

CodeJam

실시간 협업 코딩 플랫폼

Team JAMstack | 3주차 데모 (2025.01.09)

J044 김선향

J094 노주호

J102 민인애

J141 손혜준

J143 송상화

목차

1. 서비스 소개
2. 3주간의 여정
3. 시니어 리뷰어 피드백
4. 핵심 기술 구현
5. 아키텍처 개선
6. 데모
7. 향후 계획
8. Q&A



1. 서비스 소개

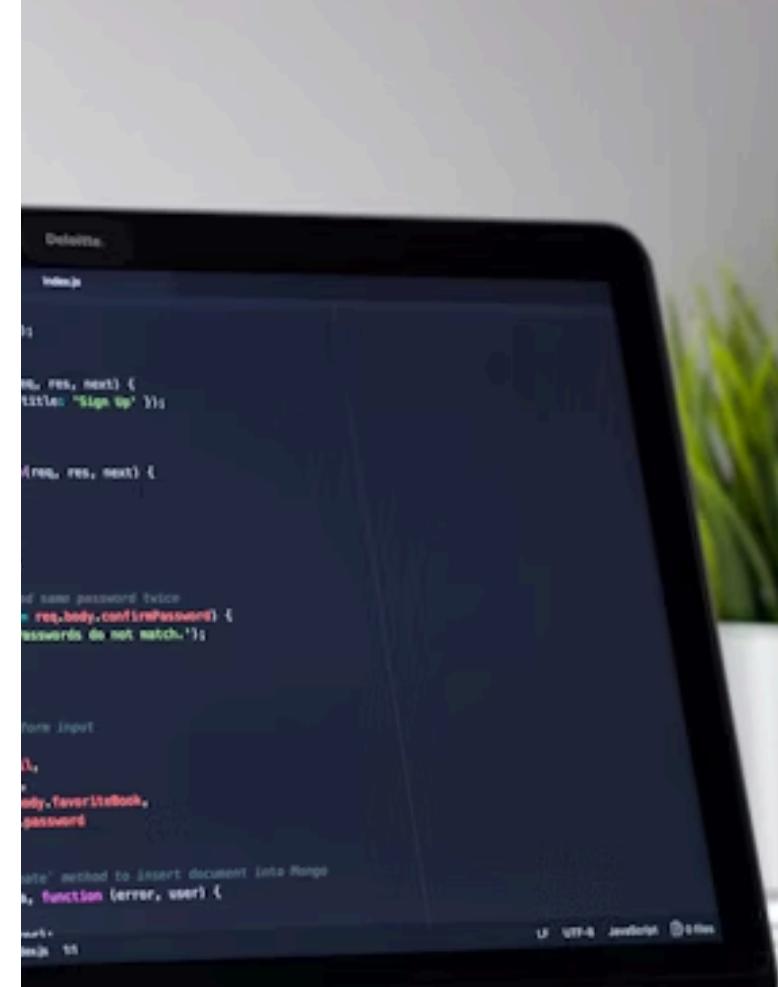
CodeJam - 실시간 협업 코딩 플랫폼

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)



2. 3주간의 여정

주요 마일스톤

1주차 (12월 8일 ~ 12월 12일)

- 서비스 기획 및 문제 정의
- 와이어프레임 제작
- 데모 데이 발표

2주차 (12월 16일 ~ 12월 20일)

- 기술 스택 확정
- Monorepo 환경 구성
- YJS 실시간 동기화 PoC

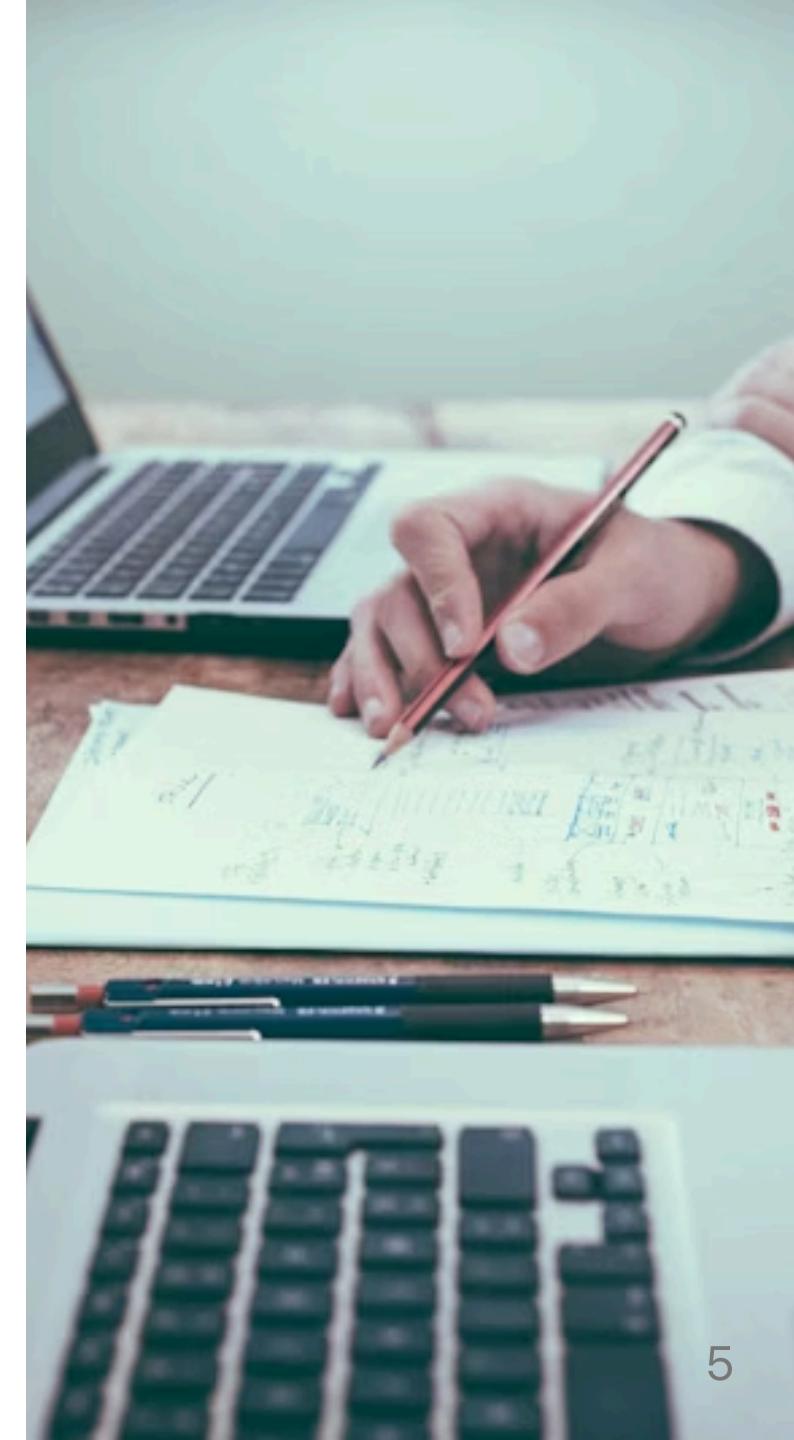


2. 3주간의 여정 (계속)

주요 마일스톤

3주차 (1월 5일 ~ 1월 9일)

- 시니어 리뷰어 피드백 (12월 21일)
- 인터미션 기간 기획 재정비
- MoSCoW 우선순위 수립
- 아키텍처 리팩터링
- DB 설계 및 구현
- 권한 시스템 구축

