

프로젝트명	웹 기반 실시간 공동 작업 도구
개요	<p>● 코딩과 문서 작업을 실시간으로 팀원이 참여하여 편집할 수 있게 한다.</p> <p>● 이 프로젝트 결과물로 소규모로 개발하는 대학교 프로젝트, 적은 인원이 투입되는 스타트업 프로젝트에서 의견 엇갈림과 코드 병합을 수월하게 한다.</p> <p>그리고, 문서만을 필요로 하는 소수의 팀(단체)에서 공동 문서 편집을 진행하여, 빠른 의견 수렴, 진행 속도를 증가시킬 수 있다.</p> <p>1. 필요성</p> <p>● 팀 프로젝트를 진행하다 보면 코드의 병합을 위해, Github를 채택하곤 한다. 하지만, 실제 사용하는 이는 적고, 사용할 수 있다고 해도, 사용법이 어려워 미숙한 사람이 대부분이다. 이로 인해, 사용하기 쉽게, 실시간으로 코딩을 수정하여, 진행의 충돌을 피하고, 더 쉽게 코드를 병합할 수 있다.</p> <p>2. 현실적 제한 요소</p> <p>● 프로젝트의 결과물은 데이터의 유지, 서버 사용으로 인한, 유지, 관리비용이 있을 것이다.</p> <p>● 서버 트래픽 관리/유지 및 비밀번호에 대한 암호 보안성이 제한적일 것이라 생각된다.</p> <p>3. 과제 내용</p> <p>1) 세부 내용 1 : 인증 체계</p> <ul style="list-style-type: none"> - 회원가입 페이지 제작 - 로그인 페이지 제작 - 데이터베이스 접속 <p>2) 세부 내용 2 : 팀별 문서 공유 세션 기능</p> <ul style="list-style-type: none"> - 팀원이 한 명이라도 있으면 문서 공유 세션을 유지시킨다. <p>3) 세부 내용 3 : 실시간 문서 공유</p> <ul style="list-style-type: none"> - CRDT 라이브러리를 이용한 실시간 문서 공유 기능 개발 <p>4) 세부 내용 4 : 팀 채팅</p> <ul style="list-style-type: none"> - 웹 소켓을 이용한 팀원 간 채팅 기능 구현 <p>5) 세부 내용 5 : 기타</p> <ul style="list-style-type: none"> - 코드 하이라이트 - ChatGPT 연동
과제 목표	<p>- 팀 프로젝트를 하면서, 코드의 병합 즉, 각자 겹치는 영역을 작성할 때, 현재 개발하는 과제의 결과물로, 여러 어려움이 있을시, 코드 부분 혹은, 문서 부분을 참고하여, 현재 진행상황, 팀원의 작업 부분 등을 확인하여 수월한 프로젝트 진행과, 소통을 가능하게 하는 것이 목표이다.</p>

결과물의 활용방안 및 기대효과	<p>창의성 측면</p> <p>- 프로젝트의 목표에 따라, 코딩 학과 혹은 소규모 프로젝트, 공유 문서를 다루는 학과 혹은, 소규모 기업 문서를 다루는 분야에서 손쉽게 접근 가능하다는 점.</p> <p>기술적 측면</p> <p>- 코드와 문서를 동시에 실시간으로 작업하고, 반영된다는 점과, 접근성이 어렵지 않다는 점이 기술적인 측면에서 기대된다.</p>																		
수행 방법	<p>과제를 위한 환경은 VScode에 Python, Django를 설치하여 사용할 것이다. 김태호(조장)은 프로젝트에 대한 세부적인 계획, 역할 분담과 코드 작성을 맡고, 김규택(조원)은 프로젝트의 주가 되는 알고리즘, 세부적인 코드 작성 기술, 방향성, 코드 작성을 맡고, 김동진(조원)은 프로젝트에 해당하는 UI, 데이터 베이스를 맡고, 코드 작성을 맡았다.</p> <p>주 1회 정기 회의를 하며, 프로젝트의 방향성을 논의하고, 진행 상황, 어려운 점, 추가적인 기능 회의를 진행하며, 수업의 일정에 따라 추가적인 회의 및 역할 분담을 할 것이다.</p>																		
	<table><tr><th>구분</th><th>성명</th><th>과제 참여 내용(역할)</th></tr><tr><td>팀장</td><td>김태호</td><td>프로젝트 회의 주도 및 진행, 전체 UI 디자인, 문서 편집기 기능 구현, 채팅 구현</td></tr><tr><td>팀원</td><td>김동진</td><td>DB 스키마 설계 및 관리, UI 디자인 검토, 사용자 관리 기능 구현, 문서-chatGPT 연동</td></tr><tr><td>팀원</td><td>김규택</td><td>프로젝트 방향 설계, 그룹 관리 기능 구현, 실시간 편집 알고리즘 분석 및 코드 적용</td></tr><tr><td>팀원</td><td></td><td></td></tr><tr><td>팀원</td><td></td><td></td></tr></table>	구분	성명	과제 참여 내용(역할)	팀장	김태호	프로젝트 회의 주도 및 진행, 전체 UI 디자인, 문서 편집기 기능 구현, 채팅 구현	팀원	김동진	DB 스키마 설계 및 관리, UI 디자인 검토, 사용자 관리 기능 구현, 문서-chatGPT 연동	팀원	김규택	프로젝트 방향 설계, 그룹 관리 기능 구현, 실시간 편집 알고리즘 분석 및 코드 적용	팀원			팀원		
	구분	성명	과제 참여 내용(역할)																
	팀장	김태호	프로젝트 회의 주도 및 진행, 전체 UI 디자인, 문서 편집기 기능 구현, 채팅 구현																
	팀원	김동진	DB 스키마 설계 및 관리, UI 디자인 검토, 사용자 관리 기능 구현, 문서-chatGPT 연동																
	팀원	김규택	프로젝트 방향 설계, 그룹 관리 기능 구현, 실시간 편집 알고리즘 분석 및 코드 적용																
	팀원																		
팀원																			

