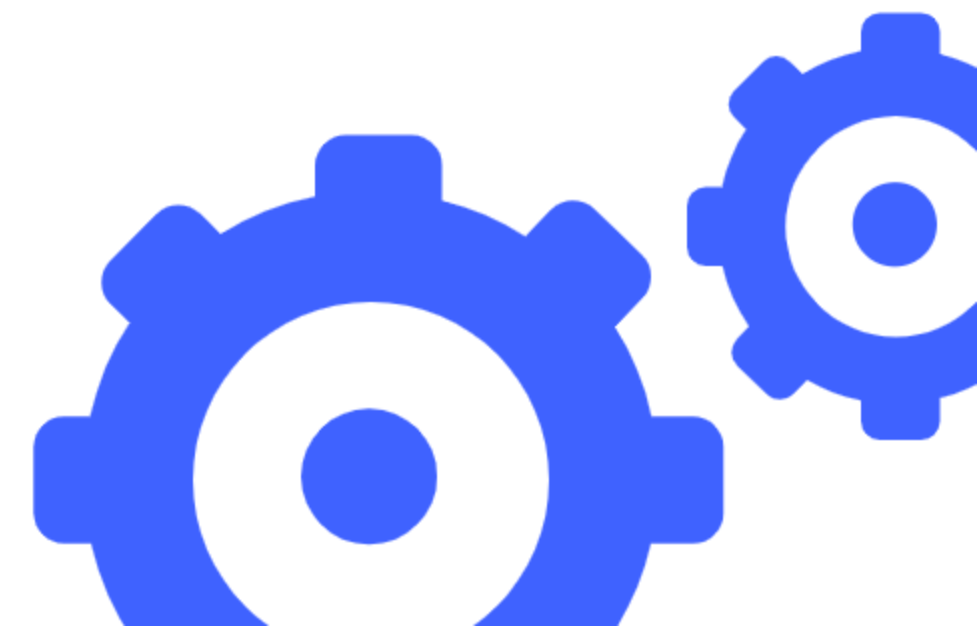




System\_Programming

# Co-Sync

3팀 : 실시간 채팅 기반 파일 협업 시스템



# 프로젝트 목적

---

## 팀 협업 중 실시간 대화와 파일 공유가 함께 이루어져야 하는 상황이 빈번하게 발생

- 파일을 채팅 중 공유하더라도, 버전 관리나 공동 작업을 위해 다른 플랫폼으로 이동해야 하는 불편함이 존재.
- 실시간 대화 중에 파일을 함께 편집할 때 업데이트 내용이 즉각 반영되지 않아, 실시간 피드백이나 조정이 어려움.
- 파일이 여러 번 수정될 경우, 변경 사항을 쉽게 확인할 방법이 부족해 버전 관리가 번거로움.

**실시간 채팅 중 파일 공동 작업 및 버전 관리 기능을 추가하여,  
채팅을 중심으로 파일을 공유, 수정, 관리할 수 있는 환경을 제공**

# 기대효과

---



## 팀 채팅

협업을 위한 채팅 기능



## 파일 동기화

팀원끼리 실시간 파일 동기화



## 파일 버전 관리

버전 관리를 위한 다양한 기능

## 팀원분담

---



조신근

- 프로젝트 관리자(팀장)
- 초기 회원가입 기능 구현
- 클라이언트-서버 연결



김무성

- 설계자
- 파일 동기화 기능 구현



서지희

- 발표자
- 채팅, 파일 협업 내역 저장



도현민

- 설계자
- 실시간 채팅 기능 구현



전호준

- 기록자
- 버전 관리 기능 구현

# 개발환경

---



Discord



Visual Studio Code



Ubuntu 22.04



GitHub




C



Notion

# 협업 도구(Notion)



11월 29일 편집 H S J 공유

## 시프 프로젝트

💡 프로젝트 개요(흐름)

