

[Django Project]

CareerPop

경력을 팝하게 터뜨리다

Team : CareerPop

[목차 페이지]



01 개요

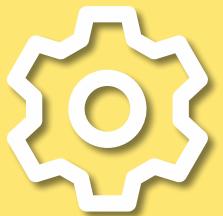
03 기능

02 역할/평가

04 구현영상



개요



이러한 부담감은 작성에 대한 어려움으로 이어져 응답자의 83%가 자소서 작성이 힘들다고 답변했다. 어려움을 느끼는 이유로는 ‘무엇을 써야 할지 막막해서’(54.4%, 복수응답)라는 답변이 많았고, ‘쓸 만한 스토리가 없어서’(46.6%), ‘기업마다 요구하는 항목이 너무 다양해서’(40.4%), ‘원래 글을 쓰는데 자신이 없어서’(26.8%), ‘매번 서류전형에서 탈락해서’(12.6%), ‘요구하는 분량이 너무 많아서’(12.5%) 등의 의견이 있었다.

작성에 가장 어려움을 느끼는 항목으로는 ‘지원동기’(22.1%)가 1위를 차지했다. 다음으로 ‘직무 관련 경험’(15.7%), ‘성장과정’(14.4%), ‘입사 후 포부’(12.6%), ‘특정 이슈에 대한 견해’(8.6%), ‘어려움을 극복한 경험’(8.4%), ‘성격의 장단점’(8.4%) 등의 순이었다.

Django 기반 웹 서비스 구현

많은 구직자들이 이력서를 작성할 때 항목을 어떻게 구성해야 할지 고민하거나, 작성 후에도 편집과 저장 과정에서 불편함을 겪는 경우가 많다.

이러한 문제를 해결하기 위해 Django 프레임워크를 기반으로, 이력서를 항목별로 작성·수정하고, 미리보기 및 PDF 저장까지 가능한 웹 서비스를 구현하였다. 사진, 학력, 경력, 스킬 등 다양한 정보를 자유롭게 입력할 수 있으며, 선택한 항목만 출력되도록 구성해 사용자 맞춤형 이력서 생성이 가능하다.

GPT API 기반 AI 첨삭 기능

자기소개서나 경력기술서를 작성할 때 많은 사용자들이 “무엇을 써야 할지 막막하다”거나 “글쓰기에 자신이 없다”는 어려움을 느낀다.

이를 보완하기 위해 OpenAI의 GPT API를 연동하여, 맞춤법 검사와 AI 첨삭 기능을 제공하는 프롬프트 엔지니어링을 설계하였다. 입력한 글을 문맥에 맞게 자연스럽게 다듬거나 강조 포인트를 보완해주는 기능으로, 사용자는 보다 완성도 높고 신뢰감 있는 자기소개서를 빠르게 완성할 수 있다.

배고은

역할

- 회원가입/로그인 및 접근 제어 구현
 - Django 기본 User 모델을 활용하여 사용자 등록 및 로그인 기능 구현
 - 로그인한 사용자만 이력서 작성 가능하도록 세션 기반 접근 제어 로직 구성
- OpenAI API 연동
 - Django 백엔드에서 OpenAI API와 안정적으로 연동
- 이력서 저장 및 불러오기 시스템
 - 사용자 이력서를 JSON 포맷으로 직렬화하여 데이터베이스에 저장
 - 학력, 경력, 스킬, 포트폴리오 등 다양한 항목을 사용자 단위로 효율적으로 관리
 - 저장된 이력서를 동적으로 복원하는 렌더링 로직 구현
- UI/UX 스타일링 및 인터랙션 설계
 - 기존 HTML 템플릿과 CSS를 활용한 전체 화면 스타일링 담당
 - 사용자 친화적인 폼 UI 및 반응형 디자인 설계
- 스킬 자동완성 기능 개발
 - JavaScript 기반 기술 스택 자동완성 기능 구현
 - 사용자 경험 향상 및 오토타자 방지
- 조건부 렌더링 기능 구현
 - 체크박스를 기반으로 특정 항목(스킬, 경력, 자기소개 등)만 화면에 동적으로 표시
 - 사용자의 선택에 따라 유연한 UI 구성 가능
- 프로필 이미지 업로드 및 미리보기 기능
 - Django ImageField를 활용한 프로필 사진 업로드 기능 구현
 - 업로드 후 즉시 이미지 미리보기가 가능하도록 프론트 처리

자체평가

이번 프로젝트에서는 AI 기술을 활용한 이력서 자동작성 및 관리 시스템을 웹 기반 애플리케이션 형태로 구현하며, 다양한 웹 개발 기술과 사용자 경험 설계 역량을 실무적으로 터득할 수 있었습니다.

Django 프레임워크를 기반으로 한 백엔드 설계에서는 회원가입/로그인 기능 및 세션 기반 접근 제어를 직접 구현하였고, 로그인된 사용자만 이력서 작성 및 저장이 가능하도록 보안과 권한 관리를 체계적으로 구축했습니다. 사용자 인증 이후에는 각 유저가 생성한 이력서 데이터를 JSON 형태로 직렬화하여 DB에 저장, 다시 웹페이지에 불러와서 동적으로 렌더링되는 흐름을 구성하며, 복잡한 데이터 모델을 다루는 설계와 처리 로직에 대한 감각을 키울 수 있었습니다.

AI 기능 측면에서는, 팀원이 구현한 이력서 첨삭 및 맞춤법 검사 로직이 안정적으로 작동할 수 있도록 OpenAI API 키를 Django 백엔드에 연동하고, 프론트엔드에서 입력된 텍스트가 API 요청을 통해 서버로 전달되고 응답을 받아 다시 사용자에게 표시되도록 데이터 흐름 전반을 설계하고 구현했습니다. 이 과정을 통해 실제 상용 AI API를 활용한 비동기 통신, 요청/응답 처리, 예외 대응 등 실무적인 연동 경험을 쌓을 수 있었습니다.

프론트엔드에서는 팀원이 미리 구성한 HTML 구조를 기반으로 CSS 스타일링 전반을 직접 설계하고 구현하였습니다. 색상, 간격, 타이포그래피, 반응형 대응 등 시각적 디자인 요소를 조율하여 사용자 친화적이고 일관된 UI를 구성하였고, 정보의 우선순위와 입력 흐름을 고려한 디자인을 통해 이력서 작성 환경의 편의성을 높였습니다. 또한, 스킬 입력 자동완성 기능은 JavaScript를 활용해 실시간 추천이 가능하도록 구현하였고, 조건부 렌더링을 통해 사용자가 원하는 항목만 동적으로 출력되도록 구성했습니다. 프로필 이미지 업로드 및 미리보기 기능도 Django와 JS를 연동하여 사용자 친화적으로 구현했습니다.

다만, 기능 구현에 집중하면서 웹페이지에 기능 사용법이나 작성 흐름에 대한 안내 문구가 부족했던 점은 아쉬움으로 남았고, 이는 향후 툴팁 제공, 가이드 섹션 추가, 예시 기반 입력 설계 등을 통해 개선이 필요하다고 느꼈습니다.

이번 프로젝트는 단기간 동안 프론트엔드와 백엔드 모두를 아우르는 전반적인 웹 개발 경험을 할 수 있었고, 협업 기반 개발 환경에서의 역할 구분과 커뮤니케이션, 그리고 실제 사용자 중심의 기능 구현과 디자인 구현 역량을 함께 성장시킬 수 있었던 의미 있는 시간이었습니다.

