

# CodeJam

실시간 협업 코딩 플랫폼

Team JAMstack | 2주차 데모 (2025.12.19)

J044 김선향

J094 노주호

J102 민인애

J141 손혜준

J143 송상화

# 목차

1. 문제 정의
2. 솔루션: CodeJam
3. 핵심 기능 및 사용자 흐름
4. 타겟 유저
5. 기술 스택 및 아키텍처
6. 프로토타입 개발 목적
7. 현재 진행 상황 및 데모
8. 향후 계획
9. Q&A



# 1. 문제 정의

현재 상황의 문제점

기존 협업 방식의 한계

- 화면 공유: 공유자 시선에 갇혀 답답함
- 코드 복사/붙여넣기: 실시간성 부족, 버전 충돌

진입 장벽

- 복잡한 설정과 무거운 IDE
- 로그인/계정 생성 필요



# 1. 문제 정의 (계속)

## 해결하고자 하는 핵심 문제

"간단하고 빠르게 시작할 수 있는 실시간 협업 코딩 환경"

## 왜 만들어야 하는가?

- 로그인 없이 빠르게 코드 공유 및 협업
- 실시간성과 가벼움의 균형
- 페어 프로그래밍, 코딩 면접, 멘토링 시나리오 지원



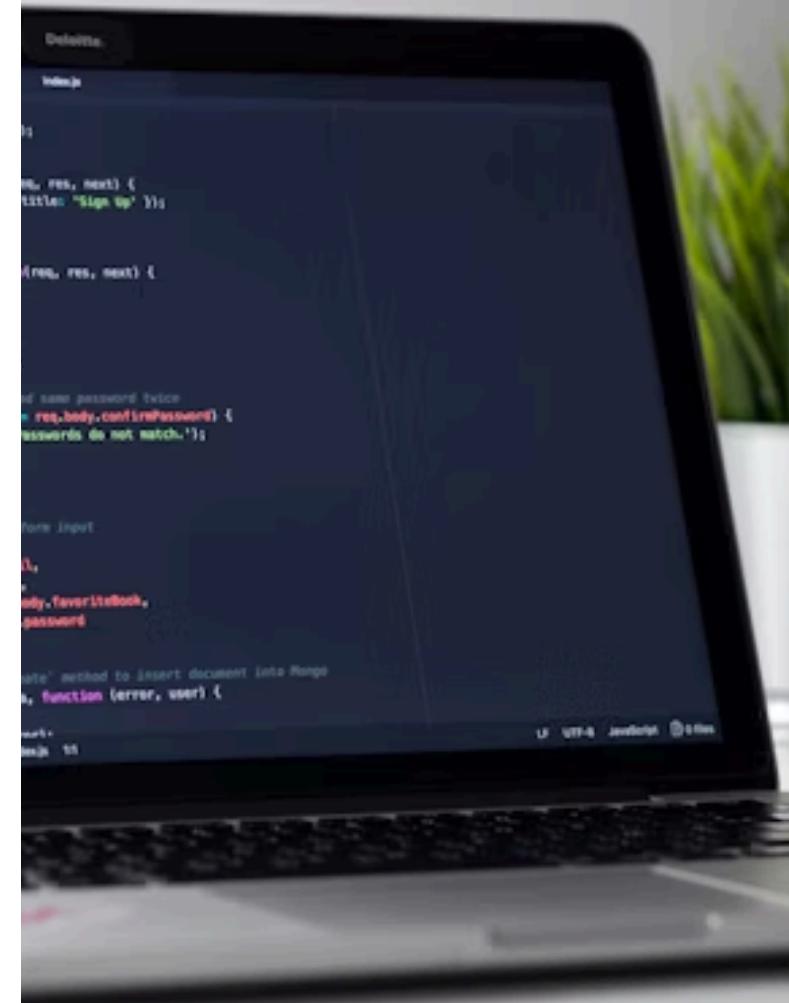
## 2. 솔루션: CodeJam

### 핵심 가치

- **Speed (빠름)**: 클릭 한 번으로 즉시 시작
- **Lightweight (가벼움)**: 필수 기능만 담은 빠른 경험
- **Real-time (실시간)**: 자연 없는 실시간 협업

