
오픈소스 소프트웨어 프로젝트 중간보고

2023.05.17.

동국대학교  3팀 MAC
dongguk university

2017112288 민한결

2019112567 안상연

2019112453 최필환

<목 차>

제 1 장 프로젝트 개요

1. 프로젝트명	1
2. 프로젝트 목적	1
3. 기대효과	1
4. 프로젝트 라이선스	1

제 2 장 요구사항 분석

1. 요구사항 수집	2
2. 요구사항 명세서	6

제 3 장 프로젝트 설계

1. 오픈소스 선정 및 소개	7
2. 사용자 인터페이스 설계	11
3. DB 설계	13

제 4 장 프로젝트 수행 계획

1. 업무분장	14
2. 개발 환경	14
3. 협업 규칙	15
4. 개발 일정	16

제 5 장 프로젝트 중간 보고

1. 진행 사항	17
2. 변동 사항	37

제 1 장 프로젝트 개요

1. 프로젝트명

동국대학교 LMS 내 협업 파일시스템 구축

2. 프로젝트 목적

동국대학교 학습관리시스템(LMS)인 ‘동국대학교 E-class’ 상에 팀 프로젝트를 위한 파일 스토리지 시스템을 구축하여, 팀 프로젝트 과정에서 생성된 파일의 이력 관리가 가능하게 한다.



[그림 1] 동국대학교 E-class UI

3. 기대효과

기존 카카오톡이나 클라우드 스토리지를 사용하여 팀 프로젝트 파일을 공유 시, 작성자, 수정 내용 파악이 어렵다. 또한 이전 버전으로 복구를 원할 시 복구하고자 하는 파일을 찾기 어렵고 기존 파일에 덮어썼을 때 복구가 불가능하다. 파일 스토리지 시스템을 구축하여 업로드일, 업로드자, 수정 메시지를 DB에 저장하면 이러한 문제점들을 해결하고 협업 능력을 올릴 수 있을 것이다.

4. 프로젝트 라이선스

오픈소스 프로젝트의 라이선스는 프로젝트에 사용된 컴포넌트 및 라이브러리의 라이선스에 따라 결정된다. 개발 환경과 사용되는 오픈소스의 라이선스를 분석하여 GPL v2 라이선스를 선정하였다. GPL v2는 오픈소스를 사용, 수정, 배포할 수 있으나, 수정된 소스 코드에도 동일한 GPL v2 라이선스를 적용해야 한다. 또한 소프트웨어를 상업적으로 이용하는 경우, 소스 코드를 공개해야 한다.

제 2 장 요구사항 분석

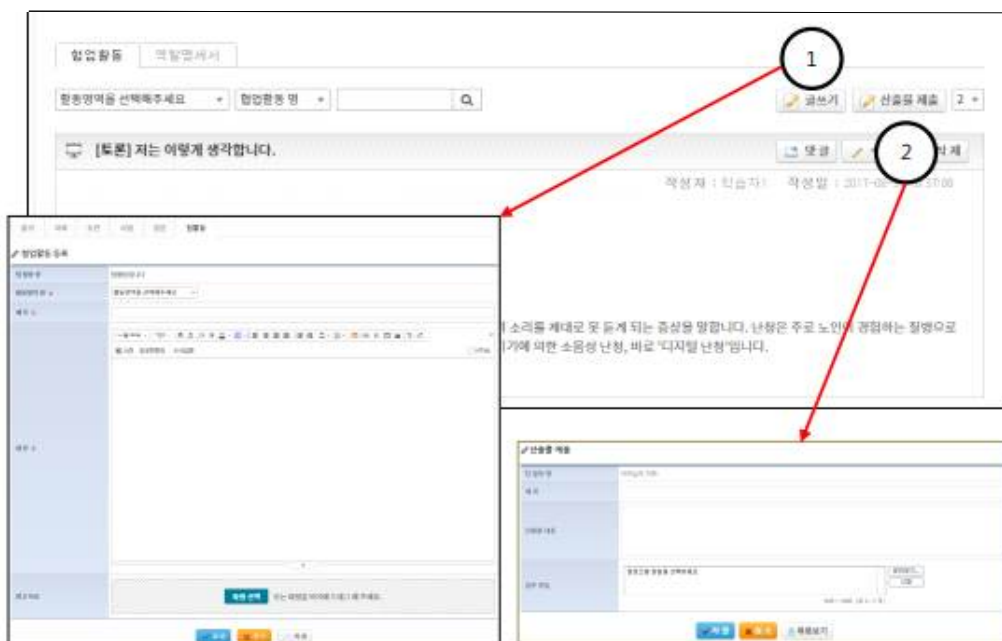
소프트웨어 기능과 제약조건의 명확화를 위해 현행 시스템 분석과 기존 사례 분석을 통해 요구사항을 수집하였다. 이를 통해 통해 파일 스토리지 시스템, 팀 구성 기능, 팀 내 공지사항 기능이 필요함을 파악하고 그에 따른 요구사항 명세서를 작성하였다.

1. 요구사항 수집

개발할 소프트웨어의 기능과 제약조건 등을 명확히 하기 위해 현행 시스템 분석과 기존 사례 분석을 통한 요구사항 수집을 시행하였다.

1.1 현행 시스템 분석

동국대 이클래스 상에 기존 팀 활동 영역을 분석하여 기능과 개선점을 파악하였다. 기존 팀 활동 영역은 [그림 2]와 같다. 기존 팀 활동 영역의 기능은 게시판 구조에서 글을 쓰고 댓글을 달거나, 파일을 업로드하는 기능과 팀장이 대표로 과제를 제출하는 기능으로 구성되어 있다. 또한 팀 구성을 교수 및 조교 권한으로만 가능하다는 제약이 있다. 협업에 필요한 파일시스템 기능과 학생들이 자율적으로 팀을 구성하는 기능을 도입하면 팀 활동 영역의 활용도를 높일 수 있을 것이다.



[그림 2] 동국대 E-class 팀 활동 영역 UI

1.2 기존 사례 분석

기존 팀 프로젝트에 진행에 주로 활용되는 플랫폼을 분석하여 개선점과 참고할 점을 정리하였다.

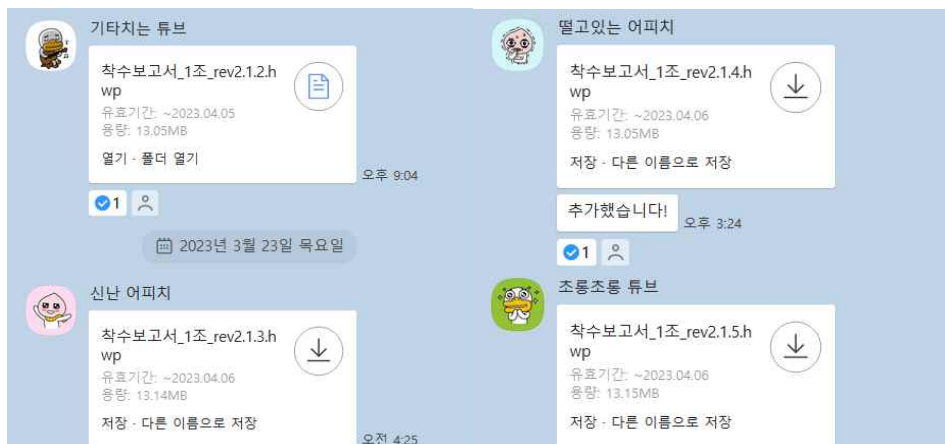
가. 카카오톡

- 기능

사람들과 메시지를 주고받을 수 있는 모바일 메신저 앱으로 기본 대화 기능과 더불어 파일, 영상, 사진 등을 첨부하는 기능, 보이스톡이라 불리는 전화 기능, 송금 기능 등 여러 가지 다양한 기능을 제공한다.

- 사용사례

한국에서는 메신저 앱 중 이용률이 압도적이다. 팀원 대부분이 이미 해당 서비스를 이용 중이고 익숙하다는 점에서 협업 시에도 카카오톡을 이용하는 경우가 많다. 하지만 협업을 위한 용도로 나온 서비스가 아니므로 타 협업툴에 비해 프로젝트 진행 과정이나 팀원들 간 업무 파악이 어려운 편이다.



[그림 3] 카카오톡 파일 공유 기능 사용 예시

- 개선점

카카오톡을 협업툴로 보았을 때 협업에서 중요한, 프로젝트 진행 상황을 정리해 보여주는 기능이나 자료(파일) 정리 기능이 없다는 단점이 있다. 또한 팀 구성 및 정보 공유에 팀원의 개인정보를 제공해야 하는 단점이 있다.

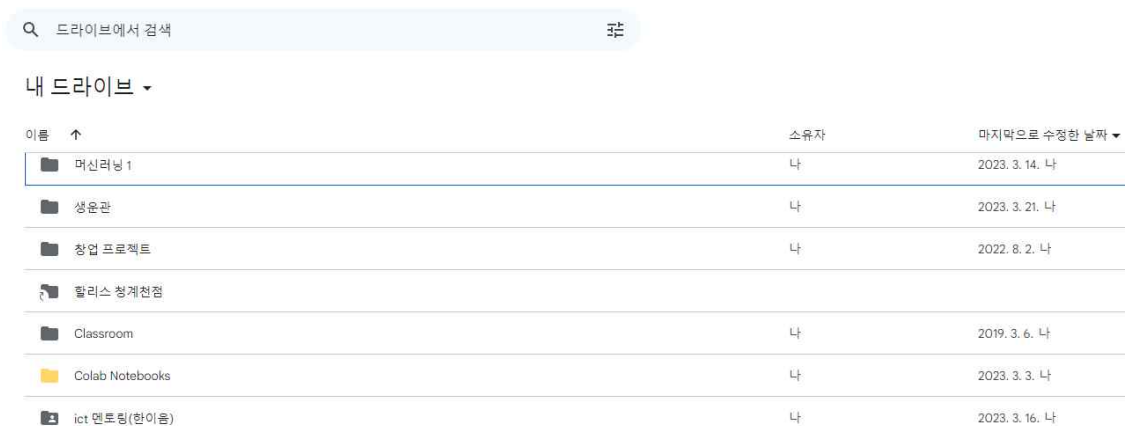
나. 구글 드라이브

- 기능

구글에서 제공하는 클라우드 기반 협업 도구이자 파일 저장/공유 서비스이다. 구글 드라이브는 구글 독스 에디터 오피스 제품군인 구글 문서 도구, 구글 시트, 구글 슬라이드와 연동되므로 문서, 스프레드시트, 프레젠테이션, 드로잉, 폼 등의 공동 편집이 가능하다.

- 사용사례

구글 드라이브에 파일 및 폴더를 업로드한 화면이다. 구글 드라이브 상에서 공동 편집이 가능하므로 수정한 날짜와 수정자가 명시된다. 또한, 로컬과 연결하여 동기화를 시킬 수도 있다.



이름	소유자	마지막으로 수정한 날짜
머신러닝 1	나	2023. 3. 14. 나
생윤관	나	2023. 3. 21. 나
창업 프로젝트	나	2022. 8. 2. 나
칼리스 청계천점		
Classroom	나	2019. 3. 6. 나
Colab Notebooks	나	2023. 3. 3. 나
ict 멘토링(한이름)	나	2023. 3. 16. 나

[그림 4] 구글 드라이브 사용 예시

- 개선점

구글 드라이브의 경우 이전 버전에 대한 관리는 30일간만 이루어진다. 학기 특성상 3~4달의 기간 동안 파일을 생성하고, 복구할 일이 많은 것을 고려하여 더 장기적인 버전관리가 지원되어야 한다. 또한, 공동 편집을 위해서는 구성원들의 이메일을 알아야 팀 구성이 가능하다. 이처럼 불필요하게 개인정보를 제공해야 한다.

