



개발자를 위한 소통형 미니 클라우드

Team 2: 강승윤, 김우현, 박현욱, 배지훈, 이동훈



프로젝트 추진 배경



접근성

직관적인 UI로
누구나 쉽게 이용



편의성

파일 공유와
동시에 소통



기능성

파일 별 페이지로
유지보수 용이

프로젝트 구성원



강승윤

- GUI 로그인 및 그룹창 구현
- ppt 제작



김우현

- GUI 파일 페이지 구현
- 프로젝트 발표



배지훈

- 백엔드 UI 기능 구현
- 전체 코드 리팩토링



박현욱

- GUI 파일 페이지 구현 및 전체 디자인 통일
- 프로젝트 발표



이동훈


- 데이터베이스 구현
- 프로젝트 기본구조 설계

개발 환경

OS :  Windows 11

Language :



IDE : 



프로젝트 코드 요약

코드 라인

- 3323 Lines
(주석 및 공백 포함)

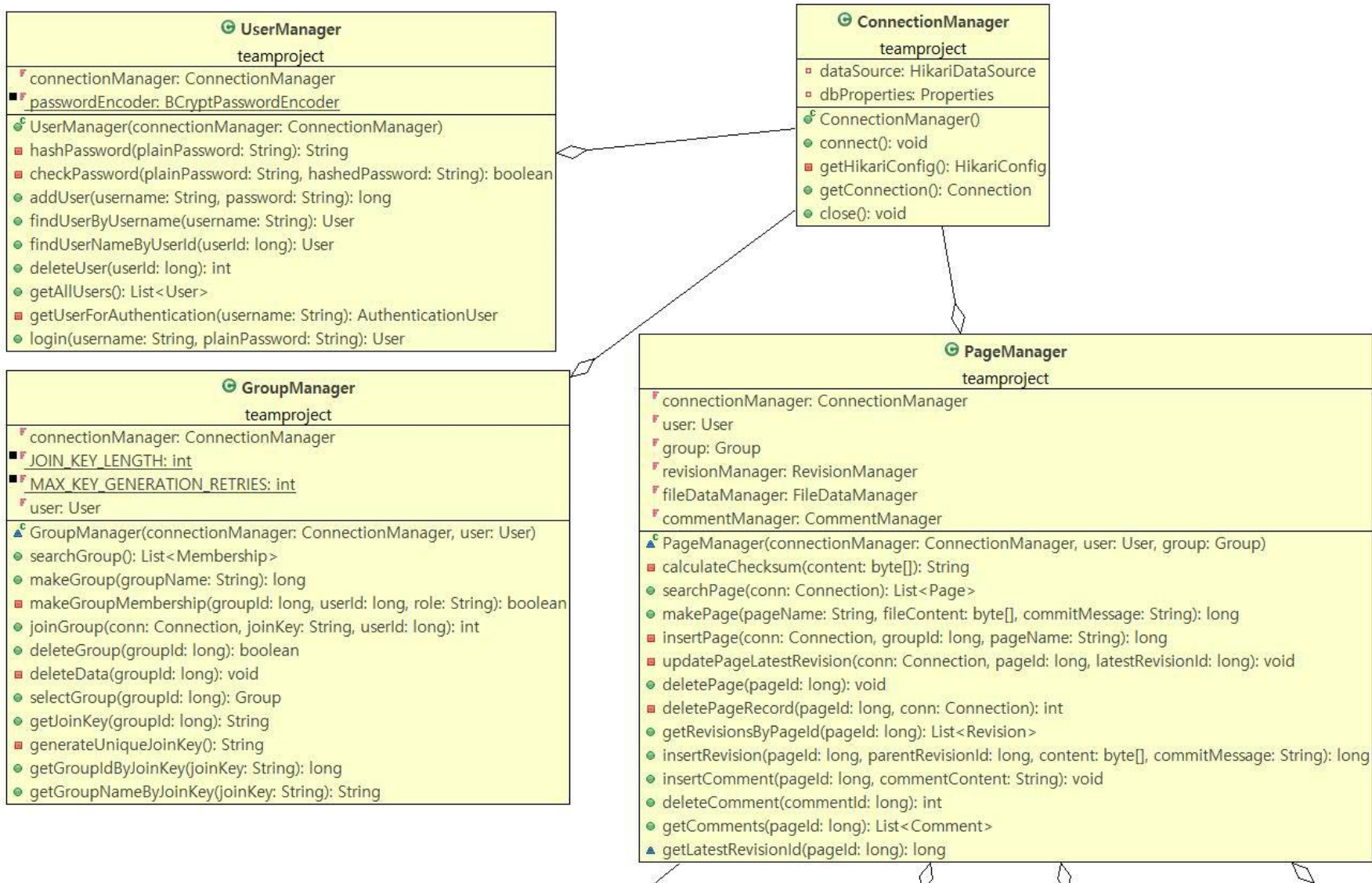
기능 구현

- 제네릭 및 컬렉션 : ○ / X
- 상속 : ○ / X
- 예외처리 : ○ / X
- 파일 입출력 : ○ / X

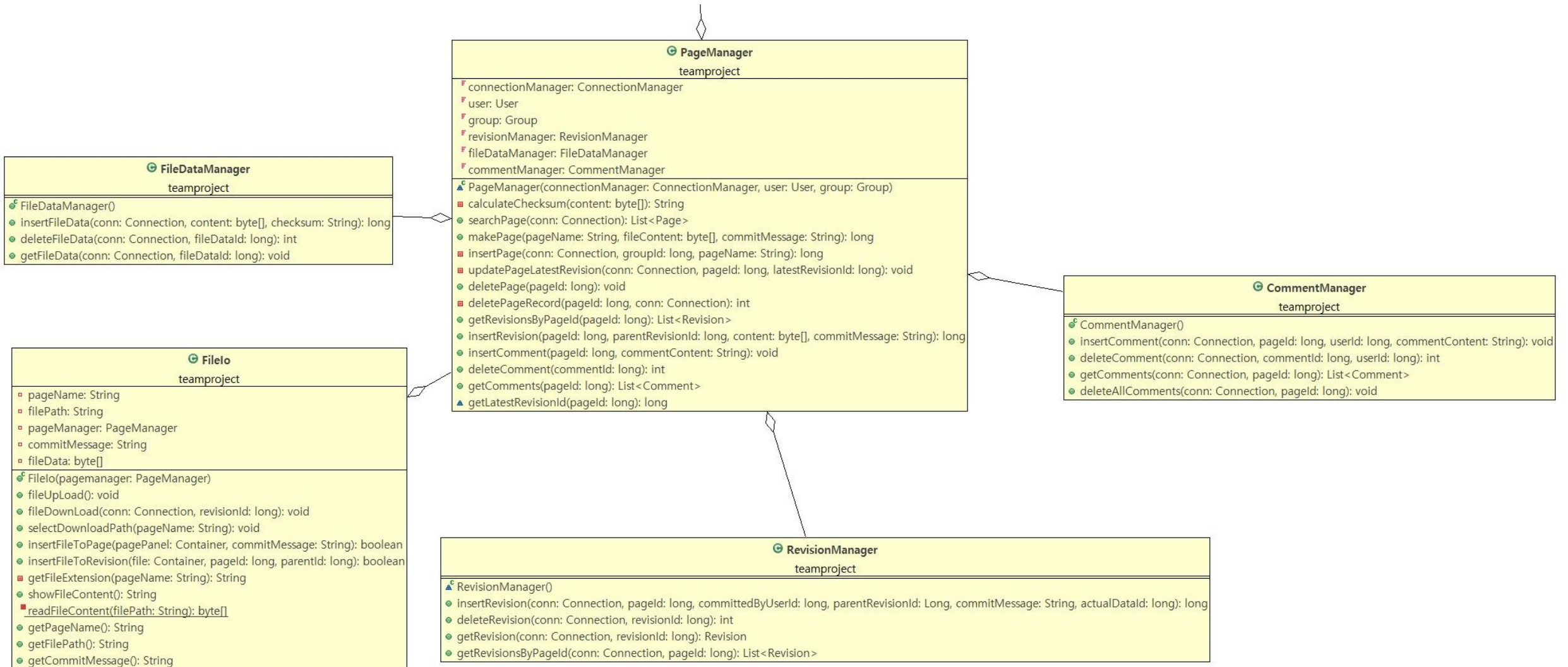
라이브러리

- commons-logging-1.2
- HikaryCP-5.1.0
- mysql-connector-j-9.3.0
- slf4j-api-2.0.9
- slf4j-simple-2.0.9
- spring-security-crypto-6.4.4

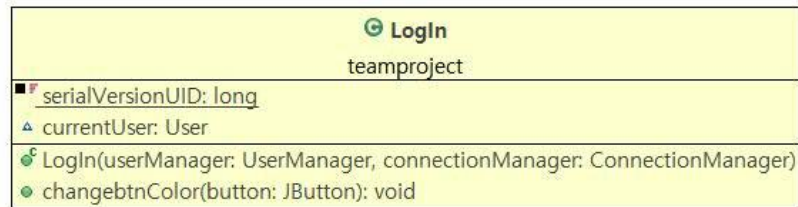
클래스 다이어그램 (백엔드)