💡 평가 기준

+ :: 1) 프로젝트 기획

🍃 주제 선정

🍃 주제 구체화

🍃 팀 내 역할 분담

2) 설계

🍃 시스템 설계

🍃 UI/UX(터미널 기반) 설계

🍃 필요한 기술 스택 및 개발 역할 분담

3) 개발

🍃 개발 환경 조성

🍃 개발

4) 테스트 및 디버깅

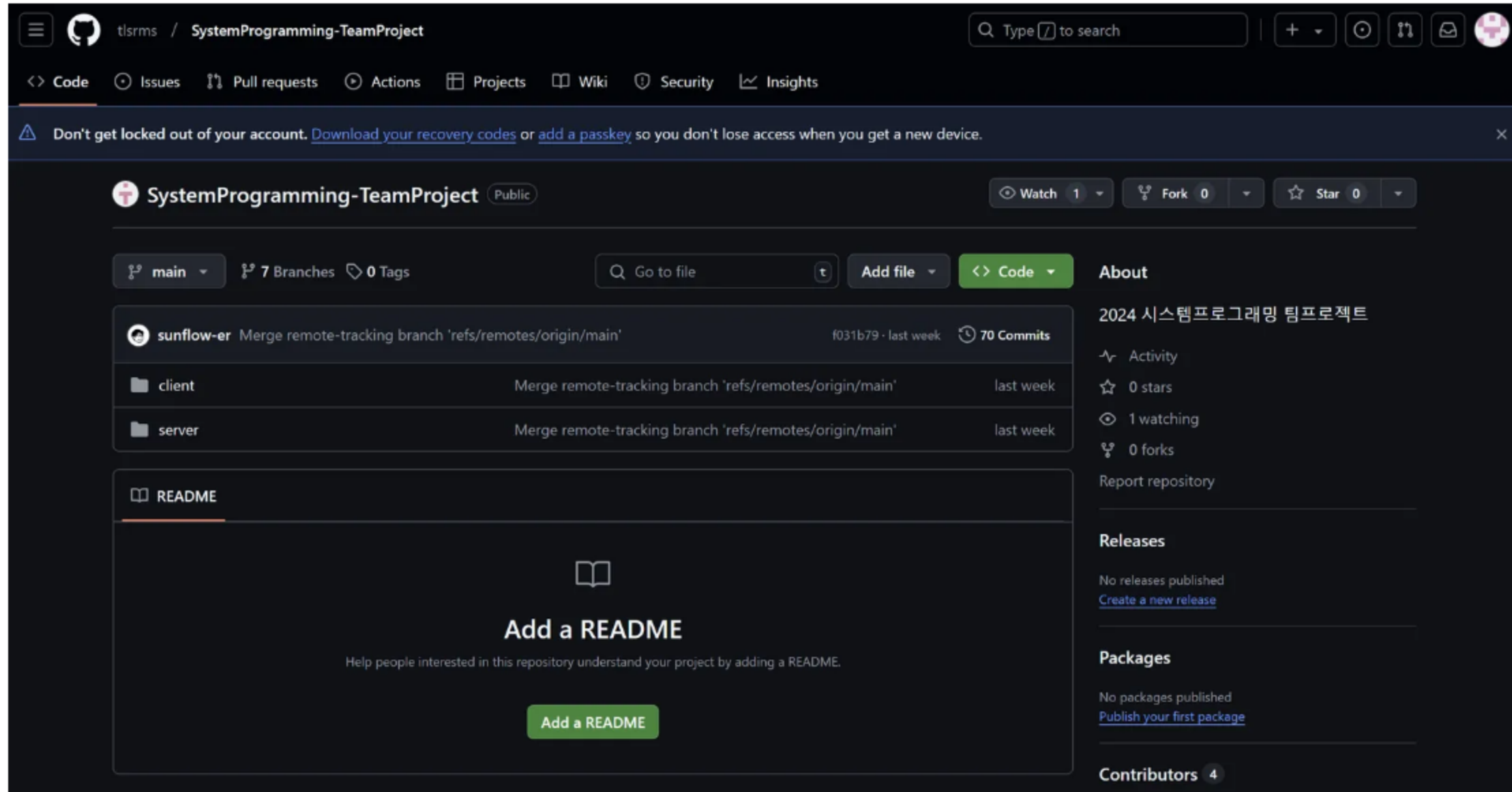
• 유닛 테스트

• 통합 테스트

• 버그 수정

Link : <https://www.notion.so/1377e299876a80a484ddface316f552e>

# 협업 도구(GitHub)



Github

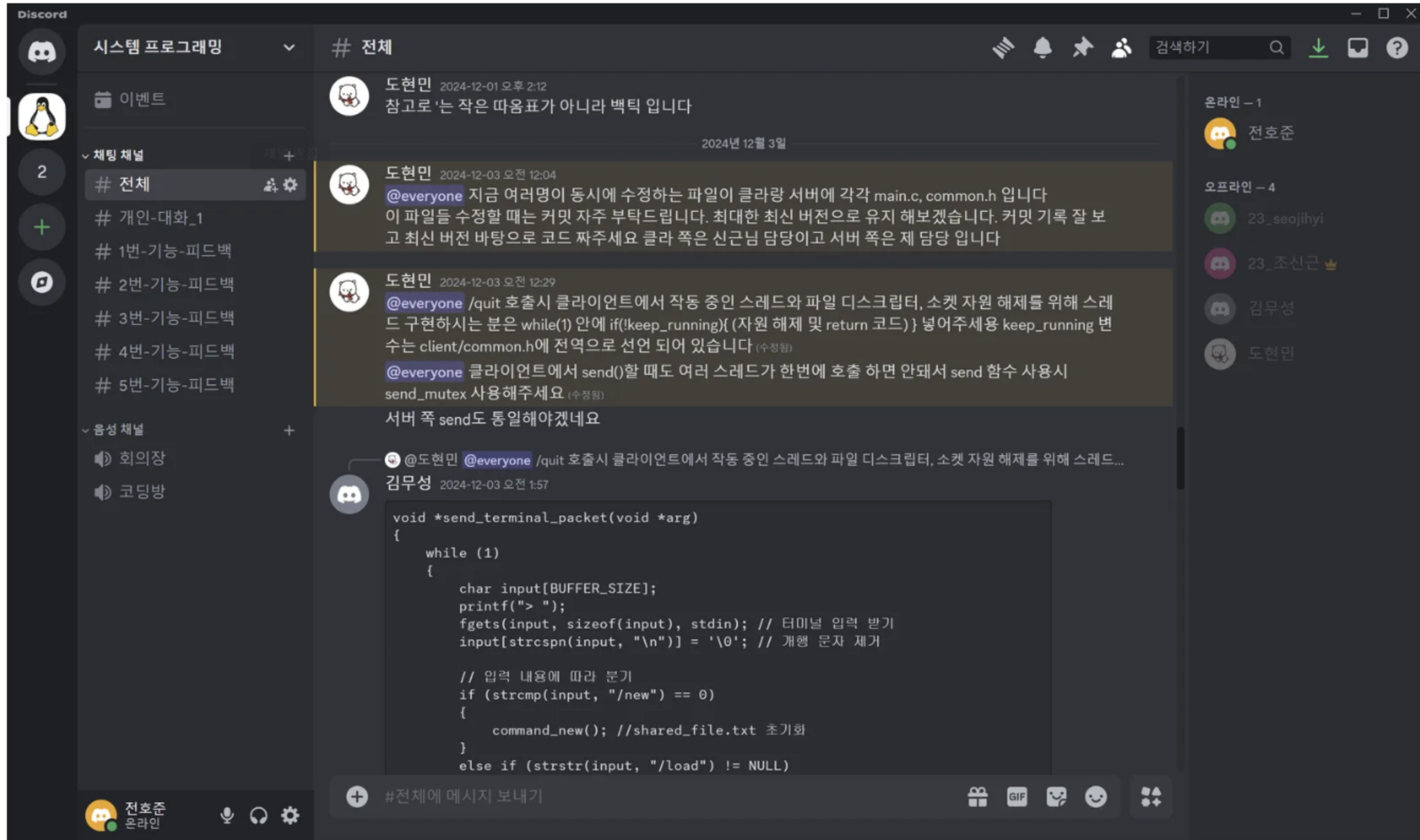
Link : <https://github.com/tlsrms/SystemProgramming-TeamProject>



Git graph



# 협업 도구(Discord)





# 기술스택

---

**Thread: 0**

**File I/O: 0**

**Signal handling: 0**

**Pipe: X**

**Multi-threading: 0**

**Other: inotify, socket**

# 코드 라인

파일명	전체 코드라인 수	순수 코드라인 수
login_control.c	65	54
broadcast.c	19	16
chat_handler.c	25	22
common.c	59	48
file_sync.c	37	31
version_control.c	234	173
main.c	501	410
file_monitor.c	186	143
send_handler.c	253	185
receive_handler.c	35	26
file_sync.h	20	17
login_control.h	39	33
version_control.h	46	39
broadcast.h	22	18
chat_handler.h	30	25
common.h	81	67
receive_handler.h	22	18
send_handler.h	94	77
file_monitor.h	71	60
Total	1,843	1,562

**TOTAL : 1562 lines**

**/\*주석, 헤더 포함 : 1843 lines\*/**

# 프로젝트 흐름

---

01

주제 선정 및  
구체화

02

역할 분담

03

설계

04

개발

05

테스트 및 디버깅  
버그 수정

06

문서 및 소스코드 정리  
발표 자료 제작

# 개발 일정

SUN	MON	TUE	WED	THU	FRI	SAT
10	11	12	13	14	15	16
주제 선정 및 변경					역할 분담	
17	18	19	20	21	22	23
	설계					
24	25	26	27	28	29	30
개발						
1	2	3	4	5	6	7
테스트, 디버깅 및 버그 수정						
8	9	10	11	12		
문서 및 소스코드 정리, 발표 자료 제작						