다. Git&Github

- 기능

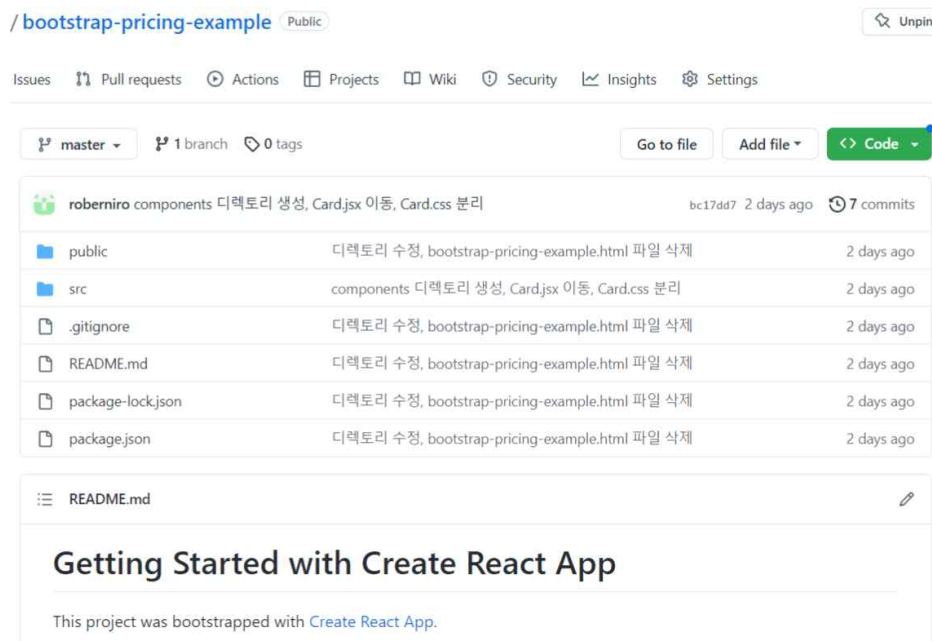
Git은 소스코드 분산 버전관리 시스템이며 Github는 Git 원격 저장소를 호스팅하는 웹 서비스이다. Git에서 사용하는 주요 명령어와 그 기능은 [표 1]과 같다.

[표 1] Git 명령어 및 기능

명령어	기능
add	이력 관리 대상 파일을 스테이징함
commit	스테이징한 파일을 로컬 리포지토리에 저장함
push	커밋한 내용을 원격 리포지토리에 업로드함
pull	원격 리포지토리에 저장한 내용을 로컬로 가져와 병합함
fetch	로컬 저장소에 원격 저장소의 변경사항을 확인해 가져옴
merge	분기된 커밋을 병합함

- 사용사례

비주얼 스튜디오 코드로 작성한 React 프로젝트를 Github 원격 리포지토리에 업로드한 화면 예시는 [그림 5]와 같다. 커밋 메시지를 통해 변경 내용을 명시하고 업로드한 시간이 기록되어 소스코드 이력 관리가 편리하다.



[그림 5] 깃허브 원격 저장소 사용 예시

– 개선점

기능이 소스코드 관리에 특화되어 있고, push 전 로컬 저장소에 커밋시 CLI에서 명령어를 입력하거나 별도의 IDE나 'Sourcetree'를 통해서만 GUI로 사용할 수 있는 점을 개선할 필요가 있다.

– 참고할 점

커밋시 커밋 메시지를 남길 수 있고, 해시를 커밋 id로 하여 자식 커밋은 부모 커밋의 해시 값을 저장한다. 이를 통해 수정 내용을 파악하기 쉽고, 파일의 변경 이력을 찾기 쉽게 한다.

1.3 결론

현행 시스템 분석과 사례 분석을 통해 수집한 요구사항은 [표 2]와 같다.

[표 2] 요구사항 수집 결과

분류	대상	문제점	요구사항
현행 시스템	E-class	파일 이력 관리▼	파일 스토리지 기능
		팀 구성 자율성▼	팀 구성 기능
기존 사례	카카오톡	파일 이력 관리▼	파일 스토리지 기능
		프라이버시 보호▼	팀 구성/공지사항 기능
	구글 드라이브	파일 이력 보존 기간▼	파일 스토리지 기능
	깃&깃허브	비전공자 접근성▼	파일 스토리지 기능

2. 요구사항 명세서

현행시스템 분석과 기존 사례 분석을 통해 파일 스토리지 시스템, 팀 구성 기능, 팀 내 공지사항 기능이 필요함을 파악하였다. 신규 시스템의 요구사항을 정리하여 [표 3]로 명세하였다.

[표 3] 요구사항 명세서

사용자	메뉴	필요 기능	기능 설명
학생	팀 활동	팀 구성	수강생 조회하여 팀 프로젝트 초대 발송한다.
		업로드	파일과 관련 정보를 업로드할 수 있다.
		다운로드	파일을 다운로드할 수 있다.
		삭제	특정 파일 및 전체 파일 삭제 기능이다.
		버전관리	이전 버전 조회 및 다운로드 가능하다.
		공지사항	팀 내의 공지사항을 등록하고 조회한다.
	메인 페이지	로그인/로그아웃	학번, 비밀번호로 접속 가능하게 한다.

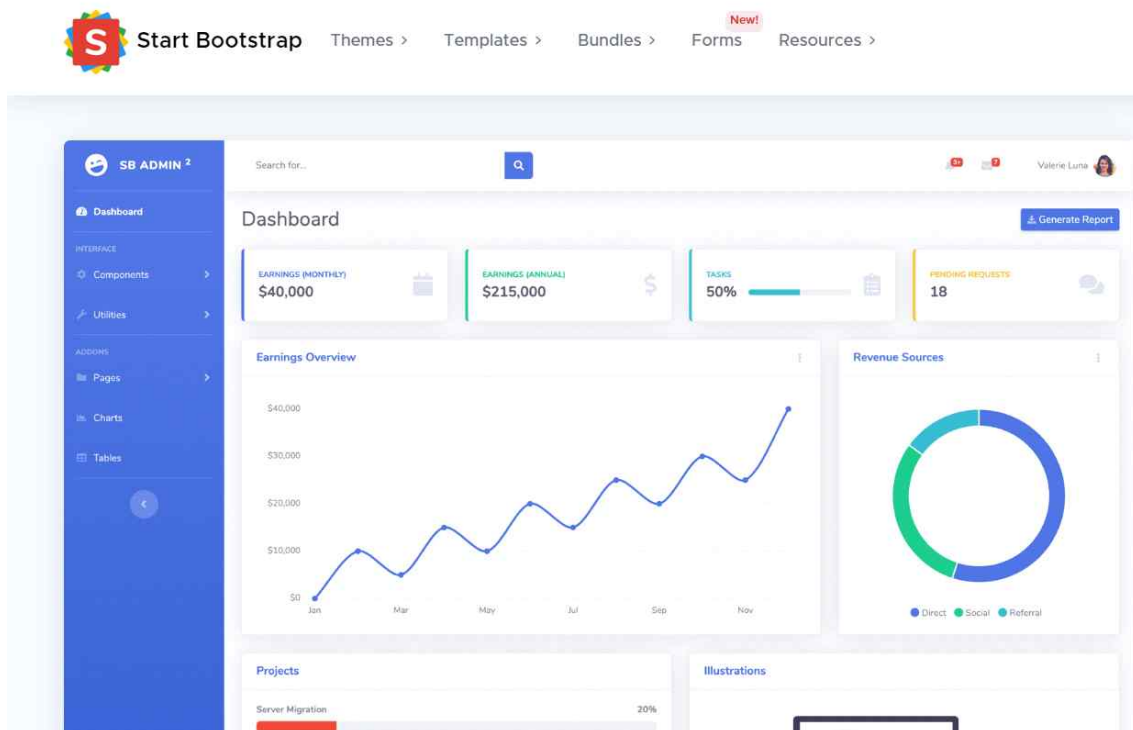
제 3 장 프로젝트 설계

앞서 분석한 요구사항을 기반으로 팀 활동 페이지 내 파일 관리 시스템 구축에 필요한 오픈소스 선정, 사용자 인터페이스 설계, DB설계를 진행하였다.

1. 오픈소스 선정 및 소개

1.1 프론트엔드

- Github: <https://github.com/startbootstrap/startbootstrap-sb-admin-2>
- 오픈소스 소개: 부트스트랩 템플릿을 무료로 제공하는 Start Bootstrap 사이트를 이용하여 관리자 대시보드 테마의 템플릿 오픈소스를 사용할 예정이다.



[그림 6] 프론트엔드 템플릿 예시

- 활용방안: 로그인 페이지는 웰컴 페이지 구현에 활용 가능하다. 파일 목록, 부가 서비스 제공을 위한 섹션 등은 팀 활동 페이지 구현에 활용할 수 있다.
- 라이선스 : MIT License

1.2 백엔드

가. 업로드

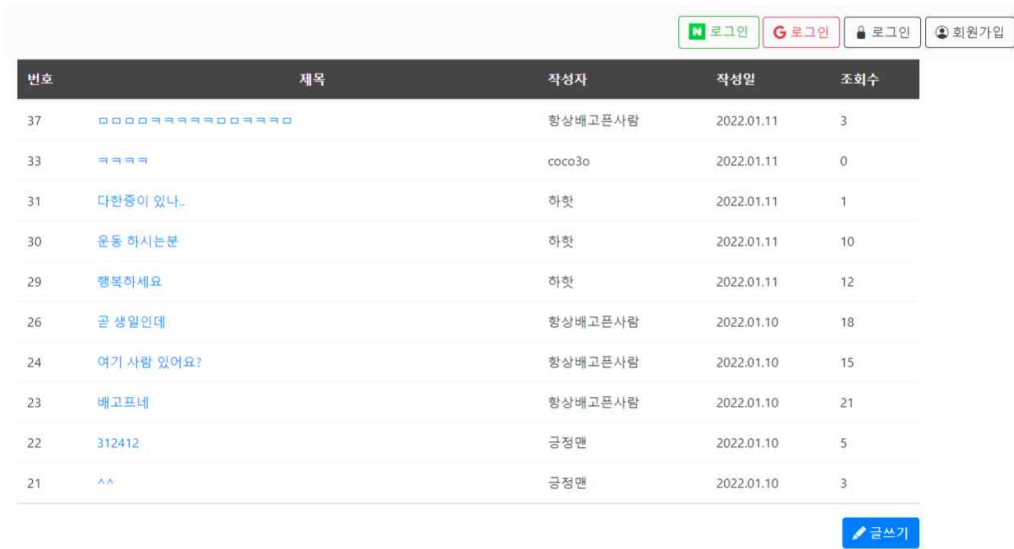
- 깃허브: <https://github.com/seungh0/spring-cloud-aws-s3>
- 오픈소스 소개: 본 오픈소스는 AWS s3를 이용하여 파일 업로드를 구현했다.
- 활용방안: 현재 오픈소스는 파일 업로드에 대한 부분만 구현이 되어 있다. 따라서 본 코드를 참고하여 폴더 및 파일의 업로드도 구현하려고 한다. 파일시스템 특성에 맞추어 다운로드, 삭제도 구현한다. 기존 이클래스와 다르게 버전관리가 가능한 이클래스를 만들 것이기 때문에, s3에서 이전 버전에 대한 파일을 가져올 로직도 구현한다.
- 라이선스: 프로젝트 라이선스가 명시되어 있지 않으나 사용된 라이선스는 [표 4]와 같다. 이를 바탕으로 해당 오픈소스의 라이선스는 GPL v2 라이선스로 판단할 수 있다.

