    initializeUserLocation();
                    return;
                }

                const kakaoApiKey = process.env.REACT_APP_KAKAO_API_KEY; // 환경 변수에서 API 키를 가져옴
                const script = document.createElement('script');
                script.src = `//dapi.kakao.com/v2/maps/sdk.js?appkey=${kakaoApiKey}&libraries=services,clusterer`;
                script.async = true;

                script.onload = () => {
                    console.log('Kakao Map script loaded');
                    initializeUserLocation();
                };

                document.head.appendChild(script);

                return () => {
                    document.head.removeChild(script);
                };
```

```
    useEffect(() => {
        if (mapRef.current) {
            console.log('Companies state changed:', companies);
            addMarkers();
        }
    }, [companies]);

    const initializeMap = () => {
        const mapContainer = document.getElementById('map');
        const mapOption = {
            center: new window.kakao.maps.LatLng(37.5665, 126.9780),
            level: 3
        };

        const map = new window.kakao.maps.Map(mapContainer, mapOption);
        mapRef.current = map;

        clustererRef.current = new window.kakao.maps.MarkerClusterer({
            map: map,
            averageCenter: true,
            minLevel: 5,
            gridSize: 60 // 클러스터의 격자 크기 설정
        });

        console.log('Map initialized');
    };

    const addMarkers = () => {
        const geocoder = new window.kakao.maps.services.Geocoder();
        const markers = [];

        companies.forEach((company) => {
            const companyName = company.corpNm;
            geocoder.addressSearch(companyName, function (result, status) {
                if (status === window.kakao.maps.services.Status.OK) {
                    const coords = new window.kakao.maps.LatLng(result[0].y, result[0].x);
                    const marker = new window.kakao.maps.Marker({
                        position: coords
                    });

                    const infowindow = new window.kakao.maps.InfoWindow({
                        content: `<div style="padding:5px;z-index:1;">${companyName}</div>`
                    });

                    infowindow.open(mapRef.current, marker);

                    markers.push(marker);

                    // 클러스터러에 마커 추가
                    if (markers.length === companies.length) {
                        clustererRef.current.addMarkers(markers);
                        console.log('Clusterer markers added:', markers);
                    }
                } else {
                    console.log('Geocoder failed for:', companyName, 'with status:', status);
                }
```

지도 마커 및 클러스터링 로직

회사 검색 (3)

회사 검색 로드뷰 기능



로드뷰를 통한 회사 위치 시각화

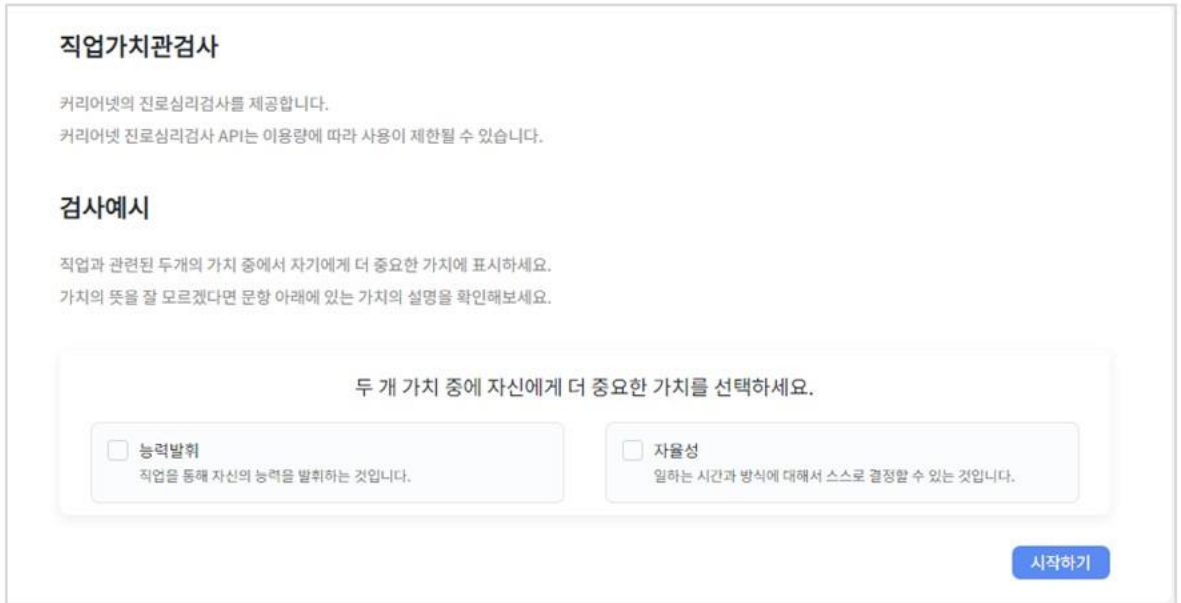
```
            const mapOption = {
                center: coords,
                level: 4
            };
            const map = new window.kakao.maps.Map(mapContainer, mapOption);

            const mapMarkerImage = new window.kakao.maps.MarkerImage(
                `/images/marker-icon(2).png`,
                new window.kakao.maps.Size(24, 35)
            );

            const mapMarker = new window.kakao.maps.Marker({
                position: coords,
                map: map,
                image: mapMarkerImage
            });

            // 닫기 버튼이 이미 존재하면 display를 block으로 설정하고, 위치와 스타일을 다시 설정합니다.
            if (roadviewCloseBtnRef.current) {
                roadviewCloseBtnRef.current.style.display = 'block';
            } else {
                // 로드뷰 닫기 버튼이 없을 경우 추가
                const closeBtn = document.createElement('div');
                closeBtn.id = 'roadview-close-btn';
                closeBtn.className = roadviewStyles['roadview-close-btn'];
                closeBtn.innerHTML = '&times;';
                closeBtn.onclick = () => {
                    console.log('로드뷰 닫기 버튼 클릭됨');
                    roadviewContainer.style.display = 'none';
                    setIsRoadviewVisible(false);
                };
                roadviewContainer.appendChild(closeBtn);
                roadviewCloseBtnRef.current = closeBtn;
            }
        } else {
            alert('해당 위치의 로드뷰를 찾을 수 없습니다.');
            console.log('로드뷰를 찾을 수 없음');

            // 로드뷰를 닫고 원래 지도 화면으로 돌아가기
            roadviewContainer.style.display = 'none';
            setIsRoadviewVisible(false);
        }
    });
};
if (typeof window !== 'undefined') {
    window.closeOverlay = function() {
        if (currentOverlayRef.current) {
```

로드뷰와 마커 오버레이 기능 구현

```
const openRoadview = (company, coords) => {
    console.log('로드뷰 아이콘 클릭됨');
    const roadviewContainer = document.getElementById('roadview');
    roadviewContainer.style.display = 'block';
    setIsRoadviewVisible(true); // 로드뷰 표시 상태 변경

    // 닫기 버튼 초기화
    if (roadviewCloseBtnRef.current) {
        roadviewCloseBtnRef.current.style.display = 'none';
        roadviewCloseBtnRef.current = null;
    }

    const roadview = new window.kakao.maps.Roadview(roadviewContainer);
    const roadviewClient = new window.kakao.maps.RoadviewClient();

    roadviewClient.getNearestPanoId(coords, 50, (panoId) => {
        if (panoId !== null) {
            console.log('로드뷰 파노라마 ID:', panoId);
            roadview.setPanoId(panoId, coords);

            // 로드뷰에 마커 추가
            const markerImage = new window.kakao.maps.MarkerImage(
                `/images/marker-icon.png`,
                new window.kakao.maps.Size(24, 35)
            );

            const rvMarker = new window.kakao.maps.Marker({
                position: coords,
                map: roadview,
                image: markerImage
            });

            // 로드뷰 마커에 커스텀 오버레이 추가
            const rvOverlay = createCustomOverlay(company, coords);
            rvOverlay.setMap(roadview);

            // 작은 지도 추가
            const mapContainer = document.createElement('div');
            mapContainer.id = 'roadview-map';
            mapContainer.style.position = 'absolute';
            mapContainer.style.bottom = '10px';
            mapContainer.style.left = '10px';
            mapContainer.style.width = '150px';
            mapContainer.style.height = '150px';
            mapContainer.style.zIndex = '9999';
            mapContainer.style.border = '2px solid #1e90ff';
            mapContainer.style.borderRadius = '8px';
            mapContainer.style.boxShadow = '0 2px 5px rgba(0, 0, 0, 0.1)  ';
            roadviewContainer.appendChild(mapContainer);
```

적성 탐색(1)

적성탐색 시작화면

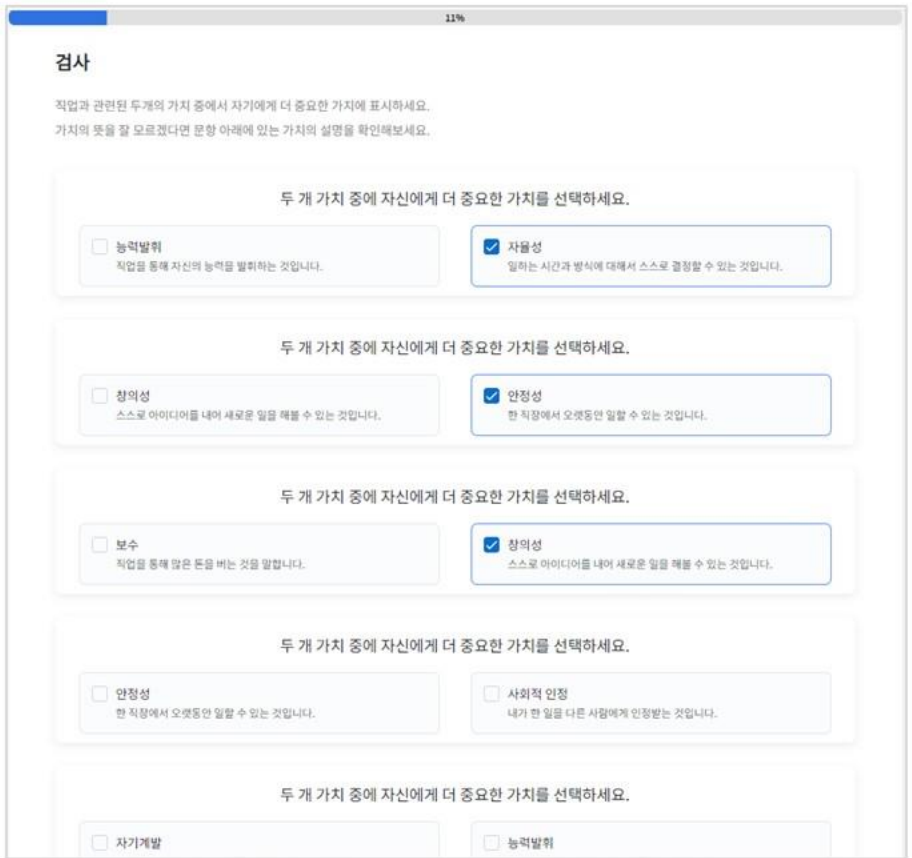


커리어넷 API 연동

```
import axios from 'axios';

export const Q_NUM = '6';

export const Q_URL =
'https://www.career.go.kr/inspct/openapi/test';  // question
export const R_URL =
'https://inspct.career.go.kr/inspct/openapi/psycho';  // result

export const GetQuestionAPI = async () => {
    const response = await
    axios.get(
        `${Q_URL}/questions?apikey=${API_KEY}&q=${Q_NUM}`
    );
    return response.data.RESULT;
};

export const PostResultAPI = async (data) => {
    const pushResponse = await
    axios.post(
        `http://www.career.go.kr/inspct/openapi/test/report`,
        data
    );
    return pushResponse;
};
```

적성 탐색(2)

적성탐색 답변선택 화면



질문별 답변 선택 관리

```
const handleAnswerSelect = (index, answer) => {
  const answerValue = questions[index].answer01 === answer ? 1 : 2;

  setSelectedAnswers((prevAnswers) => ({
    ...prevAnswers,
    [index]: answerValue,  // 1 또는 2를 그대로 저장
  }));
};

const handleNextPage = () => {
  const startIndex = currentPage * QUESTIONS_PER_PAGE;
  const endIndex = startIndex + QUESTIONS_PER_PAGE;

  const allAnswered = questions.slice(startIndex, endIndex).every(
    (_, index) => selectedAnswers[startIndex + index] !== undefined
  );

  if (!allAnswered) {
    return toast('모든 질문에 답변해주세요.', {
      className: 'custom-toast',
      draggable: true,
      position: 'bottom-center',
    });
  }

  if (currentPage < Math.ceil(questions.length / QUESTIONS_PER_PAGE) - 1) {
    const nextPage = currentPage + 1;
    setCurrentPage(nextPage);
    router.push(`/survey/play/${nextPage + 1}`);
  } else {
    submitSurvey();
  }
};
```

설문 제출 및 결과 확인

```
const submitSurvey = async () => {
  const formattedAnswers = Object.entries(selectedAnswers)
    .map(([key, value]) => {
      // key는 질문의 인덱스 (0부터 시작)
      const baseValue = Number(key) * 2 + 1;
      const adjustedValue = baseValue + (value - 1); // value가 1이면 baseValue, 2면 baseValue + 1
      return `B${Number(key) + 1}=${adjustedValue}`;
    })
    .join(' ');

  const postData = {
    apikey: API_KEY,
    qestrnSeq: Q_NUM,
    trgetSe: '100209',
    name: userName,
    gender: gender === 'men' ? '100323' : '100324',
    school: '',
    grade: '',
    email: '',
    startDtm: Date.now(),
    answers: formattedAnswers,
  };
  console.log(postData);
  try {
    const response = await PostResultAPI(postData);
    console.log('response: ',response);
    const resultData = response.data.RESULT;
    console.log('resultData: ',resultData.url);
    const url = resultData.url;
    const encodedUrl = encodeURIComponent(url);
    // 쿼리 파라미터를 객체로 전달
    router.push({
      pathname: '/survey/surveyResult',
      query: { userName, url: encodedUrl },
```

적성 탐색(3)

커리어넷 결과 PDF 추출/ CHAT-GPT 프롬프트

```java
package com.ictedu.surveyfileupload.service;

import org.apache.pdfbox.pdmodel.PDDocument;

@Service
public class PdfProcessingService {

    public List<String> extractInfoFromPDF(byte[] fileData) throws IOException {
        System.out.println("PDF 처리 시작");
        List<String> extractedInfo = new ArrayList<>();

        String[] sections = {"중졸이하", "고졸", "대졸", "대학원졸", "계열무관", "인문", "사회", "교육", "공학", "자연", "의학", "예체능"};

        try (PDDocument document = PDDocument.load(fileData)) {
            System.out.println("PDF 파일 로드 성공");
            PDFTextStripper stripper = new PDFTextStripper();
            stripper.setStartPage(3);
            stripper.setEndPage(3);
            System.out.println("PDF 3페이지 텍스트 추출 시도");
            String pageText = stripper.getText(document);

            for (String line : pageText.split("\n")) {
                line = line.trim();

                // 섹션 이름은 콘솔 출력에서 제외
                if (isSectionName(line, sections)) {
                    System.out.println("섹션 시작: " + line + " (제외됨)");
                    continue; // 섹션 이름은 무시하고 다음 라인으로
                }

                System.out.println("라인 처리: " + line);

                if (!line.isEmpty()) {
                    // 줄의 모든 직업을 쉼표로 구분하여 개별적으로 처리
                    String[] jobs = line.split(",");
                    for (String job : jobs) {
                        job = job.trim();
                        addJobToList(extractedInfo, job);
                    }
                }
            }
        }

        // 추출된 내용이 없는 경우 메시지 추가
        if (extractedInfo.isEmpty()) {
            extractedInfo.add("추출할 내용이 없습니다");
            System.out.println("추출할 내용 없음 메시지 추가");
        } else {
            System.out.println("총 추출된 직업명 개수: " + extractedInfo.size());
            System.out.println("추출된 직업명 목록: " + extractedInfo);
        }

        System.out.println("PDF 처리 완료");
```

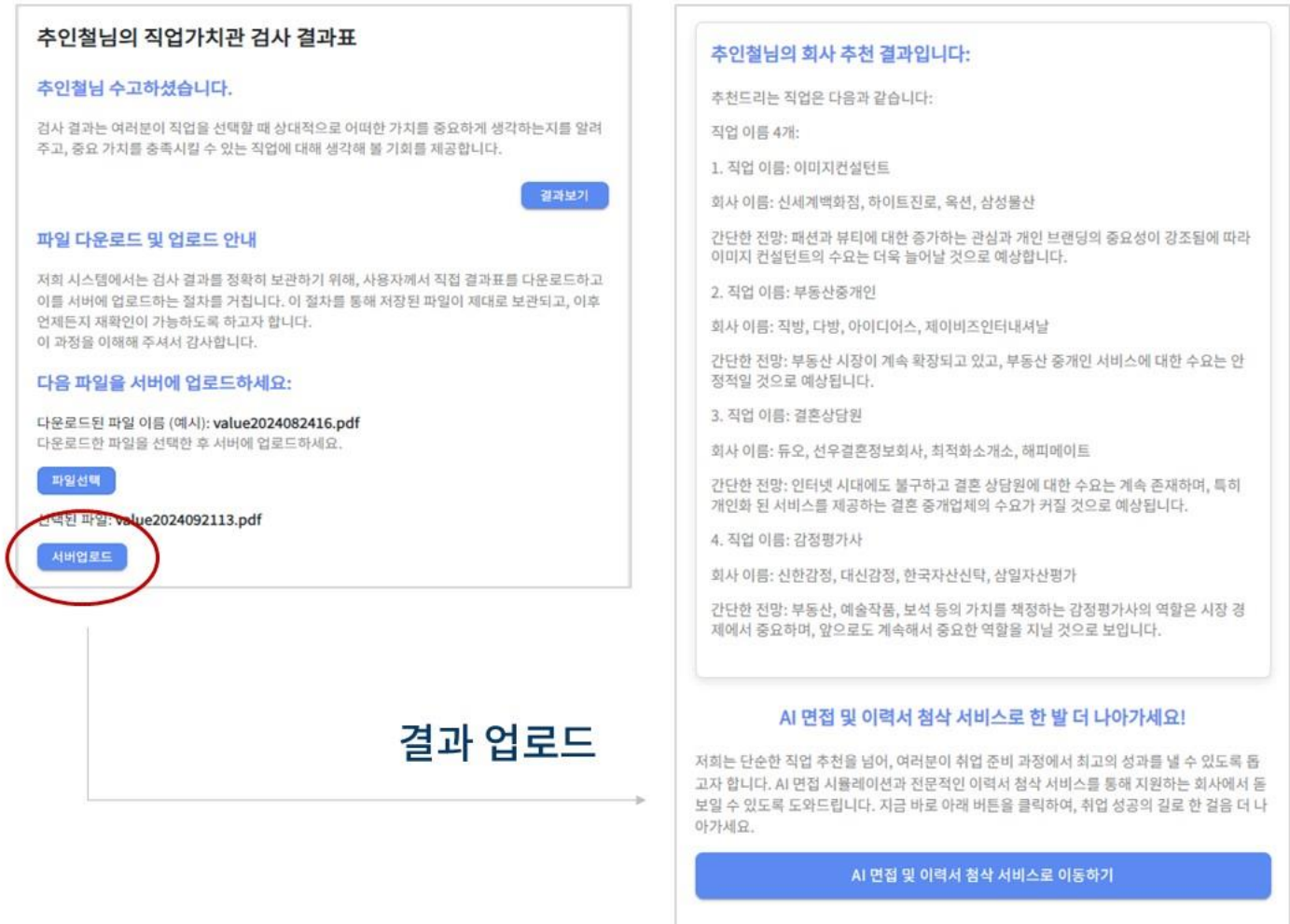
```java
private String buildPrompt(List<String> jobList) {
    StringBuilder promptBuilder = new StringBuilder();

    // 직업 목록 제시
    promptBuilder.append("다음은 내가 수집한 직업들입니다:\n");
    promptBuilder.append(String.join(", ", jobList));

    // 직업 이름의 부적절한 나눔과 수정 요구
    promptBuilder.append("\n\n1. '애널리스트'를 '애널'과 '리스트'로 나누는 것과 같은 부적절한 직업 이름이 있다면 수정하고 올바른 직업을 추천해줘.");

    // 특정 조건에 맞는 직업 제외
    promptBuilder.append("\n\n2. 다음 조건에 해당하는 직업들을 제외해줘: '중졸 이하', '고졸', '대졸', '대학원졸', '계열무관', '인문', '사회', '교육', '공학', '자연', '의학', '예체능'이라는 단어가 포함된 직업.");

    // 국내 근무 가능 직업 추천
    promptBuilder.append("\n\n3. 국내에서 근무할 수 있는 직업만 추천해줘. 외국 직업은 제외해줘.");

    // 직업 4개 선정
    promptBuilder.append("\n\n4. 총 4개의 직업을 선정해줘.");

    // 각 직업에 대한 국내 회사 4곳 추천
    promptBuilder.append("\n\n5. 선정된 직업당 국내 기업 4개씩 추천해줘. 외국 기업은 제외해줘.");


     // 회사 이름 중복 검사 및 특정 패턴 방지
    promptBuilder.append("\n\n6. 회사 이름의 뒤쪽에 같은 단어가 반복되지 않도록 해줘. 예를 들어, '병원', '사무소'처럼 특정 단어로 끝나는 회사가 여러 개 있다면, 그 단어가 계속 반복되지 않도록 다른 회사를 추천해줘.");
    promptBuilder.append("\n   예를 들어, '서울병원', '강남병원', '한강병원'처럼 '병원'으로 끝나는 회사는 1개만 추천해줘.");

