

## 결과 보고서

### 시험 글쓰기 동료 첨삭 플랫폼

Co-Write

소프트웨어학과	2013310448	김진옥
전자전기공학부	2012310132	김태홍
컴퓨터공학과	2012311683	이윤열
소프트웨어학과	2014312893	최동규
컴퓨터공학과	2015311777	최민진
소프트웨어학과	2015313050	한성혜

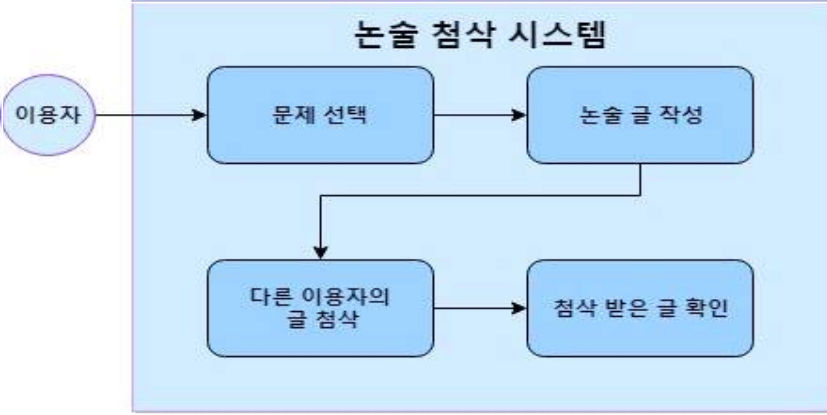
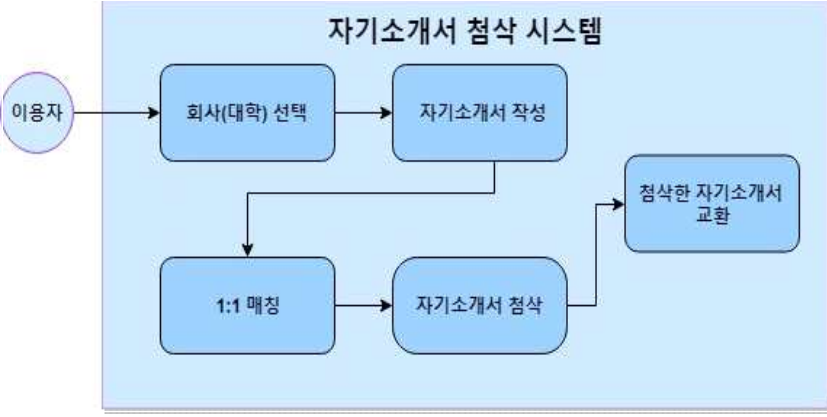
## 목차

<b>1. 과제 요약</b>	<b>1</b>
<b>2. 과제 개요</b>	<b>4</b>
<b>3. 과제 추진 배경, 필요성, 목적</b>	<b>5</b>
3-1. 과제 추진 배경	5
3-2. 과제의 필요성	7
3-3. 과제의 목적	9
<b>4. 과제 해결 방안, 수행 일정</b>	<b>10</b>
4-1. 과제 해결 방안	10
4-2. 과제 수행 일정	12
<b>5. 과제 세부내용</b>	<b>12</b>
5-1. 데이터베이스	12
5-2. 로그인	16
5-3. 논술 검색	20
5-4. 자기소개서 검색	25
5-5. 관리자 페이지	30
<b>6. 기대 효과</b>	<b>31</b>
<b>7. 기타</b>	<b>32</b>
7-1. 한계점	32
7-2. 개선 사항	32
7-3. 업무 분담	33
<b>8. 참고문헌</b>	<b>34</b>

## 1. 과제 요약

Capstone Design 과제 결과보고서 요약	
과 제 명	시험 글쓰기 동료 첨삭 플랫폼 'Co-Write'
팀 구성원 및 역할	<p>김진욱 : 자기소개서 front-end 개발 (HTML)</p> <p>김태홍 : 논술 back-end 개발 (nodejs)</p> <p>이윤열 : 로그인, 관리자 페이지 개발 (nodejs)</p> <p>최동규 : 데이터베이스 구축 (MySQL)</p> <p>최민진 : 논술 front-end 개발 (HTML)</p> <p>한성혜 : 자기소개서 back-end 개발 (nodejs)</p>
개발동기 목적 및 필요성	<p>자기소개서는 거의 모든 기업의 공개 채용 전형에서 반드시 제출해야 하는 서류 중 하나이다. 따라서 자기소개서는 취업을 준비하기 위해 가장 먼저 작성해야 할 중요한 문서라고 할 수 있다. 하지만 기본적인 맞춤법조차 틀린 채로 자기소개서를 제출하는 지원자가 의외로 많다. 취업준비생들은 평소 글쓰기에 익숙하지 않아서 오타, 맞춤법 등의 실수를 자주 저지르고 시간 여유조차 많지 않다. 따라서 자기소개서 작성에 많은 어려움을 겪을 수밖에 없다.</p> <p>최근 들어 취업준비생을 위한 다양한 취업컨설팅 서비스가 인기를 끌고 있다. 자기소개서 첨삭, 면접 속성 지도, 취업 특강 등으로 취업준비생의 부담을 덜어주는 것이다. 하지만 취업컨설팅 서비스 이용자의 만족도는 높지 않은 편이다. 가장 큰 이유는 비싼 수강료에 비해 충분한 서비스를 제공받지 못하기 때문이다. 환불 기준이나 계약 사항과 같은 중요한 정보를 소비자에게 충분히 알려주지 않는다는 문제점도 있다.</p> <p>또한 일부 업체들은 컨설턴트의 신상을 공개하지 않고 첨삭 멘토의 수와 첨삭 속도만을 내세워 홍보하기도 한다. 이러한 경우 첨삭 담당 컨설턴트들의 실력을 정확히 확인하기가 어렵다. 신원이 공개되지 않은 첨삭 멘토의 실력을 어떻게 믿을 수 있을까? 신뢰할 수 없는 소수의 '자칭' 전문가들에게만 자기소개서 첨삭을 맡기는 것은 매우 위험한 일이다.</p> <p>결론적으로, 아래의 세 가지 기준을 모두 만족하는 취업준비생을 위한 새로운 자기소개서 첨삭 서비스가 필요하다.</p> <ol style="list-style-type: none"> <li>1. 신뢰할 수 있는 전문가에게 첨삭을 받아야 한다.</li> <li>2. 첨삭의 횟수는 많으면 많을수록 좋다.</li> <li>3. 첨삭에 드는 비용은 적으면 적을수록 좋다.</li> </ol>

	<p>우리는 같은 회사에 지원한 취업준비생끼리 서로의 자기소개서를 첨삭해 주는 플랫폼을 제안하고자 한다. 이 플랫폼의 핵심은 소수의 전문가 대신 다수의 동료들에게 무료로 다양한 첨삭을 받는 것이다. 전문가에 준하는 동료들의 지성을 활용한다면, 앞서 언급한 세 가지 기준을 모두 만족하는 훌륭한 자기소개서 첨삭 서비스를 제공할 수 있을 것이다.</p> <p>또한 이 플랫폼은 자기소개서뿐만 아니라, 다양한 글쓰기 첨삭에 활용할 수 있다. 대학교 입학 논술시험을 그 예시로 들 수 있다. 대부분의 대입 논술 교육은 수험생이 목표로 하는 학교의 기출문제를 풀고 전문가가 첨삭을 해주는 방식으로 진행된다. 이러한 논술 첨삭 서비스는 앞서 언급한 기존의 자기소개서 첨삭 서비스와 똑같은 문제점을 안고 있다. 따라서 대입 논술에 이 플랫폼을 활용한다면 기존 논술 첨삭 서비스의 문제점을 충분히 해결할 수 있을 것이다.</p>
<p><b>과제 해결 방안 및 과정</b></p>	<p>(1) 대입 논술 첨삭</p> <p>대입 논술의 경우 이용자는 대입 논술 시험을 대비하고 있는 학생을 대상으로 한다. 논술 시험은 대부분 문제가 미리 주어지지 않고, 실제 시험에서 정해진 문제에 따라 본인의 글을 완성하여 제출하게 된다. 이를 대비하기 위해 학생들은 시험에 대비한 연습으로 기출 문제를 풀어보고 쓴 글에 대하여 첨삭을 받으며 도움을 얻고자 한다. 즉, 첨삭 받은 글은 본인의 학습 용도로만 활용된다. 논술 시험을 준비하는 학생들은 반복 학습을 위해 많은 문제를 풀어야 할 뿐만 아니라, 자신이 쓴 답안에 적절한 피드백을 받는 것 또한 중요하다.</p> <p>대입 논술 첨삭 시스템 이용자는 목표 대학의 기출 문제를 풀고 자신의 답안을 공유한 후 다른 동료의 첨삭을 받아야 한다. 이러한 시스템을 가장 효과적으로 운영하기 위해 문제 은행 방식으로 플랫폼을 제공한다. 문제 은행 방식은 본인이 원하는 대학과 년도 그리고 문제를 선택해 글을 작성하고 첨삭을 받는 형태이다.</p> <p>(2) 자기소개서 첨삭</p> <p>자기소개서의 경우 이용자는 대입 혹은 입사의 서류 만기일까지 자기소개서를 제출해야하는 사람이다. 자기소개서는 회사 혹은 대학별로 문항이 정해져 있다. 문항이 공개되기 이전부터 자기소개서 작성을 준비하는 경우에도 자기소개서 양식은 이전에 비해 크게 바뀌지 않는 편이다. 이용자들은 자기소개서를 작성하고 첨삭을 통해 문제점을 수정하여 이를 그대로 제출하게 된다. 즉, 첨삭 받은 글을 그대로 활용하기 때문에 첨삭자간의 신뢰성이 더욱 중요시 된다.</p> <p>자기소개서 첨삭 시스템 이용자의 경우 신뢰성 확보를 위하여 1:1 매칭을 통해 서로의 정보를 확인한 뒤 자기소개서 첨삭을 주고받게 한다. 한 번의 1:1 첨삭이 끝난 후 추가적인 첨삭을 원한다면, 다시 1:1 매칭을 통해 다른 동료와 동일한 방법으로 첨삭을 진행할 수 있다.</p>

	(3) 과제 수행 과정													
	No.	수행내용	수행일정 (단위 : 주차)											
			3	4	5	6	7	8	9	10	11	12	13	14
	1	계획수립, 자료조사	■	■										
	2	서비스 기획		■	■									
	3	제안서 작성, 계획 발표 준비			■	■								
	4	데이터베이스 구축					■	■	■	■				
	5	웹 front-end 개발					■	■	■	■				
	6	웹 back-end 개발					■	■	■	■				
	7	중간 발표 준비								■	■			
	8	테스팅, 디버깅								■	■	■	■	■
	9	최종 발표 준비											■	■
대표적 결과물 (도면, 사진 등)	<div style="text-align: center;"> <p><b>논술 첨삭 시스템</b></p>  <p>논술 첨삭 시스템 모형도</p> </div>													
	<div style="text-align: center;"> <p><b>자기소개서 첨삭 시스템</b></p>  <p>자기소개서 첨삭 시스템 모형도</p> </div>													
기업연계형 프로젝트	<input type="checkbox"/> 주제 제공 <input type="checkbox"/> 멘토링 <input type="checkbox"/> 샘플(부품) 제공 <input type="checkbox"/> 기타(현장견학 등)													
	없음													