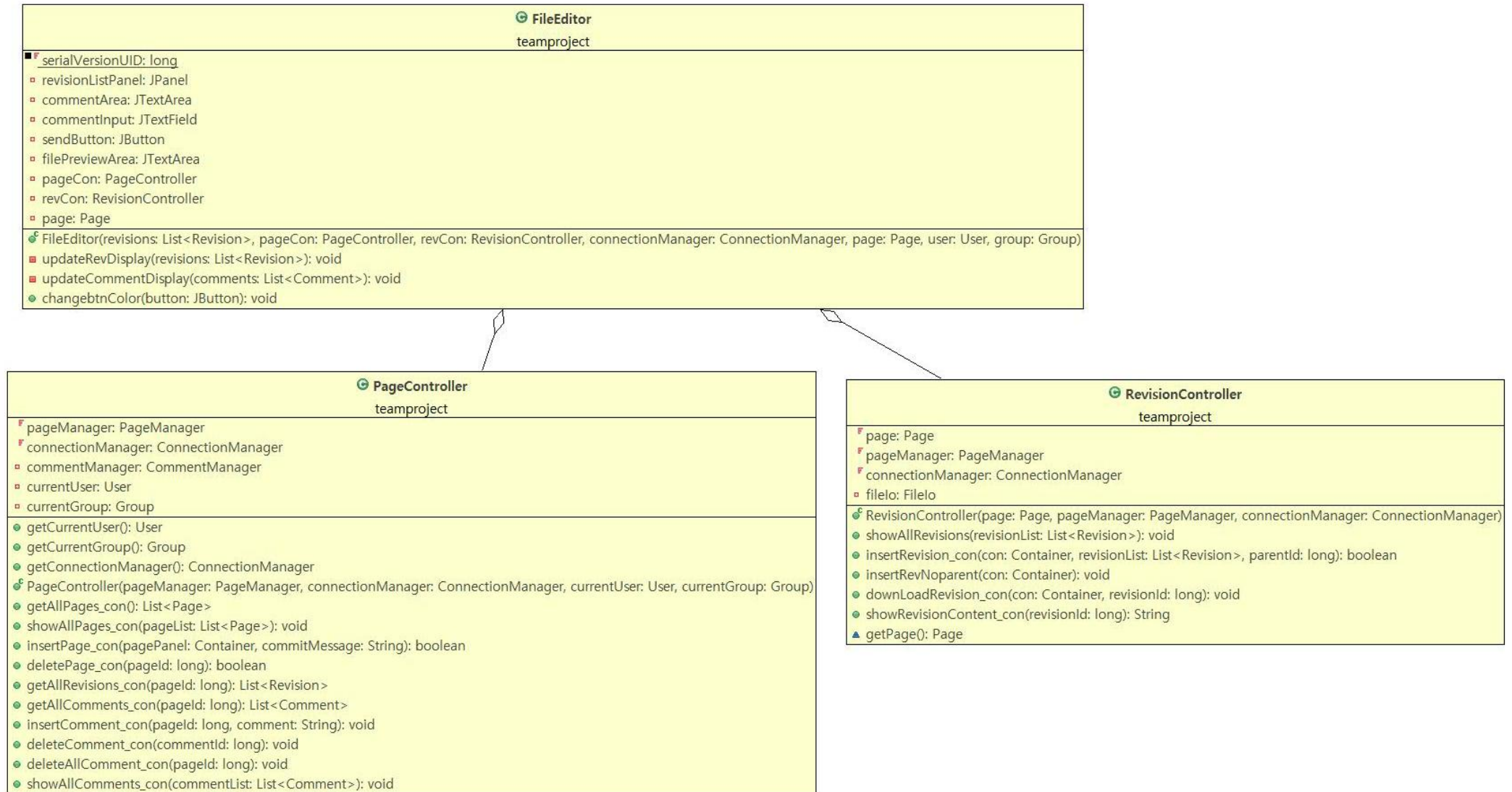
클래스 다이어그램 (백엔드)



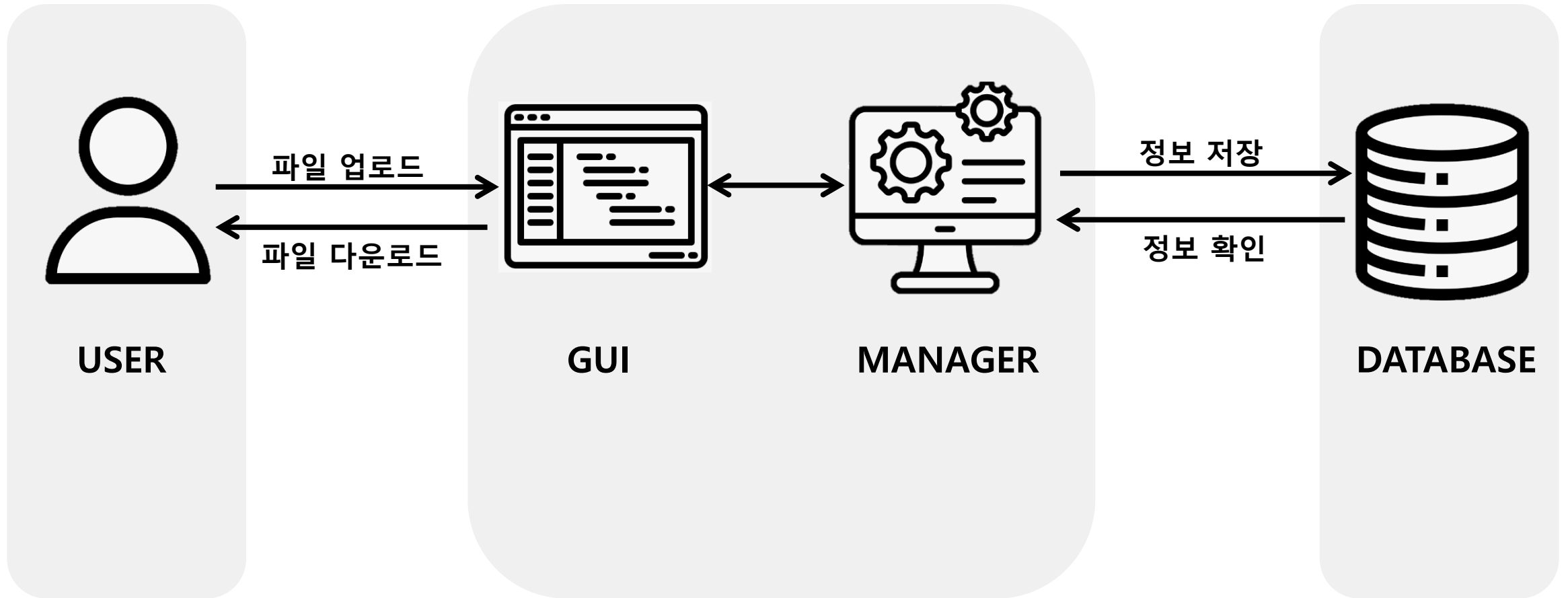
클래스 다이어그램 (프론트엔드)



클래스 다이어그램 (프론트엔드)



시스템 구성도



설계 구성 요소

구성요소 항목	항목에 대한 설명	설계과정에서 아래 항목을 다룬 이유	만족 여부
목표 설정	해결할 문제를 구체적으로 정함	개발자들이 서로 피드백을 주고 받으며 파일을 공유할 수 있는 공간을 만들고자 함	○
합성	목표 달성에 필요한 세부 기능과 시스템 구조 설계	파일을 저장 및 공유할 수 있는 데이터베이스를 설계하고 이후 자바를 활용하여 파일 처리가 가능하게끔 구현	○
분석	SW 구현 과정 중의 문제점 분석 및 해결	여러 사용자들이 접속할 수 있는 데이터베이스의 장단점을 고려하여 MySQL을 선정	○
제작	설계에 따라 SW 구현	MySQL을 이용하여 데이터베이스 설계, Eclipse IDE를 이용하여 기능을 구현	○
시험	구현한 SW의 테스트	임의로 여러 사용자 및 그룹을 생성하여 파일 공유나 저장이 정상적으로 작동하는지 확인	○
평가	작품의 완성도, 기능, 기대효과 등 평가	과제나 소규모 프로젝트 등을 위해 누구나 쉽게 활용할 수 있을 것으로 기대	○

현실적 제한 조건

제한조건 항목	항목에 대한 설명	설계과정에서 아래 항목을 다룬 이유	만족 여부
경제성	개발에 필요한 비용 (도구, HW 구입 등) 고려	오픈소스 소프트웨어만을 사용하여 데이터베이스 확장에만 개발 비용이 드므로 저렴하게 이용 가능	○
안전성	외부 침해에 대한 고려 사항을 설계에 반영	로그인 정보를 property 파일에 저장하여 외부 접근을 불가능하게 만들었으며 사용자 비밀번호를 데이터베이스에 hashing해서 저장하여 보안을 강화	○
신뢰성	설치/실행 오류 최소화 하도록 SW 개발	프로그램의 안정성을 높이기 위하여 각 단계 별로 예외 처리를 함	○
외관성	SW 활용 시 편리성 고려해 외관 디자인 설계	자바 GUI를 활용하여 구현한 기능 및 순서를 사용하기 쉽게 가시화함	○
윤리성	불법 SW 및 콘텐츠 사용 자제	오픈소스 소프트웨어만으로 개발하였으며 라이선스 규정을 준수함	○
사회적 영향	결과물이 사회에 미치는 영향을 분석해 보고서에 포함	사용자들이 파일을 공유하고 의견을 나누며 원하는 프로젝트를 진행하는 데 편의성을 제공	○

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	USERS
테이블 설명	사용자 정보를 저장		
컬럼 이름	컬럼 타입	Key	Description
user_id	bigint unsigned	Primary	
user_name	varchar(255)		로그인 시 사용할 이름
password_hash	varchar(255)		해싱된 비밀번호 값
created_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	GROUP_DATA
테이블 설명	그룹 정보를 저장		
컬럼 이름	컬럼 타입	Key	Description
group_id	bigint unsigned	Primary	
group_name	varchar(255)		그룹 이름
join_key	varchar(255)		참가 키
created_by_user_id	bigint unsigned	Foreign references USERS(user_id)	그룹을 만든 사용자의 id값
created_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	GROUP_MEMBERSHIP
테이블 설명	그룹과 사용자 사이의 관계를 저장		
컬럼 이름	컬럼 타입	Key	Description
membership_id	bigint unsigned	Primary	
group_id	bigint unsigned	Foreign references GROUP_DATA (group_id)	그룹의 id값
user_id	bigint unsigned	Foreign references USERS(user_id)	사용자의 id값
user_role	varchar(50)		해당 그룹에서 사용자의 역할
joined_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	PAGES
테이블 설명	그룹 내의 페이지에 대한 정보를 저장		
컬럼 이름	컬럼 타입	Key	Description
page_id	bigint unsigned	Primary	
group_id	bigint unsigned	Foreign references GROUP_DATA (group_id)	해당 페이지가 속한 그룹의 id 값
page_name	varchar(255)		페이지 이름
latest_revision_id	bigint unsigned	Foreign references FILE_REVISIONS(revision_id)	가장 최근에 변경된 개정판의 id값
created_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	FILE_REVISIONS
테이블 설명	페이지 내 파일 개정판에 대한 메타데이터를 저장		
컬럼 이름	컬럼 타입	Key	Description
revision_id	bigint unsigned	Primary	
page_id	bigint unsigned	Foreign references PAGES(page_id)	해당 개정판이 속한 페이지의 id값
actual_data_id	bigint unsigned	Foreign references FILE_DATA (actual_data_id)	실제 데이터의 id값
committed_by_user_id	bigint unsigned	Foreign references USERS(user_id)	파일을 수정한 사용자의 id값
parent_revision_id	bigint unsigned	Foreign references FILE_REVISIONS(revision_id)	이전 개정판의 id값
commit_message	text		등록 메시지
created_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	FILE_DATA
테이블 설명	실제 파일 데이터를 저장		
컬럼 이름	컬럼 타입	Key	Description
actual_data_id	bigint unsigned	Primary	
content	blob		파일 데이터
checksum	varchar(64)		체크섬 값
created_at	timestamp		데이터베이스에 등록된 시간

