# 실전 포트폴리오 기획: 식품 안전 규정(HACCP) 기반 MES 프로젝트 개발 및 배포 로드맵

## Part I: 전략적 기반 및 시스템 설계

이 파트에서는 프로젝트의 '왜'를 정립합니다. 단순한 기술 과제를 넘어, 식품 제조 산업의 실제 문제를 해결하는 시스템을 설계함으로써 강력한 포트폴리오를 구축하는 것을 목표로 합니다.

### 제1장: 개념 프레임워크: 일반적인 MES를 넘어서

#### 프로젝트 가치 제안 정의

이 프로젝트의 목표는 단순한 제조 실행 시스템(MES)을 만드는 것이 아닙니다. 핵심 가치는 식품 산업의 가장 중요한 규제인 **위해요소 중점관리기준(HACCP) 프레임워크를 중심으로 구축된 식품 안전 및 규정 준수 플랫폼**을 개발하는 데 있습니다. 이러한 전략적 포지셔닝은 프로젝트의 전문성과 산업 관련성을 즉각적으로 격상시킵니다.

#### 핵심 개념

* **제조 실행 시스템(MES) 개요**: MES는 전사적 자원 관리(ERP) 시스템과 공장 현장의 공정 제어 시스템 사이의 기능적 계층을 제공하여, 의사 결정권자가 생산 현장을 보다 효율적으로 운영하는 데 필요한 데이터를 제공하는 핵심 소프트웨어입니다.1
* **표준 MES 모듈**: 일반적인 MES는 생산 계획 및 스케줄링, 가동 중지 시간 추적, 전반적인 장비 효율성(OEE) 분석, 인력 추적, 재고 관리와 같은 모듈을 포함합니다.2
* **핵심 차별점: HACCP 중심 설계**: 이 프로젝트는 HACCP 시스템을 중앙 설계 기둥으로 삼습니다. HACCP의 7가지 원칙—위해요소 분석, 중요 관리점(CCP) 결정, 한계 기준 설정, 모니터링 체계 확립, 개선 조치 방법 수립, 검증 절차 및 방법 수립, 문서화 및 기록 유지—을 디지털화하고 자동화하는 것이 프로젝트의 핵심 목표입니다.4
* **대상 산업의 특수성**: 식품 및 음료와 같은 규제 산업은 품질 관리, 제품 추적, 규제 준수를 위해 MES를 도입함으로써 특히 큰 이점을 얻습니다.1 이 프로젝트는 이러한 고부가가치 요구사항을 직접적으로 해결합니다.

#### 설계 철학 및 기술적 시사점

식품 MES의 가장 큰 가치는 단순히 생산 효율성을 높이는 데 있지 않고, **리스크 완화**에 있습니다. 기술 회사의 데이터 유출도 비용이 많이 들지만, 식품 안전 사고는 공중 보건과 브랜드 생존에 치명적인 결과를 초래할 수 있습니다. 따라서 시스템 아키텍처는 데이터 무결성, 감사 가능성, 실시간 경고 기능을 다른 어떤 기능보다 우선시해야 합니다.

이러한 안전 우선 원칙은 모든 중요 데이터 포인트, 특히 CCP 로그에 대해 불변(immutable) 또는 버전 관리된 기록이 필요하다는 기술적 요구사항으로 이어집니다. CCP 측정값에 대한 단순한 UPDATE 쿼리는 애플리케이션 또는 데이터베이스 수준에서 허용되어서는 안 됩니다. 대신, 변경 사유와 함께 원본을 참조하는 새로운 수정 항목을 생성해야 합니다. 이는 잠재적 고용주에게 깊은 인상을 줄 수 있는 감사 가능한 추적 기록을 생성합니다.

### 제2장: 기능 아키텍처 및 MVP 범위 설정

#### 프로젝트 범위 정의

성공적인 프로젝트 완수를 위해 명확하고 달성 가능한 최소 기능 제품(MVP) 범위를 정의하고, 향후 확장 로드맵을 수립합니다. 이는 프로젝트가 비대해지는 것을 방지하고 실질적인 결과물을 보장합니다.

