docs/pms/PMS_인력관리_요구사항정의.md 파일의 내용 참고하고
docs/pms/PMS_공종별 투입가능 판정_참고.md 파일의 내용 포함
현재 pms/시스템관리/자원관리 모듈을 삭제하고 위 내용을 base 로 새모듈을 설계해줘
아래 설계는 docs/pms/PMS_인력관리_요구사항정의.md 의 PIM 기능과 docs/pms/PMS_공종별 투입가능 판정_참고.md 의 공종별 Eligibility 엔진 모델을 합쳐, 기존 pms/시스템관리/자원관리 모듈을 제거하고 대체하는 새 모듈 구상입니다.
모듈 목표·범위
단일 인력·자격 원장과 공종별 투입 가능 여부(Eligibility)를 계산/노출하는 핵심 모듈.
배정·근태·급여는 외부(기존 PMS 다른 모듈)로 두고, "배정 가능 여부"와 "자격·교육 관리"에 집중.
서브도메인·서비스
People Registry: 인적 기본, 직무/직책, 근로형태, 고용상태, 법정 식별 최소화(마스킹/비식별).
Qualification Service: 자격(CERT)·교육(TRN) 등록, 만료/경고, 검증(증빙) 워크플로우.
Work Type Catalog: Trade Group/Work Type 코드 관리, required_certs_all/any, trainings_all/any, role 최소값, 조건(capacity 등) 관리.
Eligibility Engine: 배정 시점에 규칙 평가(BLOCK/WARN), rule trace, override(Project/Site/Permit).
Assignment API: 현장/프로젝트에서 배정 요청 시 Eligibility 평가 + 결과/누락 리스트 반환.
Audit & Security: 민감정보 접근 롤(예: HR_SEC_ADMIN), 감사로그, 검증 워크플로우.
Notification: 만료 D-30/60/90 알림, 검증 지연, BLOCK 실패 알림.
핵심 기능 맵핑 (기존 FR 계열)
인력 마스터(FR-101/102/103) → People Registry + Dedup/Masking.
직무/직책/직급/경력(FR-201/202/203) → People Registry 확장.
자격·교육·만료/경고(FR-301~~304, FR-401~~404) → Qualification Service + Notification.
Eligibility 엔진(FR-501~603 + 참고 문서의 rule JSON) → Work Type Catalog + Eligibility Engine + Override.
권한/보안(FR-701~704) → Role-based 접근, 감사 로그.
데이터 모델(초안)
person(id, name, birth_yyyyMM, contact, status, role_tags[], project_org_refs, masked_fields…)
role_tag(id, name, scope) // TENANT_ADMIN, HR_SEC_ADMIN, SAFETY_ADMIN, PROJECT_MANAGER, PMS_VIEWER 등
employment(id, person_id, hire_type, grade, job_family, job_level, career_years, wage_type, active_flag)
qualification_cert(id, code, name, validity_months, needs_verification, alert_days[])
qualification_training(id, code, name, validity_months, alert_days[])

person_cert(id, person_id, cert_code, issued_at, expires_at, status[pending/verified/rejected], evidence_uri)
person_training(id, person_id, training_code, taken_at, expires_at, status, evidence_uri)
trade_group(code, name)
work_type(code, group_code, name, required_certs_all[], required_certs_any[], required_trainings_all[], required_trainings_any[], min_role_counts[], conditions{equipment, capacity_ton_min, site_flags_any[]}, enforcement{mode, reason_code})
eligibility_rule_version(rule_id, work_type_code, version, payload_json, effective_from, effective_to)
override(id, scope[project|site|permit], scope_ref, work_type_code, patch_json, approved_by, approved_at, reason)
assignment_request(id, project_id, date, work_type_code, assignees[])
assignment_result(id, assignment_request_id, eligible, assignee_results[], rule_trace, created_at)
audit_log(id, actor, action, entity, entity_id, diff, ts, ip)
API 설계(REST/gRPC 예시)
GET /work-types 목록/필터 (group, text, active)
POST /work-types 생성/수정 (ADMIN)
POST /eligibility/check 배정 전 검증
요청: {project_id, date, work_type_code, assignees[]}
응답: {eligible, assignee_results[{user_id, eligible, missing_certs[], missing_trainings[], expiring_soon[]}], rule_trace{base_rule_version, overrides_applied[]}}
POST /eligibility/override 프로젝트/사이트/퍼밋 단위 Patch 추가
POST /people / PATCH /people/{id} 인력 마스터
POST /people/{id}/certs / trainings 등록 + 증빙 업로드
POST /verify/cert 검증 승인/반려 (HR_SEC_ADMIN)
GET /alerts/expirations 만료 D-30/60/90 구독
GET /audit 감사 로그 조회(필터: actor, entity, 기간)
Eligibility 평가 플로우
Work Type 기본 rule 로드 → 2) 스코프별 override(Project→Site→Permit) merge → 3) 각 assignee 에 대해 required_all/any, training_all/any, min_role_counts, 조건(equipment/capacity) 평가 → 4) enforcement(BLOCK/WARN) 결정 → 5) trace 기록.
캐싱: work_type + rule_version 캐시, override 는 ETag/updated_at 기반 갱신.
성능: 배정 요청 단위 bulk 평가, DB fetch batching.