## 2. 과제 개요

Capstone Design 과제 결과보고서				
과 목 명	종합설계프로젝트			
과 제 명	시험 글쓰기 동료 첨삭 플랫폼 'Co-Write'			
팀 명	Co-Write			
지도교수 (과제책임자)	학과(부)	소프트웨어학과	성명	이 진 규
팀장 (대표학생)	학과(부)/학년	소프트웨어학과 / 4학년	성명	김 진 옥
	E-mail	stork_gm@naver.com	휴대전화	010-4923-6653
팀원	학과(부)/학년	전자전기공학부 / 4학년	성명	김 태 홍
	학과(부)/학년	컴퓨터공학과 / 4학년	성명	이 윤 열
	학과(부)/학년	소프트웨어학과 / 4학년	성명	최 동 규
	학과(부)/학년	컴퓨터공학과 / 4학년	성명	최 민 진
	학과(부)/학년	소프트웨어학과 / 4학년	성명	한 성 혜
프로젝트 참여기업	기업명		주생산품	
	주소			
과 제 수 행 비	구 분	현 금	현 물	
	국 비(LINC+)	0 원		
	기 업 체	-		
	합 계	0 원		
과제기간	2018 년 9 월 3 일 - 2018 년 12 월 21 일 (4 개월)			
<p>위와 같이, 산학협력선도대학(LINC+)육성사업단의 Capstone Design 지원사업 과제결과보고서를 제출합니다.</p> <p style="text-align: right;">2018. 12. 12.</p> <p style="text-align: right;">팀 장: 김 진 옥 (인)</p> <p style="text-align: right;">지도교수: 이 진 규 (인)</p> <p style="text-align: right;">LINC+사업단장 귀하</p>				

### 3. 과제 추진 배경, 필요성, 목적

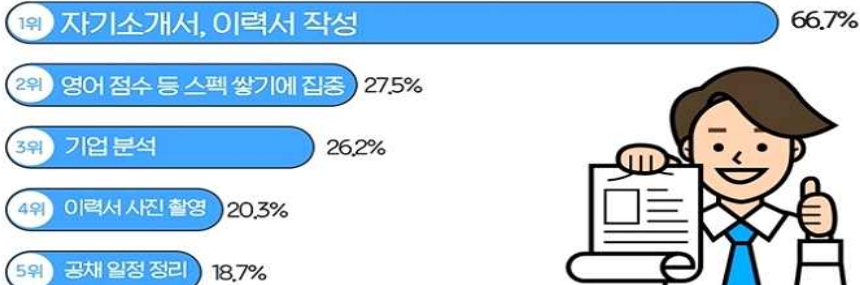
#### 3.1. 과제 추진 배경

통계청의 조사에 따르면 2017년 11월 기준 청년 실업률은 9.2%를 기록했다. 또한 블라인드 평가 방식을 도입하는 등 기업들의 채용 방식이 다양해지고 있다. 취업난이 지속되고 취업을 위해 해야 할 일들이 많아지는 가운데 취업준비생들의 고민과 스트레스는 나날이 늘어만 간다. 이렇게 어려운 상황 속에서 취업을 위해 가장 먼저 준비해야 할 일은 무엇일까?

잡코리아는 올해 상반기 신입 공개 채용에 지원한 취업준비생 866명을 대상으로 설문조사를 진행했다. 이 설문조사에서 공개 채용 준비 과정에서 가장 먼저 한 일이 무엇인냐는 질문에 3분의 2에 해당하는 66.7%의 응답자가 자기소개서와 이력서 작성이라고 답했다.

#### 신입공채 취준생, 공채 준비 ‘자소서 작성부터’

Q. 상반기 신입공채를 준비하며 가장 먼저 한 일은 무엇인가요?



\* 상반기 신입공채 지원 취준생 866명 대상 설문조사 결과, 자료제공: 잡코리아

<그림 1> 취업준비생이 신입 공개 채용 준비 과정에서 가장 먼저 한 일

자기소개서는 거의 모든 기업의 공개 채용 전형에서 반드시 제출해야하는 서류 중 하나이다. 따라서 자기소개서는 취업을 준비하기 위해 가장 먼저 작성해야 할 중요한 문서라고 할 수 있다. 하지만 기본적인 맞춤법조차 틀린 채로 자기소개서를 제출하는 지원자가 의외로 많다.

2017년 10월 잡코리아와 알바몬은 ‘자기소개서 내 맞춤법 실수’라는 주제로 설문조사를 진행했다. 이 조사에서 맞춤법이 틀린 자기소개서를 받아본 경험이 있냐는 질문에 인사담당자 733명 중 무려 90.7%가 그렇다고 답했다.

## Q 맞춤법 틀린 자기소개서 등을 받아본 경험?



그래픽 tong+

자료 취업포털 잡코리아 대상 인사담당자 733명, 취업준비생 796명

<그림 2> 맞춤법이 틀린 자기소개서를 받아본 경험

그렇다면 지원자들이 자기소개서를 작성할 때 맞춤법 실수를 하는 이유는 무엇일까? 취업준비생 796명 중 절반이 넘는 56.1%는 작성 당시에는 자신이 쓴 표현이 틀린 줄 전혀 몰랐기 때문이라고 답했다. 마감 기한에 쫓겨 작성하느라 미처 알아채지 못했다는 대답도 26.6%를 차지했다. 꼼꼼하지 않은 성격 때문에 맞춤법 실수를 놓친 지원자도 17.3%에 달했다.

## Q 지원서 내 맞춤법 실수를 하는 이유는?



그래픽 tong+

자료 취업포털 잡코리아 대상 인사담당자 733명, 취업준비생 796명

<그림 3> 자기소개서 작성 시 맞춤법 실수를 하는 이유



위의 설문조사 결과에서 알 수 있듯이, 취업준비생들은 평소 글쓰기에 익숙하지 않아서 오타, 맞춤법 등의 실수를 자주 저지르고 시간 여유조차 많지 않다. 따라서 자기소개서 작성에 많은 어려움을 겪을 수밖에 없다.

2018년 4월 잡코리아와 알바몬은 '취업준비가 막막하게 느껴진 적이 있는가?'에 대해 취업준비생 1187명을 대상으로 설문조사를 실시했다. 이들 중 거의 대부분인 98.5%가 취업준비를 막막하게 느껴본 적이 있다고 답했다.

또한 74.6%의 응답자가 족집게 과외와 같은 취업 도우미 서비스가 필요하다는 의견을 제시했다. 구체적으로 도움을 받고 싶은 부문으로는 이력서와 자기소개서 첨삭이 29.5%로 가장 많았고, 자신의 진로 적성 파악과 면접 속성 과외가 그 뒤를 이었다.

## 취준생 74.6%, 취업 도우미 필요해

※산업직 취업준비생 1,187명 대상 조사. 자료:잡코리아×알바몬



<그림 4> 취업준비생이 도움을 받고 싶은 부문

위 설문조사들의 내용을 종합해 보면 아래와 같이 세 가지로 정리할 수 있다.

- (1) 자기소개서는 취업준비생이 가장 먼저 준비해야 하는 중요한 서류이다.
- (2) 하지만 취업준비생에게 자기소개서 작성은 매우 까다로울 수밖에 없다.
- (3) 따라서 자기소개서 작성에 도움을 줄 수 있는 서비스가 필요하다.

### 3.2. 과제의 필요성

최근 들어 취업준비생을 위한 다양한 취업컨설팅 서비스가 인기를 끌고 있다. 자기소개서 첨삭, 면접 속성 지도, 취업 특강 등으로 취업준비생의 부담을 덜어주는 것이다. 하지만 취

업컨설팅 서비스 이용자의 만족도는 높지 않은 편이다. 가장 큰 이유는 비싼 수강료에 비해 충분한 서비스를 제공받지 못하기 때문이다. 환불 기준이나 계약 사항과 같은 중요한 정보를 소비자에게 충분히 알려주지 않는다는 문제점도 있다.

한국소비자원은 최근 1년간 취업컨설팅 서비스를 이용한 경험이 있는 2~30대 취업준비생 300명을 대상으로 설문조사를 실시했다. 이 설문조사에 따르면, 42.0%의 응답자가 취업컨설팅 서비스의 과도하게 비싼 이용료가 불만이라고 답했다. 일대일 맞춤 컨설팅이 기대한 만큼 원활하게 이루어지지 않아 만족스럽지 못했다는 답변이 33.7%로 그 뒤를 이었다. 취업컨설팅 서비스의 결과물이 기대 이하였다는 응답도 32.7%에 달했다. 반면 특별한 불만 사항이 없었다는 답변은 14.3%에 불과했다.



<그림 5> 취업컨설팅 서비스 불만족 이유

PC나 모바일 기반의 온라인 자기소개서 첨삭 서비스도 성행하고 있다. 이들은 오프라인 서비스에 비해 상대적으로 저렴한 가격을 내세우고 있다. 하지만 대부분의 취업준비생들은 여러 개의 자기소개서를 작성하기 때문에 그 비용조차 만만치 않다.

**【자소서 첨삭업체별 비교】**

업체명	첨삭비용	컨설턴트 보상금	컨설턴트 프로필	기타
탐티어	1000자당 4만9000원	비공개	서울대 재(휴)학 및 졸업생	첨삭 평가 기준점 미하 시 컨설턴트 자격 상실
코멘토	500자당 2750원	건당 1000~2500원	비공개(80% 대기업 재직자)	건당 최대 3명 첨삭 가능, 의뢰인이 미준 2개 선택
자소서닷컴	1500자당 1만원	전문업체와의 제휴	비공개(첨삭업체 강사진)	글자 수 추가 및 첨삭시간 단축 옵션 선택 가능
취업대학교	무료	없음	비공개	항목당 1~2줄짜리 짧은 첨삭

<그림 6> 자기소개서 첨삭업체별 비교

또한 일부 업체들은 컨설턴트의 신상을 공개하지 않고 첨삭 멘토의 수와 첨삭 속도만을 내세워 홍보하기도 한다. 이러한 경우 첨삭 담당 컨설턴트들의 실력을 정확히 확인하기가 어렵다. 신원이 공개되지 않은 첨삭 멘토의 실력을 어떻게 믿을 수 있을까? 신뢰할 수 없는 소수의 '자칭' 전문가들에게만 자기소개서 첨삭을 맡기는 것은 매우 위험한 일이다.

결론적으로, 아래의 세 가지 기준을 모두 만족하는 취업준비생을 위한 새로운 자기소개서 첨삭 서비스가 필요하다.

- (1) 신뢰할 수 있는 전문가에게 첨삭을 받아야 한다.
- (2) 첨삭의 횟수는 많으면 많을수록 좋다.
- (3) 첨삭에 드는 비용은 적으면 적을수록 좋다.

### 3.3. 과제의 목적

앞서 언급했던 세 가지 기준을 모두 만족하는 자기소개서 첨삭 서비스를 만드는 것은 현실적으로 매우 어렵다. 하지만 (1)번 기준에서 '신뢰할 수 있는 전문가'의 범위를 조금만 더 넓혀 본다면, 불가능한 일은 아니다.

A라는 회사의 자기소개서를 가장 잘 알고 있는 사람은 누구일까? 아마 대부분의 사람들은 첨삭 전문가라고 답할 것이다. 물론 컨설턴트의 수많은 자기소개서 첨삭 경험과 내공은 무시할 수 없다. 하지만 전문가들은 A회사뿐만 아니라 다른 수많은 회사들의 자기소개서를 첨삭해야 한다. 따라서 특정 회사의 자기소개서에 대해 자세히 아는 것에는 한계가 있다.