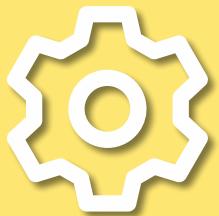


역할/평가





역할/평가



곽희원

역할

1. 자기소개서 수정/삭제 기능 구현
 - 사용자가 입력한 자기소개서를 실시간으로 수정하거나 삭제할 수 있도록 품 처리 및 삭제 로직을 구현함.
2. 경력기술서 카드 생성 및 저장 기능
 - 텍스트 기반 입력값을 받아 경력 기술 항목별로 개별 카드 형태로 추가하고, 저장되도록 처리함.
3. 경력기술서 수정/삭제 기능 구현
 - 생성된 각 경력 카드에 대해 수정 또는 삭제할 수 있는 기능 구현.
4. 글자 수 및 바이트 수 카운트 기능 개발
 - 입력 중인 글자 수와 바이트 수를 실시간으로 계산하여 사용자에게 표시 기능
 - 작성 가이드 제공 기능 구현.
5. 이력서 미리보기 및 PDF 변환
 - 사용자 입력값을 POST 방식으로 서버에 전송해 HTML 기반의 이력서 PDF를 생성할 수 있도록 함.
 - 서버에서 생성된 PDF 페이지를 새 창에서 확인할 수 있도록 fetch API와 window.open을 조합해 미리보기 페이지 구성함.
 - 미리보기 페이지가 열리면 PDF 파일 다운로드 기능 구현.
6. GPT 학습용 자기소개서 데이터 수집/정제
 - AI 첨삭 기능을 위해 다양한 유형의 자기소개서를 수집하고 정제된 형태로 가공함.
7. 프롬프트 엔지니어링 기반 문장 교정 기능 구현
 - GPT API와 연동하여 사용자의 문장을 자연스럽고 명확하게 수정하는 프롬프트 기반 교정 로직을 설계함.

자체평가

본 프로젝트는 사용자가 직접 이력서를 작성하고, 작성한 자기소개서에 대해 인공지능 기반 첨삭 피드백을 받을 수 있도록 개발한 웹 기반 이력서 관리 시스템이다.

초기에는 OpenAI의 파인튜닝 기능을 활용해 맞춤형 자소서 첨삭 모델을 만들고자 했으나, 적절한 학습 데이터 확보와 비용 문제로 인해 실제 구현까지는 이어지지 못했다.

이러한 한계를 보완하기 위해, 프롬프트 엔지니어링을 기반으로 한 시스템 프롬프트 설계 방식을 선택했다. GPT의 응답 스타일을 의도적으로 유도하여, 맞춤법 교정뿐 아니라 문장 흐름 개선과 어휘 제안까지 채용 담당자 관점에서 자연스럽고 구체적인 첨삭 결과를 제공하도록 구현하였다.

또한, 작성된 이력서를 HTML 템플릿으로 렌더링한 뒤, 서버로 전송된 데이터를 기반으로 PDF 출력용 미리보기 기능도 구현하였다. 사용자는 입력 데이터를 실시간으로 확인하며, 최종 이력서 형식을 새 창에서 바로 검토하고 저장할 수 있다.

이번 프로젝트를 통해 단순한 입력 품 설계를 넘어서, 입력 → 첨삭 → 미리보기 → 출력으로 이어지는 사용자 흐름을 전반적으로 설계하고 구현해보는 경험을 쌓을 수 있었다.

아쉬운 점은, 처음 계획했던 파인튜닝 기반의 맞춤형 모델 구현을 끝내지 못한 점이다. 향후에는 사용자 데이터를 기반으로 한 모델 학습과, 첨삭 이력 비교 및 버전 관리 기능도 함께 구현해보고 싶다.



김대원

역할/평가



역할

1. 이력서 웹사이트 기본 베이스 구현
 - Figma를 사용하여 웹사이트 UI구성
 - 이력서의 My Career list 제작
 - 이력서의 폼 위치 구성
 - PDF내보내기 디자인 제작
 - Career list 동적 렌더링 기능 설계
2. 학력, 경력, 경험/활동/교육, 포트폴리오 폼 기능 구현
 - 사용자가 필요한 폼을 선택하고 입력할 수 있도록 기본적인 틀 및 기능 구현
 - 사용자가 원하는 만큼 기제할 수 있도록 폼 추가기능 구현
 - 잘못 작성하였거나 수정할 부분이 생겼을 때 사용하도록 수정, 삭제 기능 제작
 - 데이터를 모두 저장 하도록 localStorage에 저장하는 기능 제작
 - 저장 후 카트형태로 보이는 기능 구현
 - 예외처리기능 구현
3. 코드 병합
 - JS코드 받아, 통합 저장 코드 생성 및 코드 병합
4. 기능 테스트
 - 코드 병합 후 기능 테스트 및 시나리오 수행

자체평가

이번 프로젝트에서 저는 UI 기획부터 구현, 테스트까지 전 단계에 꼭넓게 참여하며 기획형 프론트엔드 개발자로서의 역량을 종합적으로 발휘했습니다.

먼저 Figma를 활용해 웹사이트의 UI를 설계하고, 이력서 시스템의 정보 구조에 맞춰 기본 베이스를 구성했습니다. 사용자 흐름을 고려한 레이아웃과 컴포넌트 설계를 통해 직관적인 UI를 구현하는데 중점을 두었으며, 실제 사용자 요구를 반영한 구조로 Career list, 학력/경력/포트폴리오 폼을 기획했습니다.

프론트엔드 개발 단계에서는 JavaScript를 활용해 Career list의 동적 렌더링 기능 및 예외 처리, 이력서 데이터 저장 및 불러오기 기능, 그리고 통합 폼과의 연동 로직을 설계하고 구현했습니다. 기능 오류 발생 시 빠르게 원인을 분석하고 수정했으며, QA 단계에서는 기능별 시나리오를 작성하고 테스트를 수행하여 최종 완성도를 높였습니다.

전체 과정에서 기획-개발-검증을 유기적으로 반복하면서 문제 해결력, 구현력, 사용자 관점의 사고를 함께 키울 수 있었으며, 프로젝트의 완성도뿐 아니라 팀워크 및 협업 역량 또한 향상되었습니다.



역할/평가



김범진

역할

1. OpenAI API 연동작업 진행
 - cmd 통해서 OpenAI API와 웹사이트 간 연동 작업 구성
 - ChatGPT gpt-4o-mini 모델 적용
 - 맞춤법 수정 기능 구현 및 적용
 - AI 첨삭 기능 구현 및 적용
2. 자격증/어학/수상 작성 기능 제작
 - 각 필드별로 내용 형성되도록 div로 구분
 - 필드별 상세 내용 입력 필드 생성
 - 내용 저장 기능 및 내용 공백 있을 시 알림 기능 구현
 - 내용 수정 및 불러오기 기능 구현
3. 취업우대사항 작성 기능 제작
 - 각 필드별 내용 형성되도록 div로 구분
 - 필드별 상세 항목 드롭다운으로 구성
 - 병역-군필 선택 시 세부사항 입력 기능 구현
 - 내용 저장 기능 및 내용 공백 있을 시 알림 기능 구현
 - 내용 수정 및 불러오기 기능 구현

자체평가

이번 프로젝트에서는 Django를 기반으로 한 이력서 작성 웹사이트를 구축했습니다. 회원 가입·로그인 후 상세 정보를 입력·저장하고, 미리보기 페이지에서 PDF로 내려받을 수 있도록 설계했습니다. 핵심 기능은 HTML·Python으로 구현했으며, 자기소개서 작성 영역에는 OpenAI API를 연동해 맞춤법 검사와 AI 첨삭 기능을 제공하도록 했습니다. 데이터 저장은 SQLite를 사용했습니다.