테이블 명세서(테이블 정의서)

데이터베이스 이름	teamproject	테이블 이름	COMMENTS
테이블 설명	댓글 데이터를 저장		
컬럼 이름	컬럼 타입	Key	Description
comment_id	bigint unsigned	Primary	
page_id	bigint unsigned	Foreign references PAGES(page_id)	해당 댓글이 달린 페이지의 id 값
commented_by_user_id	bigint unsigned	Foreign references USERS(user_id)	해당 댓글을 단 사용자의 id값
commented_by_user_name	varchar(255)	Foreign references USERS(user_name)	해당 댓글을 단 사용자의 이름
comment_data	text		댓글 내용
created_at	timestamp		데이터베이스에 등록된 시간

프로젝트 내용(내부 클래스)

```
private static class AuthenticationUser { 3개 사용 위치
    private final long userId; 2개 사용 위치
    private final String username; 2개 사용 위치
    private final String passwordHash; 2개 사용 위치

    public AuthenticationUser(long userId, String username, String passwordHash) { 1개 사용 위치
        this.userId = userId;
        this.username = username;
        this.passwordHash = passwordHash;
    }

    public long getUserId() { return userId; } 0개의 사용위치
    public String getUsername() { return username; } 1개 사용 위치
    public String getPasswordHash() { return passwordHash; } 1개 사용 위치
}
```

- 사용자가 로그인 시 입력한 비밀번호 평문과, 해싱되어 데이터베이스에 저장된 값을 비교할 때 사용
- 내부 클래스에 해싱된 비밀번호를 받아온 후, 비밀번호를 비교하는 메소드로 전달
- 외부에서 이 값에 접근할 수 없어야 하므로 내부 클래스를 사용하여 구현

로그인 -> 그룹

Log In / Register

Sharing Cloud

ID :

PW :

로그인

회원가입

회원가입

ID :

PW :