오히려 컨설턴트보다 직접 자기소개서를 제출해야 하는 지원자가 A회사의 자기소개서를 더 잘 알고 있지 않을까? 지원자들은 자기소개서 작성을 위해 많은 것을 준비하고 공부한 사람들이다. 심지어 직접 작성한 A회사 샘플 자기소개서까지 가지고 있다. 즉 A회사의 지원자들은 적어도 A회사의 자기소개서에 한해서는 전문가에 준하는 지식과 능력을 보유하고 있다는 것이다.

그리고 오타, 맞춤법 등의 사소한 실수들은 굳이 비싼 돈 들여가며 전문가에게 의뢰하지 않더라도 충분히 찾아낼 수 있다. 개발자가 프로그래밍을 할 때, 자신의 오류는 잘 찾아내지 못한다. 오히려 다른 사람이 자신의 코드를 읽어볼 때 오류를 찾아내는 경우가 더 많다. 마찬가지로 자신의 눈으로는 미처 확인하지 못한 자기소개서의 사소한 실수들을 다른 동료가 대신 발견할 수도 있을 것이다.

이러한 생각을 바탕으로, 우리는 같은 회사에 지원한 취업준비생끼리 서로의 자기소개서를 첨삭해 주는 플랫폼을 제안하고자 한다. 이 플랫폼의 핵심은 소수의 전문가 대신 다수의 동료들에게 무료로 다양한 첨삭을 받는 것이다. 전문가에 준하는 동료들의 지성을 활용한다면, 앞서 언급한 세 가지 기준을 모두 만족하는 훌륭한 자기소개서 첨삭 서비스를 제공할 수 있을 것이다.

또한 이 플랫폼은 자기소개서뿐만 아니라, 다양한 글쓰기 첨삭에 활용할 수 있다. 대학교 입학 논술시험을 그 예시로 들 수 있다. 대부분의 대입 논술 교육은 수험생이 목표로 하는 학교의 기출문제를 풀고 전문가가 첨삭을 해주는 방식으로 진행된다. 이러한 논술 첨삭 서비스는 앞서 언급한 기존의 자기소개서 첨삭 서비스와 똑같은 문제점을 안고 있다. 따라서 대입 논술에 이 플랫폼을 활용한다면 기존 논술 첨삭 서비스의 문제점을 충분히 해결할 수

있을 것이다.

과제의 궁극적인 목적은 '시험 글쓰기 동료 첨삭 플랫폼'을 만드는 것이다. 자기소개서, 대입 논술 등의 다양한 시험 글쓰기에 이 플랫폼을 적용할 수 있다. 또한 전문가에 준하는 다수의 동료들에게 다양한 첨삭을 무료로 받을 수 있기 때문에 앞서 언급했던 세 가지 기준에 모두 부합한다.

## 4. 과제 해결 방안, 수행 일정

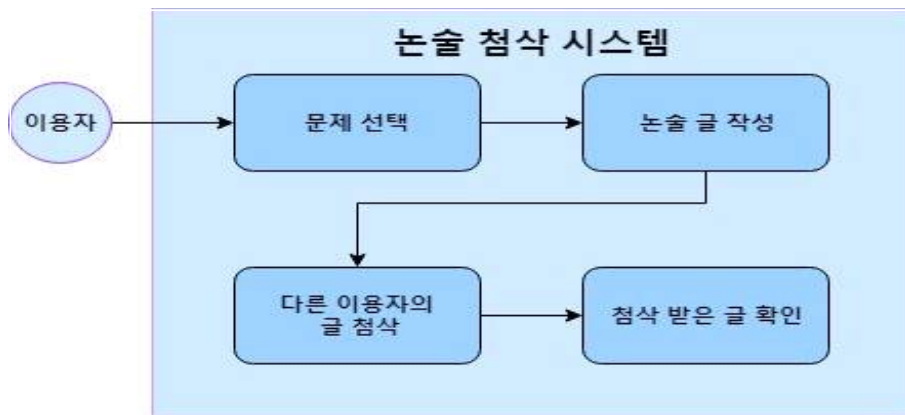
### 4.1 해결 방안

우리는 대입을 위한 논술과 자기소개서로 구분하여 첨삭 시스템을 제공하는 플랫폼을 제안한다. 이 두 글은 모두 신뢰성 있는 첨삭을 요구하지만, 글의 이용과 준비 방식 그리고 요구하는 신뢰도에서 분명한 차이가 있으므로 각 경우에 적합한 시스템을 따로 제공하게 된다.

#### (1) 대입 논술 첨삭

대입 논술의 경우 이용자는 대입 논술 시험을 대비하고 있는 학생을 대상으로 한다. 논술 시험은 대부분 문제가 미리 주어지지 않고, 실제 시험에서 정해진 문제에 따라 본인의 글을 완성하여 제출하게 된다. 이를 대비하기 위해 학생들은 시험에 대비한 연습으로 기출 문제를 풀어보고 쓴 글에 대하여 첨삭을 받으며 도움을 얻고자 한다. 즉, 첨삭 받은 글은 본인의 학습 용도로만 활용된다. 논술 시험을 준비하는 학생들은 반복 학습을 위해 많은 문제를 풀어야 할 뿐만 아니라, 자신이 쓴 답안에 적절한 피드백을 받는 것 또한 중요하다.

대입 논술 첨삭 시스템 이용자는 목표 대학의 기출 문제를 풀고 자신의 답안을 공유한 후 다른 동료의 첨삭을 받아야 한다. 이러한 시스템을 가장 효과적으로 운영하기 위해 문제 은행 방식으로 플랫폼을 제공한다. 문제 은행 방식은 본인이 원하는 대학과 년도 그리고 문제를 선택해 글을 작성하고 첨삭을 받는 형태이다.



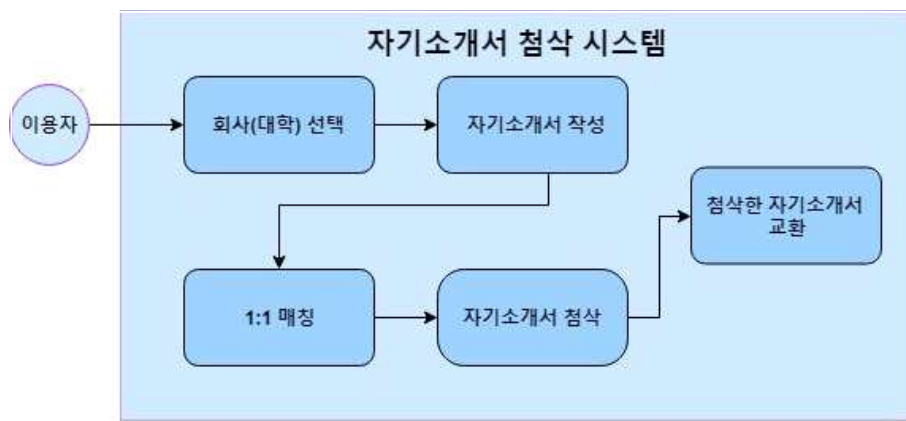
<그림 7> 논술 첨삭 시스템 모형도

논술 첨삭 시스템 사용자의 최종 목적은 본인이 작성한 글에 대한 첨삭을 받는 것이다. 따라서 이 시스템이 효과적으로 작동하기 위해서는 사용자들이 서로의 글을 첨삭하도록 유도해야 한다. 만약 다른 이용자의 글을 첨삭해 주어야만 본인의 글에 달린 첨삭을 확인할 수 있다면, 이 문제를 해결할 수 있다. 예를 들어, 자신의 글에 3개의 첨삭 댓글이 달려 있는 상태라고 가정해 보자. 하지만 사용자는 현재 첨삭 댓글의 내용을 확인할 수 없다. 만약 내용을 확인하고 싶다면, 자신도 다른 사람의 글에 3번의 첨삭을 시행해야 한다. 즉 자신의 글에 달린  $n$ 개의 첨삭 댓글을 확인하고 싶다면, 자신도  $n$ 개의 첨삭 댓글을 달아야만 하는 것이다.

## (2) 자기소개서 첨삭

자기소개서의 경우 이용자는 대입 혹은 입사의 서류 만기일까지 자기소개서를 제출해야 하는 사람이다. 자기소개서는 회사 혹은 대학별로 문항이 정해져 있다. 문항이 공개되기 이전부터 자기소개서 작성을 준비하는 경우에도 자기소개서 양식은 이전에 비해 크게 바뀌지 않는 편이다. 이용자들은 자기소개서를 작성하고 첨삭을 통해 문제점을 수정하여 이를 그대로 제출하게 된다. 즉, 첨삭 받은 글을 그대로 활용하기 때문에 첨삭자간의 신뢰성이 더욱 중요시 된다.

자기소개서 첨삭 시스템 이용자의 경우 신뢰성 확보를 위하여 1:1 매칭을 통해 서로의 정보를 확인한 뒤 자기소개서 첨삭을 주고받게 한다. 한 번의 1:1 첨삭이 끝난 후 추가적인 첨삭을 원한다면, 다시 1:1 매칭을 통해 다른 동료와 동일한 방법으로 첨삭을 진행할 수 있다.



<그림 8> 자기소개서 첨삭 시스템 모형도

자기소개서 첨삭 시스템에서도 최종 목적은 본인이 작성한 글에 대한 첨삭을 받는 것이다. 이를 위해 1:1 매칭 서비스를 제공하고 각자가 작성한 자기소개서를 서로 첨삭하게 한다. 매칭이 된 후에 정해진 기간 내에 서로의 글을 첨삭하고 이를 교환해야 하는데, 기한을 넘길 경우 자동으로 그룹에서 퇴출당하게 된다. 또한, 서로가 첨삭 받은 내용을 평가하여 이후 1:1 매칭에서 신뢰도의 척도로 사용한다. 이에 더하여, 일부러 질이 낮은 첨삭을 하는 경우, 신고 제도를 도입하여 관리자로 하여금 해당 사용자를 제재할 수 있도록 한다. 이용자들 중 여러 번의 첨삭을 받고 싶은 이용자가 있다면 한 번의 자소서 교환과 첨삭 평가가 완전히 끝나야만 다시 매칭을 신청할 수 있도록 한다.