[표 4] 파일 업로드 오픈소스 사용 라이선스

분야	SW	라이선스
언어	JAVA(Corretto JDK11)	GPL v2 라이선스
프레임워크	Spring Boot	아파치 라이선스 2.0
빌드도구	Gradle	아파치 라이선스 2.0

나. 로그인

- 깃허브: <https://github.com/hojunnnnnn/board>
- 오픈소스 소개: Spring Boot로 구현한 게시판 프로젝트로 Security 회원가입 및 로그인, OAuth 2.0 구글, 네이버 로그인, 회원 정보 수정, 회원가입 시 유효성 검사 및 중복 검사 기능, 게시판 CRUD 기능, 댓글 CRUD 기능이 있다.



번호	제목	작성자	작성일	조회수
37	□□□□□□□□□□□□□□□□	항상배고픈사람	2022.01.11	3
33	ㅋㅋㅋㅋ	coco3o	2022.01.11	0
31	다한증이 있냐..	하핫	2022.01.11	1
30	운동 하시는분	하핫	2022.01.11	10
29	행복하세요	하핫	2022.01.11	12
26	곧 생일인데	항상배고픈사람	2022.01.10	18
24	여기 사람 있어요?	항상배고픈사람	2022.01.10	15
23	배고프네	항상배고픈사람	2022.01.10	21
22	312412	금정맨	2022.01.10	5
21	^^	금정맨	2022.01.10	3

[그림 7] 게시판 오픈소스 UI

- 활용 방안: 메인 페이지의 로그인/로그아웃 기능을 구현하는 데 Security 회원가입 및 로그인 기능을 활용할 수 있다.
- 라이선스: 프로젝트 라이선스가 명시되어 있지 않으나 사용된 라이선스는 [표 5]와 같다. 이를 바탕으로 해당 오픈소스의 라이선스는 GPL v2 라이선스로 판단할 수 있다.

[표 5] 게시판 오픈소스 사용 라이선스

분야	SW	라이선스
언어	JAVA(Corretto JDK11)	GPL v2 라이선스
프레임워크	Spring Boot	아파치 라이선스 2.0
DB	MySQL	GPL v2 라이선스
빌드도구	Gradle	아파치 라이선스 2.0

다. 공지사항

- 깃허브: <https://github.com/rgl-za/ToDo>
- 오픈소스 소개: Bootstrap기반 프론트엔드 템플릿에 Spring Boot를 이용해 백엔드를 구현한 Todo-List로 할 일에 대한 CRUD가 가능하다.

Todo App

No.	Todo item	status	Actions
0	안녕하세요	N	<div style="display: inline-block; background-color: #dc3545; color: white; padding: 5px 10px; margin-right: 5px;">Delete</div> <div style="display: inline-block; background-color: #28a745; color: white; padding: 5px 10px;">Finished</div>
1	안녕	N	<div style="display: inline-block; background-color: #dc3545; color: white; padding: 5px 10px; margin-right: 5px;">Delete</div> <div style="display: inline-block; background-color: #28a745; color: white; padding: 5px 10px;">Finished</div>

[그림 8] Todo-List 오픈소스 UI

- 활용 방안: 팀 활동 페이지의 프로젝트 공지사항 기능 구현에 활용할 수 있다.
- 라이선스: 프로젝트 라이선스가 명시되어 있지 않으나 사용된 라이선스는 [표 6]와 같다. 이를 바탕으로 해당 오픈소스의 라이선스는 GPL v2 라이선스로 판단할 수 있다.

[표 6] Todo-List 사용 라이선스

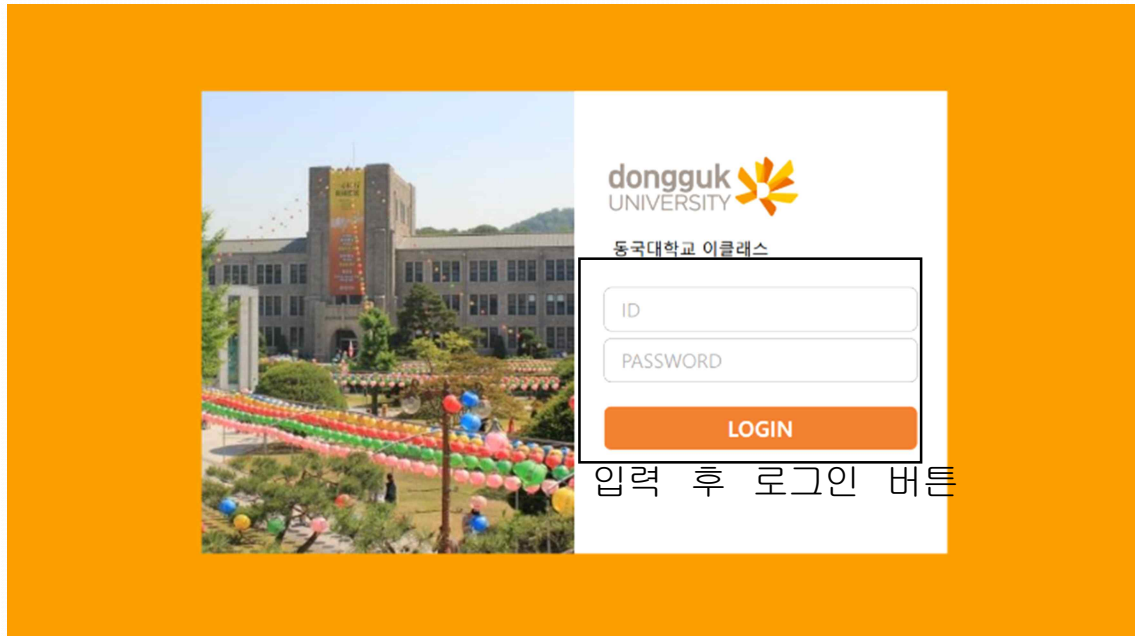
분야	SW	라이선스
언어	JAVA(Corretto JDK11)	GPL v2 라이선스
프레임워크	Spring Boot	아파치 라이선스 2.0
DB	MySQL	GPL v2 라이선스
빌드도구	Gradle	아파치 라이선스 2.0

2. 사용자 인터페이스 설계

앞서 정의한 요구사항을 기반으로 사용자 인터페이스를 설계하였다.

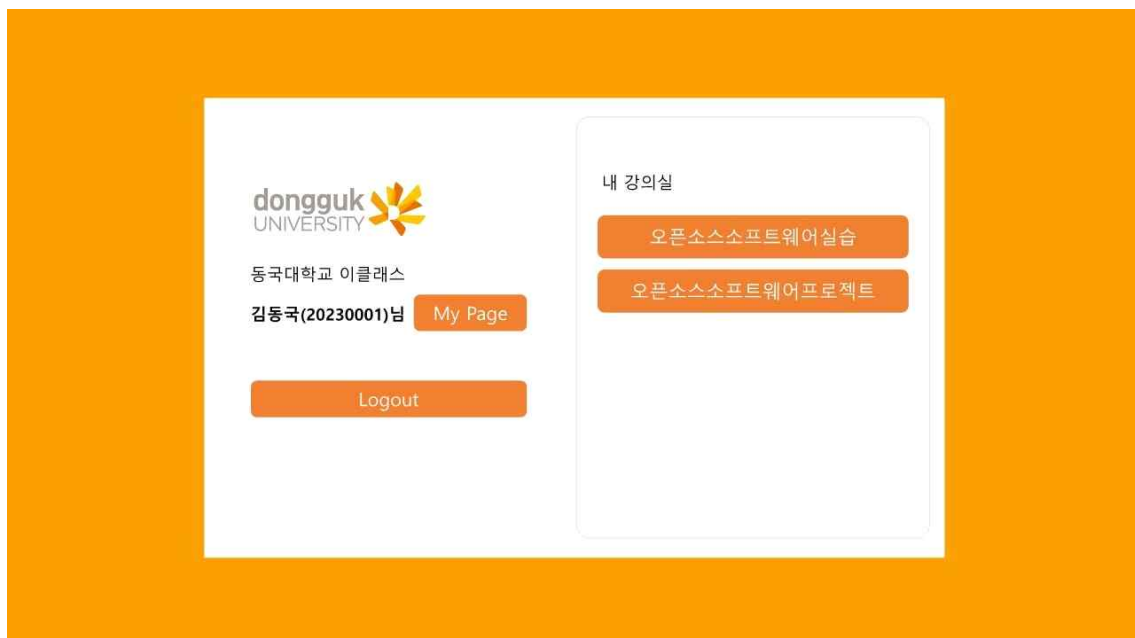
2.1 웰컴 페이지

웰컴 페이지로 로그인 화면을 구상하였다.



2.2 과목 선택 페이지

로그인한 사용자가 과목을 선택할 수 있다.



제 3 장 프로젝트 설계

2.3 팀 활동 페이지

파일 저장, 다운로드, 버전관리 버튼이 있는 파일 스토리지가 있다. 추가로 팀 구성 정보 확인 및 팀원 초대가 가능한 섹션과 공지사항을 등록하고 확인할 수 있는 섹션이 있다.



3. DB 설계

요구사항 명세와 사용자 인터페이스 설계를 바탕으로 필요한 테이블과 속성을 정의해보았다. 속성과 테이블 매핑 관계 관점에서 설명하겠다.

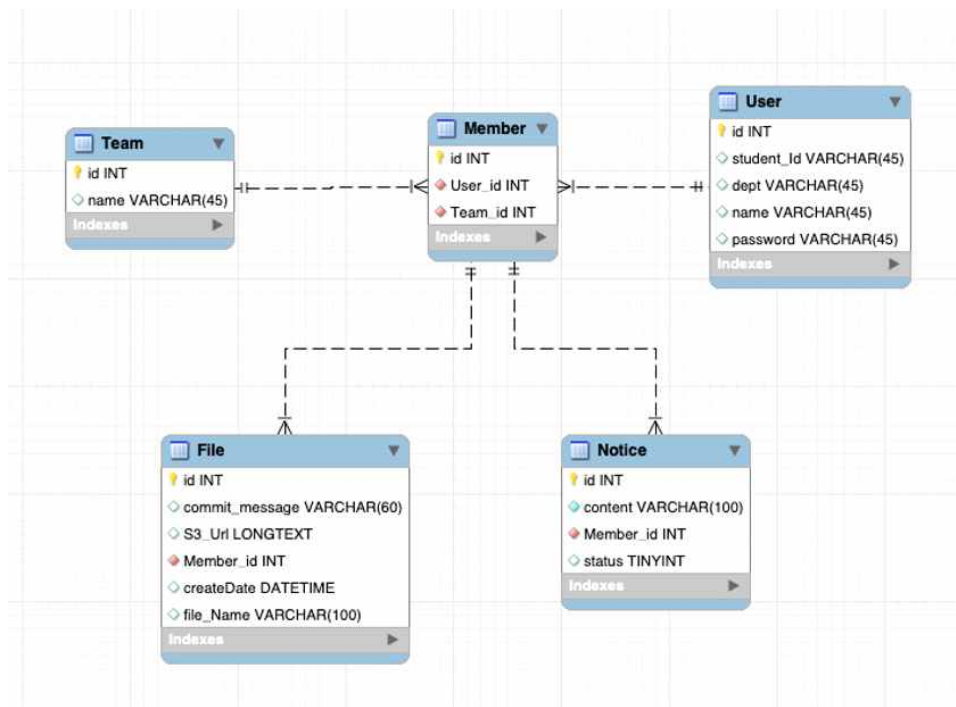
3.1 속성

- User: 로그인에 필요한 id, password를 속성으로 가지고 있고, 학생 정보인 dept(학과)와 name도 가지고 있다.
- Team: 팀 구성에 필요한 릴레이션으로 name(팀이름)을 가지고 있다.
- Member: User와 Team의 중간 매핑 테이블로 각각의 테이블인 User와 Team의 FK를 가지고 있다.
- File: 버전관리를 위한 S3_URL과 commit_message가 있다. 외래키로 Member_id를 가지고 있다.
- Notice: 공지사항으로 수행 여부를 판단하는 status가 있다.