    // 총 회사 수 16개로 제한
    promptBuilder.append("\n\n7. 총 추천 회사 수는 16개를 넘지 않도록 해줘.");

    // 각 직업에 대한 전망 요청
    promptBuilder.append("\n\n8. 추천된 각 직업에 대한 간단한 전망을 제공해줘.");

    // 조건 충족 여부 확인
    promptBuilder.append("\n\n9. 위의 조건들을 충족하는지 한 번 더 확실히 확인해줘.");

    // 양식 고지
    promptBuilder.append("\n\n<필수>\n너는 아래 양식에 맞춰서만 답변해야 해. 절대 양식을 벗어난 설명을 덧붙이거나 사족을 달지 말고, 반드시 아래 양식에만 맞춰서 답변해.");
    promptBuilder.append("\n추가 설명, 이유 또는 부가적인 조건에 대해 이야기하지 마. 네 답변은 오직 직업 이름, 회사 이름, 직업 전망에만 집중해야 해.");
    promptBuilder.append("\n절대 '이러한 조건에 맞춰 대답하다'는 식의 추가적인 설명을 하지 말고, 예시 양식 외에는 어떤 말도 하지 말아줘.");


    // 답변 양식 예시 제공
    promptBuilder.append("\n추천드리는 직업은 다음과 같습니다:\n\n");
    promptBuilder.append("직업 이름 4개:\n");

    // 각 직업과 회사 추천 목록 및 직업 전망 양식
    for (int i = 0; i < 4; i++) {
        promptBuilder.append("\n").append(i + 1).append(". 직업 이름: [회사 이름 1], [회사 이름 2], [회사 이름 3], [회사 이름 4]");
        promptBuilder.append("\n   간단한 전망: [직업 전망]");
    }
```

적성 탐색(4)

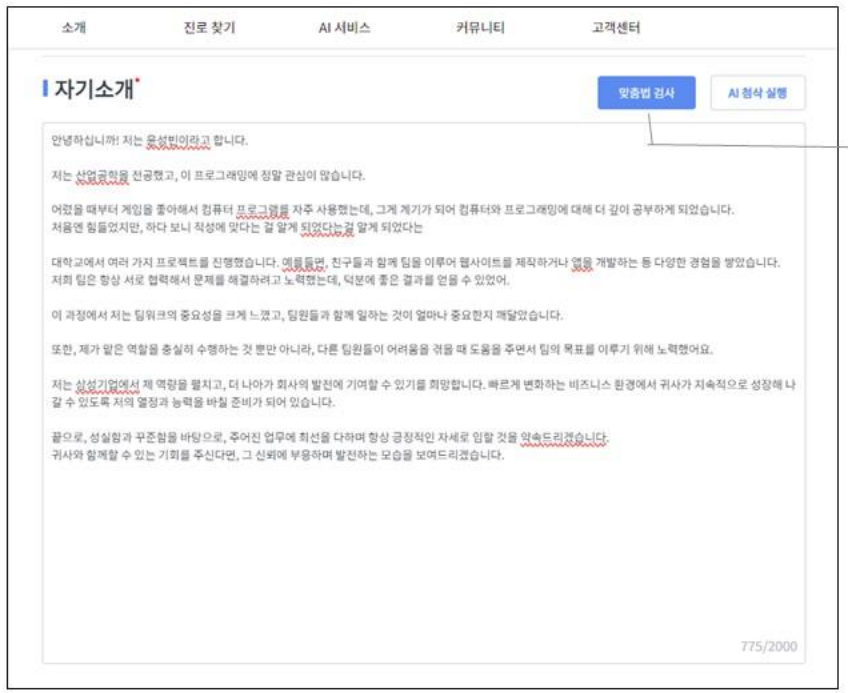
결과 처리 및 업로드

```
useEffect(() => {
  const storedUserName = localStorage.getItem('username');

  if (storedEmail) {
    setEmail(storedEmail);
  }

  if (storedUserName) {
    setUserName(storedUserName);
  }

  if (url) {
    const decodedUrl = decodeURIComponent(url);
    setValidUrl(decodedUrl);
  }

  setFileName('value2024082416.pdf');
}, [url]);

const handleShowIframe = () => {
  setShowIframe(true);
};

const handleFileChange = (event) => {
  setFile(event.target.files[0]);
};

const handleFileUpload = async () => {
  if (file && email) {
    const formData = new FormData();
    formData.append('file', file);
    formData.append('email', email);

    try {
      setLoading(true); // Start loading
      const response = await axios.post('http://localhost:8080/api/files/upload', formData, {
        headers: {
          'Content-Type': 'multipart/form-data',
        },
      });
      console.log('File uploaded successfully:', response.data);

      const jobList = response.data.split('\n').map(item => item.trim()).filter(item => item);

      const gptResponse = await axios.post('http://localhost:8080/api/survey/chatgpt', {
        jobList: jobList,
        userName: userName
      });
```

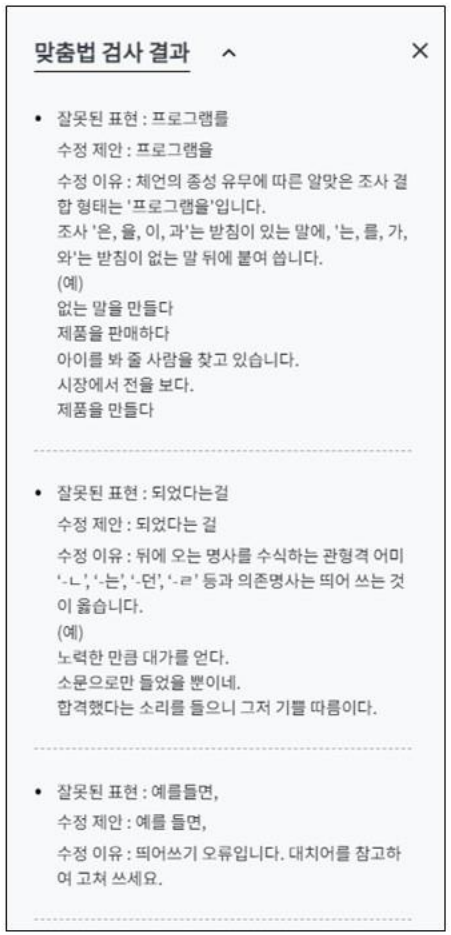
적성 검사 결과



이력서 등록 페이지(1) - hanspell 기반 맞춤법 검사



맞춤법 검사 작성 화면



맞춤법 검사 결과 화면

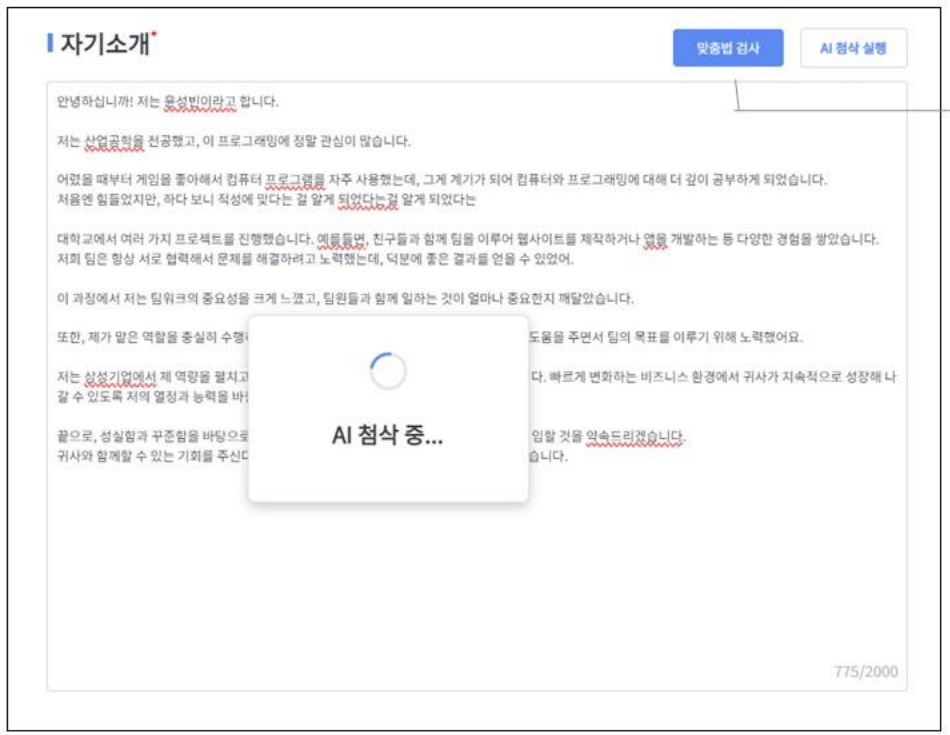
```javascript
const express = require('express');
const hanspell = require('hanspell');
const bodyParser = require('body-parser');
const cors = require('cors');  // cors 모듈을 가져옵니다.

const app = express();

// CORS 설정
app.use(cors({
  origin: 'http://localhost:3000', // 요청을 허용할 출처
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'], // 허용할 HTTP 메서드
  allowedHeaders: ['Content-Type', 'Authorization'], // 허용할 요청 헤더
}));

// 모든 경로에 대해 OPTIONS 메서드를 처리하도록 설정
app.options('*', cors());

app.use(bodyParser.json());

app.post('/check-spelling', (req, res) => {
  const sentence = req.body.sentence;

  hanspell.spellCheckByDAUM(sentence, 6000, (result) => {
    res.json(result);
  }, () => {
    console.log('Check finished');
  }, (err) => {
    console.error('Error:', err);
    res.status(500).send('Spelling check error');
  });
});

const PORT = 3001;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

hanspellserver.js 코드

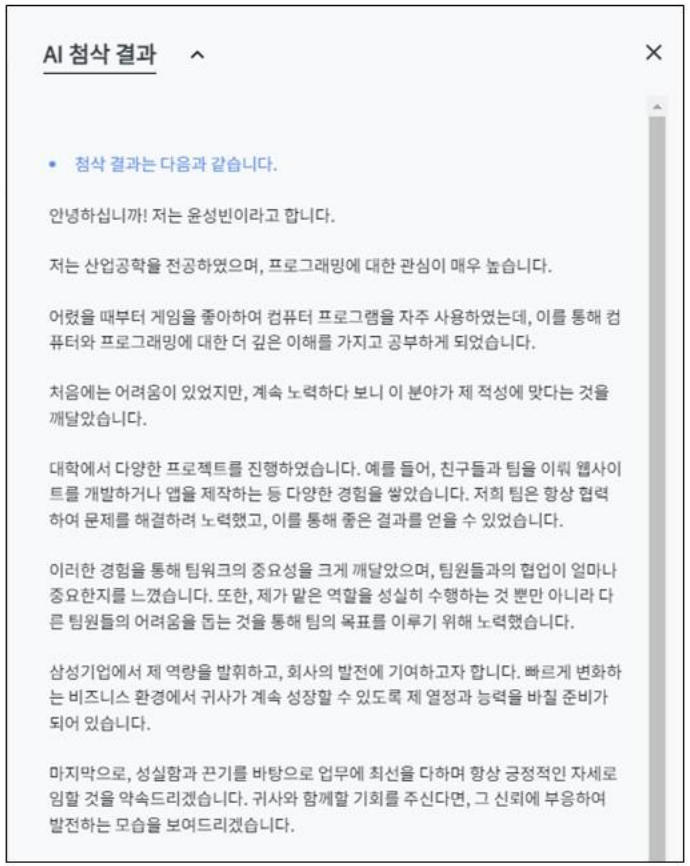
```javascript
//한스펠 맞춤법 검사
const handleProofread = async (text) => {
  if (text.trim() === '') {
    setProofreadResult([]);
    setIsProofreadSidebarOpen(true);
    return;
  }

  try {
    const response = await axios.post('http://localhost:3001/check-spelling', {sentence: text,});
    if (response.data.length === 0) {setProofreadResult([]);} else {setProofreadResult(response.data);}
    setIsProofreadSidebarOpen(true);
  } catch (error) {
    console.error('맞춤법 검사 중 오류 발생:', error);
    setProofreadResult([]);
    setIsProofreadSidebarOpen(true);
  }
};
```

hanspell api 호출 코드

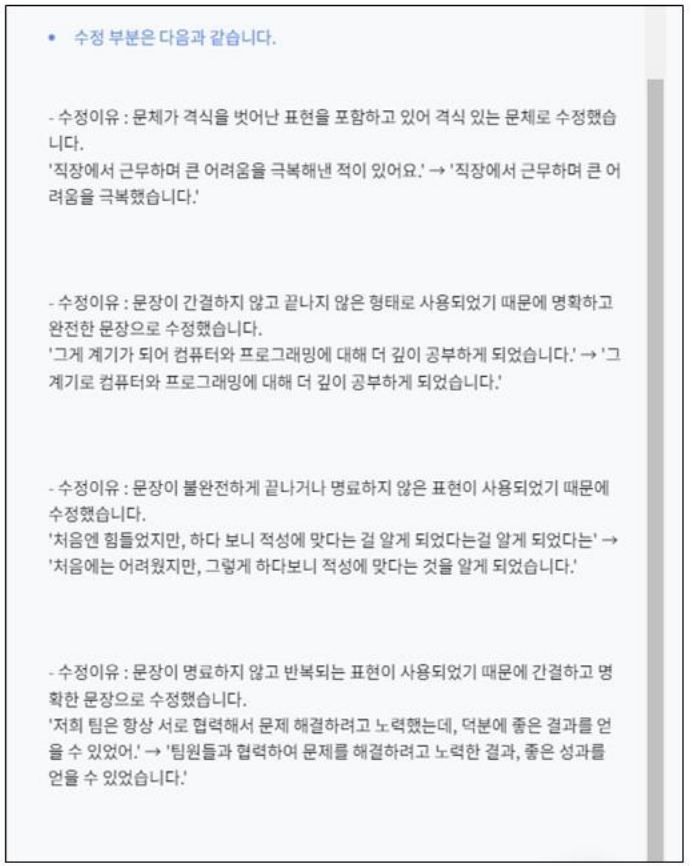
이력서 등록 페이지(2) - gpt 3.5 turbo기반 AI 첨삭



AI 첨삭 작성 화면



AI 첨삭 결과 화면 (1)



AI 첨삭 결과 화면 (2)

이력서 등록 페이지(2) - gpt 3.5 turbo기반 AI 첨삭

```jsx
const handleAiProofread = async (text) => {
  if (text.trim() === '') {
    setAiProofreadResult([{ message: "입력된 텍스트가 없습니다." }]);
    setIsAiProofreadSidebarOpen(true);
    return;
  }
  setLoading(true);
  try {
    const response = await axios.post('http://localhost:8080/api/chatgpt-self', {
      text,
    });

    if (response.data) {
      const formattedText = response.data.split('▶').map((item, index) => {
        if (index > 0) {
          return (
            <div key={index} style={{ marginTop: '16px' }}>
              <span style={{ fontSize: '6px', position: 'relative', top: '-3.5px',color:'#5A8AF2'}}>●</span>    
              <span dangerouslySetInnerHTML={applyColorToQuotes(item)} />
            </div>
          );
        }
        return <span key={index} dangerouslySetInnerHTML={applyColorToQuotes(item)} />;
      });
      setAiProofreadResult([{ message: formattedText }]);
    } else {
      setAiProofreadResult([{ message: "AI 첨삭 결과를 불러오지 못했습니다." }]);
    }
    setIsAiProofreadSidebarOpen(true);
  } catch (error) {
    console.error('AI 첨삭 중 오류 발생:', error);
    setAiProofreadResult([{ message: "AI 첨삭 중 오류가 발생했습니다." }]);
    setIsAiProofreadSidebarOpen(true);
  } finally {
    setLoading(false);
  }
};
```

gpt 3.5 turbo api 호출 코드

```java
package com.ictedu.proofread.controller;

import java.io.IOException;
import java.util.Map;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.ictedu.proofread.service.ProofreadSelfService;

@RestController
public class ProofreadSelfController {

    private final ProofreadSelfService proofreadService;

    public ProofreadSelfController(ProofreadSelfService proofreadService) {
        this.proofreadService = proofreadService;
    }

    @PostMapping("/api/chatgpt-self")
    public String getChatGPTResponse(@RequestBody Map<String, String> requestData) {
        try {
            String text = requestData.get("text");
            return proofreadService.getChatGPTResponse(text);
        } catch (IOException e) {
            e.printStackTrace();
            return "Error occurred while processing your request: " + e.getMessage();
        } catch (Exception e) {
            e.printStackTrace();
            return "An unexpected error occurred: " + e.getMessage();
        }
    }
}
```

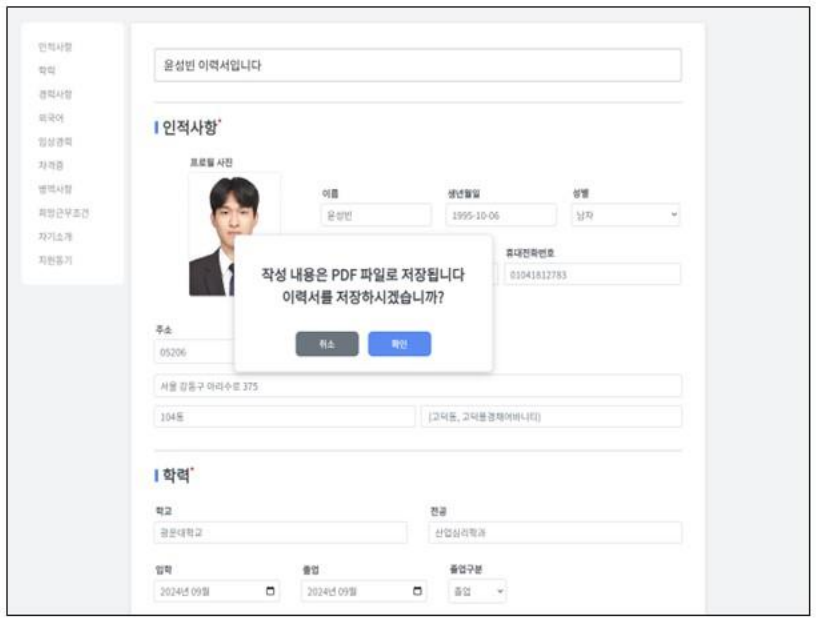
gpt 3.5 turbo 서비스 클래스 코드 (백엔드)