#### 최소 기능 제품(MVP) 모듈

* **모듈 1: 코어 및 사용자 관리**
  + **기능**: 보안이 적용된 사용자 등록 및 로그인. 역할 기반 접근 제어(RBAC)를 통해 관리자, 품질 관리자, 현장 작업자 등 다양한 사용자 유형에 따라 권한을 차등 부여합니다.
  + **근거**: 데이터 보안과 적절한 권한 관리를 보장하기 위한 모든 엔터프라이즈 시스템의 기초입니다.
* **모듈 2: 원자재 및 재고 관리**
  + **기능**: 공급업체 등록, 원자재 정보 정의, 입고 시 로트/배치 번호 추적. 이를 통해 원자재의 출처를 기록합니다.6
  + **근거**: 이는 추적성의 첫 단계("one step back")입니다. 구성 부품을 모르면 완제품을 추적하는 것은 불가능합니다.
* **모듈 3: 생산 관리**
  + **기능**: 생산 오더를 생성하고 관리합니다. 완제품에 대한 레시피/배합을 정의하고 특정 원자재 유형과 연결합니다.6 오더의 상태(예: 대기, 진행 중, 완료)를 추적합니다.
  + **근거**: 모든 MES의 핵심 운영 모듈입니다.2
* **모듈 4: HACCP 규정 준수 및 CCP 모니터링**
  + **기능**: 작업자가 진행 중인 생산 오더에 대해 정의된 중요 관리점(CCP)에서 측정값을 기록할 수 있는 디지털 로그북. 이 모듈은 프로젝트의 핵심입니다.
  + **예시**: 조리된 가금류의 내부 온도 입력 양식 (CCP: 가열 공정, 위해요소: 병원균 생존, 한계 기준: 74°C 초과).7 냉각 공정을 위한 타이머 및 온도 로그 (CCP: 냉각 공정, 위해요소: 세균 증식, 한계 기준: 4시간 이내 5°C로 냉각).7
  + **근거**: 이는 모니터링, 기록 유지, 한계 기준 설정이라는 HACCP의 핵심 원칙을 직접 구현하며, 종이 기반 프로세스를 실시간 디지털 프로세스로 전환합니다.1

#### Post-MVP / 고급 기능 로드맵

* **전체 추적성 엔진**: 순방향/역방향 추적성 보고서를 구현합니다. 완제품 로트 번호가 주어지면 사용된 모든 원자재 로트를 표시하고, 원자재 로트 번호가 주어지면 해당 원자재가 사용된 모든 완제품을 표시합니다.5
* **실시간 대시보드 및 OEE**: 전반적인 장비 효율성(OEE = 가용성 × 성능 × 품질)을 계산하고 표시합니다.2 실시간 생산 수량 및 가동 중지 이벤트를 시각화합니다.
* **개선 조치 및 이탈 관리**: CCP 로그가 한계 기준을 벗어났을 때 발생하는 상황을 관리하는 워크플로우 모듈입니다.4 여기에는 알림, 근본 원인 분석, 개선 조치 문서화가 포함됩니다.
* **문서 관리**: 표준 운영 절차(SOP), 레시피, 규정 준수 문서를 저장하고 버전을 관리하는 시스템입니다.1

### 제3장: 기술 아키텍처 및 데이터베이스 설계

#### 시스템 아키텍처 개요

* **백엔드**: Django와 Django REST Framework(DRF)를 사용하여 JSON API를 제공합니다.
* **프론트엔드**: React (Create React App 또는 Vite 사용)를 단일 페이지 애플리케이션(SPA)으로 구축합니다.
* **데이터베이스**: 트랜잭션 데이터(생산 오더, 재고)를 위해 MariaDB를 사용합니다.
* **배포**: Docker 및 Docker Compose를 통한 컨테이너화, Nginx를 리버스 프록시 및 정적 파일 서버로 활용합니다.