### 완성도 목표

실제 개발자들이 페어 프로그래밍과 코드 리뷰에  
바로 사용할 수 있는 프로덕션 레벨의 서비스



### 3. 핵심 기능

#### 1. Zero-Config & Login-Free

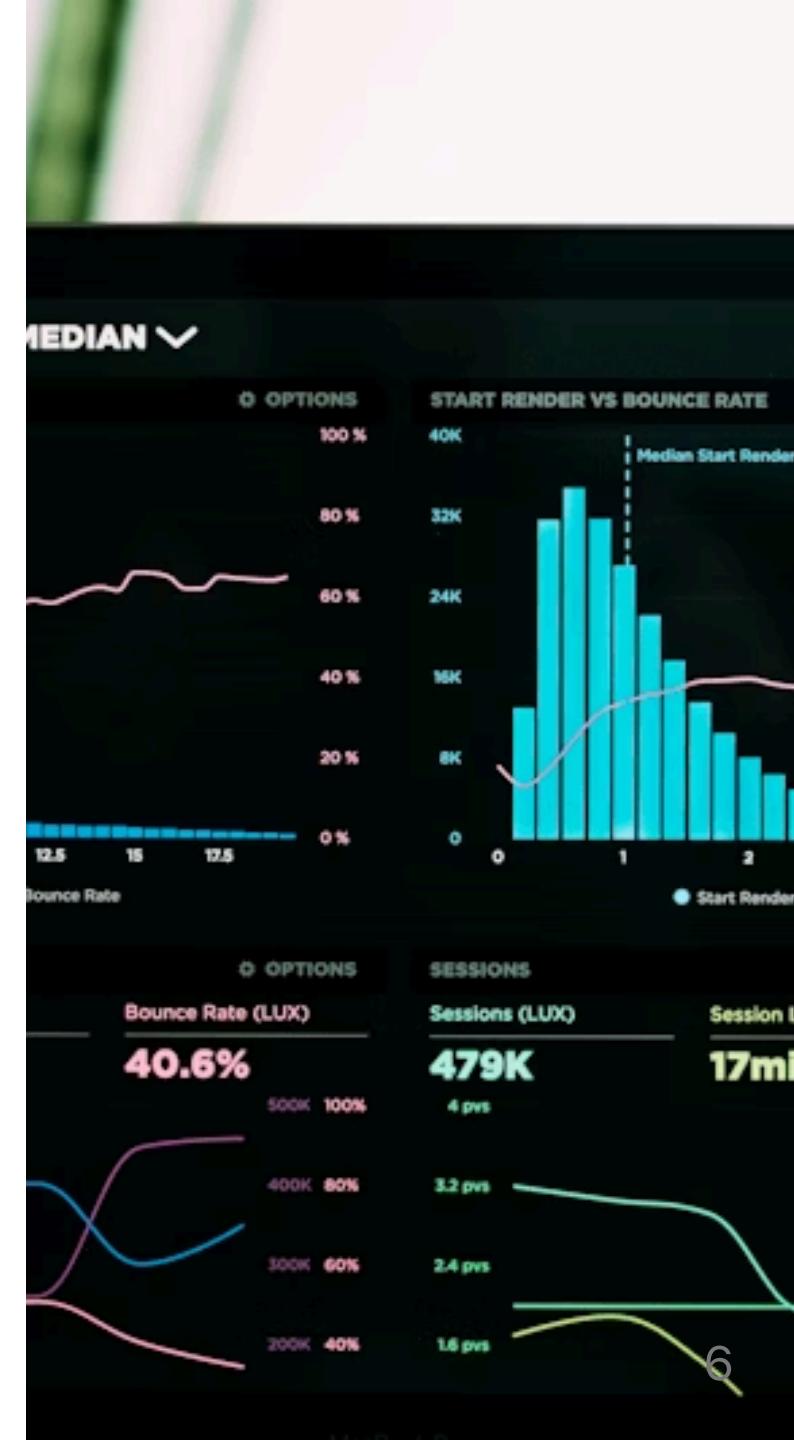
- 별도 설치/로그인 없이 즉시 사용
- URL 공유만으로 협업 시작