기대이상으로 진행되었던 부분으로는 Django 자체가 Python과 친화적인 형태의 프레임워크라는 특성 덕분에 웹사이트의 디자인 및 기능 구현이 이전 HTML과 JavaScript 만으로 웹사이트를 제작했던 당시와는 비교도 안 되는 속도로 디자인과 기능이 구현되었다는 부분입니다. 특히, 자체적으로 DB의 구성이 가능하다는 특성은 데이터의 저장을 LocalStorage에만 의존해야 했던 이전 프로젝트와는 다르게 데이터의 저장 및 유지 과정이 원활하게 이루어지는 장점으로 작용했습니다. 이외에도 API 호출과 응답 처리 과정이 깔끔해, 초기 구상했던 디자인·기능을 빠른 시간 안에 구현할 수 있었습니다.

반면 예상 외로 어려움을 겪었던 부분으로는 자기소개서 작성 부분에서의 ChatGPT 모델을 어떻게 적용해야 할지에 관한 부분이었습니다. GPT 모델 별로 내놓는 답변의 양과 질이 천차만별이었고 API 연동 후 사용과정에서 소비되는 금액 역시 모델 별로 상이했기에 이러한 부분을 고려하고 답변의 내용을 어떤 방식으로 나오게 할 지 많은 고민을 거듭할 수 밖에 없었습니다. 이후 타 기능과 합치는 과정에서도 기능 간의 충돌이 생각보다 잦아 진행 시간이 당초 예상했던 것보다 길게 소모되는 원인으로 작용하였습니다.

AI 적용 과정에서 시행착오가 있었지만, 초기 목표였던 디자인과 기능을 모두 구현하며 프로젝트를 성공적으로 마무리했습니다. 앞으로는 더 창의적인 기능을 추가해, 사용자 경험을 한층 향상시키는 웹사이트를 제작해보고자 합니다.

회원가입 페이지



기능



```
def signup_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password1 = request.POST.get('password1')
        password2 = request.POST.get('password2')

        if password1 != password2:
            messages.error(request, '비밀번호가 일치하지 않아요.')
            return render(request, 'signup.html', {'username': username})

        if User.objects.filter(username=username).exists():
            messages.error(request, '이미 존재하는 아이디입니다.')
            return render(request, 'signup.html')

        user = User.objects.create_user(username=username, password=password1)
        login(request, user) # 가입 후 바로 로그인 처리
        messages.success(request, f'{username}님 환영합니다 ❤')
        return redirect('login')

    return render(request, 'signup.html')
```

실제 회원가입 화면

signup_view 함수를 통해 사용자로부터 전달받은 회원가입 정보를 기반으로 새로운 계정을 생성하고 가입 직후 자동 로그인까지 처리

- 비밀번호 일치 여부, 아이디 중복 여부
- User.objects.create_user()로 사용자 등록
- 가입 직후 바로 login() 처리
- messages.error(), messages.success()로 상황별 알림 제공
- CSRF 보호는 템플릿에서 csrf_token 사용 전제로 함

로그인 페이지



기능



```
def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            # messages.success(request, f'{username}님 환영합니다 ❤')
            return redirect('resume_page')
        else:
            messages.error(request, '아이디 또는 비밀번호가 틀려요.')

    return render(request, 'login.html')

def logout_view(request):
    logout(request)
    return redirect('login')
```

실제 로그인 화면

login_view(request) 함수를 사용하여 사용자의 로그인 요청을 처리하고 인증에 성공하면 이어서 페이지로 리다이렉트함.

- authenticate()를 사용한 사용자 인증
- login() / logout()으로 로그인 상태 유지 및 해제
- 로그인 실패 시 messages.error()를 통해 사용자에게 안내
- POST 요청에 대해서만 로그인 로직 수행 (CSRF 보호 포함)

The image shows a simplified login form interface. It features a light pink header with a user icon. Below it is a white rectangular form containing two input fields: 'Username' and 'Password', both with placeholder text. A large red button labeled 'LOGIN' is centered below the inputs. At the bottom right of the form, there is a small link text: '아직 회원이 아니신가요? 회원가입하기'.

이력서 저장



기능



작성완료 클릭 시
saveresume() 함수 호출

```
function saveResume() {
    const title = document.getElementById('resume-title').value;
    const name = document.getElementById('name').value;
    const gender = document.getElementById('gender').value;
    const birthdate = document.getElementById('birthdate').value;
    const email = document.getElementById('email').value;
    const phone = document.getElementById('phone').value;
    const address = document.getElementById('address').value;
    const detailAddress = document.getElementById('detail-address').value;

    // sections 객체 정의
    const sections = [
        education: collectEducationData() || [],
        career: collectCareerData() || [], You, last week • 수정됨 ...
        experience: collectExperienceData() || [],
        certificate: collectCertificateData() || [],
        preference: collectPreferenceData() || [],
        portfolio: collectPortfolioData() || [],
        description: collectCareerDescriptionData() || [],
        introduction: collectIntroductionData() || [],
        skills: selectedSkills || [],
        sectionCheckboxStates: getSectionCheckboxStates()
    ];

    const formData = new FormData();
    formData.append('title', title);
    formData.append('name', name);
    formData.append('gender', gender);
    formData.append('birthdate', birthdate);
    formData.append('email', email);
    formData.append('phone', phone);
    formData.append('address', address);
    formData.append('detail-address', detailAddress);
    formData.append('sections_json', JSON.stringify(sections));
}
```

- 사용자가 입력한 이력서 기본 정보를 input 필드에서 추출하여 변수에 저장하고 학력, 경력, 스킬 등 부가 항목들은 collectXXXData() 함수로 각각의 섹션 데이터를 수집하여 sections 객체로 구성합니다.
- 체크박스 상태도 sectionCheckboxStates에 같이 저장하여 UI 복원 시 활용됩니다.
- FormData()를 활용해 서버에 보낼 데이터를 key-value 형태로 구성하고 sections 객체는 JSON 문자열로 직렬화하여 전송합니다.

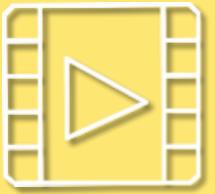
이력서 미리보기

작성완료

이력서 저장



기능



```
@login_required
def save_resume(request):
    if request.method == 'POST':
        import pprint
        pprint.pprint(request.POST)

        content_dict = {
            "name": request.POST.get("name"),
            "gender": request.POST.get("gender"),
            "birthdate": request.POST.get("birthdate"),
            "email": request.POST.get("email"),
            "phone": request.POST.get("phone"),
            "address": request.POST.get("address"),
            "detail_address": request.POST.get("detail-address"),
        }

        sections_json = request.POST.get("sections_json")
        if sections_json:
            try:
                sections_data = json.loads(sections_json)
                content_dict["sections"] = sections_data
            except Exception as e:
                print("sections_json 파싱 실패", e)

        pprint.pprint(content_dict)

        # 이미지 처리
        image_file = request.FILES.get("profile_image")
        resume = Resume.objects.create(
            user=request.user,
            title=request.POST.get("title"),
            profile_image=image_file,
            content="임시" # 우선 임시 저장
        )
```

- POST 요청이 들어오면, pprint로 입력 값을 콘솔에 출력하여 디버깅합니다.
- 로그인한 사용자만 접근할 수 있도록 @login_required 데코레이터가 사용됩니다.
- 클라이언트에서 전달받은 기본 정보를 딕셔너리 형태로 저장합니다.
- 프론트에서 전송한 JSON 형태의 이력서 추가 항목을 파싱하여 sections에 저장합니다.
- 사용자 정보 및 이력서 정보를 바탕으로 DB에 이력서를 저장합니다.

