요 약 문

과제목표(국문)	아이디어를 산출하고 이를 바탕으로 구현을 진행하여 개발 프로세스까지 완벽한 소프트웨어공학적 방법으로 구현하는 것으로 개발 과정을 이해하고 적용하는 실습 과제이다.
Purpose(영문)	It is a practical task to understand and apply the development process by calculating ideas and implementing them based on them and implementing the development process in a complete software engineering method.
과제 내용	N2T(Note to Test)에 대한 아이디어를 산출하고 이를 개발 프로세스에 적용해 최종적으로 N2T(Note to Test) 서비스를 제공함과 동시에 과정에서 여러 개발 능력을 향상시키고 협업 과정을 통해 협업을 학습하며 소프트웨어공학적 방법에 이해 및 적용을 진행한다. 우리 N2T(Note to Test) 서비스는 형식화된 노트 필기 내용을 통해 다양한 형태의 정리 노트를 제공하며 문제를 제공함으로써 학습 과정에서 큰 도움을 주고자 한다. 더하여 많은 사람들이 학습을 진행할 때 도움을 주는 것뿐만 아니라 우리가 라벨링 된 학습데이터를 통해 더 많은 활용이 가능할 것으로 기대한다. 이를 완성하기 위해 노트 필기 형식을 구현할 것이며 문제로 만들어 주기 위해 AI 특히 NLP를 통해 문제를 제공할 것이다.
기대효과 (응용분야 및 활용범위, 외부경진대회 출품 계획 포함)	개인이 공부한 내용을 토대로 문제를 제공함으로써 학습효과를 높여준다. 모여진 학습 데이터를 수집, 분석, 유통하여 데이터 산업에 진출할 수 있다. 다양한 교육 사이트 그리고 템플릿 사이트와 협력이 가능하여 이익을 창출할 수 있으며 질 높은 서비스를 제공할 수 있다. 캡스톤 디자인 경진대회, KB국민은행 소프트웨어 경진대회, SW 스타트업 창업 챌린지 공모전 등 대회에 참여를 할 계획이다.
핵심어(국문)	자동화, 형식, 인공지능, 자연어처리, 학습, 필기, 시험, 데이터, 라벨링
Keywords(영문)	Automation, Format, Artificial Intelligence, Natural Language Processing, Learning, Writing, Testing, Data, Labeling

공 란

목 차

그림 목차

표 목차

1 과제 개요

1.1 개발 배경(주제 선정 동기)

자유로운 분위기 속에서 시행한 브레인스토밍을 통해 4차 산업 혁명의 핵심 키워드 중 하나인 '자동화'와 우리 삶에서 필요한 '학습'이라는 주제를 찾게 되었다. 어떤 정보를 얼마나 정확하게 학습하고 활용할 수 있느냐는 요즘 시대에 필요한 능력이고 이 능력을 키우기 위한 최고의 전략으로 알려진 시험 효과와 인출 효과를 주제 안에 집어넣었다. 찾게 된 전략을 자동화라는 키워드에 강제 연결하여 내가 형식에 맞춰 작성한 노트 정리가 최고의 학습 전략인 시험으로 자동화하는 최종 주제 '한 번으로 끝내는 시험 완성(한시완)'을 가지고서 아이디어를 구체화하여 최종적인 아이템 이름을 'N2T(Note to Test)'로 결정했다.

1.2 과제 필요성, 중요성

1.2.1 SWOT

그림 1 SWOT

제공하는 서비스의 강점(Strength), 약점(Weakness), 기회(Opportunity), 위협(Threat)을 파악하는 SWOT 분석을 진행하였다. 이를 통해 긍정적 요인인 강점 및 기회와 위험요소인 약점 및 위협을 비교하여 서비스의 시장 상황을 파악하고, 강점과 기회는 극대화하고, 약점 및 위협은 보완할 수 있는 전략을 수립한다.

1.2.2 아이템 포지셔닝

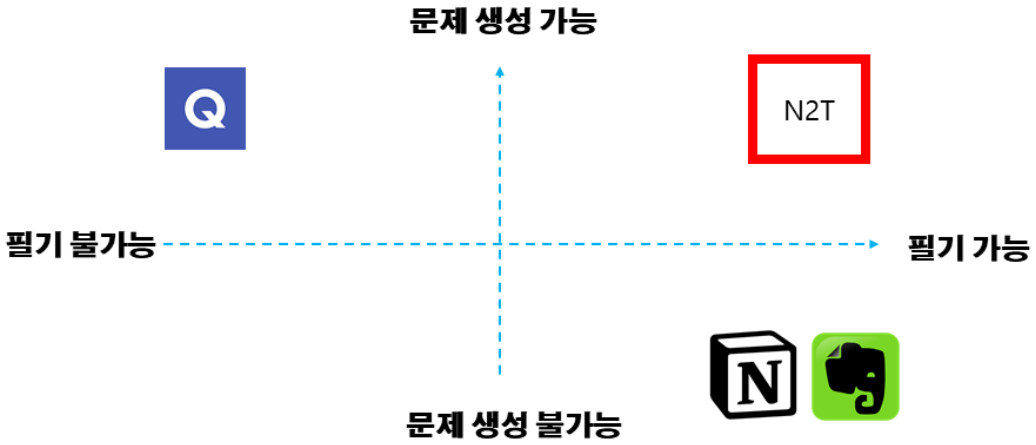

그림 2 아이템 포지셔닝

위 포지셔닝의 비교 회사를 다음과 같이 선정한 이유는 유사한 기능을 제공하는 서비스가 아직 없기 때문이다. 우리의 메인 기능인 문제 생성과 필기에 관해서는 각각 강력한 서비스가 많기 때문에 다음과 같이 포지셔닝하였고, 따라서 N2T가 두가지 기능이 동시에 가능한 사이트로서 의미를 가진다고 생각한다.

1.3 작품 개요

N2T(Note to Test)는 이름에서 알 수 있듯이 사용자의 노트 필기를 자동으로 시험 문제로 변환해주는 것이 주요 서비스이다. 사용자가 정해진 형식에 맞추어 작성한 노트 필기에서 키워드와 설명을 추출해 주관식 문제로 변환해준다. 문제의 답안은 바로 확인할 수 있도록 하고, hide & show 형식으로 버튼을 누르고 있는 동안 필기했던 부분을 띄워주어 관련 내용을 다시 상기할 수 있게 한다. 또한 추출한 키워드들을 중심으로 사용자의 노트를 보기 좋게 바꾸어 줄 수 있다. 한국사 시험공부를 예로 든다면, 사용자가 연대별로 삼국시대를 정리했을 때, <나라>, <왕> 키워드를 결합하여 각 나라별 왕을 기준으로 노트를 정리해줄 수 있다. 추가적인 기능으로는 프라이빗 노트를 제공하고, 다양한 템플릿을 등록 및 사용 가능할 수 있게 한다.

사용자가 형식에 맞춰 필기하면 이는 라벨링 된 데이터를 작성하는 것과 같기 때문에 경제가치가 있는 데이터를 확보할 수 있다. 이렇게 확보된 빅데이터를 활용하여 각 시험 카테고리별 기출 문제 제공 등 다양한 측면에서 활용방안을 찾을 수 있다.

1.4 과제 수행 관련 선행학습 결과

현 우리가 학습한 것들에 대해서만 기제를 진행하였고 앞으로 학습하는 것들을 추가 기제할 예정이다. 더하여 전북대학교 소프트웨어공학과에서 학습한 내용은 당연하게 학습이 완료되었다는 전제가 있다.

Web Page를 만드는 작업이기에 웹을 어떤 방식으로 구현할지 고민하였고 다음과 같은 추가 학습을 통해 Web page를 구상하려고 한다.

1. JS

자바스크립트를 통해 홈페이지 구축을 생각하였으며 최근 트랜드 상 많이 성장하고 있어 프로젝트 진행뿐만 아니라 학습 면에서도 크게 도움이 될 것이라 생각된다. 더하여 이전에 언어를 많이 경험하였지만 함수형 프로그램이 가능한 언어인 JS를 학습하는 것이 함수형 프로그램에 있어서도 큰 도움이 될 것이라 생각하였다. 그 외 문법적 요소는 이전 언어의 큰 틀을 벗어나지 않기 때문에 어렵지않게 학습이 가능하였다.

2. React

흔히 우리가 웹 프레임 워크로 알고 있는 React를 사용하여 구상할 계획이다. 사실 React는 웹 프레임워크란 말보다 사용자 인터페이스 만들기를 위한 JS 라이브러리가 더 알맞은 표현이다. React는 점점 많은 기능을 지원하고 강력한 사용자 인터페이스를 구현 가능하므로 프로젝트에도 적절하다고 판단하였다.

3. Webpack & Babel

웹페이지를 만들고 릴리즈 할 때 몇 가지 더 생각해야 하는 부분이 있는데 그중 하나가 여러 가지 브라우저에서 작동하게 만드는 것이다. 따라서 Babel을 통해서 98% 이상의 브라우저를 지원할 수 있게 패키징을 사용할 것이며 Webpack을 같이 사용하여 릴리즈 파일 최적화에도 신경을 쓰는 방식을 고려하였다. 많은 프레임 워크에서 자체적으로 지원하지만 직접 사용함으로써 학습적인 부분과 우리가 원하는 방향에 맞추어 개발할 예정이다.

4. 아직 학습을 계획 중인 것들

우리의 최종 목표에 다다르기 위해서는 사실 개발적인 부분보다 형식화와 NLP에 대한 비중이 더 크다. 그러므로 형식에 대해서 최적의 형식을 만들기 위해 다양한 공부를 진행할 예정이며 NLP 부분을 학습하여 우리의 기능을 더욱 활용성 있게 만들 계획이다. 따라서 앞으로 학습

계획에는 형식화와 NLP 부분이 중요하다.

1.5 과제 추진 전략 및 방법

1.5.1 스크럼 기반 애자일 방법론

애자일 방법론이란 말 그대로 기민한 개발을 가능하게 해주는 방법론을 말하는 것으로 무계획과 지나치게 많은 계획 사이의 타협점을 찾고자 하는 방법론이다. 이 애자일에는 여러 가지 방법론이 있지만, 우리 팀은 방법론 중 스크럼 기법을 사용하여 이 프로젝트를 진행하고자 한다. 스크럼이란 반복 점진적 개발 방법을 말하며 각 반복 주기(스프린트)를 통해 부분적으로 완성된 결과물이 만들어지게 된다. 스크럼에는 제품 책임자(product owner), 스크럼 마스터(servant leader), 개발팀(developer team)이라는 3가지 역할이 있다.

제품 책임자는 제품의 백로그를 관리, 작성하고 이해관계자로부터 요구사항을 추출한 뒤 스프린트의 우선순위를 관리, 조정한다. 우리 팀은 외부의 이해관계자가 있는 것이 아니므로 팀 전체가 제품 책임자가 되어 관리할 예정이다. 스크럼 마스터는 팀원들을 코칭하고 개발팀이 프로젝트 진행 중 생기는 문제를 잘 해결하도록 돕는 역할을 한다. 또한 스크럼을 잘 알고 수행할 수 있도록 책임을 가지고 스크럼 이론, 규칙을 따르도록 관리한다. 팀장 문석암은 자신의 경험을 바탕으로 스크럼 마스터가 되어 팀에게 필요한 협업 능력과 발생하는 문제를 잘 해결하도록 하는 역할을 할 예정이다.

개발팀은 요구사항을 개발하고 테스트하는 팀이다. 우리 팀은 각자가 개발의 팀원으로 최선을 다해 스프린트 목표를 달성하기 위해 노력한다.

