

# 종합설계 프로젝트 수행 보고서

팀번호	S5-10	
프로젝트명	같이보고 같이듣고 같이쓰는 실시간 웹 화이트보드 '같이보드'	
문서제목	수행보고서	
	2차발표 중간보고서	
	3차발표 중간보고서	
	최종결과보고서	●

2020.06.18

팀장	2014154014	나승철
팀원	2014154028	윤상현
지도교수	방영철	(인)

## ❖ 목차 ❖

<b>I 서론</b>	<b>4</b>
(1) 작품선정 배경 및 필요성	4
(2) 기존 연구/기술동향 분석	5
(3) 개발 목표	6
(4) 개발일정 및 역할분담	7
(5) 개발 환경	9
1). 웹앱/백엔드 통합 개발 환경	9
2). 웹앱 개발 환경	10
3). 백엔드 개발환경	11
<b>II 본론</b>	<b>12</b>
(1) 개발 내용	12
1) 시스템 수행 시나리오	12
(2) 문제 및 해결방안	13
(3) 시험 시나리오	15
1) 로드맵	15
2) 세부 테스트 계획	16
(4) 상세 설계	18
1) 웹앱 모듈 상세설계	18
2) 데이터베이스 컬렉션	28
3) 백엔드 서버 모듈 상세설계	29
4) 시퀀스 다이어그램	34
(5) Prototype 구현	41
1) 프로토타입 동작 흐름도	41
2) 프로토타입 세부 구성	42
(6) 시험/ 테스트 결과	53
1) 사용성 테스트	53
2) 성능테스트	58
(7) Coding & DEMO	59

## ❖ 문서 정보

### 문서 수정 내역

작성일	대표작성자	버전	수정내용	비고
2020.01.15	나승철 (팀장)	1.1	수행계획서	최초작성 : 서론 파트 작성 및 수정
2020.01.19	윤상현 (팀원)	1.2	수행계획서	개발환경파트 작성 및 수정
2020.01.19	나승철 (팀장)	1.3	수행계획서	본론파트 작성 및 수정
2020.01.20	윤상현 (팀원)	1.4	수행계획서	문제점 및 해결방안 작성 및 수정
2020.01.21	나승철 (팀장)	1.5	수행계획서	시험 시나리오 파트 작성 및 수정
2020.01.23	나승철 (팀장)	1.6	수행계획서	교수님 검토 후 제출준비
2020.02.23	나승철 (팀장)	2.0	수행계획서	2차 보고서 작성
2020.02.23	윤상현 (팀원)	2.1	수행계획서	웹앱 파트 상세설계 작성
2020.02.24	나승철 (팀장)	2.2	수행계획서	백엔드파트 상세설계 작성 완료 및 2.1버전과 문서 통합
2020.04.24	나승철 (팀장)	3.0	수행계획서	최초작성 : 프로토타입 구현파트 작성
2020.04.24	윤상현 (팀원)	3.1	수행계획서	프로젝트 지원도구 프로토타입 파트 작성
2020.04.25	나승철 (팀장)	3.2	수행계획서	3.0 문서 통합 후 최종 검토
2020.06.17	나승철 (팀장)	4.0	수행계획서	4.0 문서 생성 및 초안 작성
2020.06.18	윤상현 (팀원)	4.1	수행계획서	문서 내용 추가작성
2020.06.18	나승철 (팀원)	4.2	수행계획서	문서 내용 추가작성
2020.06.19	나승철 (팀원)	4.3	수행계획서	문서 내용 통합 및 검토

### 문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 1~6) II. 본론 1~3)	I. 서론 1~6) II. 본론 1~4)	I. 서론 1~6) II. 본론 1~5)	I. 서론 1~6) II. 본론 1~7)	I, II, III

이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계” 교과목에서 프로젝트  
**“같이보고 같이듣고 같이쓰는 실시간 웹 화이트보드 같이보드”**를 수행하는  
**(S5-10 나승철, 윤상현)**팀이 작성한 것으로,  
 사용하기 위해서는 팀원들의 허락이 필요합니다.

## I 서론

### (1) 작품선정 배경 및 필요성

□ 개발 배경	
수요배경	○ 예전과는 다르게, 회사나 대학교 뿐 만 아니라 초,중,고 등에서도 PBL 수업을 하는 등, 전반적인 분야에서 팀프로젝트 중심의 활동이 증가하는 추세.
	○ 팀 구성원들의 시간 조율이 힘들어서 직접 만나 팀프로젝트를 진행하기 어려움.
	○ 팀 프로젝트의 활동인 브레인스토밍이나 화상회의등의 활동 증가.
필요성 1	○ 원격으로 협업하여 브레인스토밍이나 회의를 할 수 있는 서비스 필요 ○ 원활한 프로젝트 진행을 위한 칸반과 같은 서비스 필요.
필요성 2	○ 실시간으로 서로의 작업을 보면서 상호작용할 수 있는 동기화 서비스 필요.

## (2) 기존 연구/기술동향 분석

□ 국내사례 - BeeCanvas	
장점	<input type="checkbox"/> 다양한 프로젝트 지원도구 제공( 일정관리, 공유문서함 기능 ). <input type="checkbox"/> 화상회의 서비스 제공.
단점	<input type="checkbox"/> 고정적인 크기의 화이트보드 제공. <input checked="" type="radio"/> 여러 사람이 같이 활동하는 화이트보드인 만큼, 사용자의 필요에 따라 동적으로 크기가 변하는 화이트보드가 필요.

□ 해외사례 - Microsoft Whiteboard	
장점	<input type="checkbox"/> 다양한 프로젝트 지원도구 제공( 일정관리, 칸반 기능 ). <input type="checkbox"/> 동적크기의 화이트보드 제공.
단점	<input type="checkbox"/> 화상회의 서비스 없음.

□ 해외사례 - ZiteBoard	
장점	<input type="checkbox"/> 동적크기의 화이트보드 제공. <input type="checkbox"/> 화상회의 서비스 제공.
단점	<input type="checkbox"/> 일정관리, 칸반과 같은 프로젝트 지원도구 없음.

□ 시사점 : 원활한 협업지원 서비스를 위해 도출된 요구사항	
1	<input type="checkbox"/> 동적크기를 가지는 웹 화이트보드
2	<input type="checkbox"/> 브레인스토밍과 같은 아이디어 도출 지원 도구
3	<input type="checkbox"/> 화상 회의를 위한 Video Chat
4	<input type="checkbox"/> 다양한 프로젝트 지원도구

### (3) 개발 목표

□ 목표	
주 목표	❖ 프로젝트 지원도구를 제공하는 동적크기의 실시간 협업이 가능한 웹 화이트보드 서비스 제공.
세부 목표	□ 브레인스토밍 지원을 위한 다양한 그리기 기능 구현
	□ 실시간 협업을 위한 동기화기능 구현
	□ 회원가입 없이 소셜 로그인으로 사용할 수 있는 환경 제공
	□ N:M Video Chat 서비스 제공
	□ 프로젝트 지원을 위한 Kanban 및 TaskTimer 기능 제공

## (4) 개발일정 및 역할분담

□ 프로젝트 개발일정		
구분	기한	내용
Stage-1	01.02 ~ 01.05	(가) 웹앱 프레임워크 설치 및 소셜 로그인 기능 구현 (나) 백엔드 프레임워크 설치 소셜 로그인 기능 구현
Stage-2	01.06 ~ 01.12	(가) 웹앱 그리기 도구(펜,형광펜,올가미,지우개) (나) 웹앱 인피니트캔버스, 줌인/아웃 기능
Stage-3	01.12 ~ 01.19	(가) 웹앱 그리기 도구(다각형, 원, 카드) (나) 웹앱 미니맵 (다) 제 1차 수행보고서 작성 및 제출(~01.22)
Stage-4	01.20 ~ 01.26	(가) 웹앱 도형간의 간선 잇기 기능, 단축키 기능 (나) 웹앱 칸반 기능
Stage-5	01.27 ~ 02.02	(가) 웹앱 이미지, PPT, PDF Import기능, 프로젝트 Export기능 (나) 웹앱 타임타이머 기능
Stage-6	02.03 ~ 02.09	(가) 백엔드 화이트보드 세션기능, 사용자 권한설정기능 (나) 웹앱/백엔드 : VideoChat/TextChat기능
Stage-7	02.10 ~ 02.16	(가) 커서트래킹, 그리기도구 동기화 기능 (나) 도형 그리기도구 동기화 기능 (다) 제 2차 수행보고서 작성 및 제출(~02.19)
Stage-8	02.17 ~ 02.23	(가) Redo/Undo기능 (나) 칸반/타임타이머 동기화 기능
Stage-9	02.24 ~ 03.01	(다) 공유파일함 기능

□ 역할분담		
구분	인원	내용
1	나승철	□ 프로젝트 로드맵 각 스테이지의 (가) 파트 담당.
	윤상현	□ 프로젝트 로드맵 각 스테이지의 (나) 파트 담당.
2	전체	□ 프로젝트 로드맵 각 스테이지의 (다) 파트는 공동 담당

□ 프로젝트 수행일정									
날짜 스테이지	1월					2월			
	1주	2주	3주	4주	5주	1주	2주	3주	4주
Stage-1									
Stage-2									
Stage-3									
Stage-4									
Stage-5									
Stage-6									
Stage-7									
Stage-8									
Stage-9									



## (5) 개발 환경

### 1). 웹앱/백엔드 통합 개발 환경

□ Web Storm (IDE)	
버전	2019.2.1.0
개발사	JetBrains
특징	내용
1	Angular, React, Vue.js 등의 프레임워크 지원
2	크로스 플랫폼으로 윈도우, 맥, 리눅스에서 동작

□ NodeJS	
버전	10.16.03
개발사	NodeJS Foundation
특징	내용
1	Javascript 런타임
2	다양한 라이브러리 및 프레임워크 제공
3	서버 사이드에서도 동일하게 개발가능

□ Ubuntu	
버전	18.04
개발사	캐노니컬 유한회사

□ Docker	
버전	18.09.0-0506
개발사	Docker Inc.

## 2). 웹앱 개발 환경

□ Angular CLI	
버전	8.1.2
개발사	Google
특징	내용
1	SPA(Single Page Application) 개발 지원
2	Typescript 사용
3	웹 애플리케이션 프레임워크

□ PaperJS	
버전	0.12.3
개발자	Jürg Lehni & Jonathan Puckey
특징	내용
1	HTML5 Canvas 기반 그래픽 처리 모듈
2	선 및 도형 관련 함수 지원
3	애니메이션 처리 기능 제공

### 3). 백엔드 개발환경

□ NestJS	
버전	6.0.0
개발자	Kamil Myśliwiec
특징	내용
1	Typescript 사용
2	Spring MVC와 유사한 아키텍처
3	다양한 플러그인 제공

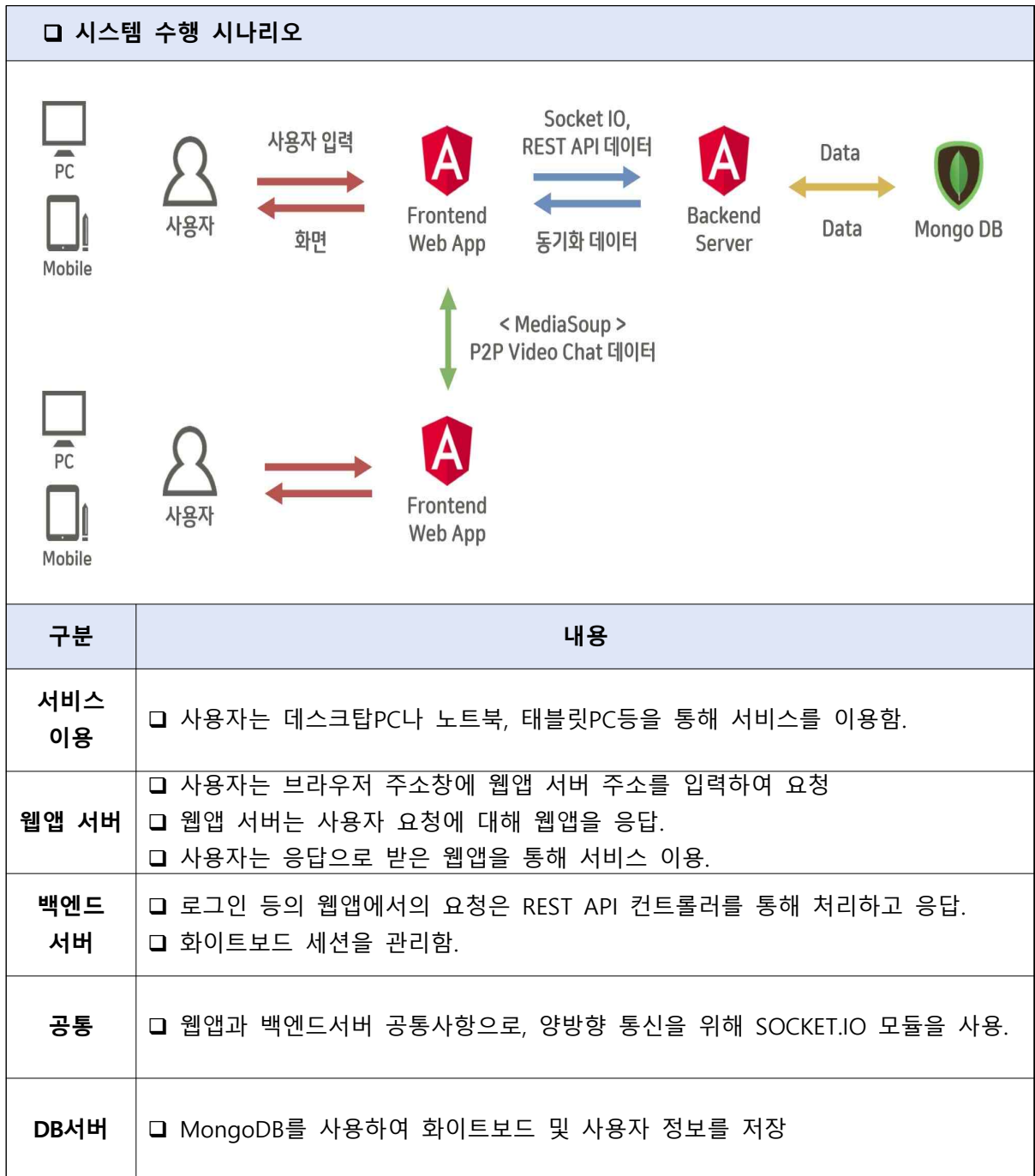
□ Postman	
버전	7.13.0
개발사	Postman, Inc.
특징	내용
1	손쉽게 API 테스트 가능
2	그룹화 및 API 테스트 기록
3	동기화 기능 제공

□ MongoDB	
버전	4.2.0
개발사	MongoDB Inc.
특징	내용
1	NoSQL 데이터베이스이고 문서 지향 데이터베이스임
2	복잡한 계층 관계를 하나의 레코드로 표현 가능
3	데이터의 구조를 미리 정의할 필요 없음

## II 본론

### (1) 개발 내용

#### 1) 시스템 수행 시나리오



## (2) 문제 및 해결방안

□ 문제점	
순번	예상되는 문제점
1	동기화가 잘 이루어질 수 있도록 설계 및 구현 필요
2	편리하고 세련된 UI 및 UX 제공
순번	실제로 만난 문제점
1	터치와 마우스 입력 사이의 차이와 운영체제마다 다른 이벤트 핸들링 방법
2	글 상자가 Word Wrap 기능을 제공하지 않아 직접 구현해야 하는 문제
3	미니맵에 반영하는 속도에 따른 성능 문제
4	도형을 그릴 때 가로길이나 세로 길이가 1이 되면 다시 되돌아오지 않는 문제
5	브라우저의 좌표와 인피니트 캔버스의 좌표가 실제로는 다르고 줌 배율에 따라서 다른곳을 가르키고 있는 문제
6	브라우저 영역 밖으로 나간 상태에서 이벤트가 발생해 특정 이벤트를 감지 못하고 넘어가는 문제
7	해당 뷰가 정의된 컴포넌트 파일에서만 이벤트 리스너 등록 가능해 리스너들의 접근성이 낮은 것과 같은 이벤트를 중복으로 정의할 수 없는 Angular CLI의 이벤트 등록 방식의 문제
8	인피니트 캔버스에 그려져 있는 그리드가 이동하거나 줌 배율을 변경하게 되면 이동하지 않아도 이동하는 것처럼 보이는 문제