배고은님 환영합니다 ♥

로그아웃

이력서 등록

이력서 관리

문제 해결에 진심인 서버 개발자 배고은, 사용자 중심 시스템을 설계합니다

이력서 저장



기능



```
function getSectionCheckboxStates() {
  return {
    education: document.getElementById('section-education').checked,
    career: document.getElementById('section-career').checked,
    skill: document.getElementById('section-skill').checked,
    experience: document.getElementById('section-experience').checked,
    certificate: document.getElementById('section-certificate').checked,
    preference: document.getElementById('section-preference').checked,
    portfolio: document.getElementById('section-portfolio').checked,
    description: document.getElementById('section-description').checked,
    introduction: document.getElementById('section-introduction').checked
  }
}
```

이 함수는 각 항목(예: 학력, 경력, 스킬 등)의 체크박스 상태를 확인해서 객체로 반환합니다.

- `document.getElementById('section-xxx').checked`는 해당 ID를 가진 체크박스가 체크되어 있는지 여부를 `true` 또는 `false`로 반환합니다.

각 스킬 별 렌더링 코드를 작성했습니다.
`resume.sections` 데이터를 바탕으로 html에
각 스킬별 정보를 동적으로 생성 후 웹 페이지
에 표시합니다.

- `resume.sections.xxxxxxx`가 배열인지 확인
후 `forEach()`로 하나씩 순회.
- 각 항목마다 새로운 `<div>`를 생성하고
클래스명을 부여.
- 작성된 요소는 `xxxxxx-list`라는 ID를 가진
DOM 요소에 추가됨

```
// 1. 학력
if (resume.sections && Array.isArray(resume.sections.education)) {
  resume.sections.education.forEach(edu => {
    const list = document.getElementById('education-list');
    const entry = document.createElement('div');
    entry.className = 'education-entry';
    entry.innerHTML =
      `

<h4>${edu.school || ''} (${edu.startDate || edu['입학년도'] || ''} ~ ${edu.endDate || edu['졸업년도'] || ''})</h4>
          <p><strong>전공:</strong> ${edu.major || edu['전공'] || ''}</p>
          <p><strong>학점:</strong> ${edu.grade || edu['학점'] || ''}</p>
          <p><strong>졸업 구분:</strong> ${edu.graduation || edu['졸업구분'] || ''}</p>

`;
    list.appendChild(entry);
  });
}
```

이력서 가져오기 및 삭제



기능



- delete_resume()은 특정 이력서를 삭제하는 기능입니다.
- 이력서 리스트에서 선택한 항목을 서버에서 삭제하는 기능을 구현하였으며, 사용자 인증 및 요청 방식도 함께 검증하였습니다.

```
@login_required
def resume_page(request):
    resumes = Resume.objects.filter(user=request.user) # 로그인한 유저 것만
    return render(request, 'resume.html', {'resumes': resumes})

@login_required
def get_resume(request, resume_id):
    try:
        resume = Resume.objects.get(id=resume_id, user=request.user)
        content_dict = json.loads(resume.content)

        resume_data = {
            'title': resume.title,
            'content': content_dict,
            'profile_image_url': resume.profile_image.url if resume.profile_image else None
        }
        return JsonResponse(resume_data)
    except Resume.DoesNotExist:
        return JsonResponse({'error': '이력서를 찾을 수 없습니다.'}, status=404)
```

- resume_page()은 로그인한 사용자에 따라 개인 이력서 리스트를 조회하고 렌더링하는 기능을 구현하였습니다.
- get_resume()은 선택한 이력서를 JSON 형태로 가져와, UI에 각 항목을 자동으로 렌더링하는 로직을 구현하였습니다.

```
@login_required
@csrf_exempt
def delete_resume(request, resume_id):
    if request.method == 'POST':
        try:
            resume = Resume.objects.get(id=resume_id, user=request.user)
            resume.delete()
            return JsonResponse({'success': True})
        except Resume.DoesNotExist:
            return JsonResponse({'success': False, 'error': 'Resume not found'}, status=404)
    return JsonResponse({'success': False, 'error': 'Invalid method'}, status=405)
```

학력 품



기능



```
function saveEducation(button) {
  } else {
    const key = input.placeholder || '선택';
    values[key] = input.value.trim();
  });
}

// 초기 환경 설정
const startDate = new Date(values['입학년도']);
const endDate = new Date(values['졸업년도']);

if (startDate > endDate) {
  // 필수 입력값 설정
  if (!values['학교명'] || !values['입학년도'] || !values['졸업년도'] || !values['전공']) {
    alert('필수 항목을 입력하세요!');
    return;
  }
}

function searchSkills() {
  const input = document.getElementById("skillSearch").value.toLowerCase();
  const suggestionsBox = document.getElementById("skillSuggestions");
  suggestionsBox.innerHTML = "";

  if (!input) {
    suggestionsBox.style.display = "none";
    return;
  }

  const matches = availableSkills.filter(skill =>
    skill.toLowerCase().includes(input) && !selectedSkills.includes(skill)
  );

  if (matches.length === 0) {
    suggestionsBox.style.display = "none";
    return;
  }

  suggestionsBox.style.display = "block";
  matches.forEach(skill => {
    const div = document.createElement("div");
    div.textContent = skill;
    div.style.padding = "5px";
    div.style.cursor = "pointer";
    div.onclick = () => {
      selectedSkills.push(skill);
      document.getElementById("skillSearch").value = "";
      suggestionsBox.style.display = "none";
      searchSkills(); // 다시 실행해서 새 목록 보여
    };
    suggestionsBox.appendChild(div);
  });
}

// 보기 카드 형태로 학력을 HTML 구성
summary.innerHTML = `


<h4>${values['학교명']} | ${values['입학년도']} ~ ${values['졸업년도']} | |
  <p><strong>전공:</strong> ${values['전공']} | |
  <p><strong>학점:</strong> ${values['학점']} | |
  <p><strong>졸업 구분:</strong> ${values['졸업구분']} | |
  <button onclick="editEducation(this)" class="cencle_save_btn" style="float: right;">수정
  <button onclick="removeEducationForm(this)" class="cencle_save_btn" style="float: right;">삭제
</div>
`;


```

- certificateFields: 각 항목의 필드 세부 정보 배열
- collectCertificateData(): 졸업, 재학, 휴학 드롭다운 생성
- showCertificateFields(): 선택한 항목에 따라 입력 필드 표시
- renderCertificateList(): 입력 품 생성
- editCertificate(): 저장된 정보를 수정 가능하도록 표시
- saveCertificate(): 필수 항목 입력 후 localStorage에 저장

학력

서울대학교 (2019-03-01 ~ 2025-05-03) (~)