# 주요기능

---

로그인/회원 가입

flag 0



클라이언트 간 채팅

flag 1



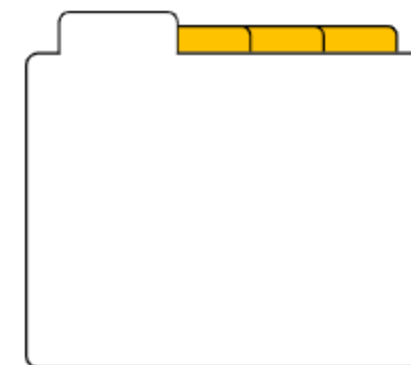
공유 파일 동기화

flag 2

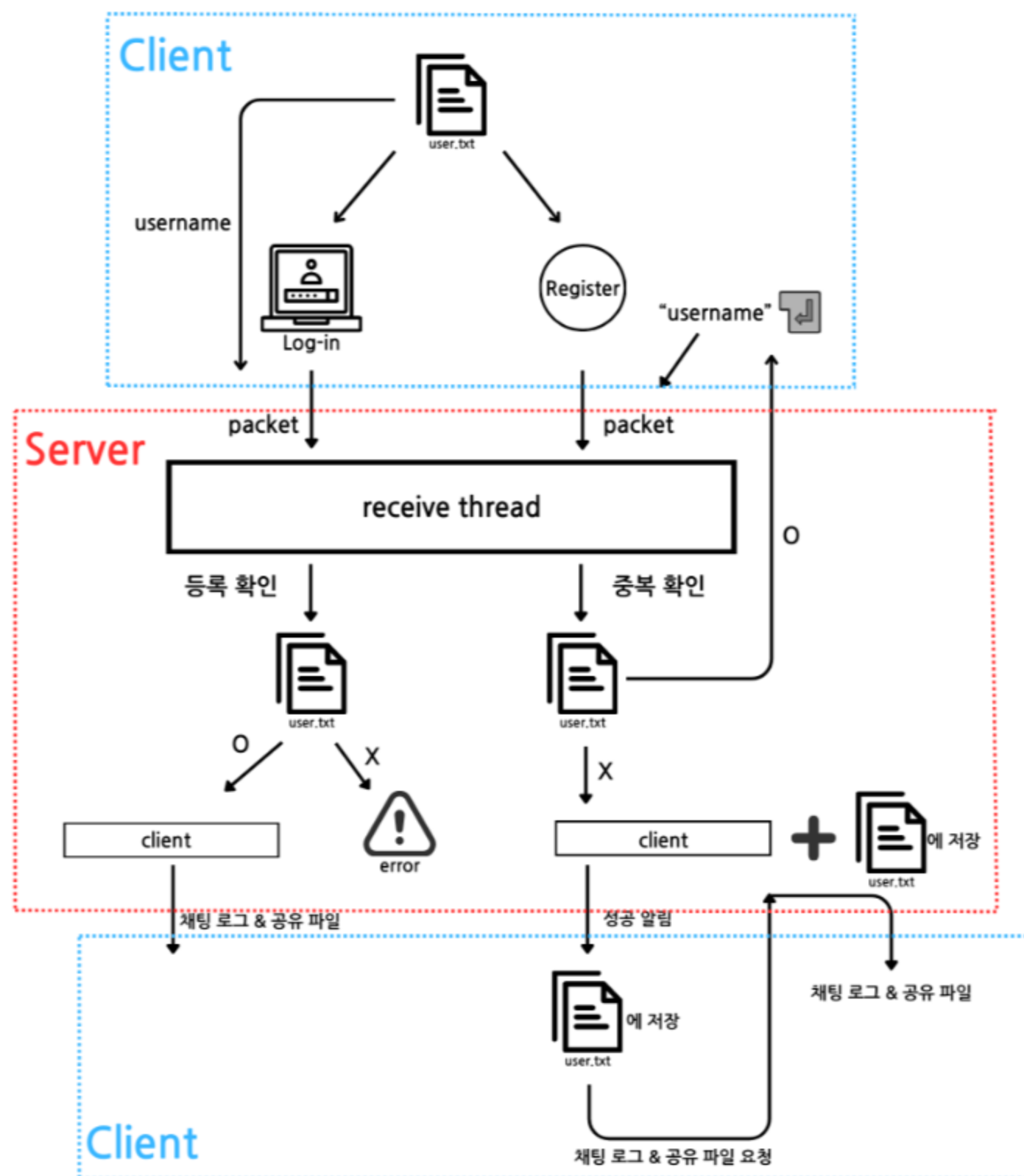


공유 파일 명령어 및 버전 관리

flag 3



# flag 0. 로그인/회원가입



## [회원 가입]

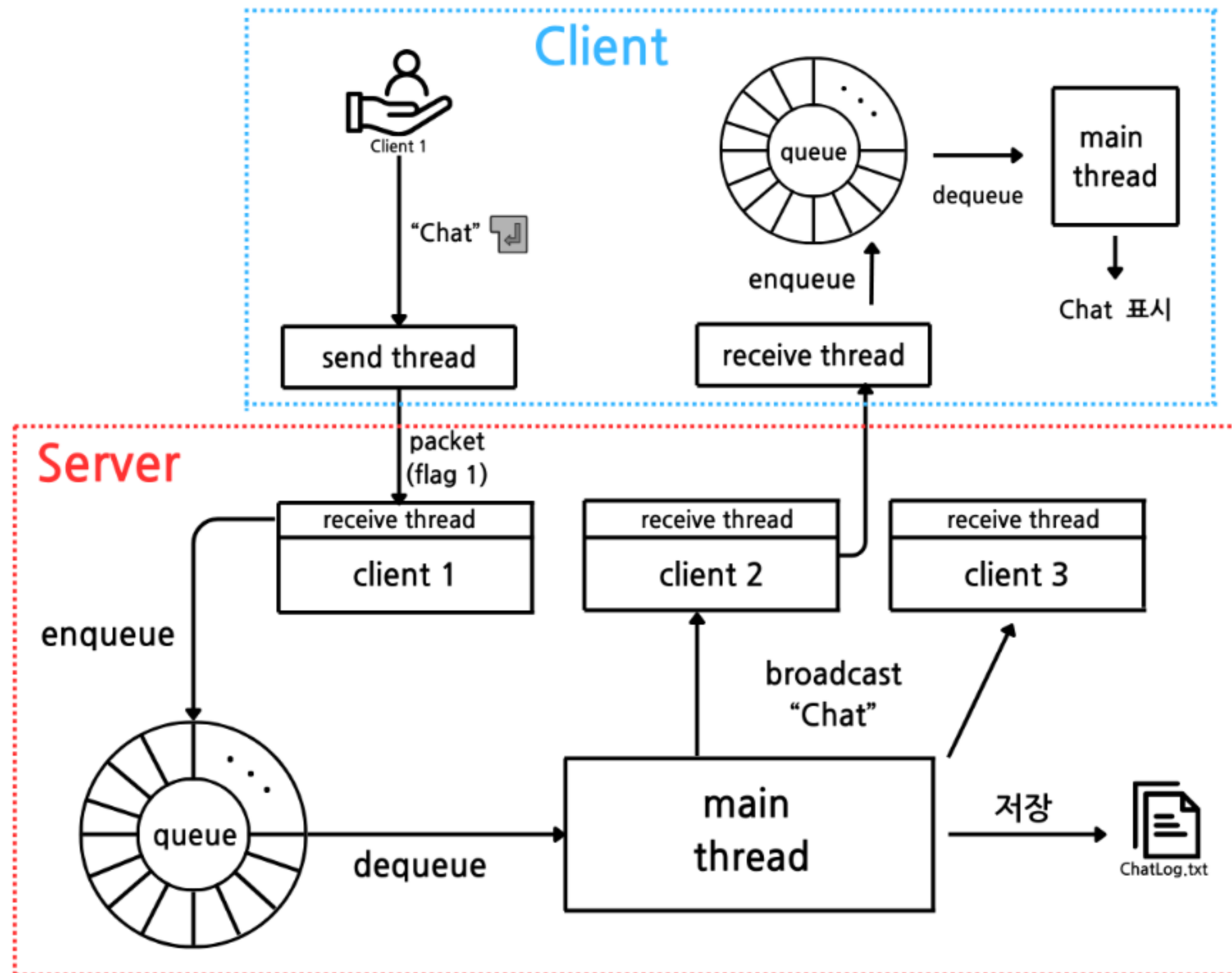