## 4.2. 과제 수행 일정

No.	수행내용	수행일정 (단위 : 주차)											
		3	4	5	6	7	8	9	10	11	12	13	14
1	계획수립, 자료조사												
2	서비스 기획												
3	제안서 작성, 계획 발표 준비												
4	데이터베이스 구축												
5	웹 front-end 개발												
6	웹 back-end 개발												
7	중간 발표 준비												
8	테스팅, 디버깅												
9	최종 발표 준비												

## 5. 과제 세부내용

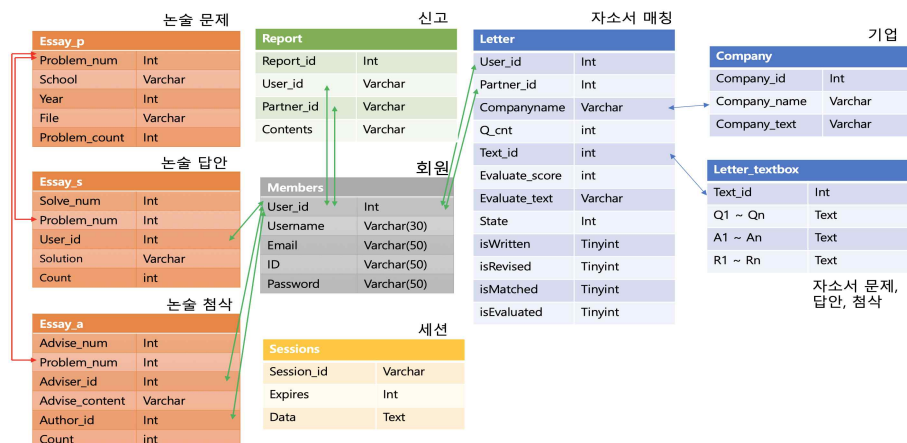
### 5.1. 데이터베이스

#### (1) 개발 환경

- ① 프리티어 : AWS(아마존 웹서비스)
- ② 운영 체제 : ubuntu-18.04.1 LTS
- ③ 사용 프로그램 : mysql Ver 14.14 Distrib 5.7.24

mysql이 사용하는 포트인 3306번 포트를 서버와 AWS 환경설정에서 열어주고 root 계정에 모든 권한을 부여했다. ip주소와 mysql root 계정을 통해 바로 디비에 쿼리문을 보낼 수 있다.

#### (2) 전체 스키마



<그림 9> 전체 스키마 모형도

### (3) 스키마 세부사항

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(30)	NO		NULL	
email	varchar(50)	NO		NULL	
id	varchar(30)	YES	UNI	NULL	
password	varchar(128)	NO		NULL	
passans	varchar(200)	YES		NULL	
passques	varchar(200)	YES		NULL	
point	int(11)	YES		0	

8 rows in set (0.00 sec)

<그림 10> Members 테이블

Members 테이블은 회원가입 시 유저가 입력하는 정보, 비밀번호 찾기에 필요한 정보, 현재 회원이 보유중인 포인트를 저장하고 있다.

Field	Type	Null	Key	Default	Extra
report_id	int(11)	NO	PRI	NULL	auto_increment
user_id	varchar(30)	YES		NULL	
partner_id	varchar(30)	YES		NULL	
contents	varchar(1000)	YES		NULL	

4 rows in set (0.00 sec)

<그림 11> Report 테이블

Report 테이블은 신고기능을 위한 테이블이다. 신고한 회원의 id, 신고당한 회원의 id, 신고 내용으로 구성되어 있다.

Field	Type	Null	Key	Default	Extra
session_id	varchar(128)	NO	PRI	NULL	
expires	int(11) unsigned	NO		NULL	
data	text	YES		NULL	

3 rows in set (0.00 sec)

<그림 12> Sessions 테이블

Sessions 테이블은 세션정보를 저장하기 위한 테이블이다.

Field	Type	Null	Key	Default	Extra
problem_num	int(11)	NO	PRI	NULL	
school	varchar(50)	YES		NULL	
year	int(11)	YES		NULL	
file	varchar(300)	YES		NULL	
problem_count	int(11)	YES		NULL	

5 rows in set (0.00 sec)

<그림 13> Essay\_p 테이블

Essay\_p 테이블은 논술 기출 문제의 정보를 저장하기 위한 테이블이다. 학교, 년도, 파일위치, count 횟수를 확인할 수 있다. file 필드에 있는 서버의 디렉토리에 접근하여 홈페이지에서 논술 문제를 바로 다운받을 수 있다.

Field	Type	Null	Key	Default	Extra
solve_num	int(11)	NO	PRI	NULL	auto_increment
problem_num	int(11)	YES		NULL	
user_id	int(11)	YES		NULL	
solution	varchar(1500)	YES		NULL	
count	int(11)	YES		NULL	

5 rows in set (0.00 sec)

<그림 14> Essay\_s 테이블

Essay\_s 테이블은 논술 답안 작성을 위한 테이블이다. 어떤 문제를 누가 풀었고 어떠한 답안을 작성하였는지에 대한 정보를 알 수 있다.

Field	Type	Null	Key	Default	Extra
advise_num	int(11)	NO	PRI	NULL	auto_increment
problem_num	int(11)	YES		NULL	
adviser_id	int(11)	YES		NULL	
author_id	int(11)	YES		NULL	
count	int(11)	YES		NULL	
advise_content	varchar(1000)	YES		NULL	

6 rows in set (0.00 sec)

<그림 15> Essay\_a 테이블

Essay\_a 테이블은 첨삭한 내용에 대한 정보를 저장하기 위한 테이블이다.



Field	Type	Null	Key	Default	Extra
company_id	int(11)	NO	PRI	NULL	auto_increment
company_name	varchar(200)	YES		NULL	
company_text	varchar(1000)	YES		NULL	

3 rows in set (0.01 sec)

<그림 16> Company 테이블

Company 테이블에는 자기소개서 첨삭 서비스를 제공할 기업 정보가 저장되어 있다.

Field	Type	Null	Key	Default	Extra
text_id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO		NULL	
partner_id	int(11)	YES		NULL	
companyname	varchar(30)	YES		NULL	
q_cnt	int(11)	YES		NULL	
evaluate_score	int(11)	YES		NULL	
evaluate_text	varchar(5000)	YES		NULL	
state	int(11)	YES		NULL	
isWritten	tinyint(1)	YES		0	
isMatched	tinyint(1)	YES		0	
isRevised	tinyint(1)	YES		0	
isEvaluated	tinyint(1)	YES		0	

12 rows in set (0.00 sec)

<그림 17> Letter 테이블

Letter 테이블은 자기소개서 첨삭 서비스의 1:1 매칭을 위한 테이블이다. 매칭된 두 회원의 id와 선택한 기업, 첨삭 평점과 내용, 프로세스 진행 상황을 저장하고 있다.

Field	Type	Null	Key	Default	Extra
text_id	int(11)	NO	PRI	NULL	
q1	text	YES		NULL	
q2	text	YES		NULL	
q3	text	YES		NULL	
q4	text	YES		NULL	
a1	text	YES		NULL	
a2	text	YES		NULL	
a3	text	YES		NULL	
a4	text	YES		NULL	
r1	text	YES		NULL	
r2	text	YES		NULL	
r3	text	YES		NULL	
r4	text	YES		NULL	

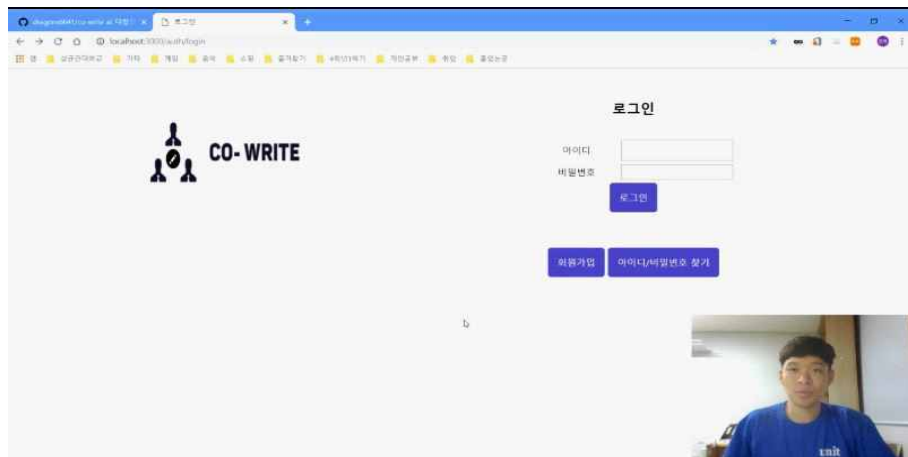
13 rows in set (0.00 sec)

<그림 18> Letter\_textbox 테이블

Letter\_textbox 테이블은 자기소개서의 문제, 답안, 첨삭내용을 저장하기 위한 테이블이다. text\_id를 통해 Letter 테이블과 연결할 수 있다.

## 5.2. 로그인

### (1) 로그인 화면



<그림 19> 로그인 화면

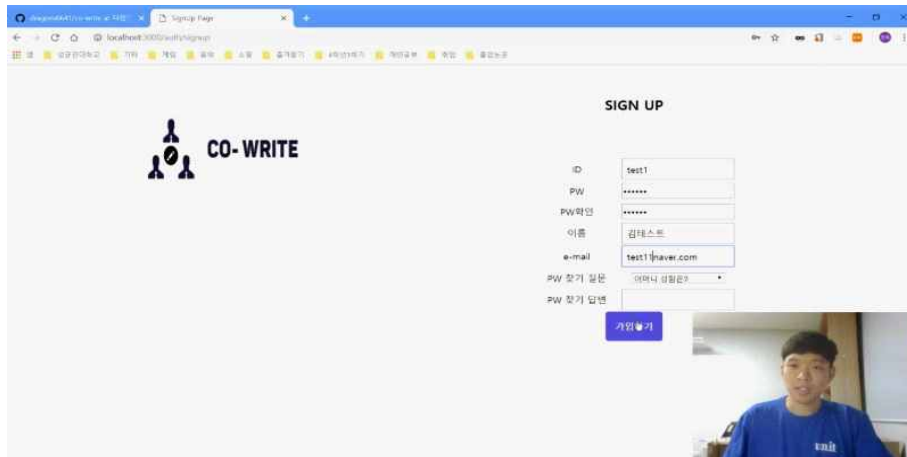
아이디와 비밀번호를 입력하고 로그인 버튼을 누르면 로그인이 가능하다. 아이디는 'id', 비밀번호는 'password'라는 변수로 login.js 파일에 전송된다.

```
router.get('/login', function(req, res){
  //res.sendFile(__dirname + '/views/login.html', {root: __dirname});
  res.render('auth/login', {});
});
router.post('/login', function(req, res){
  var id = req.body.id;
  var password = req.body.password;
  if(id == '' || password == ''){
    res.send("<script type='text/javascript'> alert('모든 항목을 채워주세요.');">history.back(0); </script>");
  }
  else{
    var sql = 'select * from members where id=? and password=?';
    var params = [id,password];
    db.query(sql, params, function (err, rows, fields) {
      if (err){
        console.log(err);
      }
      else{
        if(rows.length!=1){//로그인 실패
          res.send("<script type='text/javascript'> alert('ID 혹은 PW가 틀립니다.');">history.back(0); </script>");
        }
        else if(rows[0].id=="admin"){
          res.redirect('/admin/admin_page');
        }
        else{
          var user_id = rows[0].user_id;
          var username = rows[0].username;
          var point = rows[0].point;
          req.session.point = point;
          req.session.user_id = user_id;
          req.session.username = username;
          req.session.save(function(){
            res.redirect('/');
          });
        }
      }
    });
  }
});
```

<그림 20> login.js - 로그인

로그인 화면에서 전송받은 id, password를 데이터베이스 Members 테이블의 id, password와 비교한다. 일치하는 기존 사용자의 id와 password가 없다면, "ID 혹은 PW가 틀립니다."라는 메시지를 출력한다. id와 password가 존재한다면, 회원 정보를 세션에 저장한 후 논술/자기소개서 첨삭 페이지로 넘어가게 된다.