전공: 컴퓨터공학과

학점: 4.3

졸업 구분: 졸업 수정

경력 품



기능



```
function saveCareer(button) {
  inputs.forEach(input => {
    values[input.placeholder] = input.value;
  });

  // 필수 입력값 검증
  if (!values['회사명'] || !values['입사년월'] || !values['퇴사년월'] || !values['근무부서'] || !values['직무'] || !values['직급'] || !values['담당업무']) {
    alert('회사명, 입사년월, 퇴사년월, 근무부서, 직무, 담당업무는 필수 입력 항목입니다.');
    return;
  }

  // 날짜 형식 검증
  const startDate = new Date(values['입사년월']);
  const endDate = new Date(values['퇴사년월']);

  if (startDate > endDate) {
    alert('퇴사년월은 입사년월보다 이후여야 합니다.');
    return;
  }

  // HTML 템플릿 생성
  const careerCard = document.createElement('div');
  careerCard.className = 'career-card';
  careerCard.innerHTML = `
    <h4>${values['회사명']} || ${values['입사년월']} ~ ${values['퇴사년월']} || ${values['직무']} ${values['직급']} ${values['담당업무']}
    <p><strong>직무:</strong> ${values['직무']}</p>
    <p><strong>근무부서:</strong> ${values['근무부서']}</p>
    <p><strong>직급:</strong> ${values['직급']}</p>
    <p><strong>담당업무:</strong> ${values['담당업무']}
    <div class="card-buttons" style="display: flex; justify-content: flex-end; gap: 8px;">
      <button onclick="editCareer(this)" class="cencle_save_btn">수정</button>
      <button onclick="removeCareerForm(this)" class="cencle_save_btn">삭제</button>
    </div>
  `;

  // 기존 내용 지우고 새로운 카드 추가
  summary.innerHTML = '';
  summary.appendChild(careerCard);

  // 입력 요소 숨기기
  inputs.forEach(input => input.style.display = "none");

  // 저장/취소 버튼 숨기기
  const formButtons = entry.querySelector(".form-buttons");
  if (formButtons) formButtons.style.display = "none";

  // 요약 카드 표시
  summary.style.display = "block";

  // localStorage에 저장
  const careerData = {
    company: values['회사명'],
    startDate: values['입사년월'],
    endDate: values['퇴사년월'],
    rank: values['직급'],
    duties: values['담당업무']
  };

  // 기존 데이터 가져오기
  let savedCareers = JSON.parse(localStorage.getItem('careers') || '[]');
  savedCareers.push(careerData);
  localStorage.setItem('careers', JSON.stringify(savedCareers));
}
}
```

- careerFields: 각 항목의 필드 세부 정보 배열
- collectCareerData(): 드롭다운 생성 (필요 시)
- showCareerFields(): 선택한 항목에 따라 입력 필드 표시
- renderCareerList(): 입력 품 생성
- editCareer(): 저장된 정보를 수정 가능하도록 표시
- saveCareer(): 필수 항목 입력 후 localStorage에 저장

경력

삼성 (2024-11-26 ~ 2025-03-28)

직무: 개발자

근무부서: 개발팀

직급: 인턴

담당업무: 삼성의 개발팀에서 프로젝트업무 도움

스킬



기능



```
// 사용할 스킬 목록
const availableSkills = [
  "Python", "Java", "C", "C++", "C#", "JavaScript", "TypeScript", "Go", "Kotlin", "Swift", "Ruby", "PHP", "Rust", "Scala",
  "HTML", "CSS", "React", "Vue.js", "Angular", "Next.js", "Svelte", "Tailwind CSS", "Bootstrap", "jQuery",
  "Django", "Flask", "Spring", "Spring Boot", "Node.js", "Express", "FastAPI", "NestJS", "Ruby on Rails", "ASP.NET",
  "MySQL", "PostgreSQL", "SQLite", "MongoDB", "Firebase", "Oracle", "AWS", "Azure", "Google Cloud", "Docker", "Kubernetes",
  "Git", "GitHub", "Linux", "VS Code", "Jenkins", "Notion", "Figma", "TensorFlow", "PyTorch", "OpenCV", "Unity", "Unreal Engine"
];
```

```
// 자동완성 검색
function searchSkills() {
  const input = document.getElementById("skillSearch").value.toLowerCase();
  const suggestionsBox = document.getElementById("skillSuggestions");
  suggestionsBox.innerHTML = "";

  if (!input) {
    suggestionsBox.style.display = "none";
    return;
  }

  const matches = availableSkills.filter(skill =>
    skill.toLowerCase().includes(input) && !selectedSkills.includes(skill)
  );

  if (matches.length === 0) {
    suggestionsBox.style.display = "none";
    return;
  }

  suggestionsBox.style.display = "block";
  matches.forEach(skill => {
    const div = document.createElement("div");
    div.textContent = skill;
    div.style.padding = "5px";
    div.style.cursor = "pointer";
    div.style.borderBottom = "1px solid #eee";

    div.onclick = () => {
      selectedSkills.push(skill);
      document.getElementById("skillSearch").value = "";
      suggestionsBox.style.display = "none";

      // ✅ 리스트 실시간 업데이트
      const list = document.getElementById("skill-list");
      updateSkillList();
      searchSkills(); // 중복 제거용
    };
  });
}
```

- skillSearch 입력창에서 텍스트를 소문자로 가져옴
- skillSuggestions 영역 초기화
- 입력값이 비어 있으면 추천창 숨기고 종료
- 전체 스킬 목록 중 입력값이 포함된 스킬, 아직 선택되지 않은 스킬만 추출
- 필터링된 스킬을 추천창에 <div>로 렌더링
- 마우스로 클릭 가능한 형태로 스타일 적용
- 선택된 스킬 배열(selectedSkills)에 추가
- 입력창 초기화 및 추천창 숨김
- 중복 방지를 위해 searchSkills() 재실행

The screenshot shows a user interface for managing skills. At the top, there is a header with the word '스킬' (Skill). Below it, a list of checked skills is shown: JavaScript, Python, C, CSS, React, VS Code, Java, and Django. A search bar contains the letter 'd'. Below the search bar, a list of suggestions is displayed: Tailwind CSS, Node.js, MongoDB, Google Cloud, and Docker. At the bottom of the interface are two buttons: '취소' (Cancel) and '저장' (Save).

경험/활동/교육 품



```
function removeExperienceForm(button) {…}
// 경험/활동/교육 제작
function saveExperience(button) {
  const entry = button.closest(".experience-entry");
  const summary = entry.querySelector(".experience-summary");
  const [const values: {}] = querySelectorAll("input, textarea, select");
  const values = {};

  // 일정과 결합
  const startDate = new Date(entry.querySelector('input[placeholder="시작년월"]').value);
  const endDate = new Date(entry.querySelector('input[placeholder="종료년월"]').value);

  if (startDate > endDate) {…}

  // 필수 입력과 결합
  const activityType = entry.querySelector('select').value;
  const place = entry.querySelector('input[placeholder="기관/장소명"]').value;
  const description = entry.querySelector('textare').value;

  if (!activityType || !place || !description) {…}

  // 값 저장
  inputs.forEach(input => {
    if (input.tagName === 'SELECT') {
      values['활동 유형'] = input.options[input.selectedIndex].text;
    } else {
      values[input.placeholder] = input.value.trim();
    }
  });

  // 날짜 포맷팅
  const formatDate = (dateStr) => {
    if (!dateStr) return '';
    const date = new Date(dateStr);
    return `${date.getFullYear()}-${String(date.getMonth() + 1).padStart(2, '0')}`;
  };

  // 경험 데이터 저장
  const experienceData = {
    activityType: values['활동 유형'],
    place: values['기관/장소명'],
    startDate: values['시작년월'],
    endDate: values['종료년월'],
    description: values['활동 설명']
  };

  // localStorage에서 기존 데이터 가져오기
  let experiences = JSON.parse(localStorage.getItem('experiences')) || [];
  experiences.push(experienceData);
  localStorage.setItem('experiences', JSON.stringify(experiences));

  summary.innerHTML =
    <div class="experience-card">
      <h4>${values['활동 유형']} - ${values['기관/장소명']} | ${formatDate(values['시작년월'])} ~ ${formatDate(values['종료년월'])}</h4>
      <p><strong>활동 설명:</strong> ${values['활동 설명']}
      <div class="card-buttons" style="display: flex; justify-content: flex-end; gap: 8px;">
        <button onclick="editExperience(this)" class="cencle_save_btn">수정</button>
        <button onclick="removeExperienceForm(this)" class="cencle_save_btn">삭제</button>
      </div>
    </div>
  ;
}
```