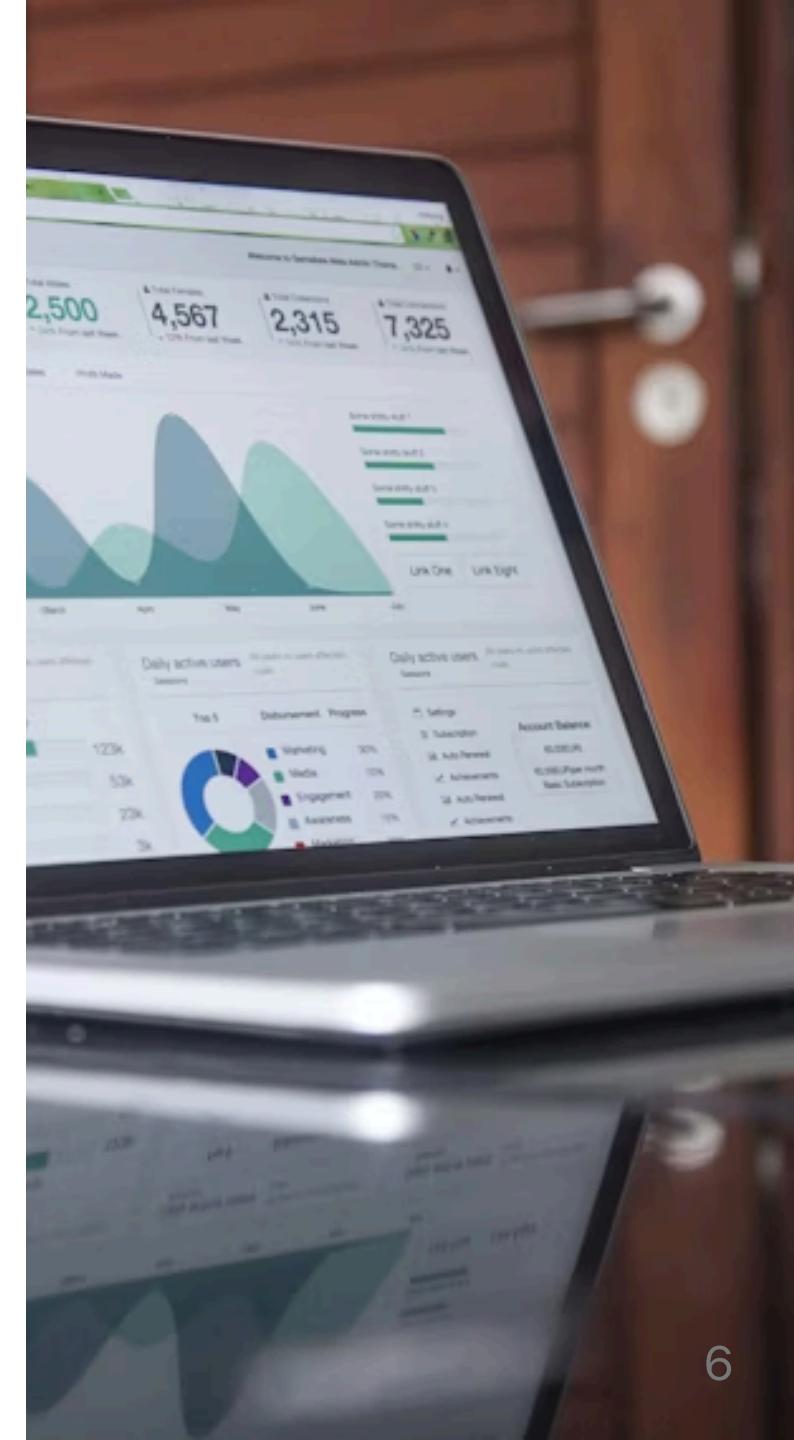


3. 시니어 리뷰어 피드백

피드백 핵심 포인트

1. 명확한 일정 산정

- 피드백: 주차별 일정 부재가 리스크
- 반영: 4주 스프린트 계획 수립
 - 1주차: 퀵스타트 방 생성
 - 2주차: 커스텀 방 생성
 - 3주차: 파일 시스템 & 동시 편집
 - 4주차: 코드 실행 완성



3. 시니어 리뷰어 피드백 반영 (계속)

2. 권한 시스템 확장

- 피드백: 선착순 6명에서 마스터 권한 필요
- 반영: 호스트 중심 권한 체계 구축
 - Host > Editor > Viewer 계층 구조
 - 호스트 비밀번호로 권한 복구
 - 편집 권한 부여/회수 기능

3. 데이터 복구 전략

- 피드백: 서버 재시작 시 데이터 손실 위험
- 반영: 스냅샷 & 리플레이 아키텍처 설계
 - 서버 자동 저장 방식 (20개 제한)
 - DB 기반 영속성 확보

3. 시니어 리뷰어 피드백 반영 (계속)

4. 성능 목표 구체화

- 피드백: 수치화된 성과 기록 필요
- 반영: 명확한 성능 목표 설정
 - 개별 방 최대 150명
 - 서버 전체 동시 접속 1,000명
 - 평균 LCP 1.0~1.5초 이내

5. 보안 강화

- 피드백: 악성 사용자 차단 및 Rate Limit
- 반영: 보안 정책 수립
 - nanold 기반 추측 불가능한 roomCode
 - IP 기반 Rate Limit 검토
 - 샌드박스 환경 코드 실행

4. 핵심 기술 구현

1. DB 설계 및 최적화

roomCode 식별자 추가

- room_id (내부 FK용) + nanoid 기반 roomCode (외부 노출용) 분리
- 대문자+숫자만 사용하여 가독성 향상 (21억 조합)
- 중복 방지: 최대 3번 재시도 로직
- 성능: 저장 전 Select로 중복 체크하여 쿼리 러너 점유 최소화