가입하기

취소

->

Sharing Cloud

그룹 ↓

홍길동님의 C언어 공부방 X

join key: Rck8Df26Lp

모두의 코드 X

join key: Wg76aiCSLF

초보자들 환영!! X

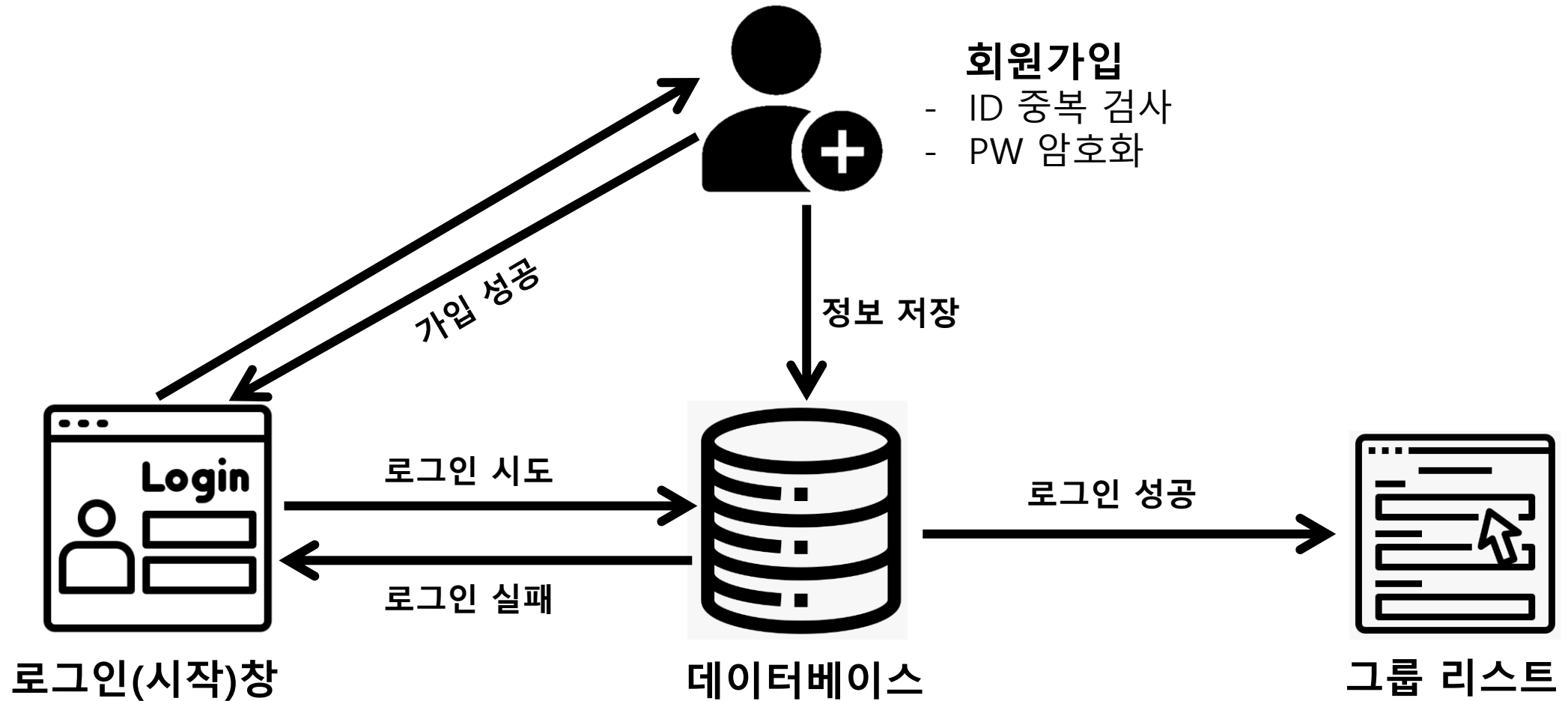
join key: Ar0Q7HbSBw

그룹 생성

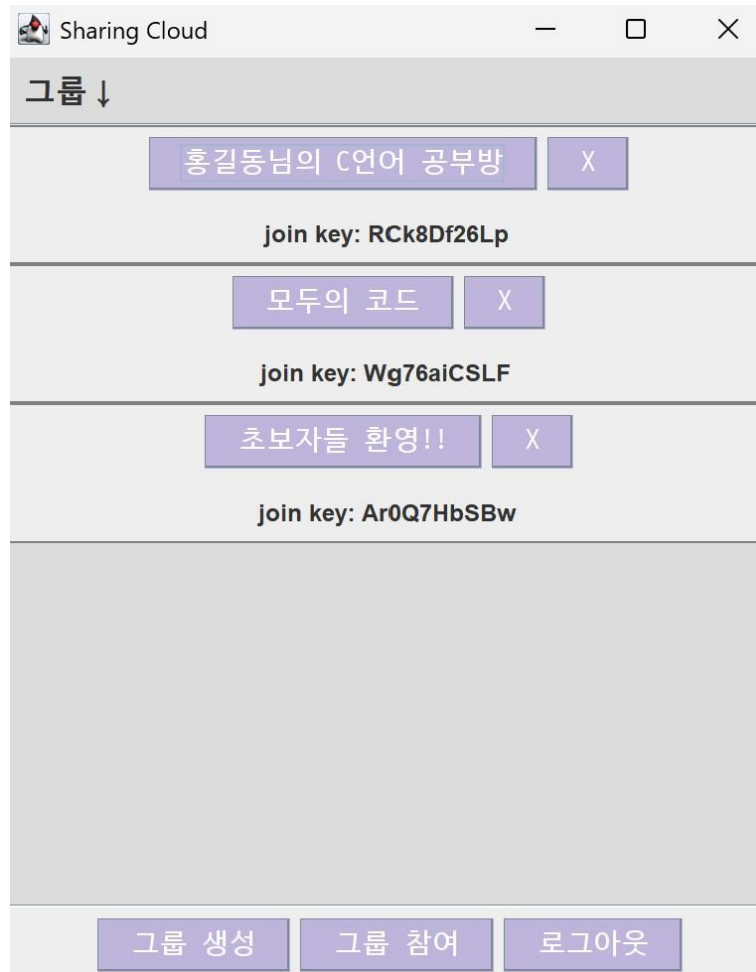
그룹 참여

로그아웃

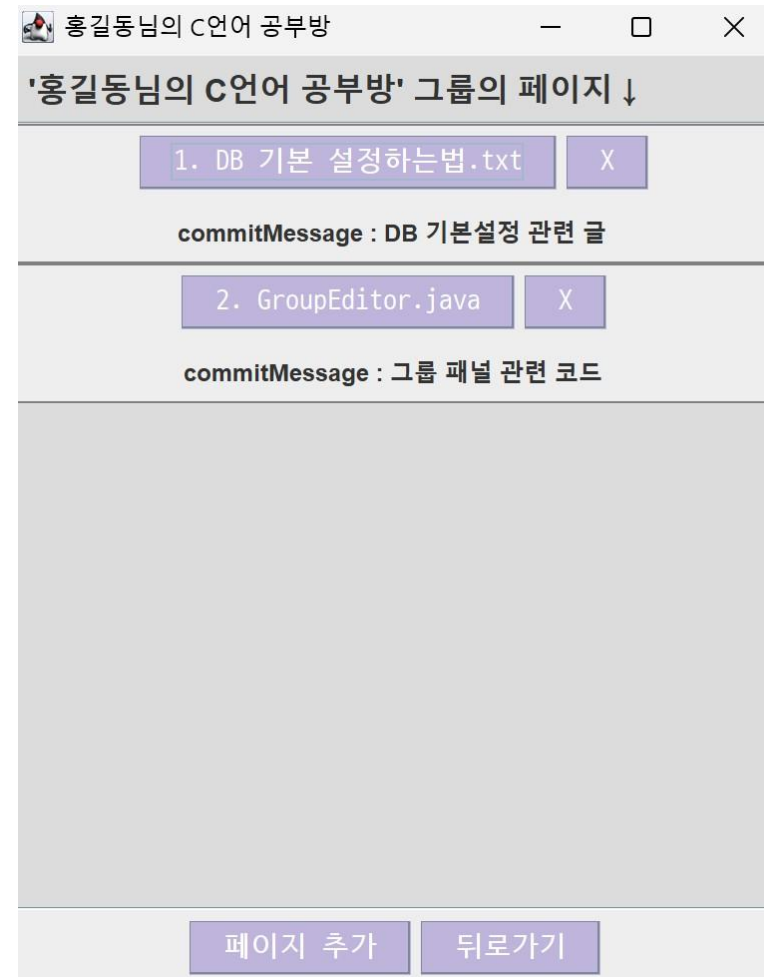
로그인 -> 그룹