#### 2. 실시간 동시 편집

- CRDT(Yjs) 기반 충돌 없는 병합
- 여러 사용자의 동시 편집 지원

#### 3. 커서 추적 (Follow Mode)

- 팀원의 커서 위치 실시간 표시
- 특정 사용자 커서 따라가기



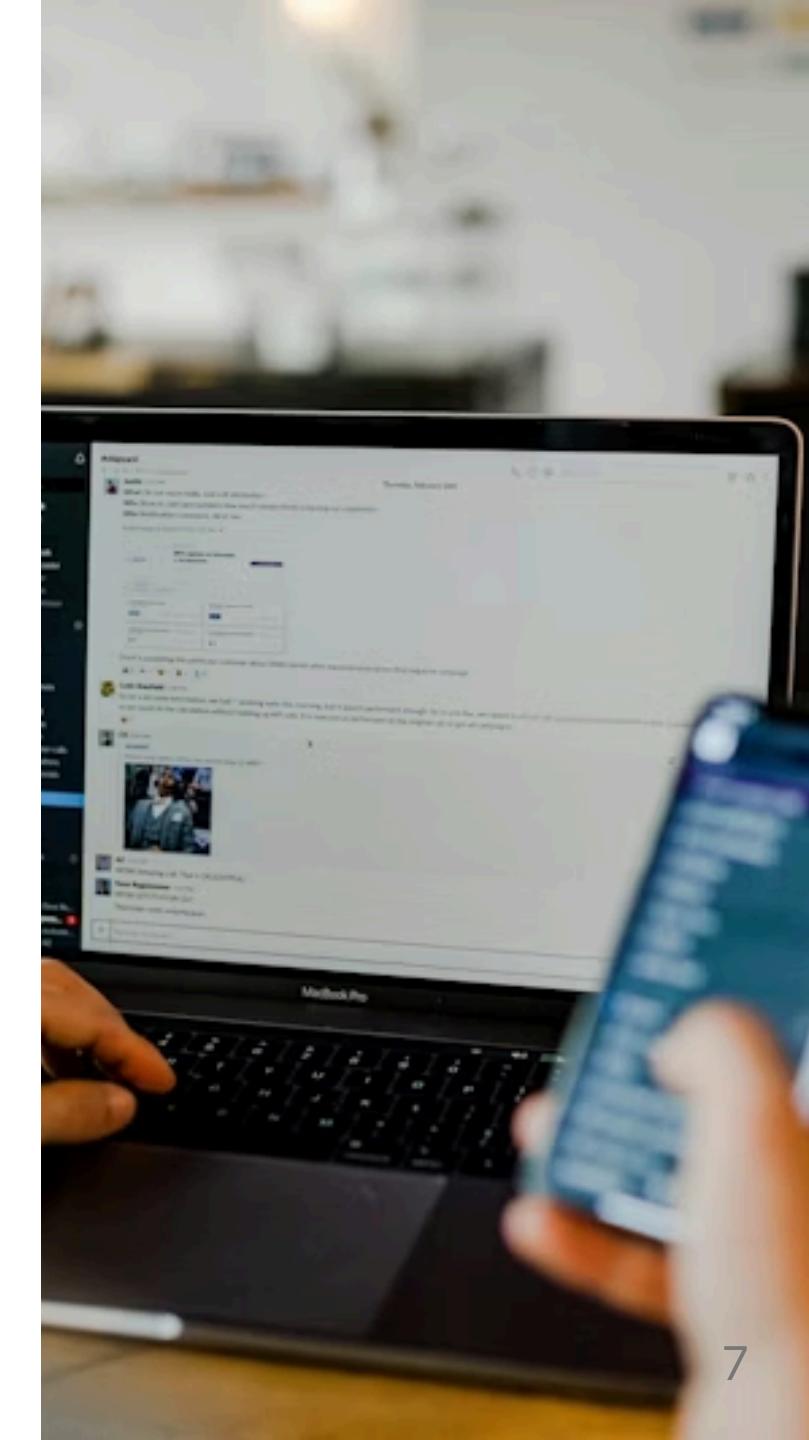
### 3. 핵심 기능 (계속)