#### 데이터베이스 전략: 다중 DB에 대한 고찰

사용자 요구사항은 MariaDB와 MongoDB를 모두 언급했습니다. Django는 다중 데이터베이스를 지원하지만 9, 이 프로젝트의 MVP는 MES의 핵심인 트랜잭션 무결성을 위해 단일 MariaDB 데이터베이스를 사용하는 것으로 설계합니다.

다만, 아키텍처 설계의 깊이를 보여주기 위해, 추후에 MongoDB와 같은 두 번째 데이터베이스를 도입할 수 있는 이유와 방법에 대해 논의할 것입니다. 예를 들어, 대용량의 비정형 로그 데이터나 센서 데이터의 경우 NoSQL 솔루션이 더 적합할 수 있습니다. 이 경우 settings.py에 두 번째 DB를 정의하고, 특정 모델을 올바른 데이터베이스로 라우팅하기 위한 데이터베이스 라우터(routers.py)를 생성하는 과정을 포함할 수 있습니다.10

#### 제안 테이블 3.1: 식품 MES를 위한 핵심 데이터베이스 스키마 (MVP)

이 스키마는 프로젝트 요구사항을 구체적인 데이터 구조로 변환한 것으로, Django 모델의 청사진 역할을 합니다. 잘 설계된 스키마는 문제 영역에 대한 깊은 이해를 보여주는 가장 중요한 산출물입니다.

| 모델명 (Model Name) | 주요 필드 (Key Fields) | 데이터 타입 (Data Type) | 관계 (Relationships) | 설계 근거 및 참조 |
| --- | --- | --- | --- | --- |
| User | (Django User 확장) role | CharField | - | 인증 및 권한 관리. 역할: 'admin', 'manager', 'operator'.12 |
| Supplier | name, contact\_info, address | CharField, TextField | - | 원자재 출처 추적, 추적성의 첫 단계.6 |
| RawMaterial | name, description, supplier | CharField, TextField | ForeignKey(Supplier) | 생산에 사용되는 원자재 카탈로그 정의. |
| MaterialLot | lot\_number, material, quantity, receipt\_date, expiry\_date | CharField, DecimalField, DateField | ForeignKey(RawMaterial) | 입고된 원자재의 특정 배치를 나타냄. 로트 단위 추적성의 핵심.5 |
| FinishedProduct | name, sku, recipe\_description | CharField, TextField | - | 공장에서 제조하는 완제품 정의. |
| ProductionOrder | order\_id, product, quantity, status, created\_at | CharField, DecimalField | ForeignKey(FinishedProduct) | 제조 공정을 관리하는 중앙 객체.2 |
| CCP | name, description, process\_step, hazard, critical\_limit\_description | CharField, TextField | ManyToManyField(FinishedProduct) | HACCP 계획에 따른 중요 관리점 정의. 이 검사가 필요한 제품과 연결.7 |
| CCPLog | order, ccp, timestamp, measured\_value, is\_within\_limits, operator, corrective\_action\_taken | DateTimeField, CharField, BooleanField, TextField | ForeignKey(ProductionOrder), ForeignKey(CCP), ForeignKey(User) | 안전 점검에 대한 불변의 기록. HACCP 모듈의 심장. measured\_value는 유연성을 위해 JSONField 사용 가능.4 |

## Part II: 개발 로드맵: 코드에서 기능까지

이 파트에서는 설계를 기능적 코드로 변환하여 애플리케이션을 구축하는 상세한 '방법'을 제공합니다.