- experienceFields: 각 항목의 필드 세부 정보 배열
- showExperienceFields(): 선택한 항목에 따라 입력 필드 표시
- showExperienceForm(): 입력 폼 생성
- editExperience(button): 저장된 정보를 수정 가능하도록 표시
- saveExperience(button): 필수 항목 입력 후 localStorage에 저장
- removeExperienceForm(button): 항목 삭제

경험/활동/교육

교육이수내역 - 글로벌아카데미 (2025.03 ~ 2025.04)

활동 설명: 글로벌아카데미에서 프로젝트 진행

자격/어학/수상



기능



```
//자격/어학/수상
function collectCertificateData() {
  const certificateEntries = document.querySelectorAll('.cert-card');
  return Array.from(certificateEntries).map(entry => {
    const detailText = entry.querySelector('span').textContent;

    // 자격증, 어학, 수상 구분
    if (detailText.includes('[자격증]')) {
      const [, info] = detailText.split('[자격증]');
      const [name, issuerAndDate] = info.split(',');
      const [issuer, date] = issuerAndDate.split('(');

      return {
        type: 'license',
        license_name: name.trim(),
        issuer: issuer.trim(),
        pass_date: date.replace(')', '').trim()
      };
    } else if (detailText.includes('[어학]')) {
      const [, info] = detailText.split('[어학]');
      const [lang, exam, score, status] = info.split(',');

      return {
        type: 'language',
        lang: lang.trim(),
        exam: exam.trim(),
        score: score.trim(),
        status: status.replace('(', '').replace(')', '').trim()
      };
    } else {
      const [, info] = detailText.split('[수상]');
      const [name, hostAndDate] = info.split(',');
      const [host, date] = hostAndDate.split('(');

      return {
        type: 'award',
        award_name: name.trim(),
        host: host.trim(),
        award_date: date.replace(')', '').trim()
      };
    }
  });
}

/* ===== 자격/어학/수상 ===== */
let certificateData = [];
let certEditIndex = null;
let editingCardEl = null;
```

```
function showcertificate(isEdit = false) {
  const form = document.getElementById('certificate-form');
  form.style.display = 'block';

  /* 새 항목일 때 원본 초기화 */
  if (!isEdit) {
    certEditIndex = null;
    resetCertForm(); // 드롭다운 초기화
  }

  showCertificateFields(); // 타입별 필드 토글
}
function cancelcertificate() {
  document.getElementById('certificate-form').style.display = 'none';

  /* ★ 습겨둔 카드가 있으면 다시 보이게 */
  if (editingCardEl) {
    editingCardEl.style.display = '';
    editingCardEl = null;
    certEditIndex = null;
  }
  resetCertForm(); // 빈 폼 유지
}
function showCertificateFields() {
  const type = document.getElementById('certificateType').value;

  /* (A) 아무 것도 선택하지 않은 상태면 전부 감춤 */
  if (!type) {
    document.querySelectorAll('.cert-fields').forEach(el => el.style.display = 'none');
    return;
  }

  /* (B) 선택값이 있을 때만 해당 블록 보여 줌 */
  document.querySelectorAll('.cert-fields').forEach(el => {
    el.style.display = el.id.startsWith(type) ? 'block' : 'none';
  });
}
```

- 각 필드의 세부 사항을 Array로 리턴하여 입력 가능하도록 설정
- 각 필드의 세부 항목은 detail 값으로 저장되도록 array 변수 설정
- collectCertificateData(): 자격증/어학/수상 부분 선택 가능하도록 드롭다운 설정
- showCertificateFields(): 드롭다운에서 선택한 필드 항목이 보여지도록 블록 설정
- renderCertificateList(): 각 필드(자격증/어학/수상)의 입력 form이 형성되도록 div 생성
- editCertificate(): 수정 버튼 클릭 시 기존 저장된 입력 사항 다시 불러오기
- savecertificate(): 모든 항목의 입력이 다 완료되었을 시 localStorage에 저장되도록 설정, 빈 항목이 있을 시 alert 출현

자격/어학/수상

[자격증] 정보처리기능사 · 한국산업인력공단 (2025-04)

+추가

수정

삭제

[어학] 영어 TOEIC 840 (취득)

수정

삭제

[수상] 장려상 · XX시 공모전 위원회 (2025-05-27)

