캡스톤디자인 II 중간보고서(표지)

프로젝트명 : 백엔드 서비스 프로바이더: 하향식 기반 방법론을 적용한 경량형 컨테이너 서비스 개발

캡스톤 디자인II, 중간보고서

Version 1.0

개발 팀원 명(팀리더):허유정
김창인

대표 연락처:010-2235-2464
e-mail: 20211939@edu.hanbat.ac.kr

캡스톤 디자인 II 중간보고서 내용

1. 요구사항 정의서에 명세된 기능에 대하여 현재까지 진척된 결과 및 그 내용을 기술하시오.

가. 소스코드일부

(※https://github.com/HBNU-SWUNIV/come-capstone24-akdong_developer.git)

1) 클라이언트의 컨테이너 실행 요청을 받아, 자원 격리(cgroup, 네임스페이스)를 적용하여 컨테이너를 실행하는 기능을 담당
2) 컨테이너가 성공적으로 실행되면 status: started를 반환
3) setupCgroups 함수를 통해 컨테이너에 메모리 및 CPU 자원 제한을 설정
4) memory.limit_in_bytes와 cpu.cfs_quota_us를 사용해 메모리와 CPU 사용량을 제한

나. Ver1.0 실행파일 및 환경설치 문서

가) 실행 환경

- Ubuntu 20.04 LTS 기반에서 개발된 컨테이너 런타임 엔진, 실행을 위해서는 Go 언어 개발 환경과 Linux cgroup, 네임스페이스 기능이 활성화된 커널이 필요

나) 설치 방법

- 주어진 깃허브 링크에서 container_daemon 실행파일을 다운로드한 후, 환경변수를 설정하고 파일을 실행

2. 프로젝트 수행을 위해 적용된 추진전략, 수행 방법의 결과를 작성하고, 만일 적용과정에서 문제점이 도출되었다면 그 문제를 분석하고 해결방안을 기술하시오.

가. 추진전략 및 수행 방법

1) 협업 도구의 활용

가) Github을 활용한 프로젝트 동시 개발

(1) 소스 코드와 변경 사항의 효과적인 추적 및 관리 가능

(2) 풀 리퀘스트 생성으로 변경 사항 검토 및 통합

(3) 개발자들 간 소스 코드 공유 및 병합 간편화

(4) 실시간 프로젝트 진행 상황 공유

나) Notion을 활용한 문서 협업

(1) 계획 공유를 통한 일정 관리

(2) 문서 공유의 불편함 최소화

(3) 작업의 투명성을 높여 팀원 간의 의사소통 강화

다) Microsoft Teams를 활용한 의사소통

(1) Github과의 연결을 통한 알림 활용

(2) 멘토와의 문서공유 및 의사소통 강화

나. 문제 해결을 위한 적용 방법

1) 주기적인 스프린트 계획 및 리뷰

- 예기치 않은 기술적 장애로 인해 특정 스프린트의 완료가 지연되는 문제가 발생하였다. 이에 팀 전체의 방향성과 목표를 명확히 설정하고, 단계별 성취를 모니터링하여 일정 지연을 최소화 하였다.

2) 소프트웨어 아키텍처 설계

- 소프트웨어 아키텍처 설계를 사전에 명확히 정의하여 팀원의 시스템 이해도를 높이고자 하였다. 하지만 시스템 전반에 대해 일관성 있는 개발을 하지 못하여 문제가 발생하였다. 이에 주기적인 설계 리뷰를 통해 팀원 간 이해도 격차를 줄이고, 문서화된 가이드를 제공하였다.

[별첨1]

프로젝트명 : 백엔드 서비스 프로바이더: 하향식 기반 방법론을 적용한 경량형 컨테이너 서비스 개발

소프트웨어 요구사항 정의서

Version 1.0

개발 팀원 명(팀리더):허유정
김창인

대표 연락처:010-2235-2464
e-mail: 20211939@edu.hanbat.ac.kr

목차

1. 시스템 개요

가. 사업의 목표

1) 컨테이너 오케스트레이션 배경

컨테이너 기술은 애플리케이션을 빠르게 배포하고 확장할 수 있어 현대 소프트웨어 개발의 핵심 요소로 자리 잡고 있다. 서버 컨테이너 수의 증가함에 따라 대규모로 운영되는 컨테이너 관리의 중요성이 증대되었으며 현대에서는 이를 효율적으로 운영하고자 컨테이너 오케스트레이션 기술을 통해 관리한다. 이는 컨테이너의 배포, 관리, 확장을 자동화하여 복잡한 컨테이너 환경을 효율적으로 운영할 수 있다.