### 제4장: Django REST Framework를 이용한 백엔드 구현

#### 단계별 가이드

1. **프로젝트 설정**: 가상 환경을 초기화하고, Django와 DRF를 설치(pip install django djangorestframework)한 후, 프로젝트와 core 앱을 생성합니다.14
2. **모델 구현**: 제안 테이블 3.1의 스키마를 models.py 파일에 Django 모델로 변환합니다. makemigrations와 migrate를 실행하여 데이터베이스에 적용합니다.12
3. **직렬화기(Serializers) 생성**: serializers.py 파일을 생성합니다. 각 모델에 대해 ModelSerializer를 만들어 복잡한 데이터 타입(쿼리셋, 모델 인스턴스)을 JSON으로 변환하고 들어오는 데이터의 유효성을 검사하는 역할을 정의합니다.14
4. **뷰(Views) 및 엔드포인트 생성**: views.py 파일을 생성합니다. Supplier, RawMaterial과 같은 모델에 대해서는 DRF의 ModelViewSet을 사용하여 신속하게 CRUD 엔드포인트를 개발합니다.14  
   CCPLog와 같이 CCP의 한계 기준에 따라 유효성을 검사해야 하는 더 복잡한 로직의 경우, ViewSet의 .create() 메서드를 오버라이드하거나 커스텀 APIView를 사용합니다.
5. **URL 라우팅**: 앱과 프로젝트 디렉토리에 urls.py를 설정합니다. DRF의 DefaultRouter를 사용하여 ViewSet에 대한 URL을 자동으로 생성합니다.14

#### 아키텍처 결정의 깊이

DRF 튜토리얼은 ModelViewSet의 편리함을 강조하지만 14, HACCP 요구사항은 단순한 데이터 저장을 넘어선 비즈니스 로직을 포함합니다. 예를 들어, 기록된 측정값이 규정을 준수하는지 여부는 백엔드에서 관련

CCP 모델에 정의된 규칙에 따라 계산되어야 합니다. 표준 ModelViewSet은 들어오는 데이터를 맹목적으로 수락할 수 있으므로, 이처럼 중요한 로직에는 ModelViewSet의 메서드를 오버라이드하거나 더 명시적인 APIView를 사용하는 것이 바람직합니다. 이는 프레임워크의 기본 기능을 넘어서는 성숙한 아키텍처 결정을 보여줍니다.

### 제5장: Django Simple JWT를 이용한 안전한 인증 구현

#### 구현 세부 정보

1. **설치 및 설정**: djangorestframework-simplejwt를 설치하고 17,  
   settings.py의 INSTALLED\_APPS에 추가합니다. DRF의 DEFAULT\_AUTHENTICATION\_CLASSES를 JWT 인증을 사용하도록 설정합니다.17
2. **URL 설정**: urls.py에 라이브러리가 제공하는 TokenObtainPairView (/api/token/)와 TokenRefreshView (/api/token/refresh/) 경로를 추가합니다.17
3. **JWT 흐름**: 클라이언트는 사용자 이름/비밀번호를 /api/token/으로 보내 액세스 토큰(단기)과 리프레시 토큰(장기)을 받습니다. 이후 요청에는 Authorization 헤더에 액세스 토큰을 포함하여 보냅니다. 액세스 토큰이 만료되면, 리프레시 토큰을 사용하여 사용자가 다시 로그인할 필요 없이 새로운 액세스 토큰을 발급받습니다.18

#### 보안 고려사항

대부분의 튜토리얼은 편의를 위해 JWT를 localStorage에 저장하는 방법을 보여주지만 20, 이는 XSS(Cross-Site Scripting) 공격에 취약합니다. 더 안전한 접근 방식은

HttpOnly 쿠키에 토큰을 저장하는 것입니다. HttpOnly 쿠키는 클라이언트 측 스크립트로 접근할 수 없어 XSS 위험을 완화합니다. 이 프로젝트에서는 구현의 단순성을 위해 localStorage를 사용하되, HttpOnly 쿠키 방식이 프로덕션 환경의 표준임을 언급하여 보안 아키텍처에 대한 이해를 보여주는 것이 중요합니다.