수정

삭제

취업우대사항



기능



```
// 6. 취업우대사항
if (resume.sections && resume.sections.preference) {
  const list = document.getElementById('preference-list');
  const prefdta = resume.sections.preference;

  // 객체인 경우 배열로 감싸기
  const preferences = Array.isArray(prefdta) ? prefdta : [prefdta];

  preferences.forEach(pref => {
    console.log(`preference item: ${pref}`);
    const card = document.createElement('div');
    card.className = 'pref-card';
    let detail = '';

    if (pref.type === 'veteran' || pref.type === '보훈대상') {
      detail = `[보훈] ${pref.v_type} ${pref['보훈구분']} (증서: ${pref.v_cert} || ${pref['증명서 번호']})`;
    } else if (pref.type === 'military' || pref.type === '병역대상') {
      const statusMap = {
        'served': '군필',
        'exempt': '면제',
        'unserved': '미필'
      };

      if (pref.m_status === 'served' || pref.m_status === '군필') {
        detail = `[군필] ${pref.m_branch} ${pref.m_rank} (${pref.m_start} ~ ${pref.m_end} / ${pref.m_reason})`;
      } else {
        detail = `[병역] ${statusMap[pref.m_status]} ${pref.m_status}`;
      }
    } else if (pref.type === 'military' || pref.type === '병역대상') {
      detail = `[병역] ${pref.m_status}`;
    } else if (pref.type === 'employment' || pref.type === '고용지원금') {
      detail = `[지원금] ${pref.e_type}`;
    }

    if (detail) {
      card.innerHTML = `${detail}`;
      list.appendChild(card);
    }
  });
}

function savePreference() {
  const type = document.getElementById('preferenceType').value;
  if (!type) {
    alert('구분 선택');
    return;
  }

  const obj = { type };

  if (type === 'veteran') {
    obj.v_type = document.querySelector('[name="v_type"]').value.trim();
    obj.v_cert = document.querySelector('[name="v_cert"]').value.trim();

    if (!obj.v_type || !obj.v_cert) {
      alert('모든 항목 입력');
      return;
    }
  } else if (type === 'military') {
    obj.m_status = document.querySelector('[name="m_status"]').value.trim();

    if (!obj.m_status) {
      alert('병역 구분 입력');
      return;
    }
  }

  // 군필일 경우에만 추가 필드 입력 받기
  if (obj.m_status === 'served') {
    obj.m_start = document.querySelector('[name="m_start"]').value.trim();
    obj.m_end = document.querySelector('[name="m_end"]').value.trim();
    obj.m_branch = document.querySelector('[name="m_branch"]').value.trim();
    obj.m_rank = document.querySelector('[name="m_rank"]').value.trim();
    obj.m_reason = document.querySelector('[name="m_reason"]').value.trim();

    if (!obj.m_start || !obj.m_end || !obj.m_branch || !obj.m_rank || !obj.m_reason) {
      alert('군필 항목 모두 입력하세요');
      return;
    }
  }
}
```

```
function savePreference() {
  const type = document.getElementById('preferenceType').value;
  if (!type) {
    alert('구분 선택');
    return;
  }

  const obj = { type };

  if (type === 'veteran') {
    obj.v_type = document.querySelector('[name="v_type"]').value.trim();
    obj.v_cert = document.querySelector('[name="v_cert"]').value.trim();

    if (!obj.v_type || !obj.v_cert) {
      alert('모든 항목 입력');
      return;
    }
  } else if (type === 'military') {
    obj.m_status = document.querySelector('[name="m_status"]').value.trim();

    if (!obj.m_status) {
      alert('병역 구분 입력');
      return;
    }
  }

  // 군필일 경우에만 추가 필드 입력 받기
  if (obj.m_status === 'served') {
    obj.m_start = document.querySelector('[name="m_start"]').value.trim();
    obj.m_end = document.querySelector('[name="m_end"]').value.trim();
    obj.m_branch = document.querySelector('[name="m_branch"]').value.trim();
    obj.m_rank = document.querySelector('[name="m_rank"]').value.trim();
    obj.m_reason = document.querySelector('[name="m_reason"]').value.trim();

    if (!obj.m_start || !obj.m_end || !obj.m_branch || !obj.m_rank || !obj.m_reason) {
      alert('군필 항목 모두 입력하세요');
      return;
    }
  }
}
```

- preferences 항목에서 type이 보훈대상/병역대상/고용 지원금으로 각각 구분되도록 드롭다운 구성
- type == military(병역대상) 선택 시, 군필/미필/면제가 구분되도록 statusMap 변수 설정 후 드롭다운 구성
- m_status == 군필일 시, 상세정보(복무기간/군별/계급/전역정보) 입력 설정 후 detail 변수에 저장
- savePreference(): 선택한 항목의 상세정보 모두 입력 시 저장 버튼 클릭으로 localStorage에 내용이 저장되도록 설정, 빈 내용 존재 시 alert 표시
- editPreference(): 내용 수정 클릭 시 기존 저장된 내용이 표시되도록 설정

취업우대사항

[보훈] 국가유공자 (증서: XXXXX)

[수정](#) [삭제](#)

[군필] 공군-병장 (2018-11~2020-09) / 만기전역

[수정](#) [삭제](#)

[지원금] 청년고용지원금

[수정](#) [삭제](#)

+추가

포트폴리오 폼



```
function showPortfolioForm() {…}
//포트 폴리오 폼 생성
function removePortfolioForm(button) {…}
//포트 폴리오 폼 제거
function savePortfolio(button) {
  const entry = button.closest(".portfolio-entry");
  const summary = entry.querySelector(".portfolio-summary");

  const inputs = entry.querySelectorAll("input, textarea");
  const fileInput = entry.querySelector(".file-input");

  const values = {};
  inputs.forEach(input => {
    if (input.type !== "file") {
      values[input.placeholder] = input.value;
    }
  });

  // 파일 이름 목록 생성
  const fileNames = [];
  for (let i = 0; i < fileInput.files.length; i++) {
    fileNames.push(fileInput.files[i].name);
  }

  summary.innerHTML =
`

<h4>${values['프로젝트명']} || ''</h4>
<p><strong>설명:</strong> ${values['프로젝트 설명']} || ''</p>
${fileNames.length > 0 ? `<p><strong>첨부파일:</strong> ${fileNames.join(', ')}` : ''}
<div style="display: flex; justify-content: flex-end; gap: 8px;">
  <button onclick="editPortfolio(this)" class="cencle_save_btn">수정</button>
  <button onclick="removePortfolioForm(this)" class="cencle_save_btn">삭제</button>
</div>


`;

  if (!values['프로젝트명'] || !values['프로젝트 설명']) {
    alert('프로젝트명, 프로젝트 설명은 필수 입력 항목입니다.');
    return;
  }

  inputs.forEach(input => input.style.display = "none");
  const formButtons = entry.querySelector(".form-buttons");
  if (formButtons) formButtons.style.display = "none";

  summary.style.display = "block";
}
//포트 폴리오 주제
function editPortfolio(button) {…}
```

- portfolioFields: 각 항목의 필드 세부 정보 배열
- collectPortfolioData(): 드롭다운 생성
- showPortfolioFields(): 선택한 항목에 따라 입력 필드 표시
- showPortfolioForm(): 입력 폼 생성
- editPortfolio(button): 저장된 정보를 수정 가능하도록 표시
- 포트폴리오 파일 저장
- savePortfolio(button): 필수 항목 입력 후 요약 카드로 저장

포트폴리오

팀 멤버

설명: 멤버 애플리케이션 제작

첨부파일: 팀멤버_앱_PPT.pdf

경력 기술서



기능

```
function updateCharCount() {
  const textArea = document.getElementById("careerDescriptionText");
  if (!textArea) return; // 텍스트 영역이 아직 없으면 그냥 할 수 종료

  const text = textArea.value;
  const total = text.length;
  const noSpace = text.replace(/\s/g, "").length;
  const byteSize = new Blob([text]).size;
  const byteNoSpace = new Blob([text.replace(/\s/g, "")]).size;

  const totalChars = document.getElementById("totalChars");
  const totalBytes = document.getElementById("totalBytes"); ...

  if (totalChars) totalChars.innerText = total;
  if (totalBytes) totalBytes.innerText = byteSize;
  if (noSpaceChars) noSpaceChars.innerText = noSpace;
  if (noSpaceBytes) noSpaceBytes.innerText = byteNoSpace;
}

function savedescription() {
  const text = document.getElementById("careerDescriptionText").value;
  alert("내용을 입력하세요.");
  return;
}

const summaryContainer = document.createElement("div");
summaryContainer.className = "career-desc-card";
summaryContainer.innerHTML =
`<div style="white-space: pre-wrap;">${text}</div>
<div style="margin-top: 10px; text-align: right;">
  <button onclick="editDescription(this)" class="cencle_save_btn">수정</button>
  <button onclick="removeDescription(this)" class="cencle_save_btn">삭제</button>
</div>
`;

document.getElementById("career-description-list").appendChild(summaryContainer);
document.getElementById("careerDescriptionText").value = "";
document.getElementById("career-description-form").style.display = "none";
updateCharCount();
}

function editDescription(button) { ... }
function removeDescription(button) { ... }

const text = document.getElementById("careerDescriptionText").value;

function insertTemplate(type) {
  const textarea = document.getElementById("careerDescriptionText");
  let content = "";

  switch (type) {
    case "project":
      content =
        `1) 프로젝트명:
        - 연계/소속회사 : (연계/소속회사가 없을 경우, 해당 항목을 지우고 작성하세요)
        - 주요 업무 : 백엔드 담당
        - 담당 역할 :
        - 기술 스택 : (운영체제, 개발언어, 데이터베이스 등)
        - 업무 기간 : YYYY.MM ~ YYYY.MM (약 N개월)
        - 개발 인원 :
        - 상세 내용 :`;
```

- 연계/소속회사 : (연계/소속회사가 없을 경우, 해당 항목을 지우고 작성하세요)
- 수행 기간 : YYYY.MM ~ YYYY.MM (약 N개월)
- 주요 역할 : 화면 설계 및 서비스 기획
- 업무 성과 : ;
break;

```
    case "dev":
```

- insertTemplate(type) 사용자가 경력기술서를 쉽게 작성할 수 있도록, 유형별 템플릿을 자동으로 삽입.
- 전달된 type 값에 따라 project, dev, design, sales 중 하나의 템플릿 문자열이 선택됨.
- 선택된 템플릿은 #careerDescriptionText textarea에 삽입되며, 이후 updateCharCount() 가 호출되어 실시간 글자 수 계산이 반영됨.