2) 컨테이너 오케스트레이션 도구의 한계

쿠버네티스는 컨테이너 오케스트레이션의 대표적인 도구로 효율적인 운영과 배포를 지원한다. 하지만 소규모 프로젝트나 단순한 애플리케이션에는 과도한 복잡성을 나타내며 특정 도구와 플랫폼에 대한 의존성이 커지는 문제가 있다. 이는 유연성을 제한하고, 새로운 요구사항에 대한 대응을 어렵게 만든다.

3) 컨테이너 오케스트레이션 도구의 제작 목표

본 사업은 컨테이너화된 애플리케이션의 배포, 관리 및 확장을 자동화하는 컨테이너 오케스트레이션 도구 제작을 목표로 한다. 기존 컨테이너 오케스트레이션 도구가 가진 복잡성과 의존성 문제를 해결하고 유연한 접근이 가능한 도구를 제공하고자 한다. 또한 특정 제품에 종속되지 않는 플랫폼 제작으로 소규모의 프로젝트에서 효율적인 운영이 가능하며 IT 비용을 절감할 수 있는 기회 제공이 가능하다.

나. 추진 범위

1) 사용 기술

컨테이너 오케스트레이션 도구 개발을 위해 Linux의 Namespace, Cgroup의 사용으로 시스템 자원을 효율적으로 격리하고 관리하고자 한다. Namepace는 각 프로세스가 독립된 환경에서 실행될 수 있도록 하며 Cgroup은 cpu, 메모리등 자원 사용량을 제어하여 컨테이너 간 안정적인 시스템 운영을 가능하게 한다. 본 프로세스 실행에 앞서 성능과 효율성을 높이기 위해 Go언어를 사용한다. Go는 뛰어난 동시성 처리 기능과 가벼운 메모리 사용으로 컨테이너화된 애플리케이션 환경에 최적의 운영을 가능하게 한다.

2) 구현 기능

컨테이너 오케스트레이션 제작에 있어 대표적인 기능은 컨테이너 런타임 관리, 스케일링, 네트워킹, 사용자 인터페이스 관리로 볼 수 있다. 사용자는 컨테이너 라이프사이클을 간편하게 제어하고 리소스를 동적으로 조정 가능하다. 또한 컨테이너 간 외부 네트워크와의 연결을 용이하게 하기 위해 통신 규칙에 맞춘 복잡한 네트워크 설정 기능을 구현하고자 한다.

