

백엔드 과제


백엔드 과제 안내서

"이벤트 / 보상 관리 플랫폼 구축"

환영합니다, 신입 백엔드 개발자님!

당신은 이번에 합류한 백엔드 팀의 새로운 팀원입니다.

첫 번째 미션으로는 실무에서 자주 사용되는 패턴을 학습하고, 실제 서비스에 적용 가능한 이벤트/보상 관리 시스템을 설계하고 구현해보게 됩니다.

 "실제 프로덕션에서 이 코드를 돌려도 될까?"를 염두에 두고 고민해보세요.

이 프로젝트는 나중에 팀에서 사용하는 실제 사내 도구 또는 실서비스의 모듈로 발전할 수 있습니다.

[실무 기획자와의 커뮤니케이션 시나리오]

PM (기획자):

유저 대상 이벤트를 자주 하는데, 조건 확인도 어렵고 보상 지급도 수작업이라 너무 비효율적이에요.

특히 유저가 조건을 달성했는지 확인하고 포인트나 아이템을 줘야 하는데, 엑셀로 수작업 하고 있어요.

개발자 (당신):

이벤트 조건이 유저 행동 기반이니까, 조건 검증 로직과 보상 지급 자동화가 필요해 보이네요.

운영자는 이벤트 만들고 보상 설정하고, 유저는 조건 만족 후 직접 요청하면 되겠네요.

PM:

맞아요. 유저가 요청하면, 운영자가 검토하거나 자동으로 주는 걸로 진행하고 싶어요.

그리고 감사 담당자는 지급 내역만 조회할 수 있어야 해요.

개발자:

알겠습니다. 역할 구분이 명확하니 인증과 권한 시스템도 꼭 함께 넣어야겠네요.

기술 스택 요약

항목	버전/도구
Node.js	18 (고정)
NestJS	최신
DB	MongoDB
인증	JWT
배포/실행	Docker + docker-compose
언어	TypeScript

과제 개요

목표:

NestJS + MSA + MongoDB 기반으로 **이벤트 보상 시스템**을 3개의 서버 구조로 구축

주요 기능:

이벤트 생성, 보상 정의, 유저 보상 요청, 관리자 및 감사자 확인 기능 포함

서버 구성:

서버	주요 역할
Gateway Server	모든 API 요청의 진입점, 인증, 권한 검사 및 라우팅

Auth Server	유저 정보 관리, 로그인, 역할 관리, JWT 발급
Event Server	이벤트 생성, 보상 정의, 보상 요청 처리, 지급 상태 저장

기능 상세

Gateway Server

- 모든 요청을 받아 라우팅 수행
- JWT 토큰 검증 및 역할(Role) 검사
- NestJS의 `@nestjs/passport`, `AuthGuard`, `RolesGuard` 사용

Auth Server

- 유저 등록 / 로그인 / 역할(role) 관리
- JWT 관리
- 예시 역할:

역할	권한 설명
USER	보상 요청 가능
OPERATOR	이벤트/보상 등록
AUDITOR	보상 이력 조회만 가능
ADMIN	모든 기능 접근 가능

Event Server

1. 이벤트 등록 / 조회

- 운영자 또는 관리자가 이벤트를 생성할 수 있어야 합니다.
- 이벤트에는 조건(예: 로그인 3일, 친구 초대 등)과 기간, 상태(활성/비활성) 정보가 포함됩니다.
- 등록된 이벤트는 목록 또는 상세 조회가 가능해야 합니다.

2. 보상 등록 / 조회

- 이벤트에 연결된 보상 정보를 추가할 수 있어야 합니다.
- 보상은 포인트, 아이템, 쿠폰 등 자유롭게 구성 가능하며 수량이 포함됩니다.

- 각 보상은 어떤 이벤트와 연결되는지가 명확해야 합니다.
-

3. 유저 보상 요청

- 유저는 특정 이벤트에 대해 **보상을 요청**할 수 있어야 합니다.
 - 시스템은 조건 충족 여부를 검증해야 합니다.
 - **중복 보상 요청은 막아야 하며**, 요청 상태(성공/실패 등)는 기록해야 합니다.
-

4. 보상 요청 내역 확인

- 유저는 본인의 요청 이력을 볼 수 있어야 합니다.
 - 운영자 / 감사자 / 관리자는 전체 유저의 요청 기록을 조회할 수 있어야 합니다.
 - 필터링 기능(이벤트별, 상태별 등)은 선택적으로 구현해도 됩니다.
-

인증 구조

- JWT 기반 인증
-

제출 방식

- GitHub 링크 제출 (public 설정)
- `README.md`에 포함:
 - 실행 방법 (Docker Compose)

Q&A: 지원자들이 자주 물어볼 만한 질문들

Q1. 이벤트 종류는 정해져 있나요?

A: 아니요!

이벤트 종류와 조건 로직은 자유롭게 정하셔도 됩니다.

예를 들어 "7일 연속 출석", "친구 3명 초대", "특정 퀘스트 클리어" 등 현실적이면서 검증 가능한 구조라면 어떤 형태도 가능합니다.

Q2. 보상 아이템은 어떤 걸로 해야 하나요?

A: 보상 종류도 자유롭게 정의하세요.

게임 아이템, 포인트, 쿠폰, 재화 등 상황에 맞는 가상의 보상을 구성하시면 됩니다. 단, DB 스키마로는 어떤 보상인지 명확히 표현되어야 합니다.

Q3. 프론트엔드는 제공되나요? 또는 만들 필요 있나요?

A: 프론트엔드는 필요 없습니다.

이 과제는 **백엔드 API 설계, 인증, 권한 제어, 비즈니스 로직 구현**에 중점을 두고 있습니다.

API 호출을 통해 테스트 가능한 수준이면 충분합니다.

Q4. 테스트 코드도 작성해야 하나요?

A: 선택사항입니다.

시간이 허락된다면 간단한 단위 테스트 또는 통합 테스트를 추가하시면 가산점이 될 수 있습니다. 특히 서비스 간 책임 분리가 잘 드러나는 부분을 테스트하면 좋습니다.

Q5. API 경로, HTTP 메서드, 응답 구조는 어떻게 정해야 하나요?

A: 구체적인 API 경로, HTTP 메서드(GET, POST, etc), 요청/응답 구조는 지원자가 자유롭게 설계해주세요.

Q6. 설계 이유나 추가 설명이 필요한 부분이 있으면 어디에 적어야 하나요?

A: `README.md`에 자유롭게 작성하시면 됩니다.

이벤트 설계, 조건 검증 방식, API 구조 선택 이유, 또는 구현 중 겪은 고민 등

어떤 부연 설명이든 간단히 정리해주시면 큰 도움이 됩니다.

꼭 정해진 형식은 없지만, 본인의 설계 의도를 보여줄 수 있는 좋은 기회입니다.