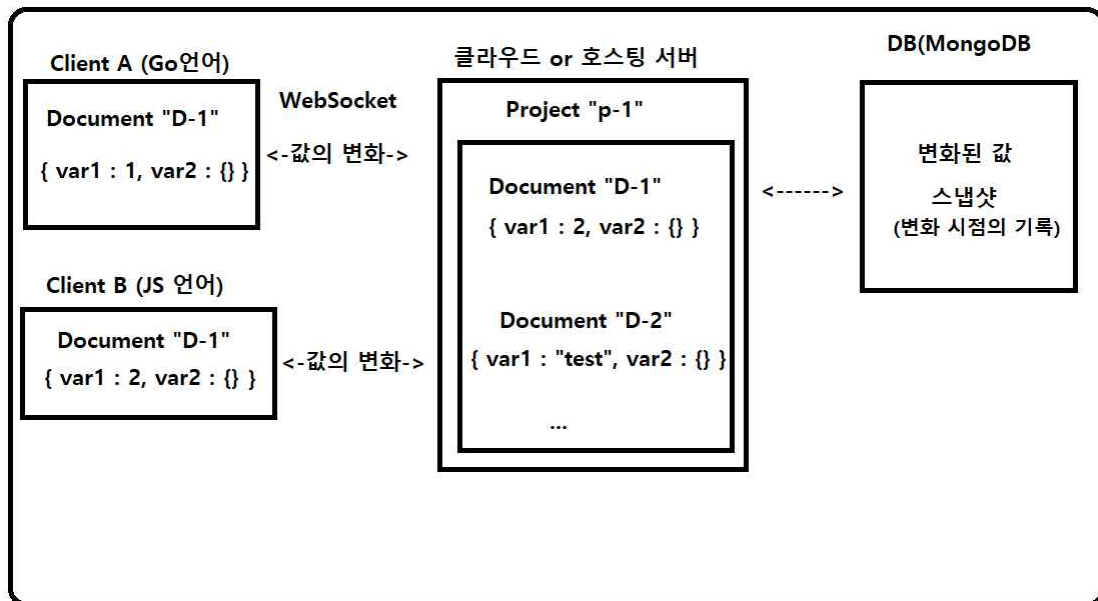
결과물 및 평가방법	<p>● 웹 기반 편집 도구로, 크게는 코드, 문서 툴이 있다. 두 개의 툴은 실시간 수정이 팀원의 화면에 실시간으로 반영된다.</p> <p>문서 툴은 기존의 문서 어플리케이션과 같이, 기능을 동일하게 사용할 수 있다.</p> <p>코드 툴은 Github와 같이 사용될 수 있고, 특정 코드에 하이라이트 기능을 지정할 수 있다.</p> <p>chatGPT를 연계해서, 코드와 관련된 더 높은 범위의 코드 제작을 수행할 수 있다.</p>					
추진 일정	세부 내용	수행기간(월) /마감기한				비고
		3	4	5	6	
	프로젝트 진행 상황 및 추진 일정 발표 동영상 제작	-	4/8	-	-	
	전체 UI 코드 작성 및 데이터베이스 연동 확인	-	4/10	-	-	
	프로그램 주 기능 설계 및 분할 작성	-	-	5/3	-	
	ChatGPT 연동 테스트 결과 판단 후, 프로그램에 적용	-	-	5/30	-	
	정기 회의	매주 월요일 오전 10시				
	오류 검출 및 수정	상시	상시	상시	상시	
기타	<p>【참고문헌】</p> <p>[1] yorkie.dev documentation (https://yorkie.dev/yorkie-js-sdk/api-reference/)</p> <p>[2] codemirror documentation (https://codemirror.net/docs/ref/)</p> <p>[3] Quill documentation (https://quilljs.com/docs/api)</p> <p>[4] Django Documents (https://docs.djangoproject.com/ko/5.1/intro/overview/)</p>					

● 기능 및 구조 설명

-실시간 편집 및 코드 에디터 연동

실시간 편집 기능을 위한 “yorkie” 라이브러리[1]와 코드 편집 환경을 구성할 수 있도록 도와주는 “codemirror” 라이브러리[2]를 사용하기 위해서 nodeJS 서버를 열고 이 기능을 기존에 사용하고자 하는 Django 웹 프레임워크와 연동합니다.