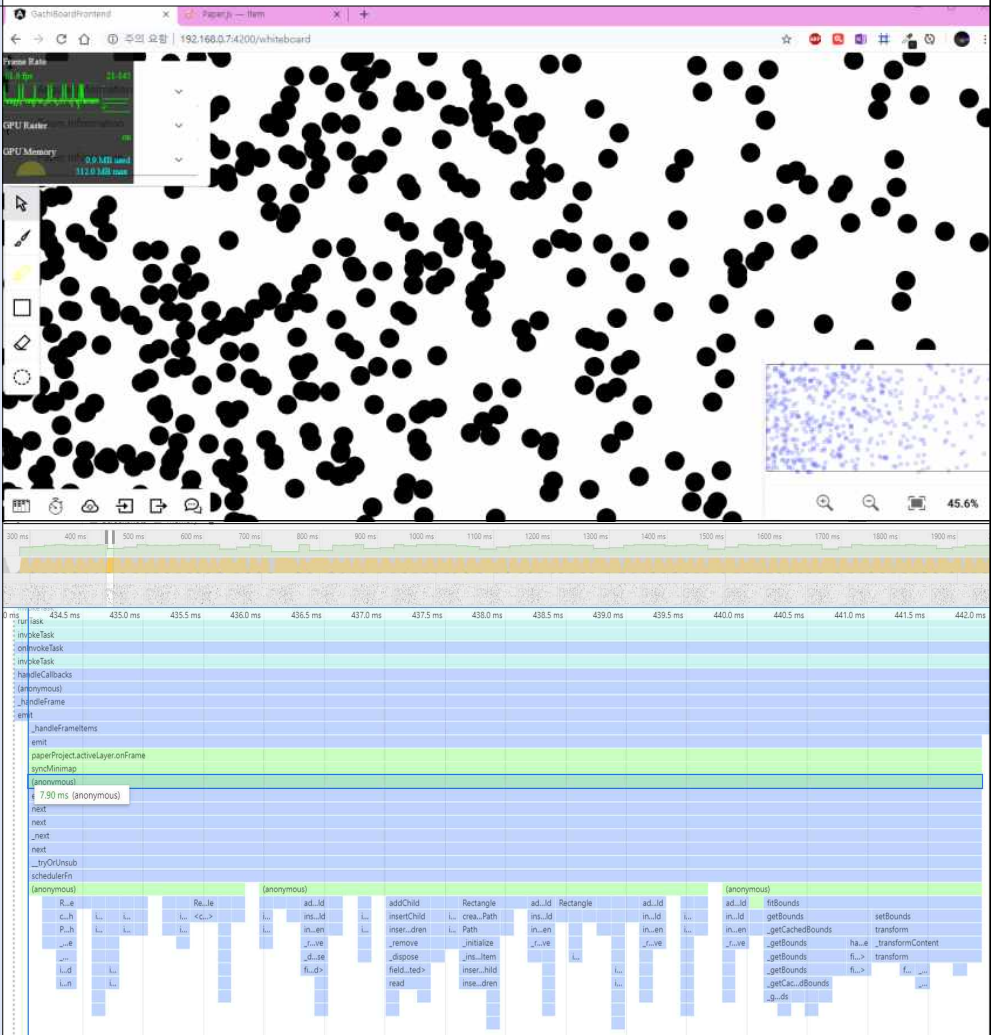
□ 해결방안	
순번	예상되는 문제점
1	<p>첫 페이지 로드 시에만 초기화 작업을 진행하고 Socket.IO를 통해 다른 사람이 추가한 데이터를 받거나 내가 추가한 데이터만을 전송한다. 이때 서버를 거쳐서 브로드캐스팅 되는데 웹앱에서 그려지거나 이동하거나 제거되는 데이터들을 모두 서버에 반영하고 브로드캐스팅 하여 빠르게 동기화 할 수 있고 서버쪽에서 추가 작업 없이 최종 결과물을 보유할 수 있도록 구성한다.</p> <p>두 번째로는 동기화 데이터의 크기에 의해서 동기화 작업이 느려지지 않도록 최적화 한다. 최소한의 정보를 서버에서 갖고 있다가 클라이언트의 요청에 따라 보드의 데이터를 전달해준다. 데이터를 전달받은 클라이언트는 최소한의 데이터를 통해 알맞은 위치에 채워 넣거나 개체를 그리게 된다.</p>
2	<p>기본적인 인터페이스들의 디자인은 Angular에서 제공하는 Material 디자인을 사용한다. 이 디자인은 Android의 개발사인 Google이 디자인 가이드를 만들어서 제공하는 디자인과 같은 디자인으로 웹 개발을 할 수 있도록 Google에서 제공하는 디자인 라이브러리가 때문에 세련된 디자인을 하기 쉽다.</p> <p>다른 프로그램에서 사용하던 단축키와 할당된 기능들을 파악해 비슷한 사용자 경험을 줄 수 있도록 함과 동시에 연관성이 있는 키로 지정한다.</p> <p>비슷한 단축키로 설정되면 사용법을 굳이 익히지 않아도 바로 사용할 수 있다는 장점이 있다.</p>
순번	실제로 만난 문제점
1	<p>모바일 환경이나 데스크톱 환경, 터치나 마우스 환경 등처럼 각 환경에 맞는 웹 디버깅 환경을 갖춘다. 그리고 이상 동작을 하는 환경의 이벤트 핸들링 방식을 디버깅해 어떤 부분이 달라 그렇게 작동하는지 파악한다.</p>
2	<p>텍스트의 길이를 계산하여 텍스트 박스의 크기에 맞게 줄을 바꿔주는 기능을 구현해야 했다. 텍스트의 길이를 일일이 계산하기는 쉽지만, 성능문제가 발생하여 Dynamic Programming 기법으로 한번 계산한 텍스트의 길이는 다시 계산하지 않도록 하여 최적화한다.</p>
3	<p>PaperJS에서 제공해주는 라이프 사이클 함수인 onFrame()를 이용해 실시간으로 미니맵을 갱신하도록 구성하였지만, 성능상의 문제가 발생하여 미니맵에 들어가는 데이터들을 래스터화 시킨 후 60FPS였던 갱신 주기를 낮게 잡았다.</p>
4	<p>PaperJS에서 도형 객체를 처리할 때 가로나 세로가 1이되면 그대로 고정되어버리는 문제였고 최소 크기를 5px로 주어 더 이상 작아질 수 없도록 제한했다.</p>
5	<p>브라우저 이벤트의 위치와 캔버스의 위치를 계산해주는 서비스를 등록해서 매번 계산하지 않아도 어디서나 접근 가능하도록 구성했다. 상호 변환이 가능하고 휠이나 터치조작을 통해서 줌배율을 변경하였을 때도 가중치를 줘서 더 많이 움직이거나 더 적게 움직일 수 있도록 조정했다.</p>
6	<p>몇몇의 이벤트를 무시해도 문제 없도록 객체가 정상인지 체크하는 로직과 같은 이벤트가 두 번 발생하더라도 문제가 생기지 않도록 구조를 조정했다.</p>
7	<p>이벤트 리스너의 등록법을 더 찾아보니 HTML5 자체적으로도 이벤트 리스너를 등록할 수 있고 돔 엘리먼트에 직접 등록할 수 있어서 컴포넌트에 종속적이지 않게 등록했다. 같은 이벤트에 리스너를 추가할 수도 있어서 리스너들을 기능별로 관리하는데에 더 편하도록 구조를 변경했다.</p>
8	<p>배경 그리드의 선을 인피니트 캔버스를 보조하는 셀들을 기준으로 그려 넣었지만 캔버스의 좌표를 기준으로 일정 수치마다 그리드 선을 그리도록 변경하여 해결하였다.</p>

### (3) 시험 시나리오

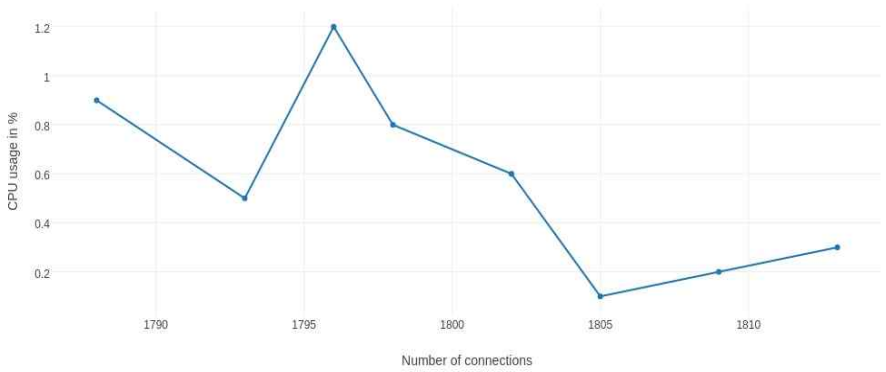
#### 1) 로드맵

□ 시스템 시험계획			
□ 소셜 로그인 기능		□ 화이트보드 기능	
○ 로그인 서비스	□ 성능 테스트 □ 사용성 테스트 □ 스트레스 테스트 □ 부하 테스트	○ 인피니트캔버스 기능	□ 성능 테스트
		○ 줌인/아웃 기능	□ 성능 테스트 □ 사용성 테스트
		○ 미니맵 기능	□ 성능 테스트 □ 스트레스 테스트
□ 그리기 도구		□ 파일 Import/Export	
○ 펜,형광펜,올가미,지우개	□ 사용성 테스트	○ Img, PPT, PDF Import기능	□ 사용성 테스트
○ 도형 그리기 기능		○ 프로젝트 Export기능	
○ 도형간의 간선 잇기 기능		○ Redo/Undo 기능	
□ 화이트보드 동기화 기능		□ 프로젝트 지원도구 동기화	
○ 커서트래킹 동기화 기능	□ 성능 테스트 □ 사용성 테스트 □ 스트레스 테스트 □ 부하 테스트	○ 칸반 동기화 기능	□ 성능 테스트 □ 사용성 테스트 □ 스트레스 테스트 □ 부하 테스트
○ 그리기도구 동기화 기능		○ 타임타이머 동기화 기능	
○ Redo/Undo 동기화 기능		○ 공유파일함 동기화 기능	
		○ VideoChat 지원	
	○ TextChat 동기화 기능		
□ 프로젝트 지원도구			
○ 칸반	□ 사용성 테스트		
○ 타임타이머			
○ 공유파일함			
○ TextChat			
○ VideoChat	□ 성능 테스트 □ 사용성 테스트 □ 스트레스 테스트 □ 부하 테스트		

## 2) 세부 테스트 계획

□ 성능, 부하, 스트레스 테스트 계획 - 웹앱		
사용 프로그램	프로그램 명	Chrome
	버전	79.0.3945.130
세부 계획	<ul style="list-style-type: none"> <li>○ 초당 60번 움직이는 원을 생성하여 강제로 부하 발생시킴.</li> <li>○ 이때 응용의 프레임 수치를 측정하여 프로그램이 느려지는지 테스트함.</li> <li>○ 추가적으로 크롬의 Performance Record기능을 사용하여 테스트 대상 구간을 측정함. 측정된 구간에 대한 함수 호출 스택을 세부 검사하여 느려지는 원인을 식별.</li> <li>○ 식별된 원인을 해결하기 위한 계획을 수립하고 핫픽스 브랜치를 생성하여 성능개선 실시.</li> </ul>	
실제사용  미니맵 기능 성능테스트		

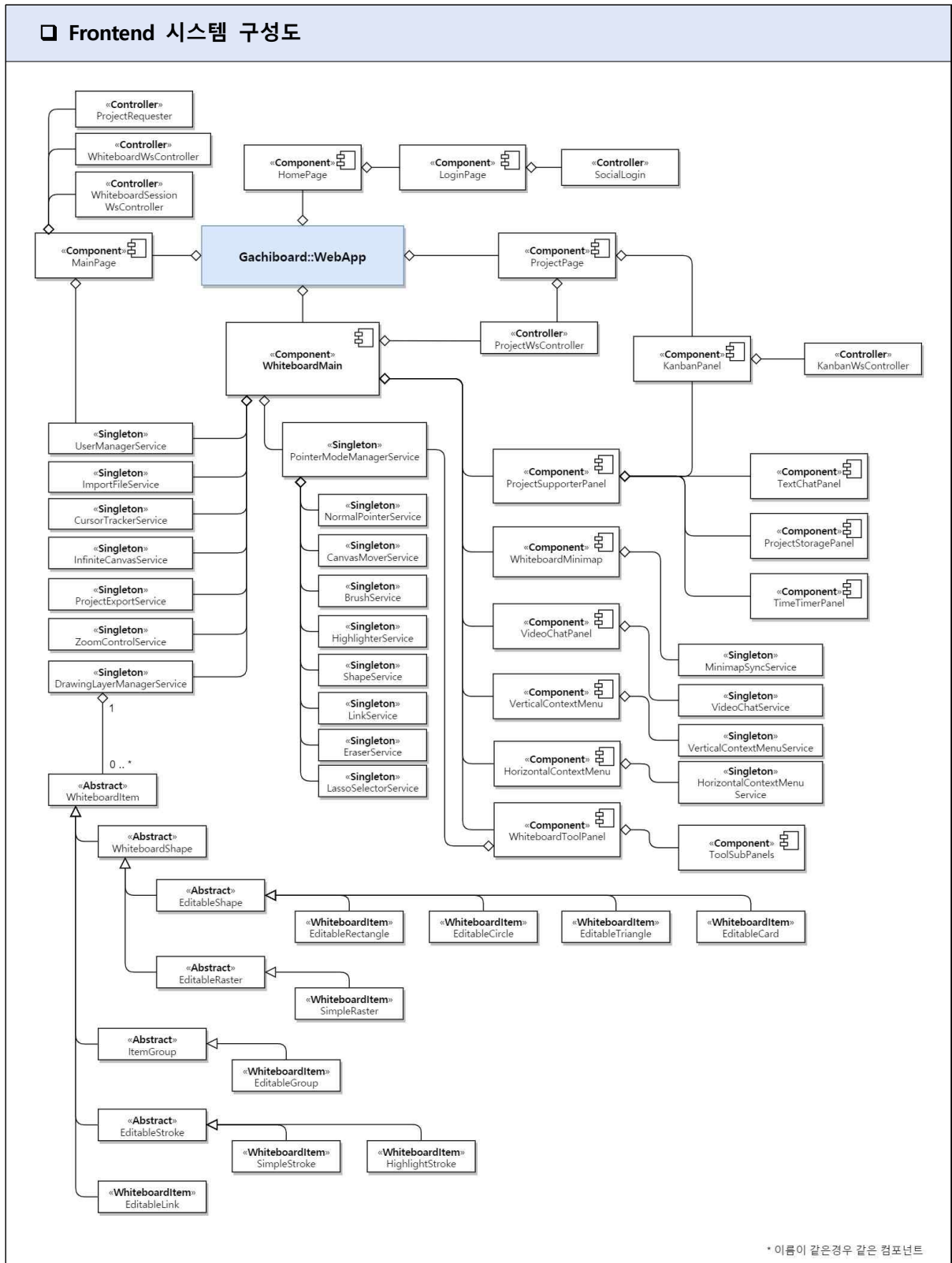


□ 성능, 부하, 스트레스 테스트 계획 - 백엔드 서버																
사용 프로그램	프로그램 명	Artillery.io														
	버전	1.6.0-29														
세부 계획	<ul style="list-style-type: none"><li>○ Artillery.io 모듈을 사용하여 Rest Api나 Socket.io를 요청하는 가상 테스트 환경을 구축.</li><li>○ 배포서버를 테스트 대상으로 지정한 상태에서 테스트 수행</li><li>○ 각 모듈마다 성능테스트를 위한 파일인 .yaml파일을 생성 및 관리</li><li>○ Artillery.io를 이용해 요청을 과도하게 발생시켜서 강제로 부하 발생시킴.</li><li>○ 부하 발생시 시스템의 상태를 점검하고, 느려지는 부분을 식별.</li><li>○ 원인을 분석하고, 성능개선 계획을 수립하고 수행함.</li></ul>															
예시	<div>CPU usage to maintain connections</div>  <table><thead><tr><th>Number of connections</th><th>CPU usage in %</th></tr></thead><tbody><tr><td>1790</td><td>0.9</td></tr><tr><td>1795</td><td>1.2</td></tr><tr><td>1800</td><td>0.8</td></tr><tr><td>1805</td><td>0.15</td></tr><tr><td>1810</td><td>0.2</td></tr><tr><td>1815</td><td>0.3</td></tr></tbody></table>		Number of connections	CPU usage in %	1790	0.9	1795	1.2	1800	0.8	1805	0.15	1810	0.2	1815	0.3
Number of connections	CPU usage in %															
1790	0.9															
1795	1.2															
1800	0.8															
1805	0.15															
1810	0.2															
1815	0.3															

□ 사용성 테스트		
사용 프로그램	프로그램 명	Google Forms
	버전	Google
세부 계획	<ul style="list-style-type: none"> <li>○ Google Forms를 통해 설문지 작성</li> <li>○ 설문지 내용을 취합하여 사용시 혼란스러운 요소가 있는 요소를 체크</li> <li>○ 체크된 요소의 원인을 분석하여 오류 수정 계획 수립</li> <li>○ 오류수정 후 사용성 테스트 재수행</li> </ul>	

## (4) 상세 설계

### 1) 웹앱 모듈 상세설계



□ WhiteboardMain Component		
기능	□ 화이트보드 기능의 메인 컴포넌트 <ul style="list-style-type: none"> <li>○ 각종 툴들로 그림을 그리거나 도형을 만드는 등의 화이트보드 전반적인 기능</li> <li>○ 프로젝트 도구로 칸반과 화상 채팅, 텍스트 채팅등에 접근 가능함</li> <li>○ 외부 문서를 가져오거나 화이트보드에 작성한 내용들을 외부로 추출할 수 있다.</li> </ul>	
변수	이름	설명
	<b>whiteboard PaperProject</b>	□ PaperJS에서 지원하는 Layer의 상위 객체. 여러개의 Layer를 등록할 수 있다. □ Project 객체가 필요한 여러 서비스들에게 할당해준다.
	<b>whiteboard PaperScope</b>	□ HTML5의 Canvas를 갖고 Project들을 관리하는 Paper의 최상위 객체 □ PaperJS의 전역 설정들을 제어할 수 있다.
	<b>connected UserList</b>	□ 현재 접속된 유저의 이름을 배열의 형태로 저장하는 변수 □ WhiteboardWSController에 의해 전달받은 SessionDto의 정보로 채워진다. □ 커서 트래커의 사용자 이름으로도 사용된다.
메서드	이름	설명
	<b>keyDown Handler()</b>	□ 키 다운 이벤트에 기능을 부여한다.
세부내용	□ 공통으로 사용되는 변수나 함수들과 라이프사이클들을 정의한 가장 상위의 WhiteboardItem 추상 클래스.	
	□ 직속 서브클래스로는 WhiteboardShape, EditableStroke, ItemGroup, EditableLink가 있고 그 하위에 EditableShape, SimpleStroke, EditableGroup 등이 존재한다.	

□ WhiteboardItem Abstract Class		
기능	□ 화이트보드의 모든 아이템들이 상속받게되는 추상클래스 □ 각종 기본 메서드들과 추상 메서드들의 정의 □ 라이프사이클 정의	
변수	이름	설명
	id	□ 모든 화이트보드 아이템을 구분하는 유니크한 ID □ DrawingLayer의 GetId 메서드를통해 서버에서 유니크한 ID를 할당받는다.
	type	□ 화이트보드 아이템의 타입을 지정 □ 이 값으로 각 화이트보드 아이템들의 타입을 구분할 수 있다.
	isOccupied	□ 해당 아이템이 다른 사용자에게 의해서 수정되어지고 있는지 체크할 수 있는 변수
	layerService	□ 화이트보드 아이템들이 DrawingLayer에 접근할때 사용하는 DrawingLayer에 대한 참조 변수
	Global, Local LifeCycle Emitter	□ 아이템 내부에서 라이프 사이클 이벤트를 발생하거나 모든 아이템들이 공유하는 라이프 사이클 이벤트를 발생시킬때 사용하는 이벤트 이미터에 대한 참조변수 □ 용도에 따라 다르게 이벤트를 발생시키고 필요한 곳에서 구독하여 사용한다.
메서드	이름	설명
	abstract exportToDto()	□ 해당하는 화이트보드 아이템의 DTO를 추출하는 메서드. □ 각 아이템 타입에 맞는 DTO로 추출될 수 있도록 추상메서드로 제공
	abstract update()	□ 해당하는 화이트보드 아이템의 DTO를 받아서 해당객체를 업데이트 하는 메서드. □ 마찬가지로 추상메서드로 제공하여 각 아이템 타입에 맞게 업데이트 할 수 있도록 메서드 제공
세부내용	□ 공통으로 사용되는 변수나 함수들과 라이프사이클들을 정의한 가장 상위의 WhiteboardItem 추상 클래스.	
	□ 직속 서브클래스로는 WhiteboardShape, EditableStroke, ItemGroup, EditableLink가 있고 그 하위에 EditableShape, SimpleStroke, EditableGroup 등이 존재한다.	