#### 권한 관리

- Host: 최초 세션 생성자
- Editor: 선착순 6명 편집 권한
- Viewer: 읽기 전용

#### 보안

- 24시간 자동 만료
- 데이터 자동 삭제로 보안 유지



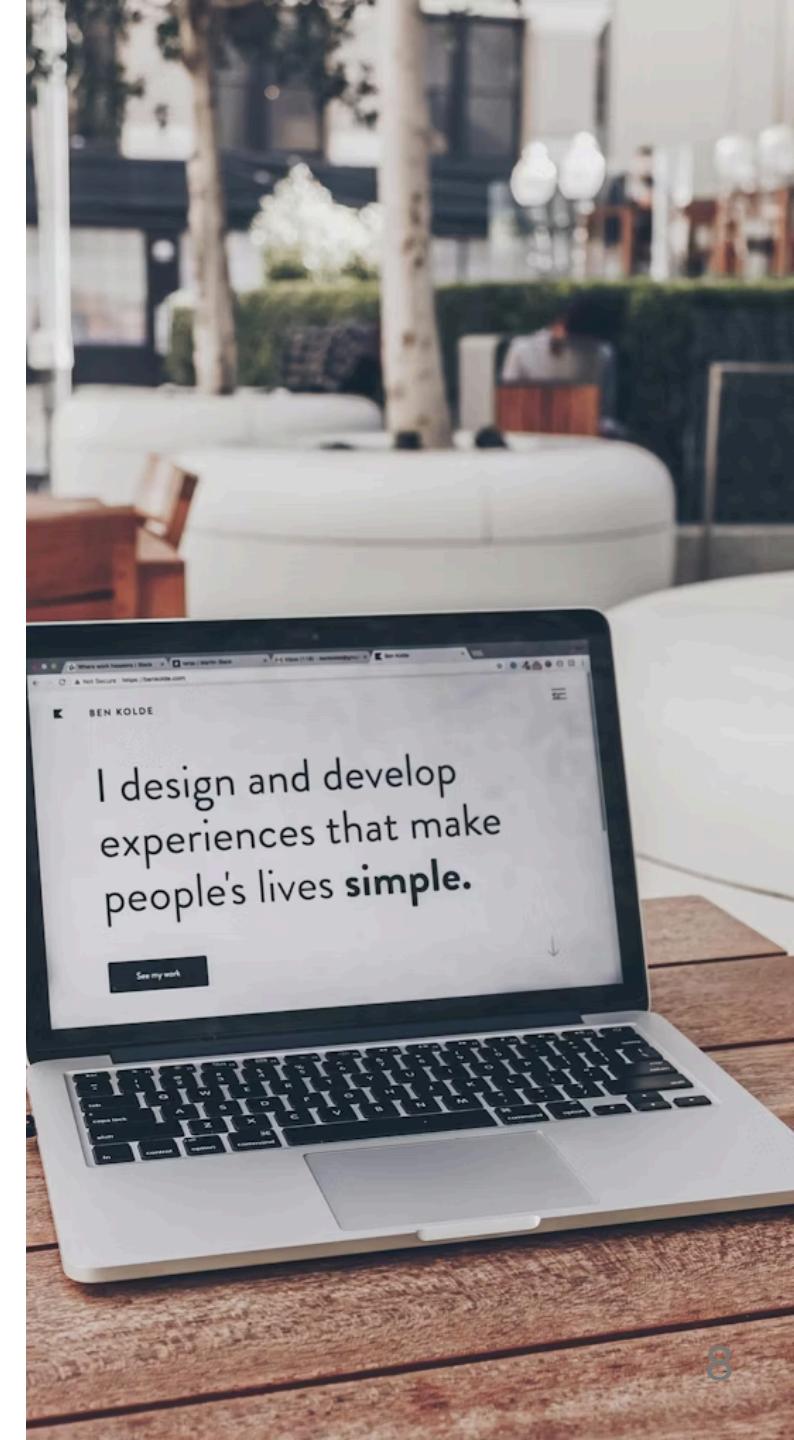
### 3. 사용자 흐름

#### 사용 시나리오

[URL 접속] → [닉네임 자동 할당] → [실시간 편집] → [권한 관리] → [자동 만료]