3.2 테이블 매핑 관계

- User, Team은 다대다 관계이므로, 중간테이블은 Member를 통해 일대다로 만들어준다.
- File과 Notice는 중간테이블인 Member 테이블과 다대일 관계이다.

3.3 ER Diagram



제 4 장 프로젝트 수행 계획

프로젝트 수행을 위한 업무분장과 협업 규칙을 작성하고, 개발 일정을 수립하였다.

1. 업무분장

이름	역할
민한결	로그인, 팀 구성, 공지 사항, 버전관리 기능 구현
안상연	웰컴 페이지, 팀 활동 페이지 화면 구성
최필환	파일 업로드, 다운로드, 삭제 구현

2. 개발 환경

개발 환경을 SW/HW로 나누어 [표 7], [표 8]로 정리하였다.

[표 7] SW 환경

분야	세부 분야	SW	라이선스
운영체제		Ubuntu 22.04.1 LTS	GPL
백엔드	언어	JAVA(Corretto JDK11)	GPL v2 라이선스
	프레임워크	Spring Boot 2.7.10	아파치 라이선스 2.0
	IDE	IntelliJ	아파치 라이선스 2.0
	DB	Maria DB	GPL v2 라이선스
	빌드도구	Gradle	아파치 라이선스 2.0
프론트엔드	언어	HTML, CSS, JavaScript	-
	디자인시스템	BootStrap	MIT 라이선스
	IDE	Visual Studio Code	MIT 라이선스
스토리지	클라우드	AWS	AWS 라이선스

[표 8] HW 환경

팀원	프로세서	메모리
민한결	Intel® Core™ i5-1240P Processor 12M Cache, up to 4.40 GHz	16GB
안상연	Intel® Core™ i5-8265U CPU @ 1.60GHz 1.80 GHz	8GB
최필환	6/8코어 Apple Firestorm 0.6 ~ 3.23 GHz + 2코어 Apple Icestorm 0.6 ~ 2.06 GHz	16GB

3. 협업 규칙

팀원과 협의를 통해 아래와 같은 협업 규칙을 정하였다.

- CSID 조직상의 원격 저장소를 clone
- 각자 local에서 수정 후, add, commit 및 push 후 Pull Request
- 기능 단위의 branch 만들어 작업 수행
- 코드 리뷰 후 문제가 없으면 merge
- commit 단위는 하나 이상의 의미 가지지 않음
- commit 양식

[날짜, 이름] (type) 설명

타입	설명
(Add) 기능 및 코드	기능 및 코드 추가
(Mod) 기능 및 코드	정상 작동 기능 및 코드 수정
(Fix) 코드	오류가 있는 코드 수정
(Delete) 기능 및 코드	필요 없는 기능 및 코드 삭제
(Docs) 문서	문서 업로드 및 수정

예시) [23.04.03. Hankyul] (Add) todo

- 정기회의: 매 수업 시간 후 30분

4. 개발 일정

분류	내용	담당	4월 4주	5주	5월 1주	2주	3주	4주	6월 1주	2주	3주
오픈소스 분석	로그인 기능 분석	민한결									
	프론트 템플릿 분석	안상연									
	파일 저장 기능 분석	최필환									
웰컴 페이지	로그인 기능	민한결									
	화면 구현	안상연									
인프라 설정	AWS 환경 설정 / 버킷 설정	최필환									
팀 활동	팀 구성 / 공지사항	민한결									
	화면 구현	안상연									
	파일 업로드, 다운로드, 삭제	최필환									
	파일 버전관리	민한결(정)									
	화면 구현	안상연									
	파일 버전관리	최필환(부)									
최종 수정 및 발표		모두									

제 5 장 프로젝트 중간 보고

본 장은 프로젝트 중간까지의 진행과 변동 사항을 다룬다. 프로젝트 진행 사항은 백엔드와 프론트엔드로 나뉜다. 백엔드는 오픈소스 개선사항과 구현 내용, 포스트맨을 사용한 테스트 결과를 수록하였다. 프론트엔드는 {}. 변동 사항으로는 요구사항, UI, 일정 변동에 대하여 변동 이유와 내용을 작성하였다.

1. 진행 사항

1.1 백엔드

1.1.1 User

가. 오픈소스 분석(게시판 오픈소스)

- 사용자 인증에 세션 방식을 사용하고 있다. 세션 방식 사용시 해커가 id 자체를 탈취하여 사용자로 위장할 수 있다. 또한 서버에서 관리하므로 요청이 많아지면 서버 부하가 심해진다.

```
14
15     session.setAttribute("user", new UserSessionDto(user));
16
17     /* 시큐리티 세션에 유저 정보 저장 */
18     return new CustomUserDetails(user);
19 }
20 }
```

[그림 9] 게시판 오픈소스 세션 인증 코드 예시

- 템플릿 엔진을 사용하여 Controller의 return값이 HTTP Response가 아니며 Model에 데이터를 담고 String 형태로 템플릿명을 반환한다.

```
@Controller
public class UserIndexController {

    @GetMapping("/auth/join")
    public String join() { return "/user/user-join"; }
```

[그림 10] 게시판 오픈소스 컨트롤러 코드 예시

나. 오픈소스 개선

- 세션 방식의 단점 해결을 위해 토큰 방식을 사용하도록 코드를 수정하였다.

```
final String token = tokenProvider.create(user); // 토큰 생성
final UserDto responseUserDto = getResponseUserDto(userDto, user, token);
return ResponseEntity.ok().body(responseUserDto); // dto 반환
```

[그림 11] 토큰 방식 코드 예시

- Fetch를 통한 React 서버와의 통신을 위해 Controller의 return을 ResponseEntity로 하고 DTO를 그 Body로 하여 JSON형태로 정보를 전달하도록 하였다.

```
@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody User
```

[그림 12] 수정한 컨트롤러 코드 예시

다. 구현

Feature	Method	Path	Description
회원가입	Post	user/signup	서버에서 입력 정보 검증, DB 저장
로그인	Post	user/signin	토큰 발급하여 client에 전달
학생 정보 조회	Get	user	헤더 토큰 정보로 사용자 정보 조회
전체 학생 목록 조회	Get	user/all	팀원 초대 위한 전체 학생 조회

라. 테스트

- 회원가입

The screenshot displays a REST client interface. On the left, a POST request is configured for the URL `http://localhost:8080/user/signup`. The request body is a JSON object with the following fields: `studentId` (20230000001), `password` (abcd1234!), `name` (김동일), and `dept` (건설환경공학과). On the right, the response body is shown in a 'Pretty' format, returning a JSON object with: `token` (null), `id` (1), `studentId` (20230000001), `password` (null), `name` (김동일), and `dept` (건설환경공학과).

[그림 13] 회원가입 Request Body

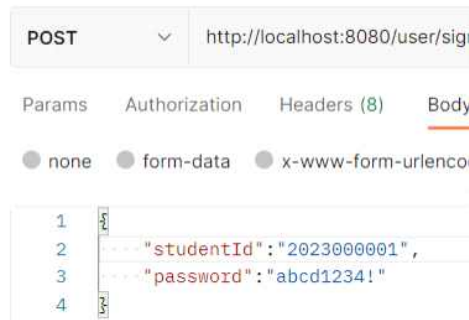
[그림 14] 회원가입 Response Body

제 5 장 프로젝트 중간 보고

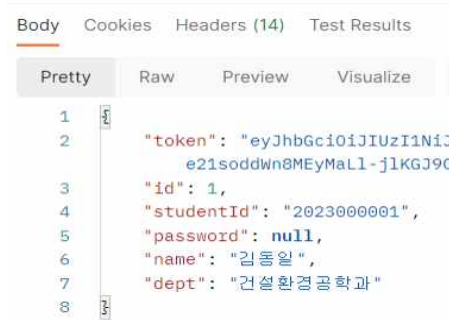
id	dept	name	password	student_id
1	건설환경공학과	김동일	\$2a\$10\$gXPbFuEpPg.xITURfIRWOE/b1D8Neha...	2023000001
2	건설환경공학과	김동이이	\$2a\$10\$K35Y9l1N48Oh81EJmwGuqP4INMeRF...	2023000002
3	건설환경공학과	김동삼	\$2a\$10\$dNACbDSeOWZB6AjvSA/5zelenX9T8fvS...	2023000003
4	건설환경공학과	김동사사	\$2a\$10\$24AZ10pSwT1XcisOsd7weJLRE784bp5...	2023000004

[그림 15] 회원가입 DB 반영

- 로그인

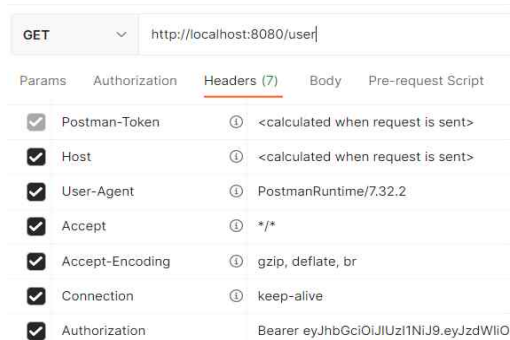


[그림 16] 로그인 Request Body

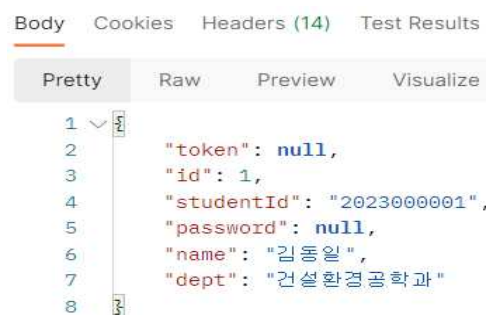


[그림 17] 로그인 Response Body

- 로그인한 학생 정보 조회

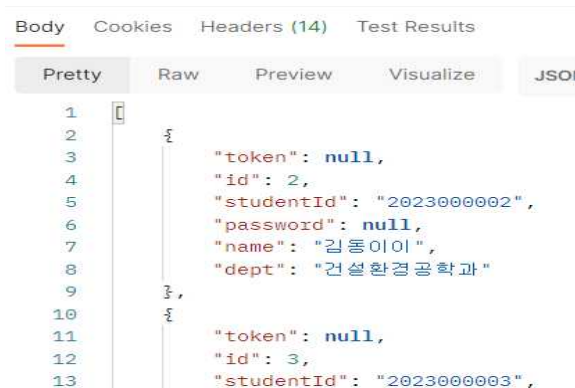


[그림 18] 학생 조회 Request Header



[그림 19] 학생 조회 Response Body

- 전체 학생 목록 조회



[그림 20] 학생 목록 조회 Response Body

1.1.2 Team

가. 오픈소스 분석(스터디 구성 오픈소스)

템플릿 엔진 사용해 Controller의 return값이 HTTP Response가 아니며 Model에 데이터를 담고 String 형태로 템플릿명을 반환한다.

```
@GetMapping("/new-study")
public String newStudyForm(@CurrentUser Account account, Model model) {
    model.addAttribute(account);
    model.addAttribute(new StudyForm());
    return "study/form";
}
```