### 제6장: React를 이용한 프론트엔드 구현

#### 개발 프로세스

1. **프로젝트 설정**: npx create-react-app 또는 npm create vite@latest를 사용하여 프로젝트를 부트스트랩합니다.22  
   components, pages, hooks, services, context와 같은 디렉토리 구조를 설정합니다.
2. **컴포넌트 계층 구조**: 재사용 가능한 컴포넌트(Button, Input, Table)와 뷰 레벨 컴포넌트/페이지(LoginPage, DashboardPage, CCPLogForm)를 설계합니다.
3. **라우팅**: react-router-dom을 사용하여 페이지 간 내비게이션을 관리합니다. 인증 상태를 확인하여 인증되지 않은 사용자를 로그인 페이지로 리디렉션하는 ProtectedRoute 컴포넌트를 구현합니다.18
4. **상태 관리**: React Hooks를 적극적으로 활용합니다. 로컬 컴포넌트 상태에는 useState, 데이터 가져오기와 같은 부수 효과에는 useEffect, 전역 상태(인증 상태, 사용자 정보 등) 관리에는 useContext와 useReducer를 조합하여 사용합니다.18
5. **양식(Forms) 관리**: 데이터 입력을 위한 양식(로그인, 생산 오더 생성, CCP 로그 제출 등)을 구축하고, 상태 및 유효성 검사를 관리합니다.

#### 코드의 확장성 및 유지보수성

MES와 같이 데이터 집약적인 애플리케이션에서는 불필요한 리렌더링을 방지하고 복잡한 상태 의존성을 관리하는 것이 중요합니다. 이를 위해 데이터 가져오기, 로딩 및 오류 상태 관리를 캡슐화하는 커스텀 훅 useApi를 설계할 것입니다. 예를 들어, useApi(endpoint) 훅은 URL 엔드포인트를 인수로 받아 { data, isLoading, error } 객체를 반환합니다. 이 패턴은 여러 컴포넌트에서 코드 중복을 줄이고, 컴포넌트를 더 깔끔하게 만들며, 테스트를 용이하게 합니다. 이는 전문적인 React 개발에서 볼 수 있는 확장 가능하고 유지보수 가능한 코드를 작성하는 능력을 보여줍니다.

### 제7장: Axios를 이용한 프론트엔드와 백엔드 통합

#### 구현 세부 정보

1. **Axios 설정**: Axios를 설치(npm install axios)하고, Django API의 baseURL이 미리 설정된 Axios 인스턴스를 내보내는 중앙 집중식 API 서비스 모듈(services/api.js)을 생성합니다.20
2. **인증된 요청**: 요청 인터셉터는 localStorage에 액세스 토큰이 있는 경우 모든 나가는 요청에 Authorization: Bearer <accessToken> 헤더를 자동으로 추가합니다.20
3. **자동 토큰 리프레시 구현**: 응답 인터셉터를 구현하여 이 프로젝트의 핵심적인 사용자 경험을 개선합니다.
   * 401 Unauthorized 상태 코드를 가진 응답을 감지합니다. 이는 액세스 토큰이 만료되었음을 의미합니다.20
   * 401 오류가 감지되면 원래 요청을 일시 중지합니다.
   * 저장된 리프레시 토큰을 사용하여 Django의 /api/token/refresh/ 엔드포인트에 요청을 보냅니다.
   * 새로운 액세스 토큰을 받으면 localStorage를 업데이트합니다.
   * 마지막으로, 원래 실패했던 요청을 새로운 토큰으로 재시도합니다. 이 과정은 사용자에게는 매끄럽게 보입니다.20

#### 비동기 프로그래밍의 깊은 이해

리프레시 토큰 인터셉터는 중요한 UX 문제를 해결하지만, 동시에 여러 API 요청이 만료된 토큰으로 전송될 때 발생하는 경쟁 조건(race condition)을 야기할 수 있습니다. 순진한 구현은 여러 개의 리프레시 요청을 발생시킬 수 있습니다. 전문적인 구현은 첫 번째 실패한 요청만이 리프레시 로직을 트리거하고, 후속 실패 요청들은 새로운 토큰을 기다리도록 '대기열'에 넣는 메커니즘을 포함합니다. 플래그나 프라미스 기반 큐를 사용하여 이 고급 로직을 설계함으로써 비동기 프로그래밍 과제에 대한 깊은 이해를 보여줄 수 있습니다.21

## Part III: 프로덕션 배포 및 프로젝트 쇼케이스

이 파트에서는 애플리케이션을 실제 환경에 배포하기 위해 컨테이너화, 배포, 그리고 프로젝트를 효과적으로 선보이는 방법에 초점을 맞춥니다.

