개 발 계 획 서

프로젝트 분야/주제	담당교수와의 컨설팅을 통해 개인별 자유주제 선정 및 개발	총 투입인원	1 명
프로젝트 명	국문	CodeSync: 소켓통신 기반 실시간 코드 협업 시스템	
	영문	CodeSync: Real-Time Collaborative Coding over Sockets	
수행기간	2025. 05. 12 ~ 2025. 06. 18		

참여자	소속	참여자성명	학년	학번	개인 Git 주소
	컴퓨터공학부	이승준	3	2021243019	https://github.com/SaMeL-dev

과제목표

가) 필요성

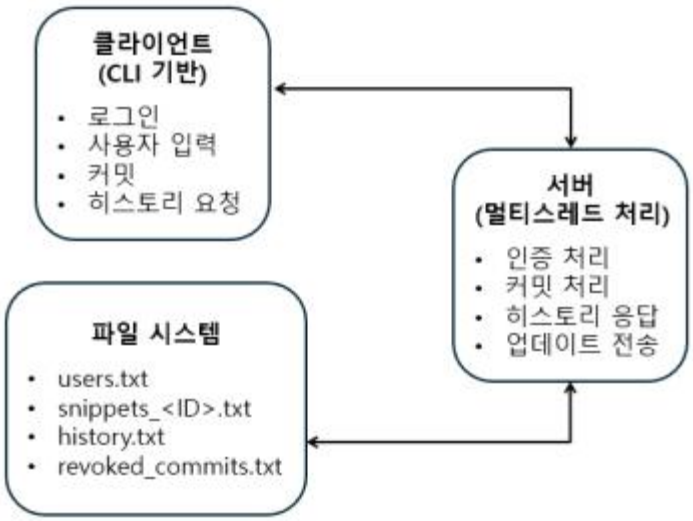
GitHub와 같은 협업 플랫폼의 구조를 직접 모사하여 구현함으로써, 소켓 통신 기반의 네트워크 처리, 파일 입출력 방식에 의한 정보 저장, 연결리스트를 활용한 자료 구조 처리 등 시스템 프로그래밍의 핵심 기술을 실제로 동작하는 형태로 구성해보는 경험이 가능함.
이를 통해 다양한 기술 요소의 작동 원리와 구현 흐름을 직접 체득하고, 다중 사용자 협업 구조의 작동 메커니즘을 심화 이해할 수 있는 기반을 마련함.
또한 콘솔 환경에서 협업 구조를 직접 설계함으로써 명령어 기반 시스템의 특성과 동작 원리를 실제로 체득할 수 있음.

나) 개발 목표

- TCP 기반 소켓 통신을 이용한 클라이언트-서버 구조 구현
- 사용자 로그인 및 인증 처리 기능 구성
- 사용자별 커밋 내용 저장 및 프로젝트 단위 디렉토리 분리 처리
- 커밋 요청 시 제목, 메시지, 코드 본문을 포함한 파일 저장 기능 구현
- 파일 입출력 기반의 사용자 정보, 커밋 히스토리 관리 기능 구성
- 클라이언트 측 커밋 히스토리 출력 시 연결리스트를 활용한 구조화 처리
- 커밋 무효화 요청 처리 및 관련 로그 관리 기능 구현
- 커밋 이후 다른 클라이언트에 동기화 요청을 전송하는 구조 설계

다) 개발결과 창출

- 여러 명의 사용자가 동시에 서버에 접속하여 커밋을 공유하고 관리할 수 있는 협업 구조 구현
- 콘솔 환경에서도 코드 기록, 커밋, 히스토리 출력 기능을 포함한 버전관리 흐름 구성
- 사용자별 코드 분리 및 실시간 히스토리 제공을 통한 협업 기능성 확보
- 향후 GUI 확장 또는 데이터베이스 연계를 고려할 수 있는 기본 구조 기반 마련



