

CodeJam

실시간 협업 코딩 플랫폼

Team JAMstack | 4주차 데모 (2025.01.16)

J044 김선향

J094 노주호

J102 민인애

J141 손혜준

J143 송상화

목차

1. 프로젝트 소개
2. 4주차 주요 성과
3. 핵심 기술 구현
4. 아키텍처 개선
5. 보안 강화
6. 한글 입력(IME) 이슈
7. 데모
8. 향후 계획
9. 4주간 배운 점
10. Q&A



1. 프로젝트 소개

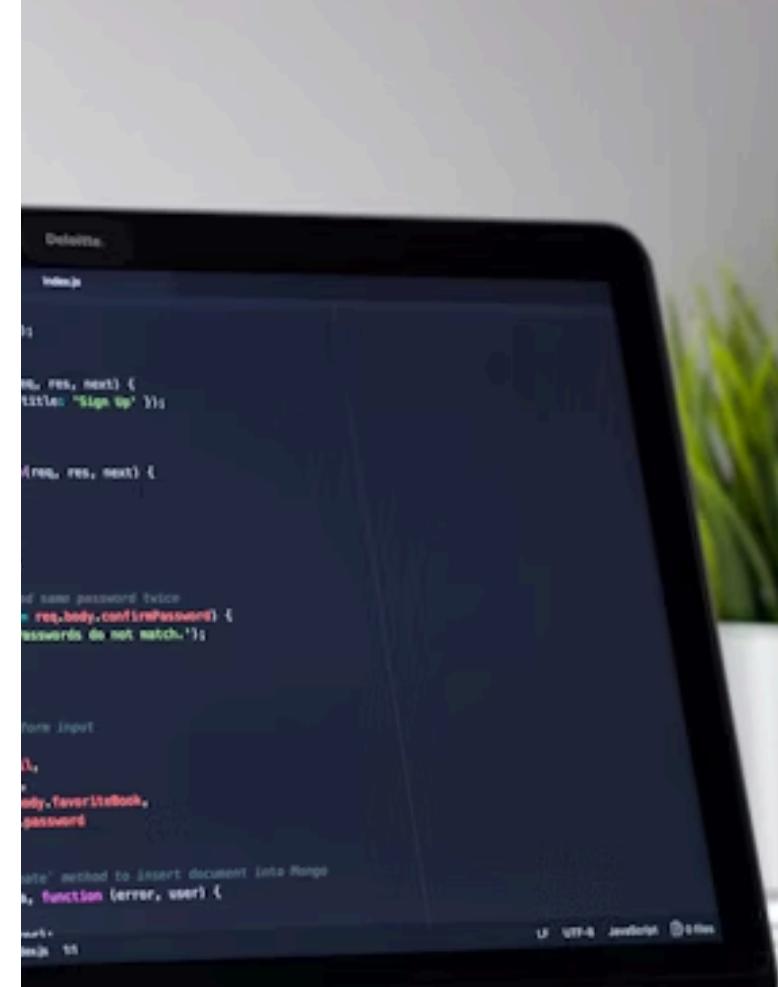
CodeJam - 실시간 협업 코딩 플랫폼

핵심 가치

- **Speed:** 로그인 없이 클릭 한 번으로 즉시 시작
- **Lightweight:** 필수 기능만 담은 빠른 경험
- **Real-time:** CRDT 기반 실시간 동시 편집

타겟 사용자

- 코딩 면접관/지원자
- 페어 프로그래밍 팀
- 알고리즘 스터디
- 대학 강의 (100~150명 동시 접속 지원)



1. 프로젝트 소개 (계속)

주요 기능

Zero-Config & Login-Free

- 버튼 한 번으로 고유 방 URL 생성
- 로그인 없이 익명으로 협업

실시간 협업 도구

- Yjs(CRDT) 기반 동시 편집, 커서 추적 및 팔로우 모드
- 권한 관리 (Host/Editor/Viewer)

고성능 에디터

- CodeMirror 6 기반 경량 에디터
- 24시간 자동 만료로 보안 유지



2. 4주차 주요 성과

이번 주 핵심 과제

데이터 영속성 확보

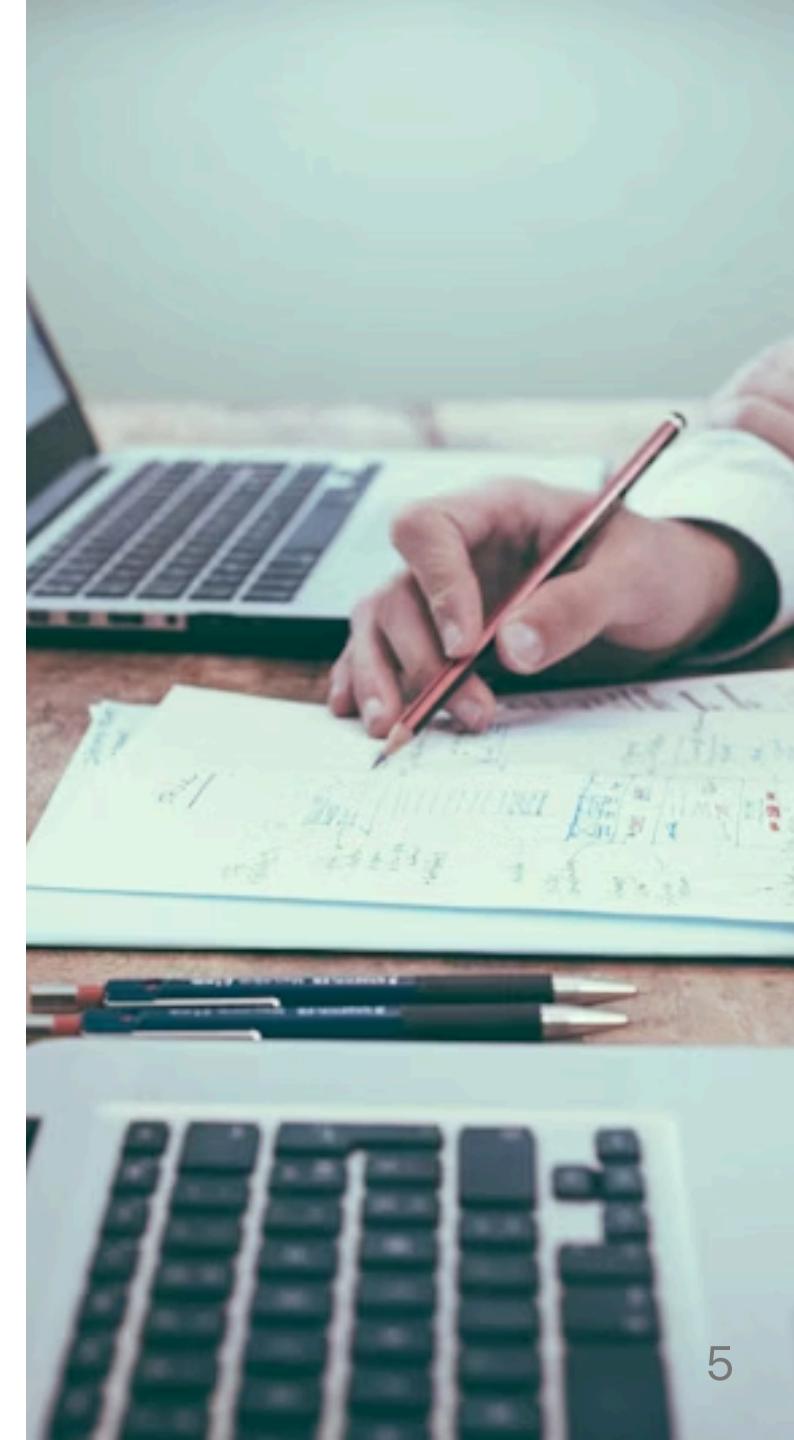
- Y-Redis 기반 문서 복구 시스템 구축
- DB와 Redis 이중화로 안정성 확보