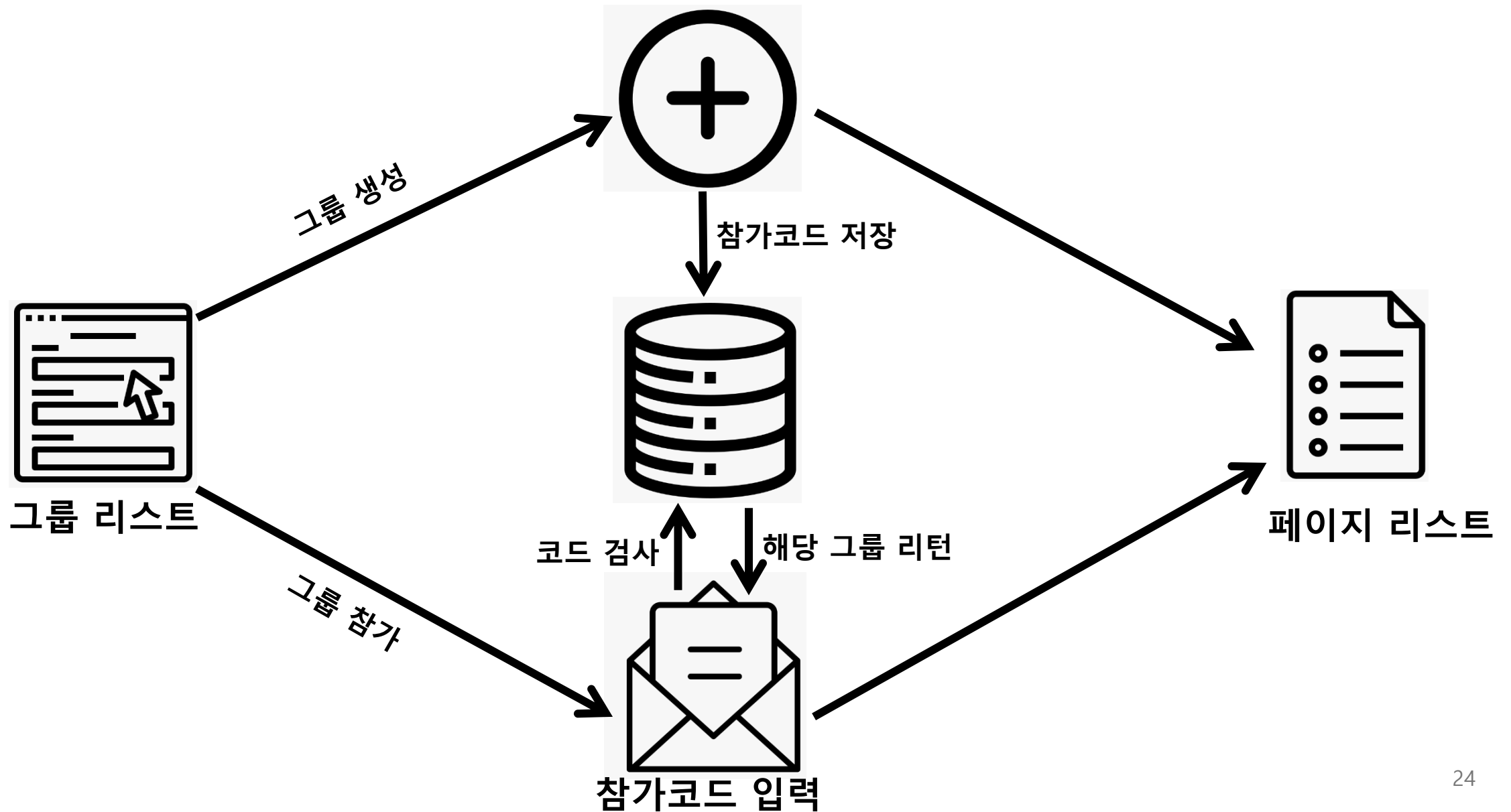
그룹 -> 페이지



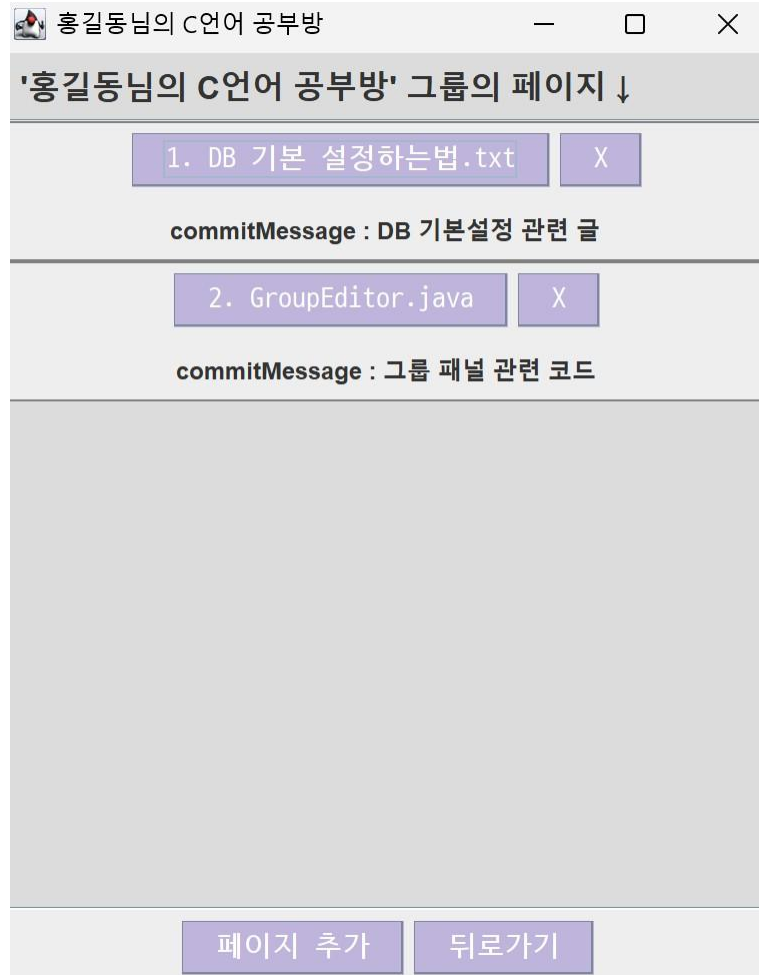
->



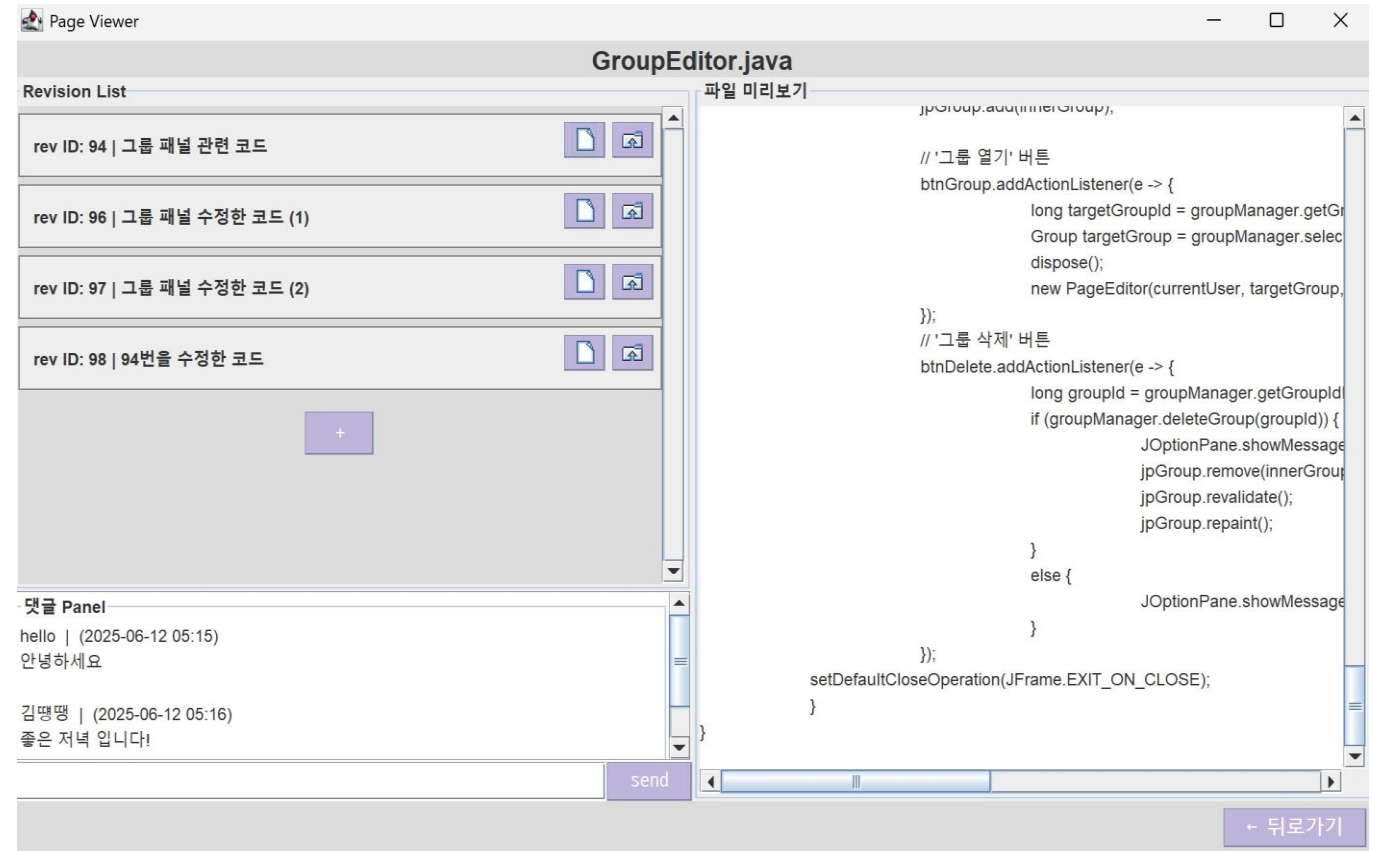
그룹 -> 페이지



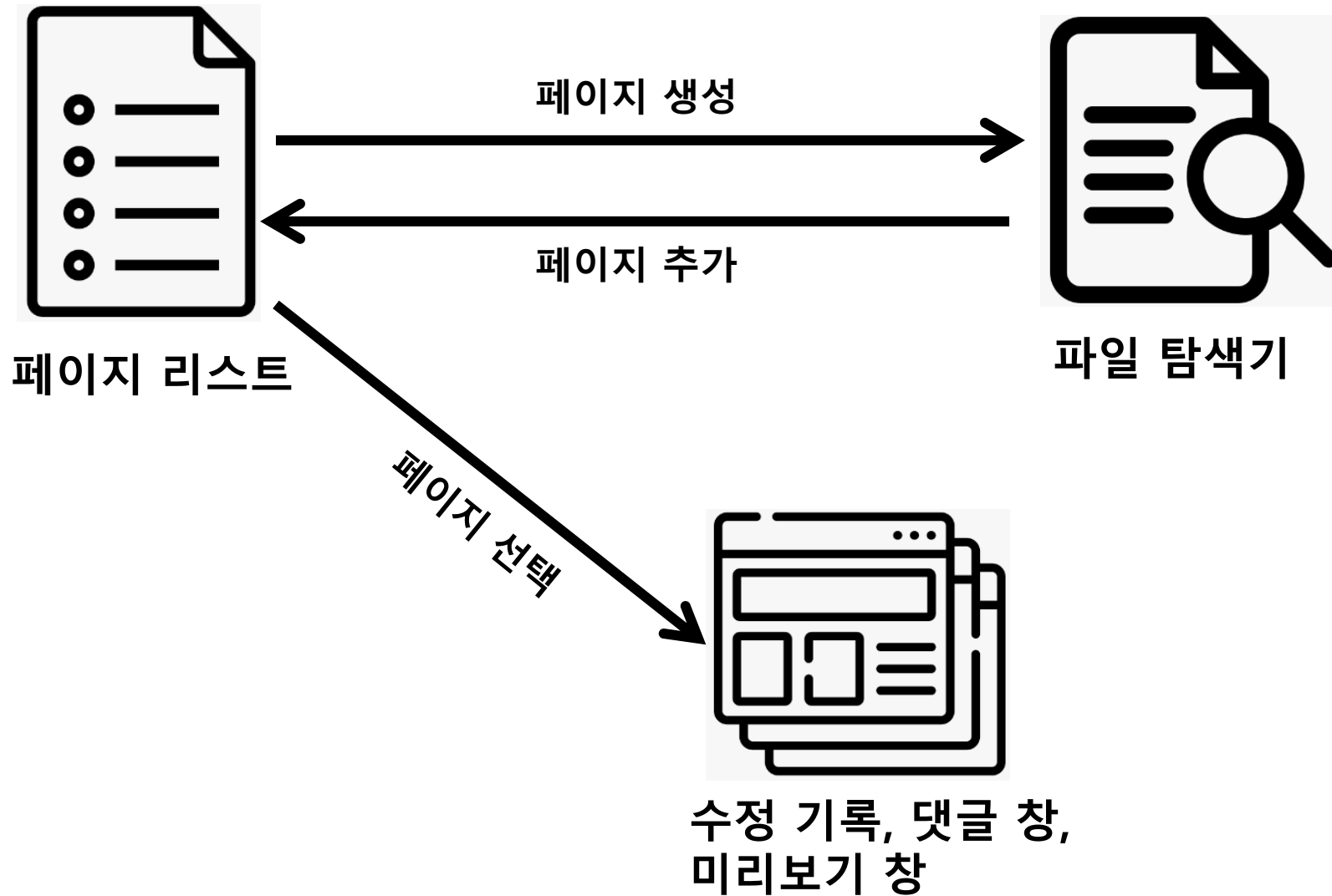
페이지 -> 리비전



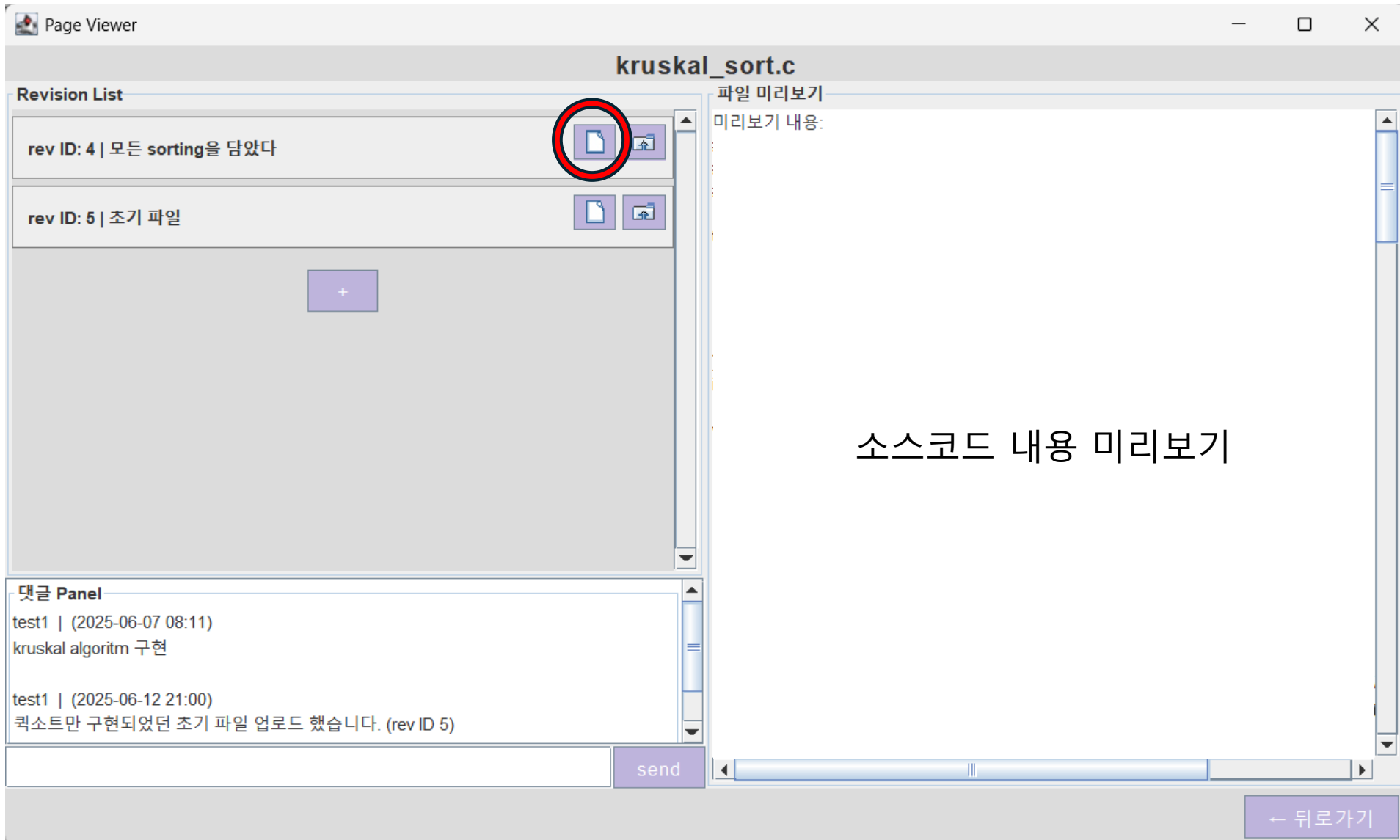
->



페이지 -> 리비전



리비전 화면 (파일 미리보기)



리비전 화면 (파일 미리보기)