[그림 1] CodeSync 시스템 구조

개발내용 요약

가) 서버 기능

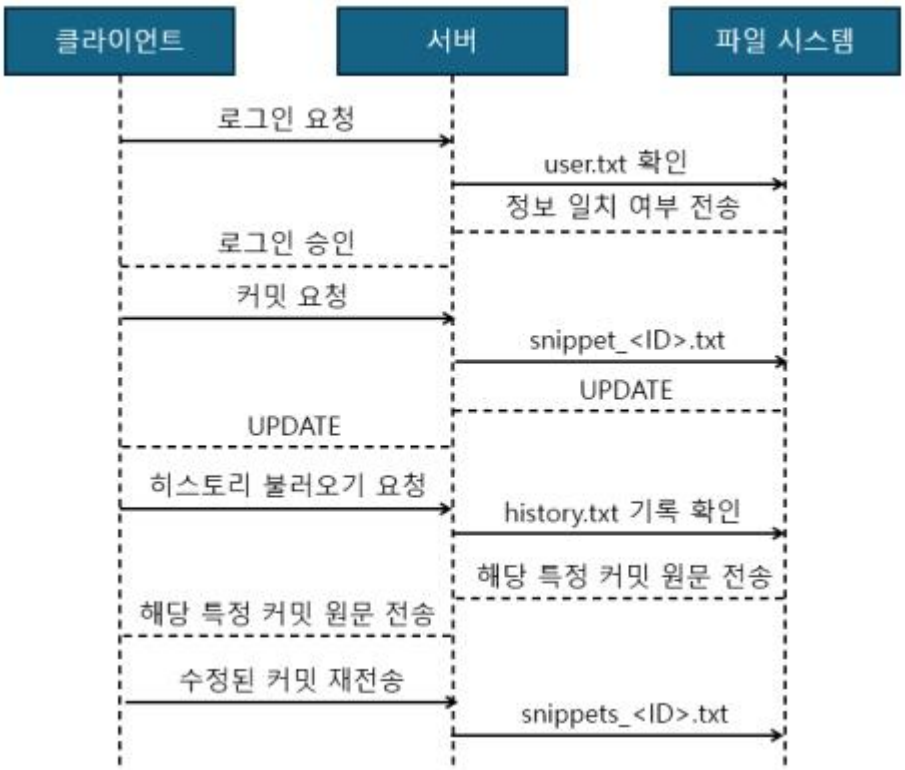
- 클라이언트의 로그인 요청을 수신하고 사용자 정보를 검증하는 기능을 수행
- 클라이언트로부터 받은 커밋 데이터를 파일 형태로 저장하고, 히스토리 파일에 로그를 추가하는 역할을 수행
- 커밋 무효화 요청이 들어올 경우 해당 커밋을 무효화 목록에 기록하는 기능을 포함
- 프로젝트 생성 및 선택 요청에 따라 사용자 전용 디렉토리를 생성하거나 해당 경로로 분기 처리
- 커밋이 등록되었을 경우, 서버에 접속 중인 다른 클라이언트에게 동기화 요청 메시지를 전송

나) 클라이언트 기능

- 사용자 ID와 비밀번호를 입력받아 로그인 요청을 서버로 전송
- 로그인 성공 시 프로젝트 생성 또는 선택 요청을 수행
- 코드 커밋 시 제목, 메시지, 코드 본문을 사용자 입력으로 구성하여 서버로 전송
- 커밋 히스토리 요청 시 연결리스트에 저장하여 순차 출력하는 기능을 수행
- 다수 사용자의 동시 요청에 대응 가능한 구조로 설계
- UPDATE 메시지를 수신한 경우, 서버로부터 최신 히스토리를 재요청하여 화면을 갱신
- 커밋 무효화, 종료 요청 등 다양한 명령어를 통해 기능 분기를 처리