이력서 등록 페이지(2) - gpt 3.5 turbo기반 AI 첨삭

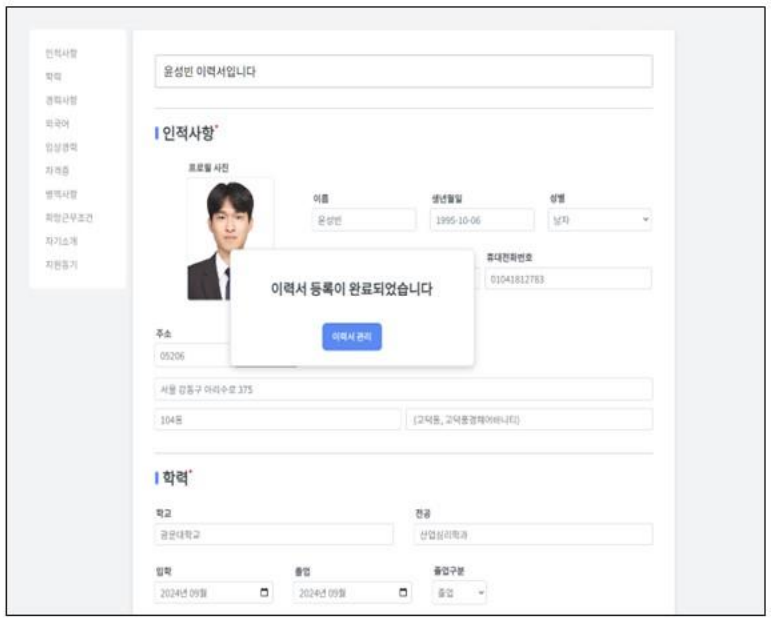
```
// 프롬프트 생성
    StringBuilder promptBuilder = new StringBuilder();
    promptBuilder.append("우리는 웹페이지의 이용자가 이력서의 '자기소개' 파트에 작성한 텍스트를 기반으로 자기소개 첨삭을 할거야.\n");
    promptBuilder.append("이용자는 아직 회사에 입사하지 않은 상태고, 회사 입사를 위한 이력서의 자기소개란에 자기소개를 적고있는 상황이야.\n");
    promptBuilder.append("회사 입사를 위한 공식적인 자기소개서 작성이니까 사용자는 공식적이고 격식있는 문체로 텍스트를 작성하겠지.\n");
    promptBuilder.append("우리는 이 텍스트를 사용자에게 받아서 분석한 뒤 AI첨삭을 해주는 역할을 하는거야.\n");
    promptBuilder.append("이용자에게 첨삭 결과를 보여줄 때는 반드시 존댓말을 사용하고 일정한 어투를 유지해야 해.\n");
    promptBuilder.append("텍스트를 읽고 자기소개 첨삭을 해주는 기준을 알려줄게. 그에 맞게 너가 메시지를 표시해주면 돼.\n");
    promptBuilder.append("첫번째 기준은 사용자가 작성한 텍스트가 공식적인 이력서 작성에 맞지 않는 문체인 경우야.\n");
    promptBuilder.append("비격식적인 표현이나 구어체를 사용하는 경우 수정을 해줘.사용자가 만약 '~했어요'와 같은 요 로 끝나거나 ~할게, ~게 , ~네 등으로 문장이 끝나는 구어체로 텍스트를 작성했다면 '~습니다,~입니다' 와 같이 다로 끝나는 문체로 수정해줘.\n");
    promptBuilder.append("또한 사용자가 축약어를 사용했다면 이를 축약된 단어가 아닌 본래의 단어로 수정해줘.\n");
    promptBuilder.append("두번째 기준은 명료성과 간결성이야.\n");
    promptBuilder.append("비슷한 단어나 문장이 계속 사용되거나 문장이 완전하게 끝나지 않은 문장이 있는지 파악해주고 있다면 문장을 간결하고 명확하게 끝나게 수정해주면돼.\n");
    promptBuilder.append("문장이 완전하게 끝나지 않은 문장의 예시로는 '은,는,이,가' 등으로 문장이 불완전하게 끝나는 경우가 있겠지. 또한 명사로 문장이 끝나버리거나.\n");
    promptBuilder.append("위의 기준들에 따라 사용자의 텍스트를 수정하여 첨삭 결과 메시지를 띄울 때, '▶ 첨삭 결과는 다음과 같습니다.'로 제목을 보여주고 밑에 수정 결과 메시지를 띄워줘.\n");
    promptBuilder.append("반드시 수정이 완료된 사용자의 텍스트 전체 문장을 한 번에 보내줘.\n");
    promptBuilder.append("또한 사용자의 텍스트를 수정할 때는 요약을 하거나 글의 흐름을 바꾸면 안돼.\n");
    promptBuilder.append("원래의 문장 구조를 유지하되, 위의 기준에 맞지 않는 부분만 수정하는 식으로 해야 해.\n");

    promptBuilder.append("수정된 텍스트 전체 문장을 보냈다면, 다시 두 줄 띄우고 '▶ 수정 부분은 다음과 같습니다.'로 제목을 보여주고 밑에 수정 이유 메세지를 보여줘.\n");
    promptBuilder.append("수정 이유를 보여줄때는 - 하이픈으로 틀을 시작하고 '수정이유' : '수정 전 문장' → '수정 후 문장' 이런 형식이 하나의 틀이라고 생각하면돼. 하나의 틀에는 하나의 반드시 하나의 하이픈만 들어가야해. 따라서 반드시 '수정이유' 앞에만 하이픈이 붙어야겠지. '수정이유'가 틀의 시작이니까.\n");

    promptBuilder.append("위의 틀에서 '수정 이유' 에는 너가 수정을 한 이유가 들어가야하고 '수정 전 문장'에는 수정을 거치기 전 사용자의 텍스트 원본만 들어가야해. '수정 후 문장'은 너가 수정을 완료한 문장만 들어가야해. \n");
    promptBuilder.append("수정이유 틀인 '수정 전 문장' 과 '수정 후 문장' 이 텍스트는 포함시키지 마.이건 내가 너에게 알려주는 틀일뿐이야. 저 틀안에 내가 요청한 문장만 사용자에게 보여주면돼.  \n");
    promptBuilder.append("너가 이해하기 쉽게 수정이유 예시를 보여주자면 다음과같아.  \n");
    promptBuilder.append(" - 수정이유 : 문체가 비격식적인 표현을 포함하고 있어 격식 있는 문체로 수정했습니다.\n\n"
            + "'대학교에서 여러 가지 프로젝트를 진행했습니더.' → '대학교에서 여러 가지 프로젝트를 진행했습니다.'\n");
    promptBuilder.append("위의 예시를 참고해서 같은 틀과 형식으로 수정이유를 보여주면돼. \n");
    promptBuilder.append("<필수>\n");
    promptBuilder.append("위에서 언급한 내용들을 모두 반드시 지켜야 해.\n");
    promptBuilder.append("그리고 너는 첨삭 결과 외에는 아무것도 표시하면 안 돼.\n");
    promptBuilder.append("너가 수정한 부분들은 하나도 빠뜨리지 않고 반드시 모두 수정이유 메시지로 사용자에게 보여줘야해.\n");
    promptBuilder.append("내가 너에게 주는 지시 사항이나, 너가 나한테 대답하는 내용은 절대로 첨삭 결과에 포함되면 안 돼.\n");
    promptBuilder.append("결과에는 오직 첨삭 메시지와 관련된 내용만 포함시키고, 그 외의 불필요한 텍스트나 내용은 절대로 포함시키지 마.\n");
    promptBuilder.append("사용자에게는 오직 첨삭 결과와 수정 이유만 보여줘야 해.\n");
    promptBuilder.append("자 그럼 아래 텍스트를 읽고 위의 지시사항에 맞게 첨삭 결과를 출력해줘.\n");
    promptBuilder.append(text);
    promptBuilder.append("텍스트를 분석할때는 반드시 원본 그대로 분석을 한 뒤 첨삭을 진행해야해.\n");
```

gpt 3.5 turbo 프롬프트 빌드 - 자기소개 첨삭용

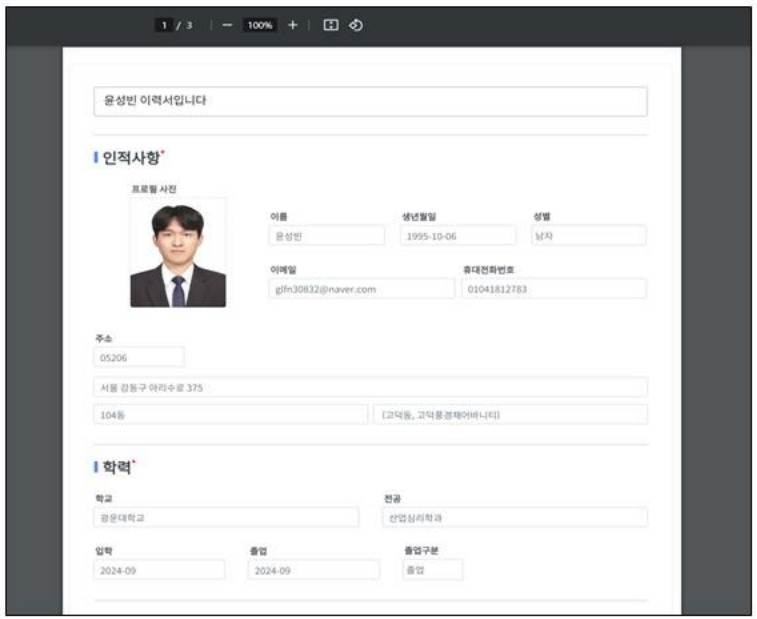
이력서 등록 페이지(3) - 페이지 캡쳐 및 PDF 변환



이력서 등록 PDF 저장 프론트 화면 (1)



이력서 등록 PDF 저장 프론트 화면 (2)



저장된 PDF 파일 화면

이력서 등록 페이지(3) - 페이지 캡쳐 및 PDF 변환

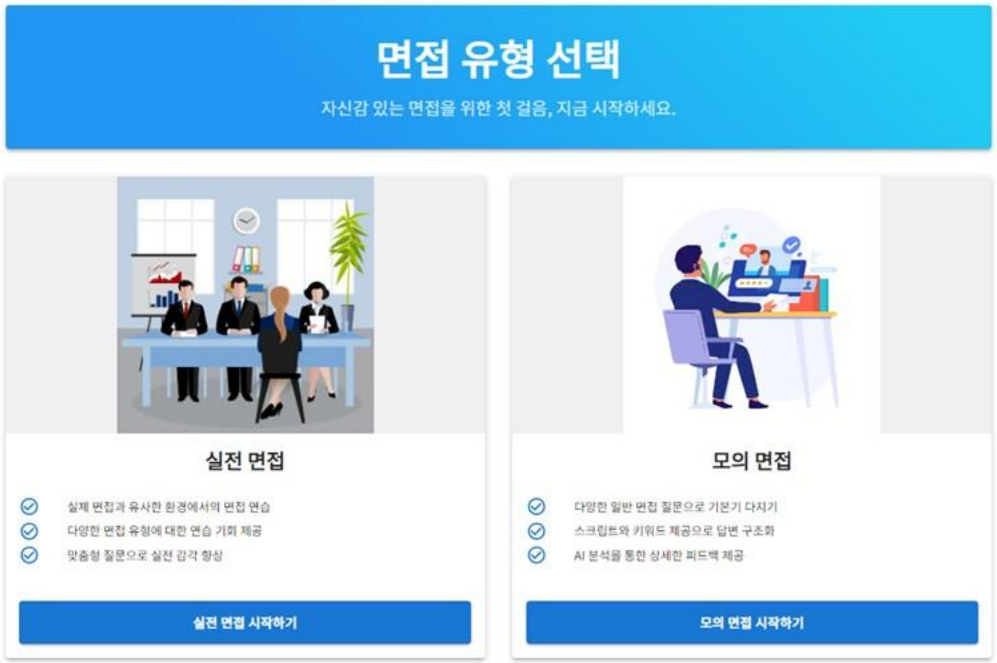
```
const generatePDF = async () => {
  const buttons = document.querySelectorAll('button');
  buttons.forEach(button => button.style.display = 'none');
  const content = document.getElementById('resume-content');
  const canvas = await html2canvas(content, {
    scale: 2,
    useCORS: true,
    scrollX: 0,
    scrollY: 0,
  });

  const imgData = canvas.toDataURL('image/png');
  const pdf = new jsPDF('p', 'mm', 'a4', true);
  const imgWidth = 207;
  const pageHeight = 295;
  const imgHeight = (canvas.height * imgWidth) / canvas.width;
  let heightLeft = imgHeight;
  let position = 0;

  pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
  heightLeft -= pageHeight;

  while (heightLeft >= 0) {
      position = heightLeft - imgHeight;
      pdf.addPage();
      pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
      heightLeft -= pageHeight;
  }
  const pdfBlob = pdf.output('blob');
  buttons.forEach(button => button.style.display = '');
  return pdfBlob;
};
```

html2canvas 라이브러리 사용
화면캡쳐시 불필요한 버튼 제거

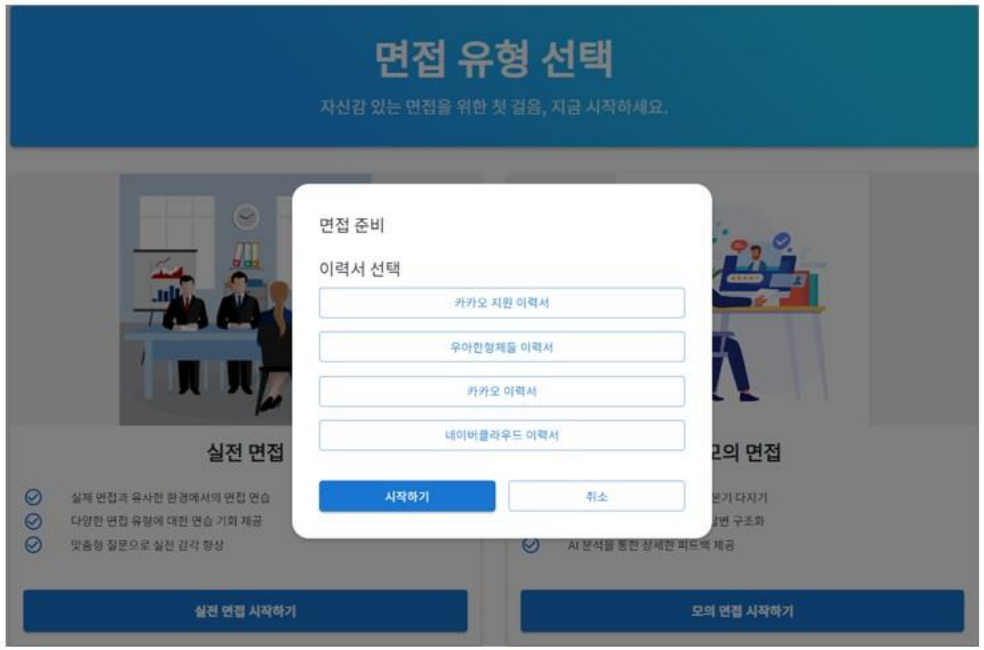
jspdf 라이브러리 사용
PDF 저장시 서식 및 속성 설정

면접 유형 선택 화면

면접 유형 선택 화면



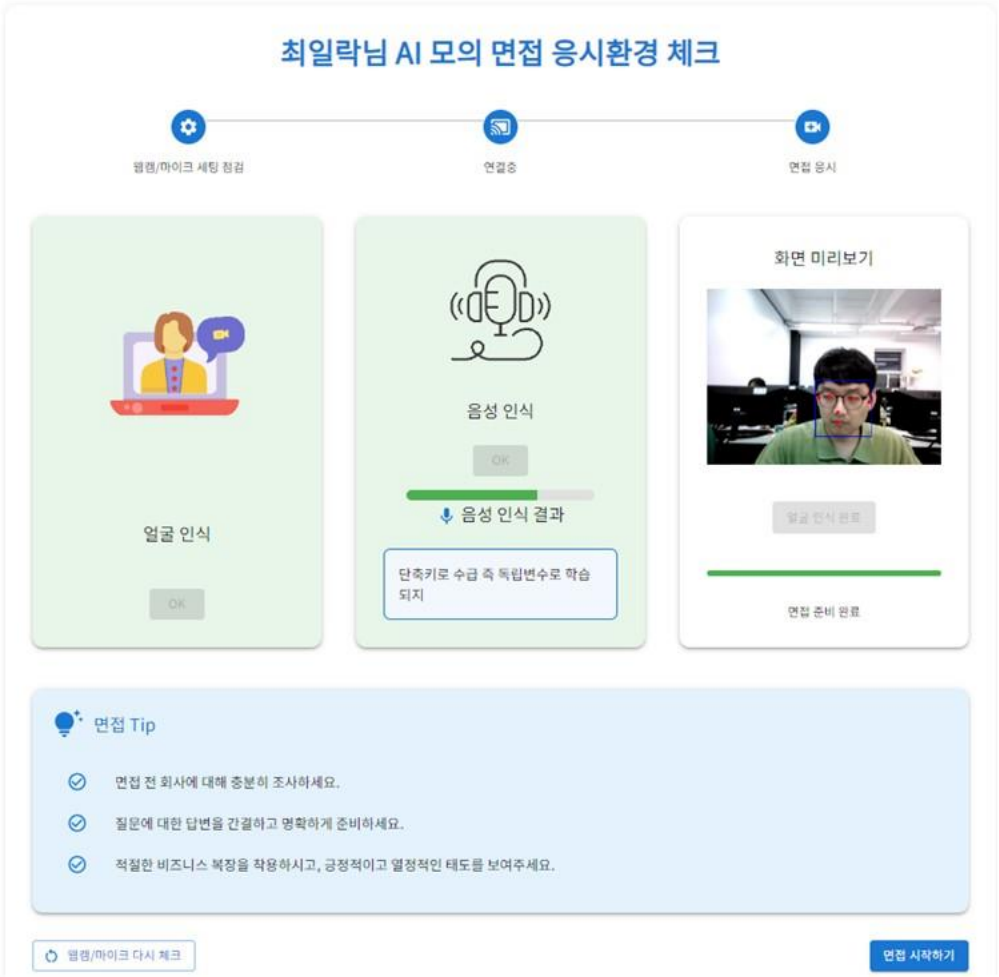
이력서 선택 화면 및 코드




```
const fetchResumes = async () => {
  const email = userStore.email;
  try {
    const response = await axios.get(`http://localhost:8080/api/resume/user-resumes?email=${email}`);
    const sortedResumes = response.data.sort((a, b) => new Date(b.createdDate) - new Date(a.createdDate));
    setResumeList(sortedResumes);
  } catch (error) {
    console.error('이력서 데이터를 불러오는 중 오류 발생:', error);
  }
};

useEffect(() => {
  fetchResumes();
}, []);
```

면접환경 세팅

AI 면접환경 준비 화면



카메라 얼굴 검출 코드

```
const onFaceDetectionResults = (results) => {
  if (!canvasRef.current) return;

  const canvasCtx = canvasRef.current.getContext('2d');
  if (!canvasCtx) return;

  const width = canvasRef.current.width;
  const height = canvasRef.current.height;

  canvasCtx.clearRect(0, 0, width, height);

  if (results.detections && results.detections.length > 0) {
    setFaceDetected(true);
    results.detections.forEach((detection) => {
      const boundingBox = detection.boundingBox;

      canvasCtx.strokeStyle = 'blue';
      canvasCtx.lineWidth = 2;
      canvasCtx.strokeRect(
        boundingBox.xCenter * width - (boundingBox.width * width) / 2,
        boundingBox.yCenter * height - (boundingBox.height * height) / 2,
        boundingBox.width * width,
        boundingBox.height * height
      );

      if (detection.landmarks) {
        detection.landmarks.forEach((landmark) => {
          canvasCtx.beginPath();
          canvasCtx.arc(landmark.x * width, landmark.y * height, 5, 0, 2 * Math.PI);
          canvasCtx.fillStyle = 'red';
          canvasCtx.fill();
        });
      }
    });
  } else {
    setFaceDetected(false);
    console.log("No face detected");
  }
};
```

면접환경 세팅

음성, 카메라 체크 코드

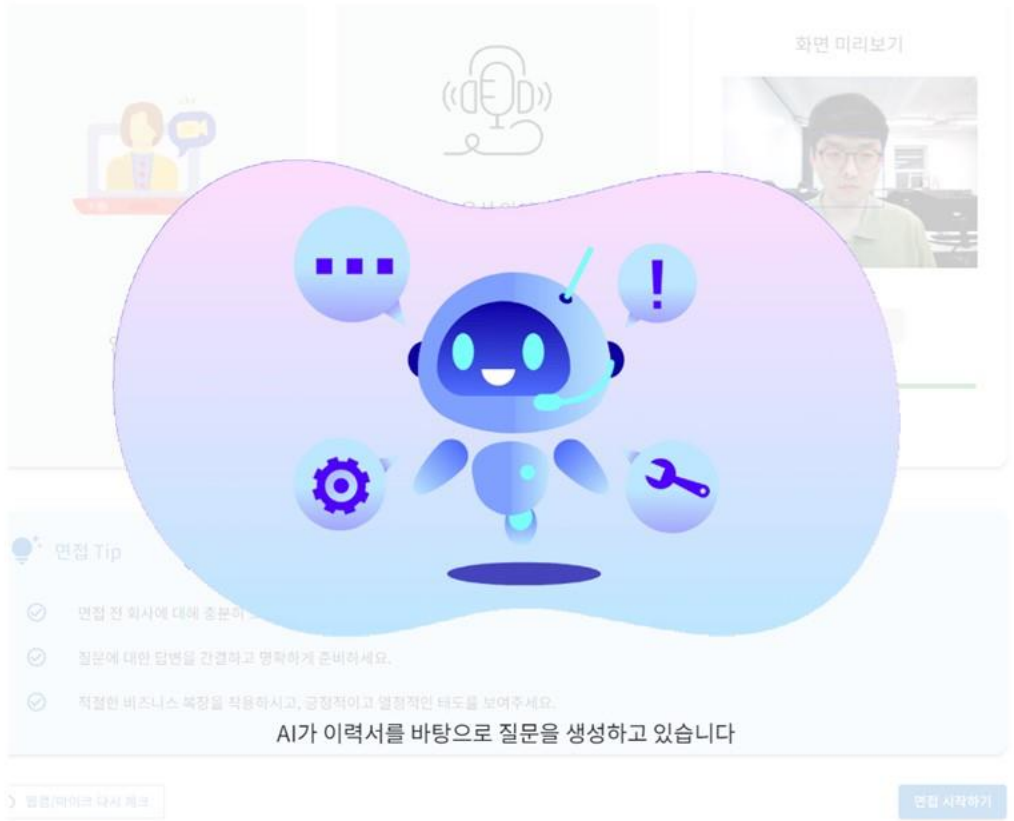
```
const checkMic = async () => {
  try {
    if (audioContextRef.current) {
      stopStream();
    }
    const mediaStream = await navigator.mediaDevices.getUserMedia({ audio: true });
    audioContextRef.current = new (window.AudioContext || window.webkitAudioContext)();
    analyserRef.current = audioContextRef.current.createAnalyser();
    const source = audioContextRef.current.createMediaStreamSource(mediaStream);
    source.connect(analyserRef.current);
    analyserRef.current.fftSize = 256;
    const bufferLength = analyserRef.current.frequencyBinCount;
    const dataArray = new Uint8Array(bufferLength);

    const updateAudioLevel = () => {
      analyserRef.current.getByteFrequencyData(dataArray);
      const average = dataArray.reduce((acc, value) => acc + value, 0) / bufferLength;
      interviewStore.setAudioLevel(average / 128);
      animationFrameRef.current = requestAnimationFrame(updateAudioLevel);
    };
    updateAudioLevel();

    interviewStore.setMicReady(true);
    startListening();
  } catch (error) {
    console.error("마이크 접근 에러:", error);
    interviewStore.setMicReady(false);
  }
};
const checkCamera = async () => {
  try {
    if (stream) {
      stopStream();
    }
    const mediaStream = await navigator.mediaDevices.getUserMedia({ video: true });
    console.log("Camera stream obtained:", mediaStream);
    interviewStore.setStream(mediaStream);
    interviewStore.setCameraReady(true);
    if (videoRef.current) {
      videoRef.current.srcObject = mediaStream;
      await new Promise(resolve => videoRef.current.onloadedmetadata = resolve);
      await initializeFaceDetection();
    }
    initializeFaceDetection();
  } catch (error) {
    console.error("카메라 접근 에러:", error);
    interviewStore.setCameraReady(false);
  }
};
```