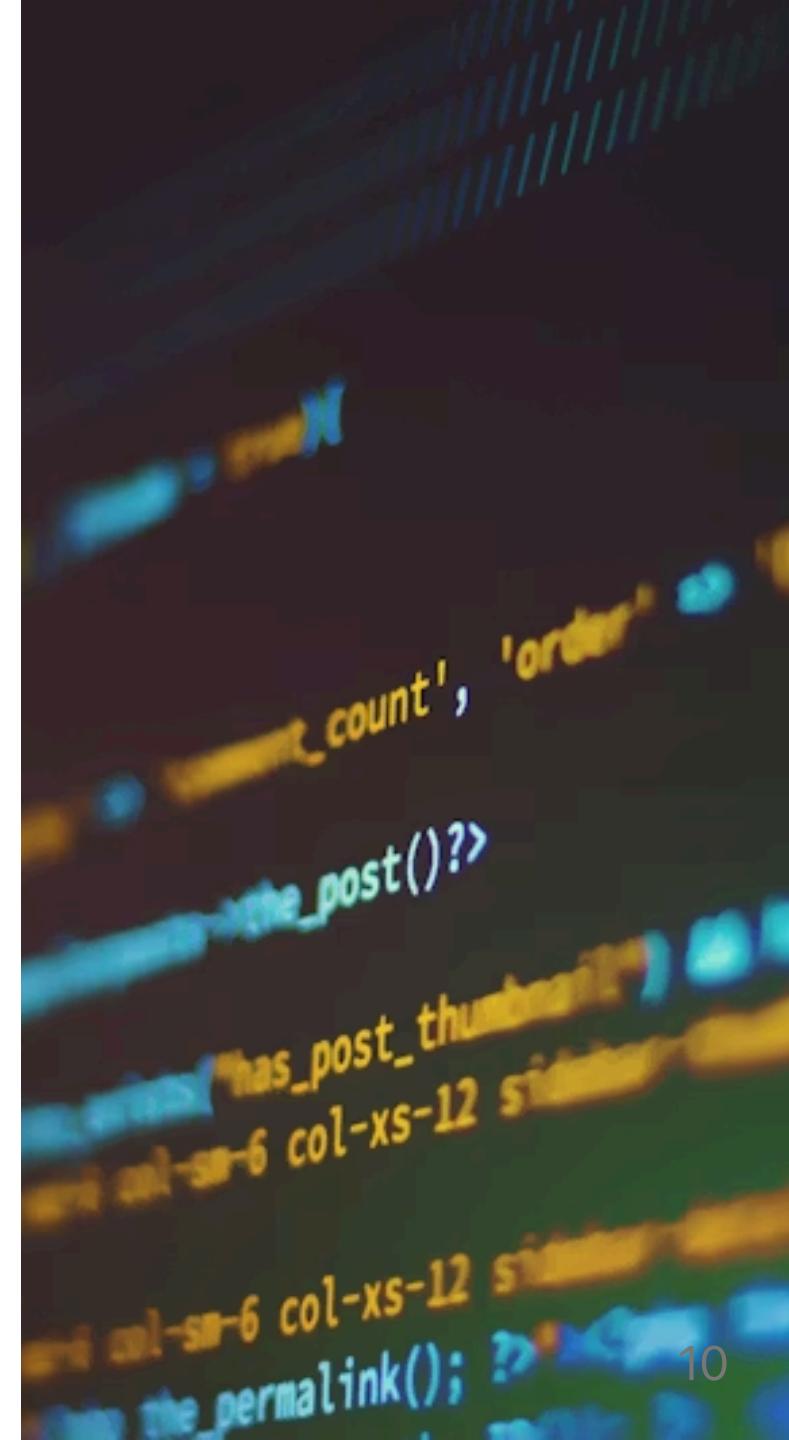


4. 핵심 기술 구현 (계속)

2. 권한 시스템

소켓 메모리 기반 권한 관리

- Redis 대신 소켓 메모리에 권한 직접 저장
 - 속도 향상
 - 클라이언트 수정 불가 (보안 강화)
- 서버 재시작 시 클라이언트 재접속하면 DB에서 권한 정보 조회하여 복구



4. 핵심 기술 구현 (계속)

3. 호스트 권한 로직

원자적 방 생성

- 방 생성 시 방장을 트랜잭션으로 즉시 참여자에 포함
- 권한 탈취 시나리오 방지

호스트 권한 양도

- 호스트 비밀번호 설정 시: 양도 정책 선택 가능 (공석/자동 양도)
- 비밀번호 미설정 시: 입장 순서대로 자동 양도
- 권한 스위치 메커니즘 구현



4. 핵심 기술 구현 (계속)

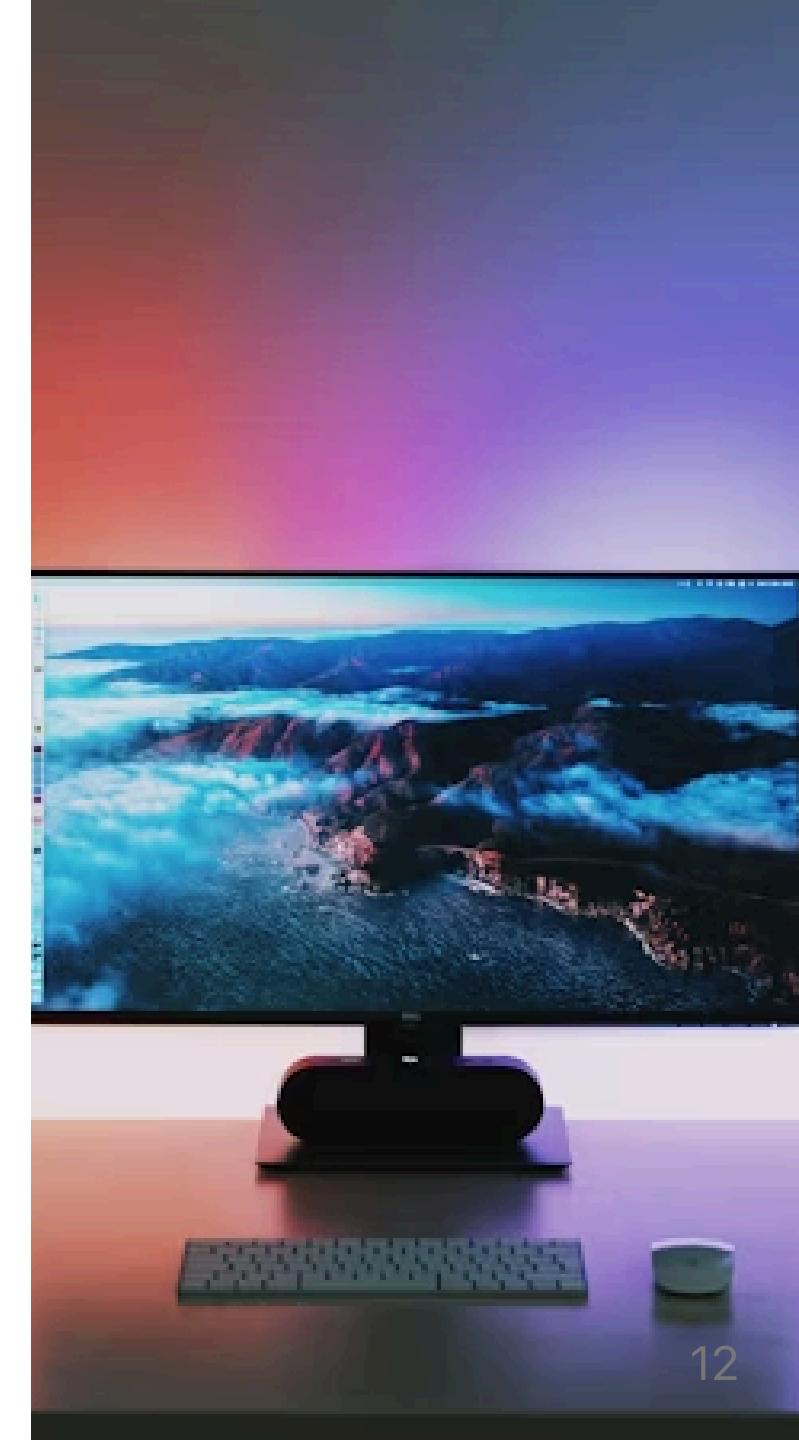
4. 멀티 파일 시스템

단일 Y.Doc 구조

- 하나의 Y.Doc 안에 여러 파일 텍스트+메타데이터 저장
- 1MB 제한 내에서 충분한 확장성
- 구현 용이성 & 성능 우위

파일명 중복 처리

- 업로드 시: overwrite/ rename 선택
- 수정 시: 기존 파일과 겹치면 경고 및 차단
- rename 규칙: 원본이름 (1), 원본이름 (2)



4. 핵심 기술 구현 (계속)