파일 시스템 완성

- UUID v7 기반 파일 식별 체계
- 다중 파일 업로드 및 중복 처리

보안 및 성능 최적화

- JWT 기반 인증 시스템
- 1MB 용량 제한 및 이중 방어



2. 4주차 주요 성과 (계속)

인프라 개선

CI/CD 파이프라인 구축

- PR 생성 시 자동 빌드/테스트/린트 검증
- Frontend: Vercel 배포 (CDN)
- Backend: NCP Container Registry + Docker

웹 서버 전환

- Nginx → Caddy 전환
- SSL 인증서 자동 관리
- 간결한 설정으로 유지보수 효율 향상



3. 핵심 기술 구현

1. Redis + DB 기반 문서 복구 시스템

문제 상황

- 서버 재시작 시 메모리 초기화로 데이터 손실

해결 방안

- Redis + DB를 활용한 실시간 영속성 확보

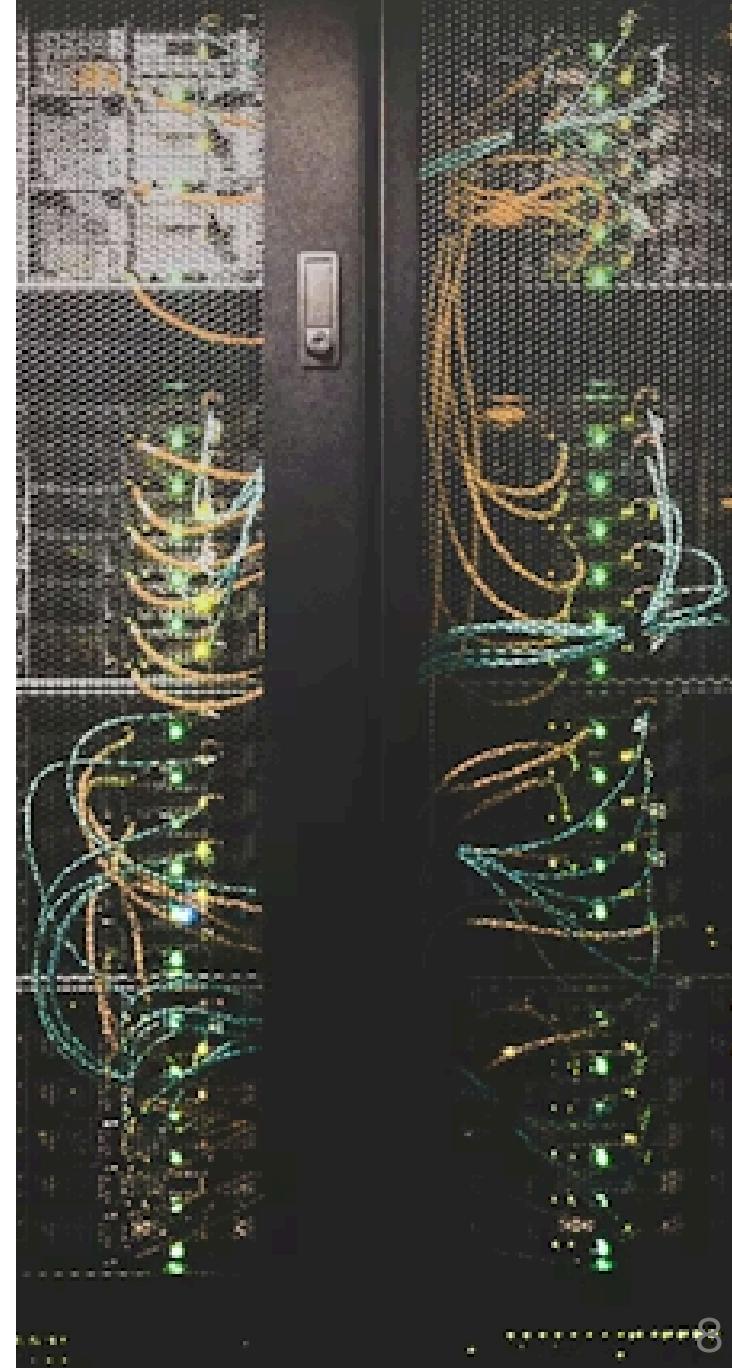


3. 핵심 기술 구현 (계속)

1. Redis + DB 복구 시스템 상세

복구 로직

- 서버에서 Y.Doc 이 메모리에 없으면 Redis 조회 및 업데이트 동기화 연결
- Redis 데이터가 있으면 편집 이력을 바탕으로 복구
- Redis 데이터가 없으면 DB에서 스냅샷을 복구
- Redis 에는 편집 이력 업데이트 조각이 쌓이는 구조
- Redis 영속성 설정(AOF)을 사용해서 Redis 장애 대응