-yorkie의 작동 방식



작성된 언어가 다르더라도 동일한 프로젝트라면 같은 방식으로 동작합니다. yorkie 서버와 연결이 되었다면 클라이언트의 문서 수정이 일어났을 때 서버의 DB에 변화된 값과 스냅샷(snapshot)을 저장하게 되며 저장하기 이전에 서버에서 클라이언트의 문서 수정에 대한 충돌을 CRDT 기법으로 처리합니다. 값의 변화에 대한 트리거는 클라이언트 사이드에서 결정되어 클라이언트의 변경사항과 서버의 변경사항을 각각 캐치합니다. 서버 없이 CRDT 알고리즘만 사용한다면 로컬에서 작성된 문서끼리 id를 비교해서 합칠 수도 있지만 원격으로 떨어진 사용자끼리 실시간으로 값의 변경을 적용하기 위해서는 웹 소켓이 필요하기 때문에 yorkie 내부에서 웹 소켓으로 연결이 되어 있습니다. 클라이언트 간의 연결성을 서버-클라이언트 모델로 해결했다는 관점에서 사실상 OT의 중앙 집중 처리 방식과 다를 것이 없어보이지만 시간 단위의 변경을 index 순서에 맞게 처리하는 것과 고유한 id로 구성된 문자를 합치는 것의 비용 차이는 분명히 있을 것이라고 생각합니다.

-실시간 코드 편집 기능 구현

yorkie는 서버를 직접 호스팅 할 수 있도록 yorkie 서버를 구축할 수 있는 시스템을 지원해주기도 하지만 서버를 직접 구축하지 않아도 회원가입제를 통해 서버를 빌려서 사용할 수도 있습니다. yorkie에서 제공하는 서버를 이용해서 프로젝트를 생성하고 Yorkie Document에 접근할 수 있는 기능을 지원합니다. 이 접근은 API Key를 사용하여 접근할 수 있는데 인증이 가능한 종단점이 구현되면 비밀 키로 접근이 가능하고 그렇지 않다면 공개 키를 사용합니다. 저희 팀에서는 회원가입을 통해 yorkie 서버를 빌리고 공개키로 접근하여 yorkie api를 사용했습니다.

-라이브러리

- i) 실시간 편집을 위한 yorkie-js-sdk
- ii) code editor 구현을 위한 codemirror
- iii) editor 설정을 위한 라이브러리
- iv) 코드 하이라이트를 위한 언어 인식 라이브러리