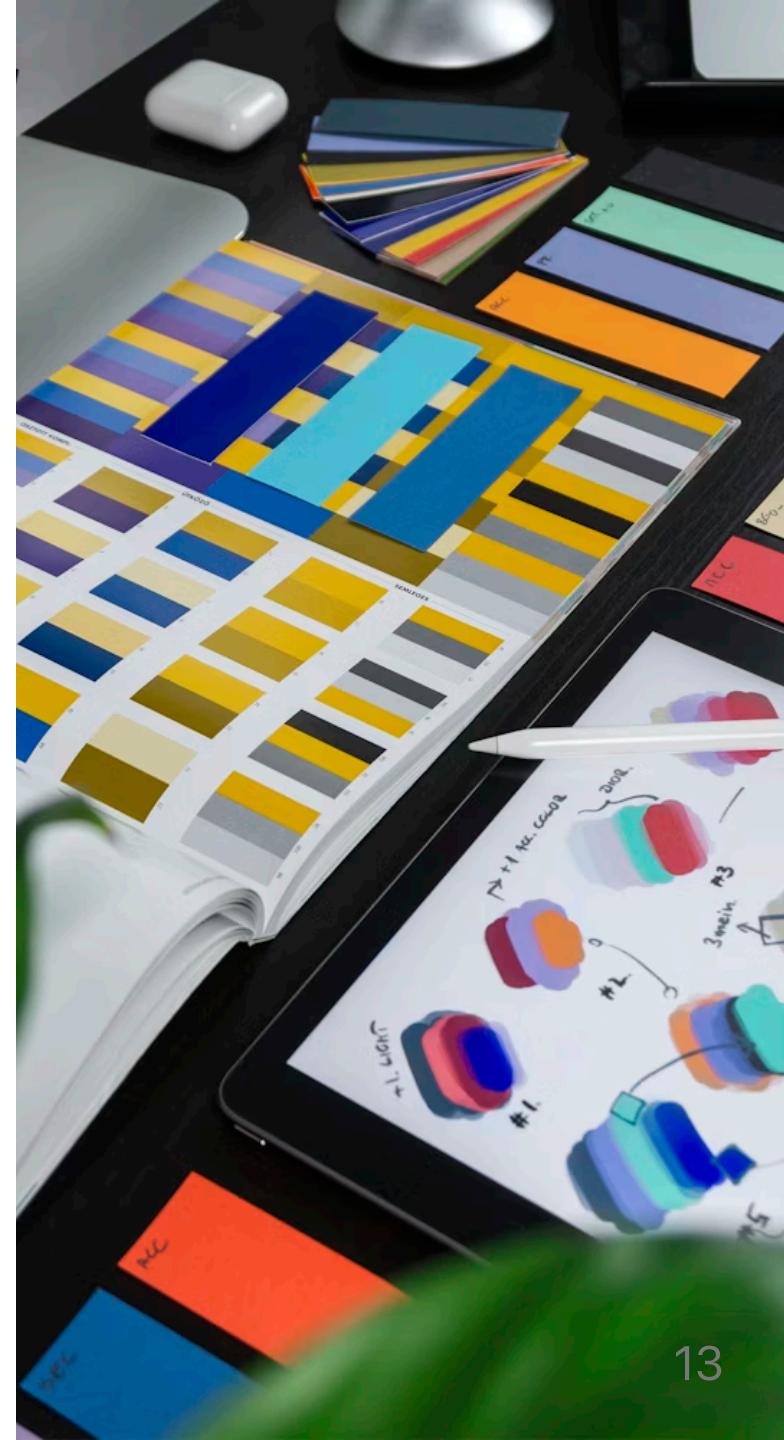
5. UI/UX 개선

메인 페이지

- 퀵 스타트 버튼 (즉시 방 생성)
- 커스텀 옵션 버튼 (세부 설정)
- 모노스페이스 폰트

방 번호 입장

- 6자리 입력 후 버튼 클릭 시 API 호출하여 검증
- 0(영)/O(오), 1(일)/I(엘) 헷갈림 방지를 위해 대문자만 사용



5. 아키텍처 개선

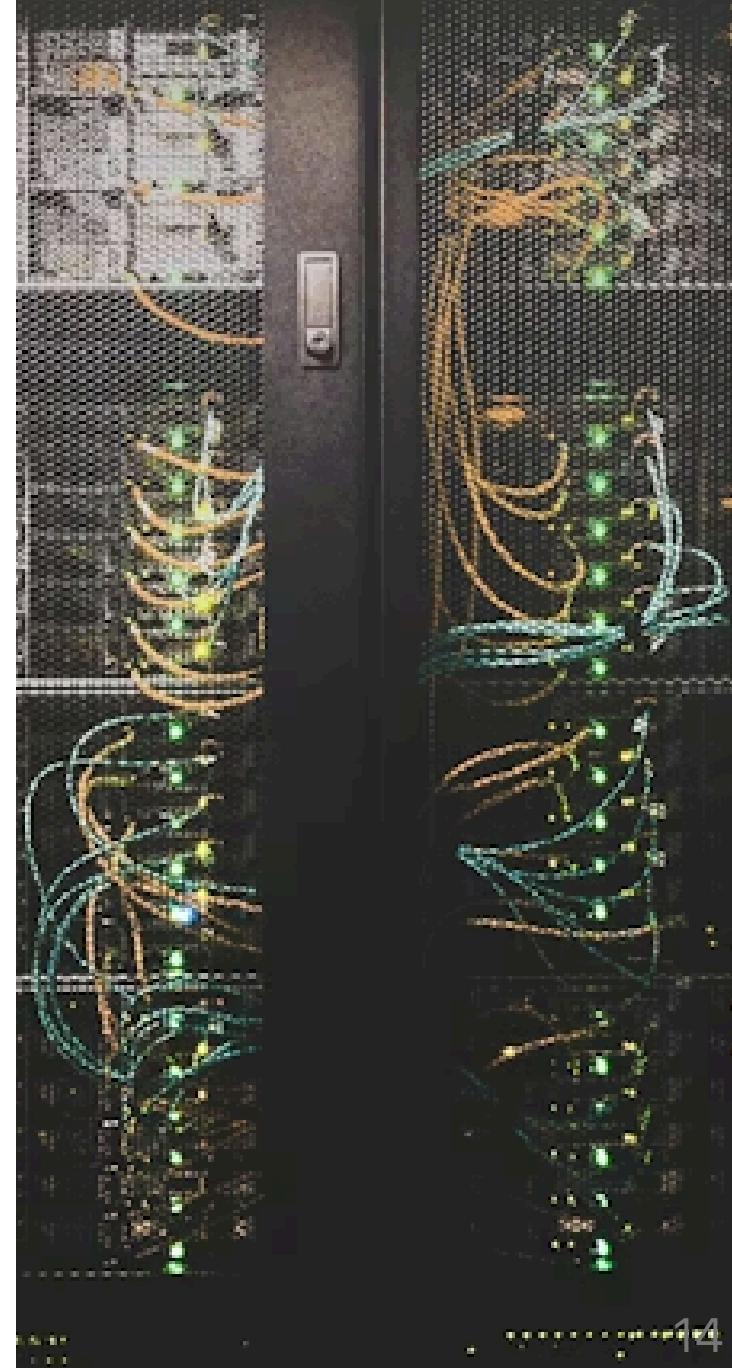
백엔드 리팩터링

모듈 분리 강화

- 게이트웨이: 얇게 유지, 소켓 연결만 관리
- 서비스 분리: Room / File / Participant / Permission
- 순환 참조 문제 해결

DB 역할

- DB: 방 설정, 비밀번호, 양도 정책 등 영속성 데이터



5. 아키텍처 개선 (계속)