- 중복 확인
- 처리
- 중복 시 에러
- 아닐 시 정보 저장 후 성공 메시지 전송

## [로그인]

- 전송
- user.txt에서 탐색
- 일치 시 로그인 승인, 채팅 로그와 공유 파일 전송
- 불일치 시 회원가입 절차



# flag 1. 클라이언트 간 채팅



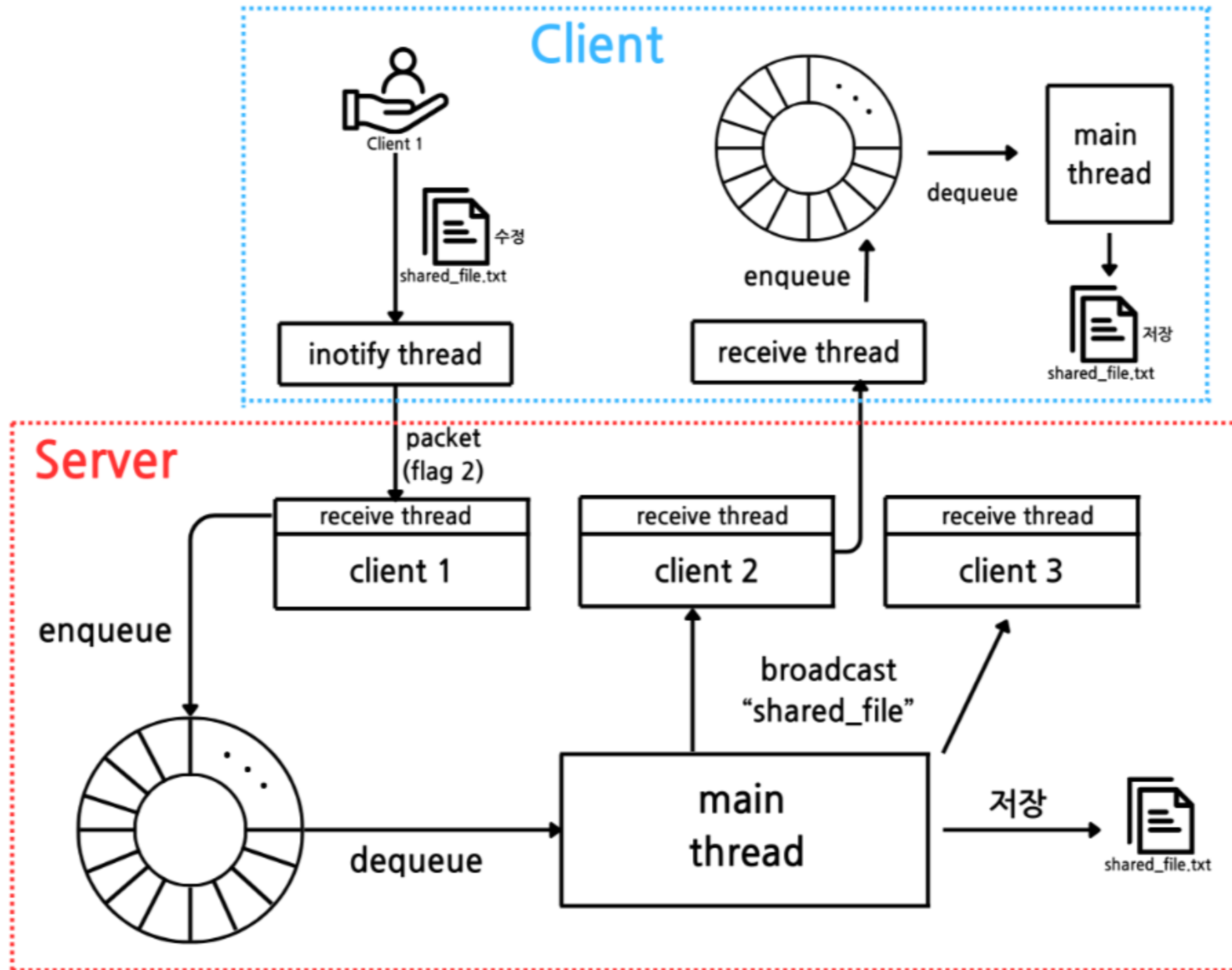
## [클라이언트]