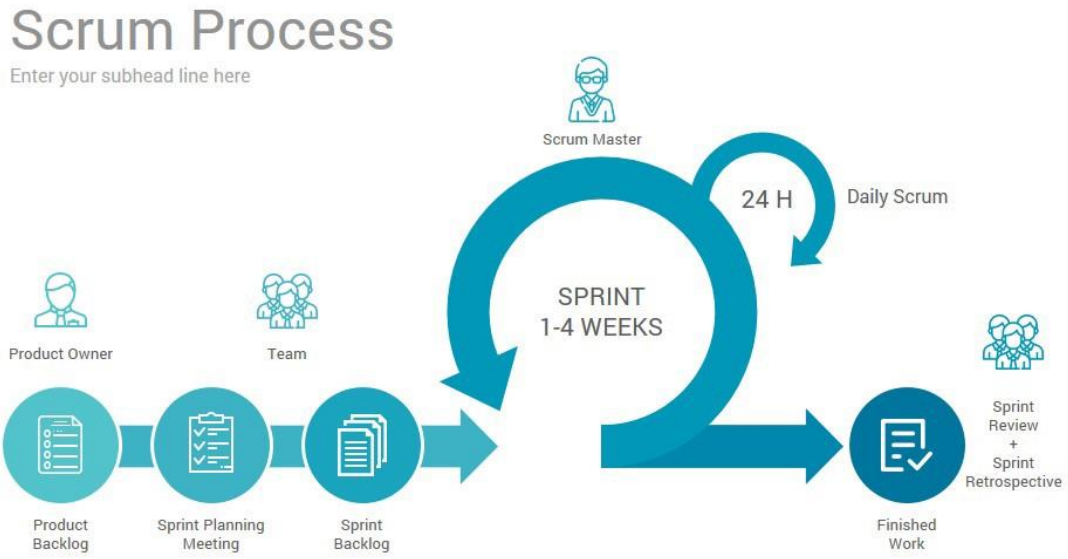

그림 3 Scrum Process

1.5.2 WBS (Work Breakdown Structure)

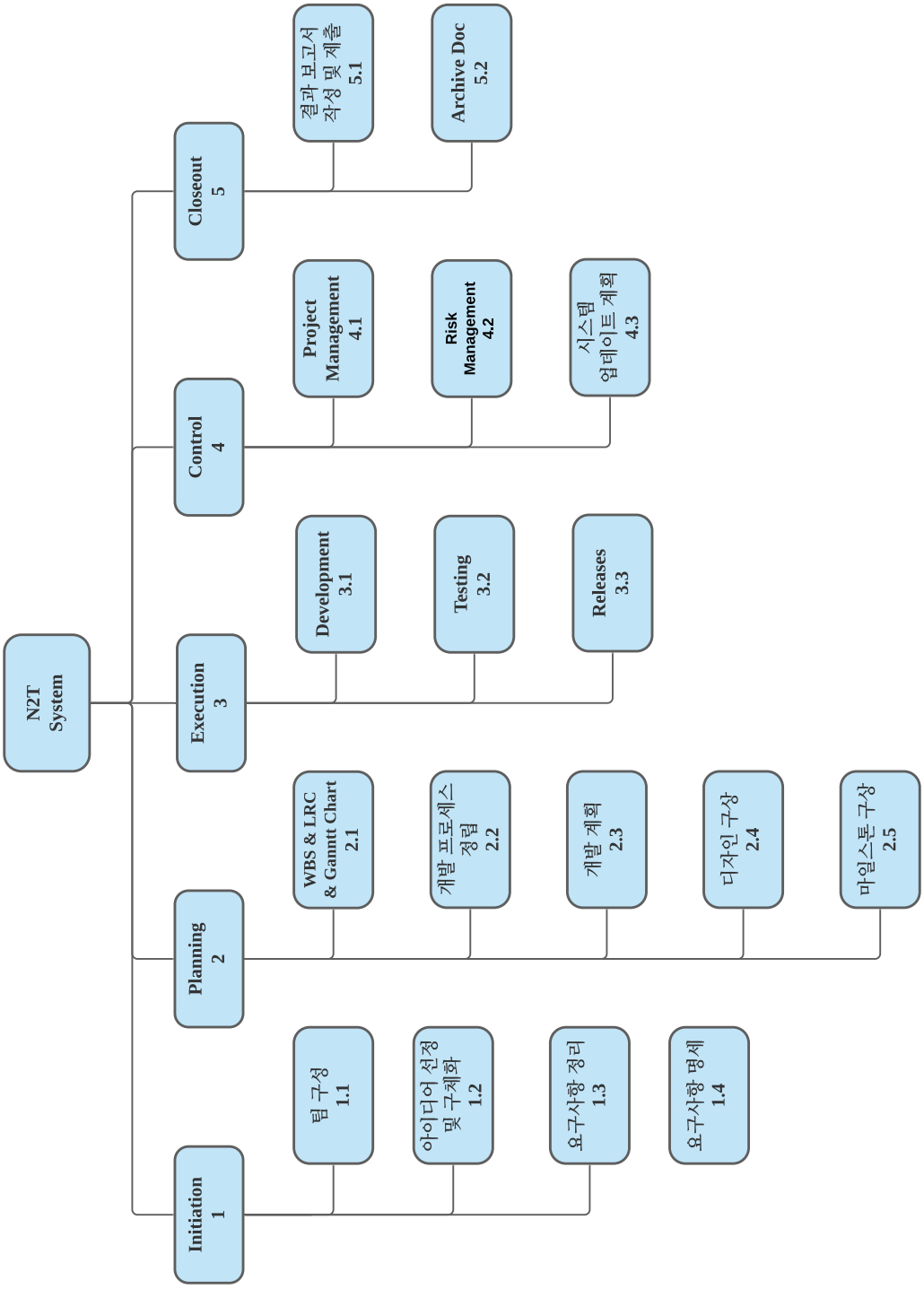

그림 4 WBS

1.5.3 LRC (Linear Responsibility Chart)

	문석암	서은지	유현진	임원용
1. Initiation				1
1.1 아이디어 선정 및 구체화	2	3	3	1
1.2 요구사항 정리	2	3	3	1
1.3 요구사항 명세	3	2	2	1
2. Planning			1	
2.1 WBS & LRC & Gantt Chart	2	2	1	3
2.2 개발 프로세스 정립	2	3	3	1
2.3 개발 계획	1	3	3	3
2.4 디자인 구상	3	1	2	3
2.5 마일스톤 구상	1	2	2	2
3. Execution	1			
3.1 Development	1	2	2	2
3.2 Testing	1	2	2	2
3.3 Release	1	3	3	3
4. Control		1		
4.1 Project Management	1	3	2	3
4.2 Risk Management	3	1	3	3
4.3 시스템 업데이트 계획	2	3	3	1
5. Closeout		1		
5.1 결과 보고서 작성 및 제출	2	1	2	2
5.2 Archive Doc	1	3	3	3
Key:				
1 = Primary responsibility				
2 = Support/work				
3 = Review				
※Final Approval = 개발자 모두가 진행합니다.※				

표 1 LRC

1.5.4 간트 차트

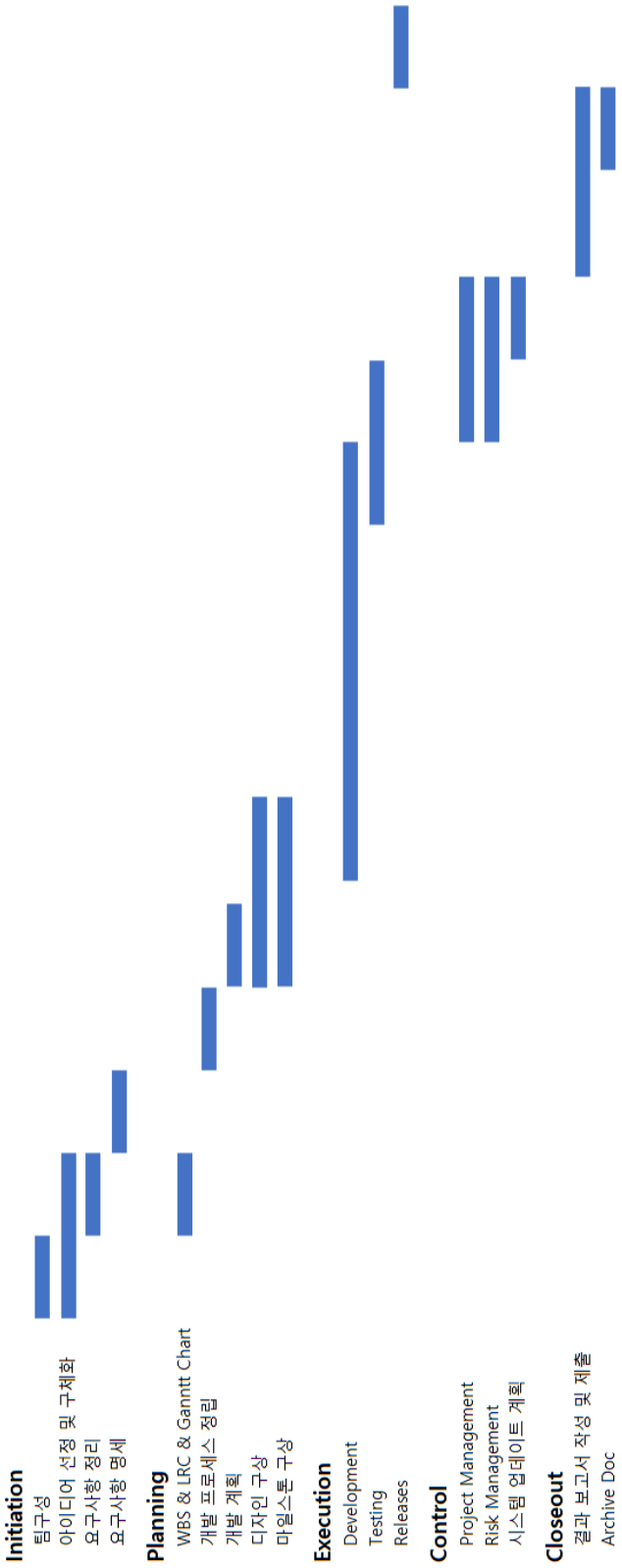

그림 5 간트 차트

1.5.5 현실적 제한 조건별 검토 반영

현실적 제한조건	상세 내용
산업표준	정보기술 응용을 준수하고 있다.
경제성	데이터라는 경제적 가치가 있는 자원을 얻을 수 있다.
윤리성	개인 정보 수집 및 데이터 이용 동의서를 제공함으로써 개인정보 및 저작권의 침해를 방지한다.
신뢰성	개인이 제공하는 정보를 통해 문제를 제공하기 때문에 문제 난이도에 대한 사용자의 기준이 다를 수 있다.
사용성	개인이 작성한 노트 필기를 통한 복습 및 시험 대비에 효과적이므로 사용성이 높다.
호환성	Web site이므로 인터넷을 통해 모든 기기에서 사용가능하다.
유지보수성	서버관리, 백업 및 복구 등의 데이터 유지보수 작업을 하여 유지보수성을 용이하게 한다.
미학	요약 내용을 깔끔하게 쓸 수 있도록 다양한 작성 템플릿을 제공함으로써 개인의 스타일로 정리를 할 수 있게 도와준다.
사회 • 정치 • 환경에 미치는 영향	학업을 중요시하는 사람들에게 복습의 효과를 높여주며 학습의 질을 높여준다.

표 2 현실적 제한 조건

1.6 과제 수행 결과 예상

1.6.1 예상되는 파급효과

1. 학습 성과를 높여준다.

자신이 공부한 내용을 토대로 문제를 제공함으로써 학습효과를 높여준다.

2. 데이터 산업 진출

사이트를 사용하는 사람들의 학습 데이터를 수집, 분석, 유통하여 가치를 창출하는 서비스로 나아갈 수 있다.

3. 템플릿 사업과 협력 가능

다양한 노트 템플릿, 시험지 템플릿을 가지고 있는 기업과 협력하여 사이트를 이용하는 사용자들에게 질 좋은 디자인의 템플릿을 제공한다.

4. 다양한 교육 사이트와 협력 가능

해당 교육 사이트 자료 모음집이라는 카테고리를 만들고 해당 교육 사이트의 강의를 듣는 사람들이 우리의 사이트를 통하여 학습 및 복습을 하고, 이후 정리된 학습 데이터들을 따로 모아 제공하는 형식으로 이익을 창출할 수 있다.

2 과제 수행 내용 및 결과

2.1 요구사항 명세서

2.1.1 기능적 요구사항

로그인

요구사항 ID		RQ-001	요구사항 명	로그인 기능
개요		로그인과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	사용자 인증 수단으로 필요 아이디 비밀번호 찾기 기능 제공 Oauth 수단 사용가능(Google 로그인)		
	화면	로그인 화면	중요도	중

표 3 로그인 요구사항 명세서

회원가입

요구사항 ID		RQ-002	요구사항 명	회원가입 기능
개요		회원가입과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	Oauth를 통한 가입 가능 자체 회원가입 기능 이메일 인증 기반 회원가입 아이디, 이메일 중복 확인 하나의 이메일 당 하나의 계정 지원 회원가입 완료 시 사용자 로그인 상태		
	화면	회원가입 화면	중요도	중

표 4 회원가입 요구사항 명세서

튜토리얼

요구사항 ID		RQ-003	요구사항 명	튜토리얼
개요		튜토리얼과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	튜토리얼 GIF 파일 제공 미리 만들어진 노트를 통한 실습 제공		
	화면	튜토리얼 화면	중요도	하

표 5 튜토리얼 요구사항 명세서

내 노트

요구사항 ID		RQ-004	요구사항 명	내 노트
개요		내 노트와 관련된 기능적 요구사항		
요구 사항 내역	상세설명	노트 이름 검색 기능 제공 노트 생성 버튼을 통해 public / private 노트 생성 노트 수정 버튼을 통해 노트 수정 노트 삭제 버튼을 통해 노트 삭제 이름 바꾸기 버튼을 통해 노트 이름 수정 폴더 추가 및 삭제 기능 제공		
	화면	내 노트 화면	중요도	상

표 6 내 노트 요구사항 명세서

노트 필기

요구사항 ID		RQ-005	요구사항 명	노트 필기
개요		노트 필기와 관련된 기능적 요구사항		
요구 사항 내역	상세설명	필기 에디터 기능 제공 분할 화면을 통한 미리보기 제공 템플릿 변경 버튼을 통해 노트 형식 변경		
	화면	노트 필기 화면	중요도	상

표 7 노트 필기 요구사항 명세서

필기 에디터

요구사항 ID		RQ-006	요구사항 명	필기 에디터
개요		필기 에디터와 관련된 기능적 요구사항		
요구 사항 내역	상세설명	툴바에 커스텀 태그 버튼 및 저장 버튼, 테스트 버튼 제공 최대 라인 수 제한 사용자가 페이지를 구분할 수 있도록 커스텀 자동저장 기능 제공 Ctrl+s, Ctrl+z, insert, delete 등의 기본 기능 제공		
	화면	노트 필기 화면	중요도	상

표 8 필기 에디터 요구사항 명세서

문제 제공

요구사항 ID		RQ-007	요구사항 명	문제 제공
개요		문제 제공과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	사용자 노트 및 커스텀 태그 기반 빈칸 문제 제공 빈칸에 답안 작성 빈칸 제출 가능		
	화면	문제 풀이 화면	중요도	상

표 9 문제 제공 요구사항 명세서

채점

요구사항 ID		RQ-008	요구사항 명	시험결과 확인 및 채점
개요		채점과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	분할 화면을 통해 좌측에는 시험 채점 결과, 우측에는 관련 필기 제공		
	화면	채점 화면	중요도	상

표 10 채점 요구사항 명세서

노트 검색

요구사항 ID		RQ-009	요구사항 명	노트 검색
개요		노트 검색과 관련된 기능적 요구사항		
요구 사항 내역	상세설명	관리자가 지정한 종목별 카테고리 분류 사용자가 지정한 태그 기반 노트 공유 공유된 노트 검색 기능 제공 - 회원 / 비회원 모두 이용 가능 - 제목 / 카테고리 / 사용자 태그 기반 검색 - 공유 노트 다운로드 가능		

표 11 노트 검색 요구사항 명세서

2.1.2 비기능적 요구사항

에디터 편리성

요구사항 ID		NRQ-001	요구사항 명	에디터 편리성
개요		에디터 편리성과 관련된 비기능적 요구사항		
요구 사항 내역	상세설명	필기 에디터가 보편적인 온라인 에디터나 메모장이 제공하는 편안한 반응을 제공해야한다.		
	화면	노트 필기 화면		

표 12 에디터 편리성 요구사항 명세서

공유 노트 검색

요구사항 ID		NRQ-002	요구사항 명	공유 노트 검색
개요		공유 노트 검색과 관련된 비기능적 요구사항		
요구 사항 내역	상세설명	공유 노트 검색 속도는 3초 이내여야 한다. 동의어나 유의어를 활용해 검색할 수 있어야 한다.		
	화면	노트 검색 화면		

표 13 공유 노트 검색 요구사항 명세서

2.2 유스케이스 명세서

유스케이스 이름	로그인
번호	1
요약	사용자가 로그인을 한다.
액터	사용자(User)
사전 조건	회원가입이 되어있어야 한다.
메인 시퀀스	1. 사용자가 로그인 페이지를 요청한다. 2. 사용자가 ID/Password 입력 후 로그인을 요청한다. 3. 로그인을 진행한다.
대안 시퀀스	2-1. ID/Password를 잘못 입력했을 경우, Login Fail 메시지를 보낸 후, 다시 입력하도록 한다.
사후 조건	로그인 완료 후 메인 페이지로 이동한다.