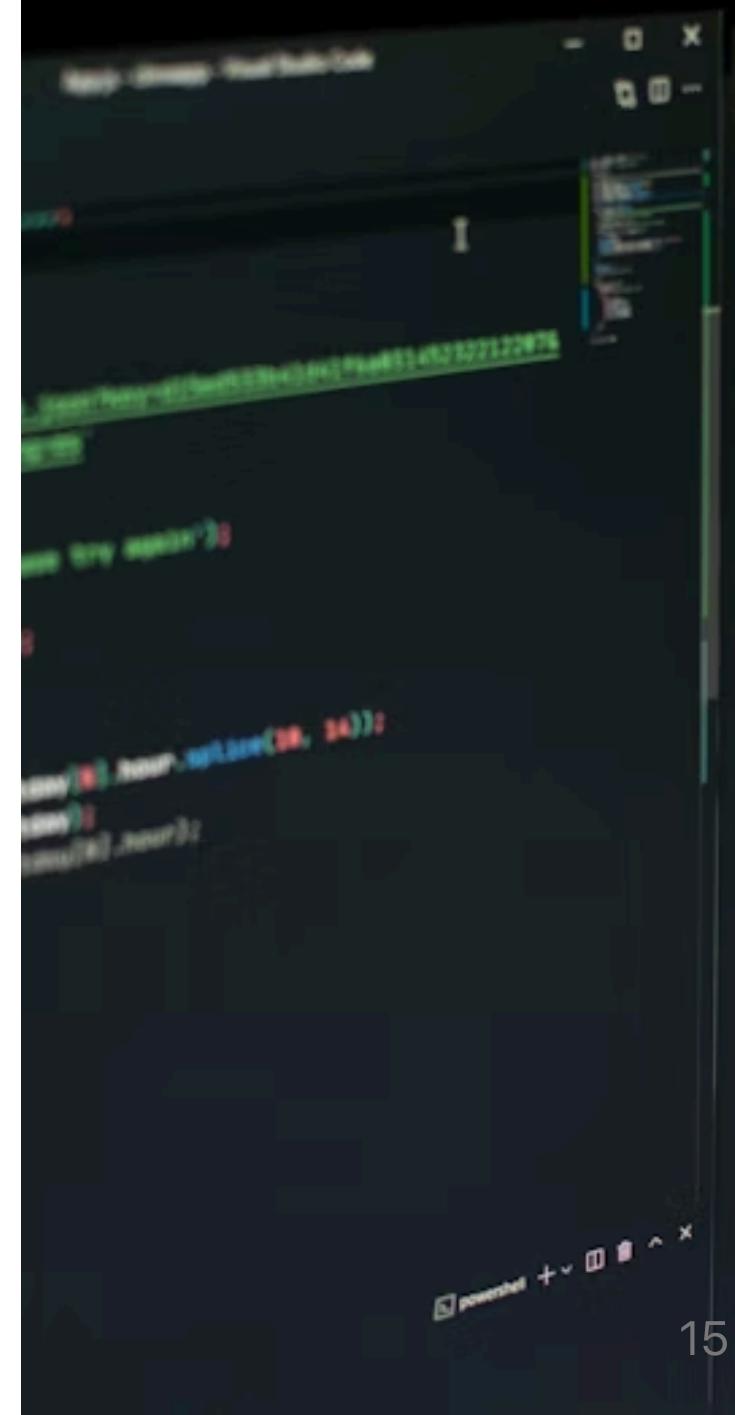
프론트엔드 구조

스토어 분리

- 소켓 스토어: 연결 관리
- 파일 스토어: 데이터 관리
- 명시적 흐름: 소켓 연결 → Y.Doc 생성 → 데이터 요청

공통 타입 관리

- Zod 도입 검토
 - 런타임 유효성 검증 + 타입 추론
 - 프론트/백엔드 동일한 검증 로직



5. 아키텍처 개선 (계속)

팀 역할 분담 (기술 책임자 제도)

분야별 전문성 확보

- **인애:** 워크스페이스 & 파일 매니저 (파일 트리, 탭 시스템)
- **혜준:** 세션 & 권한 가디언 (방 생명주기, 권한 로직)
- **선향:** 동시성 CRDT 아키텍트 (Yjs, 멀티 커서)
- **상화:** 백엔드 & 데이터 복구 매니저 (영속성, 스케줄러)
- **주호:** 에디터 코어 스페셜리스트 (CodeMirror, DX)