- 전송 : 채팅 메시지 send thread가 서버로 전송
- 수신 : 받은 메시지 receive thread가 처리 후 표시

## [서버]

- 수신 : 클라이언트 메시지 queue에 저장
- 브로드캐스트 : main thread가 모두에게 전송
- 저장 : ChatLog.txt에 저장

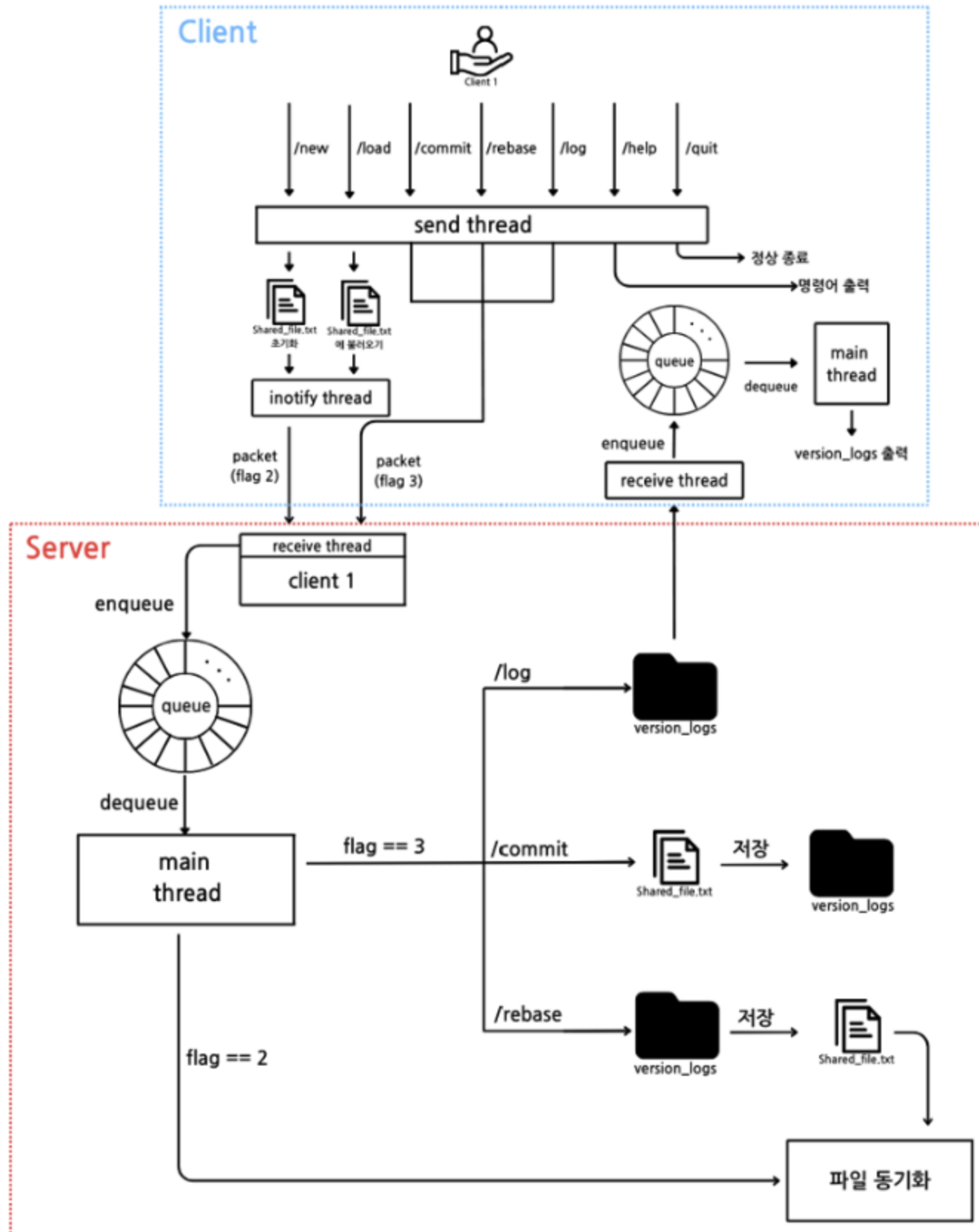
## flag 2. 공유 파일 동기화



[클라이언트]  
-수정 감지  
-수신 및 저장

[서버]  
-수신  
-브로드캐스트  
-저장

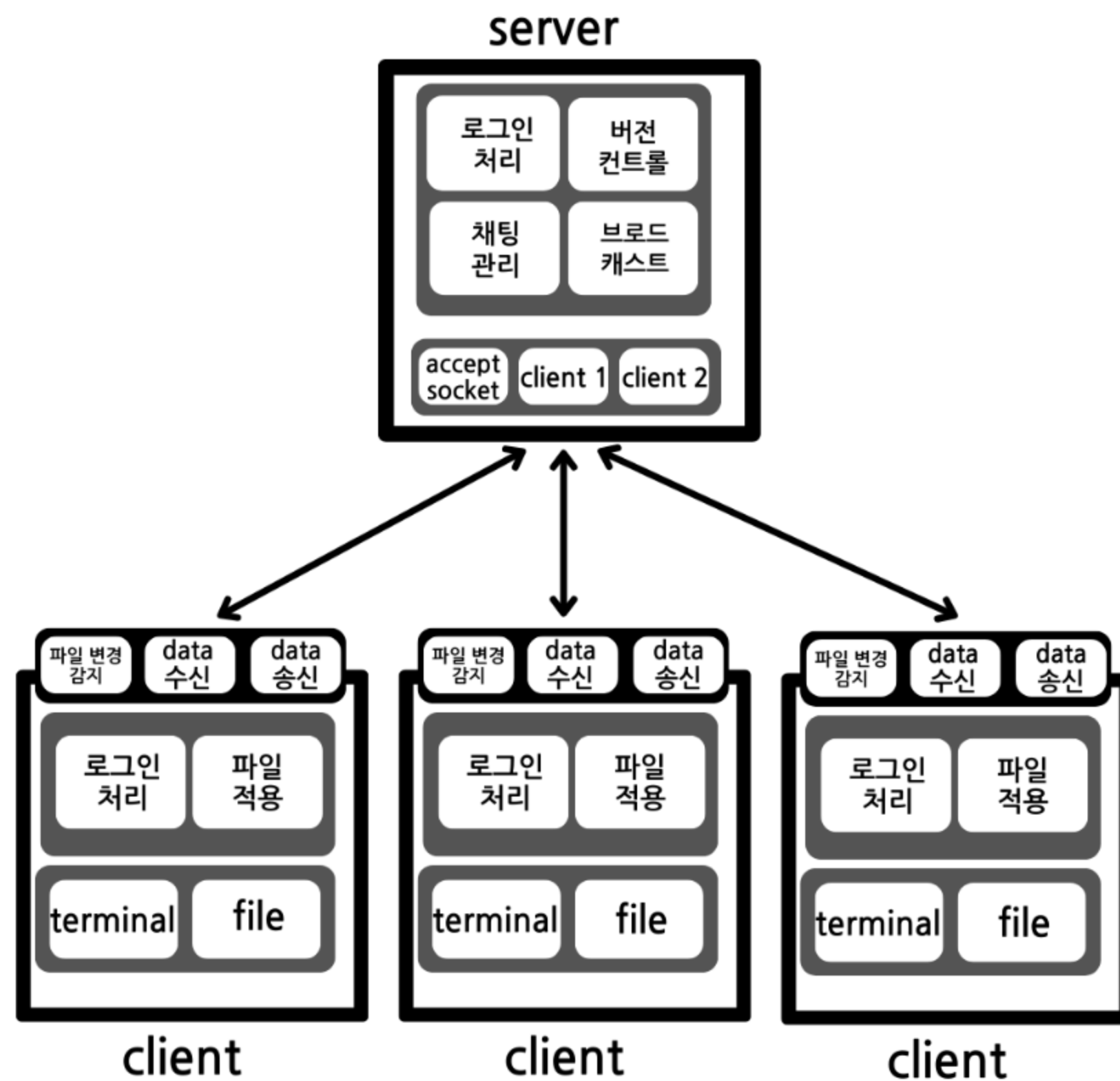
# flag 3. 공유 파일 명령어 및 버전 관리



[클라이언트]  
-명령어 입력  
-파일 감지  
-응답 수신  
-버전 로그 표시

[서버]  
-명령어 처리  
-`/commit`  
-`/rebase`  
-`/log`  
-파일 동기화

# 블록 다이어그램



# 설계구성요소

구성요소 항목	항목에 대한 설명	설계과정에서 아래 항목을 다룬 이유	만족 여부
목표 설정	협업 중 실시간 대화와 파일 공유의 불편함을 해결하기 위해, 파일 동기화와 버전 관리가 포함된 실시간 채팅 기반 협업 도구 개발	팀원들이 작업 중 채팅과 파일 공유를 효율적으로 할 수 있도록 하여, 플랫폼 간 이동 없이 작업을 진행할 수 있도록 하기 위함	○
합성	사용자 인증, 실시간 채팅, 파일 동기화, 버전 관리, 협업 내역 저장 및 재활용 기능을 포함한 전체 시스템의 구조 설계	각 기능이 독립적으로 작동하면서도 서버-클라이언트 아키텍처를 기반으로 서로 연동될 수 있도록 설계하기 위함	○