□ WhiteboardShape Abstract Class		
기능	□ EditableShape와 EditableRaster가 상속받는 슈퍼 클래스 □ 링크 포트 객체에 의해서 링크에 연결될 수 있다.	
변수	이름	설명
	linkPortMap	□ 링크포트 객체를 저장하는 Map 자료구조 □ key에는 링크 포트의 위치를 지정하는 Enum값 이며 value에는 링크포트 객체가 할당된다.
		□ 기본적으로 TOP, BOTTOM, LEFT, RIGHT 네가지의 링크포트가 할당된다.
메서드	이름	설명
	getClosest LinkPort()	□ 현재 좌표에서 가장 가까운 링크포트의 위치를 반환한다. □ 링크를 수정하는 핸들을 잡고 도형에 가져다 대면 자석처럼 가장 가까운 링크포트에 연결할 수 있게 도와준다.
세부내용	□ 공통으로 사용되는 변수나 함수들과 라이프사이클들을 정의한 가장 상위의 WhiteboardItem 추상 클래스.	
	□ 직속 서브클래스로는 WhiteboardShape, EditableStroke, ItemGroup, EditableLink가 있고 그 하위에 EditableShape, SimpleStroke, EditableGroup 등이 존재한다.	

□ DrawingLayerManager Service		
기능	□ 모든 화이트보드 아이템들을 관리하고 PaperJS의 Layer를 관리하는 서비스. □ 아이템들을 순회하거나 특정 아이템을 찾거나 ID를 통해 아이템을 가져올 수 있는 기능등을 담당	
변수	이름	설명
	drawing Layer	PaperJS의 Layer에 대한 참조 변수 PaperJS를 통해서 그리는 아이템들을 Layer 단위로 담고있다. 이 참조 변수를 통해 모든 PaperJS의 아이템에 접근 가능하다.
	current Project	PaperJS의 Project에 대한 참조 변수 Layer의 상위 객체이고 여러 Layer를 가질 수 있다. DrawingLayer에서 현재 Project에 접근할때 사용한다.
	global Selected Group	화이트 보드 아이템들을 선택했을때 선택된 아이템들이 위치하는 ItemGroup의 서브클래스를 갖는 변수 현재 선택된 아이템들에 접근할때 사용된다.
	whiteboard ItemArray	DrawingLayer에 화이트보드 아이템들이 추가되는 배열 화이트보드 아이템들을 찾을때 이 배열을 순회하며 찾는다.
메서드	이름	설명
	addTo DrawingLayer()	□ PaperJS의 아이템과 타입을 지정해 호출하면 화이트보드 아이템을 생성하여 반환하는 메서드. □ 임시 ID를 지정한다.
	findIn Whiteboard Items()	□ 해당 좌표에 있는 화이트보드 아이템중 가장 앞에 있는 아이템을 찾아주는 메서드 □ 해당 좌표에 아이템이 없다면 null을 반환한다.
	waitRequest Create WblItem()	□ 새로 생성될 화이트보드의 정보를 서버에 전송해서 유효한지 체크 후 ID를 할당받는 메서드
세부내용	□ addToDrawingLayer 메서드를 통해 일반 아이템을 화이트보드 아이템으로 생성한다. 이때 LifeCycle의 CREATE를 통해서 whiteboardItemArray에 생성된 화이트보드 아이템이 추가된다.	
	□ 화이트보드 아이템을 찾기위한 여러가지 메서드들을 제공하고 있으며 대부분 특정 좌표에 존재하는 아이템을 찾기 위한 메서드임.	

□ WhiteboardWs Controller		
기능	□ 백엔드 서버의 Whiteboard WebSocketGateway에 요청하거나, 해당 게이트웨이로부터 오는 메시지를 받는 WebSocket 컨트롤러 □ 싱글턴 클래스임.	
변수	이름	설명
	websocket Manager	□ 웹앱의 모든 WebSocket Controller들이 가지고 있는 싱글턴 클래스의 객체. □ 서버로부터 메시지가 오는 경우, 해당 객체 안에 있는 EventEmitter를 사용하여 이벤트를 발생시킴.
	socket	□ 백엔드 서버로 메시지를 보내거나 받기 위해 존재하는 소켓객체
	instance	□ WhiteboardWsController의 객체를 담고 있는 변수.
메서드	이름	설명
	waitRequestCreateWblItem	□ WblItem생성요청메세지를 전송함.
	onWblItemCreated	□ WblItem생성요청메세지를 수신함.
	waitRequestUpdateWblItem	□ WblItem수정요청메세지를 전송함.
	onWblItemUpdated	□ WblItem수정요청메세지를 수신함.
	waitRequestDeleteWblItem	□ WblItem삭제요청메세지를 전송함.
	onWblItemDeleted	□ WblItem삭제요청메세지를 수신함.
	waitRequestOccupyWblItem	□ WblItem선점요청메세지를 전송함.
	onWblItemOccupied	□ WblItem선점요청메세지를 수신함.
	waitRequestNotOccupyWblItem	□ WblItem선점해제요청메세지를 전송함.
	onWblItemNotOccupied	□ WblItem선점해제요청메세지를 수신함.
	getInstance	□ 해당 싱글턴 클래스의 객체를 리턴함.
	initInstance	□ 해당 싱글턴 클래스의 객체를 초기화함.
세부내용	□ 해당 클래스의 객체는 초기화 작업이 필요한데, WebSocketManagerService에서 초기화해준다.	
	□ 해당 클래스의 메서드를 이용하는 경우, getInstance메서드로 객체를 획득하고 사용하면 된다.	

□ WblItemFactory Class		
기능	□ 주요 기능은 DTO를 사용한 아이템 복제 □ 사용자의 직접 지정 없이 여러 아이템들을 한번에 제작할 수 있는 기능을 담당	
변수	이름	설명
	buildMode	□ 아이템을 복제할지 생성할지 결정한다.
메서드	이름	설명
	clone WblItems()	□ 복사된 DTO 배열을 받아 화이트보드 아이템과 에디터블 링크로 나눠 아이템 복제를 시작하는 메서드다. □ 각각의 아이템들마다 복사가 완료되는 시점이 다르기 때문에 Observable 모듈을 사용해 모든 아이템들이 복제가 다 될때까지 기다린다. □ 복제가 완료되면 링크를 복제한다. 링크가 가장 마지막에 복제되는 이유는 링크가 대상과 연결될때 대상이 존재하지 않으면 연결될 수 없기 때문이다.
	create WblItem()	□ 전달받은 화이트보드 아이템 DTO와 해당 타입의 화이트보드 아이템 생성자를 사용해 화이트보드 아이템을 제작하는 메서드. □ 빌드 모드에 따라 DTO의 속성과 완전히 같은 아이템(동기화)을 만들거나 ID만 다른 새로운 아이템(복제)을 만들수 있다.
	create EditableLink()	□ 전달받은 화이트보드 아이템 DTO와 EditableLink의 생성자를 사용해 링크를 제작한다. □ DTO에 저장된 LinkPort의 ownerID에 의해서 링크와 연결될 화이트보드 아이템이 무엇인지 결정된다.
세부내용	□ 동기화기능과 복사 및 붙여넣기 기능의 핵심중 하나인 아이템 복제를 담당하는 클래스	
	□ 동기화의 경우엔 다른 사용자가 생성한 아이템을 동기화하는 시퀀스에서 사용되고 붙여넣기의 경우엔 아이템을 복제하는 시퀀스에서 사용된다.	
	□ 각 시퀀스에 요구되는 ID가 다르기 때문에 buildMode를 사용해 구분한다.	



□ WhiteboardItemDto Class		
기능	□ 서버와 통신에 사용되는 Data Transfer Object	
변수	이름	설명
	id	□ 화이트보드 아이템의 ID
	type	□ 어떤 화이트보드 아이템인지 나타내는 Type
	center	□ 화이트보드 아이템이 위치하는 아이템의 중앙 좌표
	isGrouped	□ 그룹이 있는지 여부
	parentEdt GroupId	□ 그룹이 있으며 그 그룹의 상위그룹의 ID가 할당됨
세부내용	□ 서버에서 주고받기 위한 오브젝트 이므로 데이터는 최소화 되어있다.	
	□ 특별한 메서드 없이 작성되었고 다른 화이트보드 아이템들은 이 클래스를 상속받아 단계적으로 확장된다.	

PointerModeManager Service		
기능	□ 포인터의 모드를 관리하는 서비스 □ Brush, Eraser, Highlighter, Shape 등의 각종 포인터들을 관리하고 변경해준다.	
변수	이름	설명
	current PointerMode	□ 현재 어떤 포인터가 선택되어 활성화 되어있는지를 나타내는 변수 □ 포인터마다 할당된 EnumData를 갖는다.
	isThrottle	□ touch나 mouse 이벤트를 일정 간격마다 받을 수 있도록 쓰로틀링을 걸어주는 플래그 변수 □ true일때 이벤트를 스킵하고 false일때 이벤트를 받아들인다
	prevPoint	□ 이벤트 사이에서 마우스가 움직인 거리를 벡터로 구하기 위해 이전 위치를 저장하는 변수 □ 매 이벤트의 처리가 끝날때 이벤트의 위치로 초기화된다.
	current Project	□ PaperJS의 Project를 담는 참조 변수 □ Project의 view에 마우스나 터치 이벤트를 할당할 때 사용한다.
	brush, eraser, shape, ETC	□ 각각 이름에 맞는 포인터 서비스를 갖는 변수 □ currentPointerMode에 맞는 서비스에서 기능을 가져다 쓴다.
메서드	이름	설명
	initDelta()	□ 마우스 및 터치이벤트의 이벤트 발생 좌표와 이전 이벤트에서 저장한 좌표로 움직인 거리와 방향을 구한다.
세부내용	□ 각종 포인터 서비스들을 주입받아 포인터간 전환을 해주는 서비스	
	□ 사용자가 도구 패널의 특정 포인터의 버튼을 누를 때 모드가 변경되고 모드에 따라서 해당 포인터를 사용하도록 전환해주는 역할을 갖는다.	

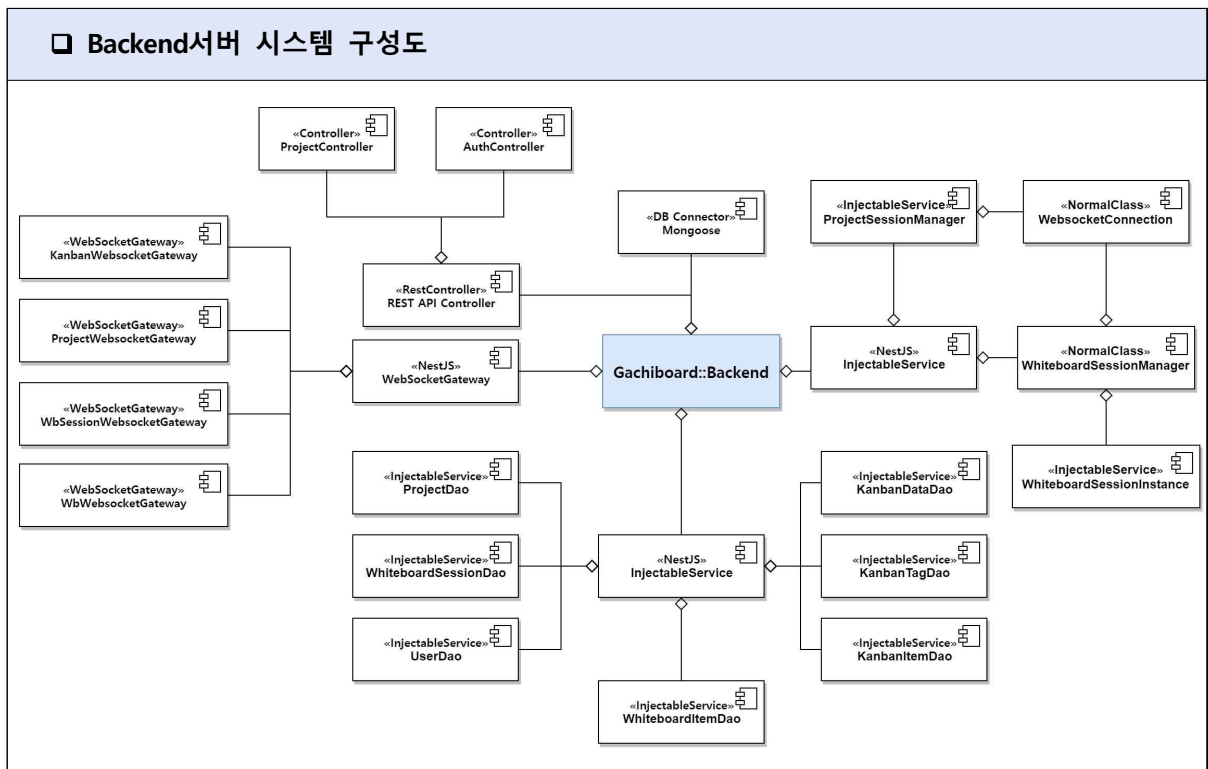
PositionCalc Service		
기능	□ 브라우저의 좌표와 InfiniteCanvas의 좌표 사이의 변환	
메서드	이름	설명
	reflectZoomWithPoint() / restoreZoomWithPoint()	□ 현재 좌표에 줌 수치를 반영하거나 되돌리는 메서드
	ngPointToCanvas() / canvasPointToNg()	□ 현재 좌표를 Canvas의 좌표로 변환하거나 되돌리는 메서드
세부내용	□ HTML5의 Canvas를 이용해 구현될 서비스로서 브라우저의 좌표와 무제한 캔버스를 구현한 Canvas의 내부 좌표가 다르기 때문에 상호 변환을 위한 서비스	
	□ Canvas를 이동하거나 줌인을 하거나 줌 아웃을 함에 따라서 좌표를 계산해서 반환해준다.	

NormalPointer Service		
기능	<ul style="list-style-type: none"> <li>아이템을 선택하거나 화면을 이동하거나 핸들을 잡고 아이템을 수정하거나 등의 대부분의 일반적인 포인터의 역할을 한다.</li> <li>조합키(Ctrl, Shift, Alt)를 눌렀을 때 정사각형 모양으로 아이템을 수정할 수 있는 등의 모드로 전환이 된다.</li> </ul>	
변수	이름	설명
	action	<ul style="list-style-type: none"> <li>터치나 마우스의 down 이벤트에 따라서 다음 액션이 어떻게 될지 정해진다. 해당 액션의 Enum값이 할당되는 변수이다.</li> </ul>
	selected Handle	<ul style="list-style-type: none"> <li>down 이벤트가 Handle의 위에서 일어났을때 해당 핸들을 이용해 아이템을 수정할 수 있도록 핸들의 객체를 담아두는 변수이다.</li> <li>move 이벤트가 발생할때 이벤트를 핸들에게 위임할때 사용된다.</li> </ul>
메서드	이름	설명
	onMouse Down()	<ul style="list-style-type: none"> <li>마우스의 클릭이 시작될때 발생하는 이벤트</li> <li>현재 선택된 아이템이 있는지 체크하고 없으면 아이템 선택을 시도한다.</li> <li>선택된 아이템이 있다면 세가지로 나뉘지는데 또 다른 아이템을 선택했을 경우 다른 아이템을 선택하거나 조합키와 함께 선택되어 다중 선택 모드에 들어갈지 결정한다.</li> <li>만약 선택된 아이템을 다시 클릭한 경우엔 action을 아이템 드래그 모드로 변경하고 빈공간을 클릭한 경우엔 모든 선택된 아이템을 선택 취소한다.</li> </ul>
	onMouse Move()	<ul style="list-style-type: none"> <li>아이템이 선택된게 없으면 캔버스를 움직이고 선택된 아이템이 있다면 아이템을 움직인다</li> <li>이전 단계에서 이벤트가 아이템 위가 아니라면 현재 선택된 아이템이 없으므로 아이템위에서 down이벤트가 발생한 경우에만 아이템을 움직일 수 있다.</li> </ul>
	onMouse Up()	<ul style="list-style-type: none"> <li>이전 동작이 캔버스를 움직인게 아니라면 각 아이템 또는 핸들의 마무리 작업을 한다.</li> </ul>
세부내용	<ul style="list-style-type: none"> <li>아이템을 선택한 경우, 선택하지 않은 경우, 핸들을 선택한 경우, 아이템을 추가로 선택한 경우, 아이템을 이동한 경우 등의 여러가지 조건들에 의해 분기가 많이 존재한다.</li> </ul>	
	<ul style="list-style-type: none"> <li>이러한 분기에 의해 아이템을 움직이거나 캔버스를 움직이는 등의 행동을 마우스나 터치로 할 수 있도록 도와주는 서비스다.</li> </ul>	
	<ul style="list-style-type: none"> <li>다른 포인터 서비스들도 동일하게 시작, 드래그, 끝의 세가지 절차로 나뉘어져 있고 각 이벤트에 해당 메서드들을 할당하는 방식으로 사용할 수 있다.</li> </ul>	