표 14 로그인 유스케이스 명세서

유스케이스 이름	회원가입
번호	2
요약	사용자가 회원가입을 한다.
액터	사용자(User)
사전 조건	
메인 시퀀스	1. 사용자가 회원가입을 요청한다. 1.1 사이트 회원가입 1.2 Oauth 인증 회원가입 2. 이름, 이메일, ID, Password, Password확인을 입력한다. 3. 회원가입을 진행한다.
대안 시퀀스	2-1. ID, E-mail 중복 확인 버튼을 통해 결과를 alert로 보낸 후, 회원가입을 진행하도록 한다.
사후 조건	회원가입 완료 후 로그인 페이지로 이동한다.

표 15 회원가입 유스케이스 명세서

유스케이스 이름	튜토리얼 (Tutorial)
번호	3
요약	사용자가 튜토리얼을 진행한다.
액터	사용자(User)
사전 조건	
메인 시퀀스	1. 사용자가 튜토리얼 페이지를 요청한다. 2. 튜토리얼 페이지를 제공한다. 3. 사용자는 간단한 설명서를 읽는다. 4. 시험보러가기 버튼을 눌러 실습페이지로 이동한다. 5. 사용자는 예제를 통해 시각화 실습을 진행한다.
대안 시퀀스	1-1.메인 페이지 하단에 있는 튜토리얼 GIF를 제공한다.
사후 조건	튜토리얼의 실습을 채점해주는 페이지로 이동한다.

표 16 튜토리얼 유스케이스 명세서

유스케이스 이름	내 노트 보기 (My note)
번호	4
요약	사용자가 마이 노트 페이지로 이동한다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다.
메인 시퀀스	1. 사용자는 헤더에 있는 'My note' 버튼을 눌러 마이 노트 페이지를 요청한다.

대안 시퀀스	
사후 조건	마이 노트 페이지로 이동한다.

표 17 내 노트 보기 유스케이스 명세서

유스케이스 이름	노트 생성
번호	5
요약	사용자가 새로운 노트를 생성한다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다.
메인 시퀀스	1. 사용자는 New 또는 Private 버튼을 클릭한다. 2. 사용자는 모달을 통해 노트의 이름을 입력한다. 3. 추가 버튼을 눌러 노트를 생성한다.
대안 시퀀스	3-1. 사용자가 노트 생성을 원하지 않는다면 취소 버튼을 통해 모달을 종료한다. 3-2. 사용자가 노트 이름을 수정하고 싶다면 수정버튼을 눌러 이름을 수정한다. 3-3. 사용자가 노트 삭제를 원할 시, 삭제버튼을 눌러 노트를 삭제한다. 3-4. 사용자가 노트 이동을 원할 시, 이동버튼을 눌러 이동할 폴더를 선택한다.
사후 조건	사용자는 생성된 노트를 클릭하여 노트 필기 페이지로 이동한다.

표 18 노트 생성 유스케이스 명세서

유스케이스 이름	폴더 생성
번호	6
요약	사용자가 새로운 폴더를 생성한다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다.
메인 시퀀스	1. 사용자는 추가하기 버튼을 클릭한다. 2. 사용자는 모달을 통해 폴더의 이름을 입력한다. 3. 추가 버튼을 눌러 노트를 생성한다.
대안 시퀀스	3-1. 사용자가 폴더 생성을 원하지 않는다면 취소 버튼을 통해 모달을 종료한다. 3-2. 사용자가 기존의 폴더 삭제를 원할 시, 삭제 버튼을 통해 폴더를 삭제한다.
사후 조건	.

표 19 폴더 생성 유스케이스 명세서

유스케이스 이름	노트 필기
번호	7
요약	사용자가 노트 필기를 진행한다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다. 노트가 생성된 상태여야 한다.
메인 시퀀스	1. 사용자가 새로운 노트를 작성하거나 기존 노트 수정을 요청한다. 2. 필기 에디터와 미리보기 페이지를 제공한다. 3. 템플릿 변경 버튼을 누르면 뜨는 모달 창을 통해 템플릿을 변경한다. 4. 정해진 형식에 맞춰 필기를 작성한다. 5. 사용자가 작성한 필기 내용은 자동 저장된다.
대안 시퀀스	
사후 조건	Test 버튼을 통해 문제 풀기 페이지로 이동한다. 뒤로가기 버튼을 통해 내 노트 페이지로 이동한다.

표 20 노트 필기 유스케이스 명세서

유스케이스 이름	문제 풀기 및 확인
번호	8
요약	사용자가 문제를 푼다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다.
메인 시퀀스	1. 문제 풀기 페이지를 요청한다. 2. 사용자는 문제를 풀 수 있다. 3. 채점 페이지를 요청한다. 4. 2분할된 화면을 통해 정답 및 관련 필기 내용을 확인한다.
대안 시퀀스	
사후 조건	

표 21 문제 풀기 및 확인 유스케이스 명세서

유스케이스 이름	템플릿 사용 및 변경
번호	9
요약	사용자가 템플릿을 변경한다.
액터	사용자(User)
사전 조건	사용자는 로그인한 상태여야 한다. 사용자가 노트 필기 페이지에 있어야 한다.

메인 시퀀스	1. 사용자가 템플릿 변경 버튼을 클릭한다. 2. 사용할 템플릿을 선택한다.
대안 시퀀스	
사후 조건	선택한 템플릿을 적용한다.

표 22 템플릿 사용 및 변경 유스케이스 명세서

유스케이스 이름	노트 공유
번호	10
요약	사용자가 작성한 노트를 공유한다.
액터	사용자(User)
사전 조건	노트가 저장된 상태여야 한다.
메인 시퀀스	1. 사용자가 자신의 노트 공유를 요청한다. 2. 키워드 및 카테고리 분류를 선택한다. 3. 공유 노트 페이지로 이동한다.
대안 시퀀스	3-1. 원하는 카테고리나 키워드 기반 검색을 요청한다. 3-2. 권한에 따라 읽기 또는 내 노트로 가져오기를 요청한다.
사후 조건	공유를 하거나 공유된 노트를 제공받는다.

표 23 노트 공유 유스케이스 명세서

2.3 유스케이스 다이어그램

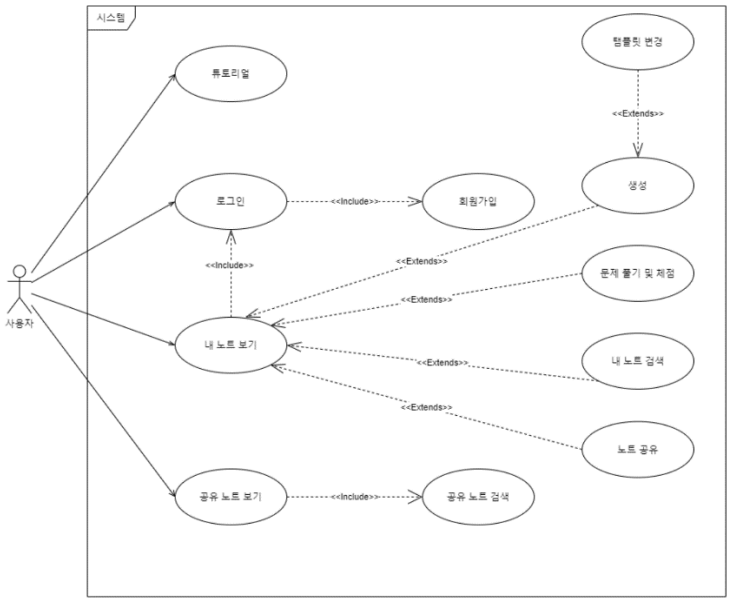

그림 6 유스케이스 다이어그램

2.4 스크럼 계획 회의

스크럼 계획 회의

Daily Scrum

1. 시간
   - 13시 30분
   - 최대 15분 이내
2. 내용
   - 오늘 할 일 계획
   - 오늘의 상태 및 컨디션
   - 현재 곤란하고 어려운 일

당일 회고

1. 시간
   - 23시 30분
   - 10분 전 개인 리뷰 진행
2. 내용
   - 만족스러운 점
   - 아쉬운 점
   - 개선 사항

정기 회의

- 월요일 20시
- 수요일 19시
- 토요일 15시

그림 7 스크럼 계획

진행사항을 매일매일 확인하기 위해 Daily Scrum을 진행한다. 이를 통해 팀원의 진행사항을 확실히 알 수 있으며 더하여 팀원들의 학습사항 등을 통해 다같이 발전 시킬 수 있는 기대가 있다.

2.5 제품 백로그

N2T 백로그						
화면	사용자 스토리	기능	Acceptance Criteria	Task	우선순위	비고
메인	홈페이지에서 원하는 기능으로 이동하고 싶다	내 노트 페이지 이동		FE : 내 노트 페이지로 이동 링크	4	한번에 작업
		검색 페이지 이동		FE : 검색 페이지로 이동 링크		
		로그인 페이지 이동		FE : 로그인 페이지로 이동 링크		
		튜토리얼 페이지 이동		FE : 튜토리얼 페이지로 이동 링크		
로그인	사용자가 사용을 위한 인증을 진행하고 싶다	로그인	Oauth 가동 (google)	Oauth 기능 추가	3	
				google을 통해 인증	3	
			아이디/비밀번호 시도 5회 제한	5회 이상 시도시 ??? 자단	4	
			자동 로그인	학습 : 사용 방안 고려하기	0	학습
			사용자 인증을 유지해야한다	FE : 사용자 인증 및 유지 방식 고려	0	고려
	사용자가 인증을 위해 사용자 등록을 하고싶다	회원가입	이메일을 통해 인증	BE : 인증메일 확인시 자동 인증	3	
				FE : 미인증 사용자 이메일 인증 안내 모달	3	
			이메일당 하나의 계정	BE : 중복확인 API	4	
				FE : error 406 발생시 홈으로	4	
			Oauth를 통한 가입 가능	BE : google Oauth 제공	3	
			아이디 중복 확인	BE : 아이디 중복 확인 미들웨 필요	3	
				FE: 입력값 검증	3	
	사용자가 잃어버린 아이디/비밀번호를 찾고싶다	아이디 찾기	이메일 인증 통한 아이디 전달	FE : 사용자에게 이메일을 받아 전달	4	
				BE : 이메일에 아이디를 담아서 전달	4	
		비밀번호 찾기	이메일 인증을 통한 비밀번호 변경	FE : 사용자에게 이메일, 아이디를 받아 전달	4	
			비밀번호 변경 링크 유효시간은 2시간	BE : 이메일에 변경 링크로 담아서 전달	4	
내 노트	노트를 작성하고 싶다	노트 작성	노트 필기 페이지로 이동	FE : 노트 필기 페이지로 링크	1	
	내 노트를 검색하고 싶다	내 노트 검색	검색 결과가 3초 이내로 나와야 한다	BE : 검색 API	2	
				FE : 검색 결과에 따라 문서를 보여줌	2	
	내 노트들의 디렉토리를 보고 싶다.	탐색기	내 노트들의 폴더 구조를 전부 보여준다	학습 : DB 구조에 대한 학습 필요	2	
				FE : API를 통한 파일 보여주기 및 이동	2	
	내 노트를 관리하고 싶다.	노트 관리	CRUD 기능을 세부 버튼으로 지원한다	FE : 노트에 관한 CRUD 기능을 지원	1	
				BE : CRUD API 필요	1	
튜토리얼	사용 방식에 대한 정보를 알고 싶다	튜토리얼	GIF가 정상적으로 실행되어야 한다	FE : Static GIF 실행	4	
			실제 기능에 링크를 두어 바로 이동 가능해야 한다	FE : 각 기능으로 이동 링크	4	
노트 필기	노트 필기를 하고싶다	필기 편집	기존 미정	학습 : MD 기반 독자적 형식 개발	0	학습
	내가 작성중인 노트를 보고싶다	미리보기	화면 분할을 가변적으로 지원해야한다	FE : 작성중인 것에 대해 형식기반 변환	2	
				FE : 화면에 분할가변적으로 변경 가능	2	
					2	
	노트의 템플릿을 변경하고싶다	템플릿 변경	템플릿 클릭 시, 즉시 변경되어야한다	FE : 미리보기 전환	1	
			머보기 클릭 시, 모달창을 띄워줘야한다	FE : 템플릿 선택 모달	1	
	노트 내용을 저장하고싶다	자동 저장	3초 안에 자동으로 저장되어야한다	BE: Local Storage(text area tag)	1	애매함
문제 풀기	내 노트를 기반으로 하는 문제를 풀고싶다	문제 풀기	빈 답안이 허용되어야한다	FE : 빈 답안도 채점 (채점은 틀림으로)	1	
			길이 제한 (실 문제 + 5글자 까지)	FE : 문제에 정답에 대해서 5글자 추가	2	
	사용자는 재시험을 보고 싶다	재시작	그 동안 작성한것을 전부 제거	FE : 항목 입력값 제거	1	
	사용자는 걸린 시간을 확인하고 싶다	타이머	로딩 완료로 타이머 시작	FE : 타이머 제공	3	
문제 채점	내가 푼 문제를 채점하고 싶다	문제 채점	대기 시간동안 splash 화면이 띄워져 있어야한다	FE : 응답 대기 시간동안 Splash 화면 보여주기	2	
		관련 필기 확인	문제와 관련된 필기를 우측 화면에 보여주어야한다	FE : 관련 필기 오른 쪽 표시	1	
			관련 필기를 클릭하면 해당 노트에 문제 관련 위치로 이동	FE : 노트 해당 위치로 링크	1	
노트 보기	내가 작성한 노트를 보고싶다	노트 보기	작성된 노트를 템플릿에 맞추어 보여준다	FE : 템플릿 기반 표시	1	
	노트를 가지고 시험을 보고싶다	문제 풀기 페이지 이동		FE : 문제 풀기 페이지 이동 링크	1	
	현재 노트의 템플릿을 바꾸고 싶다.	템플릿 변경	템플릿 클릭 시, 즉시 변경되어야한다	FE : 미리보기 전환	1	
			머보기 클릭 시, 모달창을 띄워줘야한다	FE : 템플릿 선택 모달	1	
공유 노트 검색	공유된 노트를 검색하고 싶다	공유 노트 페이지로 이동		FE : 공유 노트 페이지 이동 링크	3	

그림 8 백로그

백로그링크:
https://docs.google.com/spreadsheets/d/1tRDhWGoTO4ka3nAYuPAoZuNg4CDbVp4MCenHw5Uieaw/edit#gid=0

사용자 스토리 기반 백로그를 구현하였다. 사용자 스토리 기반으로 진행하여 사용자가 원하는 부분이 무엇인지 확실히 정의한다. 사용자의 입장에서 작성하였기 때문에 직접 개발한 개발자가 생각하지 못할 부분을 고려할 수 있는 장점이 있는 방식이며 우선순위를 통해 사용자가 생각하는 기능을 구현하기 위해 중요한 부분을 표현하였다.