협업 균형

- 모든 팀원이 함께 개발
- 책임자는 파악 및 의사결정에 집중



6. 데모

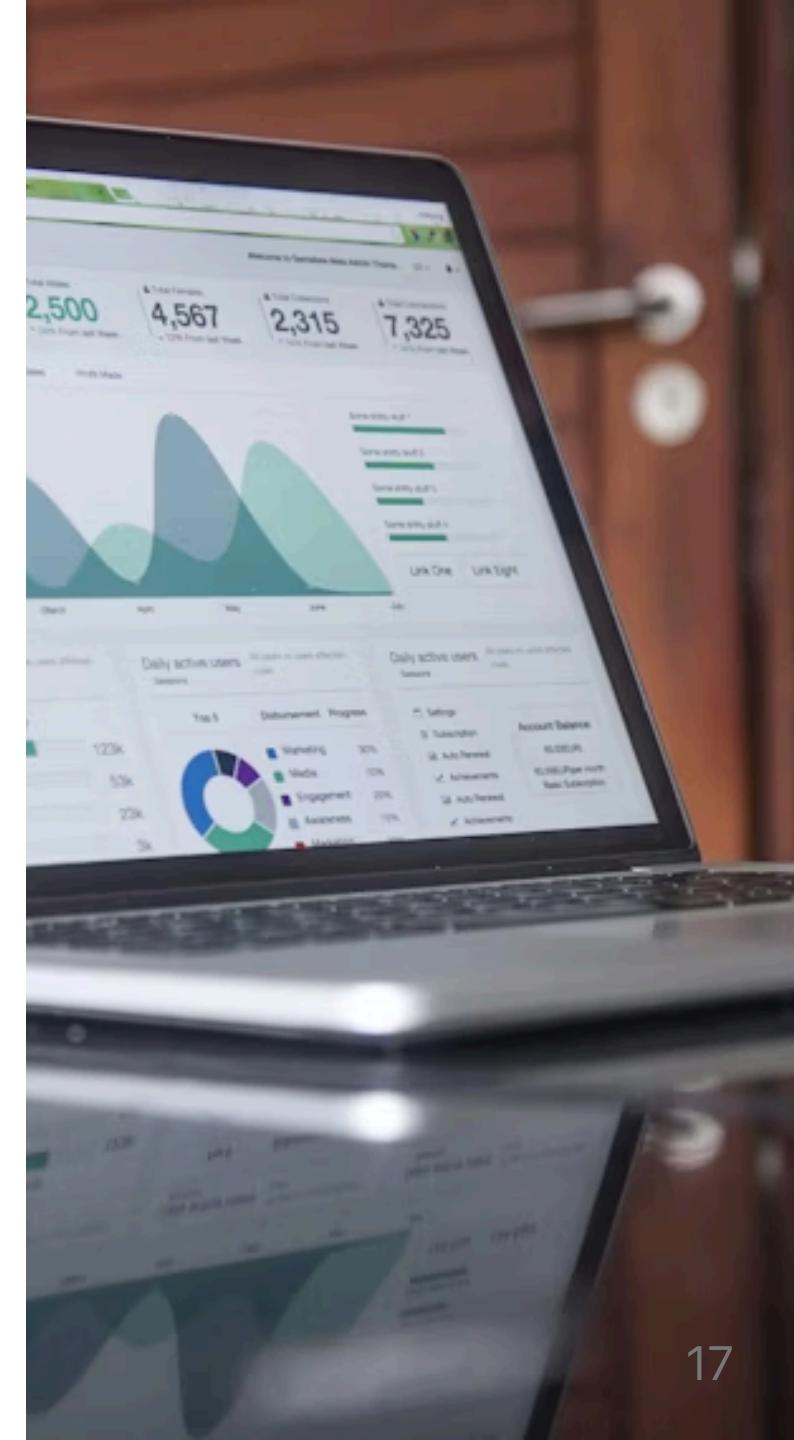
구현 완료 기능

방 관리

- 퀵 스타트 방 생성
- roomCode로 입장
- 방 링크 공유
- 호스트 권한 관리

실시간 협업

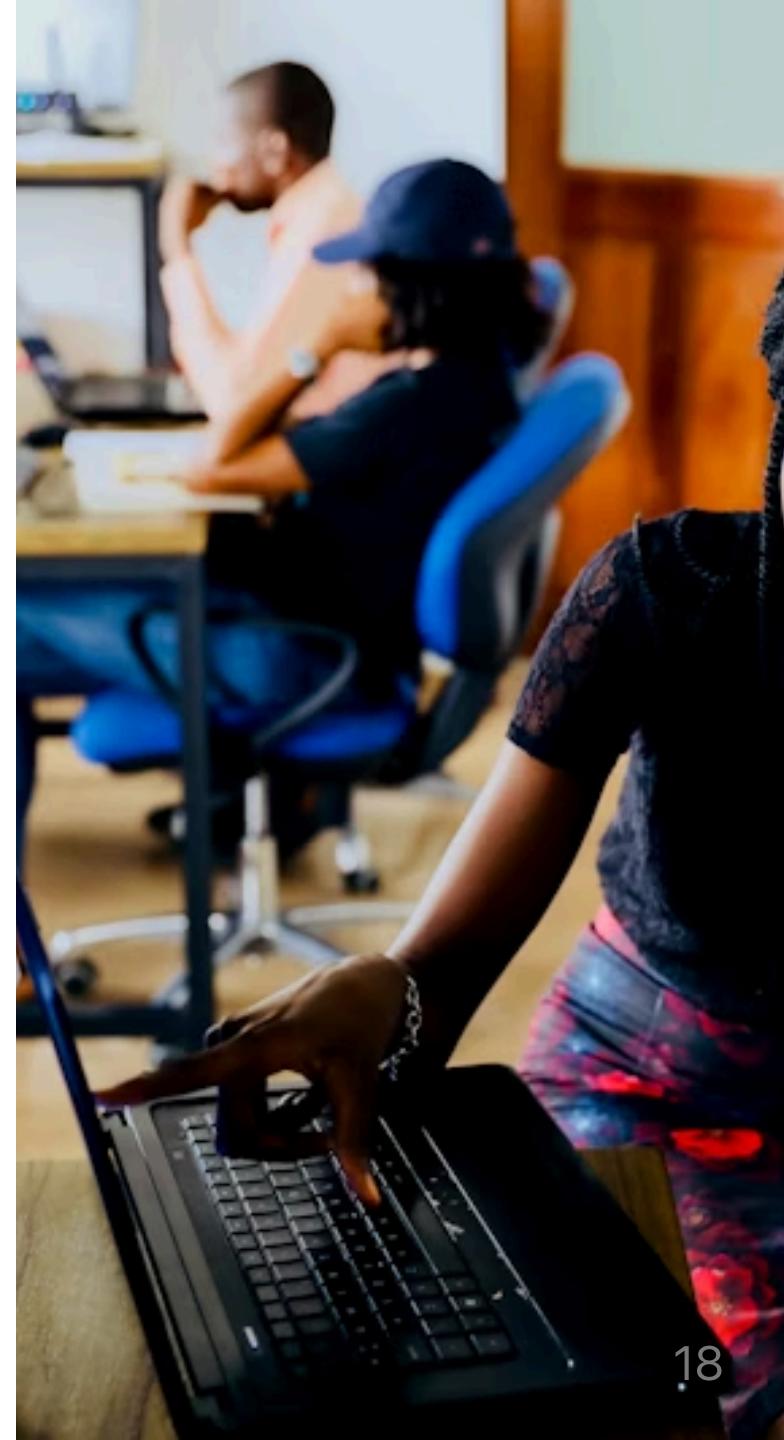
- CRDT 기반 동시 편집
- 멀티 커서 표시
- 참가자 목록 실시간 업데이트



6. 데모 (계속)

시연 예정

1. 방 생성 및 접속 (roomCode 공유)
2. 여러 사용자 동시 편집
3. 호스트 권한 부여/회수
4. 멀티 커서 동기화



7. 향후 계획

남은 개발 기간

이번 주 목표

백엔드 PR 통합 및 동작 확인

배포 준비 및 실행

데모 준비 및 피드백 수집

Quick Room 생성

남은 Must 기능

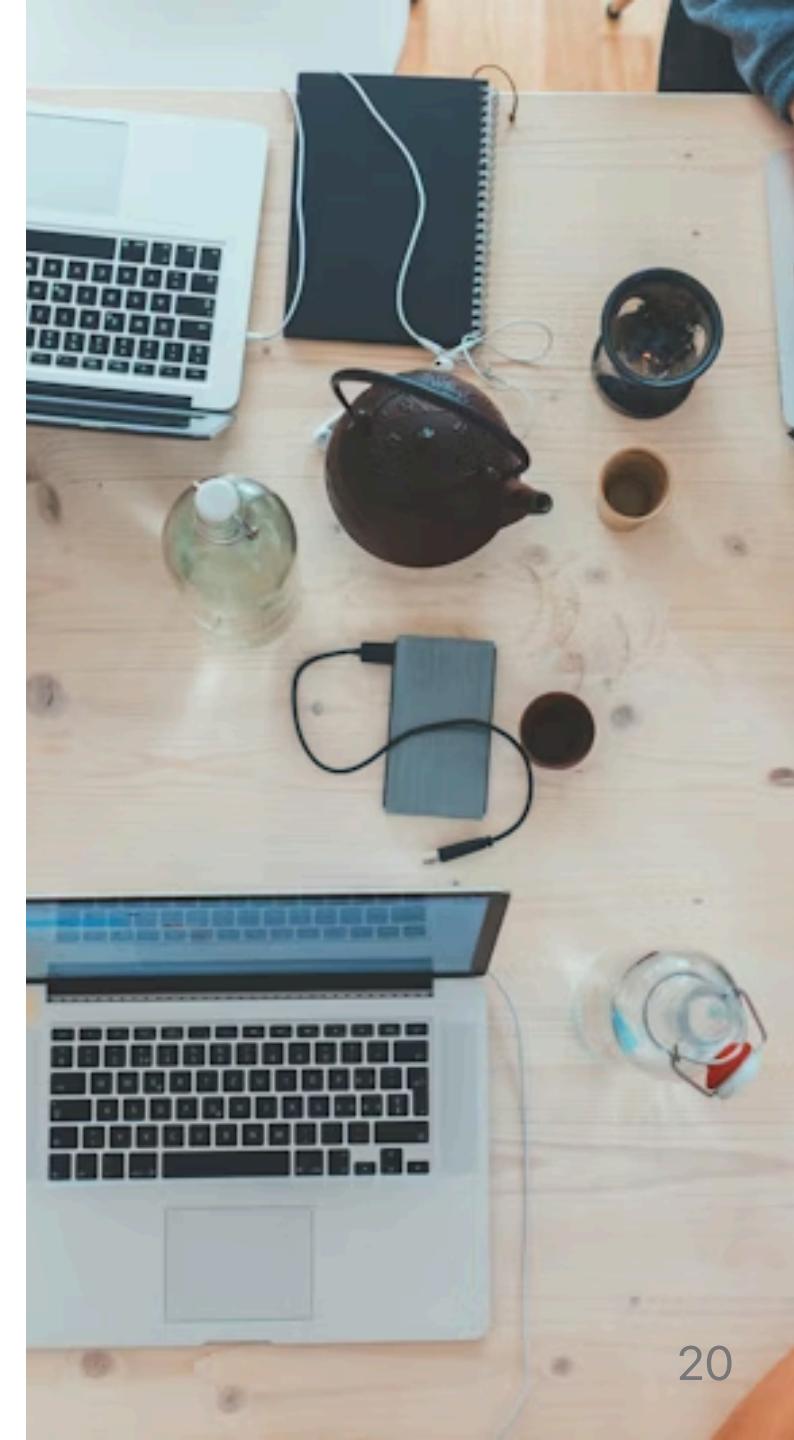
- Custom Room 생성
- 코드 복구 (서버 재시작 시)
- 코드 실행 (코드 실행 엔진, Piston API 유력)
- 채팅, 북마크, 유저 위치 추적