-그룹

1. 페이지 구조 설계 문제

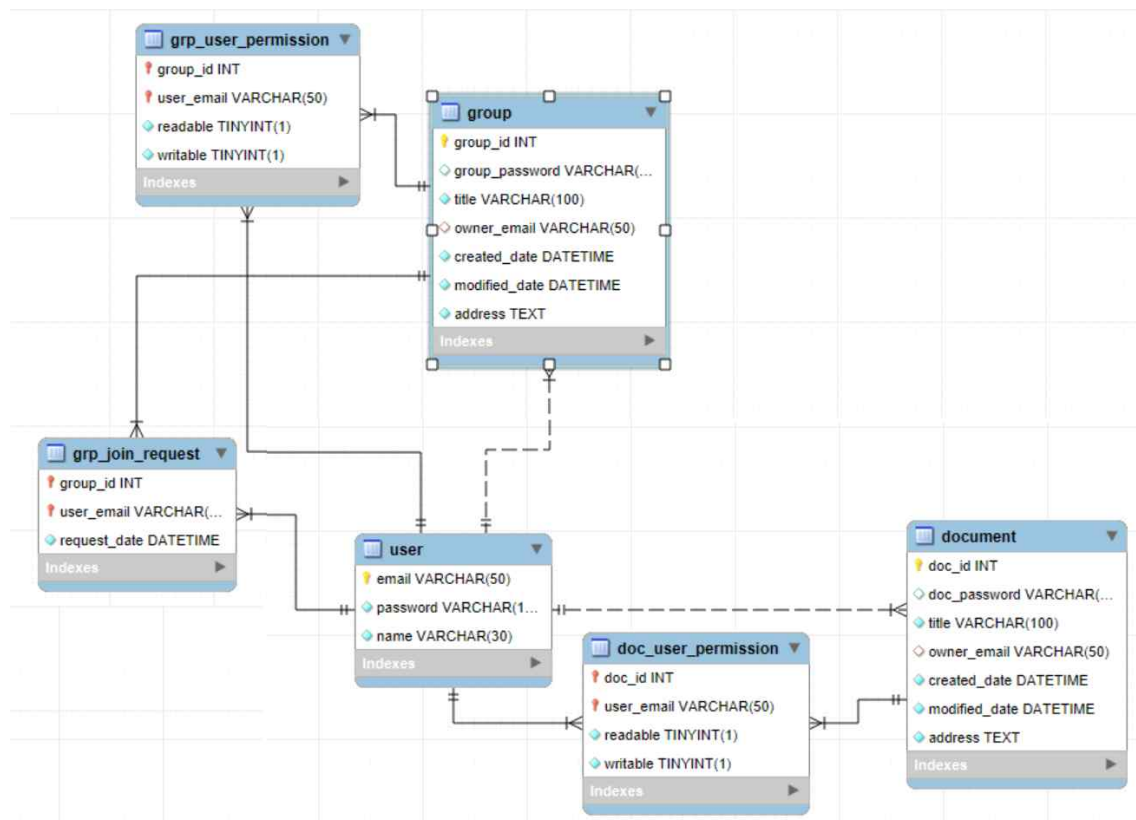
사실 이 그룹을 구현하면서 초기 설계의 문제점을 많이 느꼈습니다. 저희가 설계한 방식은 하나의 페이지 안에 페이지를 이동할 수 있는 메뉴 툴바와 그룹과 문서 편집 페이지로 분리하여 전체 페이지가 아닌 내부 페이지만 변경하는 구조였습니다. 초기 설계에 있어서 웹페이지 구조에 대한 이해가 없었기 때문에 내부 페이지를 단순히 <iframe>태그를 사용해서 새로운 윈도우를 만들어 구현을 했습니다. 그러다보니 Django Template 기반의 MPA(Multiple Page Application) 구조가 되었고 이는 Server Side Rendering으로 상위와 하위 iframe 이렇게 2개의 전체 페이지를 서버로부터 가져와서 제공해야하기 때문에 로딩 속도에 있어서 이점을 갖지 못했으며 새로 고침에 대한 history 조작이 난해해지는 문제뿐만 아니라 매번 2개의 윈도우를 서로 동기화시키기 위한 postMessage로 통신을 사용해야 했기 때문에 복잡하고 불편한 점이 많았습니다. MPA 구조는 검색 엔진과 같은 한 번에 가져오는 웹 페이지에 적합했고, 저희가 구현하고자하는 웹 앱은 일부 리소스에 대해 api로 변경할 수 있는 SPA 구조로 진행을 했어야 했습니다. 이에 대한 구조를 알았을 때는 이미 구현이 많이 진행된 뒤였고 수정하기에는 시간이 부족하다고 판단되어 구조로 인해 해결할 수 없는 문제점들은 그대로 두고 변경할 수 있는 부분들을 활용하고자 했습니다.

2. 그룹의 구현

단순히 DB만으로 그룹을 표현하기에는 부족한 부분이 많았습니다. 특히 iframe으로 상위 페이지와 하위 페이지로 구분된 상황에서는 하위 페이지가 변경되었을 때에 그 페이지가 같은 그룹의 속해있는지 여부도 생각을 해야 했고, 그룹 내부의 멤버들끼리 같은 문서를 공유하기 위해서는 그룹의 변경사항을 실시간으로 반영되어야 했으며 보안 문제를 위한 권한도 필요했습니다. 그 결과 DB스키마(그룹, 권한) + 세션(동일 그룹 판단) 및 Message 통신 + 웹 소켓(변경사항 실시간 반영) 이렇게 3가지를 조합한 방법이 나왔습니다.

i) DB 스키마

Django는 MVC 모델을 지원하며 내부적으로 SQL 쿼리문 작성 없이도 모델을 작성하면 테이블이 ORM 기능에 의해 Python 코드에 매핑됩니다.



ii) 그룹 세션이 존재할 때

1. 강제 퇴장

그룹 소유자에 의해 강퇴 처리가 된 경우 default 페이지로 이동시키는데 이 때 그룹 권한의 유무로 강퇴 여부를 판단합니다.

2. Normal Case

그룹 세션에 있는 데이터로 페이지 로드 필요한 데이터들을 만들어 보냅니다. 파라미터에는 isRedirect(강퇴 시 true), grp_owner(그룹 소유자), writable(쓰기모드인지), group_id(그룹id), doc_name(문서이름), join_requests(그룹 참가 요청 목록), memberList(그룹 멤버목록)가 있습니다.

3. 그룹 초기 입장 시

그룹에 액세스할 때 입력했던 group_id와 password 정보로 db에서 교차검증을 수행합니다. 일치한다면 그룹 세션을 만들고 위의 그룹 세션이 존재할 때와 동일한 데이터들을 만들어냅니다. 그룹의 첫 생성인 경우 문서를 생성합니다.