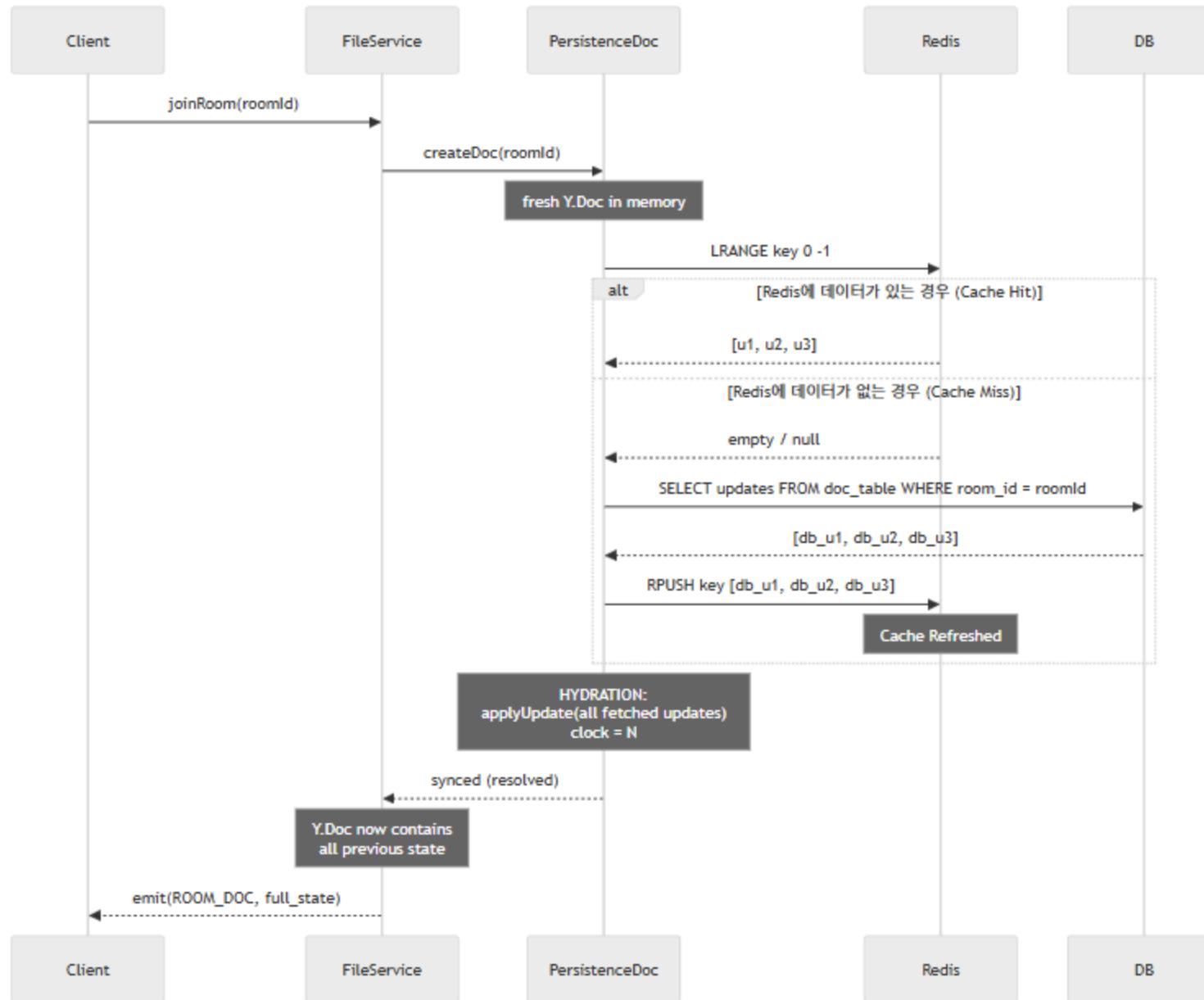


Redis vs DB 역할 분리

- Redis: 실시간 타이핑 저장 및 빠른 로드
- DB: 장기 백업 및 메모리 최적화

메모리 최적화 전략

- 특정 Redis key 의 사용 메모리가 많아지면 그 시점의 편집 이력을 압축해서 Redis와 DB에 저장하여 최적화 (Garbage Collection 과 유사)
- 참가자가 모두 Offline 상태일 때 DB에 스냅샷 저장 후 Redis 데이터를 제거



3. 핵심 기술 구현 (계속)

2. 인메모리 멀티파일 구조

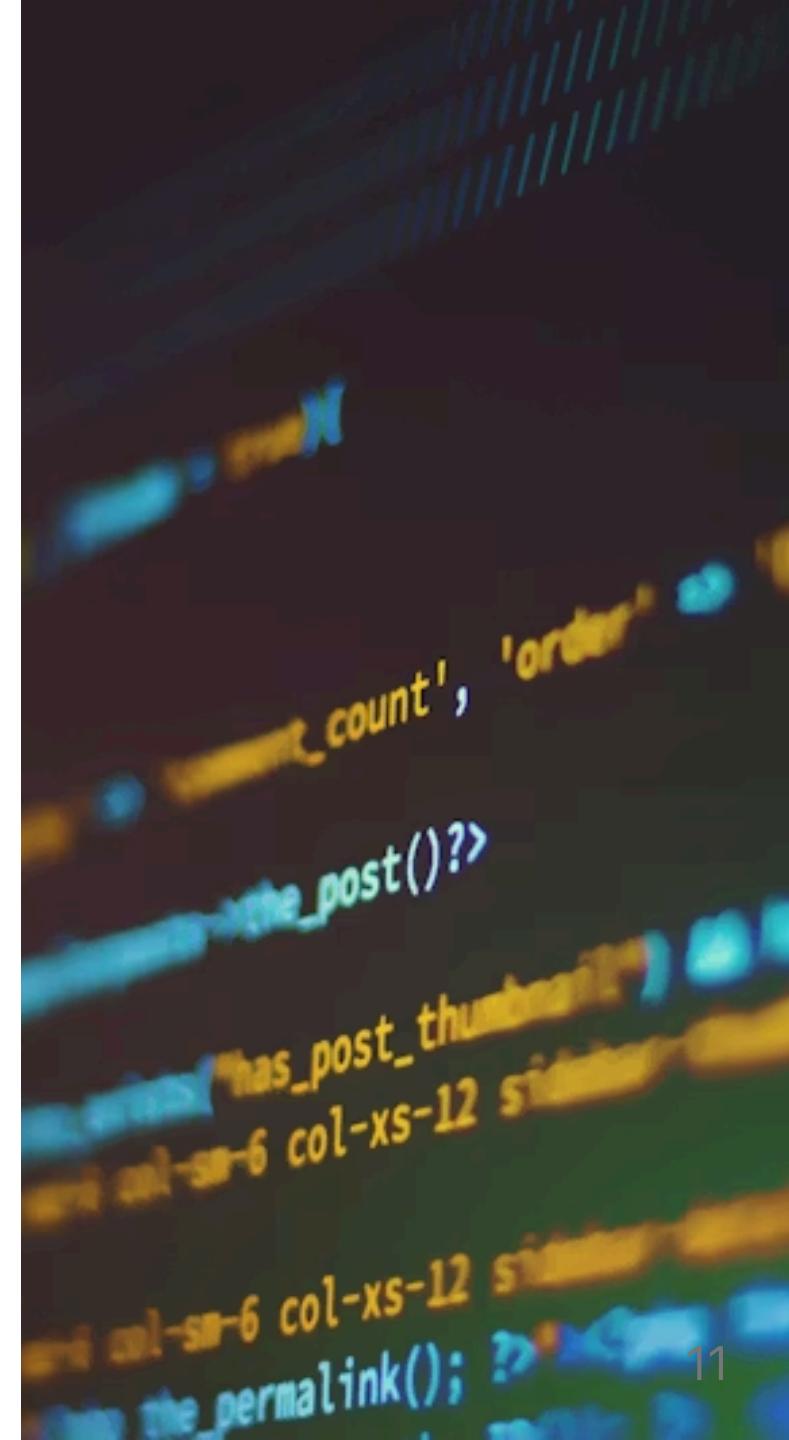
기존 구조

```
Y.Doc (프로젝트 전체 데이터)
└ [roomCode] (Y.Text) : 현재는 방 코드가 곧 파일 ID 역할을 하며 텍스트만 저장됨
```