Web Speech API 사용 코드(STT)

```
const initializeSpeechRecognition = () => {
  if ('webkitSpeechRecognition' in window) {
    recognitionRef.current = new window.webkitSpeechRecognition();
    recognitionRef.current.continuous = true;
    recognitionRef.current.interimResults = true;
    recognitionRef.current.lang = 'ko-KR';

    recognitionRef.current.onresult = (event) => {
      const transcript = Array.from(event.results)
        .map(result => result[0].transcript)
        .join('');
      setTranscript(transcript);
    };

    recognitionRef.current.onerror = (event) => {
      console.error('Speech recognition error', event.error);
      setIsListening(false);
    };

    recognitionRef.current.onend = () => {
      setIsListening(false);
    };
  } else {
    console.error('Web Speech API is not supported in this browser.');
  }
};

const speakText = useCallback((text, onEnd) => {
  if ('speechSynthesis' in window) {
    if (speechRef.current) {
      window.speechSynthesis.cancel();
    }
    const speech = new SpeechSynthesisUtterance(text);
    speech.lang = 'ko-KR';
    speech.onend = onEnd;
    speechRef.current = speech;
    window.speechSynthesis.speak(speech);
  }
}, []);
```

면접질문 생성

면접 질문 생성 화면



면접 질문 생성 화면 로직

```jsx
const LoadingOverlay = () => {
  const [currentMessageIndex, setCurrentMessageIndex] = useState(0);
  const messages = [
    'AI가 사용자의 이력서를 분석하고 있습니다',
    'AI가 이력서를 바탕으로 질문을 생성하고 있습니다',
    '잠시만 기다려주세요',
    '이제 곧 질문을 보실 수 있습니다'
  ];

  useEffect(() => {
    const interval = setInterval(() => {
      setCurrentMessageIndex((prevIndex) => (prevIndex + 1) % messages.length);
    }, 5000);

    return () => clearInterval(interval);
  }, []);

  return (
    <div className={styles.overlay}>
      <Image
        src="/images/interviewLoading.gif"
        alt="Loading"
        width={900}
        height={900}
        unoptimized
      />
      <div className={styles.loadingText}>
        {messages[currentMessageIndex]}
      </div>
    </div>
  );
};
```

모의면접 질문 생성 로직

OpenAI API GPT 질문 호출 코드

```
private String createPrompt(String type, List<String> keywords) {
    String basePrompt = "당신은 경험 많은 인사면접관으로서, %s 면접에 적합한 간결하고 자연스러운 질문을 1개 생성해주세요.\n\n" +
            "키워드: %s\n\n" +
            "주의사항:\n" +
            "- 이력서 키워드 기반으로 간결한 질문을 만들어주세요.\n" +
            "- 지원자의 경험과 역량을 파악할 수 있는 개방형 질문으로 구성해주세요.\n" +
            "- 존댓말을 사용하고, 질문의 길이는 이전보다 2/3 정도로 줄여주세요.";

    return String.format(basePrompt, type, String.join(", ", keywords));
}

private String callChatGPTApi(String prompt) {
    // ChatGPT API 설정
    String apiUrl = "https://api.openai.com/v1/chat/completions";
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(interviewApiKey); // 실제 API 키로 교체 필요

    // API 요청 본문 구성
    Map<String, Object> requestBody = new HashMap<>();
    requestBody.put("model", "gpt-3.5-turbo");
    requestBody.put("messages", Arrays.asList(
            Map.of("role", "system", "content", "당신은 한국어로 응답하는 면접 전문가입니다."),
            Map.of("role", "user", "content", prompt)
    ));

    HttpEntity<Map<String, Object>> request = new HttpEntity<>(requestBody, headers);

    // API 호출 및 응답 처리
    ResponseEntity<Map> response = restTemplate.postForEntity(apiUrl, request, Map.class);

    if (response.getStatusCode() == HttpStatus.OK) {
        List<Map<String, Object>> choices = (List<Map<String, Object>>) response.getBody().get("choices");
        Map<String, Object> message = (Map<String, Object>) choices.get(0).get("message");
        return (String) message.get("content");
    } else {
        throw new RuntimeException("ChatGPT API 호출 실패: " + response.getStatusCode());
    }
}
```

공통 질문 생성 프롬프트 코드

```
private String createCommonPrompt(List<String> keywords) {
    String[] questions = {
        "팀워크와 협업의 중요성에 대해 어떻게 생각하시나요? 과거의 경험을 바탕으로 팀 내에서 어떻게 협력하며 문제를 해결해왔는지 말씀해주시겠어요?",
        "입사 후 어떤 목표를 달성하고 싶으신가요? 이를 위해 어떤 구체적인 계획이나 전략을 가지고 계신가요?",
        "이 직장을 선택한 이유와 기대하는 점은 무엇인가요? 특히 이 회사가 귀하의 경력 발전에 어떻게 기여할 것이라고 생각하시는지 말씀해 주시겠어요?",
        "업무와 개인 생활의 균형을 어떻게 유지하고 계신가요? 이를 위해 어떤 방법이나 습관을 실천하고 계신지 설명해 주세요.",
        "팀 내에서 리더십을 발휘한 경험에 대해 설명해 주시겠어요? 어떤 상황에서 리더십을 발휘했고, 그 결과는 어땠는지 구체적으로 말씀해 주세요.",
        "이 회사에서 이루고 싶은 장기적인 비전은 무엇인가요? 그 비전을 달성하기 위해 어떤 노력을 기울일 계획이신지 알려주세요."
    };
    Random random = new Random();
    String selectedQuestion = questions[random.nextInt(questions.length)];

    String basePrompt = "당신은 경험 많은 면접관입니다. 다음의 키워드를 바탕으로 간결하고 명확한 개방형 질문을 1개 작성해주세요:\n" +
            selectedQuestion + "\n\n" +
            "키워드: %s\n\n" +
            "주의사항:\n" +
            "- 질문은 간결하면서도 지원자의 경험과 역량을 잘 평가할 수 있도록 작성하세요.\n" +
            "- 개방형 질문으로 구성하여 상세한 답변을 유도하세요.\n" +
            "- 예시 문구(질문 예시, 지원자님 등)는 사용하지 마세요.\n" +
            "- 정중한 표현으로 마무리 해주세요.";
    return String.format(basePrompt, String.join(", ", keywords));
}

private String createCompanyPrompt(String company, List<String> keywords) {
    String basePrompt = "당신은 %s 회사의 면접관입니다. 다음 키워드를 바탕으로 우리 회사에 지원한 후보자에게 물어볼 수 있는 구체적이고 깊이 있는 질문을 1개 생성해
            "키워드: %s\n\n" +
            "주의사항:\n" +
            "- 회사의 특성과 산업을 고려한 질문을 만들어주세요.\n" +
            "- 지원자의 경험과 우리 회사와의 연관성을 파악할 수 있는 질문으로 구성해주세요.\n" +
            "- 개방형 질문으로 구성하여 상세한 답변을 유도해주세요.\n" +
            "- 예시 문구(질문 예시, 지원자님 등)는 사용하지 마세요."+
            "- 정중한 표현으로 마무리 해주세요.";

    return String.format(basePrompt, company, String.join(", ", keywords));
}
```

이력서 질문 생성 프롬프트 코드

```
private String generateScript(String questionText, List<String> keywords, String companyName) {
    String scriptPrompt = "당신은 면접에 임하는 지원자입니다. 다음 질문에 대해 1분 이내로 답변할 수 있는 구체적이고 논리적인 답변을 작성해 주세요. " +
        "이 답변은 면접 분석 기준에 따라 평가될 예정이므로 각 항목을 충족하도록 유의해 주세요.\n\n" +
        "질문: " + questionText + "\n\n" +
        "답변 작성 가이드라인:\n" +
        "1. **시간 제한**: 답변은 1분(약 150-200단어) 이내로 간결하게 작성하세요.\n" +
        "2. **내용 분석**: 답변은 논리적으로 구성하고, 핵심 아이디어를 명확하게 전달하세요. 가능한 경우 간단한 예시를 포함하되, 실제 경험이 없는 내용은 언급하지 마세요
        "3. **감정 분석**: 긍정적이고 자신감 있는 어조를 유지하세요.\n" +
        "4. **언어 사용**: 전문적인 어휘와 업계 관련 용어를 적절히 사용하되, 간결성을 유지하세요.\n" +
        "5. **어조 및 텐션**: 답변의 어조는 일관되고 자연스럽게 유지하세요. 불필요한 반복이나 주저를 피하세요.\n" +
        "6. **회사 관련성**: " + companyName + "에 대한 관심과 이해를 보여주되, 해당 회사에서의 직접적인 경험이나 프로젝트를 언급하지 마세요. 대신 당신의 기술
        "7. **질문 이해도**: 질문의 핵심을 정확히 파악하고 그에 맞는 답변을 제공하세요.\n" +
        "8. **진실성**: 실제 경험과 지식에 기반한 답변만 제공하세요. 가상의 경험이나 허위 정보를 포함하지 마세요.\n" +
        "답변의 모든 항목을 만족하면서도 1분 이내로 전달할 수 있도록 작성해 주세요. 이 기준에 따라 답변이 평가될 것입니다.\n\n" +
        "키워드를 반영한 답변을 작성해 주세요. 주요 키워드: " + String.join(", ", keywords) + "\n" +
        "답변 형식: [답변 시작] 실제 답변 내용 [답변 종료]\n";

    return callChatGPTApi(scriptPrompt);
}
private List<String> extractKeywords(String text) {
    String keywordPrompt = String.format(
        "당신은 키워드 추출 전문가입니다. 다음 텍스트에서 면접 질문과 답변에 관련된 주요 키워드를 5개 이내로 추출해주세요. " +
        "키워드는 쉼표로 구분하여 나열해주세요:\n\n텍스트: %s\n\n주의사항:\n" +
        "- 추출된 키워드는 질문의 핵심 주제나 답변에서 강조된 역량, 경험을 대표해야 합니다.\n" +
        "- 일반적인 단어보다는 해당 직무나 산업과 관련된 전문적인 용어를 선호합니다.\n" +
        "- 키워드는 단일 단어 또는 짧은 구문으로 제한해주세요.", text);

    String keywordsString = callChatGPTApi(keywordPrompt);
    return Arrays.asList(keywordsString.split(","));
}
```

실전면접 질문 생성 로직

실전 질문 생성 코드

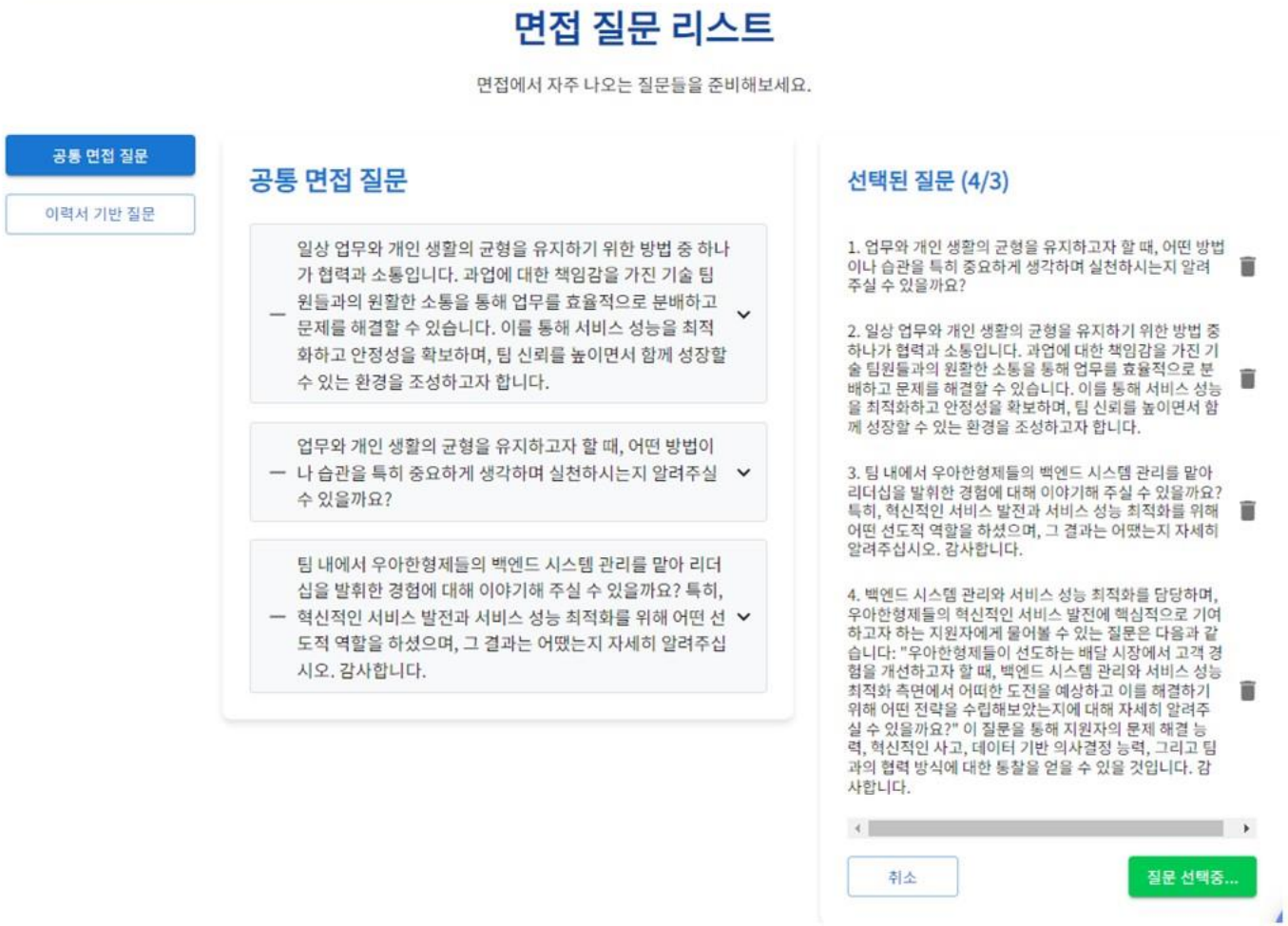
```
public List<Question> generateRealQuestions(List<String> resumeKeywords, User user, String desiredCompany) {
    List<Question> questions = new ArrayList<>();

    Interview interview = Interview.builder()
            .userId(user)
            .interviewType("real")
            .startTime(LocalDateTime.now())
            .endTime(LocalDateTime.now().plusHours(1))
            .overallFeedback("실전 면접 피드백")
            .createdTime(LocalDateTime.now())
            .updatedTime(LocalDateTime.now())
            .build();

    Interview savedInterview = interviewRepository.save(interview);

    List<String> questionTypes = Arrays.asList("공통 질문", "이력서 기반 질문", "성향 파악 질문", "상황 질문", "보상 선호 질문", "회사 관련 질문");
    List<String> companies = Arrays.asList(desiredCompany.split(", "));

    // 질문 생성
    for (String type : questionTypes) {
        String prompt = createRealPrompt(type, resumeKeywords, companies);
        String questionText = callChatGPTApi(prompt);
        String script = generateScript(questionText, resumeKeywords, "");
        List<String> generatedKeywords = extractKeywords(questionText + " " + script);
        String keywordsString = String.join(", ", generatedKeywords);

        Question question = Question.builder()
                .questionText(questionText)
                .questionType(type)
                .script(script)
                .keywords(keywordsString)
                .createdTime(LocalDateTime.now())
                .interviewId(savedInterview)
                .build();

        questions.add(question);
    }

    questionRepository.saveAll(questions);

    return questions;
}
```

실전 질문 생성 프롬프트 코드

```
private String createRealPrompt(String type, List<String> resumeKeywords, List<String> companies) {
    String basePrompt = "다음 정보를 바탕으로 실제 면접에서 물어볼 수 있는 %s 관련 질문을 생성해주세요.\n\n"
        + "이력서 키워드: %s\n\n"
        + "지원 회사: %s\n\n"
        + "질문 유형:\n"
        + "1번 공통 질문: 지원자의 경험과 역량을 전반적으로 파악할 수 있는 질문\n"
        + "2번 이력서 기반 질문: 지원자의 이력서나 자기소개서에 언급된 내용을 깊이 있게 탐색하는 질문\n"
        + "3번 성향 파악 질문: 지원자의 성격, 가치관, 업무 스타일을 파악하는 질문\n"
        + "4번 상황 질문: 특정 상황에서의 대처 능력을 확인하는 질문\n"
        + "5번 보상 선호 질문: 지원자의 동기 부여 요인을 파악하는 질문\n"
        + "6번 회사 관련 질문: 지원 회사에 대한 이해도와 적합성을 확인하는 질문\n\n"
        + "주의사항:\n"
        + "- 구체적이고 깊이 있는 질문을 만들어주세요.\n"
        + "- 지원자의 경험과 역량을 파악할 수 있는 개방형 질문으로 구성해주세요.\n"
        + "- 실제 면접관이 물어볼 법한 난이도의 질문을 만들어주세요.\n"
        + "- 필요한 경우, 꼬리 질문을 추가하여 더 깊이 있는 답변을 유도할 수 있게 해주세요.\n"
        + "- 질문은 정중한 표현으로 끝나도록 해주세요.\n"
        + "- 질문을 생성할때 1번,2번,3번,4번...9번을 붙이지 마세요.\n"
        + "- 질문은 자연스러운 문장 구조로 만들어주세요.\n\n"
        + "질문 예시:\n"
        + "직장 또는 프로젝트에서 몰입하여 성과를 낸 경험에 대해 말씀해 주시겠어요? 어떤 점에 집중하셨고, 그 경험을 통해 어떤 교훈을 얻으셨나요?\n"
        + "자기소개서에서 '솔선수범'이라는 표현을 사용하셨는데, 구체적인 사례와 그로 인한 팀이나 조직의 변화에 대해 설명해 주시겠어요?\n"
        + "우리 회사의 최근 기술 동향이나 시장 전략에 대한 견해와 함께, 지원자님께서 어떤 방식으로 기여할 수 있을지 구체적으로 말씀해 주시겠어요?\n\n"
        + "위의 예시와 같은 형식으로, %s 유형에 맞는 질문을 1개 생성해주세요. 질문은 자연스러운 대화체로 만들어주시고, 꼬리 질문이 있다면 같은 문단 안에 포함시켜 주세

    return String.format(basePrompt, type, String.join(", ", resumeKeywords), String.join(", ", companies), type);
}
```

모의 면접 질문 리스트 선택

모의 면접 질문 리스트 선택 화면

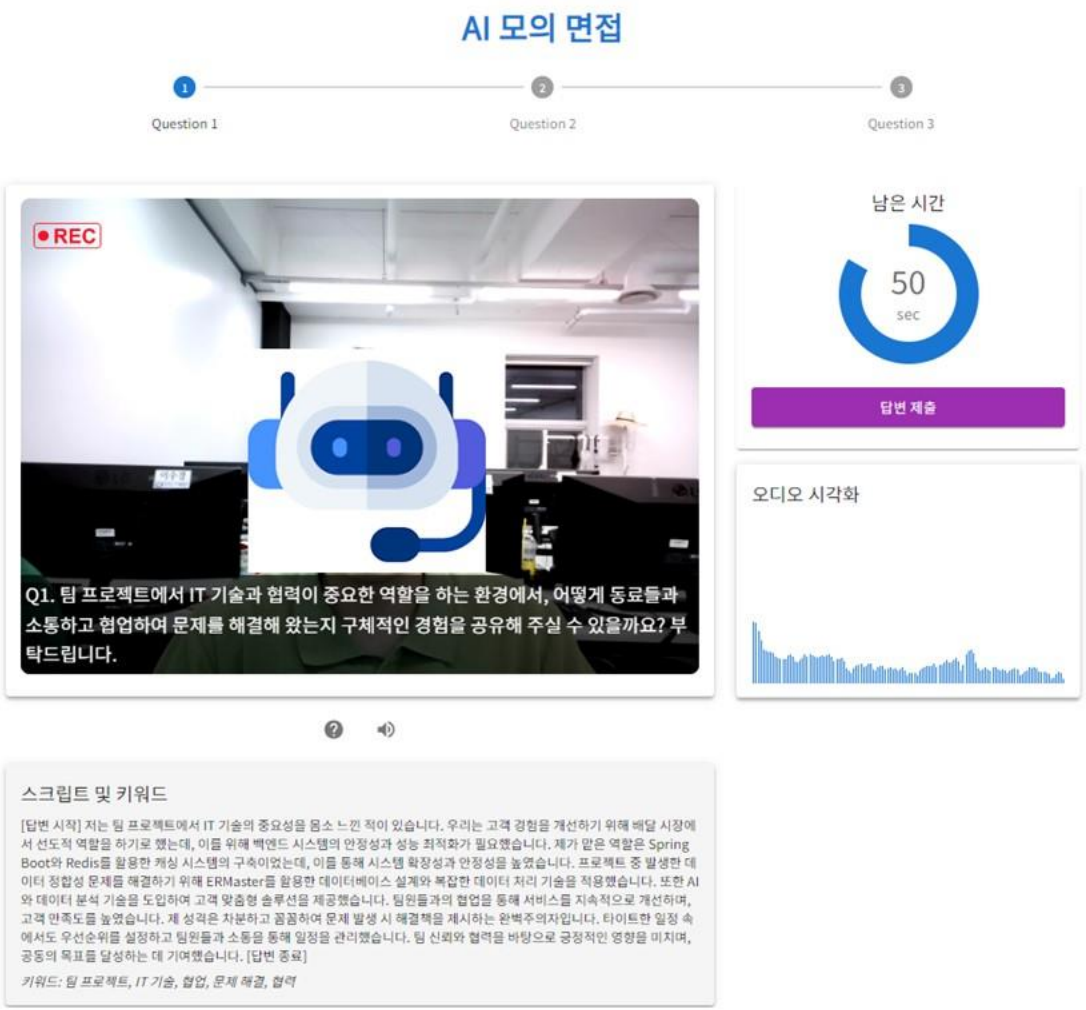


스크립트, 키워드 수정 코드

```
// 키워드를 쪼개기 위한 함수
const extractKeywords = (keywordsString) => {
  if (typeof keywordsString === 'string') {
    return keywordsString.replace('주요 키워드는 ', '').split(',').map(kw => kw.trim());
  }
  return [];
};
// 스크립트 또는 키워드 수정 모드로 전환하는 함수
const handleEdit = (type) => {
  interviewStore.setTempEdit({
    script: item.script,
    keywords: extractKeywords(item.keywords),
  });
  interviewStore.setEditMode({ id: item.id, type });
};

const handleSave = () => {
  interviewStore.saveQuestion({ id: item.id, ...tempEdit });
  interviewStore.setEditMode({ id: null, type: null });
};

const handleCancel = () => {
  interviewStore.setEditMode({ id: null, type: null });
  interviewStore.setTempEdit({ script: '', keywords: [] });
};

const handleKeywordDelete = (keyword) => {
  const updatedKeywords = tempEdit.keywords.filter((k) => k !== keyword);
  interviewStore.setTempEdit({ ...tempEdit, keywords: updatedKeywords });
};

const handleKeywordChange = (index, newKeyword) => {
  const updatedKeywords = [...tempEdit.keywords];
  updatedKeywords[index] = newKeyword;
  interviewStore.setTempEdit({ ...tempEdit, keywords: updatedKeywords });
};

const handleKeywordAdd = () => {
  interviewStore.setTempEdit({ ...tempEdit, keywords: [...tempEdit.keywords, ''] });
};
```