#### 주요 특징

- 방 생성 단계 없이 즉시 에디터 화면
- 랜덤 닉네임 자동 할당
- 고유 색상으로 사용자 식별
- 실시간 참가자 목록 표시



## 4. 타겟 유저

### 주요 타겟

#### 1. 페어 프로그래밍 팀

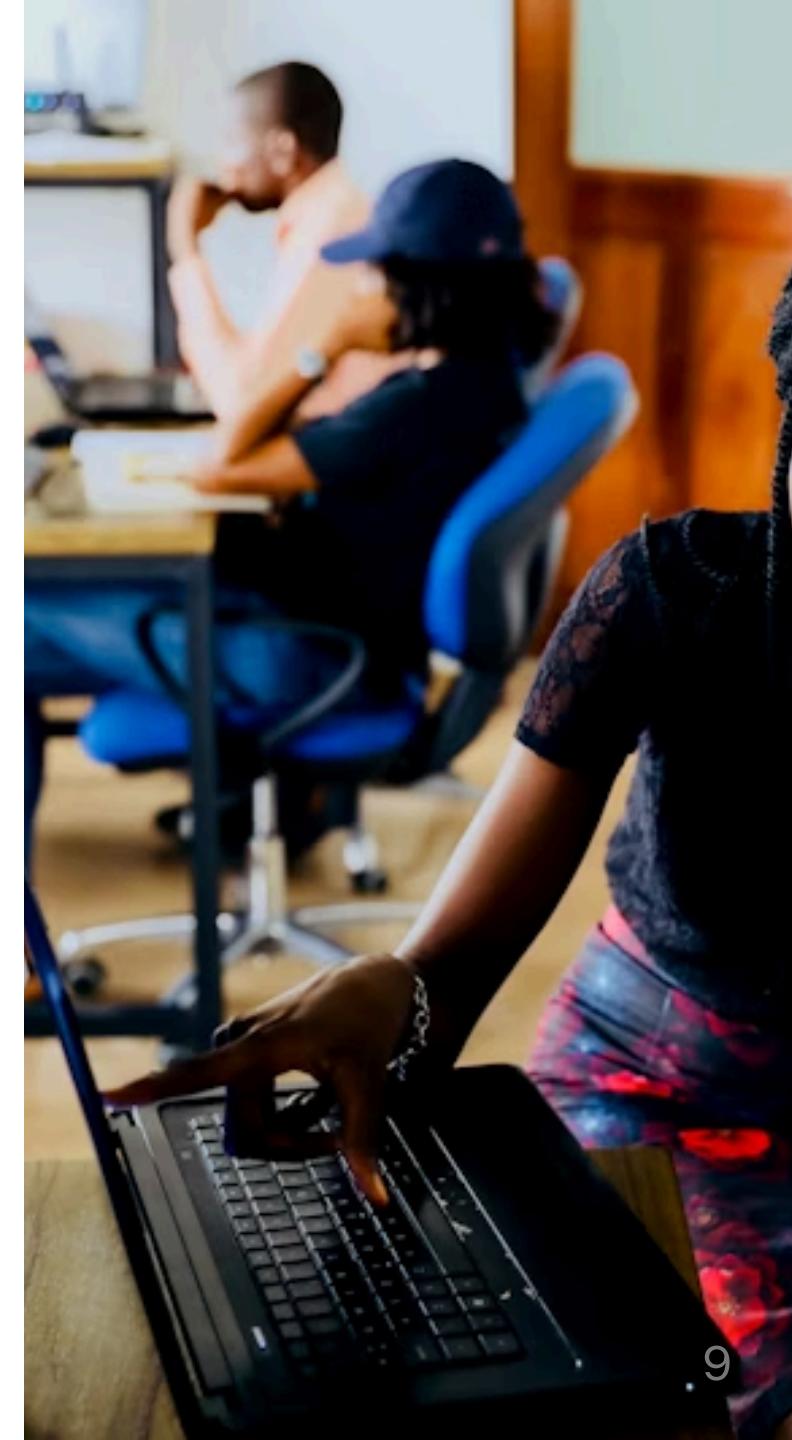
- 간단한 코드 리뷰와 실시간 협업이 필요한 개발팀

#### 2. 코딩 테스트 면접

- 빠른 세션 시작이 필요하고 로그인 없이 즉시 사용해야 하는 면접관과 지원자

#### 3. 스터디/멘토링

- 코드 공유 및 설명 시 커서 추적으로 시선을 공유하고 싶은 멘토와 멘티



## 5. 기술 스택

### Frontend

- React 19 (UI 프레임워크)
- TypeScript (타입 시스템)
- Vite (빌드 도구)
- CodeMirror 6 (코드 에디터)
- Zustand (상태 관리)
- Tailwind CSS v4 (스타일링) & Radix UI (UI 컴포넌트)
- Socket.IO Client (실시간 통신)
- Yjs (CRDT) & y-protocols (동기화 프로토콜)

### Backend

- NestJS 11 (프레임워크)
- TypeScript (타입 시스템)
- Socket.IO (실시간 통신)
- ioredis (Redis 클라이언트)
- Yjs (CRDT)
- y-protocols (동기화 프로토콜)

## 5. 기술 스택 (계속)

실시간 동기화

Yjs (CRDT) + Socket.IO

- Y Protocol 기반 자동 동기화
- lib0로 바이너리 인코딩/디코딩
- Uint8Array 형태로 효율적 전송

Infra & DevOps

Monorepo (Turborepo) + pnpm

- 프론트/백엔드 태입 공유
- NCP 서버 배포 (애플리케이션 + DB)
- Docker 활용

## 5. 아키텍처 특징

### Monorepo 구조

- `apps` : frontend, backend
- `packages` : 공통 타입, DTO, 소켓 이벤트
- `pnpm workspace`로 빌드 순서 자동 처리

### 인메모리 관리

- YJS 문서와 Awareness는 서버 메모리에 저장 (프로토타입)
- 빠른 읽기/쓰기로 실시간 성 보장
- 향후 Redis로 이전하여 확장성 확보 예정

### Low Latency

- 바이너리 데이터 전송 (`Uint8Array`)

## 6. 프로토타입 개발 목적

왜 프로토타입을 만드는가?

검증하고자 하는 핵심 가설

1. 사용자들이 실제로 "빠르고 간단한" 협업 도구를 원하는가?
2. CRDT 기반 실시간 동기화가 실제 사용 환경에서 작동하는가?
3. 로그인 없는 방식이 사용자에게 충분히 편리한가?