### 제8장: Docker 및 Docker Compose를 이용한 컨테이너화

#### 파일 생성

1. **Django용 Dockerfile**:
   * Python 베이스 이미지에서 시작하여, requirements.txt로부터 의존성을 설치하고, 애플리케이션 코드를 복사합니다.
   * 프로덕션 준비가 된 WSGI 서버인 Gunicorn을 사용하여 Django 애플리케이션을 실행하는 CMD를 정의합니다.26
2. **React용 Dockerfile (다단계 빌드)**:
   * **1단계 (빌드)**: Node.js 이미지에서 시작하여 npm install을 실행하고 npm run build를 통해 최적화된 정적 빌드를 생성합니다.28
   * **2단계 (서빙)**: 경량 Nginx 이미지에서 시작하여, 1단계에서 생성된 정적 파일을 Nginx의 HTML 디렉토리로 복사합니다.28 이 다단계 빌드 방식은 최종 Docker 이미지의 크기를 크게 줄여 더 작고 안전한 이미지를 만듭니다.
3. **docker-compose.yml**:
   * backend(Django), frontend(React), db(MariaDB) 서비스를 정의합니다.
   * 네트워크, 영구 데이터 저장을 위한 볼륨, 환경 변수를 설정합니다.26

### 제9장: Nginx를 이용한 프로덕션 환경 구성

#### Nginx 구성 (nginx.conf)

Nginx는 이 아키텍처에서 단일 진입점(single entry point) 역할을 합니다. 들어오는 모든 요청을 받아 React 앱으로 보내거나 API 요청을 Django 백엔드로 전달합니다.

* **server 블록**: 포트 80에서 수신 대기합니다.
* **location / 블록**: React 애플리케이션의 정적 파일을 루트 디렉토리에서 제공합니다. SPA의 경우, 모든 경로를 index.html로 라우팅하는 try\_files 지시어가 필수적입니다.26
* **location /api/ 블록**: /api/로 시작하는 모든 요청을 Gunicorn이 실행 중인 backend 서비스로 전달하는 리버스 프록시 역할을 합니다.26
* **location /static/, location /media/ 블록**: alias 지시어를 사용하여 공유 볼륨에서 정적 및 미디어 파일을 직접 제공합니다. 이는 Gunicorn을 거치지 않으므로 성능이 향상됩니다.26

#### 제안 테이블 9.1: Nginx 구성 치트 시트

이 표는 이 스택에서 가장 중요하고 자주 잘못 구성되는 Nginx 지시어에 대한 빠른 참조 가이드를 제공합니다.

| 위치 블록 (Location Block) | 지시어 (Directive) | 값 예시 (Value Example) | 설계 근거 및 참조 |
| --- | --- | --- | --- |
| / | root | /var/www/html; | 메인 React 앱 제공. |
| / | try\_files | $uri /index.html; | SPA에 필수적. 파일이 아닌 모든 요청을 index.html로 라우팅하여 React Router가 처리하도록 함. |
| /api/ | proxy\_pass | http://backend:8000; | API 요청을 Django/Gunicorn 컨테이너로 전달.26 |
| /api/ | proxy\_set\_header | Host $host; | Django의 ALLOWED\_HOSTS 및 CSRF 검사에 필요한 원본 호스트 헤더 전달.26 |
| /static/ | alias | /path/to/shared/static/volume/; | Django의 수집된 정적 파일을 애플리케이션 서버를 우회하여 직접 효율적으로 제공.26 |

#### 일반적인 문제 해결

* **CORS 오류**: Django의 django-cors-headers를 설정하는 것이 주된 해결책이지만, Nginx에서도 필요 시 헤더를 추가할 수 있습니다.19
* **CSRF 오류**: Nginx가 Host, X-Forwarded-For, X-Forwarded-Proto와 같은 필수 헤더를 전달해야 하며, Django의 CSRF\_TRUSTED\_ORIGINS 설정도 필요합니다.26
* **ALLOWED\_HOSTS 오류**: Django의 ALLOWED\_HOSTS에는 localhost뿐만 아니라 Nginx가 서비스하는 실제 도메인 이름이 포함되어야 합니다.26

### 제10장: 포트폴리오 프로젝트 최종화 및 발표

#### 문서화

* **README.md**: 프로젝트 개요, "왜" 이 프로젝트를 만들었는지(HACCP 규정 준수), 기술 스택, docker-compose up으로 실행하는 방법, 주요 기능 및 API 엔드포인트 설명을 포함하여 전문적으로 구성합니다.