새로운 구조

```
Y.Doc (프로젝트 전체 데이터)
└ "files" (Y.Map<string, Y.Map>) : 파일 데이터 저장소
    └ [UUID_v7_1] (Y.Map) : 개별 파일 상자
        └ "name" : "main.js" (String, 파일명)
        └ "content" : (Y.Text, 실시간 공동 편집되는 코드 내용)
    └ [UUID_v7_2] (Y.Map)
        └ "name" : "style.css" (String)
        └ "content" : (Y.Text)

    └ "meta" (Y.Map<string, any>) : 프로젝트 단위 메타데이터
        └ "latestSnapshotVersion" : 1 (Number, 최신 스냅샷 버전) //2주차 스프린트에서는 미사용
```



3. 핵심 기술 구현 (계속)

3. JWT 기반 인증 시스템

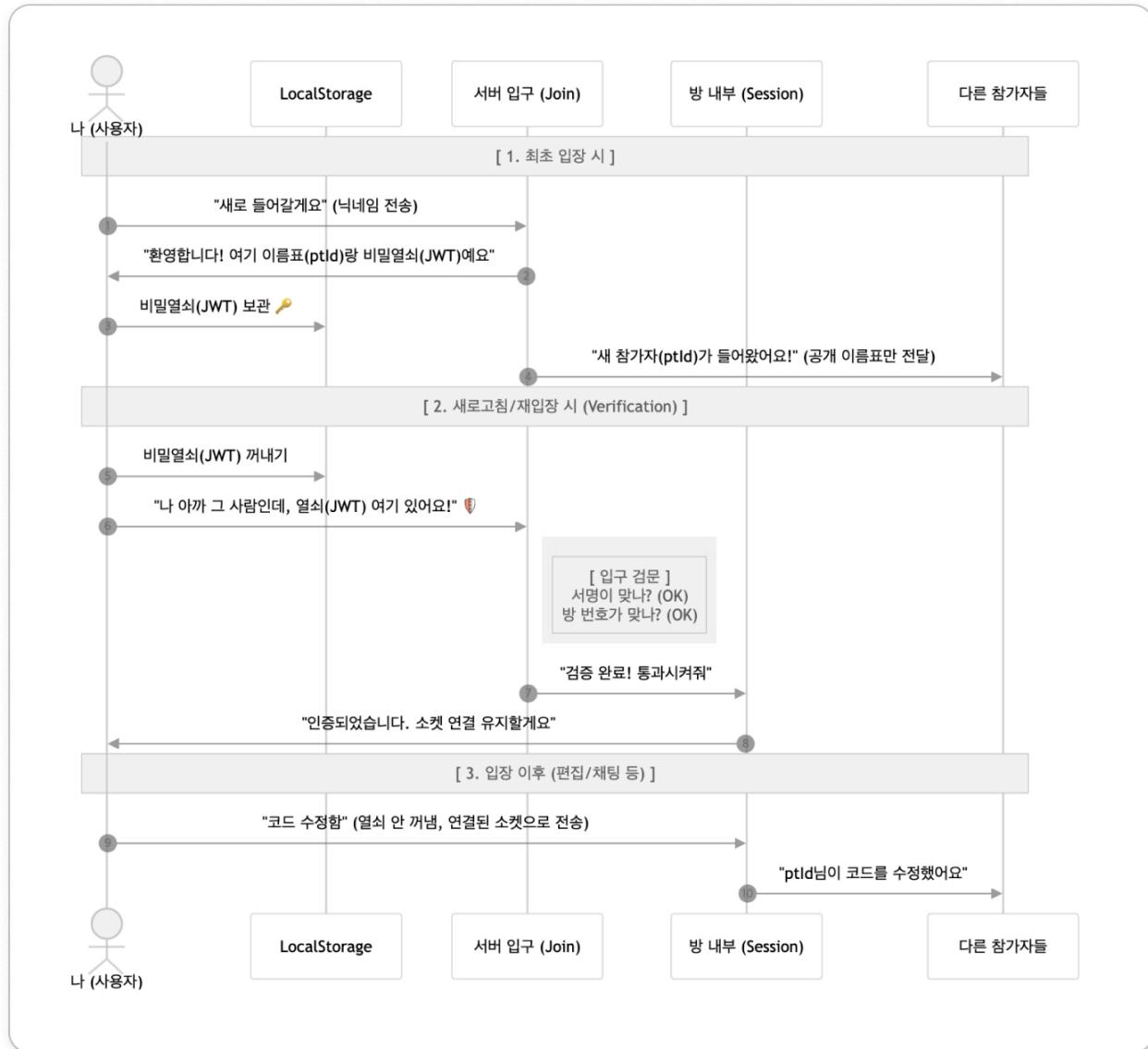
보안 취약점 해결

- 기존: PT_ID를 클라이언트에 직접 노출
- 문제: 모든 사용자의 PT_ID를 알 수 있어 타인 사칭 가능

JWT 도입

- 로컬 스토리지에 PT_ID 대신 토큰 저장
- 서버는 joinRoom 시 토큰 검증
- 소켓 연결 시 토큰 검증 후 소켓 데이터에 PT 정보 귀속
- 허용되지 않은 사용자는 편집 행위 차단



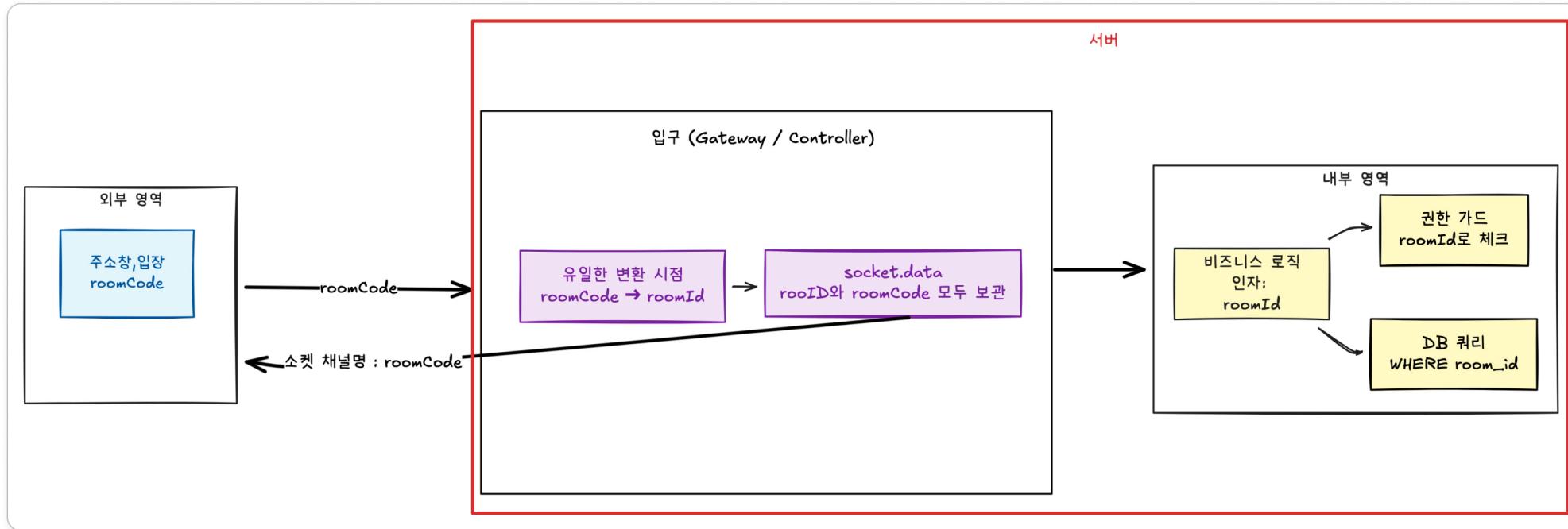


로컬 스토리지에 토큰을 저장하는 이유

- 로컬 스토리지는 보안에 좋지 않음
- 그러나 로컬 스토리지에 PT_ID 정보를 둔 이유는 새로고침 시 사용자 복구 UX를 위한 것
- 보안보다는 복구에 초점이 있는 기획이며, JWT는 최소한의 방어선