2.6 목업 및 디자인 시안

2.6.1 메인 페이지

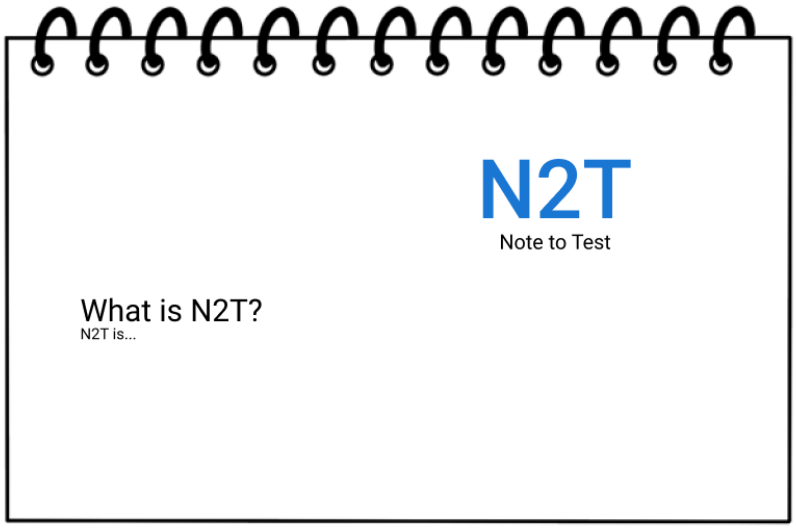

그림 9 메인 페이지 목업

그림 10 메인페이지 디자인 시안

2.6.2 마이 페이지

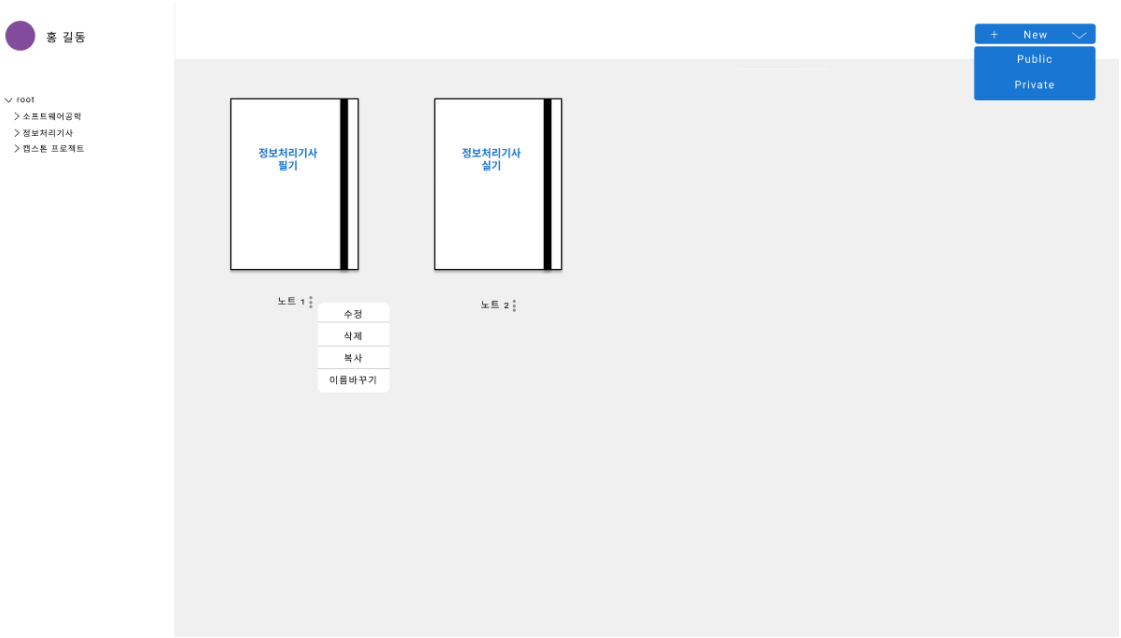

그림 11 마이페이지 목업

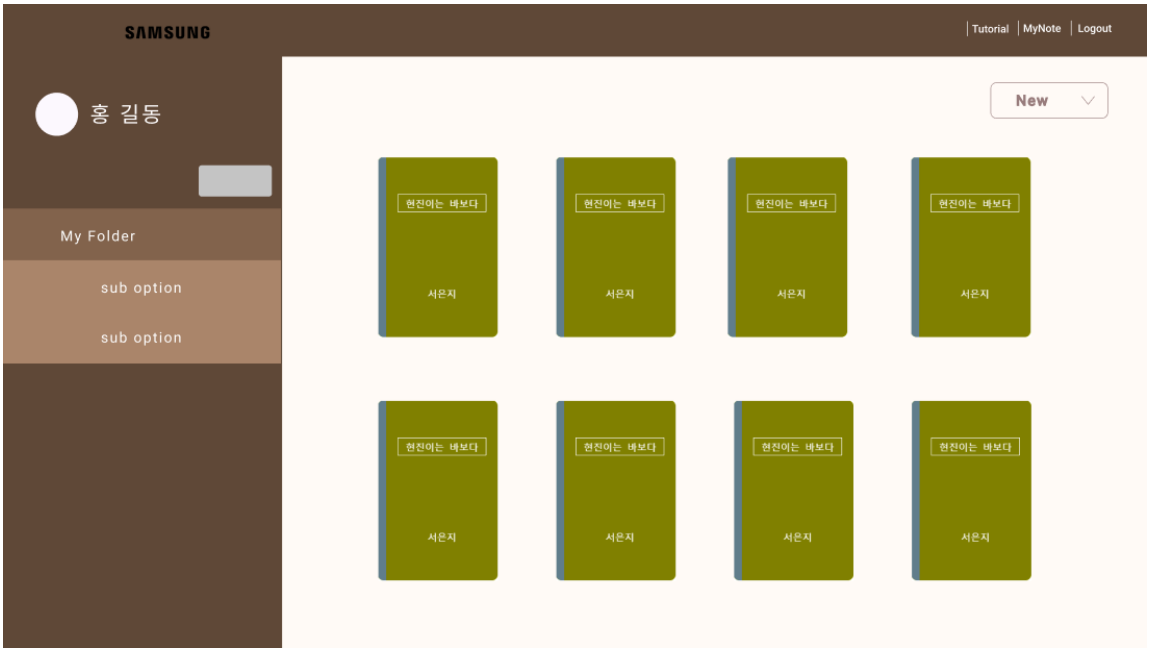

그림 12 마이페이지 디자인 시안

2.6.3 에디터 페이지

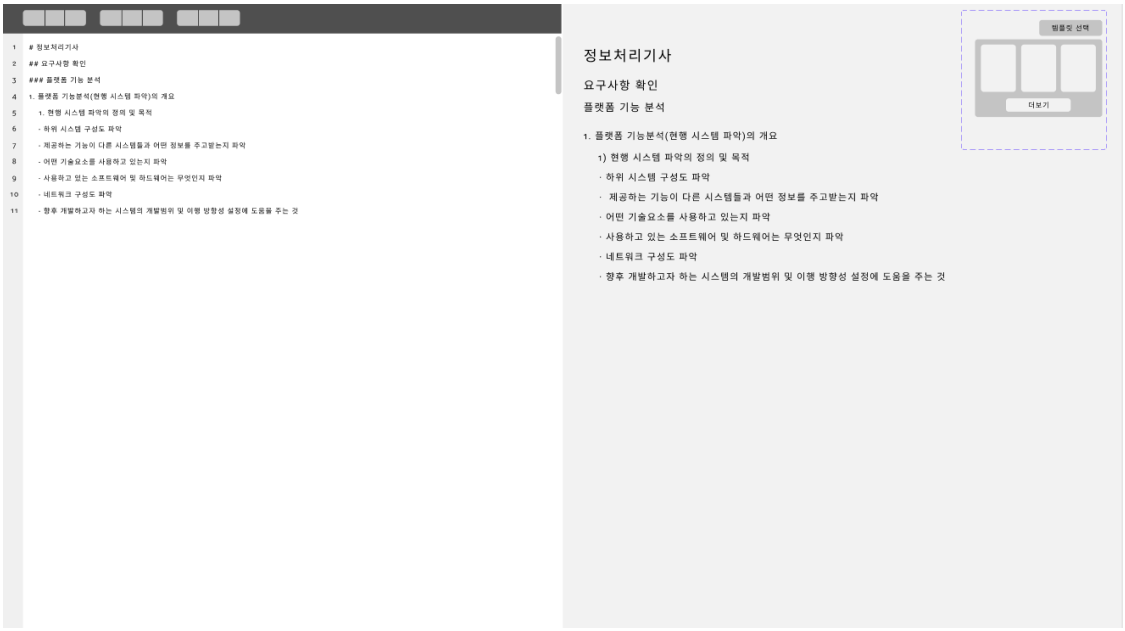

그림 13 에디터 페이지 목업

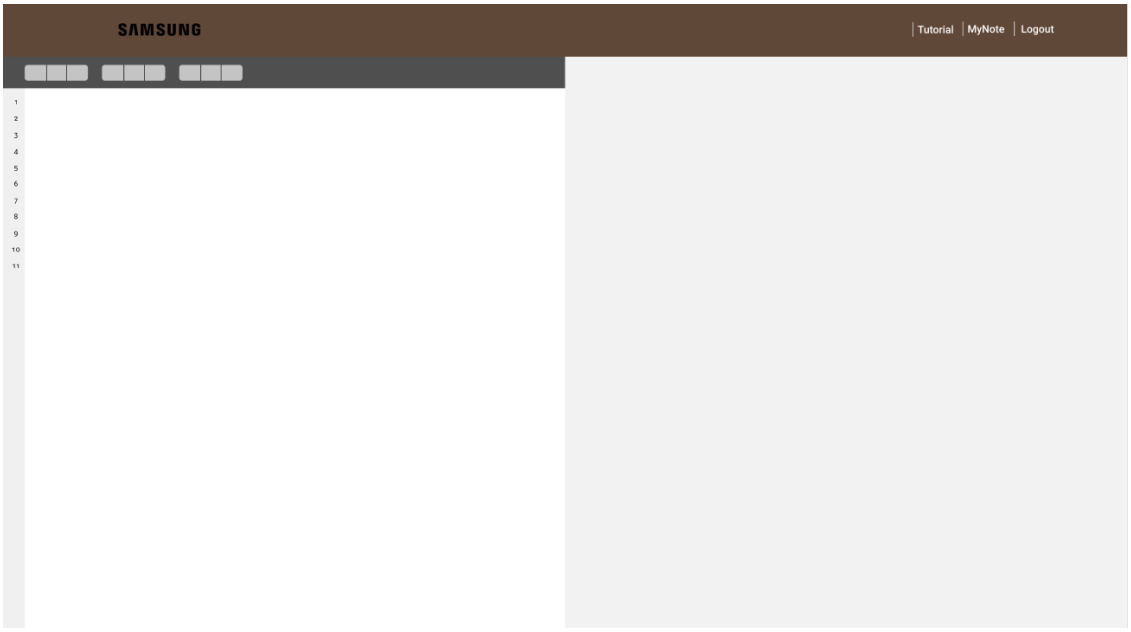

그림 14 에디터 페이지 디자인 시안

2.6.4 시험 보기 페이지

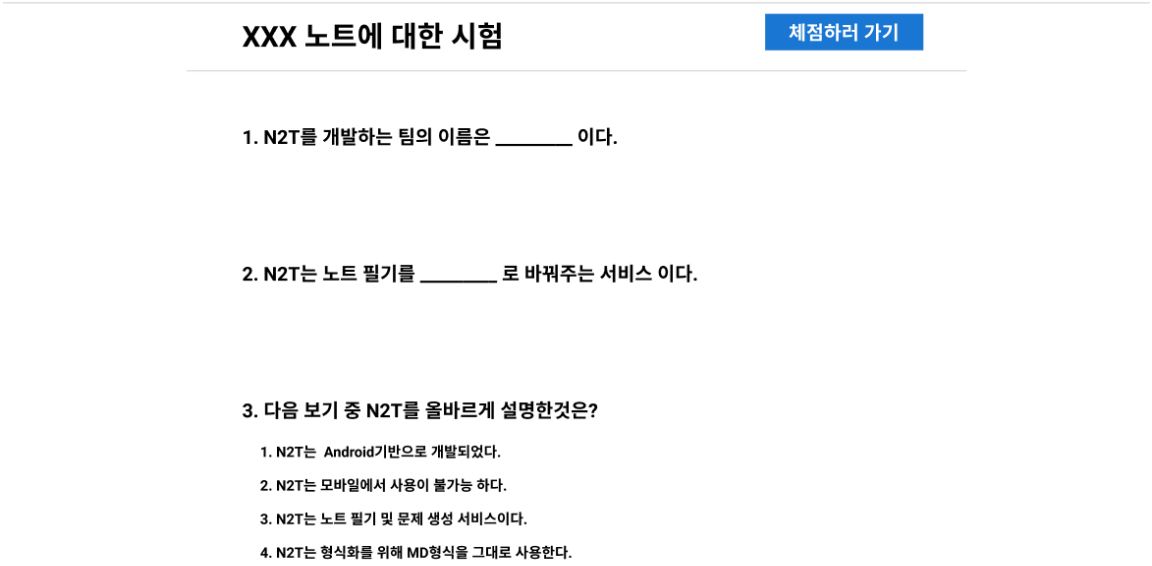

그림 15 시험 보기 페이지 목업

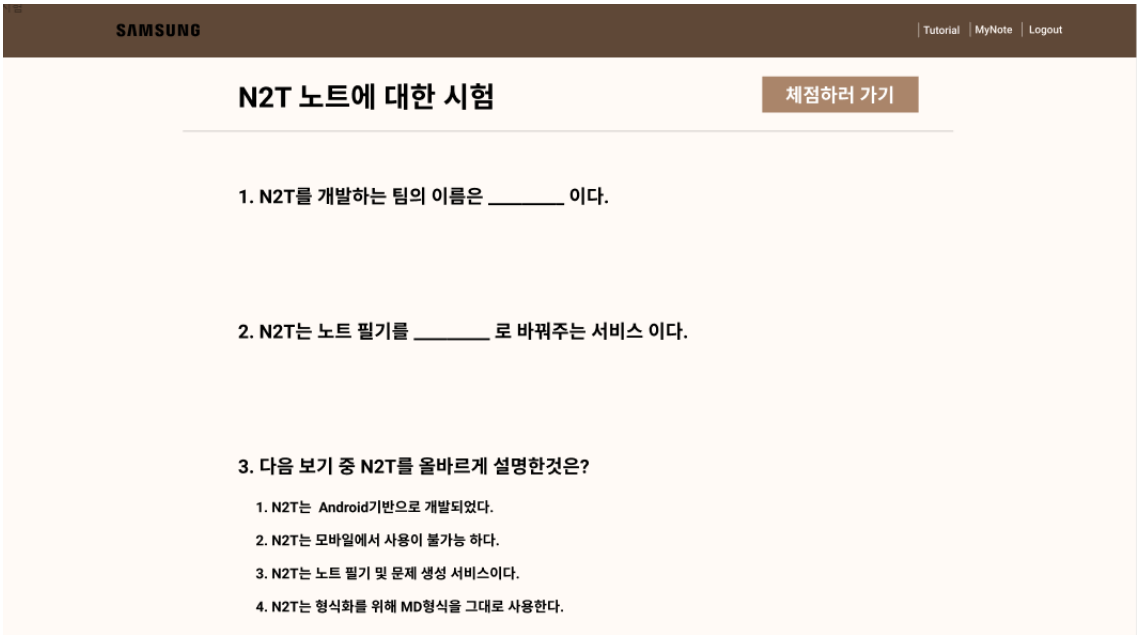

그림 16 시험 보기 페이지 디자인 시안

2.6.5 채점 페이지

그림 17 채점 페이지 목업

그림 18 채점 페이지 디자인 시안

2.6.6 검색 페이지

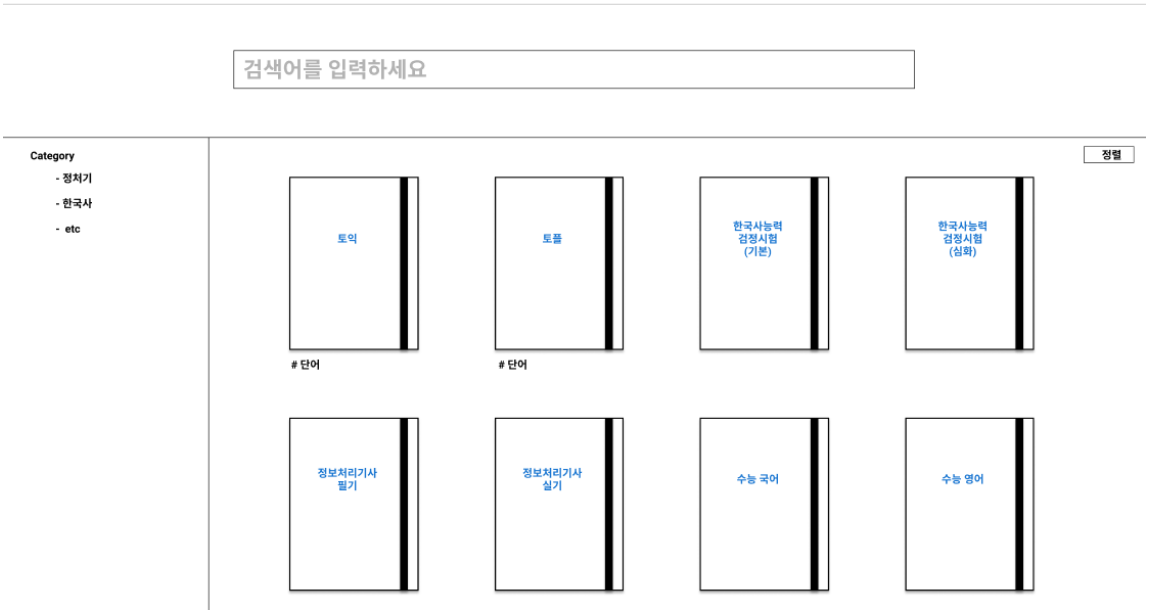