## 2) 데이터베이스 컬렉션

□ 데이터베이스 컬렉션		
* 중요한 데이터만 추려서 간략하게 설명하였음		
컬렉션 이름	변수명	세부설명
□ users_models	idToken	사용자를 식별할 수 있는 토큰
	accessToken	접속 토큰
	userName	사용자의 이름
	regDate	같이보드 서비스 가입일자
□ project_models	projectTitle	프로젝트 이름
	inviteCodeList	프로젝트 초대코드 목록.
	participantList	프로젝트 참여자 목록
	wbSessionList	프로젝트에 포함된 화이트보드 목록
	kanbanData	프로젝트의 칸반데이터의 Object_ID
□ whiteboard_session_models	title	화이트보드 이름
	createdBy	화이트보드를 생성한 사람의 IdToken
	startDate	화이트보드 세션 생성일
	wblItemArray	화이트보드 아이템의 Object_ID목록
□ whiteboard_item_packet_models	createdBy	아이템을 생성한 사람의 IdToken
	createDate	아이템 생성일
	lastModifier	최근수정한 사람의 IdToken
	modifiedDate	최근 수정일
	version	아이템의 버전. 수정할때마다 1씩 증가함.
	touchHistory	아이템 수정내역
	wblItemDto	아이템의 정보를 담고있는 Object
□ kanban_data_models	todoGroup	todo그룹에 포함된 칸반아이템 목록
	inProgressGroup	inProgress그룹에 포함된 칸반아이템 목록
	doneGroup	done그룹에 포함된 칸반아이템 목록
	kanbanTagListDto	해당 프로젝트의 모든 칸반태그 목록
□ kanban_item_models	title	해당 칸반의 이름
	userInfo	해당 칸반의 담당자의 IdToken
	color	해당 칸반의 색깔
	tagIdList	해당 칸반에 포함된 태그의 Object_ID목록
□ kanban_tag_models	title	태그 이름

### 3) 백엔드 서버 모듈 상세설계



AuthController		
기능	<ul style="list-style-type: none"> <li>□ 사용자의 OAuth2.0 로그인을 처리하는 컨트롤러 클래스</li> <li>□ 사용자에게 소셜로그인페이지를 응답하는 기능을 담당</li> <li>□ 소셜로그인 성공시, 소셜로그인서비스로부터 AuthToken과 함께 사용자정보를 받는 콜백컨트롤러 기능을 담당</li> </ul>	
컨트롤러 정보	uri	설명
	auth/google	□ 소셜로그인 페이지를 사용자에게 응답해주는 컨트롤러
	auth/google/callback	<ul style="list-style-type: none"> <li>□ 소셜 로그인 서버로부터 결과를 받는 콜백 컨트롤러</li> <li>□ 사용자에게 로그인 결과와 서버에서 만든 AccessToken을 응답해줌</li> </ul>
	auth/protected	□ AccessToken이 유효한지 검사하고 사용자에게 응답해주는 컨트롤러

□ GoogleStrategy		
기능	□ 구글 소셜로그인의 유효성 검사를 담당하는 싱글턴 클래스	
상속	□ PassportStrategyGoogle클래스를 상속받음	
변수	이름	설명
	ClientID	□ 구글 소셜로그인 서비스의 클라이언트 ID
	clientSecret	□ 구글 소셜로그인 서비스의 클라이언트 Secret
	callbackURL	□ 구글 소셜로그인 서비스의 콜백 주소
	scope	□ 사용자로부터 요구하는 권한 목록 ○ email, profile, openId 정보만 요청함
메서드	이름	설명
	validate	□ 로그인 성공시, validate메서드는 JWT형식의 AccessToken을 생성한다. □ 생성한 AccessToken과 소셜로그인서버에서 제공한 사용자 정보를 가지고 UserDto를 생성하고 반환한다.
세부내용	□ AuthController는 해당 클래스의 validate메서드를 이용함. 소셜로그인 성공시, 콜백 컨트롤러로 결과가 반환되는데, 그 콜백 컨트롤러가 수행되기 전에 GoogleStrategy의 validate메서드가 호출되어 결과를 검증하게 된다.	

□ AuthService		
기능	□ 소셜로그인서버에서 보내준 IdToken과 서비스 프로바이더 정보를 가지고 JWT 토큰을 만들어줌. □ JWT토큰을 검증하는 기능을 수행	
변수	이름	설명
	JWT_SECRET_KEY	□ JWT토큰을 생성하는데 사용하는 비밀 키
메서드	이름	설명
	validateOAuthLogin	□ OAuth로그인 결과를 검증하고, 성공시 JWT형식의 AccessToken을 생성하여 리턴하는 기능 수행.
	verifyTokenClaims	□ AccessToken을 검증하고 그 결과를 리턴함. □ 사용자가 요청하면서 실어보낸 Payload정보와 JWT_SECRET_KEY를 가지고 JWT Token을 만들고, 이 토큰정보와 DB에 저장된 사용자의 AccessToken이 같은지 검사하고, 그 결과를 반환함
세부내용	□ 컨트롤러에 @AuthGuard("JWT")어노테이션을 작성하면 JwtStrategy클래스의 validate메서드가 호출되는데, 이 메서드에서 veifyTokenClaims 메서드를 이용해서 토큰검증을 수행함.	

□ UserDao		
기능	□ 사용자정보를 DB에 저장하는 DAO클래스	
변수	이름	설명
	usersModel	□ DB에 접근하여 생성,수정,삭제,검색을 하는데 쓰이는 MongooseModel객체를 가지고 있는 변수
메서드	이름	설명
	create	□ 컬렉션에 Dto정보를 저장함.
	findAll	□ 컬렉션에 있는 모든 Document를 찾고 리턴.
	findOne	□ 컬렉션에서 id로 Document를 찾고 리턴.
	update	□ 컬렉션에서 id로 Document를 찾고 DTO 정보로 수정함
	deleteOne	□ 컬렉션에서 id로 Document를 찾고 삭제

□ WbSession-Websocket-Gateway		
기능	□ 화이트보드세션과 관련된 Websocket 이벤트를 처리하는 게이트웨이 클래스	
변수	이름	설명
	server	□ SocketIO Server인스턴스임. □ 해당 변수를 이용해서 특정 네임스페이스나 Room에 메시지를 전송할 수 있음
메서드	이름	설명
	handleDisconnection	□ 클라이언트와 연결해제시 호출되는 콜백메서드 □ 해당 콜백을 이용해서 wbSession커넥션 풀에서 연결해제된 소켓정보를 제거함.
	onWbSessionJoinRequest	□ 사용자의 화이트보드 세션 참가요청을 처리하는 콜백메서드. □ wbSession커넥션 풀에 연결정보를 추가함.
	onWbSessionCreateRequest	□ wbSession 생성요청을 처리하는 콜백메서드
	onWbSessionUpdateRequest	□ wbSession 수정요청을 처리하는 콜백메서드
	onWbSessionDeleteRequest	□ wbSession 삭제요청을 처리하는 콜백메서드
	responseAckPacket	□ 요청자에게는 성공메세지를 보내고, 다른 참여자들에게는 성공,수정,삭제등의 메시지를 브로드캐스트하는 메서드.

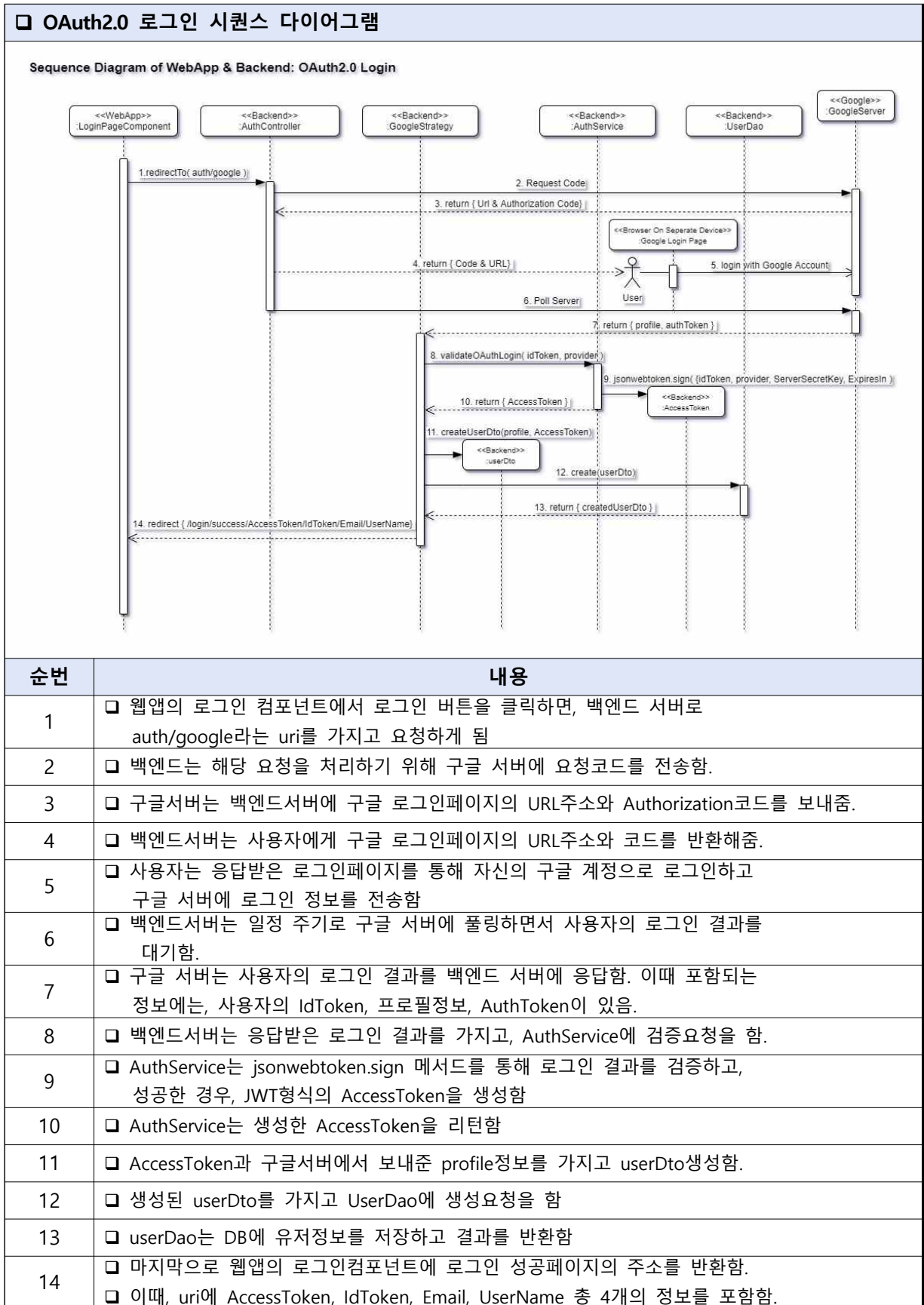
□ Wb-Websocket-Gateway		
기능	□ 화이트보드 아이템과 관련된 Websocket 이벤트를 처리하는 게이트웨이 클래스	
변수	이름	설명
	server	□ SocketIO Server인스턴스임. □ 해당 변수를 이용해서 특정 네임스페이스나 Room에 메시지를 전송할 수 있음
메서드	이름	설명
	onWblItemCreateRequest	□ WblItem 생성요청을 처리하는 콜백메서드
	onWblItemUpdateRequest	□ WblItem 수정요청을 처리하는 콜백메서드
	onWblItemDeleteRequest	□ WblItem 삭제요청을 처리하는 콜백메서드
	onWblItemOccupiedRequest	□ WblItem 선점요청을 처리하는 콜백메서드
	onWblItemNotOccupiedRequest	□ 선점한 WblItem에 대한 해제요청을 처리하는 콜백메서드
	responseAckPacket	□ 요청자에게는 성공메세지를 보내고, 다른 참여자들에게는 성공,수정,삭제등의 메시지를 브로드캐스트하는 메서드.

□ WbSessionDao		
기능	□ 화이트보드 세션정보를 DB에 저장하는 DAO클래스	
변수	이름	설명
	wbSessionModel	□ DB에 접근하여 생성,수정,삭제,검색을 하는데 쓰이는 MongooseModel객체를 가지고 있는 변수
메서드	이름	설명
	create	□ 컬렉션에 Dto정보를 저장함.
	findAll	□ 컬렉션에 있는 모든 Document를 찾고 리턴.
	findOne	□ 컬렉션에서 id로 Document를 찾고 리턴.
	update	□ 컬렉션에서 id로 Document를 찾고 DTO 정보로 수정함
	deleteOne	□ 컬렉션에서 id로 Document를 찾고 삭제
	verifyRequest	□ idToken과 wbSessionId를 가지고 userDto와 wbSessionDto를 찾아서 해당 요청이 유효한지 검사하는 메서드.

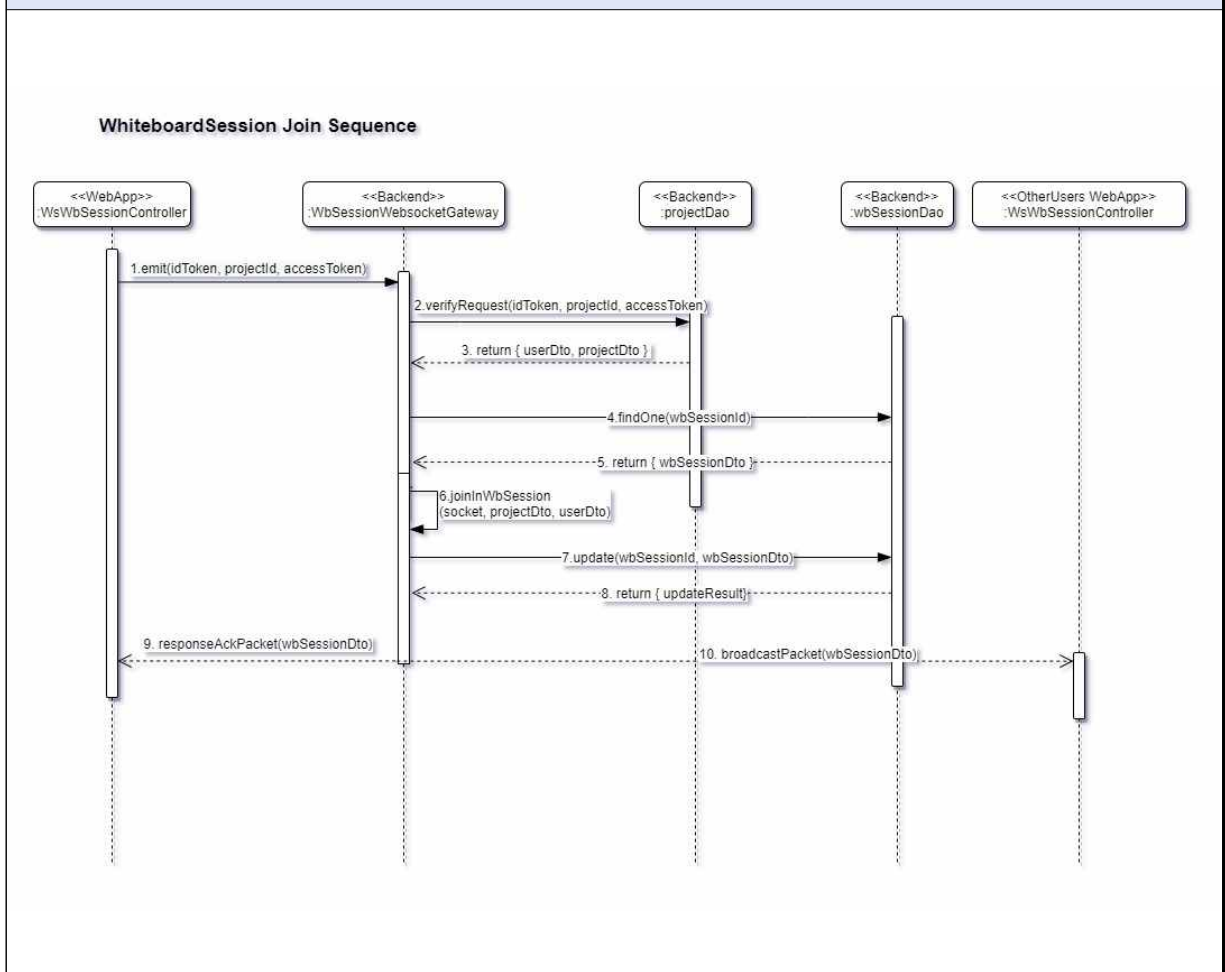


□ WblItemPacketDao		
기능	□ 화이트보드 아이템 정보를 DB에 저장하는 DAO클래스	
변수	이름	설명
	wblItemPacketModel	□ DB에 접근하여 생성,수정,삭제,검색을 하는데 쓰이는 MongooseModel객체를 가지고 있는 변수
메서드	이름	설명
	create	□ 컬렉션에 Dto정보를 저장함.
	findAll	□ 컬렉션에 있는 모든 Document를 찾고 리턴.
	findOne	□ 컬렉션에서 id로 Document를 찾고 리턴.
	update	□ 컬렉션에서 id로 Document를 찾고 DTO 정보로 수정함
	deleteOne	□ 컬렉션에서 id로 Document를 찾고 삭제

#### 4) 시퀀스 다이어그램



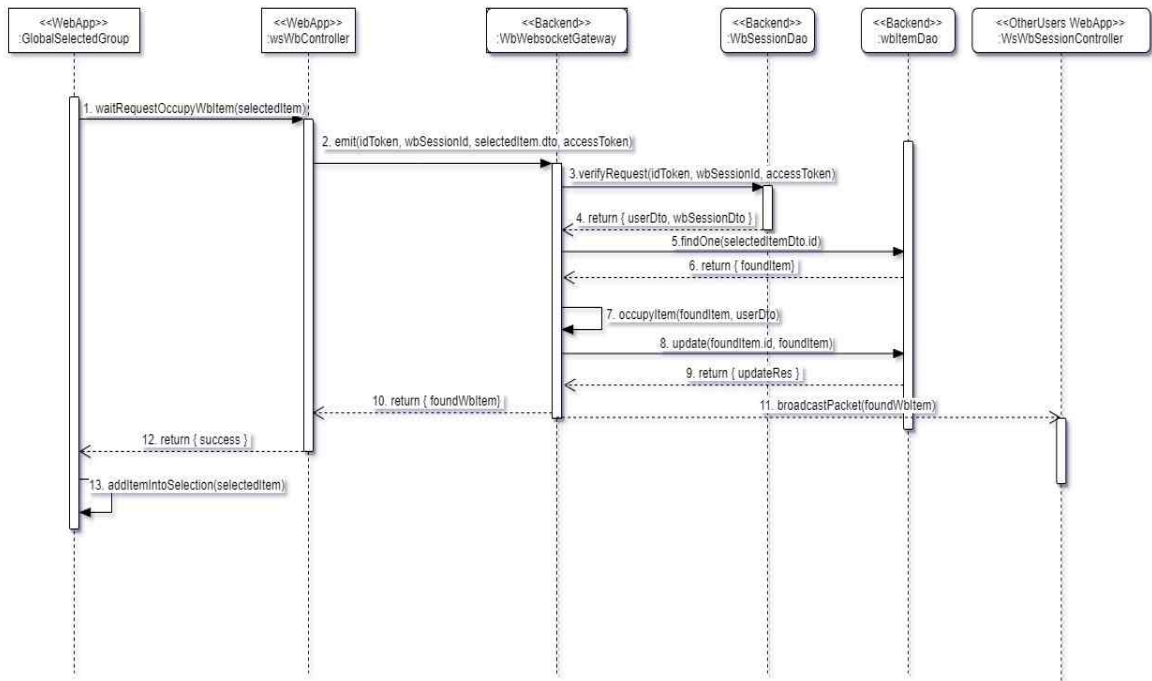
□ 화이트보드 세션 Join 시퀀스 다이어그램



순번	내용
1	□ WsWbSessionController에서 socket에게 JOIN 이벤트를 발생시키면서 idToken과 projectId를 함께 전달한다.
2	□ WbSessionWebsocketGateway의 JOIN 이벤트를 처리하는 메서드가 projectDao의 verifyRequest를 통해 ID 토큰과 프로젝트 ID, 액세스 토큰의 유효성 검사 및 project 정보와 user 정보를 요청한다.
3	□ 유효한 idToken과 projectId인 경우 userDto와 projectDto를 반환한다.
4	□ WbSessionWebsocketGateway가 wbSessionId로 wbSessionDao의 findOne 메서드를 호출해 Id에 해당하는 화이트보드 세션 DTO를 요청한다.
5	□ DB에서 wbSessionId에 맞는 세션정보를 찾아 DTO로 제작해 반환한다.
6	□ 찾은 화이트보드 세션에 socket을 Join한다.
7	□ 과정 5에서 받은 SessionDto의 ID와 DTO를 파라미터로 wbSessionDao의 update메서드를 호출한다.
8	□ 업데이트 결과를 전달 한다.
9	□ 최초 요청에 대한 wbSessionDto를 미리 정의한 AckPacket의 형태로 전달한다.
10	□ 요청한 사용자 외의 같은 세션에 존재하는 다른 사용자들에게 전달하기 위해 같은 wbSessionDto를 broadcastPacket 메서드를 통해 브로드캐스팅 한다.