3. 핵심 기술 구현 (계속)

4. room_id vs room_code 분리



- room_id : DB의 정수형 PK, 서버 내부로 적용
- room_code : Nano ID 기반, 클라이언트 통신 및 URL 노출용

3. 핵심 기술 구현 (계속)

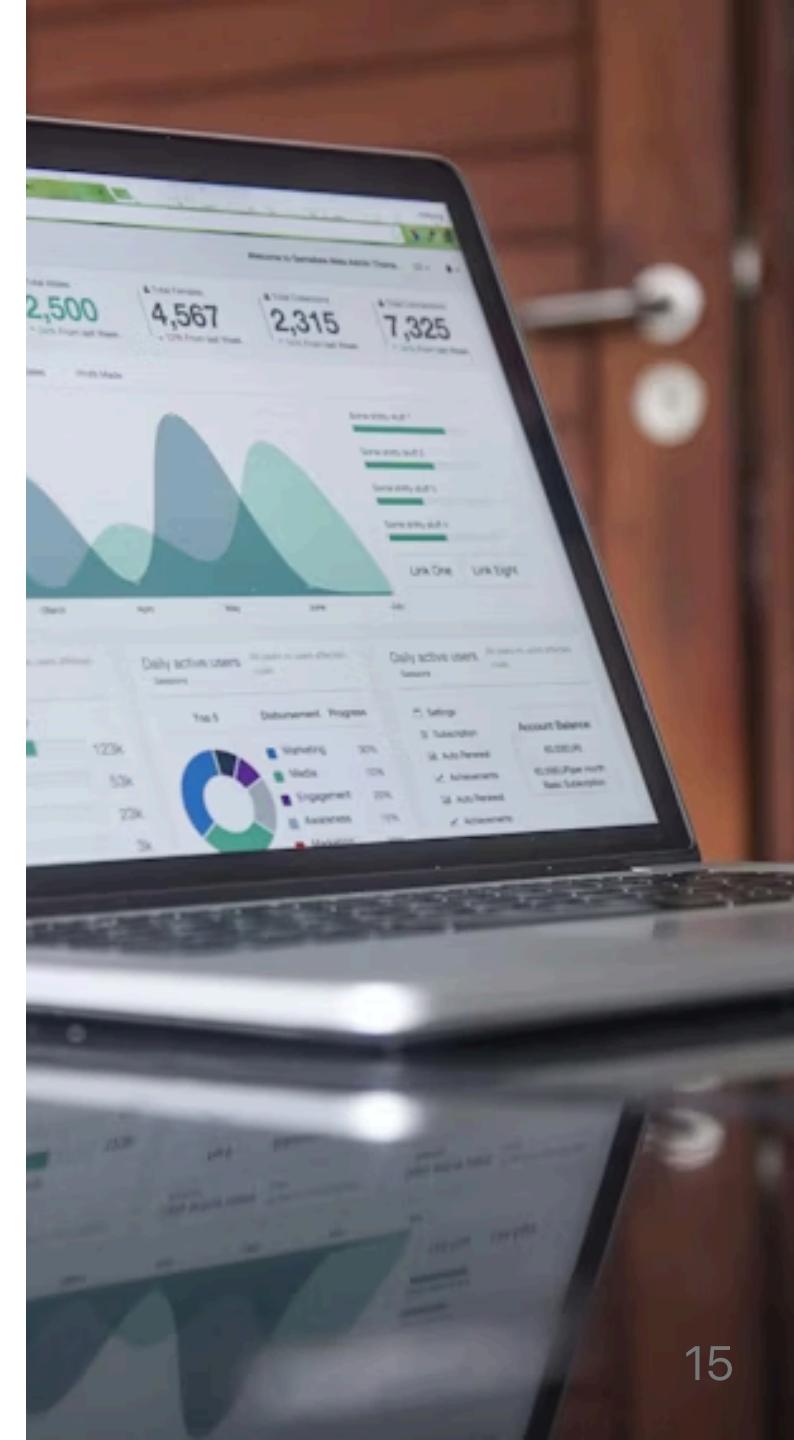
4. 다중 파일 시스템 UI

구현 완료

- 사이드바 구조: 참가자 목록 + 파일 목록 위아래 분할
- 파일 업로드: 버튼 + 드래그 앤 드롭 지원
- 중복 이름 처리: 덮어쓰기/이름 변경 선택 가능
- 파일 정렬: 특수문자 → 알파벳 → 한글 순서