그림 19 검색 페이지 목업

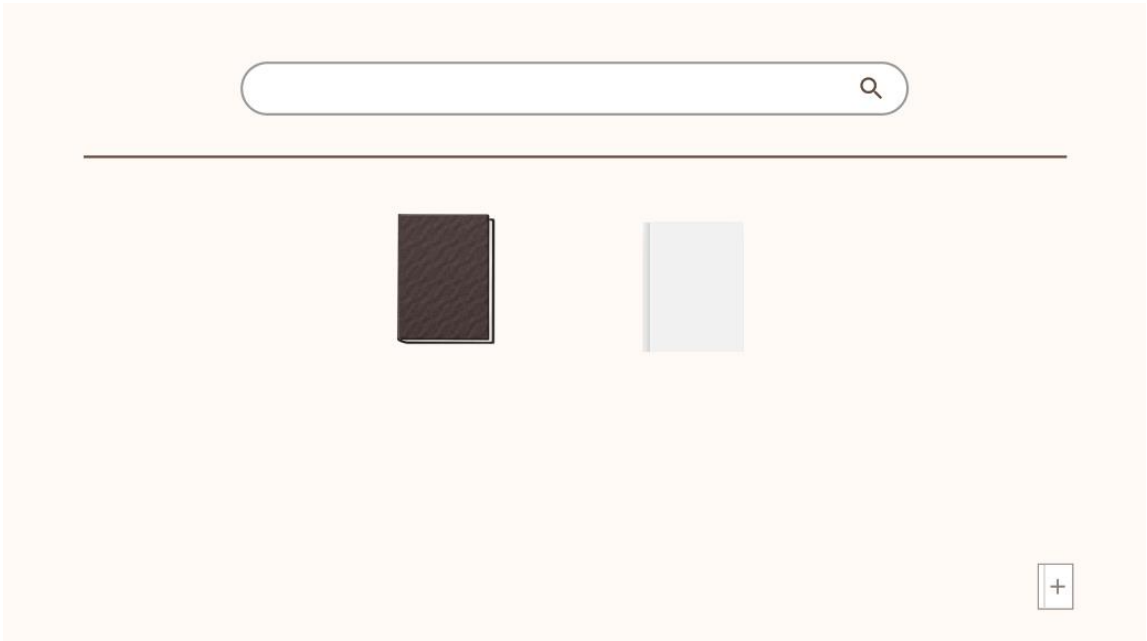

그림 20 검색 페이지 디자인 시안

2.7 DB 설계

2.7.1 초기 설계 다이어그램

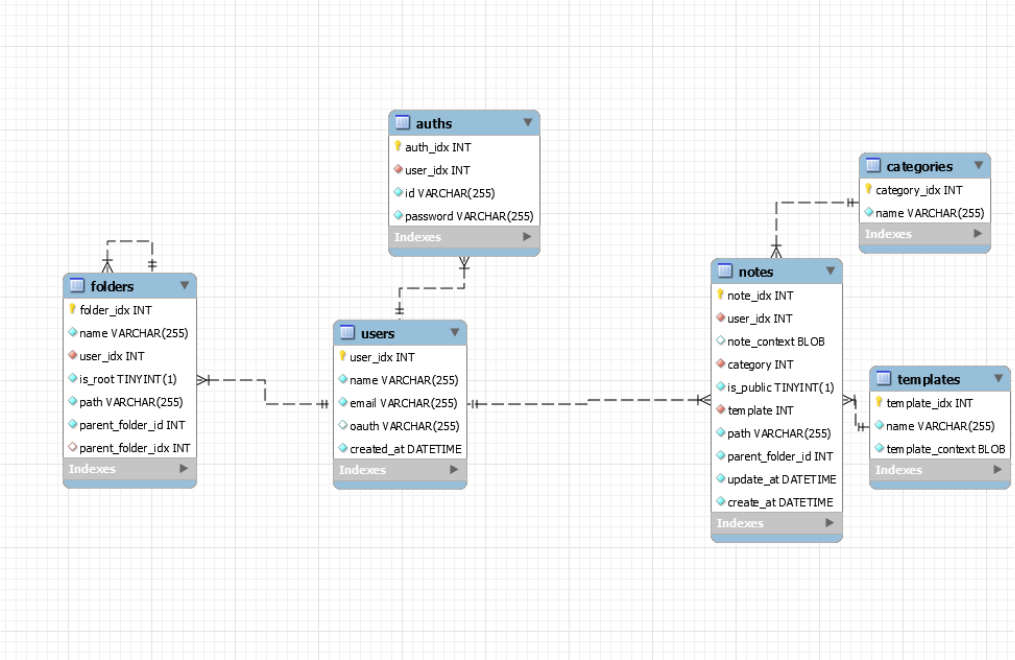

그림 21 초기 설계 다이어그램

다음과 같이 DB를 구상하였다. 우선 각 테이블은 전부 연관을 가지며 이를 통해 DB 접근 횟수를 감소시키고자 하였다.

2.7.2 최종 설계 다이어그램

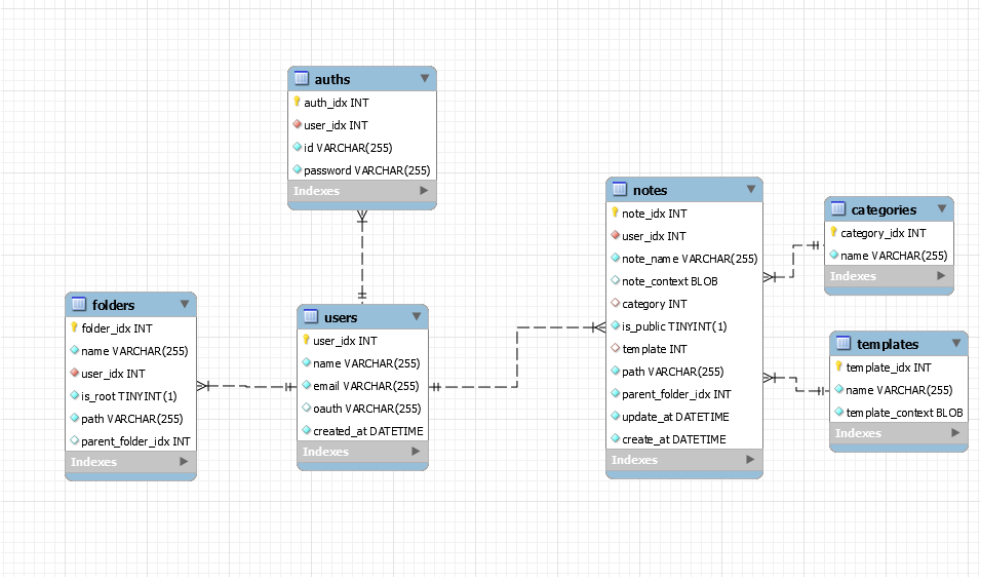

그림 22 최종 설계 다이어그램

거의 동일하나 Folder에서 논리적 오류정도를 개발 중 발견하고 수정하였으며 오타 문제도 해결하였다.

2.8 구현

2.8.1 로그인 구현

그림 23 로그인 목업

위와 같은 목업을 통해 아래와 같은 디자인 시안을 만들었다.

그림 24 로그인 다자인 시안

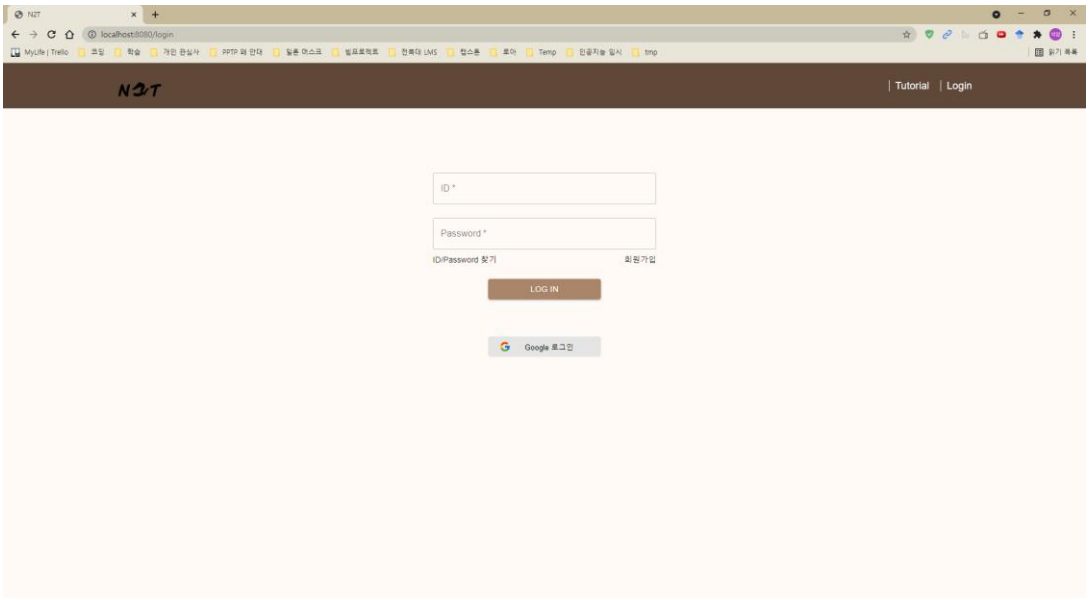

그림 25 로그인 페이지 구현 모습

위는 최종 개발의 결과물이다. 우선 차이점은 Facebook Login을 제외하여 진행하였고 그 외 부분은 구현을 진행하였다.