## 6. 프로토타입 개발 목적 (계속)

프로토타입을 통해 얻고자 하는 것

기능적 검증

- 실시간 편집의 성능과 안정성
- CRDT 충돌 해결 알고리즘의 실제 성능
- 다수 사용자 동시 접속 시 안정성

사용자 경험 검증

- 첫 진입부터 협업 시작까지의 소요 시간
- 사용자가 느끼는 불편함과 개선점
- 실제 협업 시나리오에서의 효용성



## 6. 프로토타입 개발 목적 (계속)

기술적 검증 사항

성능 및 안정성

- 네트워크 자연 환경에서의 동작
- 대용량 문서에서도 빠른 렌더링
- 효율적인 뷰포트 관리

아키텍처 검증

- 게이트웨이 경량화 전략
- 파일/룸 서비스 분리 구조
- Redis 기반 세션 관리 효율성



## 7. 현재 진행 상황

### 완료된 작업

#### 기획 및 설계

- 기술 스택 확정 및 투표
- 프로젝트 구조 설계 (Monorepo)
- FSD 구조 및 폴더 설계

#### 개발 환경 구축

- Monorepo 환경 구성 완료
- WebSocket 게이트웨이 구현
- useSocket 커스텀 훅 개발



## 7. 현재 진행 상황 (계속)

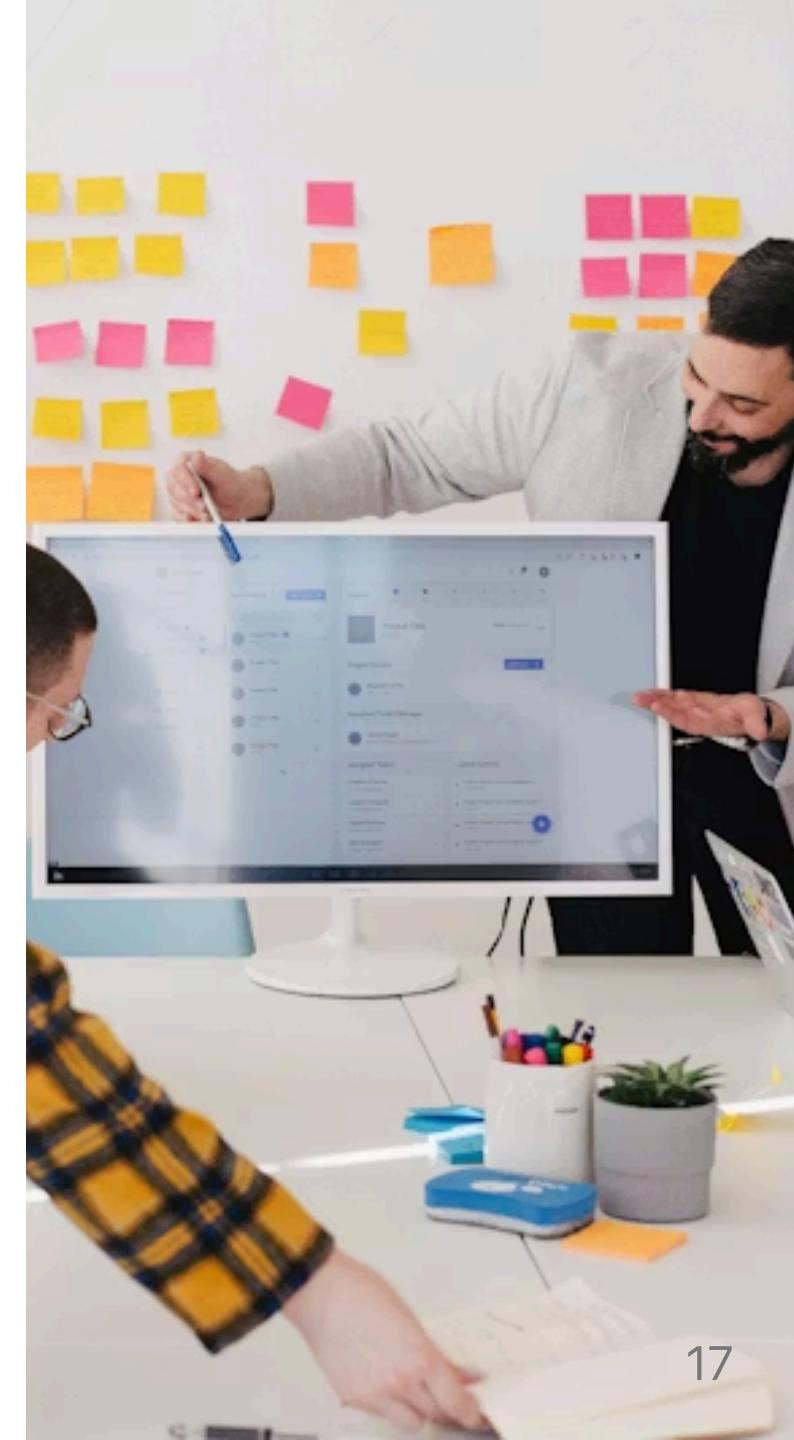
### 완료된 작업 (계속)

#### 핵심 기능 구현

- YJS 실시간 동기화 구현
- 참가자 입장/퇴장 로직 (Redis TTL)
- 권한 및 컬러 부여 그리디 로직
- 룸/파일 서비스 분리
- 참가자 목록, 코드 에디터 UI
- 라우팅 처리 (단일 방)

#### 인프라

- Redis 기반 세션 관리
- 소켓 맵(Socket Map) 구현



## 7. 기술적 도전 과제

### 1. 순환 참조 문제

원인: 게이트웨이와 서비스 간 의존성으로 인한  
순환 참조 발생

대처:

- 게이트웨이에서 소켓을 메서드 인자로 전달
- 룸/파일 서비스 분리로 역할 명확화

결과: 경량화된 게이트웨이, 확장 가능한 서비스 구조

### 2. 세션 관리

원인: 네트워크 불안정으로 인한 일시적 연결 끊김 시 사용자 경험 저하

대처:

- 연결 끊김 시 즉시 삭제하지 않고 Redis에 5분 TTL 설정
- 5분 내 재접속 시 기존 세션 복구 (색상, 권한 유지)
- Redis Key Space Notification으로 TTL 만료 이벤트 감지
- TTL 만료 시 방에서 완전히 제거하고 다른 사용자에게 알림