4. 그룹 세션 생성

그룹 세션은 명시적으로 처음 비밀번호를 입력하고 그룹에 액세스할 경우 세션을 생성합니다.

5. 세션 삭제 시점

상위 페이지인 home에서 iframe의 페이지가 변경될 때마다 현재 페이지가 그룹과 관련된 페이지인지 확인을 하고 그룹페이지가 아니면 세션을 끊기 명령에 대한 비동기 요청을 수행합니다. 이 때 편집 페이지를 벗어나게 되면 세션이 삭제됩니다.

iii) 그룹 실시간 동기화

Django와 연동할 수 있는 모듈인 Django Channels를 사용하여 웹 소켓을 구현합니다. Python은 HTTP의 프로토콜을 통신 구현하기 위한 middleware로 WSGI라는 인터페이스를 사용하는데 이와 구분하기 위해 비동기 인터페이스인 ASGI에 웹 소켓인 WS 프로토콜을 추가로 정의합니다.

-부모 창(home)과 자식 창(iframe) Message 통신

보안 문제 상 부모와 자식 서로가 각각의 윈도우에서 이루어지는 기능에 대한 직접적인 관여를 할 수 없기 때문에 상호 간의 Message로 통신을 할 필요가 있었고 통신할 때마다 보내는 데이터가 다르기 때문에 일반화를 할 필요성을 느꼈습니다. 따라서 임의로 부모와 자식창의 통신 규칙을 만들었습니다. type은 method와 data로 나뉩니다. method의 경우 method와 param 필드에서 값을 가져오고, data의 경우 말 그대로 data 필드에서 값을 가져오도록 구성했습니다. 세션으로 연결되어있기 때문에 값을 가져오는 일보다는 어떤 동작을 수행하도록 요청하는 일이 빈번합니다. 부모 창

과 자식 창의 통신이 필요한 가장 주된 이유는 그룹 접속 시 발생하는 웹 소켓의 통신과 직접적인 연관이 있고 권한의 변경되었을 때, 강제퇴장 시, 문서의 생성과 삭제 등의 전반적인 문서 작업과 그룹 관리 작업에 대한 “실시간 반영” 때문이었습니다. 이 또한 MPA 구조를 채택해서 생기는 복잡했던 이유 중 하나이기도 했습니다.

- 채팅

채팅은 Node.js를 기반으로 Express, socket.io를 활용하여 실시간 소통이 가능하게 제작하였습니다.

채팅 기능에서 아쉬웠던 점은, 메시지를 보낸 후, 나갔다 들어오면 이전에 주고받았던 채팅내용이 저장되지 않는다는 점과, 새로운 메시지가 왔는지 확인하려면 직접 채팅 화면으로 들어가봐야 할 수 있다는 점입니다.

-문서 편집기

문서 편집기는 코드 편집기와 마찬가지로 yorkie를 사용하여 실시간 편집을 가능하게 합니다. 같은 api키를 통해 yorkie 서버와 통신합니다.

다른 점은, 문서 편집기는 quill[3]이라는 라이브러리를 사용하여 편집 환경을 제공하고, 사용할 도구들을 제공합니다. 도구에 존재하지 않는 PDF 변환은 quill-to-pdf 라는 라이브러리를 사용하였습니다.

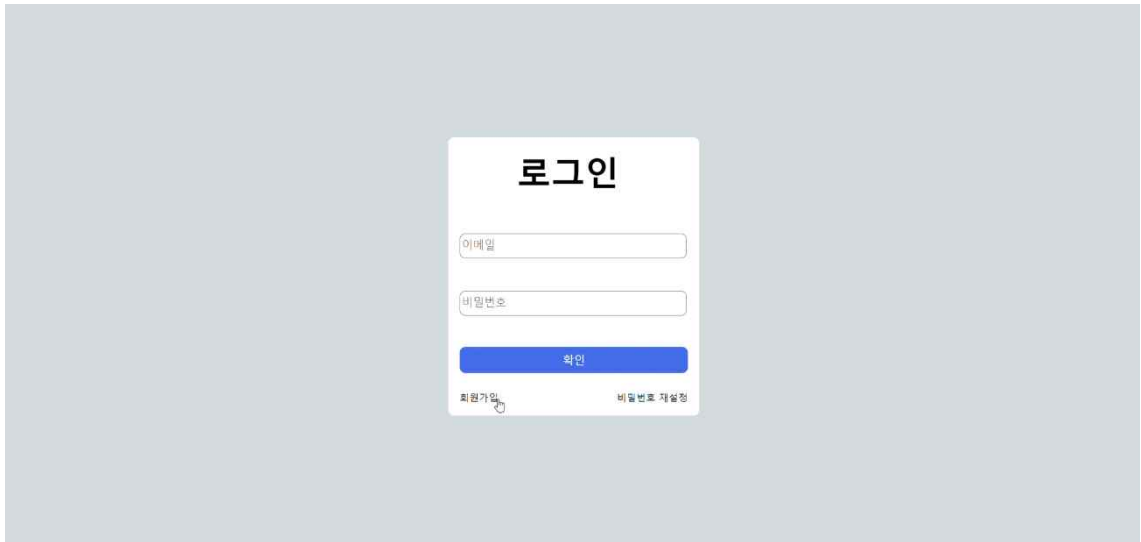
문서 편집기는 마우스 커서가 선택한 부분을 추적하여, 다른 사용자와 본인의 마우스 커서가 어느 부분을 클릭하여 사용하고 있는지 보여줍니다. 그리고 특이한 점은, 실시간 반영을 위한 수정할 텍스트, 수정된 텍스트의 위치를 추적하여 알고리즘을 통해 편집합니다.