우선 인증방식을 고려하였다. Session과 JWT 방식을 고려하였고 결과적으로 JWT를 사용하기로 하였다. 또한 이 Token을 저장하기위한 방식을 고려하였는데 우선 Cookie, LocalStorage를 고려하였다. 결과적으로 LocalStorage를 사용하기로 하였으며 이를 위해 다음과 같은 코드를 구현하게 되었다.

BE/config/passport.config.js

파일에서 Google Login 과 Local Login로 로그인 가능하게 추가하였다.

```
const passport = require('passport');
const passportJWT = require('passport-jwt');
const LocalStrategy = require('passport-local').Strategy;
const GoogleStrategy = require('passport-google-oauth20').Strategy;
```

다음 코드와 같이 Passport 라이브러리의 도움을 받아서 진행하였다.

```
async login(req, res) {
    passport.authenticate('local', { session: false }, (err, user) => {
     if (err || !user) {
       return res.status(400).json({
         result: false,
         message: 'Login Fail',
       });
     }
     req.login(user, { session: false }, (err) => {
       if (err) {
         res.send(err);
       }
       const token = jwt.sign(user.toJSON(), process.env.JWT_SECRET, { ex
piresIn: '2h' });
       return res.json({ user, token });
     });
   })(req, res);
  },
```

다음과 같이 AuthController Login method를 추가하였다 passport를 Middleware로 지나간 후 나온 결과 중 Token을 넘겨준다.

```
  async googleLogin(req, res) {
    const { user } = req;
    const token = jwt.sign(user.toJSON(), process.env.JWT_SECRET, { expir
esIn: '2h' });

    res.cookie('token', token);
   res.redirect('http://localhost:8080/');
  },
```

다음과 같이 google로그인또한 같은 방식으로 전달한다. 특별한 점은 CallBack 있기 때문에 다시 우리의 페이지로 돌려줄 필요가 있다는 점이고 이를 해결하기 위해 Cookie에 임시적으로 값을 가진다.

```
const moveTokenCookieToLocalStorage = () => {
  try {
```

```
    const token = document.cookie
      .split(';')
      .find((cookie) => cookie.includes('token='))
      .split('=')[1];
    deleteCookie('token');
    if (token) {
      storageHandler.set(token);
      window.location.replace('/');
    }
  } catch (e) {
    // return undefined;
  }
};

const deleteCookie = function (name) {
  document.cookie = `${name}=; expires=Thu, 01 Jan 1999 00:00:10 GMT;`;
};
```

따라서 FE에서 인증정보를 처리할 경우 Cookie에 있는 값을 Local Storage에 추가하였다.

2.8.2 회원가입 구현

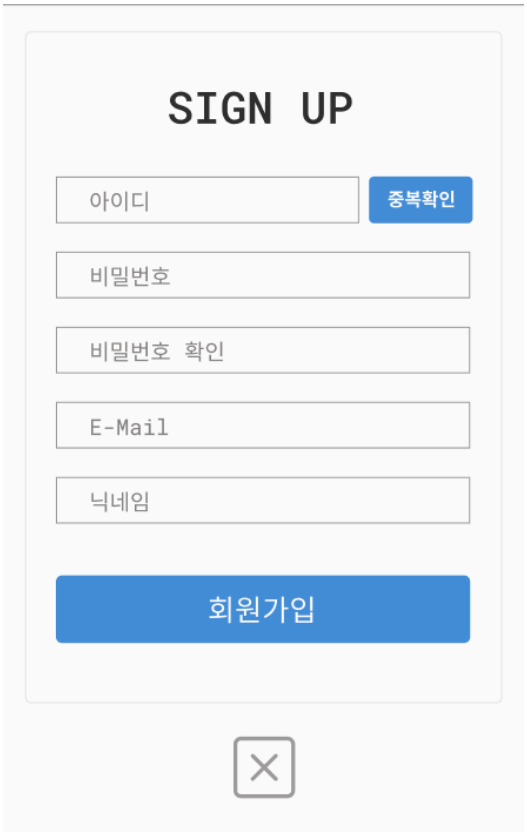

그림 26 회원가입 모달 목업

위와 같은 목업을 통해 구현을 진행하였고 결과적으로 다음과 같은 결과물을 얻을 수 있었다.

그림 27 회원가입 모달 구현 모습

위 사항에서 각각의 요소를 받아서 DB에 저장하는 API를 불러준다 여기서 중요한 것은 검증의 절차를 가지는 것이다. 따라서 다음 코드를 통해서 검증의 절차를 추가하였다.

```
const onEmailHandler = (e) => {
    setEmail(e.currentTarget.value);
     setCheck({
       ...check,
       email: false,
     });
  };
```

다음과 같은 Handler를 통해 검증의 여부를 기록한다 아이디와 이메일을 동일하게 진행하며 이를 통해 둘 다 검증 된 경우에만 회원가입이 가능하도록 되어 있다.

2.8.3 Custom MD 구현

Rule을 추가하기 위해 기존 MD lib에서 가능한 커스텀을 진행하였다. 우선 Rule 의 개념에서 우리가 원하는 Rule을 추가할 방법을 고안하였고

```
const underLineFun = (state, silent, tg, name) => {
  let {pos} = state;
```

```
  const start = pos;
  const max = state.posMax;
  const marker = tg;
  const checkTg = (pos) => {
    for (let i = 0; i < tg.length; i += 1) {
      const tmp = state.src.charCodeAt(pos + i);
      if (tmp !== tg.charCodeAt(i)) {
        return false;
      }
    }
    return true;
  };

  if (!checkTg(pos)) {
    return false;
  }

  pos += 2;

  // ???
  while (pos < max && checkTg(pos)) {
    pos += 2;
  }

  const openerLength = marker.length;

  if (state.backticksScanned && (state.backticks[openerLength] || 0) <= s
tart) {
    if (!silent) {
      state.pending += marker;
    }
    return true;
  }
  let matchEnd = pos;
  let matchStart;
  let closerLength;
  let token;
  while (state.src.indexOf(tg, matchEnd) !== -1) {
    matchStart = state.src.indexOf(tg, matchEnd);
    // 이건 왜필요해???
     matchEnd = matchStart + 2;

     // 이건 왜필요해???2222222
     while (matchEnd < max && state.src.charCodeAt(matchEnd) === '@') {
       matchEnd += 1;
     }

     closerLength = matchEnd - matchStart;

     if (closerLength === openerLength) {
       if (!silent) {
         token = state.push(name, 'test', 0);
         token.markup = marker;
```

```
        token.content = state.src
        .slice(pos, matchStart)
        .replace(/\n/g, ' ')
        .replace(/^ (.+) $/, '$1');
      }
      state.pos = matchEnd;
      return true;
    }
    state.backticks[closerLength] = matchStart;
  }
  state.backticksScanned = true;
  if (!silent) state.pending += marker;
  state.pos += openerLength;
  return true;
};
```

다음과 같은 함수를 통해 룰에 대한 부분을 함수화 시켰고 이를 통해 다음과 같은 코드로 일정한 규칙을 가진 Rule을 추가하고 이를 바탕으로 Render를 추가하여 적용된 것을 HTML에 원하는 태그로 변환할 수 있게 하였다.

```
md.inline.ruler.push('blue', (state, silent) => {
    return customFun(state, silent, '@b', 'blue');
  });

  const blueRender = (tokens, idx) => {
    return `<test><input type='text' id='test'><span id='blue'>${tokens[i
dx].content}</span></test>`;
  };
```

다음과 같이 함수를 사용해서 원하는 부분을 간단하게 구현하게 되었다.

2.8.4 MD Editor 구현

그림 28 에티터 페이지 디자인 시안

다음과 같은 디자인 시안을 가지고서 다음과 같은 결과물을 구현하였다.

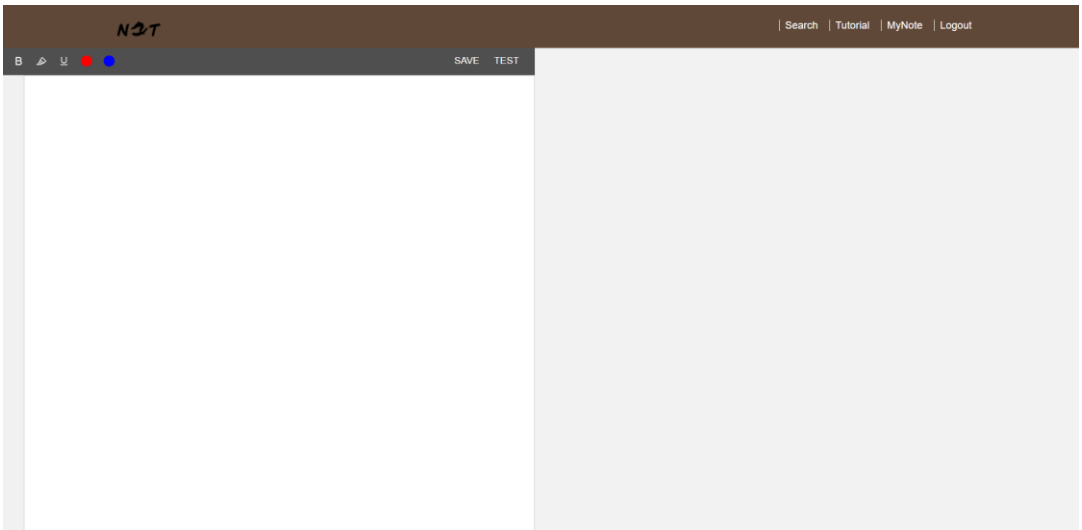

그림 29 에디터 페이지 구현 모습

왼쪽에 작성된 부분을 실시간으로 감지하여 글의 라인수를 표기해주며 오른쪽에는 HTML로 변환된 부분이 보인다.

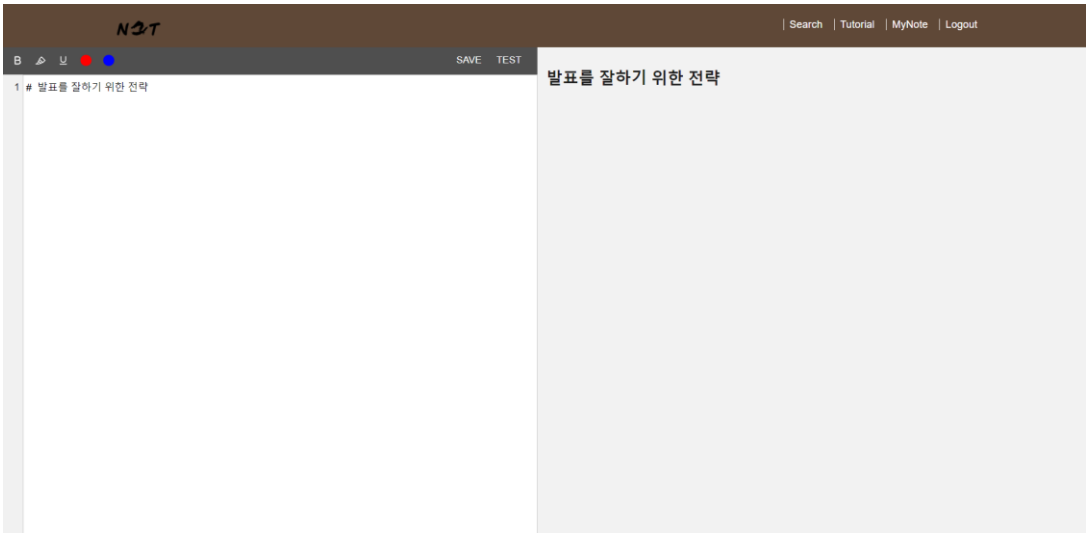

그림 30 에디터 페이지 예시

다음이 작성한 예시이다. 실시간으로 변환을 진행한다.

2.8.5 시험 문제 변환 구현

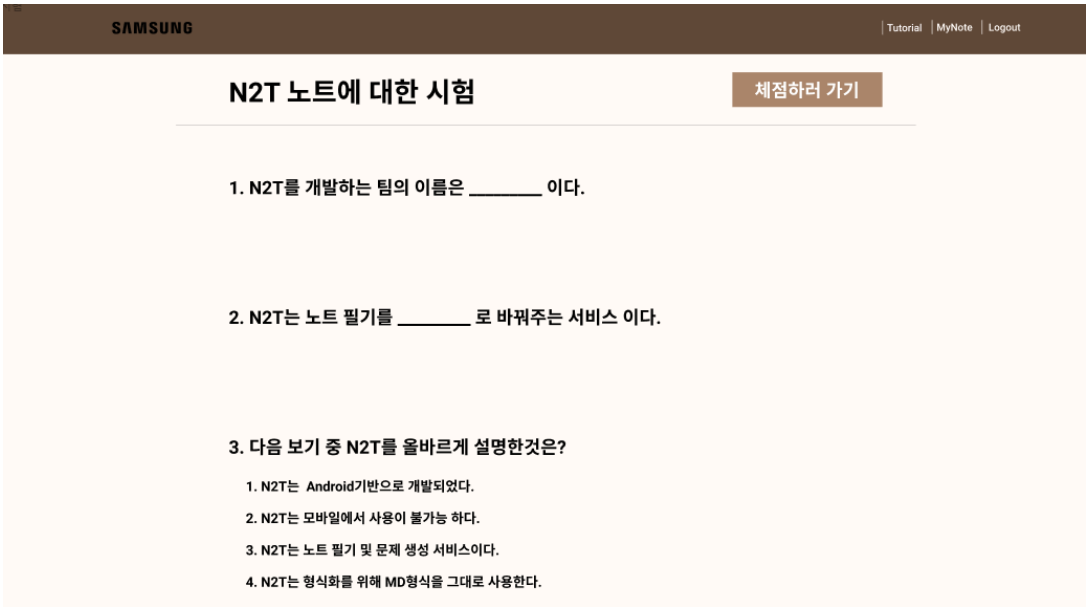

그림 31 테스트 페이지 디자인 시안

위 디자인 시안을 통해 다음과 같은 결과물을 구현하였다.