다) 공통 처리 구조

- 사용자 정보는 users.txt, 커밋 히스토리는 history.txt, 각 스니펫은 snippets_<ID>.txt 저장
- 프로젝트 단위 디렉토리를 기반으로 파일 입출력을 분기 처리
- 모든 커밋 및 요청 메시지는 특정 포맷에 따라 문자열로 구성되어 소켓 통신을 통해 송수신
- 명령 구분자는 ##END##₩n 형식을 활용하여 각 기능의 처리 단위를 구분



[그림 2] 커밋 히스토리 복원 및 재커밋 시퀀스 다이어그램

상세 개발 사항

가) 클라이언트 측 기능

- 로그인 기능은 사용자 입력 정보를 서버에 전송
- 로그인 응답을 수신하여 성공 여부를 판단
- 프로젝트 기능은 프로젝트 생성 또는 선택 요청을 서버에 전달
- 커밋 기능은 제목, 메시지, 코드 본문을 입력받아 서버에 전송
- 히스토리 기능은 서버로부터 수신한 로그를 연결리스트에 저장
- 저장된 연결리스트를 순차 출력하여 사용자에게 보여줌
- 무효화 기능은 제목을 입력받아 서버에 무효화 요청을 전송
- 동기화 기능은 UPDATE 메시지 수신 시 히스토리 요청을 재전송
- 명령어 기반 분기로 각 기능을 구분하여 수행

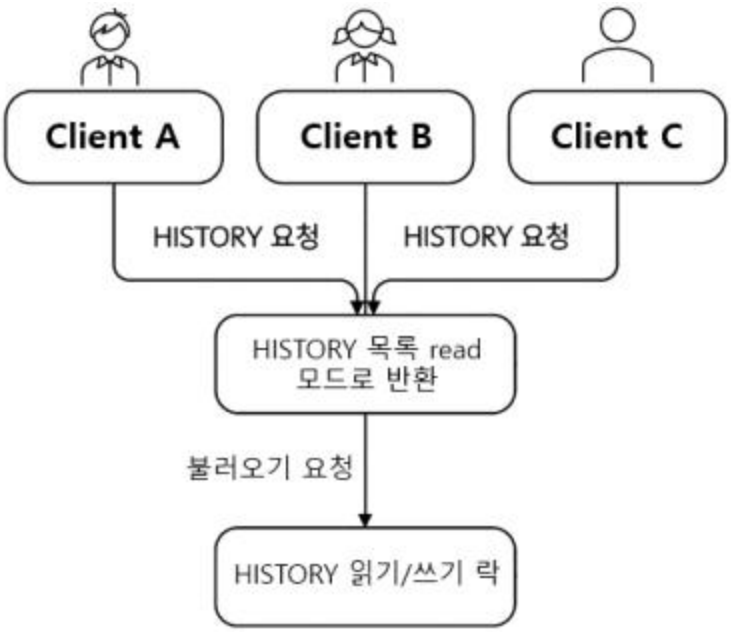
나) 서버 측 기능

- 로그인 기능은 users.txt 파일을 참조하여 사용자 정보를 검증
- 프로젝트 기능은 요청된 프로젝트명을 기준으로 디렉토리를 생성하거나 분기
- 커밋 기능은 제목, 메시지, 코드 본문을 파일로 저장하고 로그를 기록
- 무효화 기능은 제목을 revoked_commits.txt에 저장하여 관리
- 히스토리 기능은 저장된 로그를 조건에 맞게 필터링하여 전송
- 동기화 기능은 커밋 발생 시 접속 클라이언트에게 UPDATE 메시지를 전송
- 각 클라이언트 연결은 스레드 단위로 처리하여 요청을 병렬 수행

다) 공통 처리 및 기술 요소

- 소켓 통신은 TCP 기반으로 클라이언트와 서버 간의 데이터 전송을 수행
- 명령어 구분자는 ##END##₩n으로 기능 단위를 구분
- 사용자별 커밋 파일은 snippets_<ID>.txt 형식으로 저장
- 커밋 로그는 history.txt에 시간순으로 누적 저장