## (2) 회원가입 화면



<그림 21> 회원가입 화면

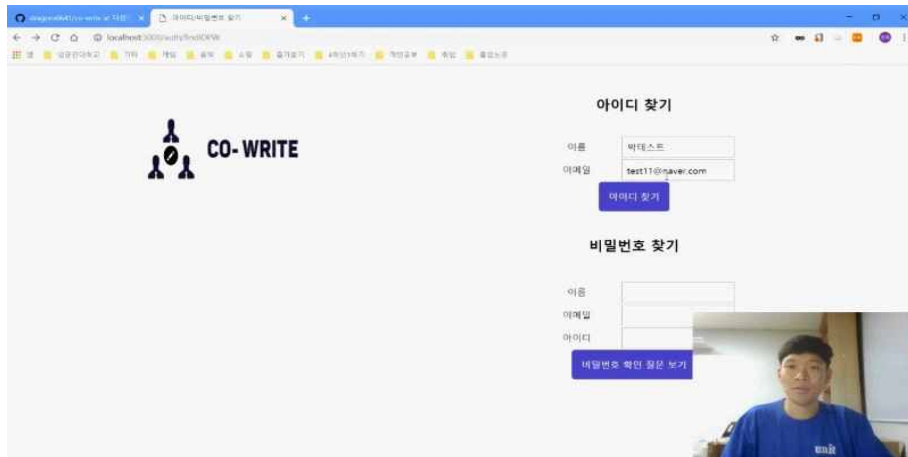
회원 정보를 입력하면 id, password, password 확인, username, email을 login.js 파일로 넘겨준 후 데이터베이스의 Members 테이블에 저장한다. 소스코드는 아래와 같다.

```
router.get('/signup', function(req, res){
  res.render('auth/signup',{});
});
router.post('/signup', function(req, res){
  var id = req.body.id;
  var password = req.body.password;
  var pwc = req.body.pwc;
  var username = req.body.username;
  var email = req.body.email;
  var passques = req.body.passques;
  var passans = req.body.passans;
  if(id==" || password==" || pwc==" || username==" || email==" || passques==" || passans==" ){
    res.send("<script type='text/javascript'> alert('모든 항목을 채워주세요.');">history.back(); </script>");
  }
  else if(password!=pwc){
    res.send("<script type='text/javascript'> alert('PW와 PW확인이 같지 않습니다.');">history.back(); </script>");
  }
  else{
    var sql = 'insert into members(id, password, username, email, passques, passans) values(?, ?, ?, ?, ?, ?)';
    var params=[id,password,username,email, passques, passans];
    db.query(sql,params, function (err, rows, fields) {
      if (err){
        if(err.errno==1062){
          res.send("<script type='text/javascript'> alert('이미 존재하는 ID 입니다.');"> history.back(); </script>");
        }
      }
      else{
        res.send("<script type='text/javascript'> alert('계정이 정상적으로 생성되었습니다.');"> location.href='/auth/login'; </script>");
      }
    });
  }
});
```

<그림 22> login.js - 회원가입

Members 테이블에 같은 아이디가 이미 존재하는 경우에는 “이미 존재하는 ID입니다.”라는 메시지를 출력한다. password와 password 확인이 같지 않을 경우 “PW와 PW확인란이 일치하지 않습니다.”라는 메시지를 출력한다. 계정이 성공적으로 생성되면 다시 로그인 창으로 이동한다.

### (3) 아이디/비밀번호 찾기 화면



<그림 23> 아이디/비밀번호 찾기 화면

이름과 이메일을 입력하면 login.js 파일에서 확인하게 된다. 소스코드는 아래와 같다.

```
//4. ID 찾기
router.get('/findIDPW', function(req, res){
  //res.render('auth/findIDPW', {root: __dirname});
  res.render('auth/findIDPW', {id:'', passques:'', passans:''});
});
router.post('/findID', function(req, res){
  var username = req.body.username;
  var email = req.body.email;
  if(username=='' || email==''){
    res.send("<script type='text/javascript'> alert('모든 항목을 채워주세요.');"history.back(); </script>");
  }
  else{
    var sql = 'select id from members where username=? and email=?';
    var params=[username, email];
    db.query(sql, params, function(err, rows, fields){
      if(rows.length!=1){//크기 1의 결과
        res.send("<script type='text/javascript'> alert('회원이 존재하지 않습니다.');"history.back(); </script>");
      }
      else{
        var id = rows[0].id;
        //res.render('findIDPW', {id:id, passques:'', passans:''});
        res.render('auth/findIDPW', {id:id, passques:'', passans:''});
      }
    });
  }
});
});
```

<그림 24> login.js - 아이디 찾기

항목을 비운 상태로 아이디 찾기 버튼을 누르면 “모든 항목을 채워주세요.” 라는 메시지를 출력한다. 정상적인 username, email 값을 입력받으면 데이터베이스의 Members 테이블에서 해당 유저의 아이디를 찾아서 아래와 같이 출력하게 된다.



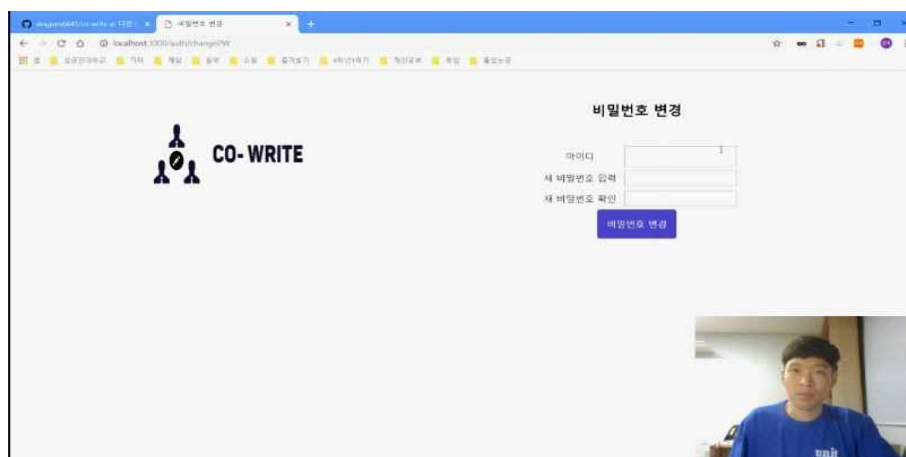
<그림 25> 아이디 찾기 성공

비밀번호 찾기도 아이디 찾기와 같은 방식으로 진행할 수 있다.

```
//5. PW 찾기
router.post('/findPW', function(req, res){
  var username = req.body.username;
  var email = req.body.email;
  var id = req.body.id;
  if(username=='' || email=='' || id==''){
    res.send('<script type="text/javascript"> alert("모든 항목을 채워주세요.");history.back(); </script>');
  }
  else{
    var sql = 'select passques, passans from members where username=? and email=? and id=?';
    var params=[username, email, id];
    db.query(sql, params, function(err, rows, fields){
      if(rows.length!=1){
        res.send('<script type="text/javascript"> alert("회원이 존재하지 않습니다.");history.back(); </script>');
      }
      else{
        var passques = rows[0].passques;
        var passans = rows[0].passans;
        res.render('auth/findIDPW', {id:'', passques:passques, passans:passans});
      }
    });
  }
});
```

<그림 26> login.js - 비밀번호 찾기

일치하는 회원이 없으면 “회원이 존재하지 않습니다.” 라는 메시지를 출력한다. 정상적인 username, email, id 값을 입력받으면 비밀번호 확인 질문을 볼 수 있다. 회원 가입 시 설정한 비밀번호 확인 답변을 입력하면 비밀번호를 바꿀 수 있다.



<그림 27> 비밀번호 변경 화면

아이디와 새 비밀번호를 입력한 후 비밀번호 변경 버튼을 누르면 로그인 창에서 새로운 비밀번호로 로그인할 수 있다.

```
router.get('/changePW', function(req, res){
  res.render('auth/changePW',{});
});
router.post('/changePW', function(req, res){
  var id = req.body.id;
  var newpassword = req.body.newpassword;
  var newpasswordc = req.body.newpasswordc;
  if(id!='' || newpassword!='' || newpasswordc!=''){
    res.send('<script type="text/javascript"> alert("모든 항목을 채워주세요.");history.back(); </script>');
  }
  else if(newpassword!=newpasswordc){
    res.send('<script type="text/javascript"> alert("PW와 PW확인이 같지 않습니다.");history.back(); </script>');
  }
  else{
    var sql = 'update members set password=? where id=?';
    var params = [newpassword, id];
    db.query(sql, params, function(err, rows, fields){
      res.send('<script type="text/javascript"> alert("PW가 성공적으로 변경되었습니다."); location.href="/auth/login"; </script>');
    });
  }
});
```

<그림 28> login.js - 비밀번호 변경

새 비밀번호와 새 비밀번호 확인란이 같지 않으면 "PW와 PW확인이 같지 않습니다." 라는 메시지를 출력한다. 정상적으로 입력할 경우 데이터베이스의 Members 테이블에서 해당 사용자의 비밀번호를 업데이트한다.

### 5.3. 논술 첨삭

#### (1) 메인 화면



<그림 29> 논술 메인 화면

논술 메인 화면 하단에는 논술 첨삭 서비스에 대한 간략한 설명이 있다. 상단 바를 통해 메인 페이지, 풀기 페이지, 첨삭 페이지, 마이 페이지, 자기소개서 첨삭 페이지로 이동할 수 있다. 상단 바 우측에는 사용자의 정보를 표시하고 로그아웃 버튼을 추가했다.

```

<li>
  <a href="My_page" data-theme="_bgp">나의 페이지</a>
</li>
<li><pre>      </pre></li>
<li>
  <a><%= username %>님 환영합니다.</a>
</li>

```

<그림 30> main.ejs

```

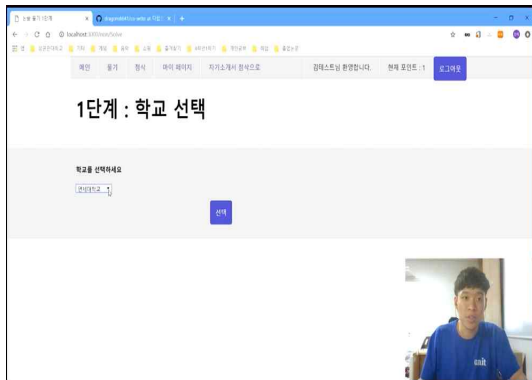
router.get('/Main',function(req,res){
  if(req.session.user_id){
    db.query('select point from members where user_id = ${req.session.user_id}',function(err,result3){
      point = result3[0].point;
      user_id = req.session.user_id;
      username = req.session.username;
      res.render('nonsul/Main', {username:username, point:point});
    });
  }
  else{
    res.redirect('/auth/login');
  }
});

```

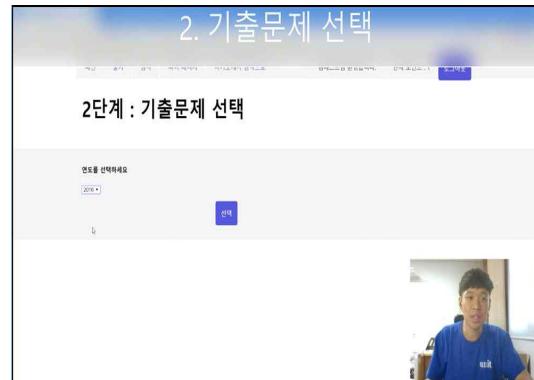
<그림 31> non.js - main router

사용자 이름과 현재 점수 포인트는 non.js 파일에서 라우팅 시에 render 함수에 key, value 형식의 array로 전달한다. main.ejs는 논술의 기본 페이지로 세션 정보를 통해, 로그인 여부를 확인한다. 세션에 정보가 있을 때는 데이터베이스에서 유저의 포인트와 세션에 저장된 유저 정보를 변수에 저장한다. 세션 정보가 없을 때는 로그인 페이지로 돌아간다.