[그림 21] 스터디 구성 오픈소스 컨트롤러 코드 예시

나. 오픈소스 개선

Fetch를 통한 React 서버와의 API 통신을 위해 Controller의 return값을 ResponseEntity로 하고 DTO를 Body로 하여 JSON형태로 전달하도록 한다.

```
@PostMapping
public ResponseEntity createTeam(@AuthenticationPrincipal String studentId,
    return teamService.createTeam(studentId, teamDto.getTeamName());
}
```

[그림 22] TeamController ResponseEntity 반환 코드 예시

```
teamRepository.save(team);
Member member = new Member(userRepository.findById(userId).get(), team, Role.Leader);
memberRepository.save(member);
ResponseDto responseTeamDto = ResponseDto.builder().data(Collections.singletonList(team));
return ResponseEntity.ok().body(responseTeamDto);
```

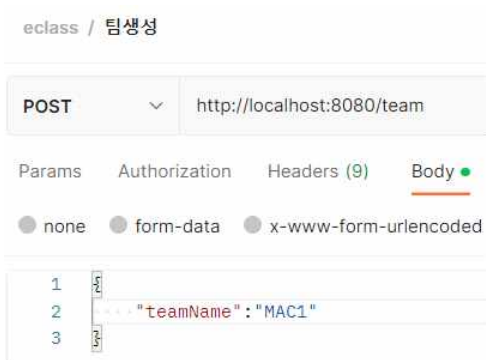
[그림 23] TeamService responseDto 생성 코드 예시

다. 구현

Feature	Method	Path	Description
팀 생성	Post	Team	팀 이름을 전달받아 팀을 생성 생성한 학생을 역할 Leader로 Member Table에 저장
팀 목록 조회	Get	team	토큰 정보로 사용자가 가입된 팀의 Id와 이름 조회
개별 팀 조회	Get	team /{teamId}	팀의 Id로 개별 팀 정보 조회
팀 삭제	Delete	team /{teamId}	역할이 Leader인 경우 팀원 초대가 이뤄지지 않은 팀을 삭제

라. 테스트

– 팀 생성



[그림 24] 팀 생성 Request Body



[그림 25] 팀 생성 Response Body

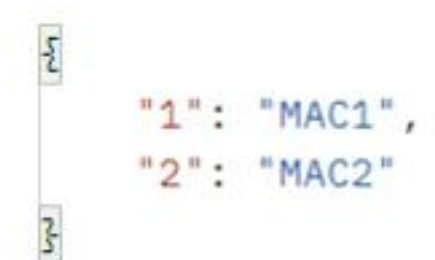
	id	team_name
▶	1	MAC1
•	NULL	NULL

[그림 26] Team 테이블

	id	role	team_id	user_id
▶	1	Leader	1	1
•	NULL	NULL	NULL	NULL

[그림 27] Member 테이블

– 팀 목록 조회



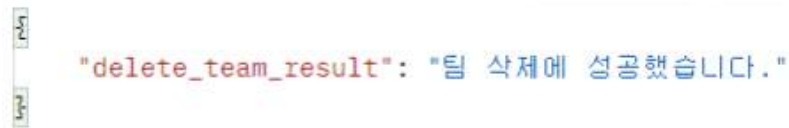
[그림 28] 팀 목록 조회 Response Body

– 개별 팀 조회



[그림 29] 개별 팀 조회 Response Body

- 팀 삭제



```
{
  "delete_team_result": "팀 삭제에 성공했습니다."
}
```

[그림 30] 팀 삭제 성공 Response Body



```
{
  "error": {
    "delete_team": "팀에 가입한 멤버가 2명 이상입니다."
  },
  "data": null
}
```

[그림 31] 팀 삭제 실패 Response Body

1.1.3 Invitation

가. 오픈소스 분석(회원 초대 수락 예제 오픈소스)

초대가 링크 형태로 Redis에 저장, TTL 만료 혹은 초대 수락 후 삭제된다.

나. 오픈소스 개선

구현 난이도를 고려하여 RDBMS를 사용할 수 있도록 하였다.

- Invitation table에 초대를 저장
- Leader와, Fellow 속성으로 팀장과 팀원 구분, User 테이블을 외래키로 사용
- isAccepted 속성으로 수락 여부를 파악
- 팀장이 초대를 삭제



```
@JoinColumn(name = "leaderId")
@ManyToOne(optional = false)
private User leader;

no usages

@JoinColumn(name = "fellowId")
@ManyToOne(optional = false)
private User fellow;

2 usages:
@Column(nullable = true)
private boolean isAccepted;
```

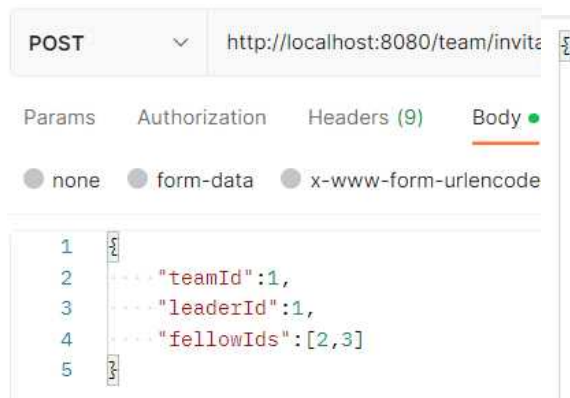
[그림 32] Invitation 테이블 속성 코드 예시

다. 구현

Feature	Method	Path	Description
팀원 초대	Post	team/invitation	팀 id와 선택한 사용자 리스트를 받아 invitation table에 저장
초대 조회	Get	team/invitation	토큰 정보로 사용자에게 온 초대 조회
초대 수락	Put	team/invitation/{invitationId}/accept	초대를 받은 사용자가 수락 버튼을 누르면 Invitation의 isAccepted 속성을 true로 변경, Member 테이블에 해당 사용자 id와 팀 id를 저장
초대 거절	Put	team/invitation/{invitationId}/reject	초대를 받은 사용자가 거절 버튼을 누르면 Invitation의 isAccepted 속성을 false로 변경, 이미 수락한 초대의 경우 변경 불가능
초대 삭제	Delete	team/invitation/{invitationId}	사용자의 역할이 Leader인 경우 발송한 초대를 삭제 가능

라. 테스트

- 팀원 초대



[그림 33] 초대 발송 Request Body



[그림 34] 초대 발송 Response Body

– 초대 조회

```
{
  "get_invitations": [
    {
      "id": 1,
      "teamId": 1,
      "leaderId": 1,
      "fellowId": 2,
      "isAccepted": null
    }
  ]
}
```

[그림 35] 초대 조회 Response Body

– 초대 수락

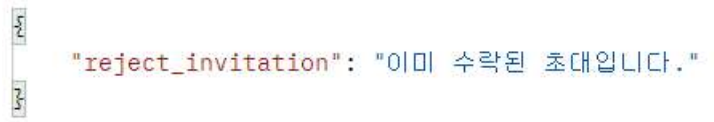
```
{
  "accept_invitation": {
    "id": 1,
    "teamId": 1,
    "leaderId": 1,
    "fellowId": 2,
    "isAccepted": true
  }
}
```

[그림 36] 초대 수락 Response Body

– 초대 거절

```
{
  "reject_invitation": {
    "id": 4,
    "teamId": 1,
    "leaderId": 1,
    "fellowId": 4,
    "isAccepted": false
  }
}
```

[그림 37] 초대 거절 성공 Response Body



```
{
  "reject_invitation": "이미 수락된 초대입니다."
}
```

[그림 38] 초대 거절 실패 Response Body

– 초대 삭제



```
{
  "delete_invitation": "초대가 삭제되었습니다."
}
```

[그림 39] 초대 삭제 성공 Response Body



```
{
  "delete_invitation": "초대한 사용자가 아닙니다."
}
```

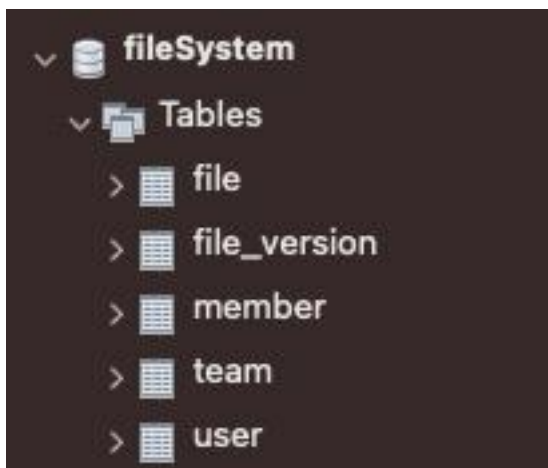
[그림 40] 초대 삭제 실패 Response Body

1.1.4 AWS 환경설정

가. DB 스키마 생성 및 RDS 연동

RDS를 생성하여 엔드포인트 및 포트정보를 통해 어플리케이션과 연동한다. DB 스키마의 경우 개발 단계이므로 ddl 옵션을 update로 설정하여, 스키마 생성 및 추가사항 반영을 한다.

– DB 스키마 생성



- RDS 생성

filesystem

요약		
DB 식별자 filesystem	CPU 4.04%	상태 🟢 사용 가능
역할 인스턴스	현재 활동 2 연결	엔진 MySQL Community

[연결 & 보안](#)
[모니터링](#)
[로그 및 이벤트](#)
[구성](#)
[유지 관리 및 백업](#)
[태그](#)

연결 & 보안

엔드포인트 및 포트	네트워킹	보안
엔드포인트 filesystem.cmwypsci3d38.ap-northeast-2.rds.amazonaws.com	가용 영역 ap-northeast-2b	VPC 보안 그룹 springboot-security (sg-0e4bc63c652014565) 🟢 활성화
포트 3306	VPC vpc-04a6f32e3f6ea838e	퍼블릭 액세스 가능

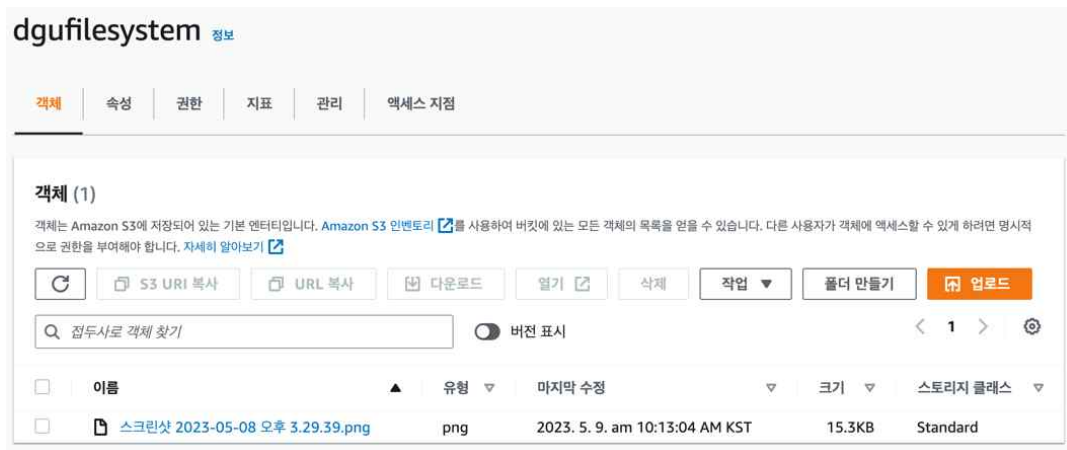
나. S3 버킷 생성 및 어플리케이션과의 연동

- application-aws.yml 설정

```

1  # AWS 버킷 정보
2  cloud:
3    aws:
4      s3:
5        bucket: dgufilesystem
6        region:
7          static: ap-northeast-2
8        stack:
9          auto: false
10
11  # AWS에서 instance profile 사용
12  credentials:
13    instance-profile: true
14  # AWS 접근 키
15  accessKey: [redacted]
16  secretKey: [redacted]
  