경력기술서

1) 프로젝트명:

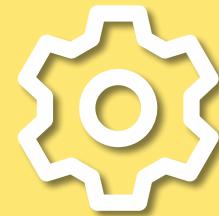
- 연계/소속회사 : (연계/소속회사가 없을 경우, 해당 항목을 지우고 작성하세요)
- 주요 업무 : 백엔드 담당
- 담당 역할 :
- 기술 스택 : (운영체제, 개발언어, 데이터베이스 등)
- 업무 기간 : YYYY.MM ~ YYYY.MM (약 N개월)
- 개발 인원 :
- 상세 내용 :

프로젝트형 개발 기술형 디자이너형 영업 성과형

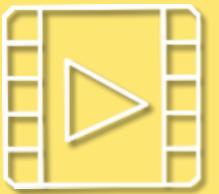
총 글자수 201자 / 365 byte

취소 저장

자기소개서



가능



```
# ----- 맞춤법 -----
@csrf_exempt
@require_POST
def spellcheck(request):
    text = json.loads(request.body).get("text", "")

    resp = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "너는 한국어 교정 도우미야. 맞춤법·띄어쓰기·오탈자만 고쳐서 원문 형식 그대로 돌려줘."},
            {"role": "user", "content": text}
        ],
        temperature=0
    )
    corrected = resp.choices[0].message.content.strip()
    return JsonResponse({"result": corrected})

# ----- AI 첨삭 -----
@csrf_exempt
@require_POST
def proofread(request):
    text = json.loads(request.body).get("text", "")

    resp = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": (
                "너는 인사담당자 관점의 첨삭 코치야. 문맥은 유지하되 "
                "① 맞춤법 ② 문장 간 흐름 ③ 구체적·적극적 어휘 제안 세가지를 반영해 "
                "수정본만 돌려줘."
                "작업에 따른 전문용어를 포함해서 작성해줘"
            )},
            {"role": "user", "content": text}
        ],
        temperature=0.3
    )
    improved = resp.choices[0].message.content.strip()
    return JsonResponse({"result": improved})
```

gpt-4o-mini : 고속, 저비용 모델
역할 지시 : "너는 한국어 교정 도우미야"
(프롬프트 엔지니어링)
교정 범위 : 오타, 띄어쓰기, 맞춤법
temperature=0 : 정확도 우선, 창의성 배제

gpt-4o-mini : 고속, 저비용 모델
역할 지시 : "너는 인사 담당자 관점의 첨삭 코치야"
교정 범위 : 문맥 유지, 맞춤법, 문장 간 흐름, 구체적
적극적 어휘, 직업에 따른 전문용어 포함
temperature=0.3 : 다소 창의적인 어휘 표현 허용

초기 계획 파인 투닝 → 프롬프트 엔지니어링
보유중인 데이터의 양이 부족 (첨삭의 경우 합격 자
소서의 내용을 임의로 수정 되지 않은 자소서 데이
터로 만들어야 했음)
비용 문제 : 파인튜닝의 경우 투닝하는데 사용되는
비용 + 검색 비용 - 비용이 부담이 크지만 프롬프트
엔지니어링은 비용이 많이 들지 않음

자기소개서



기능



자기소개서

+추가

홍길동 자기소개서

나는 컴퓨터를 잘합니다

총 글자수 12자 / 32 byte

맞춤법 검사

취소

AI 첨삭

저장

자기소개서

+추가

홍길동 자기소개서

저는 컴퓨터 활용 능력이 뛰어납니다.

총 글자수 20자 / 50 byte

맞춤법 검사

취소

AI 첨삭

저장

이력서 PDF 변환



```
function previewResume() {
  if (selectedResumeId) {
    window.open(`/preview_pdf/${selectedResumeId}` , '_blank');
    return;
  }
  const title = document.getElementById('resume-title').value;
  const name = document.getElementById('name').value;
  const gender = document.getElementById('gender').value;
  const birthdate = document.getElementById('birthdate').value;
  const email = document.getElementById('email').value;
  const phone = document.getElementById('phone').value;
  const address = document.getElementById('address').value;
  const detailAddress = document.getElementById('detail-address').value;

  const sectionCheckboxStates = getSectionCheckboxStates();
  const sections = { ... }
```

이력서 정보 수집 (데이터를 JSON 형태로 수집)

```
const formData = new URLSearchParams();
formData.append("title", title);
formData.append("name", name);
formData.append("gender", gender);
formData.append("birthdate", birthdate);
formData.append("email", email);
formData.append("phone", phone);
formData.append("address", address);
formData.append("detail-address", detailAddress);

formData.append("sections_json", JSON.stringify(sections));

fetch('/preview_pdf', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
    'x-CSRFToken': getCookie('csrftoken')
  },
  body: formData
})
  .then(response => response.text())
  .then(html => {
    const win = window.open("", "_blank");
    win.document.open();
    win.document.write(html);
    win.document.close();
  })
  .catch(err => {
    alert("미리보기 실패: " + err);
  });
}
```

서버에 POST 요청을 보내 HTML 미리보기 요청
응답으로 받은 HTML을 새 창에 랜더링하여 미리
보기 화면을 출력, 이미 저장된 이력서가 있으면
해당 ID로 GET 미리보기 실행.

이력서 PDF 변환



기능



```
<script src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10.1/html2pdf.bundle.min.js"></script>
<script>
window.onload = function () {
  const resume = document.body;
  html2pdf().set({
    margin: 0,
    filename: '이력서.pdf',
    image: { type: 'jpeg', quality: 0.98 },
    html2canvas: { scale: 2 },
    jsPDF: { unit: 'mm', format: 'a4', orientation: 'portrait' }
  }).from(resume).save();
}
</script>
```

PDF 변환 페이지

- html2pdf.js CDN을 통해 PDF 변환 도구를 불러옵니다.

- HTML > 이미지 > PDF 순서로 변환
- PDF 변환 옵션 지정
- PDF 변환 시작
- 즉시 파일 다운로드

홍길동 이력서



기본 정보

이름: 홍길동
성별: 남성
생년월일: 1889.11.11
이메일: 1234@google.com
연락처: 010-1234-5678
주소: 경기도 화성시 xx동 123-123 123동 456호

학력

서울대 (2025-05-06 ~ 2025-06-04) (~)
전공: 컴퓨터공학과
학점: 4.1



구현 영상