Page Viewer

kruskal_sort.c

Revision List

rev ID: 4 | 모든 sorting을 담았다

rev ID: 5 | 초기 파일

+

댓글 Panel

test1 | (2025-06-07 08:11)
kruskal algorithm 구현

test1 | (2025-06-12 21:00)
퀵소트만 구현되었던 초기 파일 업로드 했습니다. (rev ID 5)

send

파일 미리보기

미리보기 내용:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_NODE 20

typedef struct {
    int preNode;
    int postNode;
    int weight;
    int isSelected;
} edge;
int parent[MAX_NODE];

void quickSort(edge edges[], int left, int right) {
    int pivot, i, j;
    edge temp;

    if (left < right) {
        i = left;
        j = right + 1;
        pivot = edges[left].weight;
        //edges[right + 1].weight = 100;
        do {
            do i++; while (edges[i].weight < pivot);
            do j--; while (edges[j].weight > pivot);
            if (i < j) {
                temp = edges[i];
                edges[i] = edges[j];
                edges[j] = temp;
            }
        } while (i < j);
        temp = edges[left];
        edges[left] = edges[j];
        edges[j] = temp;
        quickSort(edges, left, j - 1);
        quickSort(edges, j + 1, right);
    }
}
```

← 뒤로가기

리비전 화면 (파일 다운로드)

The screenshot displays a web application window titled "Page Viewer" showing the revision history and content of a file named "kruskal_sort.c".

Revision List:

- rev ID: 4 | 모든 sorting을 담았다 (Icon: Download)
- rev ID: 5 | 초기 파일 (Icon: Download)

A red circle highlights the download icon for revision 4.

댓글 Panel (Comment Panel):

- test1 | (2025-06-07 08:11)
kruskal algorithm 구현
- test1 | (2025-06-12 21:00)
퀵소트만 구현되었던 초기 파일 업로드 했습니다. (rev ID 5)

파일 미리보기 (File Preview):

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_NODE 20

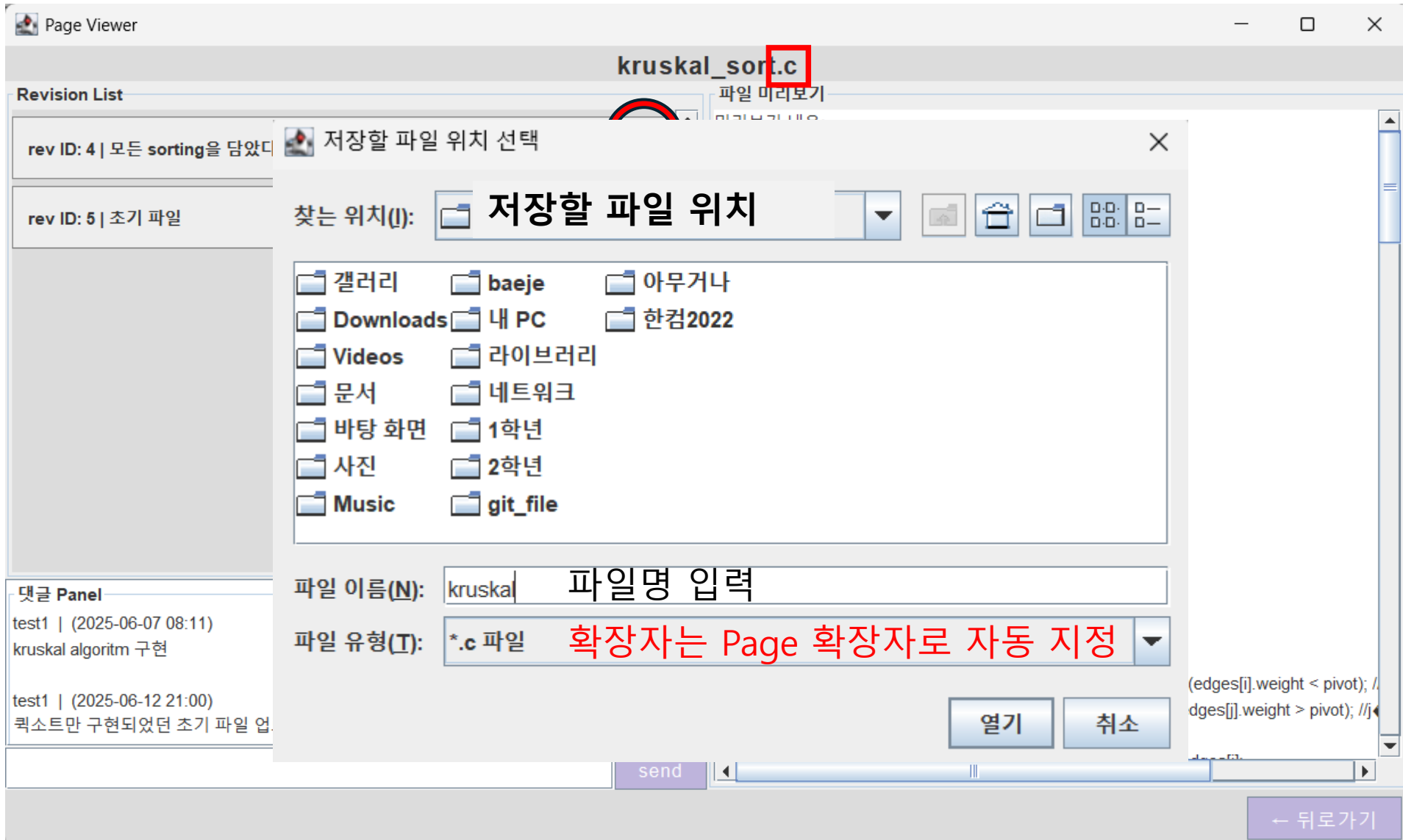
typedef struct {
    int preNode;
    int postNode;
    int weight;
    int isSelected;
} edge;
int parent[MAX_NODE];

void quickSort(edge edges[], int left, int right) {
    int pivot, i, j;
    edge temp;

    if (left < right) {
        i = left;
        j = right + 1;
        pivot = edges[left].weight;
        //edges[right + 1].weight = 100;
        do {
            do i++; while (edges[i].weight < pivot);
            do j--; while (edges[j].weight > pivot);
            if (i < j) {
                temp = edges[i];
                edges[i] = edges[j];
                edges[j] = temp;
            }
        } while (i < j);
        temp = edges[left];
        edges[left] = edges[j];
        edges[j] = temp;
        quickSort(edges, left, i - 1);
        quickSort(edges, i + 1, right);
    }
}
```

Buttons at the bottom: "send" and "← 뒤로가기" (Back).

리비전 화면 (파일 다운로드)

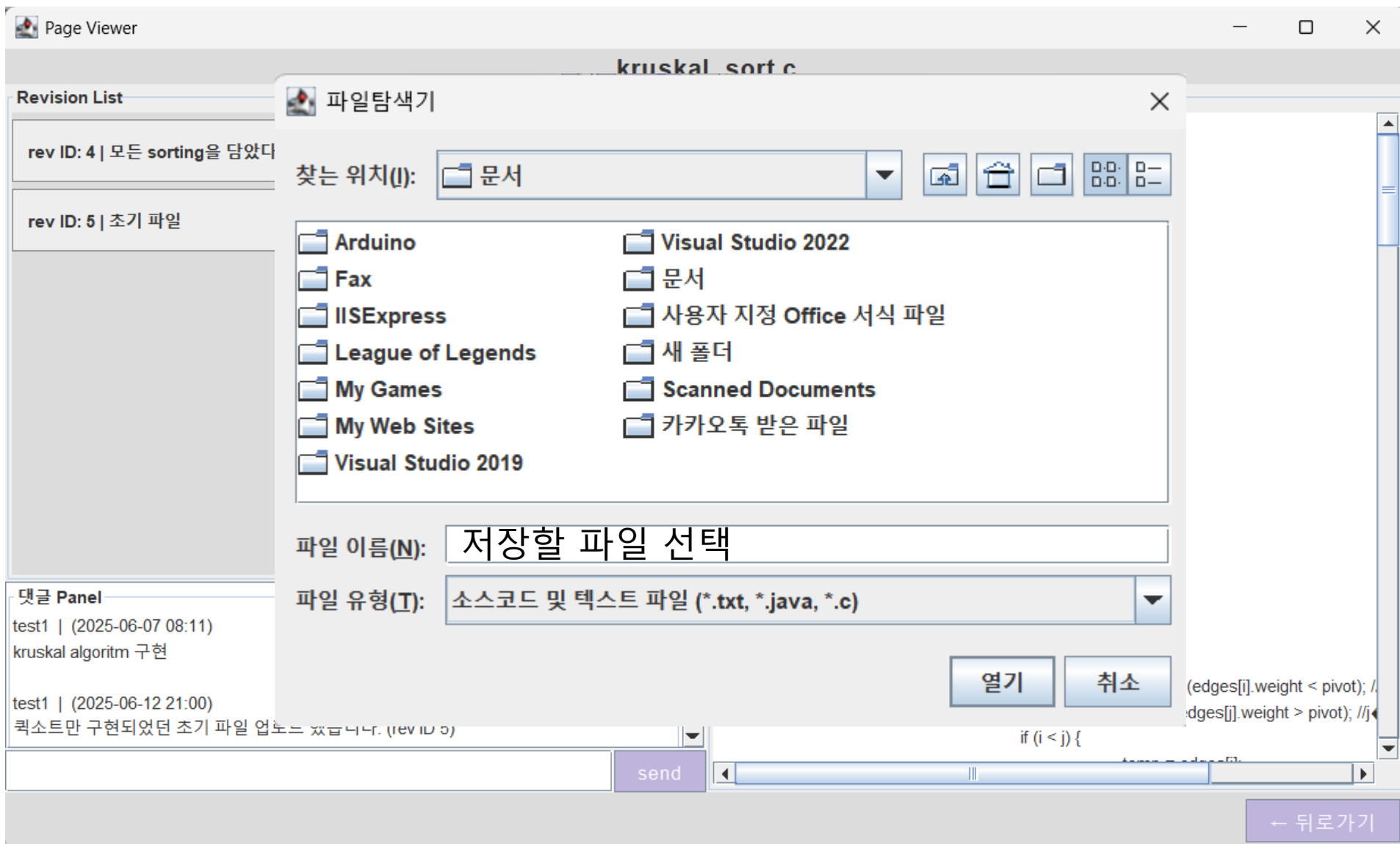


리비전 화면 (파일 업로드)