3) 시스템 구성도

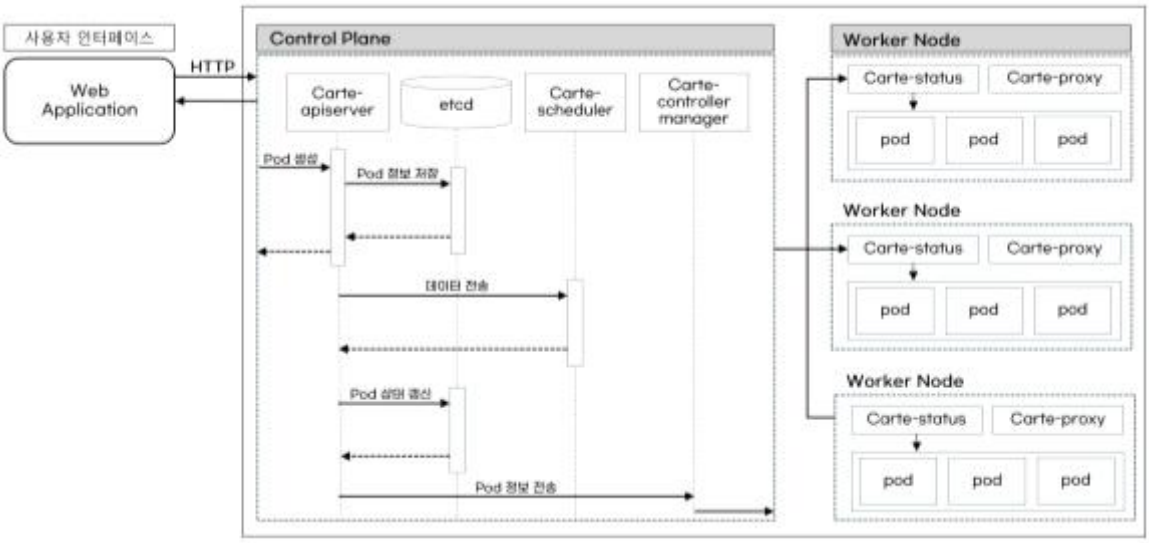

그림 1 시스템 구성도

그림 1은 전체적인 시스템 구성도를 나타낸 것이며, 컨테이너 런타임 관리, 스케일링, 네트워킹의 내용을 포함하고 있다. 각 노드는 Control Plane과 통신하여 작업을 수행하며 컨테이너 배포 및 설정 관리가 가능하다. Control Plane은 작업 지시 및 상태 모니터링을 수행하며, Worker Node는 작업을 실행하고 결과를 보고한다. 또한 각 Pod에 고유한 주소를 할당하여 통신 프로토콜을 규정한다.

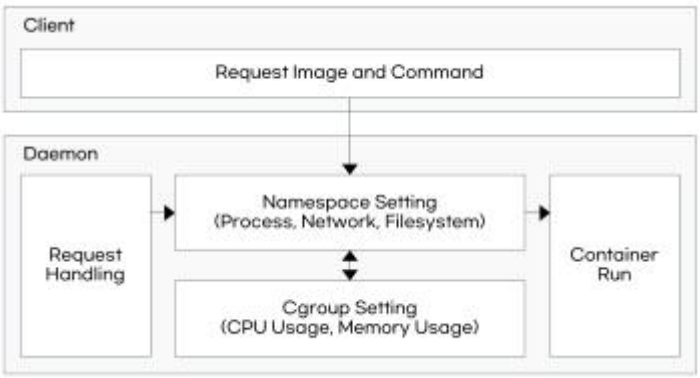

그림 2 런타임 엔진 설계도

그림 2는 컨테이너 런타임 엔진의 구성요소를 나타낸 것이다. Namespace 설정은 프로세스 격리를 수행하며 Cgroup 설정은 리소스 제한을 수행한다. 이는 각 컨테이너가 독립적으로 동작하도록 하여 자원 경쟁을 최소화하며 컨테이너 오케스트레이션 도구 제작을 위한 기반이 된다.

지정된 명령어를 통해 컨테이너를 실행할 시, 사용자가 가지고 있는 이미지의 컨테이너화를 통해 컨테이너가 독립적으로 동작하고 필요한 리소스를 효율적으로 사용할 수 있도록 한다.

2. 시스템 장비 구성요구사항

요구사항 고유번호		ECR-001		
요구사항 명칭		시스템 장비		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	경량화된 컨테이너 런타임 엔진 개발 및 테스트를 위한 장비 구성 요구사항을 정의		
	세부내용	**① 장비 품목** - 경량화된 컨테이너 런타임 엔진 개발을 위한 HW 및 소프트웨어 **② 장비 수량** - 오케스트레이션 실험을 위해 최소 3대의 장비가 필요 **③ 장비 기능** - 제한된 리소스 환경에서의 컨테이너 성능 측정 및 다중 노드 테스트 **④ 사용하는 장비 성능 및 특징** - CPU : 12th Gen Intel(R) Core(TM) i9-12900K - 메모리 : 64GB RAM - OS : Ubuntu 20.04 LTS - Kernel : 5.15.0-113-generic		

요구사항 고유번호		ECR-002		
요구사항 명칭		자원 및 장애 처리 요구사항		
요구사항 분류		시스템 장비구성 요구사항	응락수준	선택
요구사항 상세설명	정의	경량 컨테이너 런타임 엔진 성능 테스트 및 관리 장비		
	세부내용	**① 자원 제약사항** - 경량 컨테이너 런타임 엔진은 제한된 리소스 환경(예: 라즈베리 파이, 소형 컴퓨터)에서도 동작할 수 있어야 하고, 설치 및 관리 시, 소형 컴퓨터 환경에 익숙한 시스템 관리자가 필요하며, 컨테이너 실행 중 리소스 사용량을 모니터링할 수 있는 도구와 스크립트를 준비해야 함 **② 장애 처리** - 컨테이너가 과도한 리소스를 사용하는 상황이 발생하면, 즉시 프로세스를 종료하고 문제를 분석하는 단계로 넘어가야 함		