```

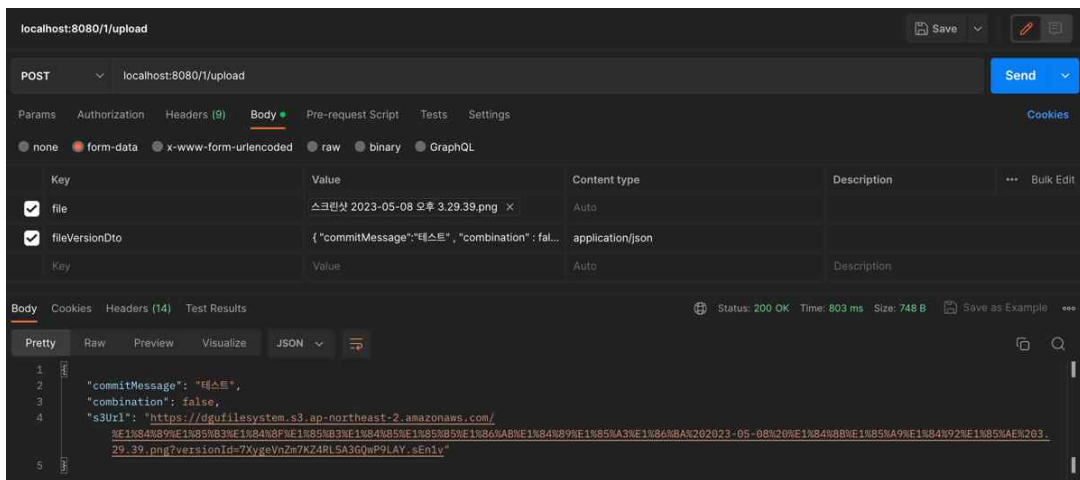
- S3 버킷 생성



1.1.5 File 업로드, 다운로드, 삭제

가. File 업로드

- POSTMAN API 테스트



토큰을 보유한 로그인한 User가 Team을 지정하여, 해당 Team의 파일 스토리지에 파일과 커밋메세지 합본 여부를 요청할 수 있다. File은 multipart-form 타입을 통해 백엔드로 요청할 수 있고, 커밋메세지와 합본 여부는 fileVersionDto를 통해 전달할 수 있다.

URL에 teamId를 pathvariable로 넣는데, User가 속한 팀이 여러 개 있을 수 있기 때문에 팀에 대한 지정이 필요하기 때문이다.

- DB조회 - File

1 • SELECT * FROM fileSystem.file;

100% 31:1

Result Grid Filter Rows: Search Edit: Export/Import:

file_id	file_name	s3file_url	member_id
2	스크린샷 2...	https://dgufilesystem.s3.ap-northeast-2.amazonaws.com/%E1%84%89%...	1

POSTMAN을 통해 요청한 파일이 저장된 것을 확인할 수 있다.

해당 File은 URL과 memberId를 갖고 있고, 이후로 동일한 File_name의 객체가 업로드되면 File객체를 추가로 저장하지 않고, fileVersion에만 추가로 저장된다.

- DB조회 - FileVersion

1 • SELECT * FROM fileSystem.file_version;

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

file_ve...	combination	commit_messa...	s3file_version_url	file_id	member_id
13	0	테스트	https://dgufilesyste...	2	2
14	0	테스트	https://dgufilesyste...	2	1

FileVersion에도 잘 저장된 것을 볼 수 있다. 여기서 현재 file_id가 2번인 fileVersion이 두 개 있는데, 이는 file_id가 동일한 파일에 대한 이후 저장 건에 대해서는 fileVersion에 저장됨을 의미한다.

- S3 조회

버전 (2)

다운로드 열기 삭제 작업

< 1 >

버전 ID	유형	마지막 수정	크기	스토리지 클래스
<input type="checkbox"/> D7KEzLqYsai1AiVrrXdMDN4RIPv_JxhX (현재 버전)	png	2023. 5. 10. am 9:21:13 AM KST	15.3KB	Standard
<input type="checkbox"/> 7XygeVnZm7KZ4RLSA3GQwP9LAY.sEn1v	png	2023. 5. 10. am 1:42:17 AM KST	15.3KB	Standard

S3 상에도 파일이 알맞게 저장된 것을 확인할 수 있다. 같은 이름. 확장자 파일이기 때문에 버전 탭에서 조회를 하면 위와 수정된 시간을 기준으로 내림차순으로 정렬되어서 보여진다.

나. File 다운로드

– POSTMAN API 테스트



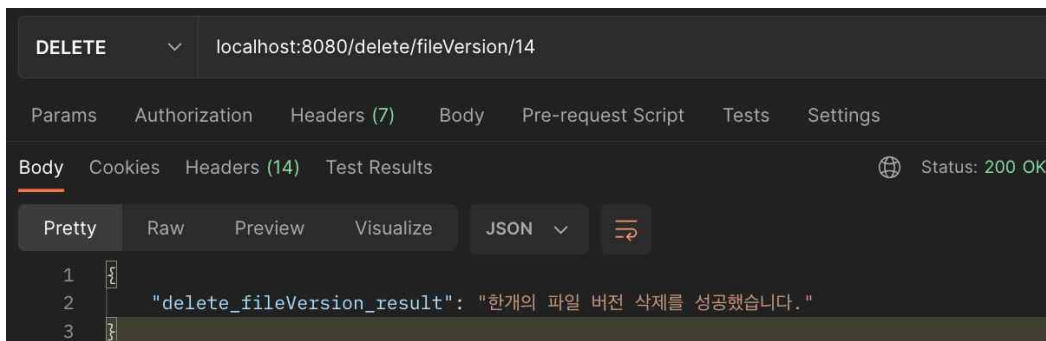
File 다운로드의 경우 특정 FileVersion에 대해 다운로드하기 때문에 URL에 pathvariable을 통해 fileId와 FileVersionId에 대한 지정을 해주어 요청해야 한다.

응답값으로는 다운로드 할 수 있도록 S3URL 링크를 반환해주었다.

해당 URL을 프론트엔드로 보내서 다운로드 처리만 구현한다면 User는 파일 다운로드를 할 수 있게 된다.

다. FileVesion 단일 삭제

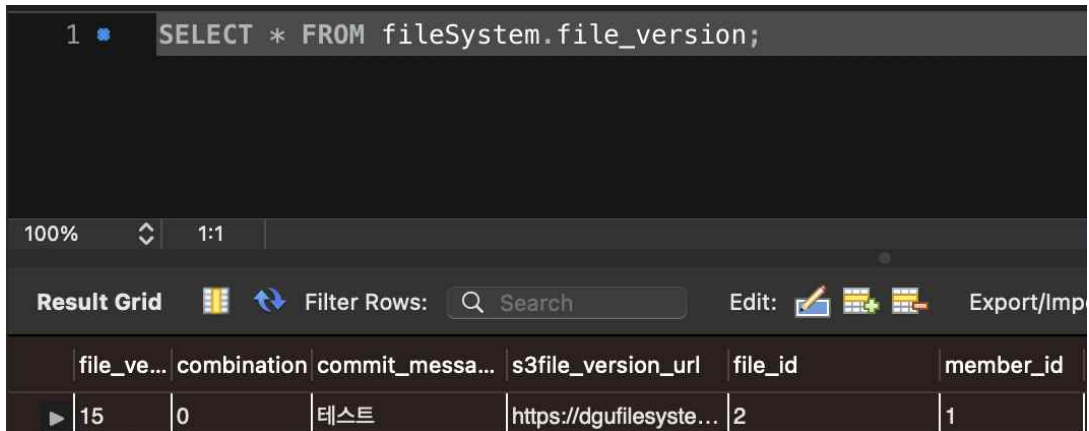
– POSTMAN API 테스트



한 개의 FileVersion만을 삭제하는 것이기 때문에 /fileVersion/{fileVersionId}와 같이 URL을 설정하였다. 별도의 요청값 없이 단순 pathvariable로 삭제할 fileVersion을 지정해준다.

응답으로 FileVersion 삭제가 성공했다는 메시지를 넘겨준다.

- DB조회 - FileVersion



```
1 SELECT * FROM fileSystem.file_version;
```

file_ve...	combination	commit_messa...	s3file_version_url	file_id	member_id
15	0	테스트	https://dgufilesyste...	2	1

FileVersion 테이블을 조회해보면 이전과는 다르게 fileVersion_id가 14번인 파일 버전에 대해 삭제가 이루어졌음을 알 수 있다.

- S3 조회

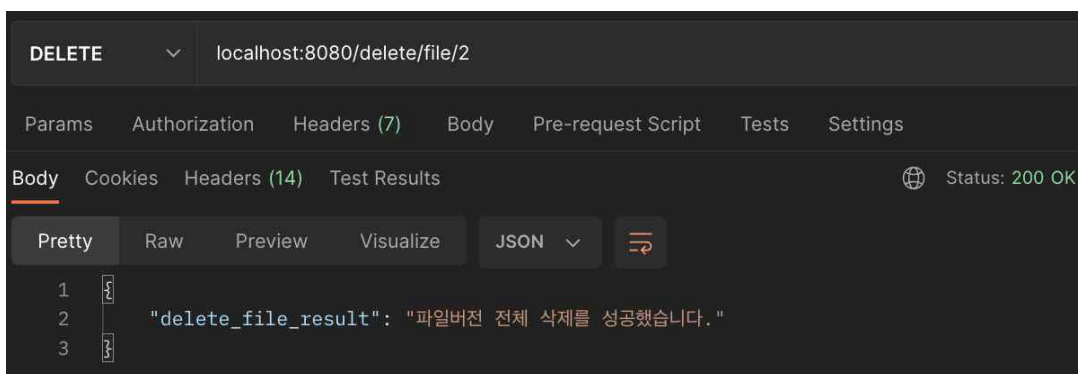


버전 ID	유형	마지막 수정	크기	스토리지 클래스
D7KEzLqYsai1AiVrrXdMDN4RIPv_JxhX (현재 버전)	png	2023. 5. 10. am 9:21:13 AM KST	15.3KB	Standard

S3에도 동일하게 fileVersion 14번에 해당하는 파일이 삭제된 것을 확인할 수 있다.