-GPT 연동 분석 기능

1. openAI API를 추가하여 chatGPT에 전송하기 위한 DTO를 생성합니다.
2. 코드 또는 문서 편집 창이 열려져 있는 경우 그곳에 있는 내용을 가져와 요약 또는 코드 분석을 할 수 있도록 명령기능을 별도로 추가합니다.
3. 사용자가 입력한 내용을 분석해서 '/'이 맨앞에 들어가면 명령으로 인식합니다.
4. 명령에 따라 GPT에서 문서 또는 코드를 분석하여 응답합니다.

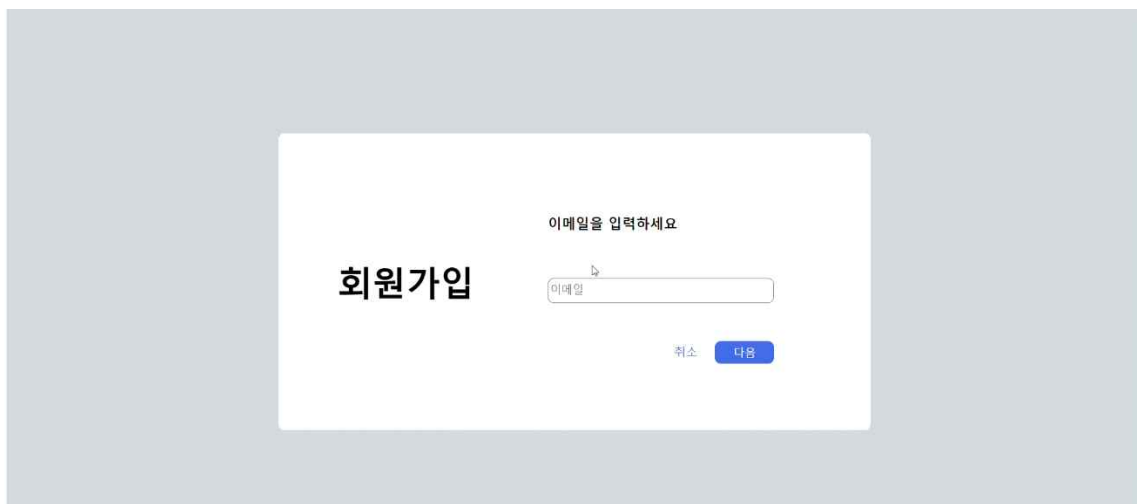
● 결과 화면

- 로그인 화면, 회원가입



The image shows a login screen with a white card centered on a light gray background. The card has the title "로그인" (Login) at the top. Below the title are two input fields: "이메일" (Email) and "비밀번호" (Password). A blue button labeled "확인" (Confirm) is positioned below the password field. At the bottom left of the card is a link for "회원가입" (Sign Up), and at the bottom right is a link for "비밀번호 재설정" (Reset Password).

처음 프로그램일 실행하면 보이는 화면입니다. 이전에 회원가입을 진행하여 DB에 정보가 존재한다면 이메일과 비밀번호를 입력하여 접속할 수 있습니다. 만약 정보가 존재하지 않는다면 회원가입을 통해 정보를 생성할 수 있습니다.



The image shows a sign-up screen with a white card centered on a light gray background. The card has the title "회원가입" (Sign Up) on the left. On the right, there is a prompt "이메일을 입력하세요" (Enter your email) above an "이메일" (Email) input field. Below the input field are two buttons: "취소" (Cancel) and "다음" (Next).

위의 화면에서 이메일을 입력하여 인증을 진행하고, 인증 완료 후, 비밀번호를 입력하여 정보를 생성합니다.

- home 화면

화면 가운데에 보이는 내용이 들어가지 않은 그룹을 나타내는 부분입니다.



만약 이전에 그룹에 가입한 적이 있다면, '내 그룹' 버튼을 클릭하여, 가입한 그룹을 볼 수 있습니다.