3. 기능 요구사항

요구사항 고유번호		SFR-FA-001		
요구사항 명칭		컨테이너 관리 기능		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	기본적인 컨테이너 관리 기능을 제공하며, 제한된 리소스 환경에서도 원활한 컨테이너 실행을 보장		
	세부내용	**① 컨테이너 생성 및 삭제** - 컨테이너를 효율적으로 생성하고, 사용하지 않는 컨테이너를 신속히 삭제할 수 있는 기능을 제공 **② 컨테이너 시작 및 종료** - 컨테이너 실행 및 종료 명령을 지원하며, 제한된 환경에서도 빠르게 시작될 수 있어야 함 **③ 명령어 호환성** - Docker 명령어와의 호환성을 제공하여 사용자는 익숙한 명령어를 통해 컨테이너를 관리할 수 있어야 함(예: carte run, carte stop, carte rm)		

요구사항 고유번호		SFR-FA-002		
요구사항 명칭		리소스 관리 기능		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	각 컨테이너의 CPU, 메모리, 네트워크 사용량을 제한하고 관리하는 기능을 제공하여 제한된 리소스 환경에서도 안정적인 시스템 운영을 보장		
	세부내용	**① 리소스 할당 및 제한** - 각 컨테이너에 대해 CPU 및 메모리의 할당을 지정할 수 있으며, 설정된 리소스 범위 내에서만 동작하도록 제어 **② 리소스 모니터링** - 컨테이너 실행 중 실시간으로 CPU 및 메모리 사용량을 모니터링할 수 있는 기능을 제공 **③ 리소스 경고** - 리소스 사용량이 설정된 한계를 초과할 경우 경고를 발생시키고, 과다 사용 시 자동으로 프로세스를 제어할 수 있는 기능을 제공		

요구사항 고유번호		SFR-FA-003		
요구사항 명칭		오케스트레이션 지원 기능		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	여러 대의 소형 컴퓨터나 임베디드 시스템에서 다수의 컨테이너를 중앙에서 관리하고 제어할 수 있는 기능을 제공		
	세부내용	**① 다중 노드 관리** - 다수의 소형 컴퓨터 환경에서 각 노드의 컨테이너를 중앙에서 관리하고 제어할 수 있는 기능을 제공 **② 클러스터링 지원** - 여러 노드를 묶어 클러스터를 구성하고, 클러스터 내에서 컨테이너의 부하 분산 및 리소스 관리를 수행 **③ 모니터링 및 제어** - 중앙 서버에서 모든 노드의 상태를 실시간으로 모니터링하고, 필요 시 원격으로 컨테이너를 시작/중지하거나 설정을 변경할 수 있는 기능을 제공		

4. 성능 요구사항

요구사항 고유번호		PER-001		
요구사항 명칭		컨테이너 시작 시간		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	제한된 리소스 환경에서도 빠른 컨테이너 시작 시간을 제공		
	세부내용	**① 처리 시간** - 컨테이너 시작 명령 후 빠르게 컨테이너를 실행, Docker 대비 향상된 처리 속도를 목표 **② 응답 시간** - 명령어 입력 후 즉시 컨테이너가 실행될 수 있도록 최소 지연 시간이 보장		

요구사항 고유번호		PER-002		
요구사항 명칭		메모리 사용량		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	각 컨테이너가 사용하는 메모리 자원을 효율적으로 관리하여 제한된 리소스 환경에서도 안정적으로 동작할 수 있어야 함		
	세부내용	**자원 사용량** - 각 컨테이너의 메모리 사용량이 최소화되도록 최적화, 불필요한 메모리 사용을 줄이고 필요 시 메모리를 자동으로 해제하는 기능을 포함		

요구사항 고유번호		PER-003		
요구사항 명칭		CPU 사용률		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	CPU 자원을 효율적으로 사용해야 하며, 안정성을 유지		
	세부내용	**CPU 사용률** - 컨테이너가 실행 중일 때 시스템의 CPU 사용량이 과도하지 않도록 관리, 여러 컨테이너가 동시에 실행되더라도 안정적인 CPU 사용률을 유지		

요구사항 고유번호		PER-004		
요구사항 명칭		처리량(Throughput)		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	다수의 컨테이너를 동시에 실행할 수 있는 충분한 처리량을 제공		
	세부내용	**동시 실행 가능 컨테이너 수** - 여러 컨테이너가 동시에 실행될 때도 시스템이 안정적으로 운영되도록 설계, 다양한 부하 상황에서도 성능 저하 없이 컨테이너가 관리		