- 무효화된 제목은 히스토리 출력 시 제외되도록 처리
- 히스토리 출력은 연결리스트를 사용하여 순서대로 구성
- 다수 사용자의 히스토리 요청 동시 열람을 위해 읽기-쓰기 락(pthread_rwlock_t) 적용
- 커밋 저장 시 단독 쓰기 락으로 동기화하여 데이터 정합성 확보
- 모든 파일 입출력은 프로젝트 디렉토리 기준으로 처리



[그림 3] 다중 사용자 히스토리 열람 시 락 적용 구조

과제수행방법

구분	개발 환경
OS	Windows 11 64bit
개발 언어	C언어
통신 방식	TCP 소켓 통신
개발 툴	Visual Studio Code(VSCode)
빌드 환경	MinGW(GCC)
입력 방식	콘솔 기반 사용자 명령어 입력 방식
실행 방식	터미널 기반 클라이언트 및 서버 실행
테스트 방식	최소 2기기, 최대 5기기 동시 접속 테스트
자료 구조	연결리스트 활용
데이터 저장	파일 입출력(txt 기반) 방식

[표 1] 개발 환경 요약 표

주요내용	추 진 일 정 (주)				
	1	2	3	4	5
시스템에 대한 요구사항분석서 작성					
요구사항분석서 기반 개발계획서 작성					
서버-클라이언트 소켓 통신 구성					
사용자 로그인 및 인증 처리					
프로젝트 생성 및 분기 처리 기능 구현					
코드 커밋 및 히스토리 파일 저장 기능 구현					
클라이언트 연결리스트 기반 히스토리 출력 구성					
무효화 및 실시간 동기화 기능 구현					
전체 기능 통합 및 예외 처리 보완					

[표 2] 개발 주요 내용과 추진일정

결과활용계획

가) 기술 내재화 및 학습 효과

- 소켓 기반 클라이언트-서버 구조 설계를 통해 C언어 기반 네트워크 프로그래밍 이해도 확보
- 구조체 기반 사용자 인증, 파일 입출력 기반 커밋 로그 저장 등 시스템 프로그래밍 기술 내재화
- 명령어 기반 협업 흐름을 구현하면서 입력 처리, 연결리스트 구성, 동기화 구조에 대한 실습 효과 확보
- 실시간 동기화, 예외 처리, 다중 사용자 구조 설계를 통해 복합 기술 통합 경험 확보

나) 향후 프로젝트 확장 및 재사용 가능성

- 본 시스템은 코드 저장 외에도 메모 관리, 제출 기록, 평가 내역 관리 등 다양한 형태로 확장 가능
- 커맨드 기반 CLI 구조와 커밋 히스토리 로그 처리 방식을 통해 팀별 협업 플랫폼으로 재사용 가능
- 구조체 분리 및 파일 기반 저장 방식은 다른 교육용 프로젝트에서 모듈 단위로 재활용 가능

다) 실무 환경 대응력 강화

- 사용자 구분, 커밋 기록, 요청 동기화 등 구조는 실제 협업 시스템의 백엔드 처리 흐름과 유사
- 명령 처리, 예외 방지, 사용자 입력 안정성 등의 요소를 구현하면서 실무 수준의 견고성 확보
- 스레드 기반 요청 처리와 실시간 반영 구조를 통해 다중 사용자 환경에서의 안정성 경험 가능
- 읽기-쓰기 락을 통한 멀티유저 히스토리 열람 동기화 설계로 고급 동시성 제어 개념 실습

라) 팀 프로젝트 기반 협업 역량 확보

- 기능별 테스트 및 통합 시나리오 구성 과정을 통해 협업 시 문서화, 설계 분담, 데이터 규칙 통일 경험 가능
- 명령 체계 설계와 데이터 연동 과정을 경험하면서 팀 단위 기능 통합 과정에 대한 실전 감각 확보
- 협업형 CLI 플랫폼 구현 경험을 기반으로 추후 API 기반 협업 시스템 개발에 연계 가능