라. File 삭제(하위 FileVersion도 같이 삭제)

- POSTMAN API 테스트



DELETE localhost:8080/delete/file/2

Params Authorization Headers (7) Body Pre-request Script Tests Settings

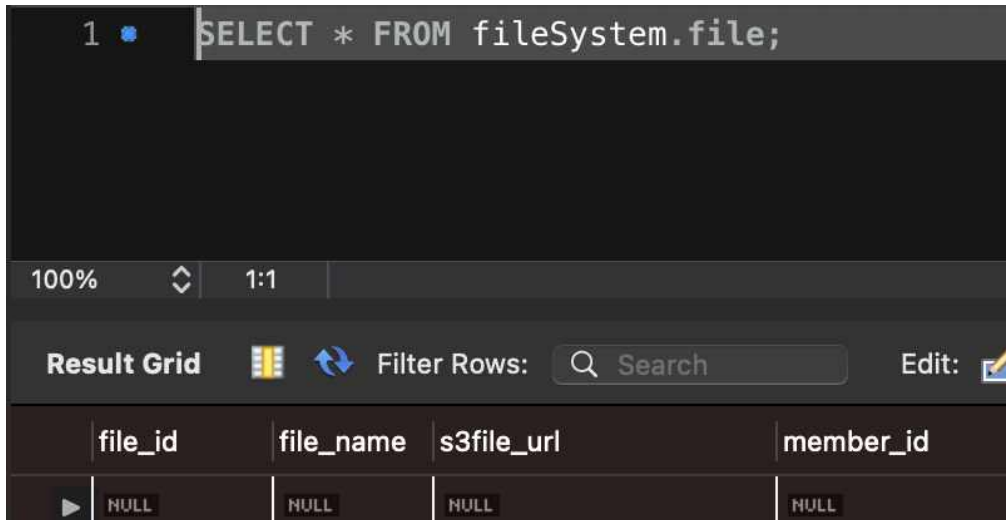
Body Cookies Headers (14) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "delete_file_result": "파일버전 전체 삭제를 성공했습니다."
3 }
```

특정 이름, 확장자의 File 자체를 삭제하는 것이기 때문에 cascade.ALL 옵션에 의해 하위 FileVersion이 함께 삭제된다. /file/{fileId}와 같이 삭제할 파일을 지정해주고, 요청하면, 파일 삭제 성공에 대한 응답 값을 리턴해준다.

- DB조회 - File

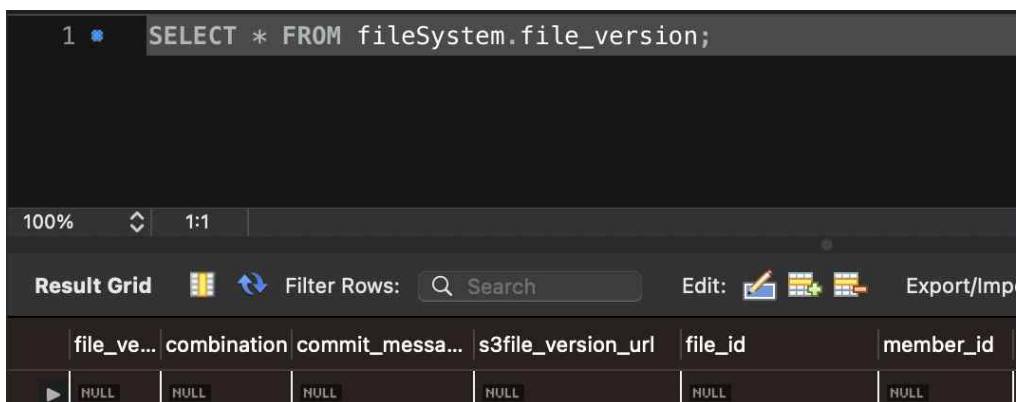


The screenshot shows a database query interface. At the top, a SQL query is entered: `SELECT * FROM fileSystem.file;`. Below the query, there are controls for zooming (100%) and a 1:1 view. A 'Result Grid' section contains a search bar and an 'Edit' button. The results are displayed in a table with the following columns: `file_id`, `file_name`, `s3file_url`, and `member_id`. The data row shows all four fields as `NULL`.

file_id	file_name	s3file_url	member_id
NULL	NULL	NULL	NULL

기존의 file_id가 2번이었던 file이 삭제된 것을 확인할 수 있다.

- DB조회 - Fileversion



The screenshot shows a database query interface. At the top, a SQL query is entered: `SELECT * FROM fileSystem.file_version;`. Below the query, there are controls for zooming (100%) and a 1:1 view. A 'Result Grid' section contains a search bar and buttons for 'Edit', 'Export/Imp', and 'Filter Rows'. The results are displayed in a table with the following columns: `file_ve...`, `combination`, `commit_messa...`, `s3file_version_url`, `file_id`, and `member_id`. The data row shows all six fields as `NULL`.

file_ve...	combination	commit_messa...	s3file_version_url	file_id	member_id
NULL	NULL	NULL	NULL	NULL	NULL

File 엔티티의 FileVersion과 맵핑 시켜준 FileVersionList 컬럼에 CASCADE All 옵션을 걸어주었기 때문에 fileid가 2인 fileVersion의 객체들이 다 삭제된 것을 확인할 수 있다.

- S3 조회



The screenshot shows an S3 object listing interface. At the top, it says '객체 (0)' (Objects (0)). Below this, there is a description: '객체는 Amazon S3에 저장되어 있는 기본 엔티티입니다. Amazon S3 인벤토리 [icon]를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 ;'. There are several buttons: 'S3 URI 복사', 'URL 복사', '다운로드', '열기', '삭제', '작업', '폴더 만들기', and '업로드'. A search bar contains the text '접두사로 객체 찾기'. A checkbox for '버전 표시' is checked. Below the search bar, there is a table header with columns: '이름', '유형', '마지막 수정', and '크기'. The table body is empty, and a message at the bottom says '객체 없음' (No objects) and '이 버킷에 객체가 없습니다.' (There are no objects in this bucket).

위와 같이 파일이 삭제되었기 때문에 S3에 파일이 존재하지 않는 것을 확인할 수 있다.

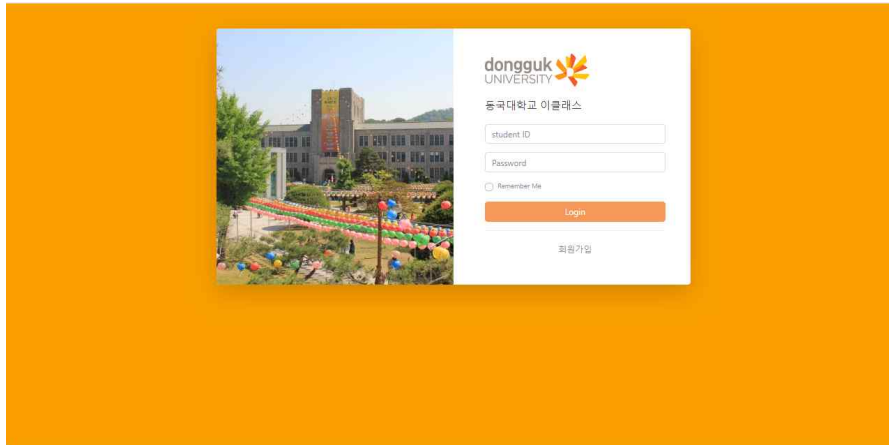
제 5 장 프로젝트 중간 보고

1.2 프론트엔드

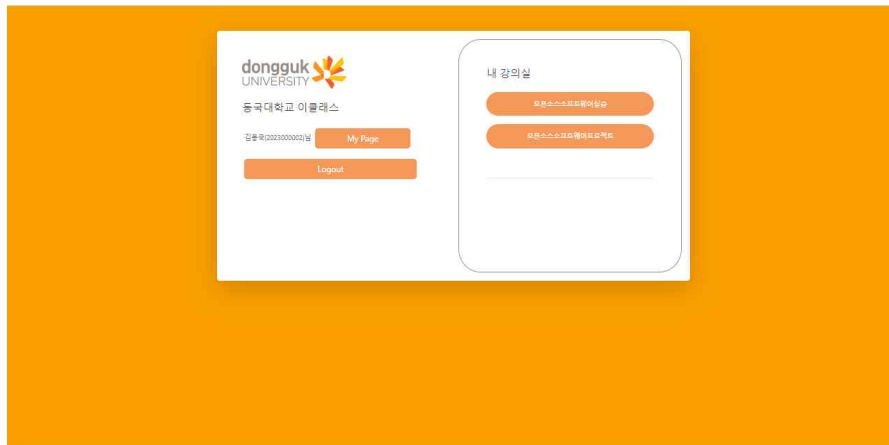
1.2.1 기존 코드 분석 및 수정

가. UI 디자인 수정

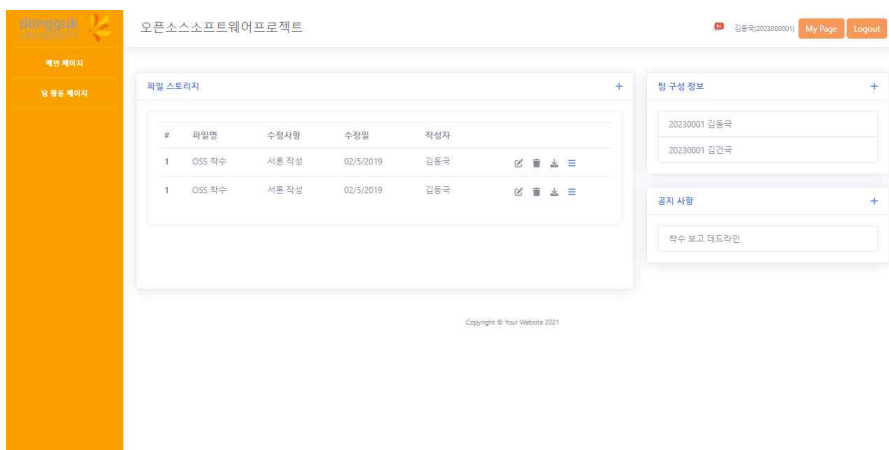
기존 템플릿 디자인을 제작한 UI 프로토타입에 맞게 수정하였다.



[그림 41] 로그인 UI



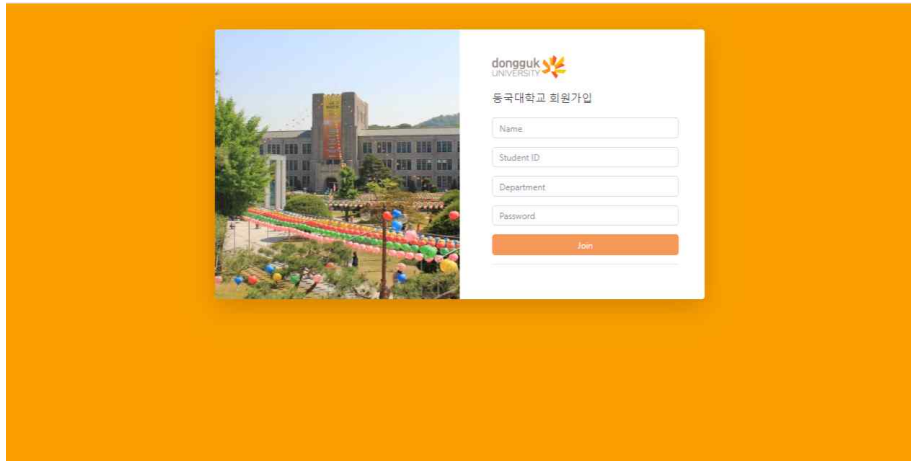
[그림 42] 과목 선택 UI



[그림 43] 팀 활동 UI

나. 회원가입 페이지 추가

로그인 페이지 UI에서 기존 학번, 비밀번호 2가지 input에 이름, 학과를 입력 받는 2개의 input을 추가하였고 login 버튼을 join 버튼으로 변경하였다.



[그림 44] 회원가입 UI

1.2.2 리액트 도입

백엔드와의 연동을 위해 리액트 환경으로 변경 후, css 파일과 이미지 파일들을 리액트 폴더로 이동시키고 모든 페이지들의 기존 html 코드를 리액트 환경에 맞는 js 파일로 수정하였다. 추가로 AppRouter.js 파일로 각 페이지들을 연결하였다.