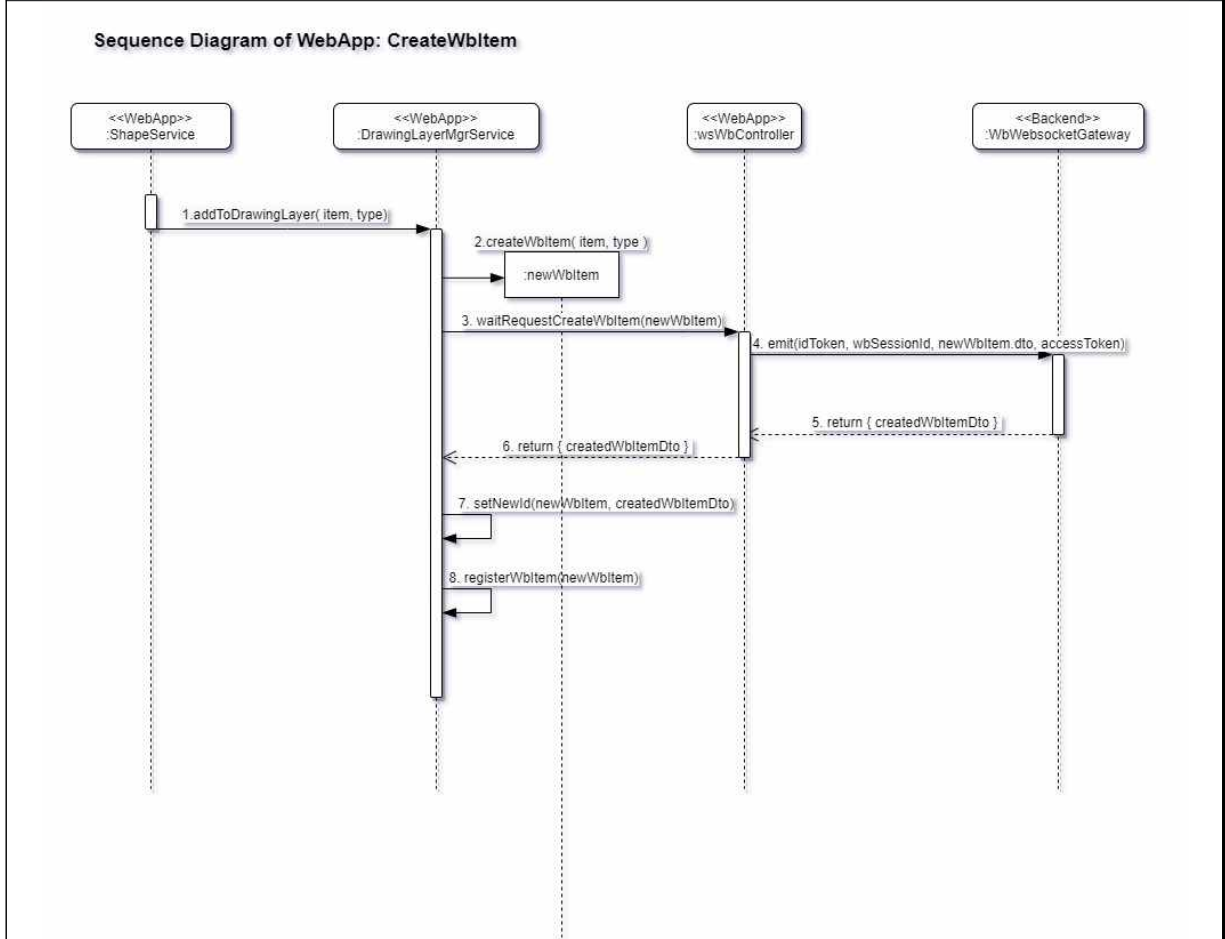
## □ WblItem 선택요청 시퀀스 다이어그램

Sequence Diagram of WebApp & Backend: Occupy Item



순번	내용
1	□ 웹앱의 WblItem선택도구인 globalSelectedGroup으로 아이템을 선택할때마다, 선택한 아이템을 선택하기 위해 wsWbConntroller에 선택메세지 전송을 요청.
2	□ wsWbController는 wbWebsocketGateway에 선택요청 메시지를 전송.
3	□ WbWebsocketGateway는 메시지를 수신하고, 이 메시지가 유효한지 검증하기 위해 WbSessionDao의 verifyRequest메서드를 호출.
4	□ wbSessionDao의 verifyRequest메서드는 패러미터로 받은 idToken, wbSessionId를 가지고 DB에서 UserDao, WbSessionDto를 찾음. □ UserDao의 AccessToken과, 요청메세지에 있던 AccessToken을 비교하여 사용자 검증을 수행하고, wbSessionId에 해당하는 Document가 DB에 있는지 확인하고, 결과를 반환함. □ 만약 검증을 실패하게 되면 reject메서드를 호출하여 시퀀스를 중단하고 사용자에게 NakPacket을 전송함.
5	□ 선택요청한 아이템을 WblItemDao를 통해 찾음.
6	□ WblItemDao는 찾은 Document를 반환.
7	□ 찾은 Document에 대해, 요청자를 해당 아이템의 선택자로 등록.
8	□ 수정된 정보를 WblItemDao를 통해 DB에 저장.
9	□ WblItemDao는 수정 결과를 반환.
10	□ WbWebsocketGateway는 요청한 웹앱에 수정된 WblItemDto를 전송.
11	□ 해당 화이트보드 세션에 접속해 있는 다른 유저들에게도 WblItemDto를 전송.
12	□ wsWbController는 GlobalSelectedGroup에게 성공이벤트를 전송
13	□ GlobalSelectedGroup는 선택그룹에 아이템을 넣는 작업을 완료함.

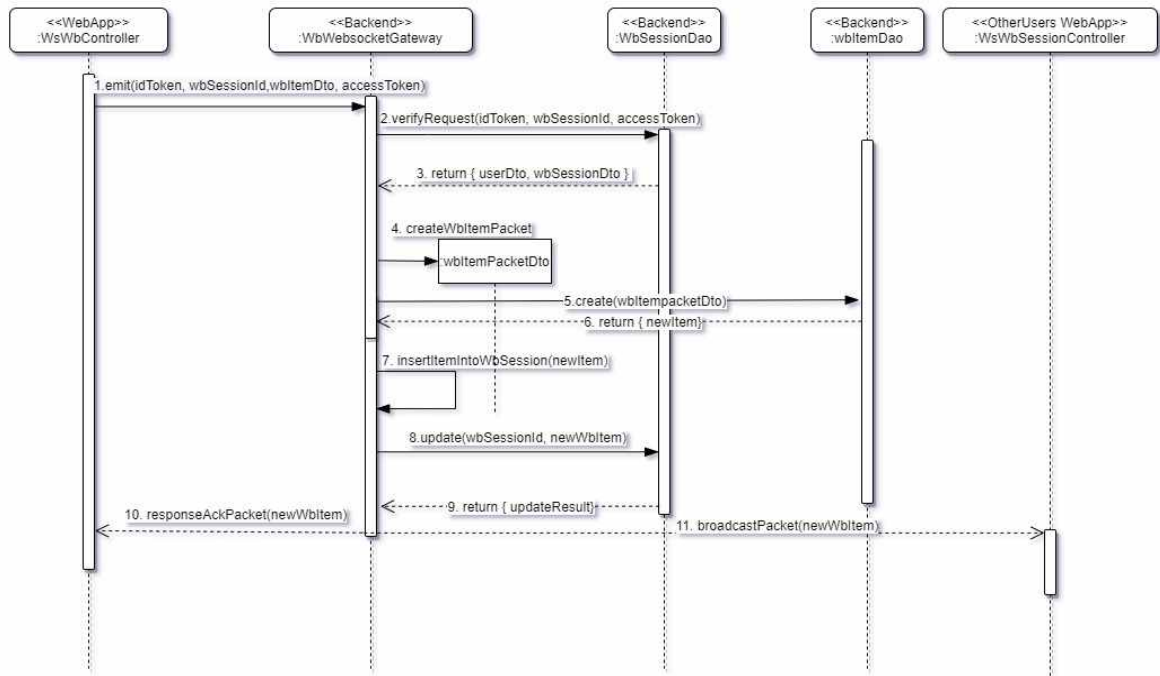
□ WblItem 생성요청 웹앱 파트 시퀀스 다이어그램



순번	내용
1	□ 도형 서비스에 의해 생성된 아이템을 DrawingLayerService의 addToDrawingLayer를 호출한다.
2	□ DrawingLayerService는 전달받은 item과 type을 토대로 멤버 메서드인 createWblItem을 호출해 화이트보드 아이템 객체를 생성한다. 이때 화이트보드 아이템의 ID는 임시로 부여된다.
3	□ 이렇게 생성된 화이트보드 아이템을 DTO로 추출하여 wsWbController의 waitRequestCreateWblItem을 호출한다.
4	□ 전달받은 DTO로 소켓 통신에 사용될 packet을 구성하고 socket의 emit 메서드를 통해 백엔드의 WbWebsocketGateway에게 화이트보드 아이템 생성처리를 요청한다.
5	□ 백엔드가 전달받은 DTO에 유효한 ID를 부여하여 다시 웹앱의 wsWbController에게 emit 메서드를 사용해 전송한다.
6	□ 백엔드에게서 받은 최신화 된 DTO를 그대로 반환한다.
7	□ 유효한 ID라고 할당받은 새로운 ID를 임시로 부여했던 화이트보드 아이템에 적용한다.
8	□ 화이트보드 아이템을 DrawingLayer에 등록한다.

□ Wbltem 생성요청 백엔드 파트 시퀀스 다이어그램

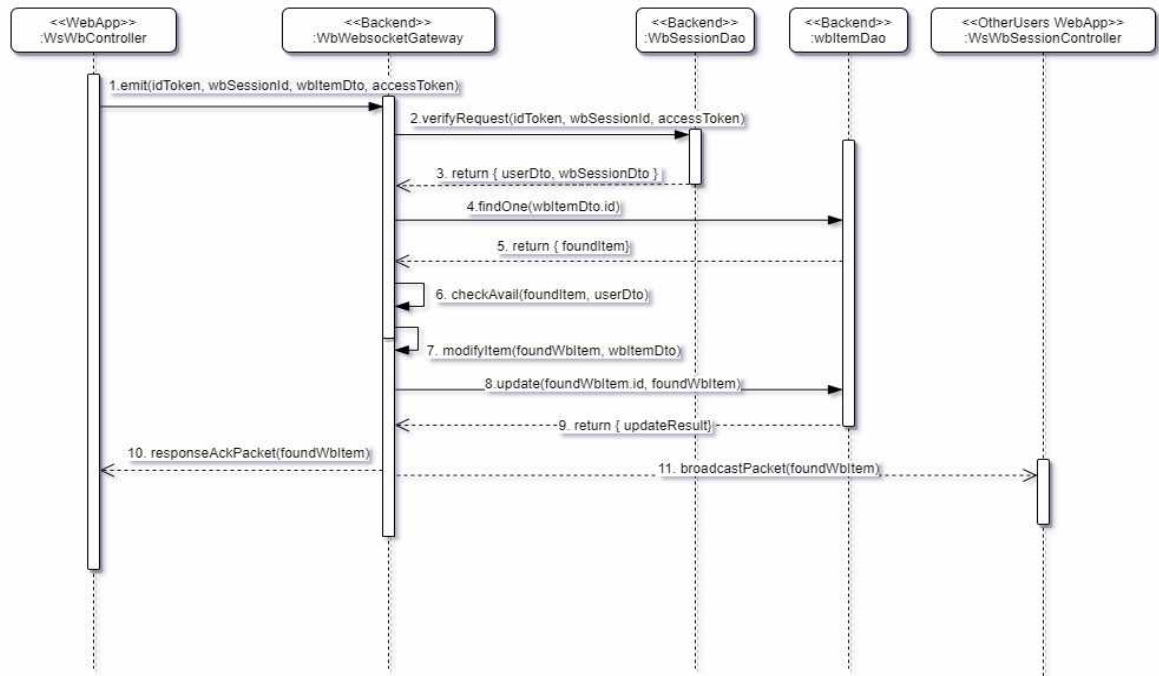
Sequence Diagram of Backend: CreateWbltem



순번	내용
1	□ wsWbController는 wbWebsocketGateway에 생성요청 메시지를 전송.
2	□ WbWebsocketGateway는 메시지를 수신하고, 이 메시지가 유효한지 검증하기 위해 WbSessionDao의 verifyRequest메서드를 호출.
3	□ wbSessionDao의 verifyRequest메서드는 패러미터로 받은 idToken, wbSessionId를 가지고 DB에서 UserDto, WbSessionDto를 찾음. □ UserDto의 AccessToken과, 요청메세지에 있던 AccessToken을 비교하여 사용자 검증을 수행하고, wbSessionId에 해당하는 Document가 DB에 있는지 확인하고, 결과를 반환함. □ 만약 검증을 실패하게 되면 reject메서드를 호출하여 시퀀스를 중단하고 사용자에게 NakPacket을 전송함.
4	□ 생성요청메세지의 WbltemDto를 가지고 wbltemPacket객체를 생성
5	□ wbltemDao로 Document생성.
6	□ 생성된 Document를 반환
7	□ wbSession에 생성한 Wbltem의 아이디를 추가.
8 ~ 9	□ WbSessionDao를 통해 WbSession 수정 후 결과 받음
10	□ wbltemPacketDto안의 WbltemDto에도 새로 발급받은 아이디를 저장
11 ~ 12	□ WbltemDao를 통해 WbltemPacket을 수정 후 결과 받음
13	□ WbWebsocketGateway는 요청한 웹앱에 수정된 WbltemDto를 전송.
14	□ 해당 화이트보드 세션에 접속해 있는 다른 유저들에게도 WbltemDto를 전송.