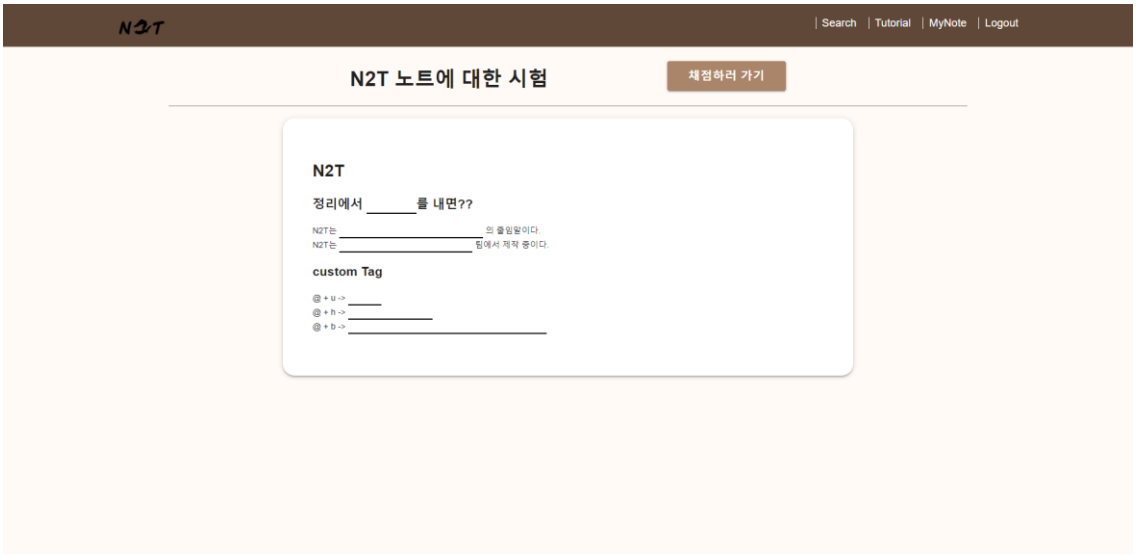

그림 32 테스트 페이지 구현 모습

위를 구현하기 위해서 우리가 HTML 태그로 변환한 부분을 다음과 같은 로직을 통해 빈칸으로 변환하는 작업을 진행하였다. 이를 통해 실제 정답 값을 같이 가져온다.

```
useEffect(() => {
    const testList = document.querySelectorAll('test');
    setClick(false);
    const tmp = [];
    const boundary = document.getElementById('boundary');
    const maxWidth = boundary.offsetWidth;

     testList.forEach((testTag) => {
      const answer = testTag.lastChild;
      const answerWidth = answer.offsetWidth;
      answer.style.display = 'none';
      const textBox = testTag.firstChild;
      tmp.push(answer.innerText);
      textBox.style.width = `${2 * answerWidth}px`;
      textBox.style.maxWidth = `${maxWidth * 0.7}px`;
    });
    setCorrectAnswer(() => tmp);
    setLoading('visible');
  }, []);
```

2.8.6 채점 구현

그림 33 채점 페이지 디자인 시안

위와 같은 디자인 시안을 바탕으로 다음과 같은 결과물을 완성하였다.

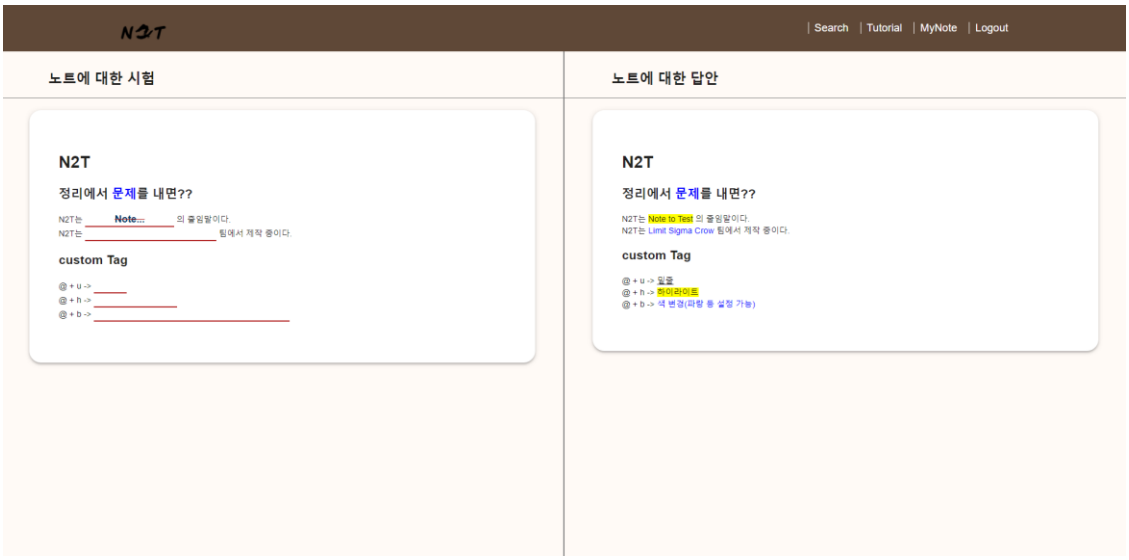

그림 34 채점 페이지 구현 모습

다음과 같이 정답을 표기하는 방식을 정답을 맞춘 경우에는 기존 노트와 동일하게 표현하였고 틀린 부분은 취소선을 통해 틀림을 확인하게 되었고 직관적으로 노트와 바로 비교 할 수 있게 하였다.

```
useEffect(() => {
    if (correctAnswer.length === 0) {
      alert('비정상적인 접근입니다.');
      history.goBack();
    }
    const testList = document.querySelectorAll('test');
    const boundary = document.getElementById('boundary');
    const maxWidth = boundary.offsetWidth;
    let count = 0;
    testList.forEach((testTag) => {
      const answerBox = testTag.lastChild;
      const textBox = testTag.firstChild;
      if (answer[count] === correctAnswer[count]) {
        textBox.style.display = 'none';
      } else {
        const answerWidth = answerBox.offsetWidth;
        textBox.value = answer[count];
        textBox.disabled = true;
        textBox.style.width = `${2 * answerWidth}px`;
        textBox.style.maxWidth = `${maxWidth * 0.7}px`;
        answerBox.style.display = 'none';
      }

       count += 1;
    });
    setLoading('visible');
  }, []);
```

다음과 같이 정답과 작성한 답을 비교하여 HTML로 표시해주는 방식이다.

2.8.7 내 노트 보기 구현

그림 35 내 노트 페이지 디자인 시안

위와 같은 디자인시안을 바탕으로 다음과 같은 결과물을 완성하였다.

그림 36 내 노트 페이지 구현 모습

사용자 기반으로 사용자가 작성한 노트 데이터를 가져오는 방식으로 구현하였으며 여기서 바로 수정을 진행할 수 있게 하였다.

```
async get({ userIdx }) {
    const query = {};
    query.where = { user_idx: userIdx };
    const result = await notesModel.findAll(query);
    return result;
  },
```

다음과 같은 부분으로 DB에 접근하여 가져왔으며 List로 있는 모든 노트를 보여준다. 그 외 사용자의 이름 정도를 보여준다.

2.8.8 검색 구현

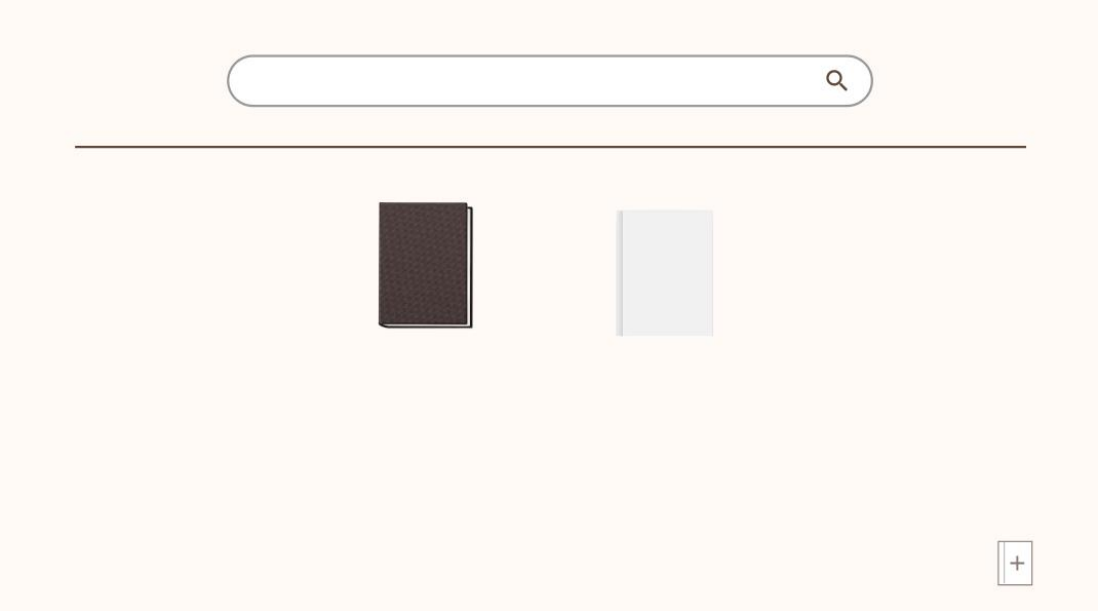

그림 37 검색 페이지 디자인 시안

위와 같은 디자인 시안을 통해 다음과 같은 결과물을 구현하였다.

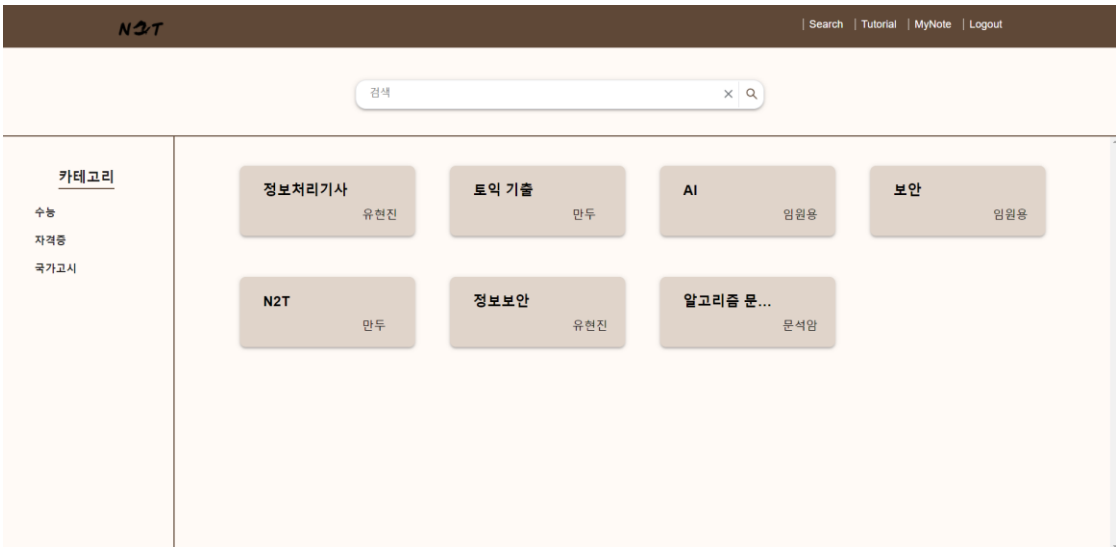

그림 38 검색 페이지 구현 모습

각각의 노트 항목을 가져오는 방식은 노트를 가져오는 방식과 매우 유사하다 하지만 검색에 대한 부분을 구현하기 위해 ORM 에서 query를 다음과 같이 하였다.

```
async getShearNote({ searchQuery }) {
   const query = {};
```

```
    query.where = {
      is_public: true,
      note_name: {
        [Op.like]: `%${searchQuery}%`,
      },
    };
    const result = await notesModel.findAll({
      where: {
          is_public: true,
        note_name: {
          [Op.like]: `%${searchQuery}%`,
        },
      },
      include: [
        {
          model: users,
          as: 'user',
        },
      ],
    });
    return result;
  },
```

위와 같이 %을 통해 앞뒤로 단어가 있는 경우에도 찾을 수 있게 하여 검색기능을 구현하였다.