## (2) 문제 풀기



<그림 32> 문제 풀기 - 학교 선택



<그림 33> 문제 풀기 - 기출문제 선택

문제 풀기 화면에서는 학교와 기출문제 연도를 선택한 후 다음 페이지에 전송한다. ejs파일을 render하여 화면을 구성하고, 학교와 연도는 post방식으로 전송된다.

```

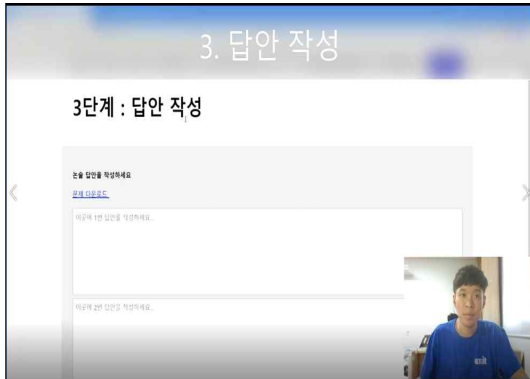
router.post('/Solve2_process', function(req,res){
  year = req.body.year;
  db.query('select problem_num, problem_count from essay_p where year = ${year} and school = '${name}',function(err,result){
    if (err){
      console.log(err)
    }
    else{
      if (result.length == 0){
        res.send('<script type="text/javascript"> alert("존재하지 않는 파일 입니다."); history.go(-2); </script>');
      }
      else{
        temp_num = result[0].problem_num;
        prob_count = result[0].problem_count;
        res.redirect('Solve3');
      }
    }
  });
});

```

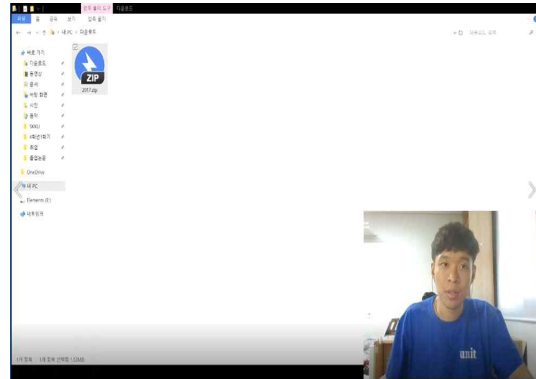
<그림 34> non.js - solve2 process router

non.js 파일에서는 전송받은 학교와 연도를 이용하여 데이터베이스에 해당 문제 파일이 있는지 확인한다. 만약 없는 경우 "존재하지 않는 파일입니다."라는 메시지를 출력하고 학교

선택 페이지로 돌아간다. 기출문제 파일이 데이터베이스에 저장되어 있는 경우 각 문제 번호와 문제 수를 변수에 저장 후 3단계로 넘어간다.

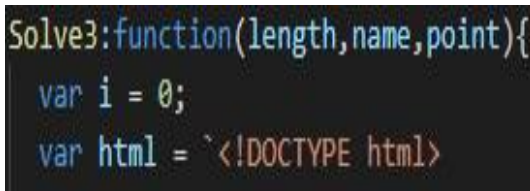


<그림 35> 문제 풀기 - 답안 작성

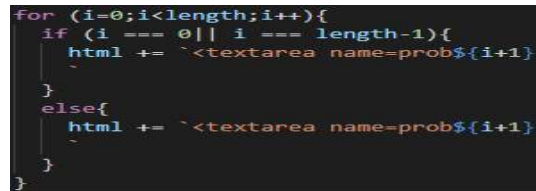


<그림 36> 문제 풀기 - 문제 다운로드

3단계에서는 문제 수만큼의 답안 입력창을 제공한다. “문제 다운로드” 링크를 클릭하면 문제를 다운받을 수 있다. 링크를 클릭하면 express의 download 기능을 이용하여 서버에 미리 저장한 논술 파일을 사용자에게 전송한다.



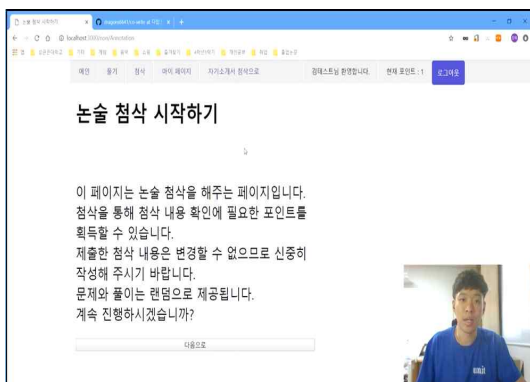
<그림 37> template.js - solve 함수



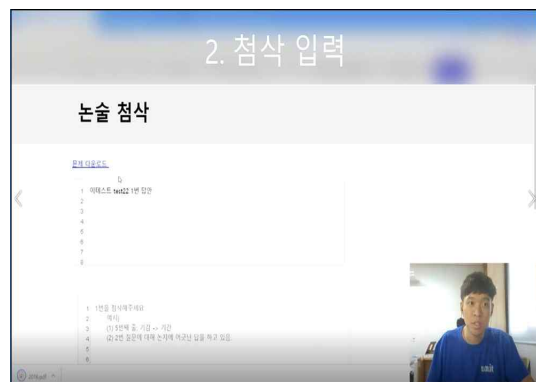
<그림 38> for문을 이용한 textarea 추가

함수의 parameter는 총 3개이다. length는 문제의 수이고, name과 point는 상단 바의 사용자 정보에 출력된다. 그리고 for문을 이용해서 length 값만큼의 textarea를 만든다.

#### (4) 첨삭하기



<그림 39> 논술 첨삭 - 시작하기



<그림 40> 논술 첨삭 - 첨삭 입력



논술 검색 시작 화면에 간략한 안내문을 포함한 annotation.ejs 파일을 render하고 annotation2 페이지로 이동한다.

```
router.post('/Annotation2',function(req,res){
  db.query('select problem_num, user_id from essay_s where user_id != ${req.session.user_id} order by rand() limit 1;', function(err,result1){
    temp_id = result1[0].user_id;
    temp_num = result1[0].problem_num;
    db.query('select solve_num, solution, count from essay_s where problem_num = ${result1[0].problem_num} and user_id = ${result1[0].user_id}', function(err,result2){
      if(err){
        console.log(err)
      }
      else{
        var html = template.Annotation2(result2.length,result2, req.session.username,point);
        res.send(html);
      }
    })
  })
});
```

<그림 41> non.js - annotation2 router

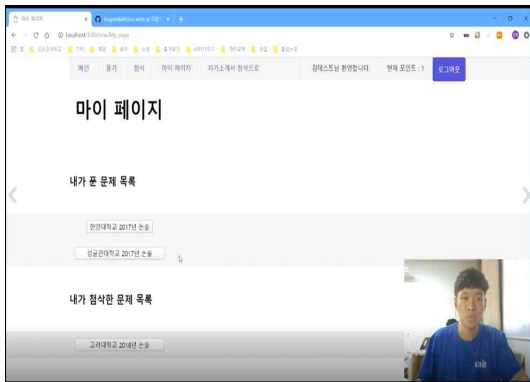
annotation2에서는 다른 사람이 제출한 논술 문제 답안을 랜덤으로 데이터베이스에서 선택한다. 해당 답안의 문제 수와 풀이 내용을 변수에 저장하여 template.js의 annotation2함수에 parameter로 전송한다. 각 parameter는 문제 수, 문제 풀이 내용을 저장한 array, 그리고 상단 바에 저장할 사용자 정보를 의미한다.

```
router.post('/Annotation3',function(req,res){
  var leng = Object.keys(req.body).length;
  var i = 0;
  contents = Object.values(req.body);
  for(i=0;i < leng; i++){
    var input = {
      'problem_num' : temp_num, 'adviser_id' : req.session.user_id, 'author_id' : temp_id, 'advise_content' : contents[i], 'count' : i+1;
    };
    db.query('insert into essay_a set ?;', input, function(err,result){
      if(err){
        console.log(err)
      }
      else{
        console.log('success');
      }
    });
  }
  db.query('select point from members where user_id = ${req.session.user_id}',function(err,result1){
    db.query('update members set point = ${result1[0].point}+1 where user_id = ${req.session.user_id}', input, function(err,result2){
    });
  });
  point += 1;
  res.render('nonsul/Annotation3', {username:username, point:point})
});
```

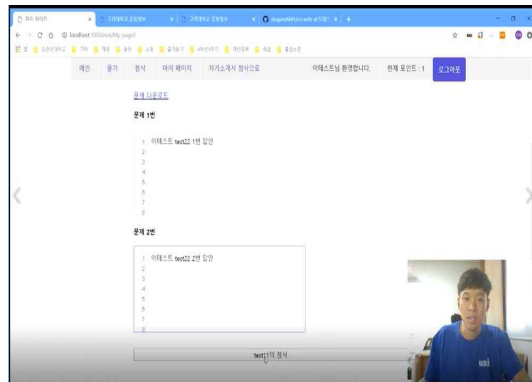
<그림 42> non.js - annotation3 router

annotation3에서는 전송 받은 문제 수와 검색 내용을 데이터베이스에 저장한다. 그리고 검색을 해준 사용자의 검색 포인트를 1만큼 증가시키고 데이터베이스에 업데이트한다. 업데이트한 검색 포인트를 불러오기 위해 데이터베이스에 새로운 쿼리문을 전송하는 것은 효율적이지 못하다. 따라서 쿼리문을 따로 전송하지 않고 변수에 저장한 포인트를 1만큼 증가시키는 것으로 대체했다. 이후 annotation3 페이지를 render 해준다.

## (5) 마이 페이지



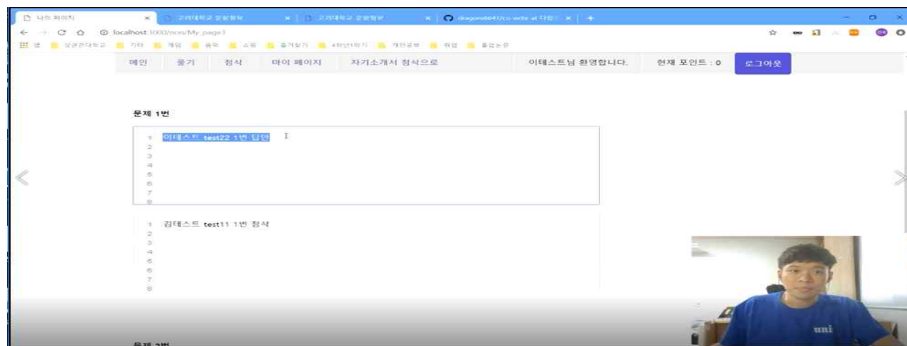
<그림 43> 논술 마이 페이지 - 문제 목록



<그림 44> 논술 마이 페이지 - 문제 확인

my\_page에서는 내가 푼 문제들의 목록과 내가 찜한 문제의 목록을 볼 수 있다. 각 문제를 클릭하면 my\_page2 화면으로 이동할 수 있다.

my\_page2에서는 문제를 푼 사람의 id와 문제 다운로드 링크, 풀이 내용, 찜삭자의 id를 확인할 수 있다. 아래의 찜삭 확인 버튼을 누르면 포인트를 1만큼 차감한 후 my\_page3 화면으로 이동한 후 다른 사용자가 자신의 글에 찜삭한 내용을 확인할 수 있다.