결과: 일시적 연결 끊김과 실제 퇴장을 구분, 안정적인 사용자 경험

## 7. 기술적 도전 과제 (계속)

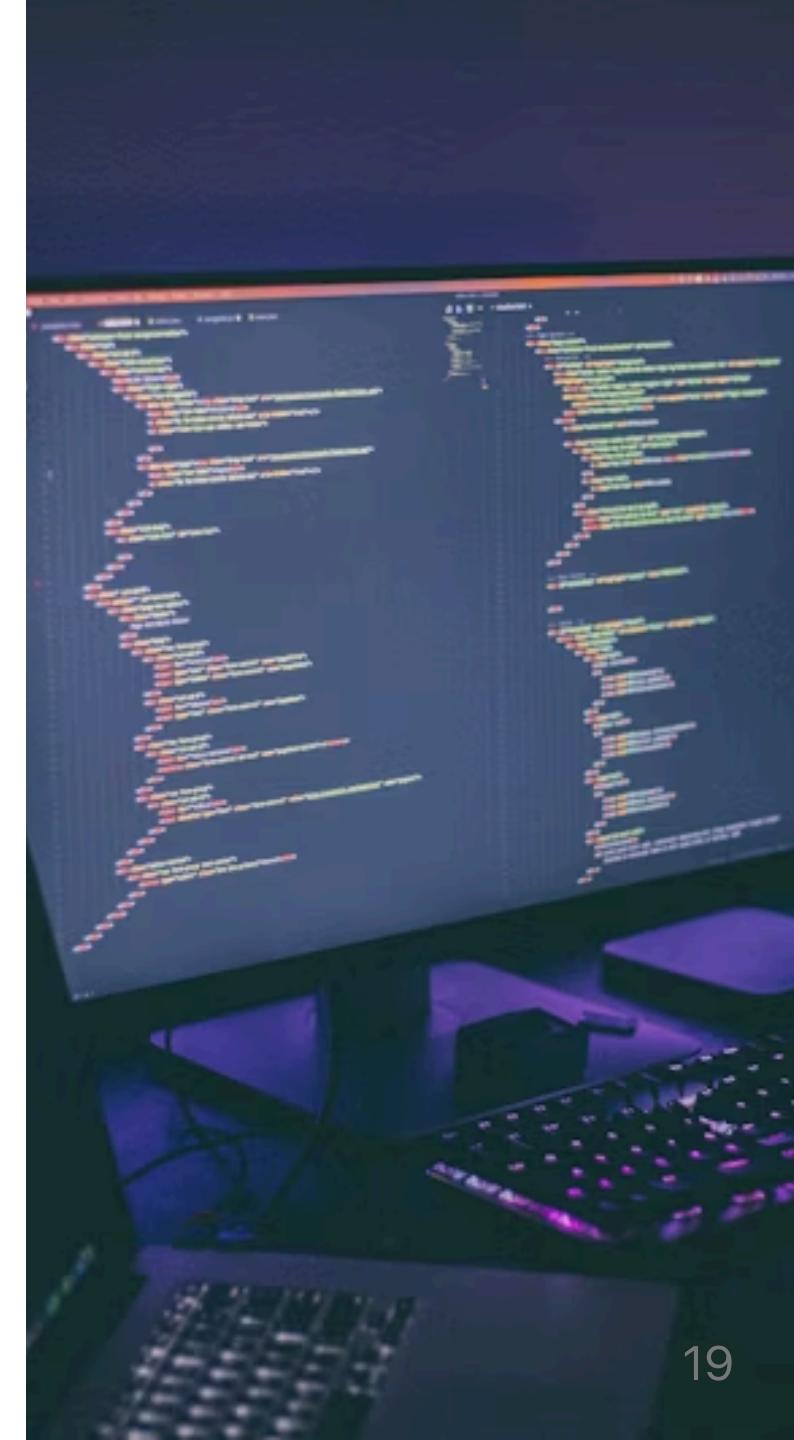
현재 해결 중인 문제

성능 최적화

- useSocket useYdoc 의존성 분리
  - useYdoc 내에서 socket 을 연결하지 않도록

기능 안정화

- 다수 사용자 동시 작업 테스트



## 7. 데모

실시간 동기화 시연

보여드릴 기능

1. URL 접속 및 즉시 시작
2. 랜덤 닉네임 자동 할당
3. 여러 사용자 동시 편집
4. 실시간 커서 위치 표시
5. 참가자 목록 실시간 업데이트
6. 선택 영역(Selection) 동기화



## 8. 향후 계획

### 단기 목표

### 기능 추가 및 리팩토링

- 방 생성 및 삭제
- 권한 관리 시스템 완료 (host의 권한수정)
- 성능 최적화 및 버그 수정
- 코드 구조 리팩토링

### 배포

- http -> https로 변경
- 도메인 연결
- 부스트캠프 내부 테스트



## 8. 향후 계획 (계속)

### 중기 목표 (프로젝트 종료 시)

#### 기능 확장 로드맵

1. 코드 공유: 현재 단일 파일에서 → 여러 파일 동시 편집
2. 프리뷰 기능: 마크다운, HTML 등 실시간 렌더링 미리보기
3. 비주얼 캔버스: 다이어그램/플로우차트 협업 도구
4. 구조화된 에디터: 노션 같은 WYSIWYG 에디터로 확장