UI 개선 사항

- 상단 파일 탭은 로컬 상태로 관리 (동기화 안 함)
- 사용자 목록에서 '나'와 '호스트' 상단 고정



4. 아키텍처 개선

현재 문제점

- 소켓 코드가 여러 파일에 분산되어 유지보수 어려움 😭
- 비즈니스 로직과 소켓 통신이 혼재 -> 중앙화 필요

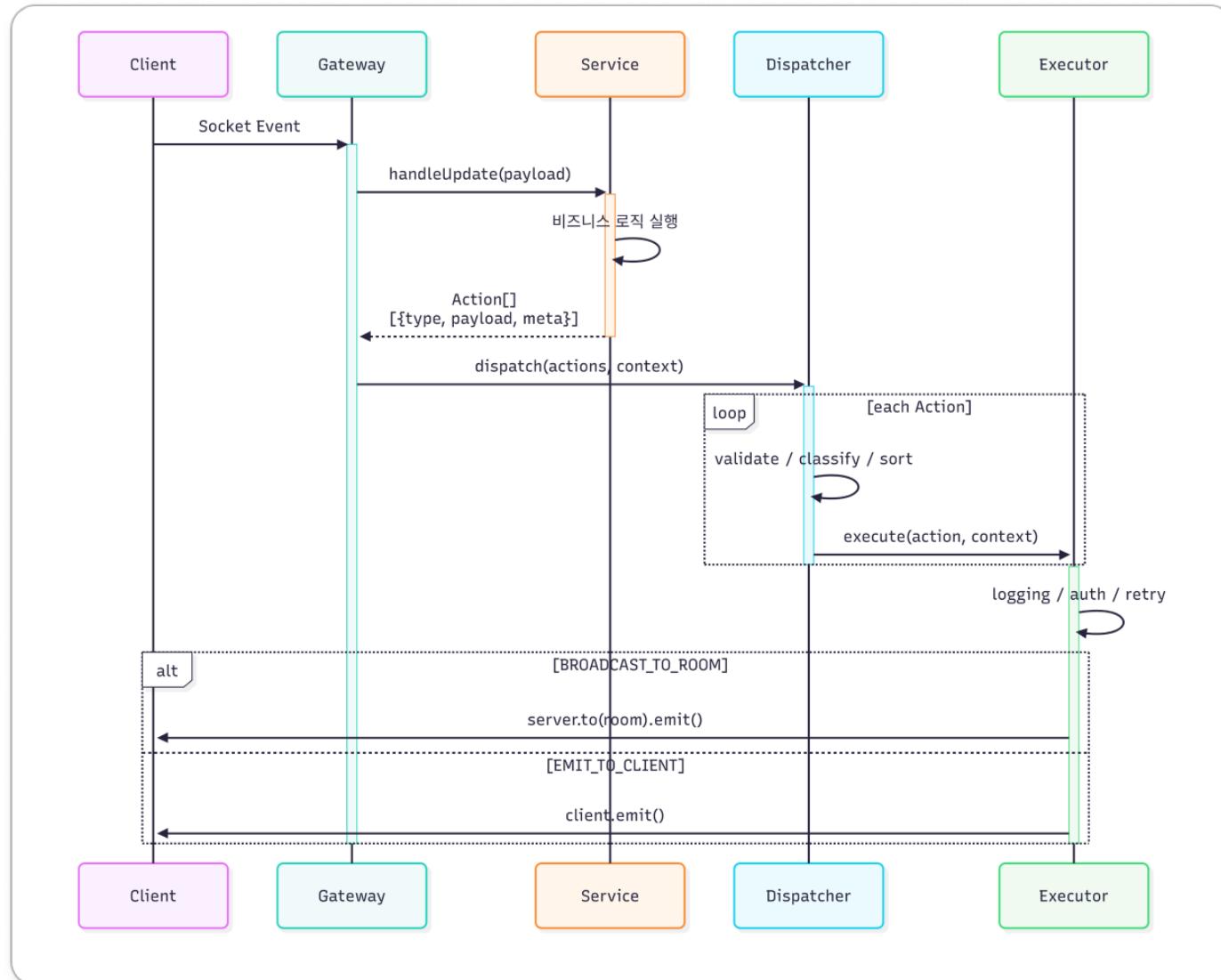
백엔드 리팩터링

Flux-like 아키텍처 검토 중

- 서비스: 순수 비즈니스 로직 수행 후 Action 반환
- Gateway: 클라이언트 요청 수신 및 서비스 호출
- Dispatcher: 중앙에서 Action 수신 및 로깅
- Executor: Action을 Socket.io 명령으로 변환 전송

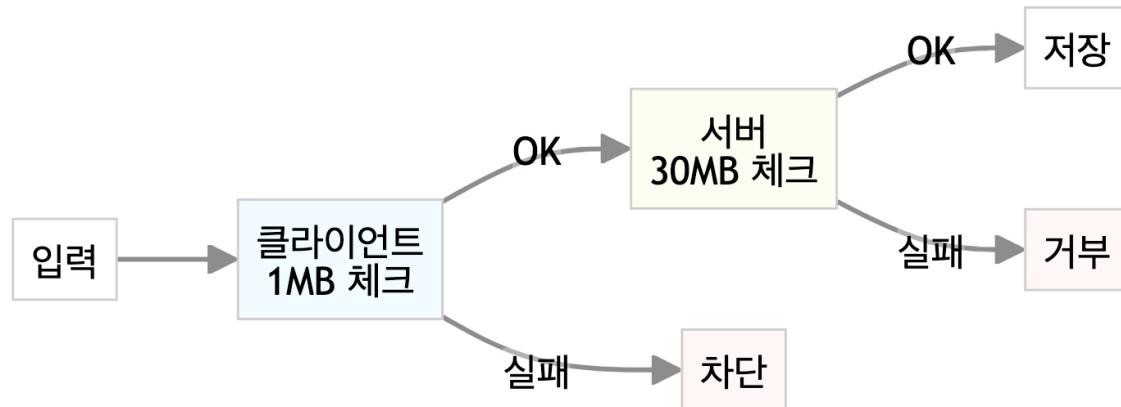


4. 아키텍처 개선 (계속)



5. 보안 강화

1MB 용량 제한 도입



클라이언트 (1MB Soft Limit)

- 용량 게이지: 퍼센트 표시 + 호버 시 "XX KB / 1 MB" 상세 정보
- 입력 차단: 이벤트 핸들러로 선택적 차단

서버 (30MB Hard Limit)

- 제한: 30MB (설계 문서에는 3MB로 적혀있지만 실제 구현은 30MB)
- 위치: `FileService.handleFileUpdate()` 에서 검증