```
demo
├─ node_modules
├─ public
│  └─ dongguk_logo.jpg
│  └─ dongguk_logo.png
├─ src
│  └─ AppRouter.js
│  └─ bootstrap.css
│  └─ Join.js
│  └─ Login.js
│  └─ Select.js
│  └─ Team.js
├─ package-lock.json
└─ package.json
```

[그림 45] 프로젝트 디렉토리

```

import React from "react";
import "./index.css";
import App from "./App";
import Login from "./Login";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Join from "./Join";
import Select from "./Select";
import Team from "./Team";
//import {Typography, Box} from "@mui/material";

function AppRouter() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/join" element={<Join />} />
          <Route path="/select" element={<Select />} />
          <Route path="/team" element={<Team />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default AppRouter;

```

[그림 46] AppRouter.js

1.2.3 회원가입 페이지와 백엔드 연동

fetch 함수를 사용하여 `http://localhost:8080/user/signup` 주소로 POST 요청 보냈다. status가 400일 경우(bad request) json 데이터를 불러와서 error에 해당하는 메시지를 alert창으로 보여주도록 하였다.

```

fetch("http://localhost:8080/user/signin", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(data),
})
.then((response) => {
  if (response.ok) {
    return response.json();
  } else {
    return response.json().then((jsonData) => {
      // `showErrorMessages` 함수를 호출하여 메시지 출력
      // 에러를 throw 하여 다음 catch 블록으로 이동
      throw new Error(showErrorMessages(jsonData));
    });
  }
})
.then((data) => {
  sessionStorage.setItem("token", data.token);
  goSelect();
})
.catch((error) => {
  alert(error.message);
});

const showErrorMessages = (jsonData) => {
  const errorMessages = Object.values(jsonData.error).join("\n");
  // 메시지들을 결합하여 alert 창에 보여줍니다.
  return errorMessages;
};

```

[그림 47] 회원가입 연동 코드

입력이 없는 input이 있는 경우 모든 항목을 입력해야 한다는 메시지를 alert창으로 보여주도록 하였다.

```
if (!name || !studentId || !dept || !password) {
  alert("모든 항목을 입력해주세요.");
  return;
}
```

[그림 48] 미입력 input 검증 코드

1.2.4 로그인 페이지와 백엔드 연동

요청 보내고 bad request에 따른 경고창을 출력하는 코드는 회원가입 페이지와 동일하게 구현하였고, 로그인 이후 페이지부터 할당받은 토큰을 이용하여 각 사용자에게 맞는 서비스를 제공할 것이기 때문에 세션 스토리지에 토큰을 저장하는 코드를 추가하였다.

```
fetch("http://localhost:8080/user/signin", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(data),
})
.then((response) => {
  if (response.ok) {
    return response.json();
  } else {
    return response.json().then((jsonData) => {
      // `showErrorMessages` 함수를 호출하여 메시지 출력
      // 에러를 throw 하여 다음 catch 블록으로 이동
      throw new Error(showErrorMessages(jsonData));
    });
  }
})
.then((data) => {
  sessionStorage.setItem("token", data.token);
  goSelect();
})
.catch((error) => {
  alert(error.message);
});
```

[그림 49] 로그인 연동 코드

1.2.5 과목 선택 페이지, 팀 활동 페이지와 백엔드 연동.

세션 스토리지에 저장되어있는 토큰을 요청 헤더에 넣어 사용자 정보를 불러와 사용자의 이름과 학번을 각 페이지에 보여준다. 로그아웃 버튼 클릭 시, 세션 스토리지에 저장되어있는 토큰 삭제 후 로그인 페이지로 이동한다.

```
function Select() {
  const [userInfo, setUserInfo] = useState({});
  const navigate = useNavigate();

  useEffect(() => {
    const token = sessionStorage.getItem("token");
    console.log(token);
    fetch("http://localhost:8080/user", {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
      .then((response) => response.json())
      .then((data) => {
        setUserInfo(data);
      })
      .catch((error) => console.log(error));
  }, []);

  const handleLogout = () => {
    sessionStorage.removeItem("token");
    navigate("/login");
  };

  const goTeam = () => {
    navigate("/team");
  };
}
```

[그림 50] 그림 61

2. 변동 사항

2.1 협업 규칙 변동

기존 협업 규칙의 경우 팀원 이름으로 된 개발 브랜치에 push하고 main에 merge하는 방식으로 진행하기로 하였다. 이는 배포 버전과 개발 버전의 분리가 이뤄지지 않으므로 아래와 같이 브랜치 규칙을 변경하였다.

[표 9] 브랜치 규칙

이름	역할
main	배포 가능한 버전
develop	기능 개발을 위한 브랜치들을 병합
feature/팀원명	팀원별 할당된 기능 개발 후 커밋

2.2 요구사항 변동

기존 팀 구성 방식은 팀장이 일방적으로 팀원을 선택하는 방식이었다. 이는 실제 팀 프로젝트팀 구성 방식과는 상이하다. 따라서 팀원에게 초대장을 발송하는 방식으로 변경하였고 구현 완료하였다.

[표 10] 요구사항 변동

사용자	메뉴	필요 기능	기능 설명
학생	팀 활동	팀 생성	팀이름을 입력하여 팀을 생성하고 생성한 학생을 팀장으로 등록한다.
		팀원 초대	팀장이 학생 목록에서 선택하여 초대장을 발송하고 상대 학생은 초대장을 수락하여 팀에 가입하거나 초대장을 거절할 수 있다.
		업로드	파일과 관련 정보를 업로드할 수 있다.
		다운로드	파일을 다운로드할 수 있다.
		삭제	특정 파일 및 전체 파일 삭제 기능이다.
		버전관리	이전 버전 조회 및 다운로드 가능하다.
		공지사항	팀 내의 공지사항을 등록하고 조회한다.
	메인 페이지	로그인/로그아웃	학번, 비밀번호로 접속 가능하게 한다.

2.3 UI 변동

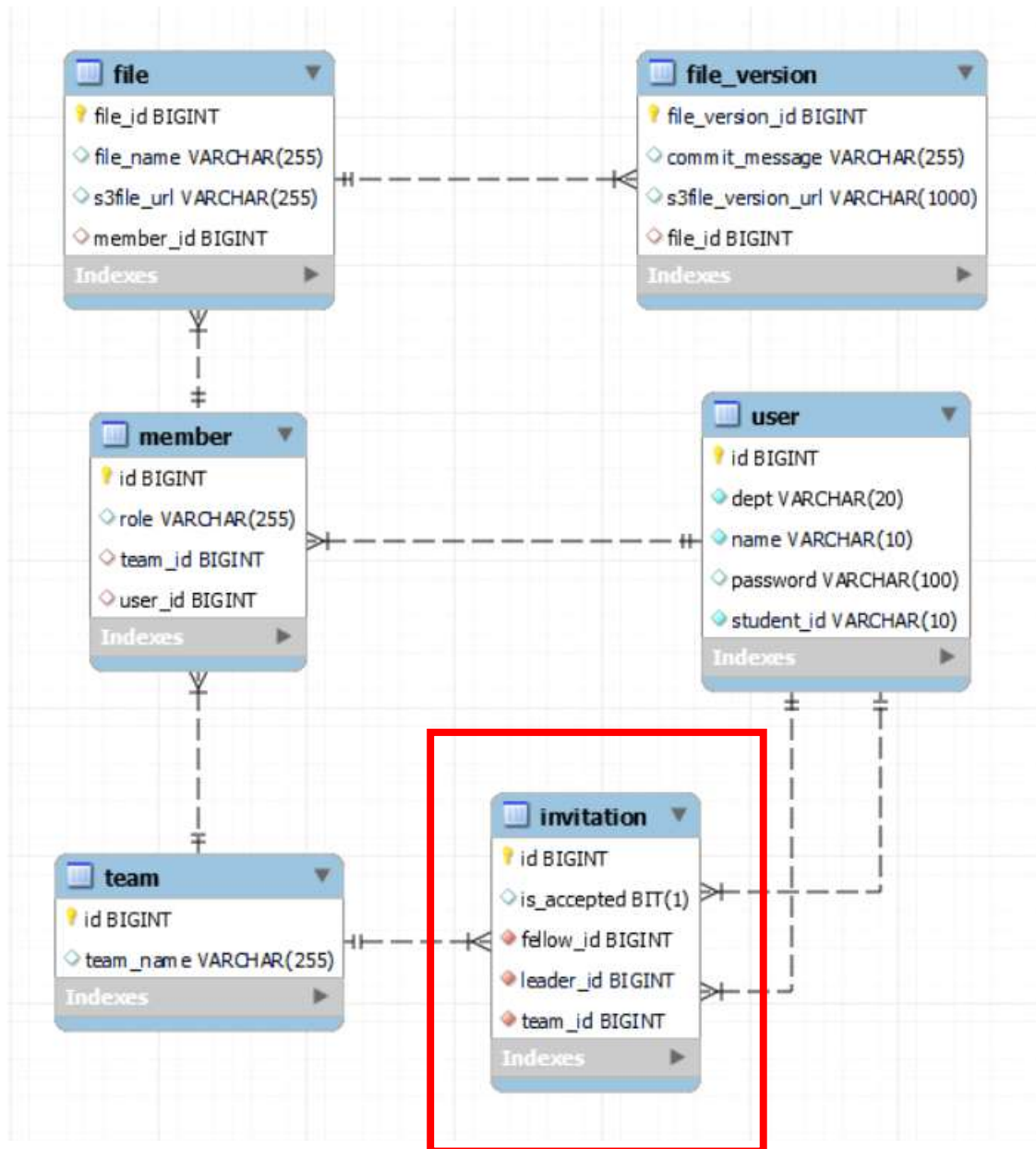
팀 활동 페이지 초대 기능과 팀 선택 및 생성 기능 구현을 위하여 UI를 수정하였다. 첫 번째로, 팀원 초대 기능을 위해 사용자 정보 옆에 벨 형태의 아이콘을 추가하였고, 아이콘을 누르면 초대자와 팀명을 알려주고 수락, 거절 버튼을 누를 수 있는 창이 뜨게 하였다. 두 번째로, 좌측 팀 활동 메뉴에 드롭다운 메뉴를 추가하여 사용자가 속해 있는 팀 목록과 팀을 생성할 수 있는 플러스 버튼을 넣어주었다.



[그림 51] 팀 활동 페이지 수정된 UI

2.4 DB 변동

초대장 저장을 위한 Invitation 테이블의 추가를 반영한 ERD는 아래와 같다.



[그림 52] Invitation 테이블 추가 반영 ERD

2.4 개발 일정 변동

최필환 팀원의 경우 파일 스토리지에 대한 백엔드 부분이 조기 마무리되어 파일 목록 페이지 처리를 추가하였다. 기존 팀 활동 영역 요청 및 응답 처리를 팀 구성/공지사항 요청 및 응답처리로 세분화하고 안상연 팀원과 민한결 팀원이 분담하는 것으로 일정을 조정하였다. 이에 민한결 팀원이 담당하였던 예외 처리, 배포는 최필환 팀원이 통합 테스트와 함께 진행하기로 하였다.

분류	내용	담당	4월4주	5주	5월1주	2주	3주	4주	5월5주	6월1주	6월2주
웰컴 페이지	로그인 기능 구현	민한결									
	프론트 템플릿 수정	안상연									
파일 스토리지	AWS 설정/파일 업로드 구현	최필환									
팀 활동 페이지	팀 구성 기능 구현	민한결									
웰컴 페이지	로그인 요청 및 응답 처리	안상연									
파일 스토리지	버전별 파일 관리/다운로드	최필환									
	파일 관련 기능 구현										
팀 활동 페이지	공지사항 기능 구현	민한결									
	팀 활동 요청 및 응답 처리	안상연									
	팀 구성 요청 및 응답 처리										
	파일 업로드, 다운로드, 삭제	최필환									
	파일 목록 페이지 처리										
	예외 처리, 배포	민한결									
	공지사항 요청 및 응답 처리										
	파일 스토리지 요청/응답	안상연									
	통합 테스트, 배포	최필환									
최종 수정 및 발표		모 두									