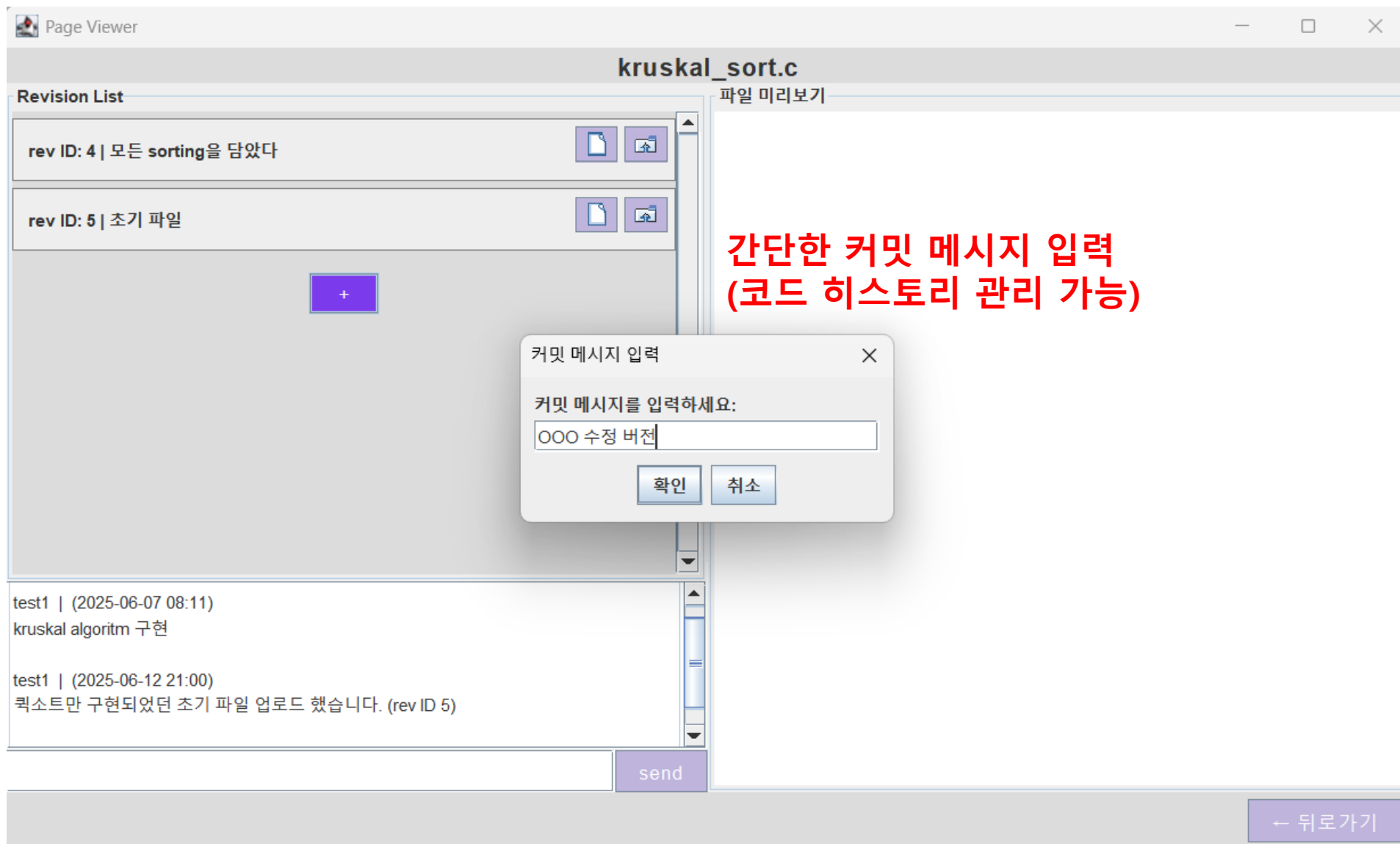
The screenshot shows a web application window titled "Page Viewer" with a sub-header "kruskal_sort.c". The interface is divided into several sections:

- Revision List:** A table-like structure showing two revisions. The first row is "rev ID: 4 | 모든 sorting을 담았다" with document and folder icons. The second row is "rev ID: 5 | 초기 파일" with similar icons. Below this, a large grey area contains a red circle around a small purple square with a white plus sign, indicating a file upload button.
- 댓글 Panel (Comment Panel):** Located at the bottom left, it contains two comments:
 - test1 | (2025-06-07 08:11)
kruskal algorithm 구현
 - test1 | (2025-06-12 21:00)
퀵소트만 구현되었던 초기 파일 업로드 했습니다. (rev ID 5)
- 파일 미리보기 (File Preview):** A large text area on the right showing the content of the selected file. It contains C code for a Kruskal's algorithm implementation, including headers, a struct definition, and a quickSort function.
- Buttons:** A "send" button is located at the bottom center, and a "← 뒤로가기" (Back) button is at the bottom right.

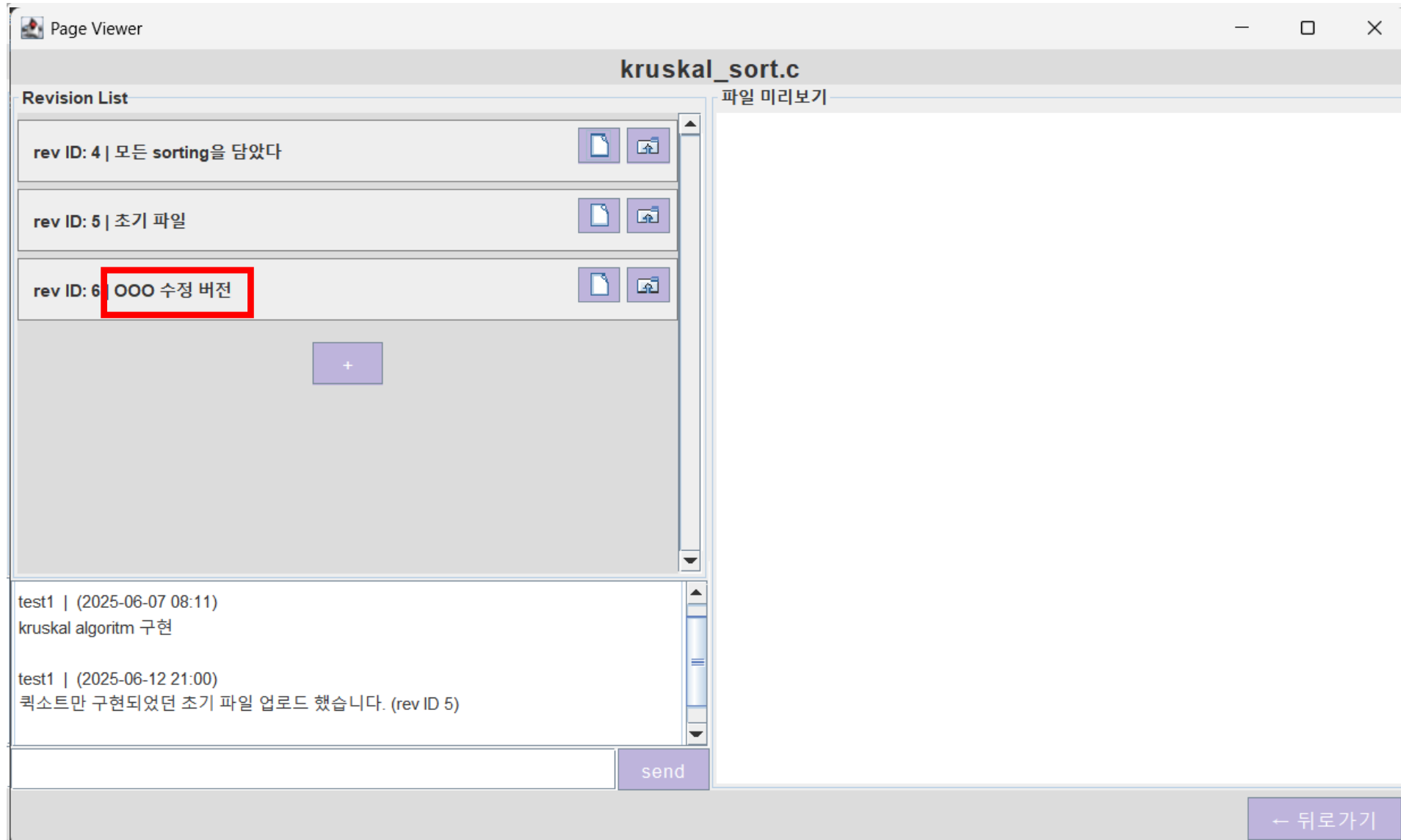
리비전 화면 (파일 업로드)