6. 한글 입력(IME) 이슈

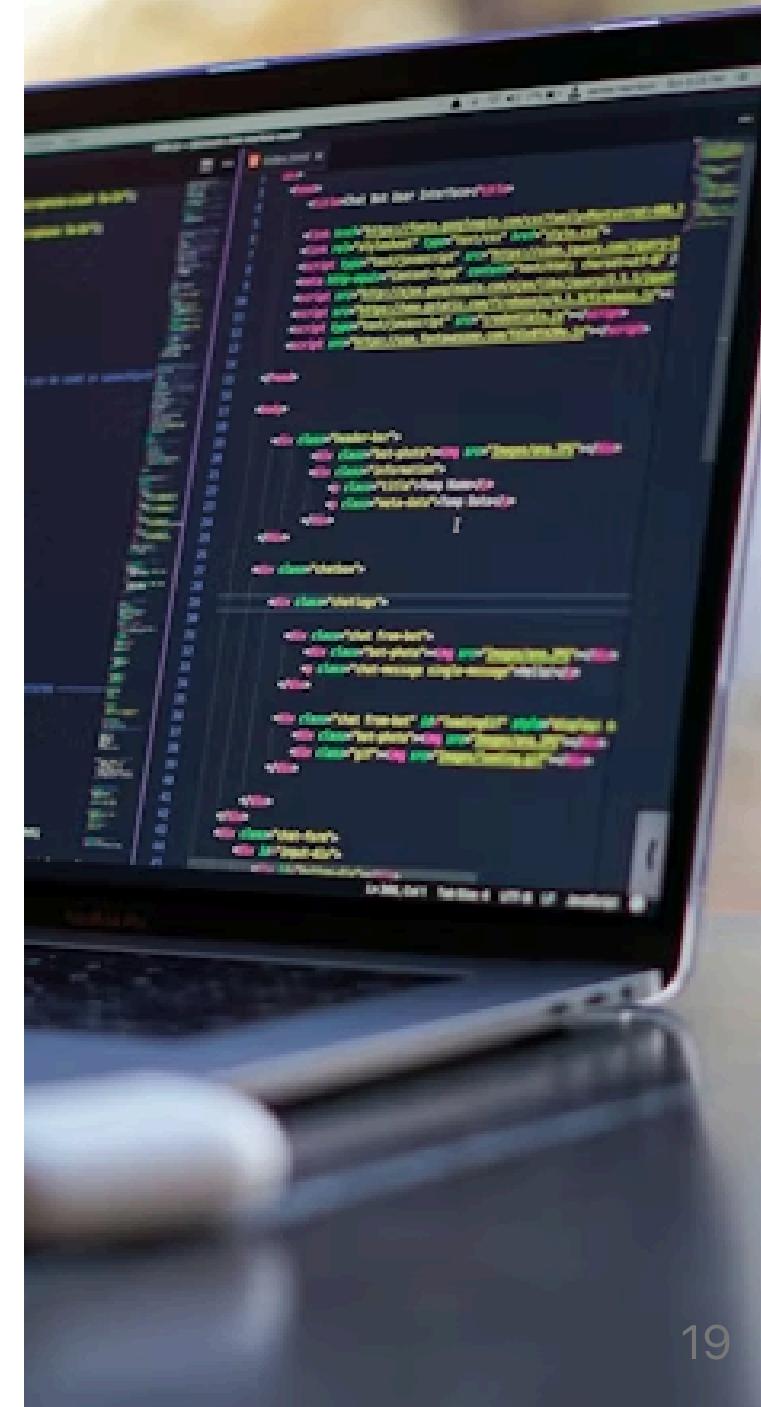
문제 상황

증상

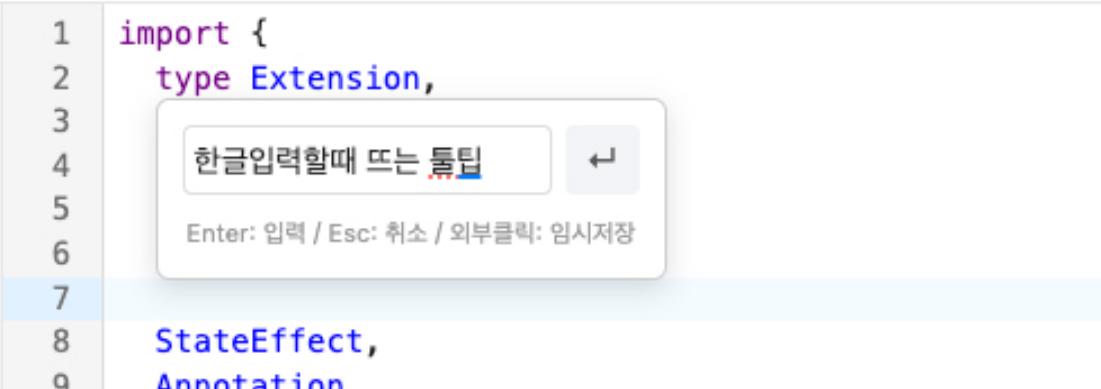
- 한글 조합 중 다른 사용자의 편집 데이터 수신 시 글자 조합 깨짐

원인 분석

- YJS 업데이트가 CodeMirror 뷰에 즉시 반영
- 컴포지션 이벤트 처리 중 외부 변경사항이 끼어들어 조합 상태 깨짐



6. 한글 입력(IME) 이슈 (계속)



A screenshot of a code editor showing a tooltip. The code editor has a light gray background with syntax highlighting. A tooltip box is overlaid on the screen, containing the text "한글입력할때 뜨는 툴팁" (Tooltip that appears when entering Korean input) and a small icon of a hand pointing up. Below the tooltip, there is a status bar with the text "Enter: 입력 / Esc: 취소 / 외부클릭: 임시저장". The code in the editor is:

```
1 import {
2   type Extension,
3
4   한글입력할때 뜨는 툴팁
5
6   Enter: 입력 / Esc: 취소 / 외부클릭: 임시저장
7
8   StateEffect,
9   Annotation,
10 } from '@codemirror/state'.
```

해결 방안

시도한 방법

- 컴포지션 세션 중 YJS 업데이트를 큐에 저장
- 세션 종료 시 fast-diff로 최종본과 스냅샷 비교 후 일괄 반영
- CodeMirror Extension 형태로 개발 (플러그인 분리 가능)