<그림 45> 논술 마이 페이지 - 찜삭 확인

my\_page3에서는 사용자가 확인하려고 하는 찜삭 내용이 자신의 것인지를 먼저 확인한다. 자신이 작성한 찜삭 내용은 포인트 차감 없이 자유롭게 확인할 수 있다. 다른 사용자의 찜삭을 보려고 하는 경우, 우선 사용자의 현재 찜삭 포인트를 확인한다. 사용 가능한 포인트가 1 이상일 경우에는 찜삭 내용을 공개하고 포인트를 1만큼 차감한다. 이후 각 문제 번호별로 자신의 답안과 다른 사람의 찜삭 내용을 열람할 수 있다.

## 5.4. 자기소개서 탐색

### (1) 마이 페이지



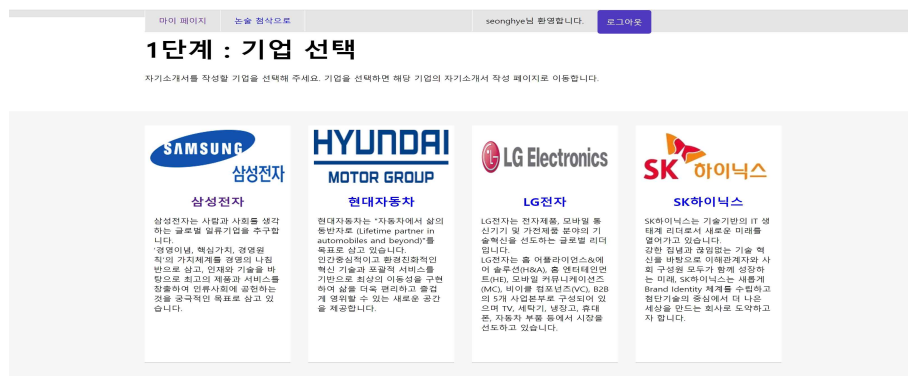
<그림 46> 자기소개서 마이 페이지

마이 페이지 상단에는 navigator 바가 있으며 마이 페이지, 논술 탐색 플랫폼으로 각각 이동할 수 있도록 했다. 또한 오른쪽 위에 사용자 정보와 로그아웃 버튼을 배치했다. 자신이 작성한 자기소개서 목록을 진행 중인 프로세스와 함께 볼 수 있다. 각 프로세스에 대한 설명은 아래와 같다.

- ① 매칭 대기 중 : 1:1 매칭을 기다리고 있는 상태
- ② 탐색하기 : 탐색을 진행할 수 있는 단계
- ③ 탐색 대기 중 : 상대방의 탐색을 기다리고 있는 상태
- ④ 탐색 보기 및 평가 : 상대의 탐색을 확인하고 탐색 내용 평가를 진행할 수 있는 단계
- ⑤ 프로세스 완료 : 모든 프로세스가 종료된 상태

주황색으로 표시된 진행 중인 프로세스를 클릭하면 프로세스 진행 화면으로 이동할 수 있다. 하단의 “새로운 자기소개서를 작성” 버튼을 누르면 기업 선택 페이지로 이동한다.

### (2) 기업 선택



<그림 47> 기업 선택 화면

기업 선택 화면에서는 기업의 로고, 이름, 슬로건을 확인할 수 있다. 자기소개서를 작성할 기업을 선택하면 자기소개서 작성 페이지로 이동한다.

### (3) 자기소개서 작성

[illegible]

<그림 48> 자기소개서 작성 화면

자기소개서 작성 화면에서는 문제, 답안, 권장 글자 수를 입력할 수 있다. 공통 문항의 문제는 미리 입력되어 있으며 사용자가 자유롭게 수정할 수 있다. 최소/최대 글자 수는 권장 글자 수의 80%, 120%로 미리 설정되어 있다. 글자 수 범위를 만족하지 않는 답안은 제출할 수 없다. 하단의 “제출하기” 버튼을 눌러서 작성한 자기소개서를 제출하면 1:1 매칭을 기다릴 수 있다. 매칭이 완료된 경우 “첨삭하기” 프로세스를 클릭해서 자기소개서 첨삭 화면으로 이동할 수 있다.

(4) 자기소개서 첨삭

[illegible]

<그림 49> 자기소개서 첨삭 화면

자기소개서 첨삭 화면에서는 1:1 매칭 후 상대방의 자기소개서를 보고 첨삭을 진행할 수 있다. 페이지의 좌측에는 상대방의 자기소개서 답안이, 우측에는 첨삭을 적을 수 있는 입력창이 있다. 좌측의 자기소개서에는 줄 번호를 매겨서 첨삭의 편의성을 높였다. 문항별 첨삭은 100자 이상 작성해야 하며 조건을 만족하지 않을 경우 제출이 불가능하다. 하단의 “제출하기” 버튼을 눌러서 작성한 첨삭 내용을 제출할 수 있다. 상대방이 첨삭을 완료하면 “첨삭 보기 및 평가” 프로세스를 클릭해서 첨삭 확인과 평가를 진행할 수 있다.

### (5) 첨삭 확인과 평가

<그림 50> 첨삭 확인과 평가 화면

첨삭 확인과 평가 화면에서는 상대방의 첨삭을 확인할 수 있다. 페이지의 좌측에는 자신의 자기소개서 답안이, 우측에는 상대방의 첨삭 내용이 있다. 페이지의 하단에는 첨삭 평점과 평가 내용 입력창이 있다. 첨삭 평점은 1점에서 5점까지 선택 가능하다. 첨삭 평가는 10자 이상 작성해야 제출이 가능하다. 첨삭 내용이 만족스럽지 않은 경우 “신고하기” 버튼을 누르면 신고 페이지로 이동한다. 평가 또는 신고를 완료한 경우 모든 프로세스가 종료된다.

### (6) 신고하기

<그림 51> 신고 페이지

신고 페이지에서는 상세 내용을 10자 이상 작성한 후 신고를 제출할 수 있다. 하단의 “제출하기” 버튼을 누르면 신고가 접수된다. 관리자 페이지에서 신고 목록을 확인할 수 있다.

#### (7) 1:1 매칭

```

69      //matching
70      sql = "select * from letter where isMatched=null";
71      conn.query(sql, function(err, rows, fields){
72          if(rows.length!=0){
73              sql = "select user_id from letter where text_id=?";
74              params = [rows[0].text_id];
75              conn.query(sql, params, function(err, rows, fields){
76                  sql = "update letter set state=?, partner_id=?, isMatched=? where text_id=?";
77                  params = [2, rows[0].user_id, 1, text_id];
78
79                  conn.query(sql, params, function(err, rows, fields){
80                      sql = "update letter set state=?, partner_id=?, isMatched=? where text_id=?";
81                      params = [2, user_id, 1, text_id-1];
82                      conn.query(sql, params, function(err, rows, fields){
83                          res.redirect('step3?companyname='+companyname);
84                      });
85                  });
86              });
87          }
88      } else{
89          res.redirect('match_waiting');
90      }

```

<그림 52> 1:1 매칭 - 소스코드

데이터베이스의 Letter 테이블에서 isMatched 변수가 null인 행이 있는지를 찾아 확인한다. 기다리고 있는 상대가 있다면 매칭을 진행한다. Letter 테이블 각 행의 partner\_id, state, is\_Matched 변수를 업데이트 해주면 매칭이 완료된다.

#### (8) 침삭 제출 확인

```

119      router.post('/step3', function(req, res){
120          var text_id = req.body.tid;
121          var r1 = req.body.r1;
122          var r2 = req.body.r2;
123          var r3 = req.body.r3;
124          var r4 = req.body.r4;
125          var user_id = req.session.user_id;
126          var sql = "update letter_textbox set r1=?, r2=?, r3=?, r4=? where text_id=?";
127          var params = [r1,r2,r3,r4,text_id];
128          conn.query(sql,params, function(err, rows, fields){
129              sql = "update letter set isRevised=? where text_id=?";
130              params = [1, text_id];
131              conn.query(sql, params, function(err, rows, fields){
132                  sql = "update letter set state=? where user_id=?";
133                  params=[3,user_id];
134                  conn.query(sql, params, function(err, rows, fields){
135                      //all revised -> state=4
136                      sql = "select companyname, text_id, isRevised from letter where user_id=?";
137                      params = [user_id];
138                      conn.query(sql, params, function(err, rows, fields){
139                          var companyname = rows[0].companyname;
140                          if(rows[0].isRevised==1){
141                              sql = "update letter set state=? where text_id=?";
142                              params=[4, rows[0].text_id];
143                              conn.query(sql, params, function(err, rows, fields){
144                                  sql = "update letter set state=? where text_id=?";
145                                  params=[4, text_id];
146                                  conn.query(sql, params, function(err, rows, fields){
147                                      res.redirect('step4?companyname='+companyname);
148                                  });
149                              });
150                          }
151                          else{
152                              res.render('si/si_revise_waiting',{});
153                          }

```

<그림 53> 침삭 제출 확인 - 소스코드

상대방의 침삭 제출 여부를 데이터베이스의 Letter 테이블에서 가져온다. 상대가 이미 침삭을 제출했다면 Letter 테이블의 state 값을 업데이트한다.

#### (9) 글자 수 세기

```

47 function CheckStrLength(index)
48 {
49     var text, text_len = 0, rec_len = 0, min_target, max_target, min_len, max_len, cur_target, message;
50     if (index == 1)//1번 문항
51     {
52         text = document.getElementById("a1");
53         rec_len = document.getElementById("rec_q1").value;
54         min_target = document.getElementById("min_a1");
55         max_target = document.getElementById("max_a1");
56         cur_target = document.getElementById("cur_a1");
57         message = document.getElementById("message_a1");
58     }
59     else if (index == 2)//2번 문항
60     {
61     }
62     else if (index == 3)//3번 문항
63     {
64     }
65     else if (index == 4)//4번 문항
66     {
67     }
68     if ((rec_len < 100) && (rec_len > 4000))
69     {
70         alert("권장 글자 수는 100자 이상 4000자 이하여야 합니다.");
71         return;
72     }
73     min_len = parseInt(rec_len * 0.8);
74     max_len = parseInt(rec_len * 1.2);
75     min_target.value = min_len;
76     max_target.value = max_len;
77     text_len = text.value.length;
78     cur_target.value = text_len;
79     if (text_len < min_len)//글자 수 미달
80     {
81         message.innerHTML = "<font color = red>글자 수 미달</font>";
82         ans_status[index - 1] = text_len - min_len;
83     }
84     else if (text_len > max_len)//글자 수 초과
85     {
86     }
87     else//글자 수 적절
88     {
89     }
90     CheckError();
91     return;
92 }

```

<그림 54> 글자 수 세기 - 소스코드

자기소개서 작성, 침삭, 평가 및 신고에서 글자 수 세기 기능을 사용했다. textarea의 글자 수를 확인 하여 최소 글자 수보다 적을 경우 글자 수 미달, 최대 글자 수보다 많을 경우 글자 수 초과라는 메시지를 확인할 수 있다.

## 5.5. 관리자 페이지

## 신고 내역 조회

현재까지 접수된 신고 내역을 확인할 수 있습니다.