5. 인터페이스 요구사항

요구사항 고유번호		SIR-001		
요구사항 명칭		CLI		
요구사항 분류		인터페이스	응락수준	필수
요구사항 상세설명	정의	사용자가 명령줄에서 컨테이너 관리 기능을 사용할 수 있는 CLI 제공		
	세부내용	① 컨테이너 이미지 생성 및 확인 등의 기본 명령 지원 ② 컨테이너 생성, 시작, 중지, 삭제 등의 기본 명령 지원 ③ 컨테이너 오케스트레이션 기능 CLI 인터페이스 제공 ④ 사용자 정의 스크립트 작성 및 실행 기능 지원		
기타 고려 사항		**사용자 경험** - CLI는 사용자 친화적인 명령어 구문을 제공하여, 사용자가 쉽게 사용할 수 있어야 함		

요구사항 고유번호		SIR-002		
요구사항 명칭		API 인터페이스		
요구사항 분류		인터페이스	응락수준	필수
요구사항 상세설명	정의	다른 시스템과 연동할 수 있도록 REST API를 제공		
	세부내용	① 시스템이나 애플리케이션이 컨테이너 런타임 엔진을 제어하고 상태를 조회할 수 있도록 REST API를 제공 ② API를 통해 컨테이너 생성, 시작, 중지 등의 기능을 외부 시스템에서 호출 ③ API를 통해 각 컨테이너의 상태나 자원 사용량 데이터를 요청할 수 있어야 하며, JSON 형식으로 응답을 제공		
기타 고려 사항		**① 보안** - API 통신은 암호화되어야 하며, 인증 및 권한 관리가 필수적 **② 확장성** - 추후 기능 확장 시 새로운 API 엔드포인트가 쉽게 추가될 수 있도록 설계		

6. 데이터 요구사항

요구사항 고유번호	DAR-001		
요구사항 명칭	컨테이너 데이터 식별 및 분류		
요구사항 분류	데이터	응락수준	필수
요구사항 세부내용	**① 데이터 식별** - 엔진에서 필요한 모든 데이터를 식별, 용도를 정의 **② 성능 데이터 관리** - 각 컨테이너 실행 시 성능 데이터(예: 메모리, CPU 사용량, 실행 시간)를 자동으로 기록하고 저장할 수 있는 구조가 마련		

요구사항 고유번호	DAR-002		
요구사항 명칭	데이터 관리 및 보존		
요구사항 분류	데이터	응락수준	필수
요구사항 세부내용	**데이터 보존 기간** - 시스템에서 수집된 성능 데이터는 일정 기간 동안 보존되어야 하며, 분석 및 최적화를 위한 충분한 기간이 지나면 안전하게 삭제되거나 별도의 보관 서버로 이관		

7. 테스트 요구사항

요구사항 고유번호	TER-001		
요구사항 명칭	테스트 방안		
요구사항 분류	테스트	응락수준	필수
요구사항 세부내용	**① 단위 테스트** - 각 기능 모듈(예: 컨테이너 생성, 시작, 중지, 삭제)에 대해 개별적인 단위 테스트를 수행하여, 해당 기능이 정상적으로 동작하는지 확인 **② 통합 테스트** - 각 모듈 간의 상호작용을 확인하기 위해 통합 테스트를 수행 **③ 성능 테스트** - 다양한 부하 상황에서도 안정적으로 동작하는지 확인하기 위해 성능 테스트 수행. 메모리 및 CPU 사용량, 컨테이너 시작 시간 등을 측정하여 성능을 분석 **④ 시스템 테스트** - 모든 기능이 시스템 차원에서 통합적으로 작동하는지 점검		

8. 보안 요구사항

요구사항 고유번호	SER-001		
요구사항 명칭	인증 및 권한 관리		
요구사항 분류	보안	응락수준	필수
요구사항 세부내용	**① 사용자 인증** - 시스템에 접근하는 모든 사용자는 계정 인증 절차를 거침 **② 권한 관리** - 각 사용자는 시스템 내에서 수행할 수 있는 작업에 대해 세부적인 권한이 부여되어야 하며, 불필요한 권한 부여는 최소화		