면접 녹화

AI 면접 녹화 화면



AI 면접환경 준비 화면

```jsx
// 원형 타이머 컴포넌트
const CircularTimer = ({ timeLeft }) => (
  <Box position="relative" display="inline-flex" flexDirection="column" alignItems="center">
    <Typography variant="h6" component="div" color="textPrimary" gutterBottom>
      남은 시간
    </Typography>
    <Box position="relative" display="inline-flex">
      <CircularProgress
        variant="determinate"
        value={(timeLeft / 60) * 100}
        size={150}
        thickness={7}
      />
      <Box
        top={0}
        left={0}
        bottom={0}
        right={0}
        position="absolute"
        display="flex"
        flexDirection="column"
        alignItems="center"
        justifyContent="center"
      >
        <Typography variant="h4" component="div" color="textSecondary">
          {timeLeft}
        </Typography>
        <Typography variant="subtitle1" component="div" color="textSecondary">
          sec
        </Typography>
      </Box>
    </Box>
  </Box>
);
```

면접 영상 업로드

Google Storage Bucket 업로드 코드

```
@Autowired
public VideoUploadService(ResourceLoader resourceLoader, @Value("${spring.cloud.gcp.credentials.location}") String credentialsPath)
    // GCP 인증 파일 로드
    Resource resource = resourceLoader.getResource(credentialsPath);
    if (!resource.exists()) {
        throw new IllegalArgumentException("Credentials file not found at " + credentialsPath);
    }
    try (InputStream keyFile = resource.getInputStream()) {
        GoogleCredentials credentials = GoogleCredentials.fromStream(keyFile);
        this.storage = StorageOptions.newBuilder().setCredentials(credentials).build().getService();
    }
}

public VideoEntity processVideo(MultipartFile video, Long userId, Long questionId) throws IOException {
    // GCS에 파일 업로드
    String videoLink = uploadToGoogleCloudStorage(video);

    VideoEntity videoEntity = new VideoEntity();
    videoEntity.setUserId(userId);
    videoEntity.setQuestionId(questionId);
    videoEntity.setFilePath(videoLink);  // GCS 링크 저장
    videoEntity.setFileName(video.getOriginalFilename());
    videoEntity.setFileSize(video.getSize());
    videoEntity.setUploadDate(LocalDateTime.now());

    return videoRepository.save(videoEntity);
}

private String uploadToGoogleCloudStorage(MultipartFile video) throws IOException {
    // 유니크한 파일명 생성
    String uniqueFileName = UUID.randomUUID() + "_" + StringUtils.cleanPath(video.getOriginalFilename());

    // 파일을 Google Cloud Storage에 업로드
    BlobInfo blobInfo = BlobInfo.newBuilder(bucketName, uniqueFileName)
            .setContentType(video.getContentType())
            .build();
    storage.create(blobInfo, video.getBytes());

    String fileLink = String.format("https://storage.googleapis.com/%s/%s", bucketName, uniqueFileName);
    logger.info("Video uploaded to Google Cloud Storage: {}", fileLink);

    return fileLink;  // GCS 링크 반환
}
```

면접 영상 FastAPI로 분석 요청

```
@PostMapping("/upload-video")
public ResponseEntity<?> uploadVideo(@RequestParam("video") MultipartFile video,
                                     @RequestParam("userId") Long userId,
                                     @RequestParam("questionId") Long questionId) {
    logger.info("Received file: {} with size: {}", video.getOriginalFilename(), video.getSize());
    try {
        // 1. Google Drive에 파일 업로드 및 링크 생성
        VideoEntity videoEntity = videoUploadService.processVideo(video, userId, questionId);

        // 2. FastAPI 서버에 Google Drive 링크를 보내 분석 요청
        String analysisUrl = fastApiServerUrl + "/analyze-video/";
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        // 요청 바디에 video_id와 Google Drive 링크를 전달
        Map<String, Object> requestBody = new HashMap<>();
        requestBody.put("video_id", videoEntity.getId());
        requestBody.put("video_link", videoEntity.getFilePath());  // Google Drive 링크 전달

        HttpEntity<Map<String, Object>> request = new HttpEntity<>(requestBody, headers);
        ResponseEntity<String> fastApiResponse = restTemplate.postForEntity(analysisUrl, request, String.class);

        // 응답 처리
        Map<String, Object> response = new HashMap<>();
        response.put("message", "Video processed successfully");
        response.put("videoId", videoEntity.getId());
        response.put("filePath", videoEntity.getFilePath());  // Google Drive 링크 포함

        if (fastApiResponse.getStatusCode() == HttpStatus.OK) {
            response.put("analysisStatus", "Analysis started");
        } else {
            response.put("analysisStatus", "Failed to start analysis");
            logger.error("Failed to start video analysis: {}", fastApiResponse.getBody());
        }

        return ResponseEntity.ok(response);
    } catch (IOException e) {
        logger.error("Error saving video: ", e);
        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body(Map.of("error", "Error saving video: " + e.getMessage()));
    } catch (Exception e) {
        logger.error("Error processing video: ", e);
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(Map.of("error", "Error processing video: " + e.getMessage()));
    }
```

면접 답변 분석

분석을 위해 작업 큐 및 면접 영상 다운로드 코드

```python
def download_video_from_gcs(gcs_url: str, local_path: str) -> str:
    """
    GCS에서 비디오 파일을 로컬로 다운로드하는 함수
    """
    logger.info(f"Downloading video from GCS: {gcs_url}")
    bucket_name = "cloud-storage-upload"
    gcs_file_path = gcs_url.split(f"https://storage.googleapis.com/{bucket_name}/")[1]

    credentials_path = os.path.join(os.path.dirname(__file__), 'credentials.json')
    credentials = service_account.Credentials.from_service_account_file(credentials_path)

    client = storage.Client(credentials=credentials)
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(gcs_file_path)

    local_file_path = os.path.join(local_path, gcs_file_path)
    os.makedirs(os.path.dirname(local_file_path), exist_ok=True)
    blob.download_to_filename(local_file_path)

    logger.info(f"Video downloaded to {local_file_path}")
    return local_file_path
# 스레드 풀 및 작업 큐 설정
executor = ThreadPoolExecutor(max_workers=5)  # 동시 작업 수 증가
task_queue: asyncio.Queue = asyncio.Queue(maxsize=10)  # 큐 크기 제한
task_statuses: Dict[int, Dict[str, Any]] = {}

# 큐 처리를 위한 비동기 함수
async def process_queue():
    while True:
        video_id = await task_queue.get()
        try:
            task_statuses[video_id] = {"status": "processing", "progress": 0}
            await background_video_analysis(video_id)
            task_statuses[video_id] = {"status": "completed", "progress": 100}
        except Exception as e:
            task_statuses[video_id] = {"status": "failed", "error": str(e)}
        finally:
            task_queue.task_done()
```

면접 영상 분석 완료시 백엔드에 메세지 전송 코드

```python
def send_analysis_complete_to_spring(video_id: int, analysis_results: Dict[str, Any]) -> bool:
    try:
        headers = {
            'Content-Type': 'application/json',
        }

        message = {
            "videoId": video_id,
            "message": "분석이 끝났습니다",
            "analysisResults": analysis_results
        }

        json_message = json.dumps(message, cls=NumpyEncoder)

        logger.info(f"Sending analysis complete message to Spring backend for video_id: {video_id}")
        logger.info(f"URL: {SPRING_BACKEND_URL}")
        logger.info(f"Message: {json_message}")

        response = requests.post(SPRING_BACKEND_URL, data=json_message, headers=headers)

        logger.info(f"Response status code: {response.status_code}")
        logger.info(f"Response content: {response.text}")

        if response.status_code == 200:
            logger.info(f"Analysis complete message sent successfully to Spring backend for video_id: {video_id}")
            return True
        else:
            logger.error(f"Failed to send analysis complete message to Spring backend. Status code: {response.statu
            logger.error(f"Response content: {response.text}")
            return False
    except requests.RequestException as e:
        logger.error(f"Error sending analysis complete message to Spring backend: {e}", exc_info=True)
        return False
```

면접 답변 분석

Google Cloud Storage Bucket 업로드 코드

```python
def upload_to_gcs(local_file_path: str, bucket_name: str, destination_blob_name: str):
    """
    Google Cloud Storage에 비디오 파일을 업로드하는 함수
    """
    credentials_path = os.path.join(os.path.dirname(__file__), 'credentials.json')
    storage_client = storage.Client.from_service_account_json(credentials_path)
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    # MIME 타입을 video/mp4로 설정하여 업로드
    blob.upload_from_filename(local_file_path, content_type='video/mp4')

    print(f"File {local_file_path} uploaded to {destination_blob_name}.")
    return f"https://storage.googleapis.com/{bucket_name}/{destination_blob_name}"

def save_analyzed_video(original_file_path, all_frames):
    """
    분석된 프레임을 하나의 비디오로 저장하고 GCS에 업로드하는 함수
    """
    if len(all_frames) == 0:
        print("No frames to save.")
        return None

    temp_dir = tempfile.gettempdir()
    timestamp = int(time.time())
    analyzed_file_path = os.path.join(temp_dir, f"analyzed_{timestamp}.mp4")

    try:
        height, width, _ = all_frames[0].shape
        fps = 30
        print(f"Video size: {width}x{height}, FPS: {fps}")

        frame_files = []
        for idx, frame in enumerate(all_frames):
            frame_file = os.path.join(temp_dir, f'frame_{idx:04d}.png')
            cv2.imwrite(frame_file, frame)
            frame_files.append(frame_file)

        ffmpeg_path = r'C:\Users\ict02-17\Desktop\ffmpeg-7.0.2-essentials_build\bin\ffmpeg.exe'
```

면접 분석 프레임 =>영상 변환 코드

```python
        ffmpeg_command = (
            f'"{ffmpeg_path}" -y -framerate {fps} -i "{temp_dir}/frame_%04d.png" '
            f'-c:v libx264 -pix_fmt yuv420p "{analyzed_file_path}"'
        )

        process = subprocess.Popen(ffmpeg_command, shell=True, cwd=temp_dir,
                                   stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout, stderr = process.communicate()

        if process.returncode != 0:
            print(f"FFmpeg stdout: {stdout.decode()}")
            print(f"FFmpeg stderr: {stderr.decode()}")
            raise Exception(f"FFmpeg command failed with return code {process.returncode}")

        if not os.path.exists(analyzed_file_path):
            raise FileNotFoundError(f"File {analyzed_file_path} was not created.")

        print(f"Analyzed video saved at {analyzed_file_path}")

        bucket_name = "cloud-storage-upload"
        destination_blob_name = f"analyzed_videos/analyzed_{timestamp}.mp4"
        gcs_url = upload_to_gcs(analyzed_file_path, bucket_name, destination_blob_name)

        os.remove(analyzed_file_path)

        for frame_file in frame_files:
            os.remove(frame_file)

        return gcs_url

    except Exception as e:
        print(f"Error saving analyzed video: {e}")
        return None
```

면접 답변 분석

면접 답변 분석 Json형식으로 저장 코드

```
async def analyze_interview_with_claude(transcription, question, company_name, answer_duration):

    keywords = extract_keywords(transcription, top_n=50)
    json_format = '''
    {
        "content_analysis": {
            "logic_score": 0,
            "key_ideas": "",
            "specific_examples": "",
            "consistency_score": 0
        },
        "sentiment_analysis": {
            "tone": "",
            "confidence_score": 0
        },
        "language_pattern_analysis": {
            "professional_vocab_score": 0,
            "repetitive_expressions": "",
            "grammar_structure_score": 0,
            "industry_specific_terms": []
        },
        "tone_tension_analysis": {
            "consistency_score": 0,
            "tension_detected": "",
            "hesitations": 0,
            "long_pauses": 0
        },
        "insight_analysis": {
            "creativity_score": 0,
            "problem_solving_score": 0
        },
        "star_analysis": {
            "situation_score": 0,
            "task_score": 0,
            "action_score": 0,
            "result_score": 0
        },
```

Claude API 분석 호출 및 프롬프트 코드

```
        "company_fit_analysis": {
            "alignment_score": 0,
            "company_knowledge": "",
            "industry_understanding": ""
        },
        "question_comprehension": {
            "relevance_score": 0,
            "missed_points": []
        },
        "answer_duration_analysis": {
            "duration": 0,
            "pace_score": 0,
            "comment": ""
        },
        "keywords": [],
        "overall_quality": 0,
        "improvement_suggestions": ""
    }
    '''

    try:
        message = await asyncio.to_thread(
            client.messages.create,
            model="claude-3-sonnet-20240229",
            max_tokens=2000,
            system="당신은 면접 답변을 분석하는 경험 많은 인사 담당자입니다. 분석 결과를 JSON 형식으로 제공해주세요.",
            messages=[
                {"role": "user", "content": f"""다음 면접 질문과 답변, 그리고 회사명을 주어진 기준에 따라 상세히 분석해주

질문: {question}
답변: {transcription}
회사명: {company_name}
답변 시간: {answer_duration}초

분석 기준:
1. 내용 분석:
   - 논리성 평가 (1-10점)
   - 핵심 아이디어 도출
   - 구체적인 예시 제공 여부
   - 답변의 일관성 (1-10점)
```

면접 답변 분석

면접 답변 음성 분석 코드(librosa 사용)

```python
def analyze_audio(audio_file_path):
    y, sr = librosa.load(audio_file_path)

    # 음정 분석
    pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
    pitches = pitches[pitches > 0]
    avg_pitch = np.mean(pitches) if len(pitches) > 0 else 0

    # 템포 분석
    onset_env = librosa.onset.onset_strength(y=y, sr=sr)
    tempo, _ = librosa.beat.beat_track(onset_envelope=onset_env, sr=sr)

    # 볼륨 분석 (RMS)
    rms = librosa.feature.rms(y=y)[0]
    avg_volume = np.mean(rms)

    # 스펙트럼 중심 분석
    spectral_centroids = librosa.feature.spectral_centroid(y=y, sr=sr)[0]
    avg_spectral_centroid = np.mean(spectral_centroids)

    # 분석 결과
    audio_analysis = {
        "average_pitch": avg_pitch,
        "tempo": tempo,
        "average_volume": avg_volume,
        "average_spectral_centroid": avg_spectral_centroid
    }

    # 피드백 메시지 생성 (위임)
    feedback_messages = generate_audio_feedback(audio_analysis)

    # 결과 반환
    return {
        "audio_analysis": audio_analysis,
        "feedback_messages": feedback_messages
    }
```

CLOVA Speech Recognition API 호출 코드

```python
def transcribe_with_clova(audio_file_path):
    headers = {
        'X-NCP-APIGW-API-KEY-ID': CLOVA_CLIENT_ID,
        'X-NCP-APIGW-API-KEY': CLOVA_CLIENT_SECRET,
        'Content-Type': 'application/octet-stream'
    }

    url_with_params = CLOVA_API_URL + '?lang=Kor&completion=sync'

    try:
        with open(audio_file_path, 'rb') as audio_file:
            audio_data = audio_file.read()

        response = requests.post(
            url_with_params,
            headers=headers,
            data=audio_data
        )

        print(f"Response Status Code: {response.status_code}")
        print(f"Response Headers: {response.headers}")
        print(f"Response Content: {response.content}")

        if response.status_code == 200:
            result = response.json()
            if 'text' in result:
                return result['text']
            else:
                print("No 'text' field in the response")
                return None
        else:
            print(f"Error: {response.status_code}, {response.text}")
            return None
    except Exception as e:
        print(f"Error during API call: {e}")
        return None
```

면접 답변 분석

CLOVA Sentiment API 호출 코드

```
def analyze_sentiment(text):
    headers = {
        'X-NCP-APIGW-API-KEY-ID': CLOVA_CLIENT_ID,
        'X-NCP-APIGW-API-KEY': CLOVA_CLIENT_SECRET,
        'Content-Type': 'application/json'
    }
    data = {
        'content': text
    }

    response = requests.post(SENTIMENT_API_URL, headers=headers, json=data)

    if response.status_code == 200:
        return response.json()
    else:
        print(f"Sentiment Analysis Error: {response.status_code}, {response.text}")
        return None

def summarize_text(text):
    if len(text.split()) < 10:  # 단어가 10개 미만인 경우
        print("Text too short for summarization")
        return {"summary": "Text too short for summarization"}

    headers = {
        'X-NCP-APIGW-API-KEY-ID': CLOVA_CLIENT_ID,
        'X-NCP-APIGW-API-KEY': CLOVA_CLIENT_SECRET,
        'Content-Type': 'application/json'
    }
    data = {
        'document': {
            'content': text
        },
        'option': {
            'language': 'ko',
            'model': 'news',
            'tone': 0,
            'summaryCount': 3
        }
    }
```

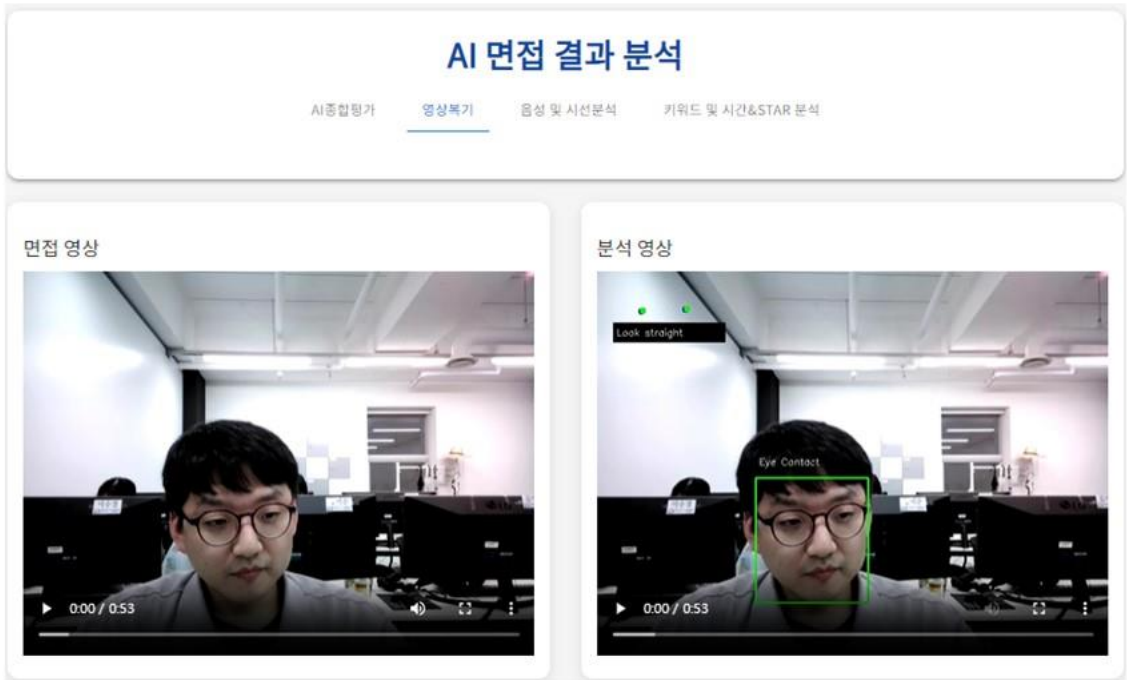
CLOVA Summary API 호출 코드

```
def summarize_text(text):
    if len(text.split()) < 10:  # 단어가 10개 미만인 경우
        print("Text too short for summarization")
        return {"summary": "Text too short for summarization"}

    headers = {
        'X-NCP-APIGW-API-KEY-ID': CLOVA_CLIENT_ID,
        'X-NCP-APIGW-API-KEY': CLOVA_CLIENT_SECRET,
        'Content-Type': 'application/json'
    }
    data = {
        'document': {
            'content': text
        },
        'option': {
            'language': 'ko',
            'model': 'news',
            'tone': 0,
            'summaryCount': 3
        }
    }

    try:
        response = requests.post(SUMMARY_API_URL, headers=headers, json=data)

        if response.status_code == 200:
            return response.json()
        else:
            print(f"Summary Error: {response.status_code}, {response.text}")
            return {"summary": "Summarization failed"}
    except requests.RequestException as e:
        print(f"Request failed: {e}")
        return {"summary": "Summarization request failed"}
```

면접 결과 화면(1)