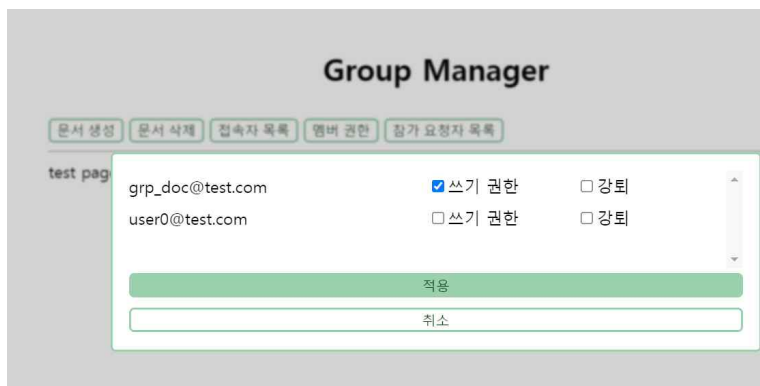
가입한 그룹이 없다면 해당 그룹을 클릭하여, 요청을 보낼 수 있습니다.

-그룹

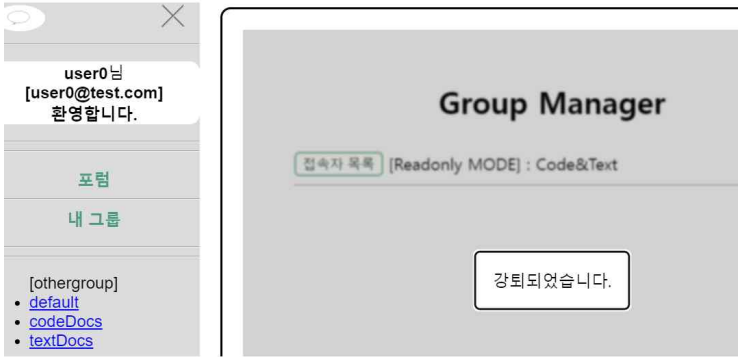
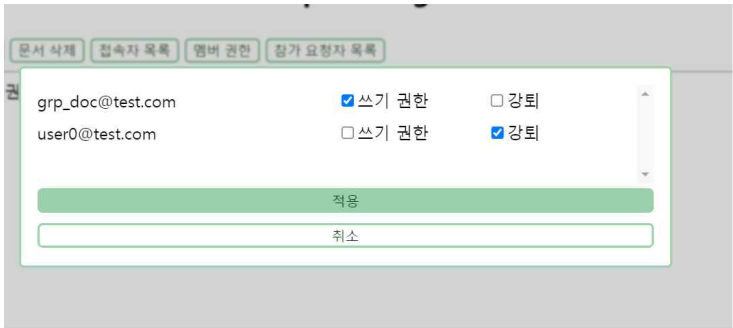
자신의 그룹으로 들어오게 되면, 위의 사진처럼 그룹에 가입되어 있고, 현재 접속중인 인원의 정보를 볼 수 있습니다.



그룹장은 그룹원의 권한을 지정할 수 있습니다. 쓰기 권한을 주지 않으면 아래 화면처럼 'Readonly' 라는 문구가 보이게 되고, 해당 그룹원은 코드, 문서 편집기에서 작성, 수정을 진행할 수 없습니다.

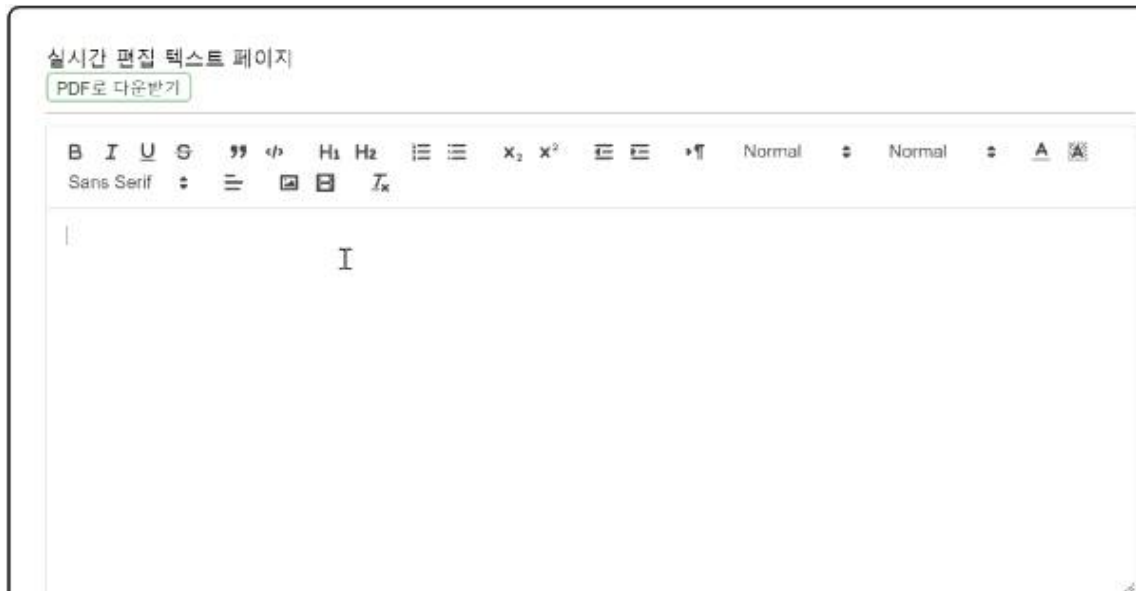


그룹장은 그룹원을 강퇴할 수 있습니다. 아래의 사진처럼 진행하면 해당하는 그룹원은 더 이상 그룹에 접속할 수 없습니다.