#### 코드 품질

코드가 깔끔하고, 주석이 달려 있으며, PEP 8(Python) 및 Prettier(React)와 같은 컨벤션을 따르는지 확인합니다. 깨끗한 저장소는 전문 개발자의 상징입니다.

#### 면접 예상 질문 및 답변 전략

* **비즈니스 가치**: 코드가 무엇을 하는지만 설명하지 말고, 어떤 비즈니스 문제를 해결하는지 설명해야 합니다. ("이것은 단순한 CRUD 앱이 아니라, 식품 안전과 규제 준수를 보장하여 비용이 많이 드는 리콜 위험을 줄이는 디지털 솔루션입니다.")
* **아키텍처 결정**: 자신의 선택을 방어할 준비를 해야 합니다. 왜 Django와 React를 선택했는가? 왜 JWT를 사용했는가? 왜 다단계 Docker 빌드를 사용했는가? 데이터베이스 스키마를 왜 그렇게 설계했는가? 이 보고서의 내용을 가이드로 삼아 답변을 준비합니다.
* **도전 과제 및 해결책**: 직면했던 어려움(예: 리프레시 토큰 경쟁 조건 해결, CSRF를 위한 Nginx 구성)과 이를 어떻게 해결했는지 논의합니다. 이는 문제 해결 능력을 보여줍니다.

#### 최종 배포

Docker화된 애플리케이션을 AWS EC2나 DigitalOcean Droplet과 같은 클라우드 제공업체에 배포하여 채용 담당자가 실시간으로 접근할 수 있도록 최종 가이드를 제공합니다. 라이브 데모는 어떤 설명보다 강력한 인상을 남깁니다.