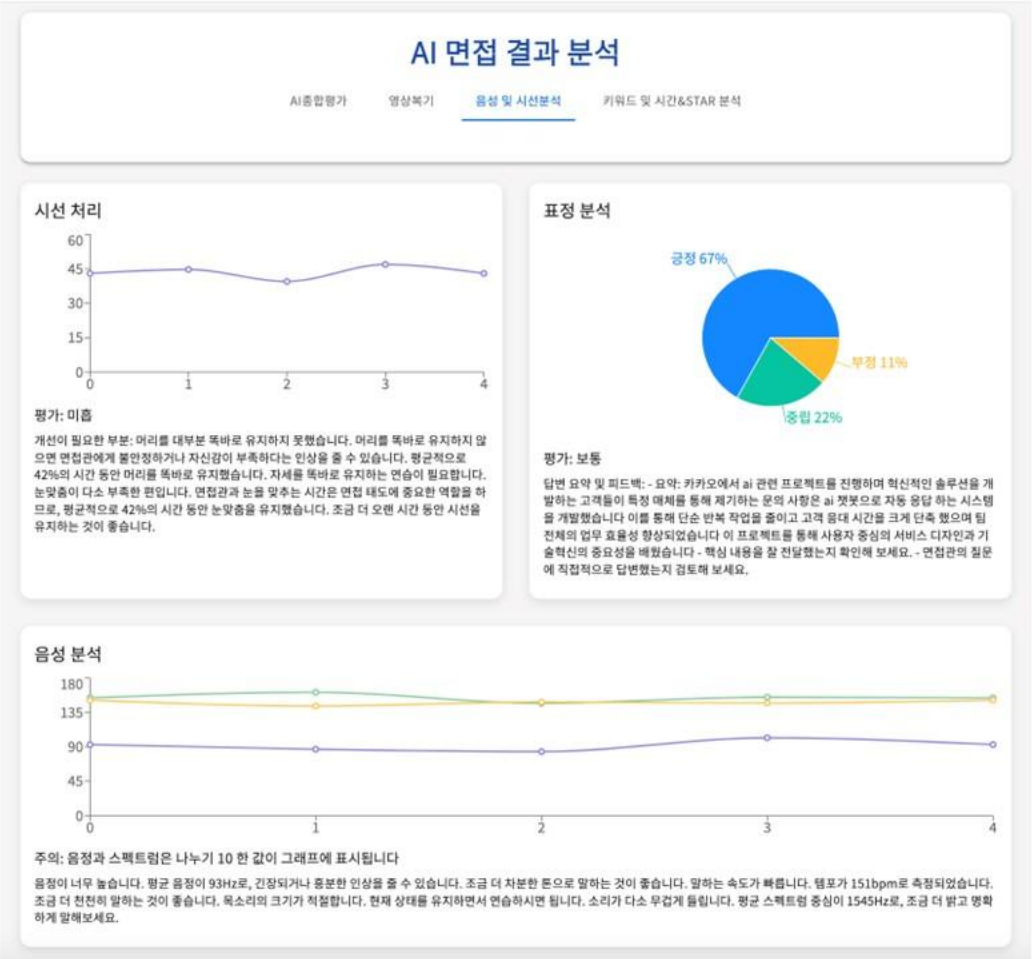
AI 종합평가

영상복기

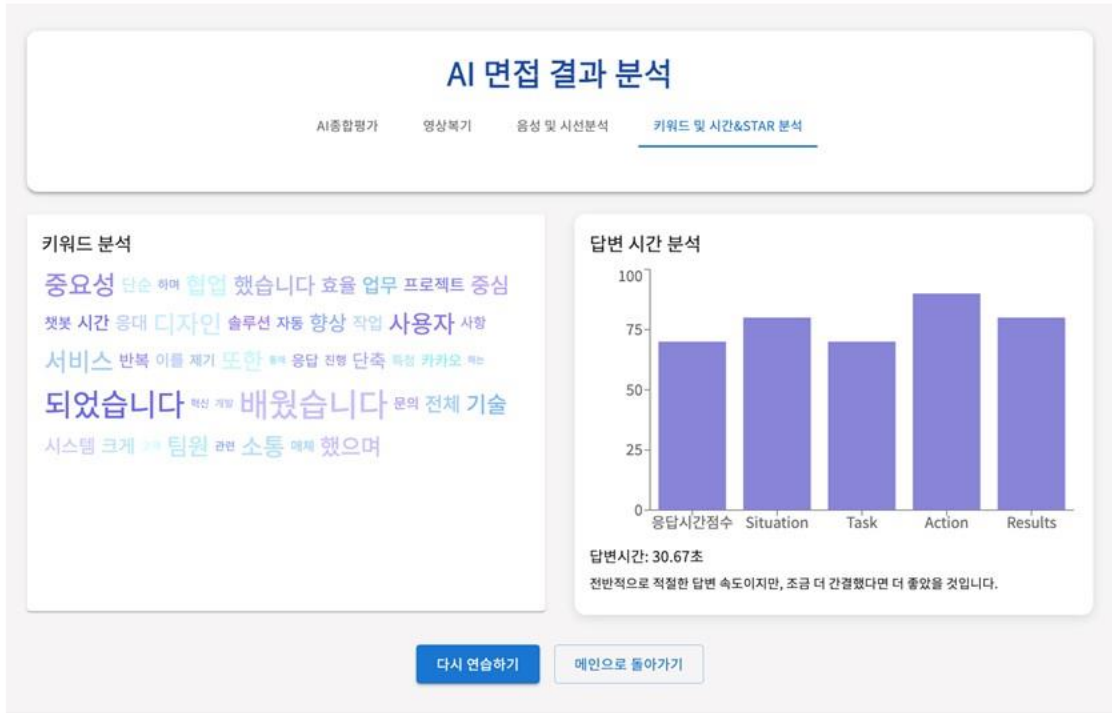


면접 결과 화면(2)

음성 및 시선분석



키워드 및 시간 & STAR 분석

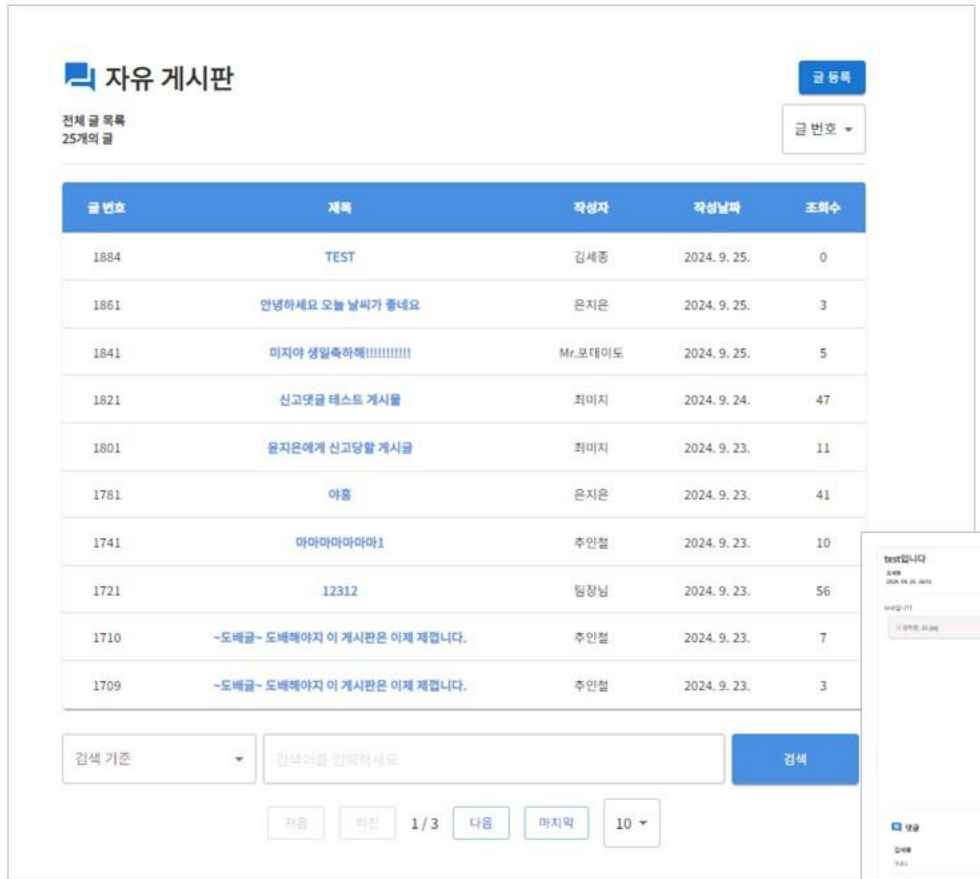


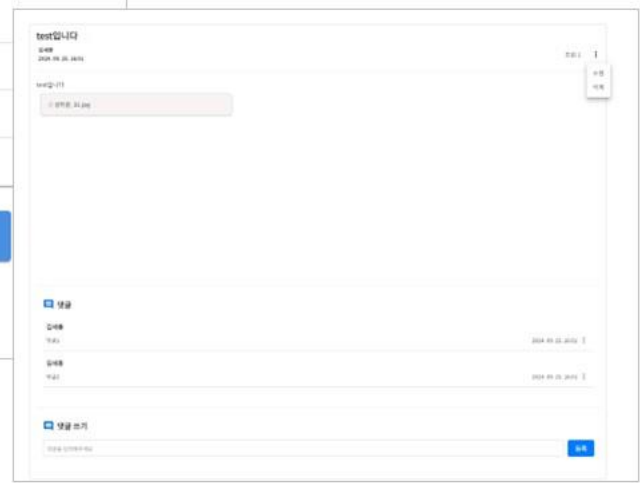
게시판 및 상세보기

```java
@Repository
public interface BbsRepository extends JpaRepository<Bbs, Long> {

    // 상태가 VISIBLE인 게시글만 조회
    List<Bbs> findByStatus(String status);

    // 삭제된 게시글만 조회
    List<Bbs> findByDeleted(int deleted);

    // 특정 사용자가 작성한 게시글 중 상태가 VISIBLE인 게시글 조회
    List<Bbs> findByUserIdAndStatus(User userId, String status);

    // 특정 사용자가 작성한 게시글 중 삭제 여부와 상태로 필터링된 게시글 조회
    List<Bbs> findByUserIdAndDeletedAndStatus(User userId, int deleted, String status);

    // 특정 사용자와 삭제된 게시글 조회
    List<Bbs> findByUserIdAndDeleted(User userId, int deleted);

    // 사용자와 status 필터로 게시글 조회
    List<Bbs> findByUserIdAndStatusAndActiveTrue(User userId, String status);

    // status가 NULL이거나 'VISIBLE'인 게시글 조회 (NULL 방지)
    @Query("SELECT b FROM Bbs b WHERE b.status IS NULL OR b.status = 'VISIBLE'")
    List<Bbs> findVisibleOrNullStatusPosts();

    List<Bbs> findByUserId_Id(Long userId);
}
```

DB에서 게시글 정보 조회

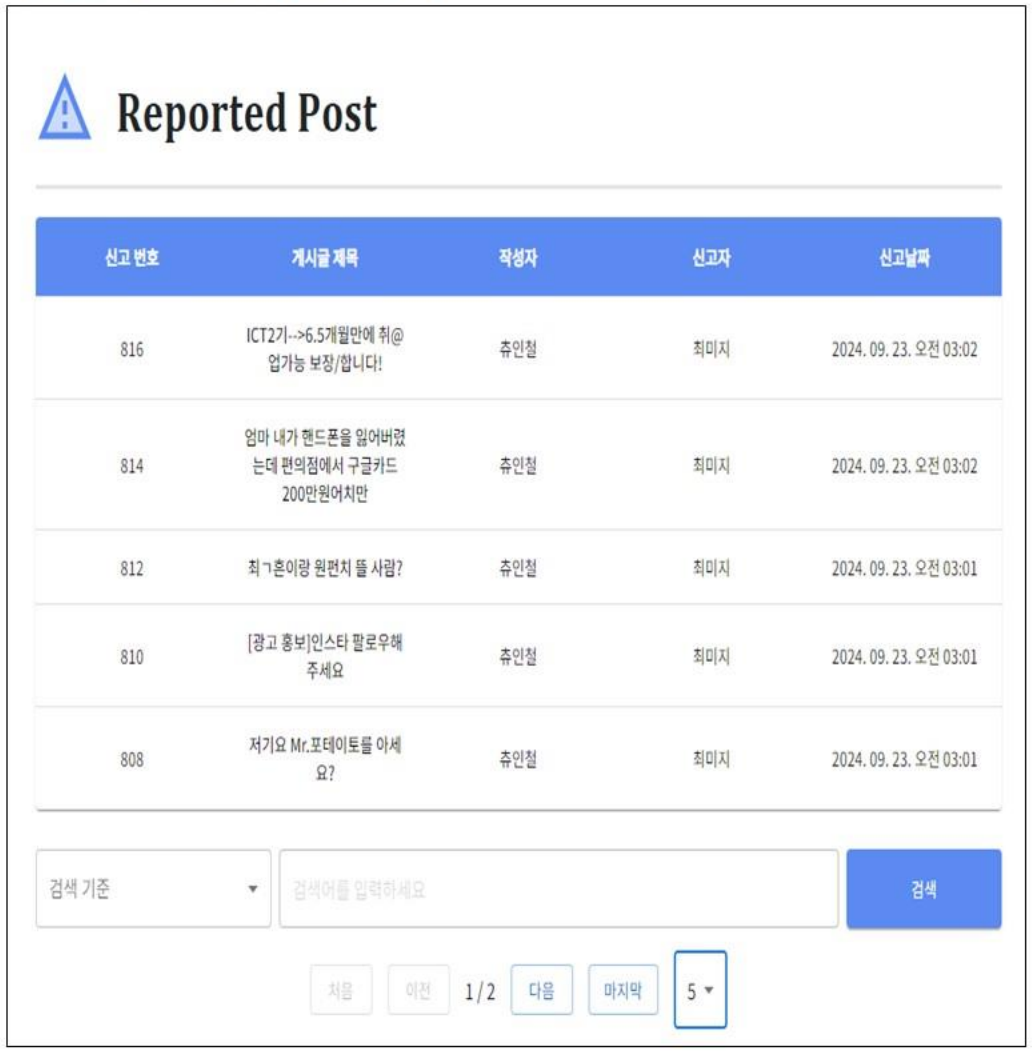
```java
//게시글 목록 조회
@GetMapping
public List<Bbs> getAllBbs() {
    List<Bbs> bbsList = bbsService.findAllByDeleted(0);  // 삭제되지 않은 글만 조회
    return bbsList;
}

// 게시글 단건 조회 (GET 요청)(원본)
@GetMapping("/{id}")
public ResponseEntity<Bbs> getBbsById(
    @PathVariable("id") Long id,
    @RequestParam Map<String, String> params){
    // 쿼리 파라미터 로그
    System.out.println("Query Params: " + params);

    boolean increment = Boolean.parseBoolean(params.getOrDefault("increment", "true"));
    System.out.println("Parsed increment value: " + increment);

    Optional<Bbs> bbsOptional = bbsService.findById(id);
    if (bbsOptional.isPresent()) {
        Bbs bbs = bbsOptional.get();
        if (increment) {
            bbsService.incrementHitcount(id); // 조회수 증가
        }


        return ResponseEntity.ok().body(bbs);
    } else {
        return ResponseEntity.notFound().build();
    }
}
```





게시글 목록 호출과 게시글 상세정보 호출

신고된 게시글 관리



신고 접수된 게시글 테이블


```jsx
// 신고된 댓글 처리 추가
const CommentItem = ({ comment, setComments }) => {
  const [isEditing, setIsEditing] = useState(false);
  const [newContent, setNewContent] = useState(comment.content);
  const [anchorEl, setAnchorEl] = useState(null);
  const [isReportModalOpen, setReportModalOpen] = useState(false);

  const commentOwnerId = Number(comment.user?.id) || 0;
  const currentUserId = Number(userStore.id) || 0;

  // 신고된 댓글에 대한 처리
  if (comment.deletedReason === 1) {
    return (
      <div className={styles.reportedCommentContainer}>
        <span className={styles.commentIcon}>⚠</span>
        신고 접수된 댓글, 현재 관리자의 검토가 진행 중입니다.
      </div>
    );
  }
```

```jsx
// 게시글 본문 컴포넌트
const PostContent = ({ post, openReportModal }) => {
  const router = useRouter();
  const { id } = router.query;
  const [anchorEl, setAnchorEl] = useState(null);
  const postOwnerId = Number(post.userId?.id) || 0;
  const currentUserId = Number(userStore.id) || 0;

  // 신고된 게시글 처리
  if (post.deletedReason === 1) {
    return (
      <div className={styles.reportedContainer}>
        <span className={styles.reportedIcon}>⚠</span>
        신고 접수된 게시글, 현재 관리자의 검토가 진행 중입니다.
      </div>
    );
  }
```

신고 처리 .jsx 코드

신고된 게시글 관리

신고된 게시글 삭제,복구 기능

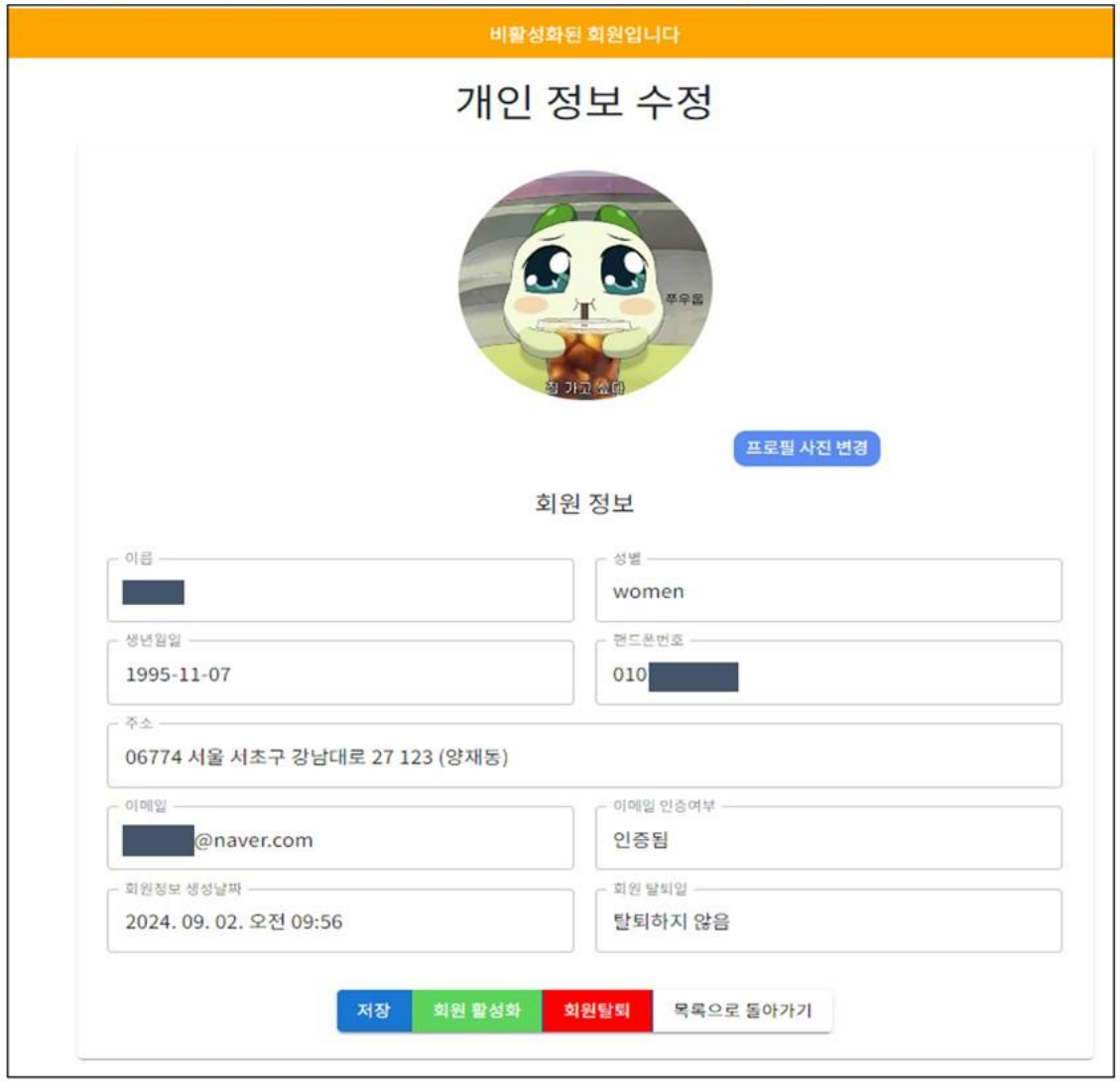
```jsx
// 게시글 복구하는 함수
const handleRestore = () => {
    if (typeof window !== 'undefined') {
        if (window.confirm("게시글을 [신고->일반] 게시글로 복구하시겠습니까?")) {
            axios.put(`http://localhost:8080/api/adminreported/updatestatus/${reportId}/VISIBLE`)
                .then(() => {
                    alert('게시글이 복구되었습니다.');
                    router.push('/adminPage/adminReportedPostPage');
                })
                .catch(error => {
                    console.error('Error restoring post:', error);
                    alert('게시글 복구 중 오류가 발생했습니다.');
                });
        }
    }
};
```

신고 게시글 복구.jsx 코드

신고 게시글 삭제.jsx 코드

```jsx
// 게시글 삭제하는 함수
const handleDelete = () => {
    if (typeof window !== 'undefined') {
        if (window.confirm("게시글을 완전히 삭제하시겠습니까? 영구 삭제 시, 복구가 어렵습니다")) {
            axios.delete(`http://localhost:8080/api/adminreported/delete/${reportId}`)
                .then(() => {
                    alert("게시글 삭제가 완료되었습니다.");
                    router.push('/adminPage/adminReportedPostPage');
                })
                .catch(error => {
                    console.error('Error deleting post:', error);
                    alert('삭제 중 오류가 발생했습니다.');
                });
        }
    }
};
```