	신고한 사용자	신고당한 사용자	신고 내용
<input type="checkbox"/>			

[회원 삭제 하기](#)

<그림 55> 관리자 페이지

논술/자기소개서 첨삭 서비스에서 자신이 작성한 답안에 대해 불성실한 첨삭을 받았을 때 신고를 제출할 수 있다. 관리자 페이지에서는 신고 내역을 확인할 수 있다.

```
<!--  
START MODULE AREA 3: Text | Text  
-->  
<section>  
  <div data-layout="r">  
    <div data-layout="all">  
      <form action="/admin/admin_page" method="post">  
        <table border="1px solid black" style="border-collapse: collapse;" cellspacing="10px">  
          <tr>  
            <th>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</th>  
            <th>신고한 사용자</th>  
            <th>신고당한 사용자</th>  
            <th>신고 내용</th>  
          </tr>  
          <script>  
            if ('<%= row_length%>'== '0'){  
              document.write("<p>신고 내역이 없습니다.</p>");  
            }  
            else{  
              var arr_report_id = '<%= arr_report_id%>'.split(",");  
              var arr_user_id = '<%= arr_user_id%>'.split(",");  
              var arr_partner_id = '<%=arr_partner_id%>'.split(",");  
              var arr_contents = '<%=arr_contents%>'.split(",");  
              var i;  
              for(i=0;i<arr_user_id.length;i++){  
                document.write("<tr><th><input type='checkbox' name='report[]' value="+arr_report_id[i]+"></th>");  
                document.write("<th>"+arr_user_id[i]+"</th>");  
                document.write("<th>"+arr_partner_id[i]+"</th>");  
                document.write("<th>"+arr_contents[i]+"</th>");  
              }  
            }  
          </script>  
        </table>  
        <br>  
        <button type="submit" class="btn" >회원 삭제 하기</button>  
      </form>  
    </div>  
  </div>  
</section>  
<!--  
END MODULE AREA 3: Text | Text  
-->
```

<그림 56> admin.js

신고한 사용자, 신고 당한 사용자, 신고 내용을 데이터베이스의 Report 테이블에서 불러온 후 출력하는 방식임을 알 수 있다.



## 6. 기대 효과

학생들은 글 쓸 일이 많다. 자기소개서를 써야 할 때도 있고, 과제물 보고서를 작성해야 할 때도 많다. 글을 완성하고도 더 좋은 글을 작성하기 위해 몇 번이고 검토하곤 한다. 하지만 아무리 되고를 거듭해도 자신의 글이 완벽하게 보이지 않는다.

이렇게 글을 쓰면서 불안한 마음을 가질 수밖에 없는 이유는, 다른 사람이 자신의 글을 어떻게 평가할지 글쓴이는 전혀 알 수 없기 때문이다. 최종적으로 글을 평가하는 인사담당자나 과제 채점 조교가 글을 읽는 순간에는 이미 평가가 끝난 상태이다.

만약 글을 쓴 사람이 최종적으로 글을 제출하기 전에 다른 사람의 피드백을 받을 수 있다면 많은 도움이 될 것이다. 하지만 다른 사람에게 자세한 피드백을 받기란 쉽지 않다. 다른 사람은 그 글을 써보지 않았기 때문에 어떠한 목적으로 글을 써야 하는지 이해하기 쉽지 않기 때문이다. 이런 경우 “잘 쓰셨네요.” 라는 형식적인 답변이 돌아오는 경우가 대부분이다. 구체적으로 어떤 부분을 어떻게 고치라는 자세한 답변을 해주는 경우는 거의 없다.

우리가 제안한 플랫폼은 이러한 문제를 상당 부분 해결할 수 있다. 같은 글을 쓴 사람들끼리 첨삭을 해주는 방식이기 때문에 최소한 자신이 첨삭해 주어야 하는 글에 대해서는 아주 잘 이해하고 있다. 따라서 플랫폼 이용자는 자신이 다른 사람의 글에 정성껏 첨삭을 해준 만큼 자신도 양질의 첨삭을 받을 수 있다. 앞서 소개한 대로 다른 사람의 글을 첨삭해준 만큼 자신의 글에 달린 첨삭 댓글 내용을 확인할 수 있기 때문이다.

이 플랫폼에서 기대할 수 있는 긍정적 효과는 세 가지가 있다.

(1) 많은 사람들의 지성을 활용하여 정보수집과 문제 해결의 속도가 빨라진다. 다수의 첨삭을 통해 소수의 전문가가 첨삭해주는 것과 비슷하거나 더 좋은 효과를 기대할 수 있다. 또한 오타, 맞춤법 등의 사소한 실수를 효과적으로 수정할 수 있다. 뿐만 아니라 전체적인 글의 방향성과 내용 또한 첨삭을 통해 더욱 나아질 여지가 충분하다.

(2) 자신의 글을 첨삭 받으려면 자신도 다른 사람들의 글을 첨삭해주어야 한다. 이 과정에서 다른 사람의 글을 읽고 배울만한 점은 자신의 글쓰기에 활용할 수 있다. 이러한 선순환을 통해 플랫폼 이용자들의 글쓰기 실력을 전체적으로 향상시킬 수 있다.

(3) 플랫폼 그룹 구성원들이 서로 첨삭을 해주면서 공동체의 연대감을 느낄 수 있다. 이 경우 글쓰기 첨삭 이외의 추가적인 효과를 기대할 수 있다. 시험을 준비하면서 궁금한 사항이 있으면 자유롭게 질의응답을 하고 시험 정보를 그룹 멤버에게 공유해 주는 것을 예시로 들 수 있다. 그룹 구성원들이 서로를 신뢰할 수 있고 상호보완의 관계를 유지할 수 있다면 장기적인 관점에서 볼 때 공동체 문화를 형성할 수 있는 좋은 씨앗이 될 수 있다.

## 7. 기타

### 7.1. 한계점

#### (1) 수리 논술, 과학 논술 검색 서비스 제외

최초 계획은 언어 논술, 수리 논술, 과학 논술을 모두 아우르는 논술 검색 서비스를 제공하는 것이었다. 하지만 수리 논술과 과학 논술은 풀이 방법이 어느 정도 정해져 있고 수식을 컴퓨터로 입력하는 데 어려움이 많아서 일단은 제외하기로 했다.

#### (2) 대입 자기소개서 검색 서비스 제외

제안서 작성 단계에서는 취업뿐만 아니라 대입 자기소개서도 검색할 수 있도록 할 계획이었다. 그러나 개발 과정에서 시간이 부족했기 때문에 우선은 취업 자기소개서에 집중하고 이후 대입 자기소개서 검색까지 서비스를 확대하기로 했다.

### 7.2. 개선사항

#### (1) 논술 검색 시 다른 문제를 푼 사용자의 답안 검색 가능

계획발표 시점까지는 같은 논술 문제를 푼 사용자의 답안만을 검색할 수 있도록 했다. 하지만 논술 전형을 준비하는 학생들은 하나의 대학만을 바라보고 준비하기 보다는 비슷한 수준의 여러 대학을 목표로 삼는다. 이러한 점을 반영하여 다른 문제를 푼 사용자의 답안도 랜덤 배정의 대상에 포함되도록 개선했다.

#### (2) 자기소개서 검색 시 다른 회사 지원자와도 1:1 매칭 가능

중간발표 이전까지는 같은 회사 지원자끼리만 1:1 매칭이 되도록 했다. 그러나 과도한 제한으로 매칭 범위가 줄어들어 검색이 신속하게 진행되지 못하는 문제가 발생했다. 자기소개서 검색은 논술 검색에 비해 검색 진행 속도의 중요성이 매우 크다. 따라서 회사와 상관없이 1:1 매칭을 하는 방식으로 바꾸어 짧은 시간 내에 최대한 많은 검색을 받을 수 있도록 수정했다.

#### (3) 논술 기출문제의 제공 범위 확대

기존의 목표는 주요 10개 대학의 최근 3년 기출문제를 제공하는 것이었다. 이를 최근 10년으로 넓혀 사용자에게 최대한 많은 기출문제를 제공하고 있다.

#### (4) 자기소개서 작성 시 글자 수 체크 가능

초기에는 글자 수 제한의 절반 이상을 넘기면 자기소개서 제출이 가능했다. 현재는 글자 수를 확인할 수 있도록 했고, 권장 글자 수의 80% 이상 120% 이하를 작성해야 제출이 가능하다.

### 7.3. 업무 분담

#### (1) 결과 보고서 작성 담당업무

번호	성명	학과	학번	담당업무
1	김진옥	소프트웨어학과	2013310448	과제 요약, 과제 개요, 기타, 참고문헌 작성 편집 및 결과 보고서 제출
2	김태홍	전자전기공학부	2012310132	과제 세부내용 작성 (논술 첨삭)
3	이윤열	컴퓨터공학과	2012311683	과제 세부내용 작성 (로그인, 관리자 페이지)
4	최동규	소프트웨어학과	2014312893	과제 세부내용 작성 (데이터베이스)
5	최민진	컴퓨터공학과	2015311777	과제 추진 배경, 필요성, 목적 과제 해결 방안, 수행 일정 기대 효과 작성
6	한성혜	소프트웨어학과	2015313050	과제 세부내용 작성 (자기소개서 첨삭)

#### (2) 개발 과정 담당업무

번호	성명	학과	학번	담당업무
1	김진옥	소프트웨어학과	2013310448	계획 발표 준비, 시연 영상 제작, 자기소개서 front-end 개발 (HTML)
2	김태홍	전자전기공학부	2012310132	계획 발표 준비, 논술 back-end 개발 (nodejs)
3	이윤열	컴퓨터공학과	2012311683	최종 발표 준비, 로그인, 관리자 페이지 개발 (nodejs)
4	최동규	소프트웨어학과	2014312893	중간 발표 준비, 데이터베이스 구축 (MySQL)
5	최민진	컴퓨터공학과	2015311777	최종 발표 준비, 논술 front-end 개발 (HTML)
6	한성혜	소프트웨어학과	2015313050	중간 발표 준비, 자기소개서 back-end 개발 (nodejs)

## 8. 참고문헌

- [1] 데일리잡, 신입공채 취준생, 공채 준비? '자소서 작성부터', 이상준 기자, 2018년 3월  
<http://m.newsjob.co.kr/news/articleView.html?idxno=37887>
- [2] 중도일보, 취업 준비중 가장 막막한 순간은?... "이력서·자소서 쓸 때", 최고은 기자, 2018년 4월  
<http://m.joongdo.co.kr/view.php?key=20180404010002208#cb>
- [3] 동아닷컴 비즈N, 인사담당자 10명중 4명 "자소서 맞춤법 실수 반복되면 탈락", 최용석 기자, 2017년 10월  
<http://bizn.donga.com/HotClick/3/0115/20171012/86710394/1>
- [4] 한국소비자원 보도자료, 취업컨설팅 서비스, 계약 관련 소비자불만 많아 개선 필요, 2018년 1월  
[http://www.kca.go.kr/brd/m\\_32/view.do?seq=2376&multi\\_itm\\_seq=0](http://www.kca.go.kr/brd/m_32/view.do?seq=2376&multi_itm_seq=0)
- [5] 잡앤조이, '자소서 좀 봐 주세요~'...늘어나는 유료 첨삭 서비스 신뢰성은?, 이도희 기자, 2017년 2월  
[http://jobnjoy.com/portal/jobnews/plan\\_explan\\_view.jsp?nidx=190587&depth1=1&depth2=1&depth3=1](http://jobnjoy.com/portal/jobnews/plan_explan_view.jsp?nidx=190587&depth1=1&depth2=1&depth3=1)
- [6] 백준온라인저지(Baekjoon Online Judge)  
<https://www.acmicpc.net/>