현재 상태

- 근본적 해결 어려움
- 임시 해결책: ASCII 외 언어는 툴팁 입력창에서 별도 입력 후 붙여 넣기 방식 우회

7. 데모

구현 완료 기능

방 관리

- 퀵 스타트 방 생성
- roomCode로 입장
- 방 링크 공유

실시간 협업

- CRDT 기반 동시 편집
- 멀티 커서 표시
- 참가자 목록 실시간 업데이트

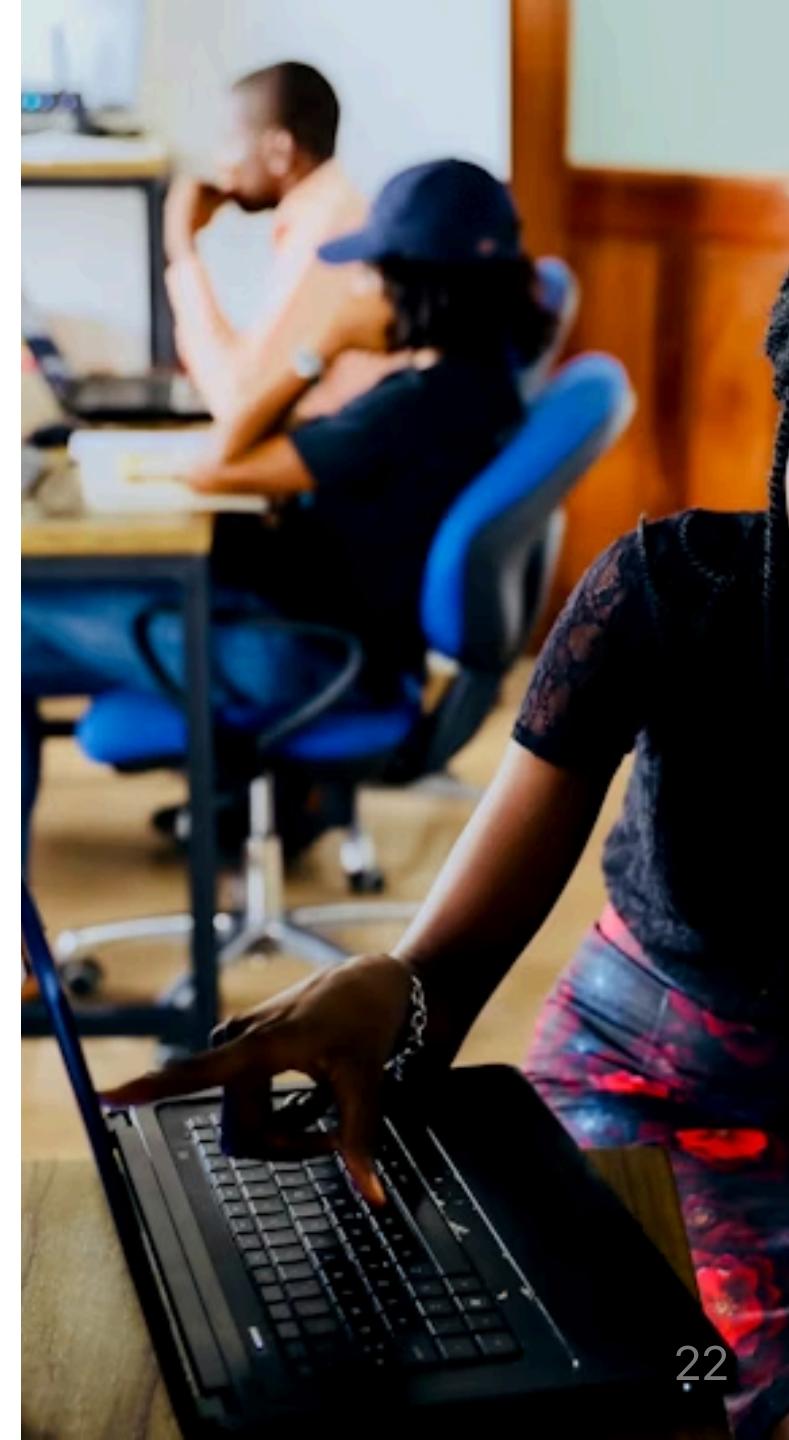
파일 시스템

- 다중 파일 업로드
- 파일 이름 중복 처리
- 파일 탭 전환

7. 데모 (계속)

시연 예정

1. 방 생성 및 접속
2. 다중 파일 업로드 및 편집
3. 여러 사용자 동시 편집 - 영어 입력이 편해요...!!
4. 호스트 권한 부여/회수



8. 향후 계획

남은 개발 기간

이번 주 목표

- Y-Redis 문서 복구 시스템 구축
- JWT 기반 인증 시스템 완성
- 파일 시스템 UI 구현
- CI/CD 파이프라인 구축
- 1MB 용량 제한 도입

남은 Must 기능

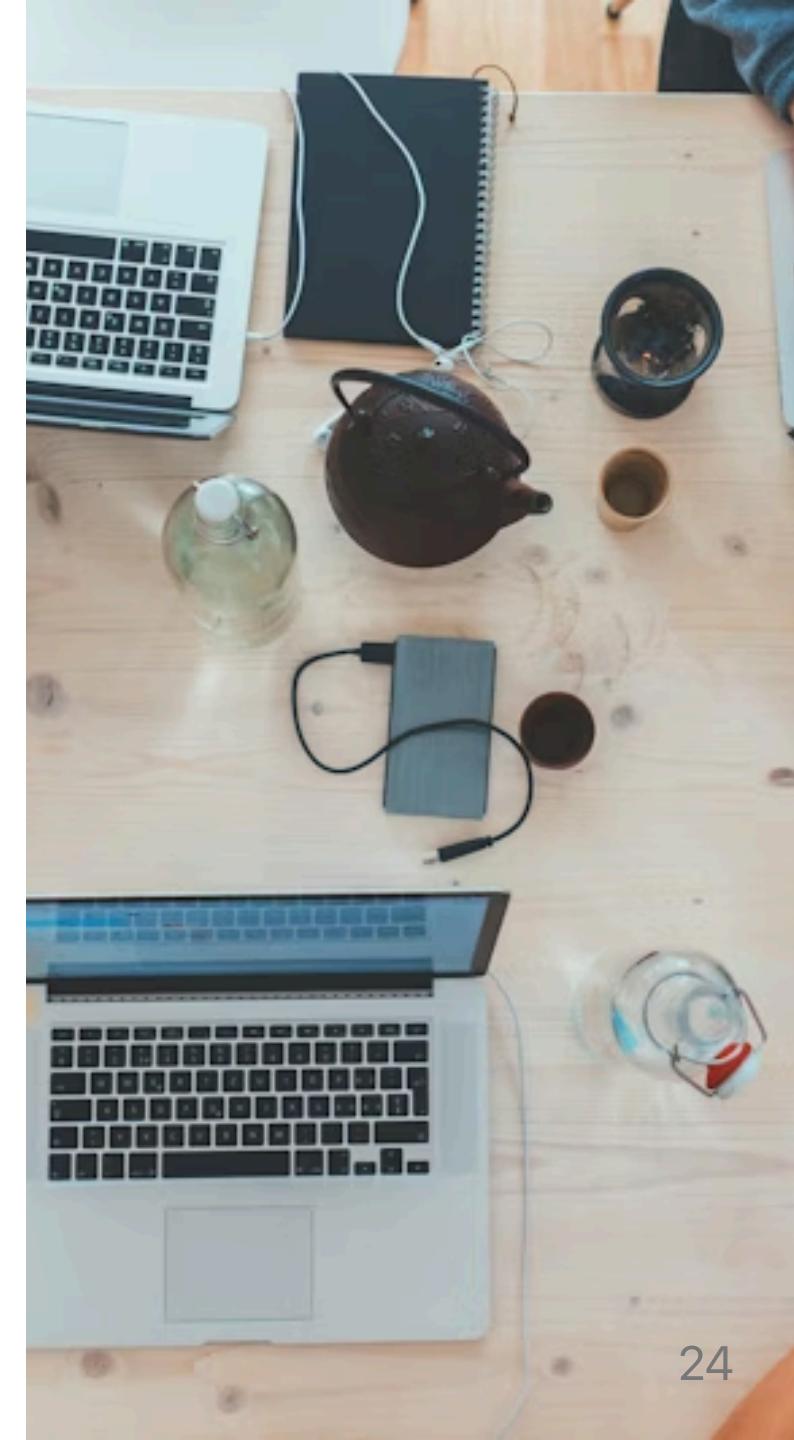
- Custom Room 생성
- 코드 실행 (Piston API)
- 채팅 기능
- 북마크 및 유저 위치 추적
- 한글 입력 이슈 완전 해결



8. 향후 계획

5주차 개발해야할 것

1. 방 만들기 할 때 Custom 설정 가능



9. 4주간 배운 점

기술적 성장

실시간 동기화 심화

- Y-Redis 아키텍처 이해
- Clock 알고리즘 및 동시성 제어
- 영속성 전략 수립

보안 및 성능

- JWT 인증 시스템 구축
- 이중 방어 체계 설계
- DB/Redis 역할 분리 전략



9. 4주간 배운 점 (계속)

협업 및 프로젝트 관리

PR 단위 축소의 중요성

- 작은 단위로 자주 공유하는 것이 전체 생산성 향상

초안 공유의 효과

- 작업에 들어가기 전 디스커션에 초안 공유
- 충돌이 덜 일어남
- 역할 분리가 명확해짐



10. Q&A

받고 싶은 피드백

1. 오류를 많이 많이 찾아주세요
2. 한글 입력 이슈의 임시 해결책이 크게 불편하지 않은가요? (봐주세요 🙏)
3. 1MB 용량 제한이 적정하다고 느끼시나요?
4. 기타 추가되었으면 하는 기능이 있으시다면 편하게 의견 남겨 주세요.

질문 주시면 답변드리겠습니다.

감사합니다!