## □ WblItem 수정요청 시퀀스 다이어그램

Sequence Diagram of Backend: Update WblItem

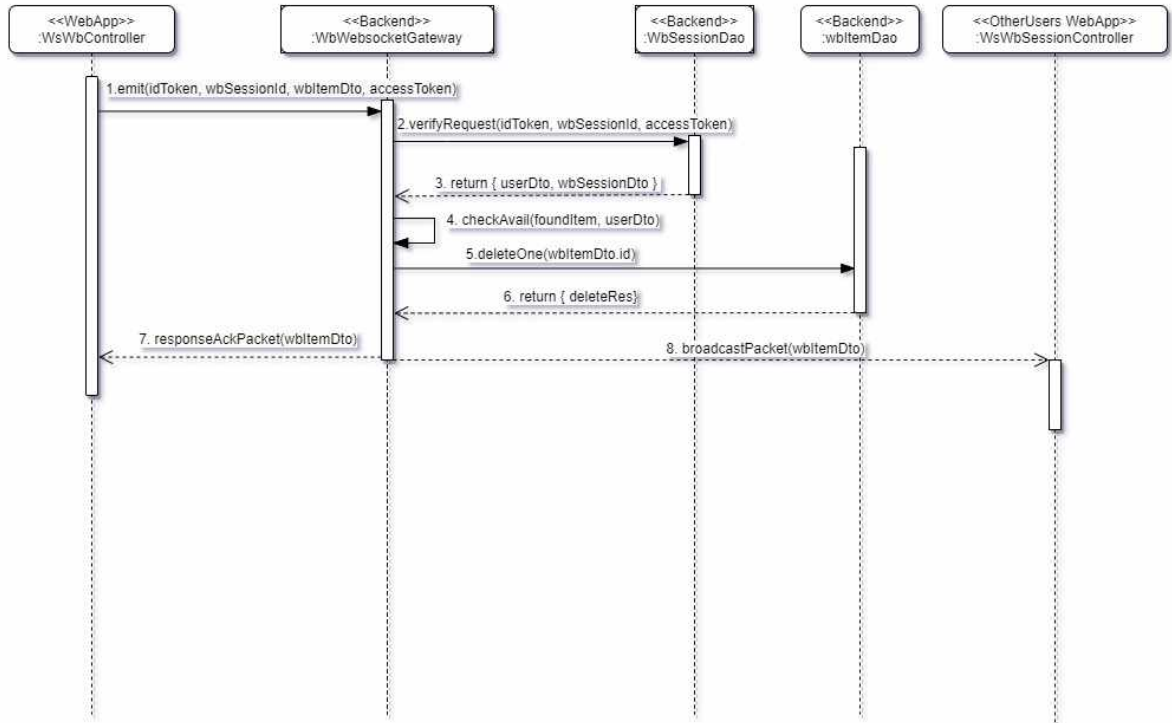


순번	내용
1	□ wsWbController는 wbWebsocketGateway에 수정요청 메시지를 전송.
2	□ WbWebsocketGateway는 메시지를 수신하고, 이 메시지가 유효한지 검증하기 위해 WbSessionDao의 verifyRequest메서드를 호출.
3	□ wbSessionDao의 verifyRequest메서드는 패러미터로 받은 idToken, wbSessionId를 가지고 DB에서 UserDto, WbSessionDto를 찾음. □ UserDto의 AccessToken과, 요청메세지에 있던 AccessToken을 비교하여 사용자 검증을 수행하고, wbSessionId에 해당하는 Document가 DB에 있는지 확인하고, 결과를 반환함. □ 만약 검증을 실패하게 되면 reject메서드를 호출하여 시퀀스를 중단하고 사용자에게 NakPacket을 전송함.
4 ~ 5	□ 수정요청메세지의 WblItemDto의 ID를 가지고 WblItemDao에서 찾고 리턴
6	□ 찾은 아이템을 선택한 사용자의 IDToken과 UserDto의 IDToken을 비교. 같으면 진행하고 실패시 사용자에게 NakPacket전송
7	□ wblItemDto정보대로 찾은 아이템을 수정.
8 ~ 9	□ WblItemDao를 통해 DB에서 아이템을 수정하고 결과 받음
10	□ WbWebsocketGateway는 요청한 웹앱에 수정된 WblItemDto를 전송.
11	□ 해당 화이트보드 세션에 접속해 있는 다른 유저들에게도 WblItemDto를 전송.



❑ WblItem 삭제요청 시퀀스 다이어그램

Sequence Diagram of Backend: Delete WblItem

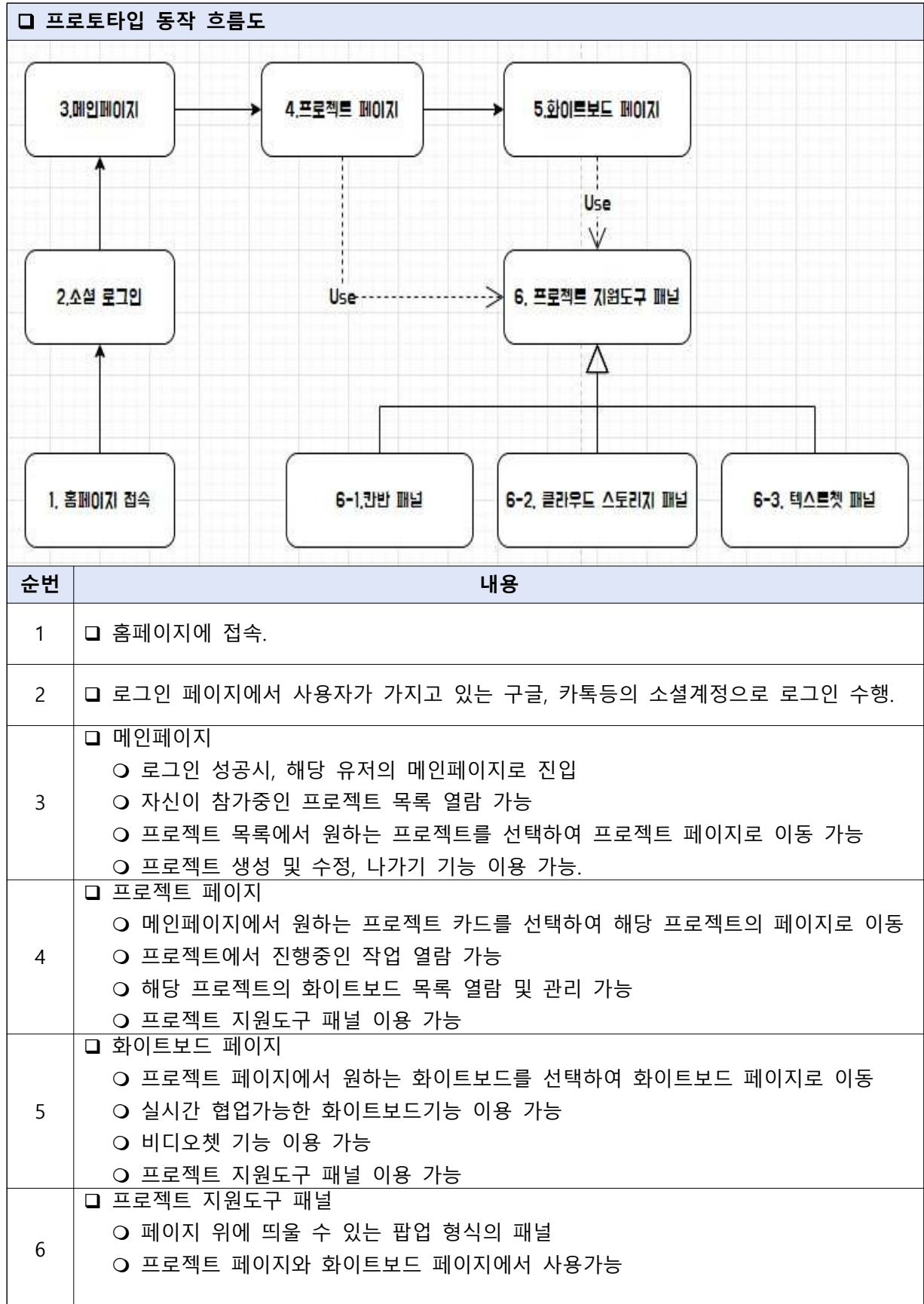


순번	내용
1	❑ Delete 요청할 wblItem의 Dto와 idToken, wbSessionId, accessToken을 사용해 socket에게 Delete 이벤트를 발생시킨다.
2	❑ WbWebsocketGateway에 정의된 Delete 이벤트 처리 메서드에서 WbSessionDao에게 유효성 검사를 요청한다.
3	❑ idToken과 wbSessionId와 accessToken를 DB와 대조하고 유효하다면 userDto와 wbSessionDto를 반환한다. 유효하지 않다면 reject를 호출하여 시퀀스를 중단한다.
4	❑ 제거 대상인 아이템이 현재 사용가능한지 체크한다. 다른 사람에 의해 선점이 되었는지 체크하게 된다.
5	❑ 제거 가능한 상태라고 판단되면 wblItemDao에 wblItemDto의 Id로 제거 요청을 보낸다.
6	❑ DB에서 파라미터로 전달된 Item의 ID에 해당하는 정보의 제거작업을 수행하고 결과를 반환한다.
7	❑ 아이템의 제거 정보가 담긴 Dto를 AckPacket에 담아 WsWbController에게 전달해준다.
8	❑ 나머지 사용자들에게도 아이템 제거를 동기화 하기 위해 제거 정보가 담긴 Dto를 socket을 통해 브로드캐스팅 해준다.



## (5) Prototype 구현

### 1) 프로토타입 동작 흐름도



2) 프로토타입 세부 구성


□ 홈페이지 프로토타입

GachiBoardProductDevTeam

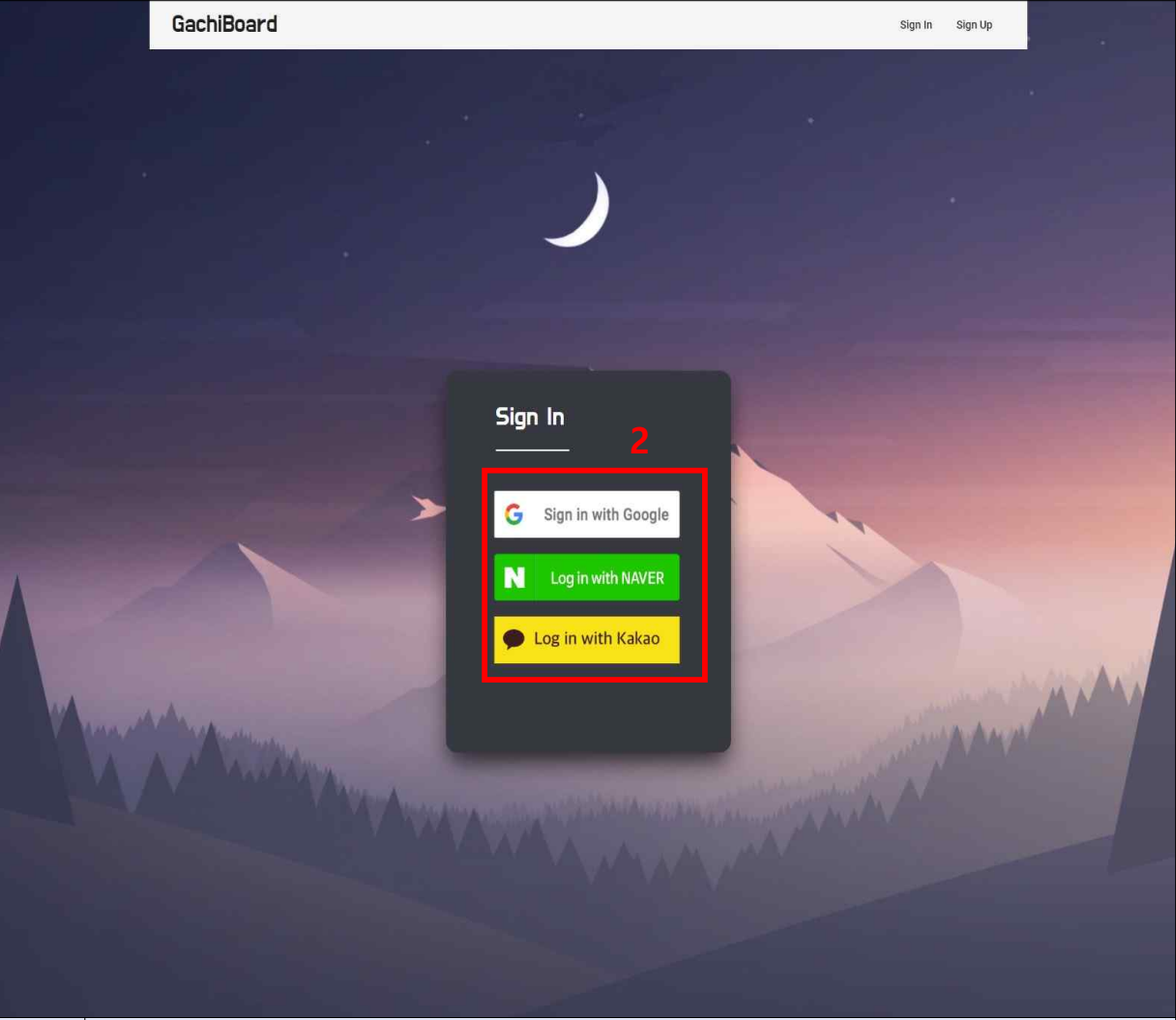
2Sign InSign Up

언제나 같이보고 같이듣고 같이써요

동료들이 멀리 떨어져 있어도 마치 옆에 있는 것처럼 팀원들과 아이디어를 주고 받을 수 있어요



순번	내용
1	□ 같이보드 서비스의 홈페이지
2	□ 우상단의 버튼을 통해 로그인하거나 로그아웃 할 수 있음.

□ 로그인 페이지 프로토타입	
	
순번	내용
1	□ 로그인 페이지
2	□ 각각의 소셜로그인 버튼을 통해, 가지고 있는 소셜계정으로 서비스를 이용할 수 있음.

**□ 메인 페이지 프로토타입**

GachiBoard

My Project   Sign Out

나승철님이 참여중인 프로젝트예요

**3**

→

**3-1**

**2**

→

**4-1**

☺ 같이보드의 사용설명서예요!

**1. 프로젝트 생성하기**

현재 페이지에서는 프로젝트를 생성할 수 있어요.  
프로젝트를 만들게 되면, 만들 사람이 해당 프로젝트의 관리자가 됩니다.  
프로젝트의 관리자가 되면, 다른 사용자를 초대하거나 내보낼 수 있어요.

**2. 프로젝트 이용하기**

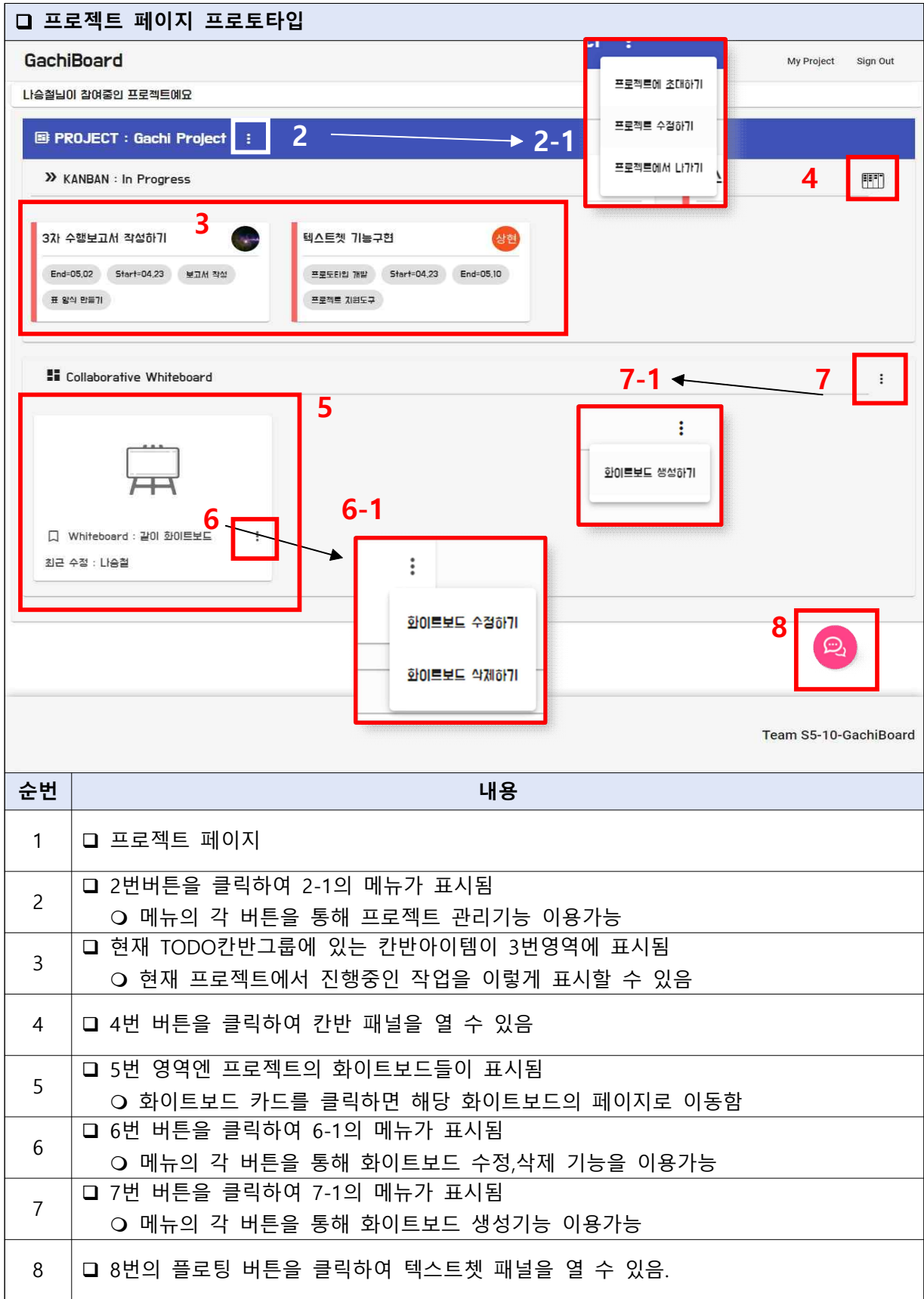
프로젝트를 생성하면, 다양한 기능을 이용할 수 있어요.  
프로젝트 구성원과 실시간으로 같이 사용할 수 있는 칸반, 화이트보드 서비스가 있습니다.  
프로젝트 하나당, 하나의 칸반을 사용할 수 있어요.  
화이트보드는 최대 5개까지 생성할 수 있습니다.