7. 향후 계획 (계속)

최종 계획

- 100~150명 동시 접속 부하 테스트
- 성능 최적화 (번들 사이즈, LCP)
- 보안 강화 (Rate Limit, 샌드박스)



8. 3주간 배운 점

기술적 성장

CRDT 기반 실시간 동기화

- Yjs 아키텍처 이해
- 바이너리 데이터 전송 최적화
- 동시성 문제 해결

아키텍처 설계

- Monorepo 구조 설계
- 모듈 분리 및 의존성 관리



8. 3주간 배운 점 (계속)

협업 및 프로젝트 관리

MoSCoW 우선순위

- Must/Should/Could 기준 명확화
- Impact-Effort Matrix 활용

기술 책임자 제도

- 분야별 전문성 확보
- 의사결정 효율화

시니어 리뷰어 피드백

- 일정 산정의 중요성
- 수치화된 성과 기록



Q&A

받고 싶은 피드백

1. 메인 페이지 UI 시안에 대한 의견
2. 퀵 스타트 vs 커스텀 옵션 배치 방식이 직관적인가요?
3. roomCode 6자리 입력 방식의 사용자 경험은 어떤가요?
4. 전체적인 서비스 정체성이 잘 드러나나요?

질문 주시면 답변드리겠습니다.

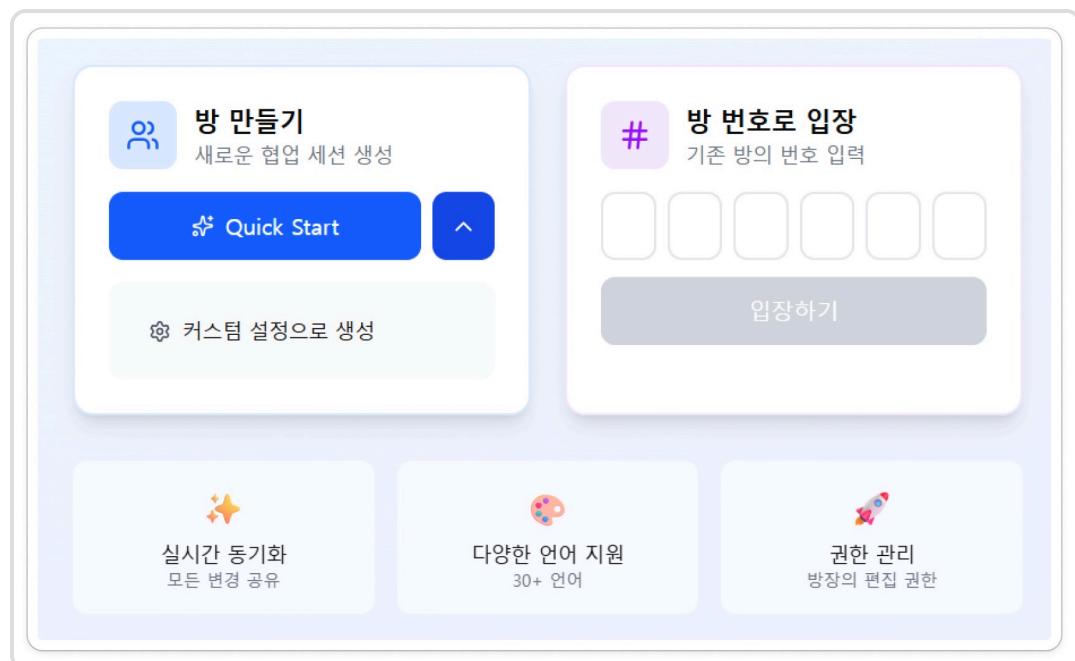
감사합니다! 🎉



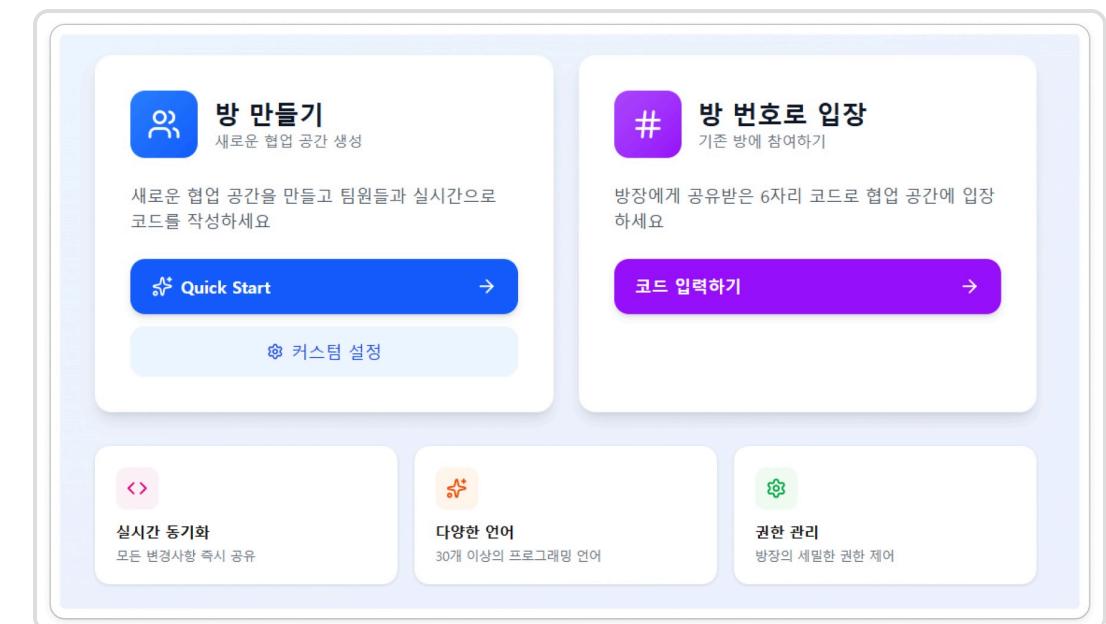
UI 시안 투표

메인 페이지 디자인 선호도 조사

어떤 디자인이 가장 마음에 드시나요?