요구사항 고유번호	SER-002		
요구사항 명칭	데이터 보안		
요구사항 분류	보안	응락수준	필수
요구사항 세부내용	**① 데이터 전송 보안** - 데이터를 송수신할 때는 SSL/TLS와 같은 안전한 통신 프로토콜을 사용 **② 접근 제어** - 접근할 수 있는 사용자는 최소화되며, 각 사용자는 자신의 권한 내에서만 데이터를 조회하거나 수정		

요구사항 고유번호	SER-003		
요구사항 명칭	네트워크 보안		
요구사항 분류	보안	응락수준	필수
요구사항 세부내용	**네트워크 접근 통제** - 시스템은 네트워크 접근 제어(Access Control List)를 통해 승인된 사용자와 시스템만이 네트워크를 통해 접근		

9. 품질 요구사항

요구사항 고유번호		QUR-001		
요구사항 명칭		신뢰성		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	안정적으로 동작하며, 지정된 조건에서 장애 없이 서비스를 제공		
	세부내용	**가용성** - 시스템 가용성은 99% 이상, 시스템 중단 시간은 최소화		

요구사항 고유번호		QUR-002		
요구사항 명칭		사용성		
요구사항 분류		품질	응락수준	선택
요구사항 상세설명	정의	사용자가 쉽게 접근하고 관리할 수 있어야 하며, 직관적인 사용자 인터페이스와 명령 체계를 제공		
	세부내용	**① 사용자 편의성** - CLI 명령어는 직관적이고 쉽게 사용할 수 있도록 설계 **② 매뉴얼** - 시스템 사용자를 위한 매뉴얼을 제공		

요구사항 고유번호		QUR-003		
요구사항 명칭		사용성		
요구사항 분류		품질	응락수준	선택
요구사항 상세설명	정의	다른 컨테이너 런타임에서 사용되는 이미지와 호환		
	세부내용	**타 컨테이너 런타임 이미지 호환성** - Docker 등 다른 컨테이너 런타임에서 사용되는 이미지 형식과 호환되어야 한다. 사용자는 기존에 사용하던 Docker 이미지 등을 경량화된 런타임에서도 사용		

10. 제약 사항

요구사항 고유번호	COR-001		
요구사항 명칭	개발 환경 제약		
요구사항 분류	제약사항	응락수준	필수
요구사항 세부내용	**① 소프트웨어 제약** - 개발 환경은 최소한의 리소스 사용을 목표로 해야 하며, 대규모 개발 프레임워크 사용이 제한 **② OS 제약** - 개발 및 배포는 Linux 기반 운영 체제를 중심으로 이루어져야 한다. 타 운영 체제는 추가 지원을 고려하지 않으며, Linux 커널 관련 기술 최적화를 우선		

요구사항 고유번호	COR-002		
요구사항 명칭	시스템 설계 제약		
요구사항 분류	제약사항	응락수준	필수
요구사항 세부내용	**경량화 최적화** - 시스템 설계는 최소한의 리소스 사용을 보장하기 위해 경량화된 아키텍처를 채택, 복잡한 모듈이나 기능의 도입은 제한		

11. 프로젝트 관리 요구사항

요구사항 고유번호	PMR-001		
요구사항 명칭	버전관리		
요구사항 분류	프로젝트 관리	응락수준	필수
요구사항 세부내용	- 소스코드 및 구성 파일의 버전 관리 기능 - 다중 브랜치 병합 기능을 통한 여러 개발자 동시 작업 가능 - 팀원 간의 협업을 위한 기능 포함으로 코드 공유 및 협업 작업에 용이 - 프로젝트 릴리스 및 배포를 위한 버전 태깅 및 릴리스 노트 생성 기능		

요구사항 고유번호	PSR-001		
요구사항 명칭	품질관리		
요구사항 분류	프로젝트 지원	응락수준	선택
요구사항 세부내용	**품질 감사** - 주기적으로 품질 감사를 실시하여 프로젝트가 품질 목표를 충족하고 있는지 평가		

요구사항 고유번호	PSR-002		
요구사항 명칭	커뮤니케이션 관리		
요구사항 분류	프로젝트 지원	응락수준	선택
요구사항 세부내용	**커뮤니케이션 도구** - 효율적인 커뮤니케이션을 위해 적절한 도구(Notion, Microsoft Teams 등) 사용, 모든 중요한 의사소통은 문서화되어 저장		