회원(조회, 활성화/비활성화, 탈퇴) 관리



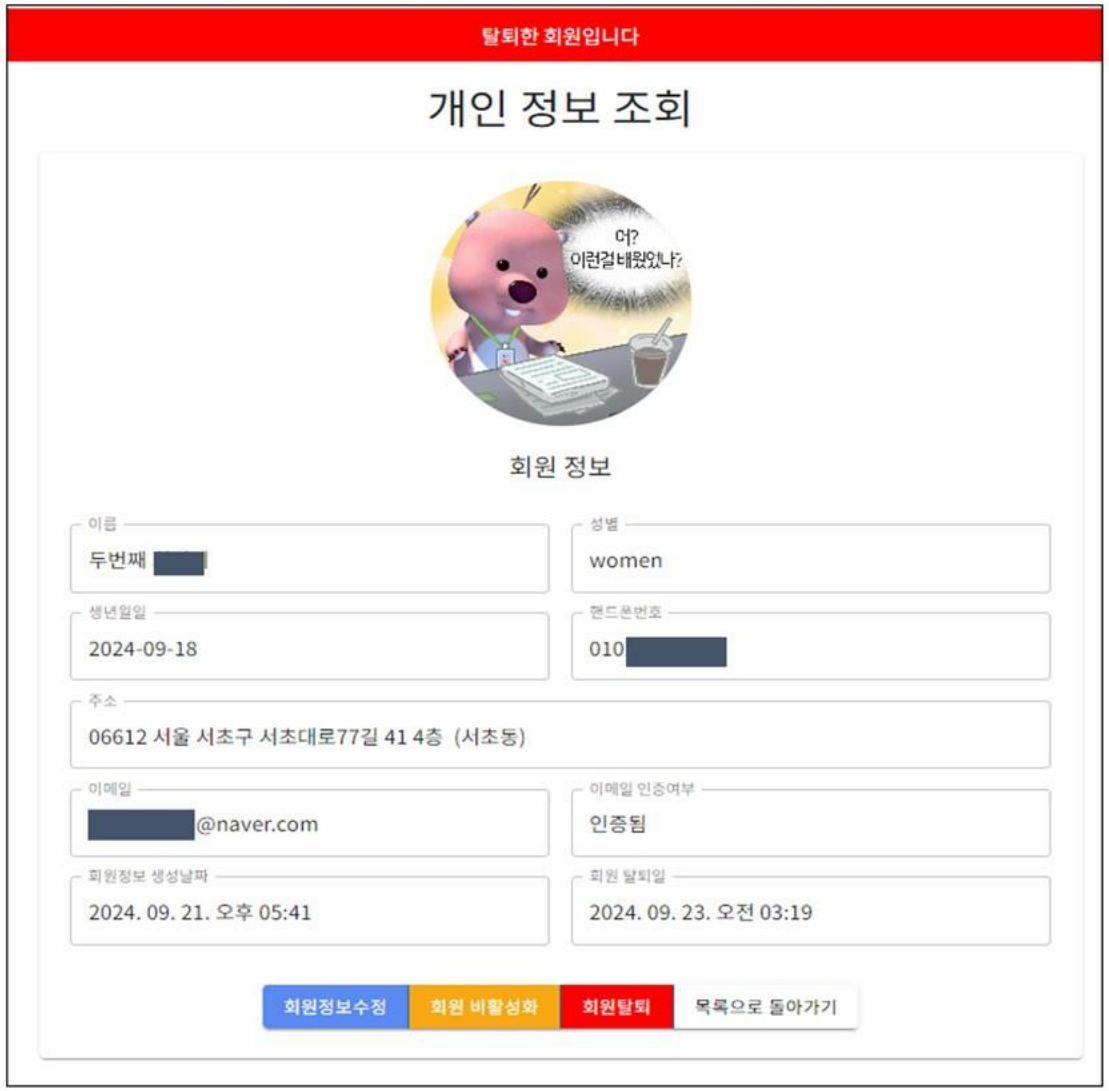
유저 상세보기,활성화/비활성화 기능

```jsx
// 사용자 비활성화 핸들러
const handleDeactivateClick = async () => {
  try {
    const response = await axios.post('http://localhost:8080/api/auth/deactivateUser', null, {
      params: { email: user.email },
    });
    alert(response.data);
    await viewUserStore.fetchUserByEmail(user.email);
  } catch (error) {
    console.error('사용자 비활성화 중 오류가 발생했습니다:', error);
    alert('비활성화 요청 중 오류가 발생했습니다.');
  }
};

// 사용자 비활성화 -> 활성화 핸들러
const handleActivateClick = async () => {
  try {
    const response = await axios.post('http://localhost:8080/api/auth/activateUser', null, {
      params: { email: user.email },
    });
    alert(response.data);
    await viewUserStore.fetchUserByEmail(user.email);
  } catch (error) {
    console.error('사용자 활성화 중 오류가 발생했습니다:', error);
    alert('활성화 요청 중 오류가 발생했습니다.');
  }
};
```

```jsx
{isActivated === 0 ? (
  <Button onClick={handleActivateClick} className={styles.userDetailsActivateButton}>
    회원 활성화
  </Button>
) : (
  <Button onClick={handleDeactivateClick} className={styles.userDetailsDeactivateButton}>
    회원 비활성화
  </Button>
)}
```

활성화,비활성화 기능.jsx 코드

회원(조회, 활성화/비활성화, 탈퇴) 관리



유저 상세보기,탈퇴 기능

탈퇴 후 로그인 화면

```jsx
// 탈퇴 핸들러
const handleDeleteClick = async () => {
  try {
    const response = await axios.post('http://localhost:8080/api/auth/deleteUser', null, {
      params: { email: user.email },
    });
    alert(response.data);
    router.push('/adminPage/adminUserPage');
  } catch (error) {
    console.error('사용자 탈퇴 중 오류가 발생했습니다:', error);
    alert('탈퇴 요청 중 오류가 발생했습니다.');
  }
};
```