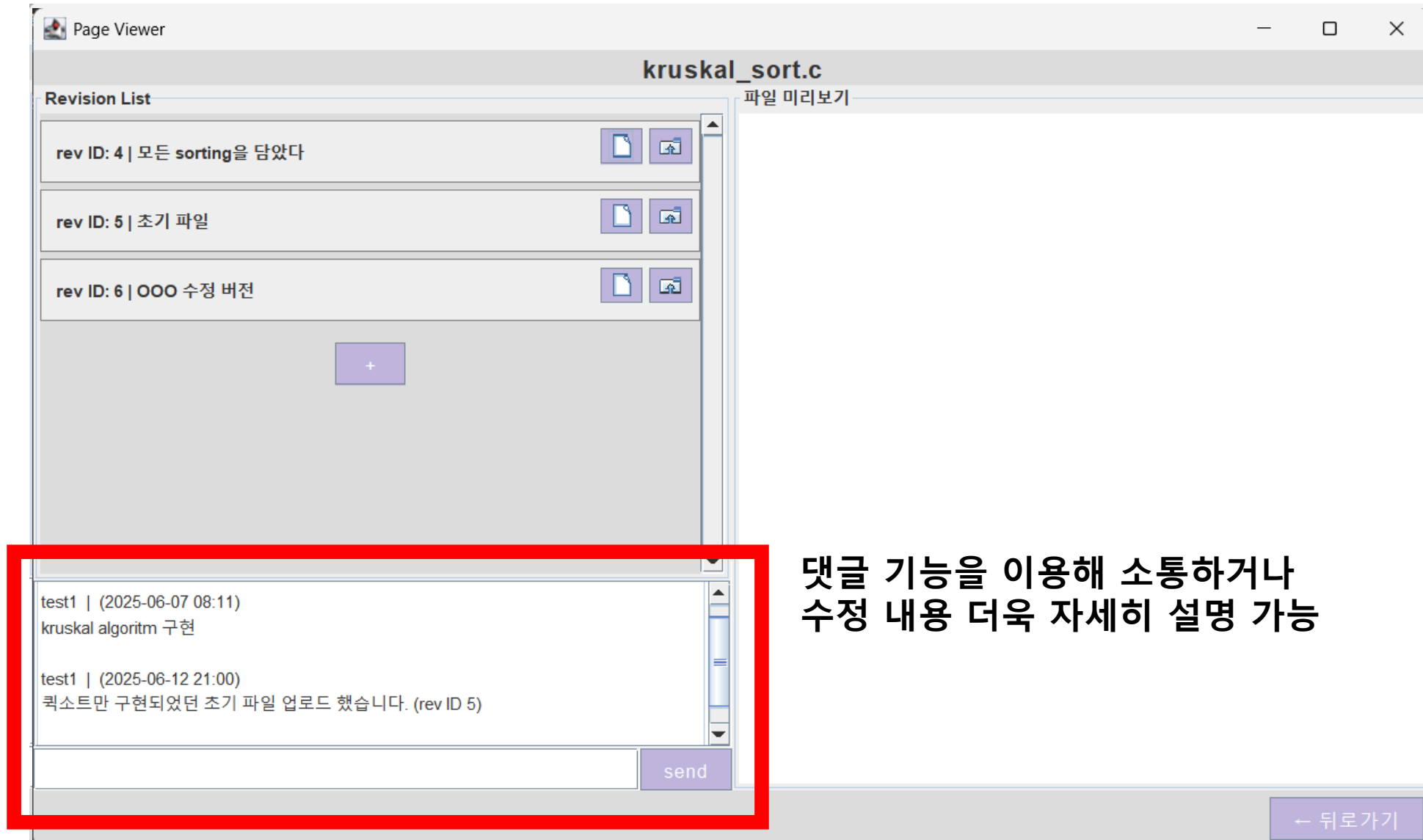
리비전 화면 (파일 업로드)



리비전 화면 (파일 업로드)

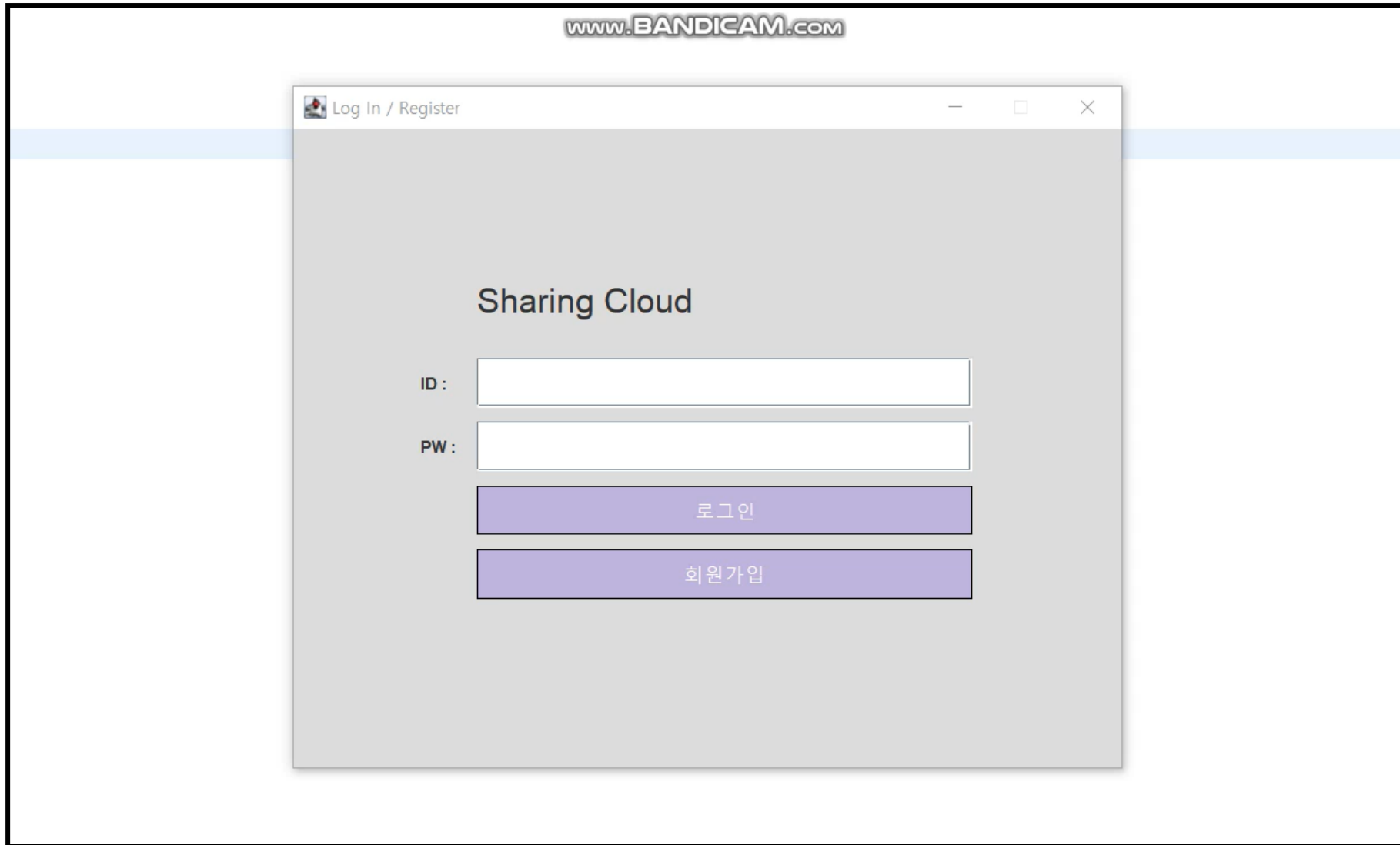


리비전 화면 (댓글 창)

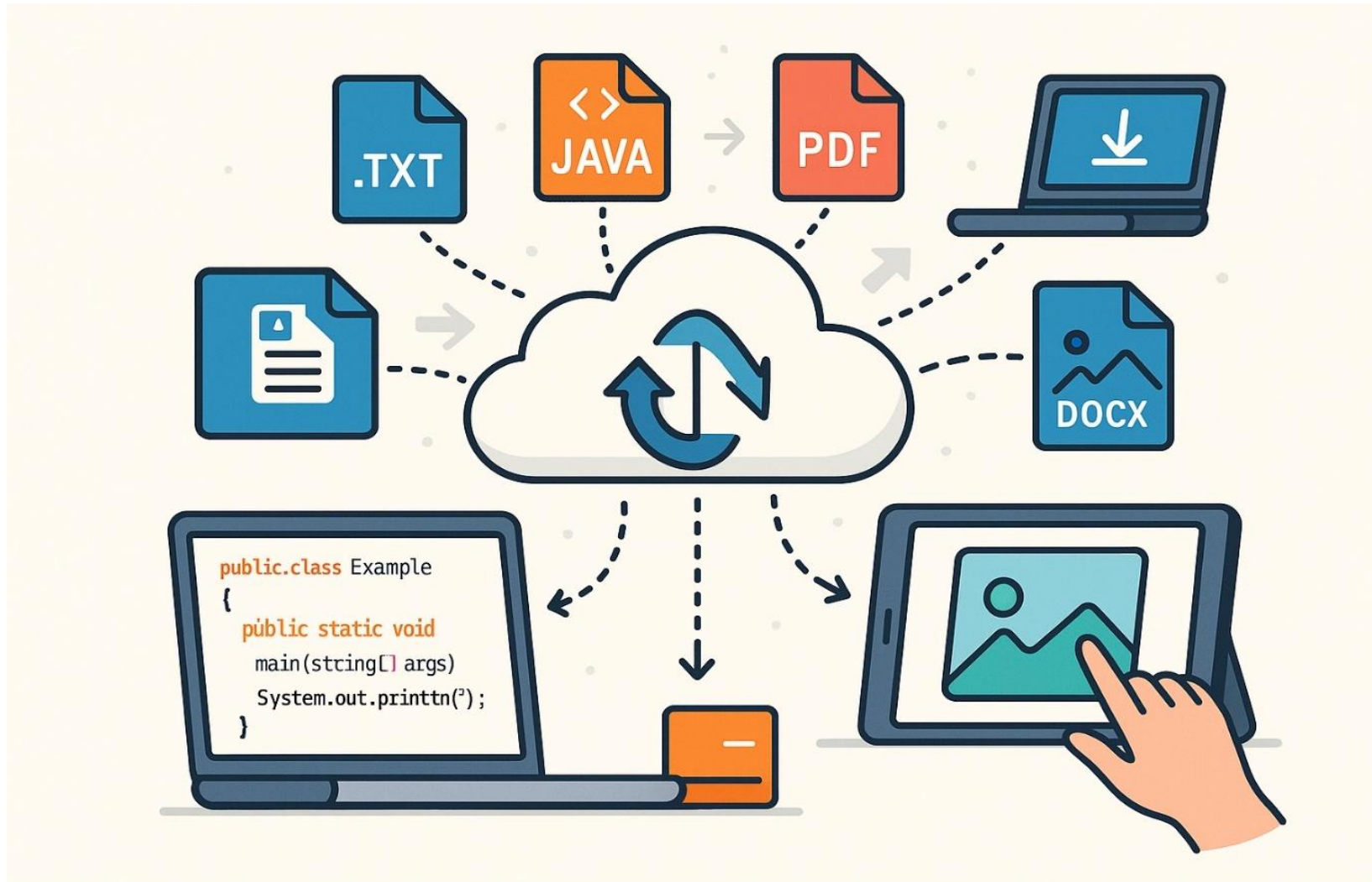


댓글 기능을 이용해 소통하거나
수정 내용 더욱 자세히 설명 가능

프로그램 데모 영상



향후 확장성 및 발전성



감사합니다



자바프로그래밍 002 분반

Team 2: 강승윤, 김우현, 박현욱, 배지훈, 이동훈