#### 참고 자료

1. What is an manufacturing execution system (MES)? - SAP, 9월 1, 2025에 액세스, <https://www.sap.com/products/scm/digital-manufacturing/what-is-mes.html>
2. 9 modules of the MES software that efficiently manage production operations, 9월 1, 2025에 액세스, <https://www.sagesoftware.co.in/blogs/how-mes-software-manages-production/>
3. Manufacturing Execution System - MES Software - Infor, 9월 1, 2025에 액세스, <https://www.infor.com/solutions/scm/manufacturing-execution-system>
4. The Hazard Analysis Critical Control Point (HACCP) Food Safety Management System, 9월 1, 2025에 액세스, <https://support.monnit.com/article/294-haccp>
5. 12 Core MES Features for Food Manufacturing - FoodReady, 9월 1, 2025에 액세스, <https://foodready.ai/blog/key-mes-features-for-food-manufacturing/>
6. MES system for food production - what do you need to know? - Kotrak, 9월 1, 2025에 액세스, <https://kotrak.com/en/blog/mes-system-for-food-production/>
7. HACCP Plan Examples: Samples for Different Industries (2025), 9월 1, 2025에 액세스, <https://iqx.net/blog/haccp-plan-examples>
8. Determining critical control points and their critical limits - Canada.ca, 9월 1, 2025에 액세스, <https://inspection.canada.ca/en/food-safety-industry/preventive-control-plans/critical-control-points>
9. Multiple databases | Django documentation, 9월 1, 2025에 액세스, <https://docs.djangoproject.com/en/5.2/topics/db/multi-db/>
10. How to Set Up Multiple Databases in Django [2025] | Blogs | Free ..., 9월 1, 2025에 액세스, <https://www.horilla.com/blogs/how-to-set-up-multiple-databases-in-django/>
11. Can I have 2 DBs in Django? - Reddit, 9월 1, 2025에 액세스, <https://www.reddit.com/r/django/comments/122g0yk/can_i_have_2_dbs_in_django/>
12. Django Tutorial | Learn Django Framework - GeeksforGeeks, 9월 1, 2025에 액세스, <https://www.geeksforgeeks.org/python/django-tutorial/>
13. 8 Critical Control Point Examples for HACCP | ADL Associates (Warwick) Ltd, 9월 1, 2025에 액세스, <https://www.adlassociates.co.uk/8-critical-control-point-examples-for-haccp/>
14. Getting Started with Django REST Framework (DRF) in 2024/2025 ..., 9월 1, 2025에 액세스, <https://python.plainenglish.io/getting-started-with-django-rest-framework-drf-in-2024-2025-a-beginner-friendly-guide-with-cd686a71976f>
15. How to Build a CRUD API with Django REST Framework: A Beginner's Guide for 2024/2025, 9월 1, 2025에 액세스, <https://python.plainenglish.io/how-to-build-a-crud-api-with-django-rest-framework-a-beginners-guide-for-2024-2025-fcf1b34e7326>
16. Django Rest Framework Full Crash Course 2025 | How to create API in Python - YouTube, 9월 1, 2025에 액세스, <https://www.youtube.com/watch?v=lyEwtBiyESM>
17. seankwarren/Django-React-jwt-authentication: A tutorial for user authentication between a django backend and react frontend using jwt - GitHub, 9월 1, 2025에 액세스, <https://github.com/seankwarren/Django-React-jwt-authentication>
18. Django and React -Implementing Token Authentication with Simple JWT - Sarath Kumar R S, 9월 1, 2025에 액세스, <https://sarathkumarrs.medium.com/django-and-react-implementing-token-authentication-with-simple-jwt-b2a4f290f401>
19. JWT Authentication With React JS and Django | by Ronak Chitlangya - Medium, 9월 1, 2025에 액세스, <https://medium.com/@ronakchitlangya1997/jwt-authentication-with-react-js-and-django-c034aae1e60d>
20. “Handling JWT Tokens with Axios Interceptors in React: A Comprehensive Guide” | by Saurav sharma | Medium, 9월 1, 2025에 액세스, <https://medium.com/@connect2saurav/handling-jwt-tokens-with-axios-interceptors-in-react-a-comprehensive-guide-7751f7b23386>
21. Refresh Access Token with Axios Interceptors in React.js (with TypeScript) - Stackademic, 9월 1, 2025에 액세스, <https://blog.stackademic.com/refresh-access-token-with-axios-interceptors-in-react-js-with-typescript-bd7a2d035562>
22. React JS 19 Full Course 2025 | Build an App and Master React in 2 Hours - YouTube, 9월 1, 2025에 액세스, <https://www.youtube.com/watch?v=dCLhUialKPQ>
23. React Tutorial, 9월 1, 2025에 액세스, <https://react-tutorial.app/>
24. React Refresh Token with JWT and Axios Interceptors - BezKoder, 9월 1, 2025에 액세스, <https://www.bezkoder.com/react-refresh-token/>
25. Refreshing Tokens With Axios Interceptors - YouTube, 9월 1, 2025에 액세스, <https://www.youtube.com/watch?v=16-1mTdGBoM>
26. Django, React, and Docker Walk Into a Bar — Only Nginx Gets ..., 9월 1, 2025에 액세스, <https://dev.to/documendous/django-react-and-docker-walk-into-a-bar-only-nginx-gets-served-pcf>
27. Docker | Towards serving React (Nginx) with Django API (gunicorn) - YouTube, 9월 1, 2025에 액세스, <https://www.youtube.com/watch?v=e63EBEFJkH0>
28. Deploy 4 Tier React Node MongoDB Application With Nginx | by Saifur Anik | Medium, 9월 1, 2025에 액세스, <https://saifuranik.medium.com/deploy-4-tier-react-node-mongodb-application-with-nginx-67add197577c>
29. How to dockerize a Django and React application - Honeybadger Developer Blog, 9월 1, 2025에 액세스, <https://www.honeybadger.io/blog/docker-django-react/>