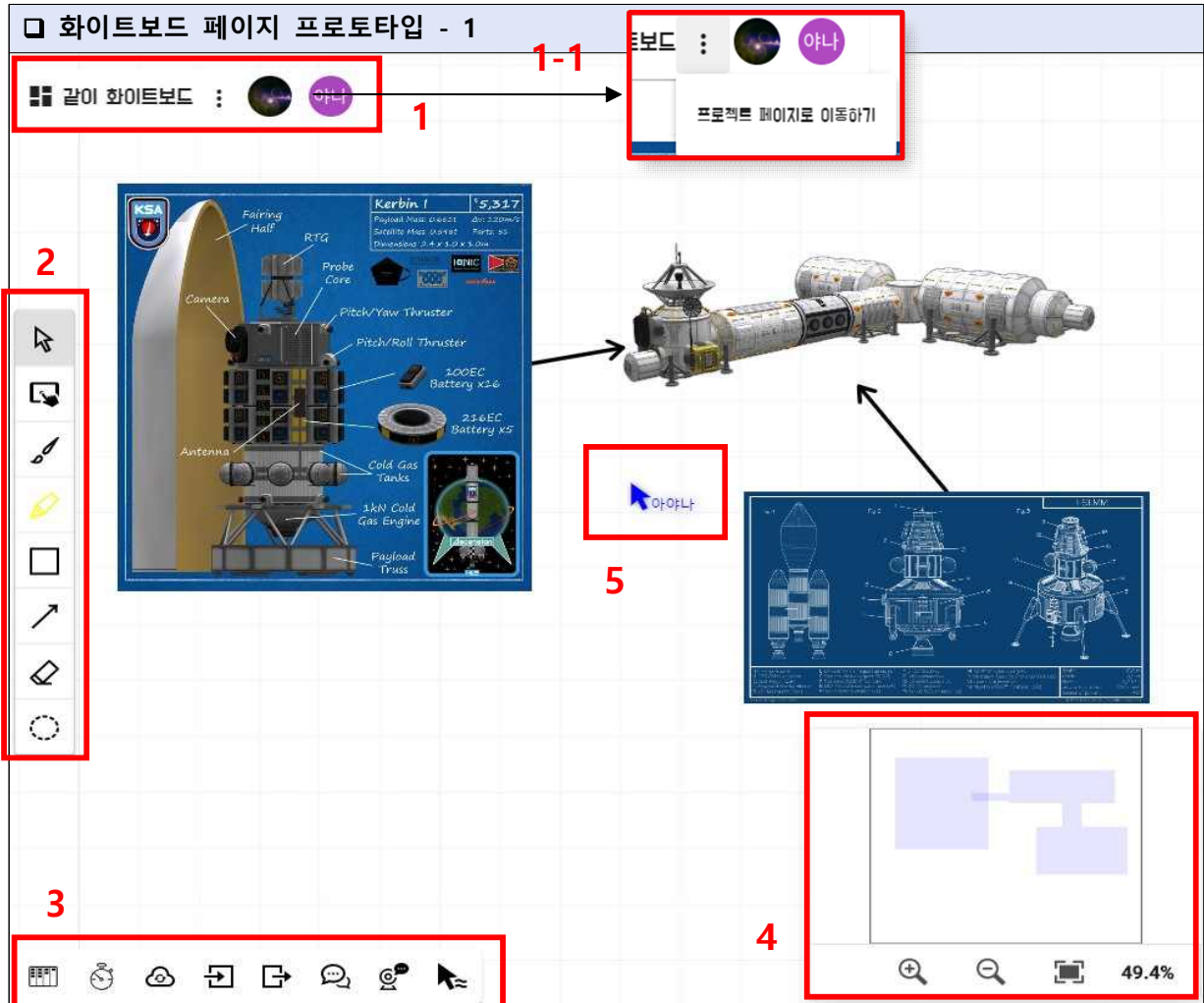
**프로젝트 수정하기**

**프로젝트에서 나가기**

Team S5-10-GachiBoard

순번	내용
1	□ 사용자의 메인페이지.
2	□ 사용자가 참여중인 모든 프로젝트가 스크린샷 2번영역처럼 표시됨. □ 원하는 프로젝트의 카드를 클릭하면, 해당 프로젝트의 페이지로 이동함.
3	□ 3번 버튼을 클릭하면 3-1상자안의 그림처럼 메뉴가 표시됨. ○ 사용자는 메뉴를 통해 프로젝트 생성하기 기능을 이용 가능
4	□ 4번 버튼을 클릭하면 4-1상자 안의 그림처럼 메뉴가 표시됨 ○ 사용자는 메뉴를 통해 프로젝트를 수정, 나가기 기능을 이용 가능





순번	내용
1	<ul style="list-style-type: none"> <li>□ 해당 영역에 화이트보드의 이름과 메뉴버튼, 접속자목록이 표시됨 <ul style="list-style-type: none"> <li>○ 메뉴버튼을 클릭하면 1-1과 같은 메뉴가 표시됨. 해당 버튼으로 프로젝트페이지로 돌아가는 기능 이용 가능</li> <li>○ 현재 해당 화이트보드에 접속중인 사용자의 아이콘이 표시됨</li> <li>○ 사용자 아이콘을 클릭하여 화이트보드에서 해당 사용자가 있는 위치로 이동함</li> </ul> </li> </ul>
2	<ul style="list-style-type: none"> <li>□ 화이트보드 도구패널</li> <li>□ 패널의 각 버튼을 통해 사용자는 브러쉬나 형광펜, 도형도구 등의 기능을 이용 가능</li> </ul>
3	<ul style="list-style-type: none"> <li>□ 프로젝트 지원도구 패널</li> <li>□ 패널의 각 버튼을 통해 사용자는 칸반기능, 타임타이머등의 지원도구 기능을 이용가능</li> </ul>
4	<ul style="list-style-type: none"> <li>□ 화이트보드를 미니맵으로 표시해주는 패널. 현재 사용자가 보고 있는 화면을 검정색 사각형으로 표현하고, 화이트보드 아이템은 하늘색 영역으로 표시함</li> <li>□ 미니맵 하단의 버튼을 이용해 줌인/아웃 기능과 전체화면 기능 이용 가능</li> <li>□ 현재 화이트보드상 줌 배율을 퍼센트로 표시함</li> </ul>
5	<ul style="list-style-type: none"> <li>□ 다른 사용자가 가리키고 있는 곳을 5번 안의 커서로 표시함 <ul style="list-style-type: none"> <li>○ 사용자가 화이트보드에서 마우스를 이동하면 자동으로 커서위치정보가 화이트보드에 접속중인 다른 사용자들에게 브로드캐스트됨</li> <li>○ 이 기능을 통해 화이트보드에 접속중인 사용자 간 서로의 위치를 파악할 수 있음</li> </ul> </li> </ul>

**□ 화이트보드 페이지 프로토타입 - 2**

Project Space : 상현

2

Broadcast cam

Broadcast screen

End video chat

1-1

53.4%

순번	내용
1	<b>□ 비디오챗 기능</b> <ul style="list-style-type: none"> <li>○ 1번 사각형안의 프로젝트 지원도구 버튼을 클릭하면, 1-1의 메뉴가 표시됨</li> <li>○ 메뉴를 통해 기기의 캠을 사용할건지, 아니면 화면공유를 사용할건지 선택가능</li> </ul>
2	<b>□ 캠을 선택한 경우, 2번의 사각형처럼, 화이트보드 참여자들이 화상체팅기능 이용가능</b>

KPU 기업을 품는 산학융합 선도대학  
한국산업기술대학교

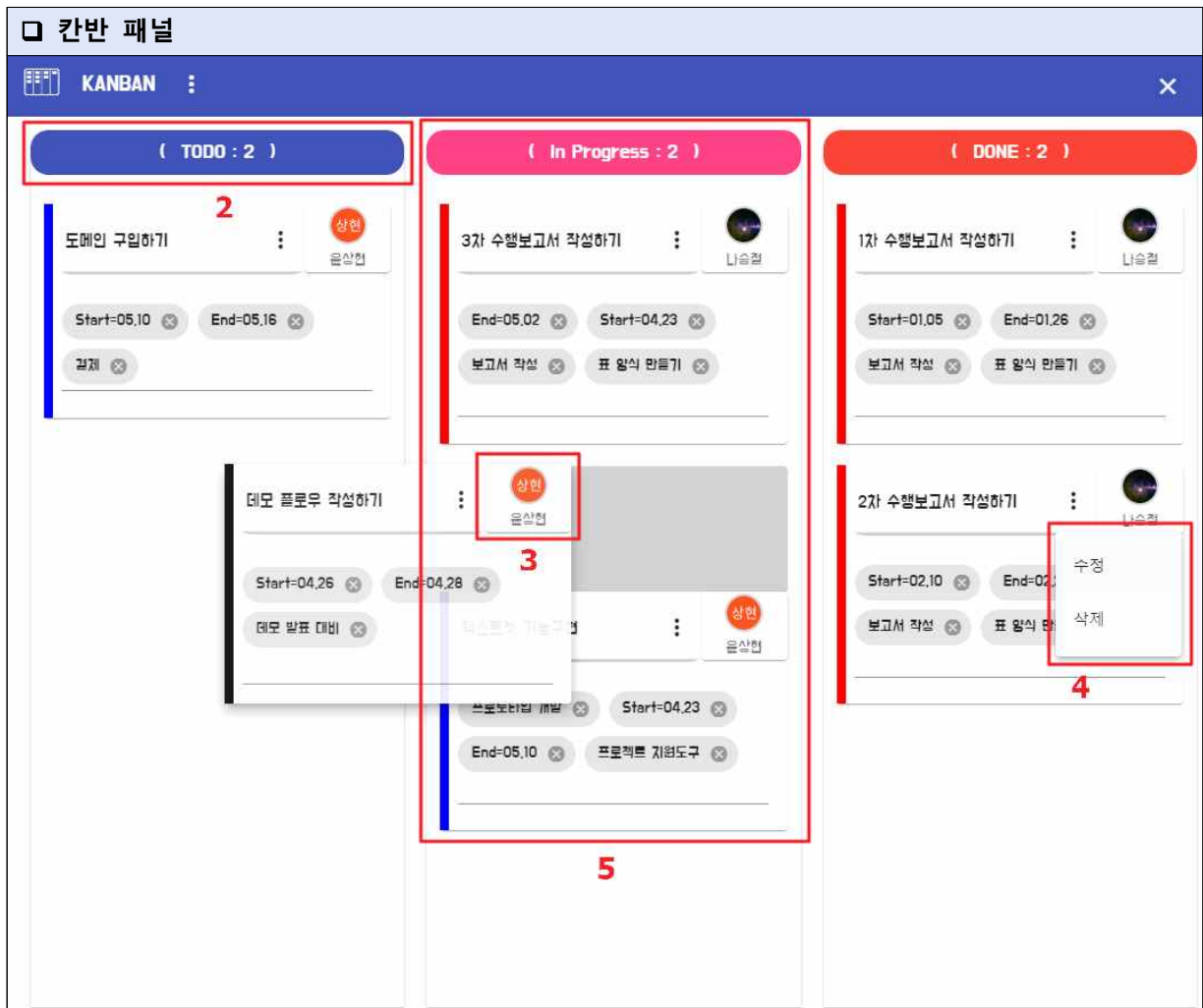
- 47 -



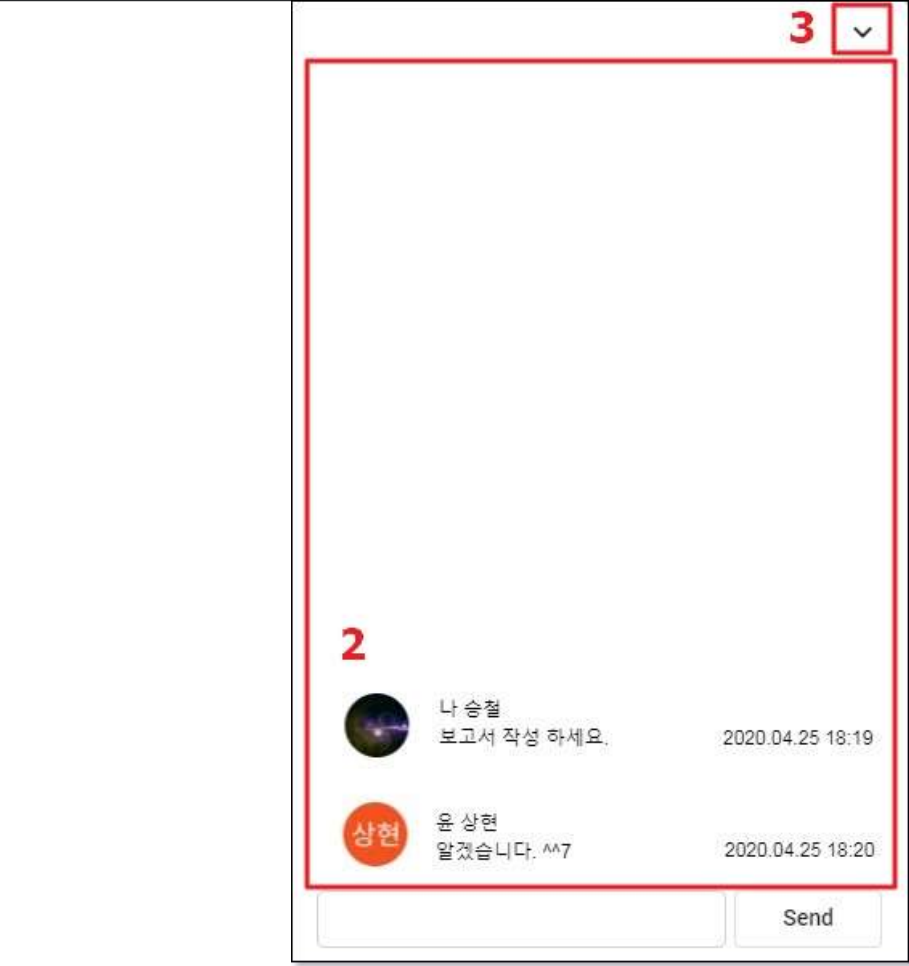
□ 화이트보드 페이지 프로토타입 - 3

순번	내용
1	□ 비디오챗 기능(2) ○ 반대로 메뉴에서 브로드캐스트 스크린 버튼을 선택하면 화면공유 기능을 수행함
2	□ 그러면 2번 사각형처럼 서로의 화면을 공유하고 볼 수 있음 ○ 화면공유기능에 쓰인 API는 데스크톱 환경의 크롬 브라우저만 지원하는 관계로, 크롬 브라우저를 쓰는 데스크톱 유저만 자신의 화면을 공유할 수 있음. ○ 하지만 이렇게 공유된 화면은 다른 기기를 쓰는 사용자들이 보는건 할 수 있음.
3	□ 3번 사각형 안의 버튼을 통해 해당 비디오챗 패널을 조정할 수 있음 ○ 오른쪽 버튼을 누르면 패널 크기를 바꿀 수 있음 ○ 왼쪽 버튼을 누르면, 전체화면 크기로 볼 수 있음.





순번	내용
1	□ 프로젝트 지원도구인 칸반 패널을 화면에 펼쳤을 때의 모습.
2	□ 각 컬럼의 타이틀부분을 누르면 칸반 아이템을 생성할 수 있으며 아이템의 이름과 태스크 매니저를 지정할 수 있다.
3	□ 태스크 매니저가 표시되는 부분을 잡고 드래그해서 다른 컬럼으로 옮길 수 있다. □ 이미지에에는 '데모 플로우 작성하기' 아이템을 In Progress 컬럼으로 옮기고 있는 상황이다.
4	□ 칸반 아이템의 ... 버튼을 누르면 아이템의 이름을 수정하거나 삭제할 수 있다. 수정의 경우 생성할 때 나왔던 패널이 나오므로 수정하고 저장하면 된다.
5	□ In Progress라고 표시되는 이 컬럼은 프로젝트 페이지에서도 볼 수 있어서 현재 프로젝트의 진행중인 목록들을 보기 쉽게 나열해 준다.

□ 텍스트 채팅 패널	
	
순번	내용
1	□ 텍스트 채팅을 할 수 있는 패널이다. 프로젝트 페이지나 화이트보드 페이지에서 이 창을 띄울 수 있으며 이 창을 띄우는 버튼은 새로운 메시지가 몇 개나 도착했는지 확인할 수 있도록 뱃지가 달려있다.
2	□ 채팅 메시지가 표시되는 부분이다. 프로젝트에 있는 사람들과 채팅한 메시지들이 시간순서대로 나열된다.
3	□ 텍스트 채팅 패널을 닫을 수 있는 버튼이다.



순번	내용
1	<ul style="list-style-type: none"> <li>□ 같은 프로젝트에 참여중인 사용자들이 함께 쓸 수 있는 클라우드 공간이다.</li> <li>□ 클라우드에 이미지나 동영상, 압축파일등을 업로드하여 프로젝트 구성원 간에 파일을 공유할 수 있다.</li> </ul>
2	<ul style="list-style-type: none"> <li>□ 좌측 사이드 메뉴를 이용하여 폴더를 생성하거나 파일을 업로드할 수 있다.</li> </ul>
3	<ul style="list-style-type: none"> <li>□ 현재 위치한 폴더에 있는 파일 및 폴더 목록이다. 다른 사용자가 이름을 바꾸거나 파일 및 폴더를 삭제하는 경우, 변경된 상태로 동기화된다.</li> </ul>
4	<ul style="list-style-type: none"> <li>□ 현재 위치한 경로를 표시하는 컴포넌트이다.</li> <li>□ 폴더버튼을 누르면, 해당 폴더로 이동한다. 예를들어, root를 클릭하면 루트 폴더로 이동하게 된다.</li> </ul>
5	<ul style="list-style-type: none"> <li>□ 사이드바 우측의 빈 공간에 마우스 커서를 올리고 우클릭을 하는 경우 나타나는 패널이다.</li> <li>□ 이 패널을 이용하여 폴더를 생성하거나, 이전 폴더로 이동할 수 있다.</li> </ul>

**☐ 클라우드 스토리지 패널 - 2**







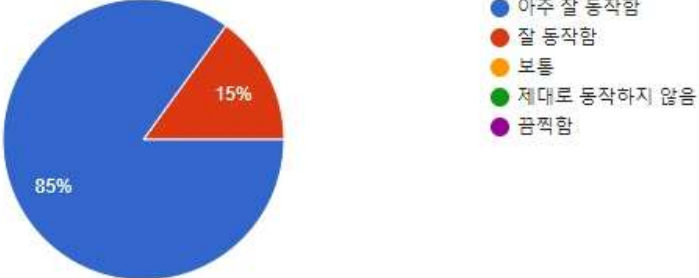
순번	내용
1	☐ 좌측 사이드메뉴의 파일 업로드 버튼을 클릭하면, 업로드할 파일을 선택하는 창이 표시된다.
2	☐ 확인버튼을 클릭하면 오른쪽처럼 파일을 업로드 중이라는 패널이 표시된다.
3	☐ 다운로드를 하는 경우엔 맨 아래 그림처럼 파일을 다운로드 중 이라는 패널이 표시된다

## (6) 시험/ 테스트 결과

## 1) 사용성 테스트

□ 소셜 로그인 기능		
기본 응답	다음은 같이보드의 소셜로그인 기능입니다. 소셜 로그인 기능은 사용하기 쉽습니까? 응답 20개	
	<p>85% 10%</p> <ul style="list-style-type: none"> <li>아주 쉬움</li> <li>쉬움</li> <li>보통</li> <li>복잡함</li> <li>아주 복잡함</li> </ul>	
추가 응답	추가로 이용하고자 하는 소셜 로그인(OAuth) 서비스가 있나요?	
	페이스북 ID로 로그인	4
	애플 ID로 로그인	2
	자체 로그인	1
순번	내용	
1	□ 15%의 제외한 나머지 사용자들은 아주 쉽다고 응답하였다.	
2	□ 추가로 페이스북 ID를 사용해 로그인을 시도하려는 사람이 많았으므로 향후 추가해야할 OAuth로 페이스북 계정을 추가했다.	
3	□ 애플 ID로 로그인 하기 위한 Apple Developers에 가입하는데에 10만원의 비용이 필요하므로 일단 보류했다.	