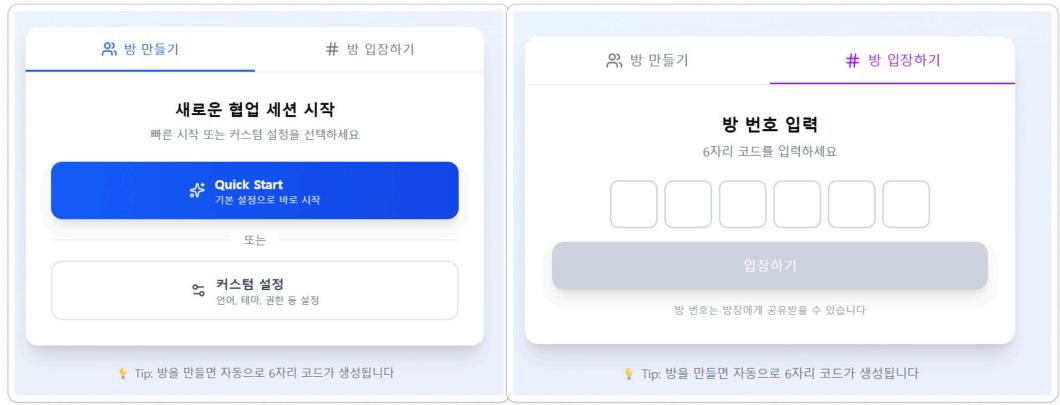


1번

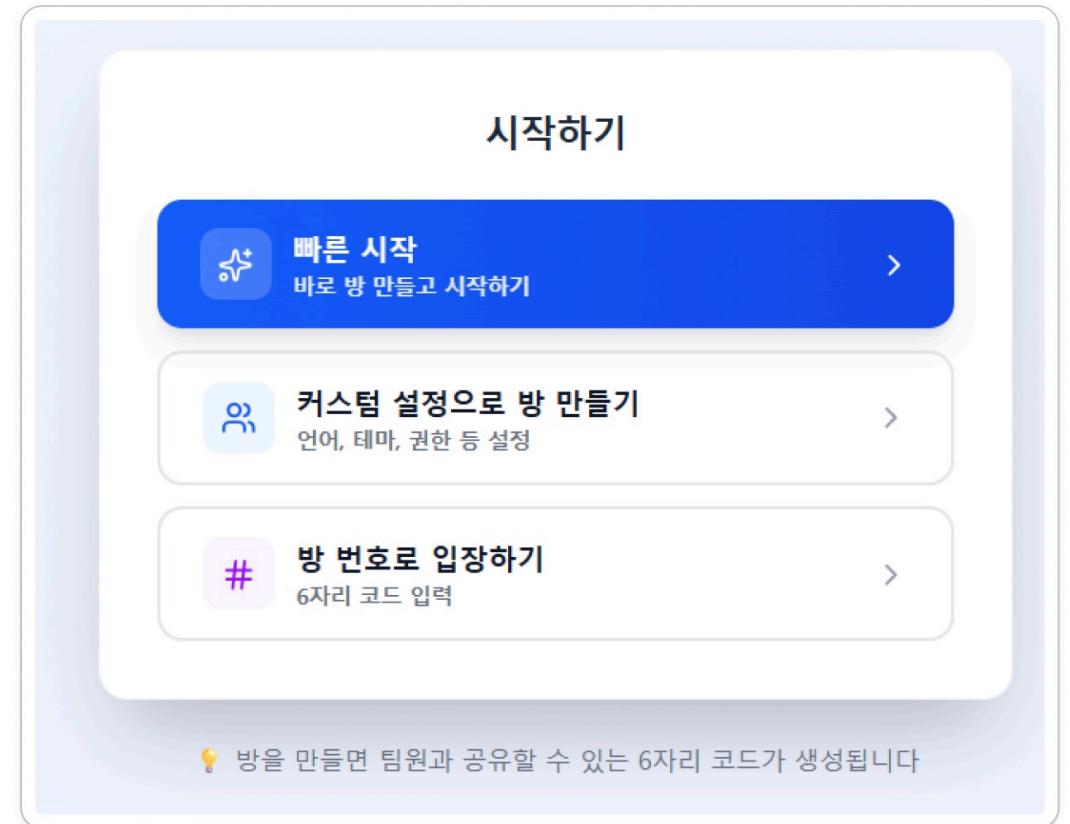


2번

UI 시안 투표 (계속)

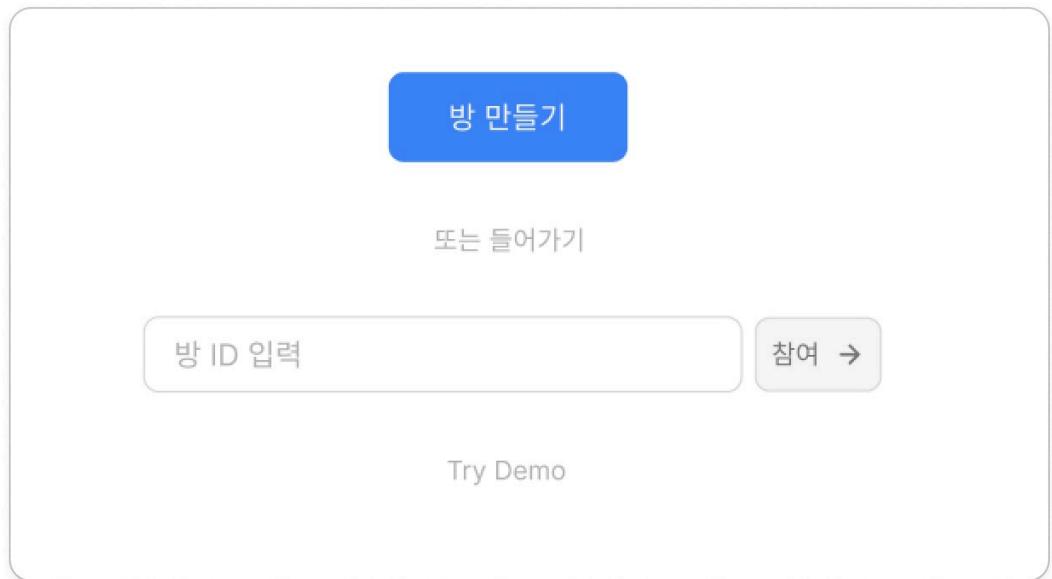


3번

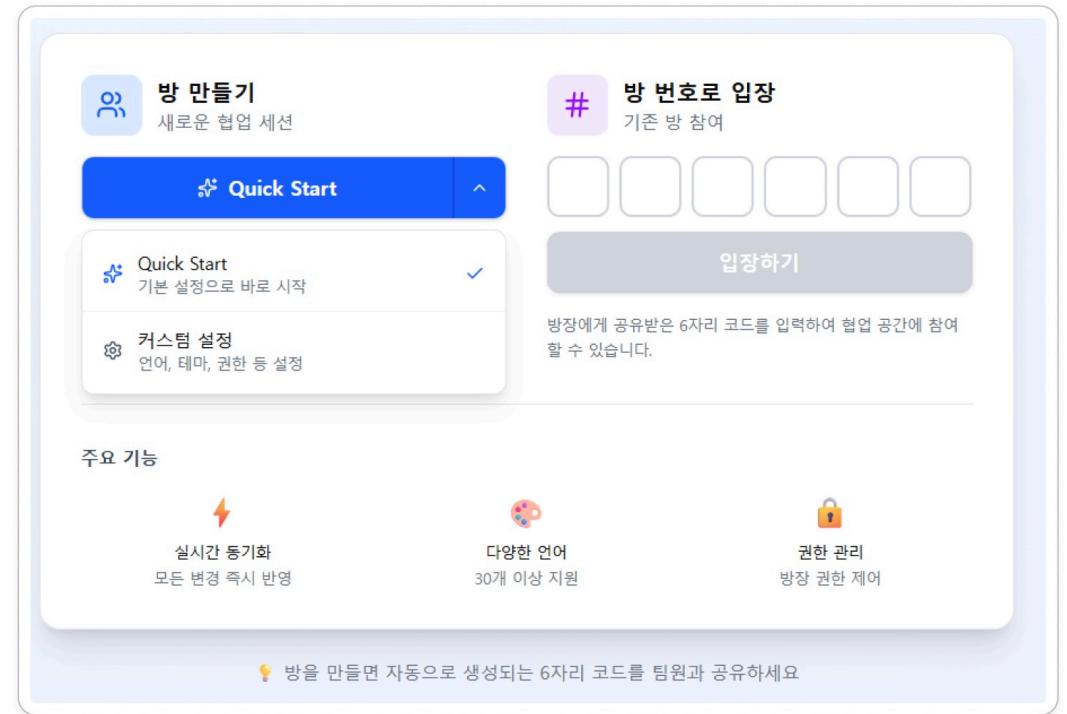


4번

UI 시안 투표 (계속)



5번



6번