#### 추가 계획 중인 기능

- 다중 파일 지원
- 히스토리 기능 (스냅샷)
- 코드 실행 (서버리스)

#### 인프라 개선

- DB/Redis 서버 분리

## 8. 향후 계획 (계속)

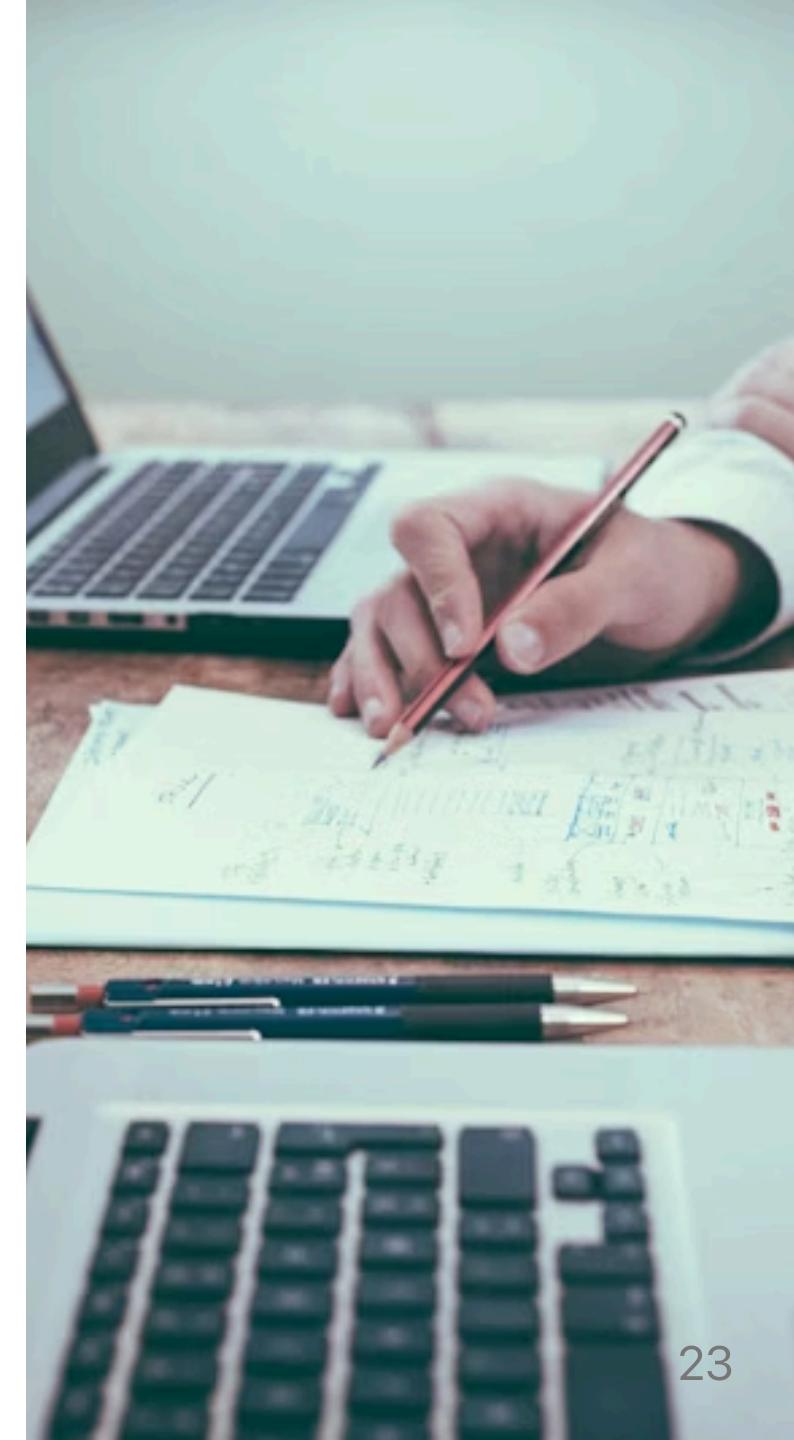
성공 지표

사용성

- 최소 50회 이상의 실제 사용 세션
- 긍정적인 사용자 피드백 (80% 이상)

기술적 완성도

- 6명 동시 편집 시 지연 100ms 이하
- 24시간 안정적 세션 유지
- 네트워크 불안정 환경에서도 동작



# Q&A

## 받고 싶은 피드백

1. 프로토타입 사용 경험은 어땠나요?
2. 실제 페어 프로그래밍에 사용할 의향이 있나요?
3. 불편하거나 개선이 필요한 부분은?
4. 추가되면 좋을 기능은?

질문 주시면 답변드리겠습니다.

감사합니다! 🎉