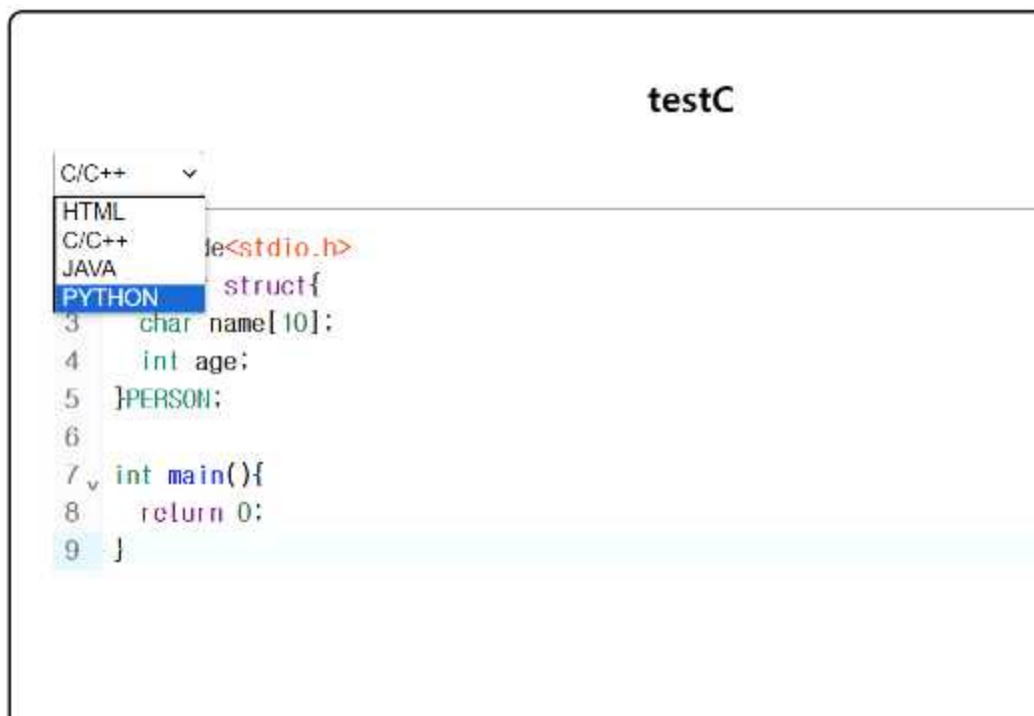


- 편집기

문서 페이지입니다. 위의 화면에서 문서를 작성할 수 있습니다. 위의 기능에 대해서 설명했던 것처럼 입력, 수정, 삭제는 모두 실시간으로 다른 사용자에게 반영됩니다.



코드 페이지입니다. 위의 화면에서 코드를 작성할 수 있습니다. 화면에서 보이는 것처럼, 작성할 코드의 종류를 선택 지정할 수 있습니다. 지정한 코드 유형에 따라, 입력하는 코드의 하이라이트가 자동 반영됩니다. 이 또한, 실시간으로 다른 사용자에게 반영됩니다.



- 채팅

채팅 화면입니다. 아래의 텍스트박스에 상대방에게 보낼 메시지를 입력하고, Enter 키 혹은 전송 버튼을 클릭하면 메시지가 전송됩니다.
자신이 보낸 메시지는 하얀색 색상이 지정되고, 상대방이 보낸 메시지는 파란색으로 지정됩니다.

The image displays two side-by-side chat windows. The left window has a dark gray background and contains two messages: a blue bubble on the left with the text '김태호: Hello' and a white bubble on the right with the text 'Hi'. The right window also has a dark gray background and contains two messages: a white bubble on the left with the text 'Hello' and a blue bubble on the right with the text '김규택: Hi'. At the bottom of each window is a white text input field with the placeholder text '메시지를 입력해주세요...' and a small button labeled '전송'.

- **GPT**

gpt 채팅 화면입니다. 텍스트박스에 상대방에게 보낼 메시지를 입력하고, Enter키 또는 전송 버튼을 클릭하면 메시지가 전송됩니다. 사용자의 메시지는 우측에 표시되고, chat gpt의 메시지는 좌측에 표시됩니다.