2.8.9 Tutorial 구현

튜토리얼은 개발 중 디자인을 같이 하였으며 결과는 다음과 같다

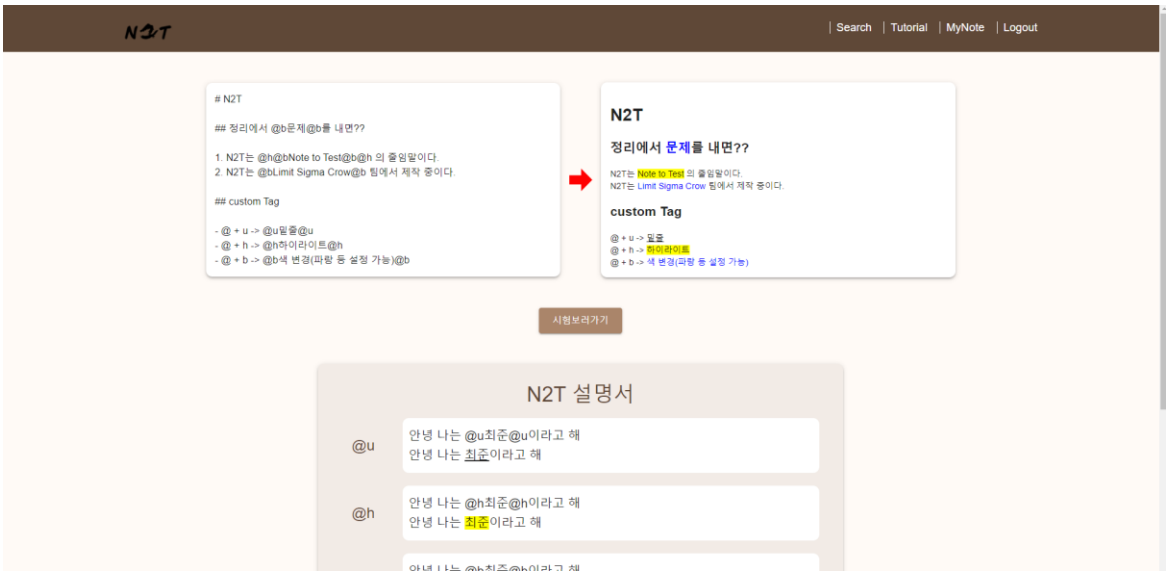

그림 39 튜토리얼 페이지 구현 모습

위 모든 방식을 사용하여 임시 데이터를 넘겨주고 이를 통해 Test와 체점까지 되는 것을 직

접 확인할 수 있다.

2.8.10 폴더 구현

폴더는 MyNotePage 와 동일한 위치에 존재하며 Note의 방향을 잡는 방식으로 구현하였다. 특징이 있는 코드는 없다.

3 목표 달성도 및 기여도

3.1 목표 달성도

번호	기능	목표 달성도	퍼센트(%)
1	로그인		80%
2	회원가입		80%
3	튜토리얼		100%
4	내 노트 보기		70%
5	노트 필기		60%
6	필기 에디터		70%
7	문제 제공		100%
8	채점하기		100%
9	노트 검색		50%

표 24 목표 달성도

제품 백로그의 우선순위에 따라 중요도를 두고 개발을 진행하였으나, 에디터 툴바 부분을 완벽하게 개발하지 못하였고, 노트 검색에 대한 요구사항도 기본적인 검색은 제공하지만 완전히 충족하지는 못하였다. 요구사항들의 부족한 부분은 추가 개발할 예정이다.

3.2 기여도

3.2.1 Custom MD를 통한 사용성 증진

노트 필기에 사용되는 형광펜, 밑줄, 글씨 색상 변경 등의 보편적인 강조 효과를 태그로 커스텀하여 사용자에게 제공함으로써 접근성이 낮았던 MD의 인식을 개선 및 사용성 증진한다. 또한 사용자의 편의성을 증신 시키기 위해서 앞으로 더욱 많은 커스텀 Tag를 추가할 예정이다.

이를 통해 사용성을 증가시키는 방향으로 발전할 계획이다.

3.2.2 맞춤형 문제를 통한 학습 능률 향상

일반적인 정보를 문제로 제공하는 기존 문제 제공 서비스와 달리 사용자가 선택한 정보를 단답형이나 서술형 문제로 제공하는 사용자 맞춤형 서비스를 제공함으로써 효율적인 선택적 학습을 도와준다.

3.2.3 다양한 분야에 대한 빅데이터 확보

형식화된 노트 필기법을 제공하여 얻은 라벨링 된 데이터를 통해 학습에 관한 빅데이터를 관련 분야에 활용할 수 있다. 이를 통해 관련 연구를 진행하는 사람들에게 데이터를 제공하거나 또는 효율적인 필기방식에 관한연구 등 다양한 분야에서 활용가능 하며 교육 쪽에서도 공유되는 노트로서 기능을 활용 할 수 있을 것이라 생각한다.

4 과제 결과의 활용 계획 및 향후 보완점

4.1 향후 보완 계획

4.1.1 디자인 구조 보완

다소 통일되지 못한 각 페이지들의 디자인을 웹 디자인 툴을 사용하여 통일성 있게 수정한다.

4.1.2 Refactoring

통일되지 못한 레이아웃, 컴포넌트 구조 수정 및 컴포넌트 재사용성 높이고, Mock-Up에서 정한 UI 디자인 요소를 Global theme로 정리하여 사용한다. 변수 이름 및 함수 이름을 더 직관적으로 수정한다.(CamelCase)

4.1.3 도메인 구매 및 배포

서비스 제공을 위해 배포를 진행할 예정이며 이에 도메인을 구매하여 우리의 홈페이지를 추가할 예정이다. 배포는 CI 툴을 활용하여 앞으로의 진행사항을 점진적으로 진행할 예정이다.

4.1.4 UX 보완

사용자 경험에 대한 설문을 배포하여 결과로 얻은 의견을 제품에 반영한다.

4.1.5 활용 방안

사용자의 노트 데이터를 수집하여 해당 분야의 여러 교육기관들과 협력하여 수익을 창출한다. 또한 여러 노트 템플릿을 작성하여 각 분야에 적합한 노트 정리를 할 수 있도록 돕는다.

5 과제 수행 과정에서 수집한 기술 정보

5.1 Custom MD

markdown-it이라는 Markdown parser을 사용하여 Custom MD 진행. 데이터는 rule들의 nested chain들을 통해 파싱되며, core, block, inline china 등이 존재함. parsing 한 결과들은 renderer로 넘겨져 사용될 token들이 된다. parsing 과정은 다음과 같다.

1. 전체적인 입력 값을 변환하기 전에 다듬는 core_parse 과정.

2. 입력 값을 개행이나 들여쓰기 등으로 나눠 토큰화하는 block_parse 과정.

3. 토큰화된 단락들을 토큰화하는 inline_parse 과정을 수행.

토큰을 가지고 render_rule에 맞게 html 형태로 변환되며 사용자는 rule들의 nested chain에 새로운 규칙을 넣어준 뒤 해당 rule에 대한 render_rule을 추가하여 Custom MD를 설정할 수 있다.

5.2 React

그림 40 React 로고

React는 자바스크립트 라이브러리의 하나로서 페이스북과 개별 개발자 및 기업들 공동체에 의해 유지보수되고 있는 웹 프레임워크이다. 리액트는 싱글 페이지 어플리케이션(SPA)나 모바일 어플리케이션 개발에 사용된다.리액트는 메모리 데이터 구조 캐시를 만들고 결과 차이를 계산하여 다음 브라우저에 표시되는 DOM을 효과적으로 업데이트한다.

Hook은 UI의 상태 관련 동작 및 side effects을 캡슐화하는 가장 간단한 방법으로 React에 처음 도입되었다.Hook은 클래스와 고차 컴포넌트의 복잡성을 피할 수 있게 해주고 계층의 변화 없이 상태 관련 로직을 재사용할 수 있도록 한다. 또한 class없이 React의 기능들을 사용하는 방법을 제시한다. 기본적인 React의 클래스 컴포넌트 사용법을 익힌 후 함수형 컴포넌트와 Hook을 사용하여 프로젝트를 진행하였다.

5.3 Express

그림 41 Express

Express는 Node.js의 핵심 모듈인 http와 Connect 컴포넌트를 기반으로 하는 웹 프레임워크다.보통의 작동 방식은 메인 파일을 통해 진입하여 다음과 같은 단계를 밟아 실행된다.

1. 컨트롤러, 유틸리티, 도우미, 모델과 같은 자체적인 모듈을 비롯한 서드파티 의존 모듈을 인클루드.
2. 템플릿 엔진과 해당 템플릿 엔진의 파일 확장자와 같은 Express.js 앱 설정 구성.
3. 오류 핸들러, 정적 파일 폴더, 쿠키 및 기타 파서와 같은 미들웨어 정의.
4. 라우팅 정의.
5. 데이터베이스 연결
6. 앱 구동

5.4 Sequlize

그림 42 Sequelize 로고

Sequlize란 Node.js에서 MySQLI을 사용할 때 raw Query문을 사용하지 않고 더욱 쉽게 다룰 수 있도록 도와주는 라이브러리이다. Sequelize는 Object-Relational Mapping(ORM)으로 분류되며 객체와 관계형 데이터베이스의 관계를 매핑 해주는 도구다. Sequelize를 사용하여 raw Query문을 사용하지 않고 Javascript를 이용해 MySQL을 사용할 수 있다.

5.5 GitHub

그림 43 GitHub 로고

GitHub에서 제공하는 많은 기능들을 사용하였다.사용한 기능에 대한 설명은 아래와 같다.

Fork : 다른 사람의 저장소를 복사하는 기능, 개인 사용자들은 서로의 원격 저장소를 읽고 쓸 수 있는 기능.

Issue : 저장소 안에서 사용자들 사이의 문제를 정의하고 논의하는 기능.

Pull Request : Fork한 저장소를 수정해 다시 원본 저장소에 병합해달라는 요청을 보내 사용자 사이의 상호작용을 일으키게 하는 기능.

Wiki : 저장소와 관련된 체계적인 기록을 남기는 기능.

협업을 하기 위해 각자 개인 저장소에 프로젝트를 Fork하고 Issues를 만들었다. Issues를 기반으로 개발해 Pull requests를 진행했다. 또한 Wiki를 적극적으로 활용하여 문서관리를 진행했

다. Wiki에는 팀의 Ground Rules부터 요구사항 명세서, 유스케이스, 제품 백로그, API 문서, 회의록 등을 기록하고 관리하였다.

Git-flow 방법을 사용하여 프로젝트를 진행했다. Git-flow는 총 5가지의 브랜치를 사용해서 프로젝트를 운영한다. 각 브런치는 아래와 같다.

master : 기준이 되는 브랜치로 제품을 배포하는 브랜치.

develop : 개발 브랜치로 개발자들이 이 브랜치를 기준으로 각자 작업한 기능들을 합친다.

feature : 단위 기능을 개발하는 브랜치로 기능 개발이 완료되면 develop 브랜치에 합친다.

release : 배포를 위해 master 브랜치로 보내기 전에 먼저 QA(품질검사)를 하기위한 브랜치.

hotfix : master 브랜치로 배포를 했는데 버그가 생겼을 때 긴급 수정하는 브랜치.

master와 develop가 중요한 메인 브랜치이고 나머지는 필요에 의해 운영하는 브랜치이다. 해당 프로젝트를 진행하면서 feature 브랜치를 많이 사용했다.

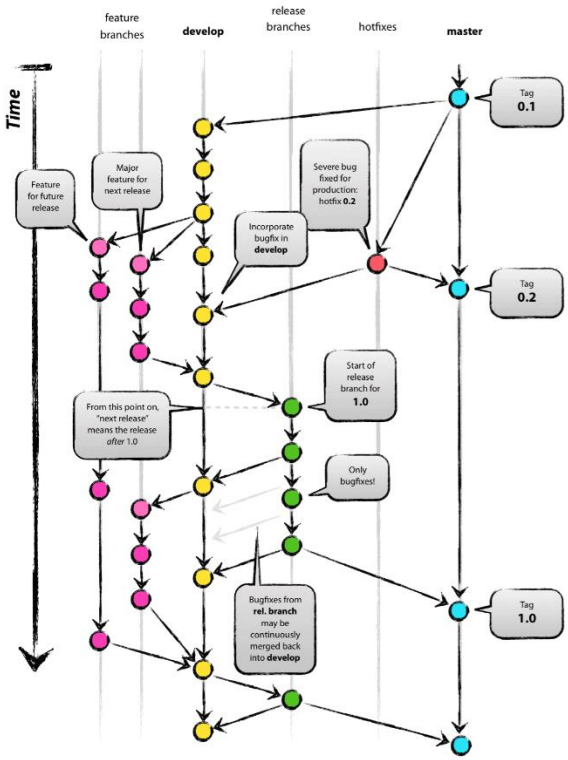

그림 44 Git-flow 예시

6 과제수행 시 어려운 기술(지식) 문제 해결

6.1 Custom MD Rule 생성에 대한 문제 해결

문서에 Rule에 관한 설명이 없으며 또한 사용자들이 이 부분을 변경하여 사용하려고 하지 않기 때문에 직접 라이브러리를 전부 분석해야 하는 어려움이 있었다. 결과적으로 해결 하였지만 꽤나 많은 시간을 소비하였다.

6.2 FE Buffer 사용 문제 해결

React 컴포넌트 안에서 서버에서 받은 Blob 데이터를 String으로 변환하려고 할 때 문제가 발생했다. Buffer나 메모리, 데이터 통신, 파일 시스템 모듈은 Node.js가 있어야만 실행이 가능하며 웹 브라우저에서는 Node.js가 내장되어 있지 않아 사용할 수 없다. 웹 브라우저가 Node.js의 기능을 사용하기 위해서는 서버에서 처리를 하여 전달받아야 하는 부분을 알게 되었다.

6.3 List Component 배열에 대한 문제 해결

노트 검색 결과 및 내 노트 페이지에서 해당 노트 컴포넌트들을 리스트로 출력하면 결과가 한 줄로만 나오는 문제가 발생하였다. 따라서 결과를 출력하는 컴포넌트에 flex 속성을 부여하여 노트 컴포넌트들 간의 상하 여백과 좌우 여백을 동일하게 주기 위해 space-around를 사용하였는데, 이는 width를 기반으로 하기 때문에 열이 맞지 않는 문제가 발생하였다. flex 대신 grid 속성을 부여해 한 행이 모두 차면 다음 행의 첫번째 열부터 채우는 grid-auto-flow: row를 사용하는 것으로 문제를 해결하였다.

7 참고문헌 및 자료 출처

7.1 이다을, "시험 효과에 대한 뇌과학적 이해와 교육적 시사점", 국내석사학위논문 서울교육대학교 교육 전문대학원, 2016, 서울

7.2 Butler, Andrew C., and Henry L. Roediger III. "Testing improves long-term retention in a simulated classroom setting." *European Journal of Cognitive Psychology* 19.4-5 (2007): 514-527

7.3 Test format andcorrective feedback modify the effect of testing on long-termretention

7.4 The testing effect in free recall isassociated with enhanced organizational processes.

7.5 Baars, Bernard J., and Nicole M. Gage. Cognition, brain, and consciousness: Introduction to cognitive neuroscience . Academic Press, 2010.