분석	클라이언트-서버 간 데이터 동기화 지연, 파일 충돌 문제, 서버 동작 시 다중 사용자 연결 안정성 문제 분석 및 해결	실시간으로 데이터를 동기화하면서 사용자 간의 충돌을 방지하고, 안정적인 연결 상태를 유지하기 위해 발생 가능한 문제를 선제적으로 해결하기 위함	○
제작	소켓 프로그래밍으로 실시간 채팅 구현, 멀티스레드 구조로 클라이언트-서버 동시 처리, inotify로 파일 변경 감지, 버전 관리 디렉터리 기반 파일 복원 및 기록 기능 구현	소프트웨어 구현 단계에서 설계된 구조와 알고리즘을 충실히 반영하여, 사용자가 체감하는 주요 기능이 제대로 작동하도록 하기 위함	○
시험	다양한 사용자 환경에서 실시간 채팅, 파일 동기화, 버전 관리 기능의 정상 작동 여부를 테스트하며, 서버 재시작 후에도 협업 데이터가 정상적으로 불러와지는지 검증	사용자가 시스템을 사용할 때 발생할 수 있는 모든 상황을 미리 점검하여, 예상치 못한 오류를 방지하고 높은 안정성을 보장하기 위함	○
평가	최종적으로 모든 기능이 정상적으로 작동하며, 팀원들이 실시간 협업 환경에서 생산성을 높일 수 있는지 평가	개발 과정에서 의도했던 목적을 달성했는지 확인하고, 협업 효율성과 사용자 만족도를 측정하기 위함	○



# 현실적 제한 조건

제한조건 항목	항목에 대한 설명	설계과정에서 아래 항목을 다룬 이유	만족 여부