회원 탈퇴기능.jsx 코드

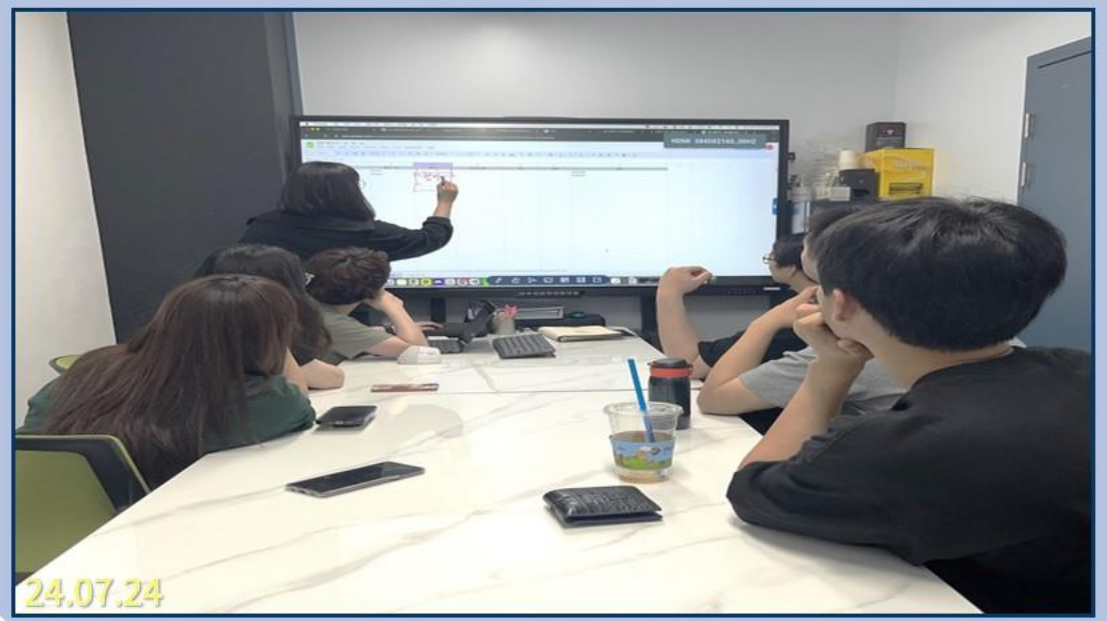
06

프로젝트 시연

07

후기


24.08.01

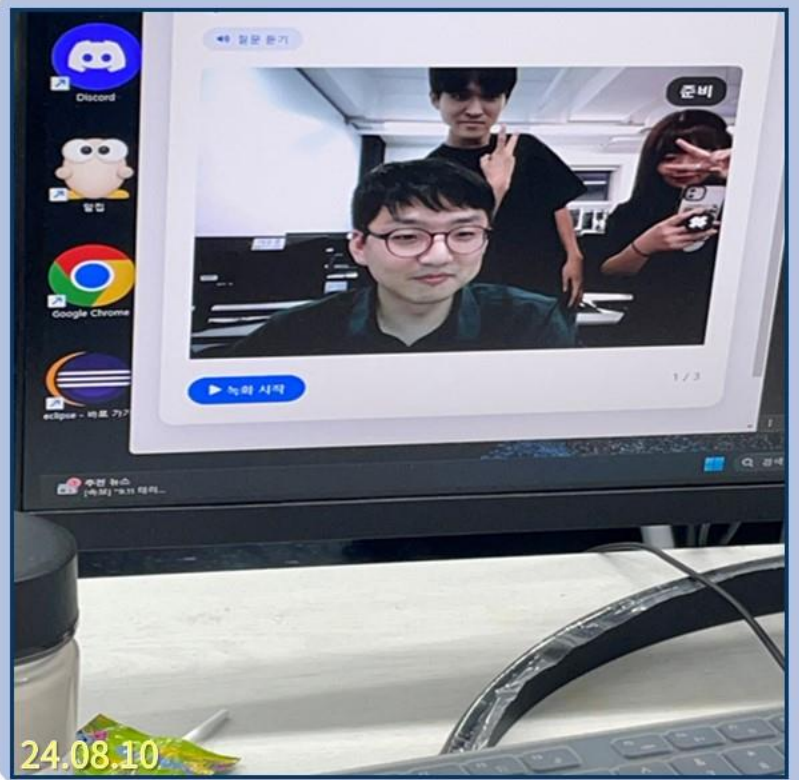

24.07.24

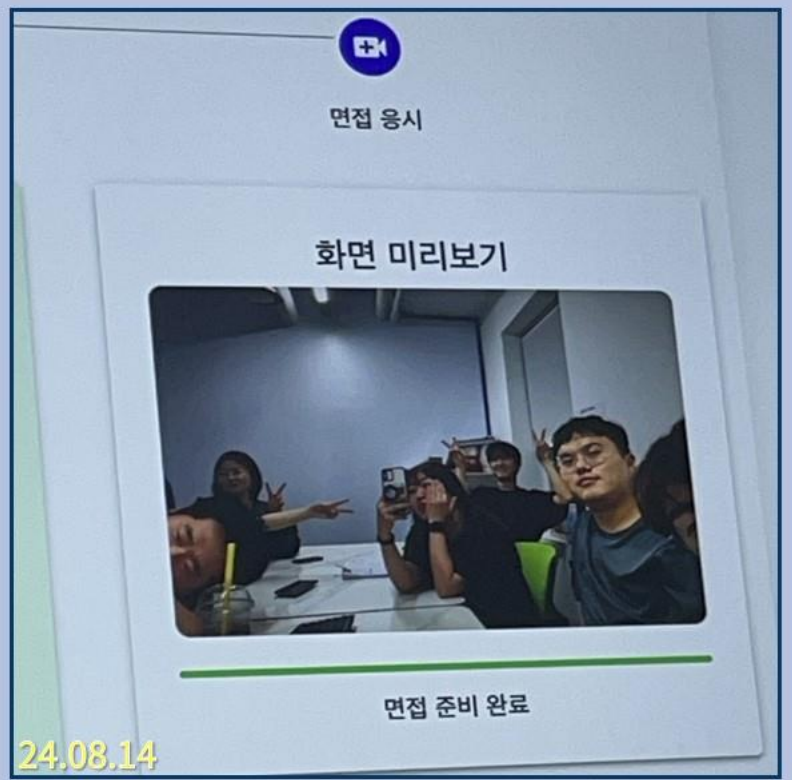

24.09.10

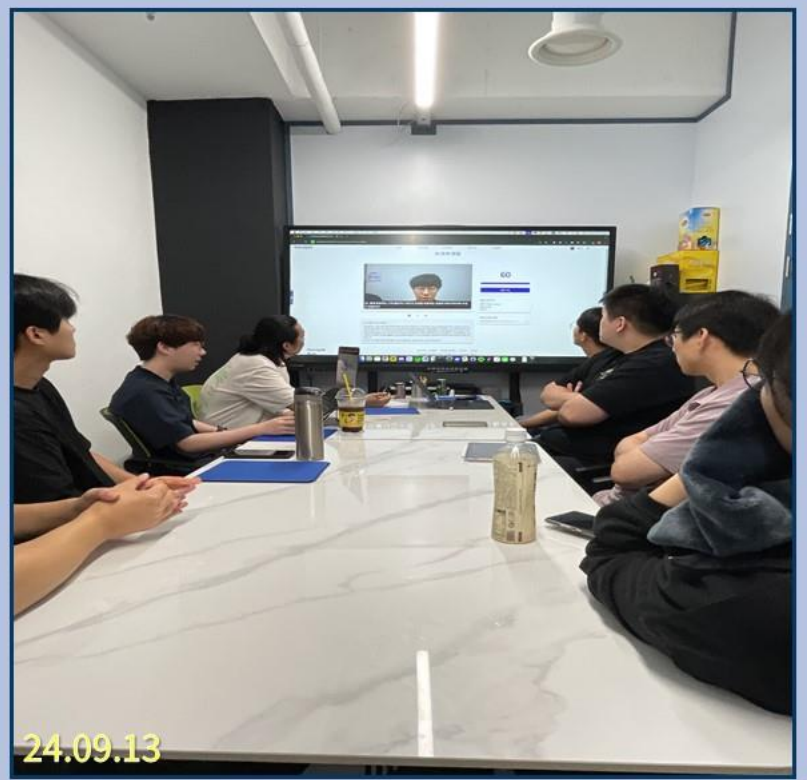

24.08.18


24.09.12

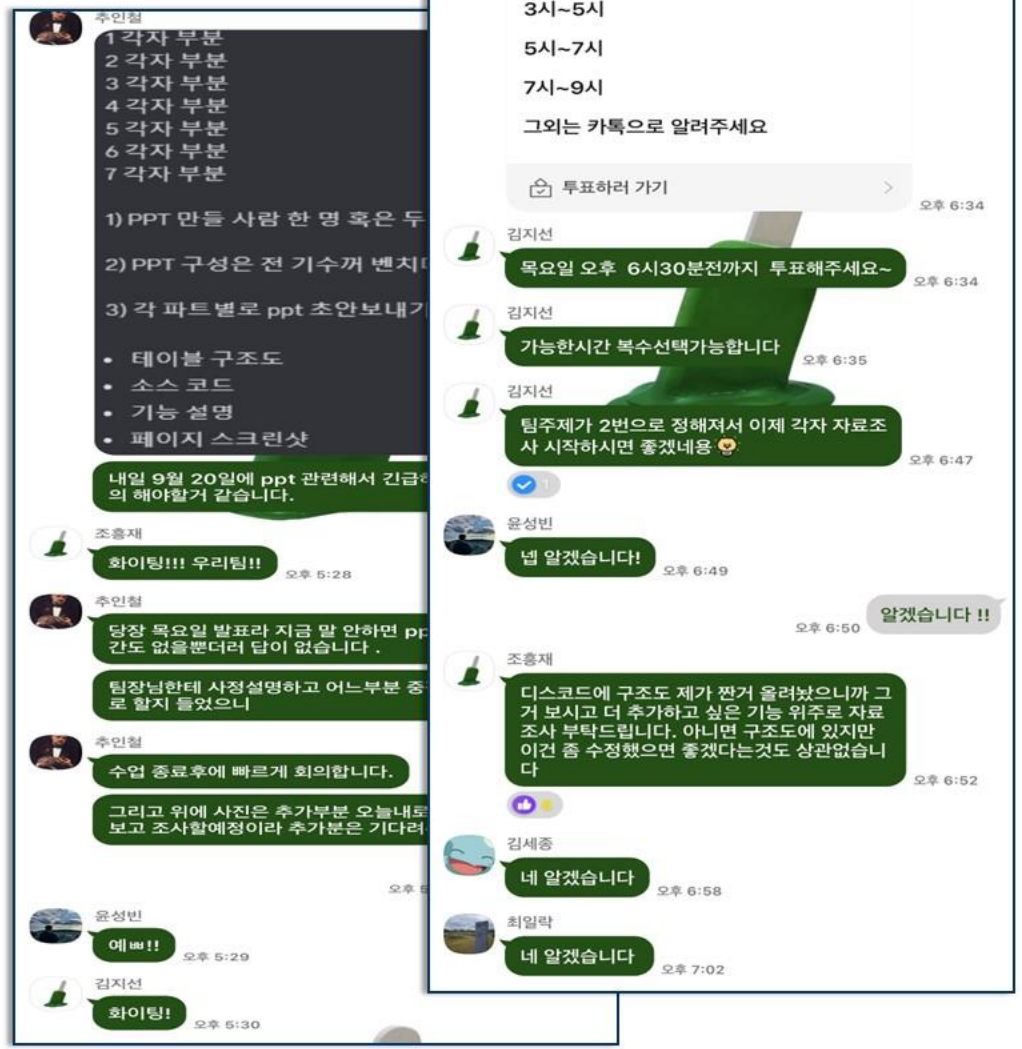

24.08.15

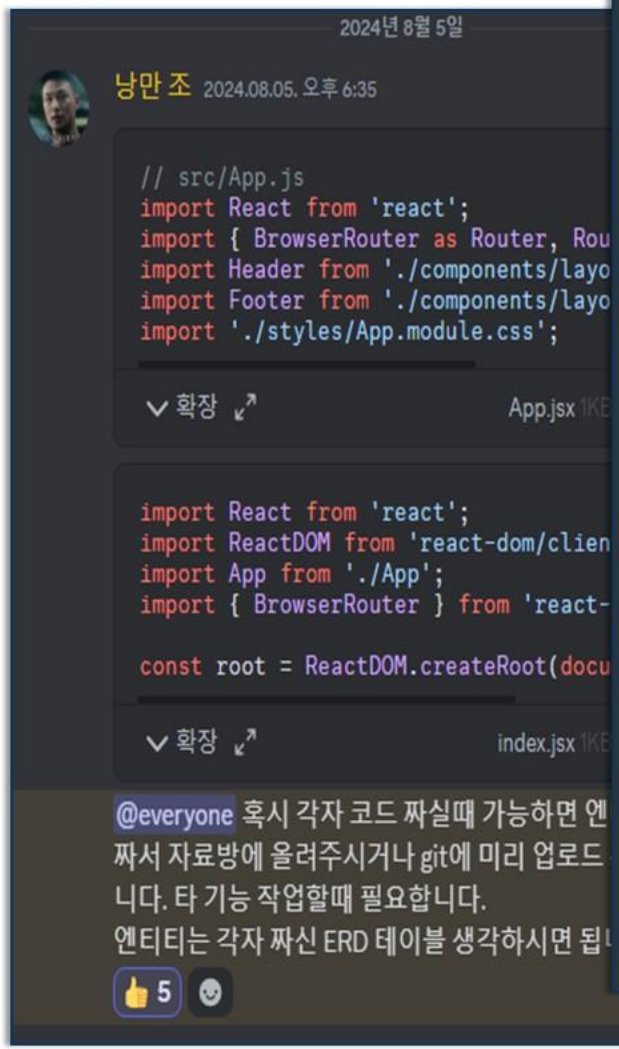

Discord
Google Chrome
준비
녹화 시작
24.08.10

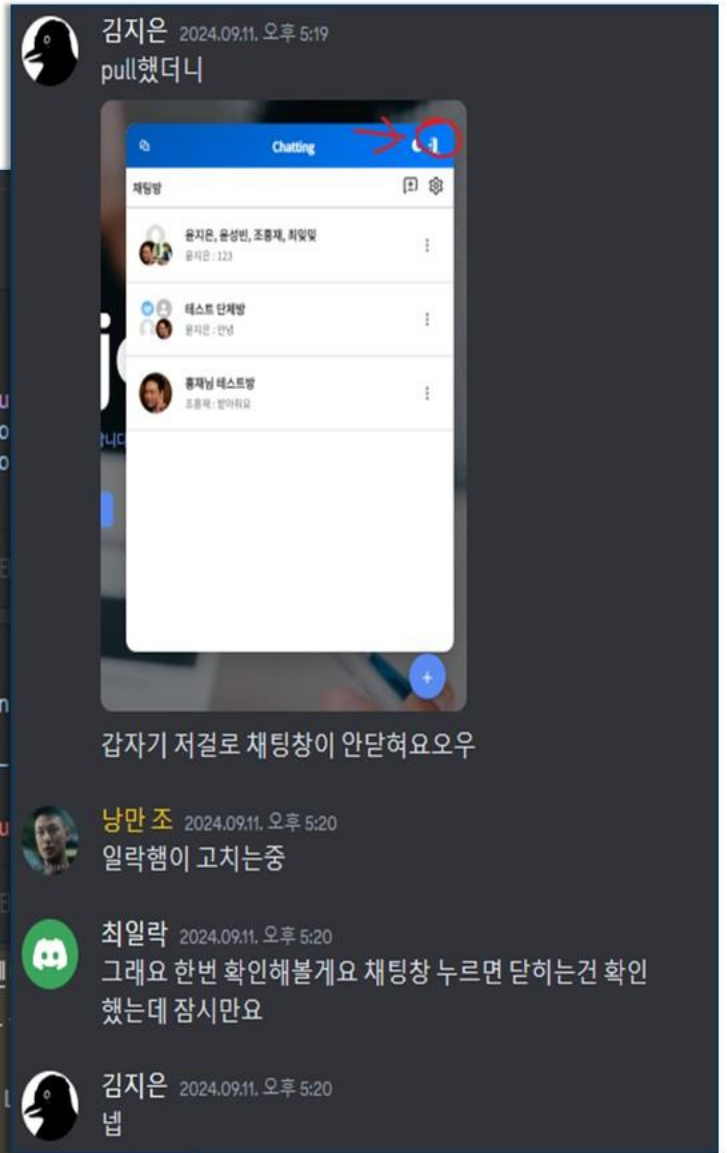

면접 응시
화면 미리보기
면접 준비 완료
24.08.14


24.09.13

TALK


추인철
1 각자 부분
2 각자 부분
3 각자 부분
4 각자 부분
5 각자 부분
6 각자 부분
7 각자 부분
1) PPT 만들 사람 한 명 혹은 두
2) PPT 구성은 전 기수꺼 벤치
3) 각 파트별로 ppt 초안보내기
테이블 구조도
소스 코드
기능 설명
페이지 스크린샷
내일 9월 20일에 ppt 관련해서 긴급
의 해야할거 같습니다.
조흥재
화이팅!!! 우리팀!!
오후 5:28
추인철
당장 목요일 발표라 지금 말 안하면 pp
간도 없을뿐더러 답이 없습니다 .
팀장님한테 사정설명하고 어느부분 중
로 할지 들었으니
추인철
수업 종료후에 빠르게 회의합니다.
그리고 위에 사진은 추가부분 오늘내로
보고 조사할예정이라 추가분은 기다려
오후 5
윤성빈
에뻐!!
오후 5:29
김지선
화이팅!
오후 5:30
윤지은
윤지은님의 글을 공지로 등록하였습니다.
Q. 정기회의 시간(일요일)
1시~3시
3시~5시
5시~7시
7시~9시
그외는 카톡으로 알려주세요
투표하러 가기
오후 6:34
김지선
목요일 오후 6시30분전까지 투표해주세요~
오후 6:34
김지선
가능한시간 복수선택가능합니다
오후 6:35
김지선
팀주제가 2번으로 정해져서 이제 각자 자료조
사 시작하시면 좋겠네용
오후 6:47
윤성빈
넵 알겠습니다!
오후 6:49
알겠습니다 !!
오후 6:50
조흥재
디스코드에 구조도 제가 짠거 올려놨으니까 그
거 보시고 더 추가하고 싶은 기능 위주로 자료
조사 부탁드립니다. 아니면 구조도에 있지만
이건 좀 수정했으면 좋겠다는것도 상관없습니
다
오후 6:52
김세종
네 알겠습니다
오후 6:58
최일락
네 알겠습니다
오후 7:02


2024년 8월 5일
낭만 조 2024.08.05. 오후 6:35
// src/App.js
import React from 'react';
import { BrowserRouter as Router, Rou
import Header from './components/layo
import Footer from './components/layo
import './styles/App.module.css';
확장
App.jsx
import React from 'react';
import ReactDOM from 'react-dom/clien
import App from './App';
import { BrowserRouter } from 'react-
const root = ReactDOM.createRoot(docu
확장
index.jsx
@everyone 혹시 각자 코드 짜실때 가능하면 엔
짜서 자료방에 올려주시거나 git에 미리 업로드
니다. 타 기능 작업할때 필요합니다.
엔티티는 각자 짜신 ERD 테이블 생각하시면 됩
5


김지은 2024.09.11. 오후 5:19
pull했더니
Chatting
채팅방
윤지은, 윤성빈, 조흥재, 최일락
윤지은 : 123
테스트 단체방
윤지은 : 안녕
흥재님 테스트방
조흥재 : 받아줘요
갑자기 저걸로 채팅창이 안닫혀요오우
낭만 조 2024.09.11. 오후 5:20
일락햄이 고치는중
최일락 2024.09.11. 오후 5:20
그래요 한번 확인해볼게요 채팅창 누르면 닫히는건 확인
했는데 잠시만요
김지은 2024.09.11. 오후 5:20
넵

컨벤션

Git

- commit 컨벤션
  - `feat` : 새로운 기능 추가
  - `fix` : 버그 수정
  - `docs` : 문서 수정
  - `style` : 코드 포맷팅, 세미콜론 누락, 코드 변경이 없는 경
  - `refactor` : 코드 리팩토링
  - `test` : 테스트 코드, 리팩토링 테스트 코드 추가
  - `chore` : 빌드 업무 수정, 패키지 매니저 수정

feat: 아이디,비번 기능 구현
날짜: 2024.07.18

배포 과정에서 핫픽스 때는 컨벤션 설정 필요

2024.07.07 회의

https://www.youtube.com/@user-be1bh9zt2o

- 최일락 - AI 모의 면접 항목
  - KoGPT2 SKT-AI
    - 넣으면 좋은 단어, 빼면 좋은 단어 등
  - GitHub - d102-preview/preview: AI 기반 면접 연습 웹 서
  - 상 수상
  - 면접 예상 공통 질문.docx 18.0KB
- 김지선 - AI 모의 면접 평가
  - 2팀 자료조사_김지선.pdf 1033.3KB
- 윤성빈 - 이력서 첨삭
  - https://www.notion.so/AI-dcb206e13e2242f9b99f5bb74
  - 이력서(자소서) AI 관련 자료조사
    1. **AI 기반 텍스트 분석 API**: OpenAI의 GPT-3.5 언어모델
       - 대부분의 이력서 AI 첨삭 사이트에서 gpt 모델을 사
       - **과정**
    2. 작성한 자기 소개서에 대한 표절 탐지를 어떻게 할 것
       - small seo Tools / Duplichecker 등 api를 찾아봤으
       - 위의 gpt의 언어모델로는 표절 탐지는 불가
- 최미지 - 소셜 로그인
  - https://developers.kakao.com/docs/latest/ko/kakaolog
    - //참고 사이트 https://innovation123.tistory.com/181

2024.08.12 임시회의

2024.08.14 회의전까지 완성

- 김세종
  - UI git 업로드
  - 글 조회, 등록, 수정, 삭제 기능 구현 (모든 게시판) - 1
- 조흥재
  - <Component {...pageProps} authStore={authStore}
- 김지선
  - 면접 결과 처리 UI 끝 20%
  - ~~설문조사 UI~~
  - ~~설문조사 안스를~~
  - 메인화면~~, 헤더 마우스 오버~~
- 최일락
  - 면접 UI 완성 95%
  - ~~챗봇 UI 완성~~
  - 임시로 전체 UI 확인하는 링크나 버튼(기능 제거) 90%
  - 주가로 데이터 학습
- 추인철
  - 상세 UI 완성 95%(1차 후)
- 윤성빈
  - ~~이력서 첨삭 UI 완성~~
  - API 1번 연결 70%

2팀: AI 면접평가 서비스

팀명: FocusJob
기간: 2024.06.24 - 2024.09.25

주요 기능

AI 기반 면접

- 관리자가 설정한 질문을 토대로 사용자가 면접 후 AI가 평가
- 면접 답변 내용 기반 AI 꼬리 질문 생성
- 사용자가 입력 혹은 등록한 이력서 기반 질문 생성

이력서 첨삭

- 이력서 작성
- 이력서 AI 맞춤법 검사

회사 상세 검색

- 카카오 맵 API를 기반으로 사용자가 검색한 회사 표시
- 상세조건 설정을 통해 지도 효율 극대화
- 사이드바에 목록 형태로 검색결과 간략히 표현

실시간 채팅

- 카카오톡 1:1 채팅, 그룹채팅과 비슷한 기능
- 채팅이 오면 오른쪽위 종 모양 아이콘에 알림

우리 서비스를 써야하는 이유

- 면접에 대한 경험이 적은 사용자를 위한 AI 면접으로 부담감
- 사용자의 답변을 바탕으로 면접의 질문은 어떠한 식으로 이

팀원 소개


조흥재
총괄
팀장
기술승배 갓철환


김지선
부팀장


윤지온
팀원

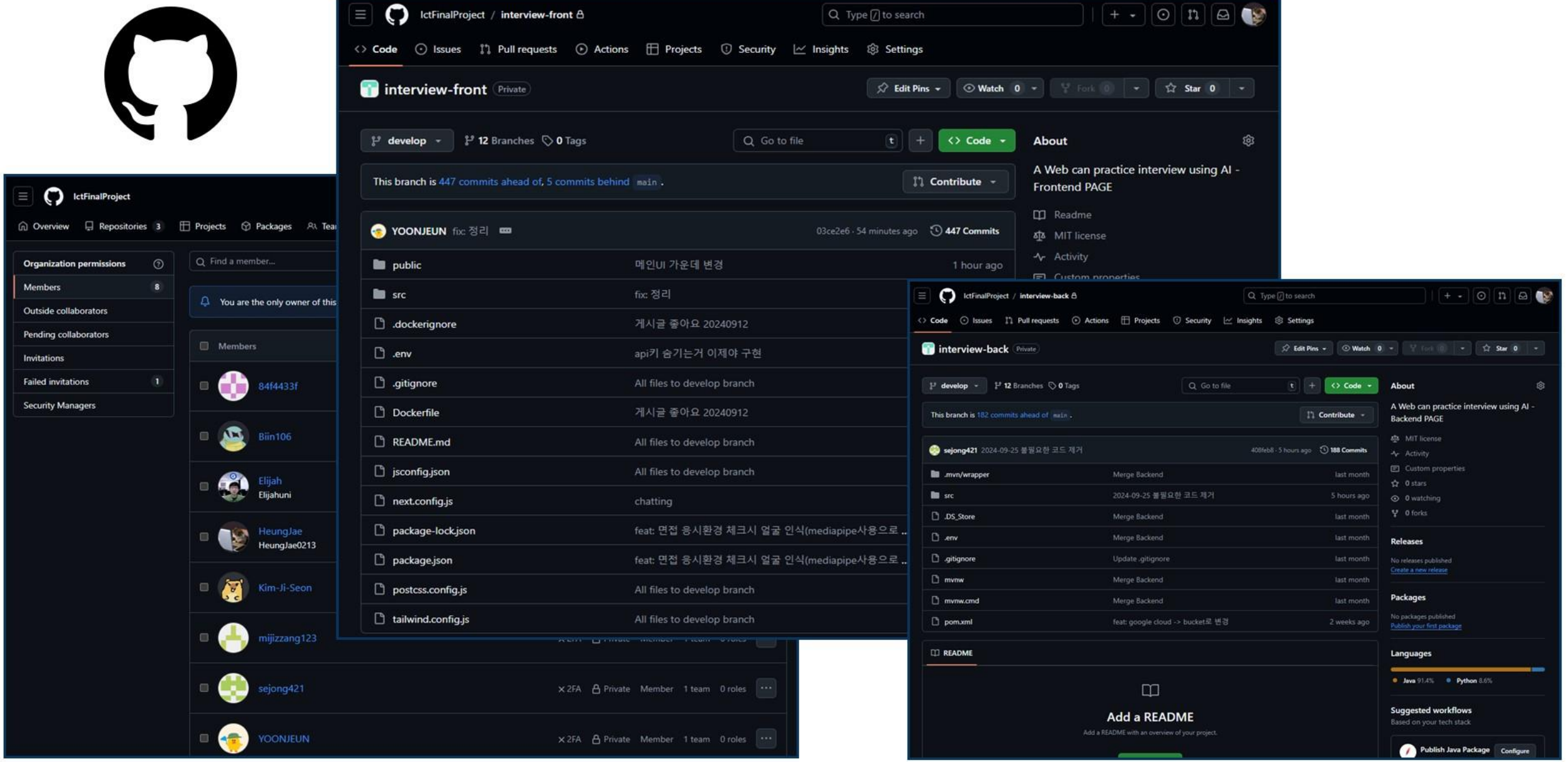

윤성빈
팀원


최미지
팀원
1


최일락
팀원


추인철
팀원


김세종
팀원


IctFinalProject / interview-front
Code Issues Pull requests Actions Projects Security Insights Settings
interview-front Private
develop 12 Branches 0 Tags
This branch is 447 commits ahead of, 5 commits behind main.
YOONJEUN fix: 정리 03ce2e6 · 54 minutes ago 447 Commits
public 메인UI 가운데 변경 1 hour ago
src fix: 정리
.dockerignore 게시글 좋아요 20240912
.env api키 숨기는거 이제야 구현
.gitignore All files to develop branch
Dockerfile 게시글 좋아요 20240912
README.md All files to develop branch
jsconfig.json All files to develop branch
next.config.js chatting
package-lock.json feat: 면접 응시환경 체크시 얼굴 인식(mediapipe사용으로 ..
package.json feat: 면접 응시환경 체크시 얼굴 인식(mediapipe사용으로 ..
postcss.config.js All files to develop branch
tailwind.config.js All files to develop branch
About
A Web can practice interview using AI - Frontend PAGE
Readme
MIT license
Activity
IctFinalProject
Overview Repositories 3 Projects Packages
Organization permissions
Members 8
Outside collaborators
Pending collaborators
Invitations
Failed invitations 1
Security Managers
Members
84f4433f
Biin106
Elijah Elijahuni
HeungJae HeungJae0213
Kim-Ji-Seon
mijizzang123
sejong421 2FA Private Member 1 team 0 roles
YOONJEUN 2FA Private Member 1 team 0 roles
IctFinalProject / interview-back
interview-back Private
develop 12 Branches 0 Tags
This branch is 182 commits ahead of main.
sejong421 2024-09-25 불필요한 코드 제거 408feb8 · 5 hours ago 188 Commits
.mvn/wrapper Merge Backend last month
src 2024-09-25 불필요한 코드 제거 5 hours ago
.DS_Store Merge Backend last month
.env Merge Backend last month
.gitignore Update .gitignore last month
mvnw Merge Backend last month
mvnw.cmd Merge Backend last month
pom.xml feat: google cloud -> bucket로 변경 2 weeks ago
README
Add a README
Add a README with an overview of your project.
About
A Web can practice interview using AI - Backend PAGE
MIT license
Activity
Custom properties
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Java 91.4% Python 8.6%
Suggested workflows
Based on your tech stack
Publish Java Package Configure

조흥재

팀 프로젝트를 진행하면서 커뮤니케이션의 중요성을 절실히 깨달았습니다. 메소드나 패키지 명을 선언할 때, DB 컬럼명이나 테이블명을 정할 때 등 각 팀원이 선호하는 코드 스타일과 방식의 차이로 인해 일관성이 떨어지는 경우가 있었습니다. 이는 프로그램에 오류가 발생했을 때 가독성을 해치며 에러를 빠르게 찾아내지 못하는 문제로 이어졌습니다. 결국, 팀원들과 컨벤션을 정해 규칙을 마련함으로써 이러한 문제를 해결할 수 있었습니다. 이 경험을 통해 팀 프로젝트에서는 개인의 역량뿐만 아니라 팀원들과의 원활한 소통과 협력이 무엇보다 중요하다는 것을 깨달았습니다. 앞으로 이러한 경험을 바탕으로, 기술적 역량을 쌓고 팀워크를 강화해 나가며, 미래에 더욱 발전하는 개발자로 성장하고 싶습니다.

김지선

React를 활용해 프로젝트의 프론트엔드 화면을 구성하고, 커리어넷 API를 이용한 적성탐색 기능을 구현하면서 수업 때 배웠던 내용을 다시 한 번 상기하며 실전에 적용할 수 있었습니다. 이를 통해 배운 이론을 실제 프로젝트에 자연스럽게 접목하며 실무적인 감각을 키울 수 있었습니다. 이번 프로젝트는 단순한 개발을 넘어, 협업을 통해 함께 배우고 성장하는 것이 얼마나 중요한지 깨닫게 해주었고, 그 과정에서 저뿐만 아니라 팀원들도 함께 성장할 수 있었습니다. 덕분에 프로젝트의 완성도를 한층 더 높일 수 있었습니다.

추인철

프로젝트를 진행하면서 각자의 역할 분배와 원활한 소통을 통해 프로젝트의 방향성을 잡아갔습니다. 팀원들과 협력하며 서로의 의견을 존중하고, 공동의 목표를 달성하기 위해 함께 노력한 것이 큰 도움이 되었습니다. 이번 프로젝트를 통해 개발자로서뿐만 아니라 협업자로서도 많은 성장을 이뤘습니다. 어려움이 생길 때마다 새로운 시도를 통해 해결책을 모색하고, 이를 통해 기술적인 역량뿐 아니라 협업 능력도 크게 향상되었습니다. 앞으로도 이러한 경험을 바탕으로 팀과 함께 더 나은 결과를 만들어내며 지속적으로 성장해 나가고자 합니다. 이번 프로젝트는 저에게 한 단계 더 발전할 수 있는 중요한 기회였으며, 앞으로도 더 큰 성과를 이루기 위해 노력하겠습니다.

최일락

프로젝트를 진행하며 면접 데이터를 분석하고 질문을 최적화하는 과정에서 많은 기술을 배우고 성장할 수 있었습니다. 특히 OpenAI API를 활용한 지능형 챗봇 구현에서는 큰 문제 없이 성능을 최적화할 수 있었고, 파인 튜닝과 사용자 인터페이스 개선을 통해 사용자 경험을 크게 향상시켰습니다. 팀원들과의 협업 덕분에 다양한 아이디어를 나누고 함께 문제를 해결하는 긍정적인 경험을 쌓을 수 있었습니다. YOLO, ResNet, KoBERT 등의 기술을 학습하고 실제 시스템에 적용하려 했지만, 시간이 부족해 아쉬움이 남았고, 데이터를 충분히 수집하지 못해 챗봇의 파인 튜닝 성능에 한계가 있었던 점은 다음 프로젝트에 대한 동기부여가 되었습니다. 앞으로 해결해야 할 도전들이 많지만, 이를 보완할 계획이며, 추후 플랫폼 배포도 진행할 예정입니다.

윤지은

팀 프로젝트를 통해 팀플은 최종 결과물 뿐만 아니라 팀워크 및 협력이 중요하다는 것을 깊이 깨닫게 되었습니다. 각자의 역할을 넘어서 동료를 돕는 것이 팀의 성공에 얼마나 중요한지 실감했습니다. 소통과 협력, 의견 조율의 과정은 팀워크의 핵심이었으며, 이를 통해 많은 성장을 이뤘습니다. 또한, 새롭거나 익숙하지 않은 다양한 기술을 다루면서 기술적으로도 발전하는 것을 경험할 수 있었습니다. 이러한 가치있는 경험은 앞으로의 프로젝트에서도 큰 도움이 될 것입니다. 이후로도 더욱 경험과 지식을 쌓고, 지속적으로 열심히 공부하여 실력있는 개발자로 성장하기 위해 노력할 것입니다. 이번 프로젝트에서 얻은 교훈을 바탕으로 끊임없이 발전하는 개발자가 되겠습니다.

최미지

AI 기반 모의면접 서비스 프로젝트에서 관리자 페이지 개발을 맡아 중요 기능들을 구현했습니다. 기능 통합 문제로 어려움을 겪었지만, 팀원들과의 주기적인 회의와 GitHub 소통 덕분에 하나씩 해결할 수 있었습니다. 이 과정에서 저는 단순히 기술적 성장을 넘어, 커뮤니케이션이 협업 성공에서 가장 중요한 점이라는 걸 깨달을 수 있었습니다. 서로의 역할을 존중하고 의견을 나누는 과정에서 진정한 팀워크가 무엇인지 배울 수 있었습니다. 이번 프로젝트 경험은 제게 큰 자부심을 안겨주었고, 앞으로 더 나은 개발자로서 성장할 수 있는 밑거름이 되었습니다.

윤성빈

프로젝트 초반에는 '내가 맡은 파트만 잘 마무리하면 된다' 라는 생각이 들었습니다. 하지만 프로젝트가 진행될수록 협업의 중요성을 깊이 깨달았습니다. 각자의 역할에 책임을 다하며 성실히 임하는 팀원들의 모습을 보며 동기부여를 얻었습니다. 새로운 분야에 도전하며 많은 어려움이 있었지만 끝까지 포기하지 않고 문제를 극복해낸 경험은 제게 큰 성취감과 자신감을 안겨주었습니다. 이러한 경험을 바탕으로 한층 더 성장하였다는 믿음을 갖게 되었습니다

김세종

팀워크를 통해 어려움을 극복하고 성장할 수 있었습니다. 처음 접하는 기술과 복잡한 문제들로 가득했던 이번 프로젝트는 저에게 큰 도전이었습니다. 그러나 그 과정을 통해 새로운 기술을 배우고 문제를 해결하는 능력이 크게 향상되었으며 처음이라 지치고 많이 힘들었지만 함께 이끌어 주려는 팀원들의 소중함을 깨닫고 또한 팀원들과 공유하여 문제를 해결하고 목표를 달성하는 과정에서 협업의 중요성을 느끼는 프로젝트였습니다. 이러한 경험은 제 커리어에 있어 중요한 자산이 될 것이며 앞으로의 업무에서도 팀워크를 더욱 중요시 하고 적극적으로 임해야겠다는 생각이 들었습니다 .

Q&A

THANKS