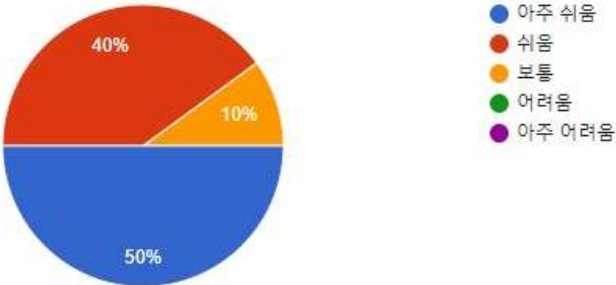
□ 그리기 도구		
기본 응답	<p>다음은 같이보드의 화이트보드 중 그리기 도구입니다. 각각의 도구들은 사용하기 편했습니까?</p> <p>응답 20개</p> 	
추가 응답	어떤점이 불편했나요?	
	단축키들의 사용법이 필요하다.	1
	도형의 프리셋이 좀더 있었으면 좋겠다.	1
	지우개 도구로 한 획의 일부만 지울 수 있었으면 좋겠다.	1
추가 응답	그리기 도구 중 추가하면 좋을 기능이 있나요?	
	브러시로 쉬프트 누른채 그리면 직선을 그릴 수 있었으면 좋겠습니다.	1
	연필기능이 있었으면 좋겠습니다.	1
	채우기(페인트통) 기능이 있었으면 좋겠습니다.	1
순번	내용	
1	□ '아주 편함'이 주를 이루지는 못했지만 최소 '편함' 이상의 응답을 받았다.	
2	□ 여러 가지 추가기능을 많이 응답받았다. 링크 기능의 프리셋을 계획 해뒀지만 도형, 브러시의 프리셋도 추가하고 여러 가지 사용자 편의성에 대한 고민을 해야한다.	
3	□ 그리기 도구로써 부족한점은 많았지만 기본기능 자체는 잘 동작하여 기본응답을 좋게 받은 것으로 보인다.	

□ 화이트보드 동기화 기능		
기본 응답	<p>같이보드의 화이트보드에는 화이트보드를 동기화하는 기능이 포함되어 여러명과 동시에 그릴 수 있습니다. 이 기능은 사용에 고민할 필요없이 잘 동작하였습니까?</p> <p>응답 20개</p> <div style="display: flex; align-items: center; justify-content: center;">  <div style="margin-left: 20px;"> <ul style="list-style-type: none"> <li><span style="color: blue;">●</span> 아주 잘 동작함</li> <li><span style="color: red;">●</span> 잘 동작함</li> <li><span style="color: orange;">●</span> 보통</li> <li><span style="color: green;">●</span> 제대로 동작하지 않음</li> <li><span style="color: purple;">●</span> 끔찍함</li> </ul> </div> </div>	
	추가 응답	<p>제대로 동작하지 않았던 부분이 있었습니까?</p> <p style="text-align: center;">없음</p>
	순번	내용
	1	□ '아주 잘 동작함'이 85%, '잘 동작함'이 15%로 과반수가 잘 동작한다고 응답했다.
	2	□ 동기화기능 테스트를 위해 최소 두명씩 진행하였고 동기화기능에 문제 없이 진행된 것으로 보인다.

□ 화이트보드 불러오기 / 내보내기 기능		
기본 응답	다음은 같이보드의 화이트보드 중 불러오기, 내보내기 기능입니다. 해당 기능은 사용하기 쉬웠습니까? 응답 20개	
	<div><div><div><div><div></div><div>아주 쉬움</div></div><div><div></div><div>쉬움</div></div><div><div></div><div>보통</div></div><div><div></div><div>어려움</div></div><div><div></div><div>아주 어려움</div></div></div><div><div><div><div>60%</div></div><div><div>35%</div></div><div><div></div></div></div></div></div></div>	
추가 응답	그리기 도구 중 추가하면 좋을 기능이 있나요?	
	메뉴 한글화가 필요합니다.	3
	이미지를 추가하는 최대 용량이 작다.	1
순번	내용	
1	□ 메뉴 한글화는 메뉴 자체를 한국어로 적는 것 보다 다중언어 지원방식을 통해 여러 국가 언어를 담은 스크립트를 작성하는 쪽으로 업데이트할 계획이다.	
2	□ 이미지 최대 용량의 제한은 로딩방식 최적화를 진행한 후에 차차 늘려나갈 계획이다.	

□ 프로젝트 칸반 기능		
기본 응답	다음은 같이보드의 프로젝트 기능 중 칸반기능 입니다. 칸반은 프로젝트 관리도구이며 현재 진행 중인 카드들은 프로젝트 페이지에서도 볼 수 있습니다. 칸반기능은 사용하기 쉬웠습니까? 응답 20개	
	<div><div><div><div><div></div><div>70%</div></div><div><div></div><div>25%</div></div><div><div></div><div></div></div></div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div>아주 쉬움</div><div>쉬움</div><div>보통</div><div>어려움</div><div>아주 어려움</div></div></div>	
추가 응답	사용하기 어려웠던 점은 무엇이었나요?	
	사용법에 대한 간단한 설명이 필요합니다.	2
	칸반 아이템을 이동하는 부분을 알기 쉽게 표시해줬으면 좋겠습니다.	1
순번	내용	
1	□ '아주 쉬움'과 '쉬움'이 대부분이고 1명의 응답자가 '보통'을 선택하였다.	
2	□ 사용성 자체는 쉽고 간단했지만 최초 사용시에 사용법을 알려줬으면 조금더 빠르게 사용법을 습득할 수 있다는 의견이 많았다. 따라서 칸반에도 반투명 레이어 위에 사용법을 최초 로딩시에 보여줄 수 있도록 추가할 계획이다.	
3	□ 칸반 아이템의 이동을 아이템 전체를 잡고 이동할 수 있도록 방법을 강구해야 한다.	



□ 화이트보드 비디오채팅 기능		
기본 응답	<p>다음은 같이보드의 화이트보드 기능 중 비디오채팅 기능입니다. (이미지는 화면공유 중) 비디오 채팅 기능을 쉽게 이용할 수 있었습니까?</p> <p>응답 20개</p> 	
추가 응답	사용하기 어려웠던 점은 무엇이었나요?	
	비디오 채팅의 패널크기가 작다.	1
순번	내용	
1	□ 50%의 응답자가 '아주 쉬움'을 선택했고 40%의 응답자가 '쉬움'을 선택했으나 10%의 응답자가 '보통'을 선택했다. 다른 기능들에 비해 조금 낮은 사용성을 갖는 것 같아 보인다.	
2	□ 비디오 채팅의 패널크기를 좀더 다양하게 선택할 수 있도록 향후 업데이트 계획에 추가했다.	

## 2) 성능테스트

□ Artillery.io를 이용한 성능테스트	
테스트 대상	□ 화이트보드 동기화 기능
테스트 사유	□ 서비스 중에서 가장 많은 요청이 발생하는 구간이므로, 화이트보드 아이템 생성요청기능으로 테스트하기로 결정하였음.
테스트 시나리오	□ 30초동안 화이트보드에 도형객체를 생성하는 요청을 초당 30번 수행. ○ 테스트 과정에서 발생하는 총 요청횟수 = 900회 □ 최소, 최대 소요시간, 중간값, p95, p99값을 도출

□ 성능테스트 결과													
<p>Test Scenario Result</p> <table border="1"> <caption>Test Scenario Result Data</caption> <thead> <tr> <th>Statistical Measure</th> <th>Value (msec)</th> </tr> </thead> <tbody> <tr> <td>min</td> <td>49.1</td> </tr> <tr> <td>max</td> <td>136.8</td> </tr> <tr> <td>median</td> <td>59.8</td> </tr> <tr> <td>p95</td> <td>81.3</td> </tr> <tr> <td>p99</td> <td>105.4</td> </tr> </tbody> </table>		Statistical Measure	Value (msec)	min	49.1	max	136.8	median	59.8	p95	81.3	p99	105.4
Statistical Measure	Value (msec)												
min	49.1												
max	136.8												
median	59.8												
p95	81.3												
p99	105.4												
구분	내용												
결과 내용	<p>□ 초당 30회씩 요청하는 상황을 조성하여 최선, 최악의 상황일 때 걸리는 시간을 측정한 결과, 최선의 상황엔 49.1msec가 소요되었고, 최악의 상황에선 136.8msec가 소요됨을 알 수 있었음</p> <p>□ 또한 p99값이 105.4msec인 점에서, 다수의 사용자가 접속해서 서비스를 이용하게 되면, 평균적으로 105.4msec정도 기다려야 응답받는다는 것을 알 수 있었음.</p> <p>□ 초기에 0.5초 이하의 응답시간만 유지하면 원활한 서비스 이용이 가능할 것이라고 예측하였고, 테스트 결과 구현된 서비스가 이를 만족함을 알 수 있었음.</p>												

## (7) Coding & DEMO

### 1) Coding

#### □ 주요 소스코드

❖ 서비스의 주요부분인 동기화 기능에 중점을 두어 기술함 ❖

#### ( 1 ) 화이트보드 그리기 기능

```
public endPath() {
    if(!this.newPath) {
        this.newPath.simplify(3);
        //addToDrawingLayer를 이용하여 화이트보드 아이템 생성
        this.layerService.addToDrawingLayer(this.newPath, WhiteboardItemType.SIMPLE_STROKE);
        this.newPath = null;
        this.layerService.tempProject.activeLayer.removeChildren();
    }
}
```

#### ○ brushService클래스의 endPath 메서드

- 사용자가 마우스나 전자펜으로 화이트보드에 그림을 그리다가 떼는 순간 호출되는 콜백함수.
- 사용자가 그린 정보를 newPath값을 검사하고, simplify메서드로 최적화한다.
- 레이어서비스의 addToDrawingLayer메서드의 패러미터로 newPath를 넘긴다.

( 2 ) 화이트보드 아이템 생성하고, 화이트보드 레이어에 추가하는 기능

```
public addToDrawingLayer(item, type, ...extras){
    let newWhiteboardItem: WhiteboardItem =null;
    //Stroke 형태인 경우
    if(DrawingLayerManagerService.isEditableStroke(type)){
        //path데이터인 경우, 스트로크 타입의 WblItem을 생성한다.
        switch (type) {...}
    }
    else if(DrawingLayerManagerService.isEditableRaster(type)){
        //이미지 데이터인 경우, 래스터 타입의 WblItem을 생성한다
        newWhiteboardItem =new SimpleRaster(this.getWbld(), item, this);
        this.uiService.spin$.next(true);
    }
    else if(DrawingLayerManagerService.isEditableShape(type)){
        //도형 데이터인 경우, EditableShape타입의 WblItem을 생성한다.
        let editText:PointText = extras[0];
        let newTextStyle:TextStyle = extras[1];
        switch (type) {...}
    }
    else if(DrawingLayerManagerService.isEditableLink(type)) {
        //링크 데이터인 경우, 링크타입의 WblItem을 생성
        newWhiteboardItem
            =new EditableLink(this.getWbld(), item, extras[0], extras[1], this, extras[2], extras[3]);
    }
    if (newWhiteboardItem) {
        //생성한 WblItem 데이터를 가지고 백엔드서버에 생성요청한다
        let wsWbController = WsWhiteboardController.getInstance();
        wsWbController.waitRequestCreateWblItem(newWhiteboardItem.exportToDto())
            .subscribe((packetDto) => {...});
    }
    return newWhiteboardItem;
}
```

○ DrawingLayerManagerService클래스의 addToDrawingLayer 메서드

- ❑ 그림데이터나 도형데이터를 가지고 화이트보드 아이템을 생성하고, 이를 서버에 전송하는 메서드
- ❑ 함수의 두 번째 패러미터인 type을 가지고, 생성할 화이트보드 아이템을 결정한다
- ❑ 화이트보드 아이템을 만드는데 정보가 추가적으로 필요한 경우엔, ...extras에 채워서 메서드를 호출한다.
- ❑ 마지막으로 생성한 화이트보드 아이템의 데이터를 백엔드 서버에 전송한다.  
이때, WsWhiteboardController클래스의 waitRequestCreateWblItem메서드를 통해 전송한다.  
화이트보드아이템은 DTO형태로 변환하여 전송한다.
- ❑ 서버에서 응답메세지가 오면, subscribe함수의 패러미터로 전달한 람다함수가 콜백함수로서 호출된다. 화이트보드 아이템의 아이디값은 백엔드서버에서 부여해주는데, 이 콜백함수에서 백엔드서버에서 부여해준 아이디값을 기존에 생성한 화이트보드 아이템에 부여한다.
- ❑ 그런 다음, 화이트보드의 Drawing Layer에 화이트보드 아이템을 추가한다.

### ( 3 ) 화이트보드 아이템을 서버에 전송하는 기능

```
waitRequestCreateWblItem( wblItemDto:WhiteboardItemDto ){
    return new Observable <any >((subscriber)=>{
        let packetDto = this.websocketManager
                        .createWbSessionScopePacket(
                            wblItemDto,WebsocketPacketActionEnum.CREATE);

        let newSeq = this.websocketManager.wsPacketSeq;
        packetDto.wsPacketSeq = newSeq;
        this.socket.emit(HttpHelper.websocketApi.whiteboardItem.create.event, packetDto);
        //#### 요청 완료
        this.socket.once(HttpHelper.websocketApi.whiteboardItem.create.event
                        + HttpHelper.ACK_SIGN + newSeq,

            (wsPacketDto:WebsocketPacketDto)=>{
                switch (wsPacketDto.action) {
                    case WebsocketPacketActionEnum.ACK:
                        subscriber.next(wsPacketDto);
                        break;
                    case WebsocketPacketActionEnum.NAK:
                        subscriber.error(wsPacketDto);
                        break;
                }
            });
    });
}
```

#### ○ wsWhiteboardSessionController클래스의 waitRequestCreateWblItem 메서드

- ❑ 화이트보드 아이템 데이터를 서버에 전송하기 위해, 패킷DTO를 생성한다.
- ❑ 패킷 DTO는 seq번호, 이벤트유형, 화이트보드 아이템 데이터, 전송자의 ID, Socket.IO의 Room정보를 포함한다. 패킷 DTO를 만들기 위해 createWbSessionScopePacket()메서드를 이용한다.
- ❑ emit메서드를 통해 패킷을 전송하고나면, 전송할 때 쓴 event 문자열에 ACK\_SIGN과, 전송할 때 사용한 seq번호를 합친 문자열을 채널이름으로 사용하여 서버의 응답을 기다린다.  
만약 이벤트 이름을 채널이름으로 써서 서버의 응답을 기다리게 되면, 다른 요청에 대한 응답과 섞이는 현상이 발생하기 때문에 이런 조치를 취했다.  
만약 event이름이 wblItem\_create이고, seq번호가 765이면, 응답을 기다리는 채널의 이름은 wblItem\_create\_ACK\_765가 된다.
- ❑ 해당 채널을 통해 서버로부터 응답받게 되면, 받은 데이터의 action필드를 살핀다.  
만약 ACK에 해당하는 ENUM값을 갖고 있다면, subscribe메서드의 람다함수로 wsPacketDto값을 넘기고, NAK에 해당하는 ENUM값을 갖고 있다면, 예외처리한다.

( 4 ) 화이트보드 아이템 생성요청을 받아서 처리하는 Websocket 컨트롤러의 기능

```
@SubscribeMessage(HttpHelper.websocketApi.whiteboardItem.create.event)
async onWblItemCreateRequest(socket: Socket, packetDto: WebsocketPacketDto) {
    let wbSession: WhiteboardSessionInstance =
        this.wbSessionManager.getWbSession(packetDto.namespaceValue);
    let release = await wbSession.mutex.acquire();
    try {
        let resolveParam = await this.whiteboardItemDao.saveWblItem(packetDto);
        let userDto = resolveParam.userDto;
        let projectDto = resolveParam.projectDto;
        let createdWblItemPacket = resolveParam.createdWblItemPacket;
        packetDto.dataDto = createdWblItemPacket.wblItemDto;
        WbWebsocketGateway
            .responseAckPacket( socket, HttpHelper.websocketApi.whiteboardItem.create,
                               packetDto, createdWblItemPacket);

    } catch (rejection) {
        this.wsWblItemErrorHandler(rejection, socket, packetDto,
                                   HttpHelper.websocketApi.whiteboardItem.update.event);
    } finally {
        release();
    }
}
```

○ WbWebsocketGateway 클래스의 onWblItemCreateRequest 메서드

- ❑ 화이트보드 아이템 생성요청이 들어오면 이를 처리하는 웹소켓 컨트롤러
- ❑ 화이트보드 아이템을 생성하는 경우, ProjectDTO 객체를 두고 Race Condition이 발생할 수 있으므로, Mutex락을 설정한다. 이 Mutex락은 화이트보드 세션마다 따로 존재한다. 따라서 한 세션에 뮤텍스락이 걸려있더라도, 다른 화이트보드 세션에 영향을 주지 않는다.
- ❑ WhiteboardItemDao 클래스의 saveWblItem 메서드를 이용해, 패킷을 검사하여 유효한 요청인지 검사하고, 유효하다면 DB에 저장하는 작업을 수행한다.
- ❑ 성공하게 되면, responseAckPacket 메서드를 호출하여, 요청자에게는 Ack 패킷을 전송하고, 그 외의 해당 세션에 접속해있는 사용자들에게는 노멀 패킷을 브로드캐스트해준다.
- ❑ 실패하게 되면, WblItemErrorHandler 메서드가 호출되어 NAK 패킷을 요청자에게만 전송한다.
- ❑ 성공하던 실패하던, 작업이 끝나면 Mutex락을 해제한다.

## 2) Demo

□ 데모 사이트 주소

<https://gachiboard.meryu.me>