경제성	프로그램 개발 및 테스트 과정에서 사용하는 소프트웨어와 하드웨어 자원의 효율성을 극대화하기 위해 비용을 최소화	개발 초기 비용 부담을 줄이고, 협업 도구의 비용 대비 효율성을 높이기 위해 상용 하드웨어 및 오픈소스 소프트웨어 사용	○
안전성	사용자 데이터와 협업 파일의 보안을 강화하고, 시스템의 안정성을 보장	서버와 클라이언트 간 데이터 전송 시 암호화, 인증된 사용자만 접속 가능하도록 설계	○
신뢰성	프로그램 설치와 실행 중 발생할 수 있는 오류를 최소화하기 위해 시스템 호환성과 안정성을 확인	다양한 운영체제 환경에서 프로그램이 안정적으로 작동하도록 테스트하며, 설치 및 실행 오류를 사전에 점검	○
외관성	사용자 인터페이스(UI)와 명령어가 직관적이고 사용하기 쉽게 설계됨	CLI 기반 사용자 환경에서 직관적으로 사용할 수 있는 명령 체계를 설계하여 학습 곡선을 낮추고 생산성을 향상	○
윤리성	저작권 문제를 방지하고, 모든 소스코드 및 리소스는 오픈소스를 사용하거나 직접 개발	프로젝트의 윤리적이고 법적인 문제를 예방하기 위해, 불법 SW나 리소스 사용을 배제하고 오픈소스와 직접 개발을 통해 구현	○
사회적 영향	협업 도구를 통해 팀 생산성을 높이고, 실시간 채팅 및 파일 공유를 통해 팀 간 의사소통과 협업을 효율화	프로그램이 다양한 분야(교육, 비즈니스 등)에 적용될 가능성을 평가하며, 효율성을 높여 생산성 향상에 기여할 수 있는 방법을 도출	○



# 시연

---



<https://youtu.be/BqtsQhYKDrS>



# Q&A

자유롭게 질문해주세요!