UI/화면 흐름
관리자 콘솔: Work Type/Trade 코드 편집, rule JSON 편집기+검증기, Override 관리, 감사로그.
HR/Safety 뷰: 자격/교육 등록·검증, 만료 캘린더, 경고 현황.
현장/PM 뷰: 배정 전 Eligibility 체크, 실패 사유/누락 목록, 증빙 업로드, WARN 강행 시 확인 로그.
알림 센터: 만료 D-30/60/90, 검증 지연, BLOCK 실패.
보안·감사
접근 제어: HR_SEC_ADMIN 만 민감 필드/증빙 조회·검증; SAFETY_ADMIN 은 교육/안전 자격 검증; PROJECT/SITE_MANAGER 는 조회+배정 요청.
감사: 조회/등록/변경/검증/강행(WARN override) 모두 audit_log 에 남김.
데이터 보호: 민감 필드 마스킹 저장(또는 컬럼 암호화), 증빙은 presigned URL + 만료시간 제한.
마이그레이션(기존 자원관리 대체)
현 pms/시스템관리/자원관리의 인력/직무/자격 데이터를 People/Employment/Person_Cert/Person_Training 으로 매핑.
공종/직무 카탈로그는 Work Type Catalog 로 이관 후 코드 정규화(TG_/WT_).
기존 배정 로직 호출부를 POST /eligibility/check 로 교체; 호환 기간 동안 WARN 모드 병행 로그 확보.
만료 알림 로직은 Notification 서비스로 이동, D-배치 스케줄러 재연결.
운영·배포 고려
Rule/Override JSON 스키마에 대한 lint/validation 파이프라인 추가(CI).
대량 배치(만료 스캔/알림)는 비동기 작업 큐로 분리.
RBAC 정책은 중앙 Auth 연동(JWT claim→role_tag 매핑).
필요하면 위 설계를 기반으로 상세 ERD·API 스펙 초안이나 마이그레이션 체크리스트를 작성해 줄 수 있어요.
위 설계내용을 다큐멘트에 PMS_자원관리_설계.md 로 파일을 만들고
참고자료는 파일명 변경 : PMS_자원관리_요구사항정의.md
새로 추가되는 내용이 있으면 PMS_자원관리_설계.md 에 업데이트하고 개발을 진행합시다.
새 모듈 설계 내용을 docs/pms/PMS_자원관리_설계.md 에 정리했습니다. 기존 요구사항 참조 파일은 PMS_자원관리_요구사항정의.md 그대로 두고, 설계 파일을 기준으로 이후 변경사항을 추가하면 됩니다.
변경 내용
새 파일 docs/pms/PMS_자원관리_설계.md: People/Qualification/Work Type/Eligibility/Override/Assignment/API/보안/마이그레이션 설계 수록, Trade Group/Work Type 예시와 JSON 스키마 포함.

참고 지침: 추가 요구사항이나 코드 변경 시 이 설계 파일에 섹션 단위로 반영, 요구사항 파일은 레퍼런스로 유지.
다음에 할 일 제안
설계 확정 후 초기 ERD/마이그레이션 스크립트 초안 작성.
Work Type/Rule JSON 스키마에 대한 lint/validation 파이프라인 추가.
/eligibility/check API 스펙 확정 및 기존 배정 호출부 교체 계획 수립.
단계별 진행
단계별 진행 제안해요:
1 단계 설계 확정: docs/pms/PMS_자원관리_설계.md 리뷰/수정, 용어·코드(TG_/WT_) 확정.
2 단계 데이터 모델 구체화: ERD 초안, 컬럼 스펙/제약, 마이그레이션 매핑 표 작성(기존 자원관리 → 새 스키마).
3 단계 스키마/밸리데이션: Rule/Override JSON 스키마 정의, lint/CI 검사 추가, 샘플 카탈로그(seed) 작성.
4 단계 서비스/API 설계 상세: /eligibility/check, Override, People/Cert/Training CRUD, Audit/Alert API 스펙 문서화.
5 단계 구현 1 차: 핵심 도메인(Work Type Catalog + Eligibility Engine + People/Qualification 저장소) 및 기본 CRUD, 배정 체크 API.
6 단계 검증/이행: 단위/통합 테스트, 기존 배정 호출부를 새 API 로 교체(WARN 모드 병행), 알림/스케줄러 연동.
7 단계 운영/보안: RBAC 적용, 감사로그, 증빙 스토리지 정책, 캐시 전략, 모니터링/대시보드 구축.
진행
docs/pms/PMS_자원관리_설계.md 에 “단계별 추진 계획”을 추가했습니다. 현재 파일을 기준으로 단계별로 진행하면 됩니다. 다음 단계로 1 단계 설계 확정(용어·코드 리뷰)부터 시작하시죠.
진행
설계 문서에 1·2 단계 산출물을 바로 반영했습니다.
추가된 내용
용어·코드 정책: TG_/WT_ naming, CERT/TRN 코드, 롤 태그, 상태 값 정의.
데이터 모델 스펙 초안: 주요 테이블별 PK/FK·핵심 컬럼 나열(ERD 작성의 시작점).
다음 진행 제안
1 단계 마무리: 코드/용어 확정 리뷰.
2 단계 심화: 컬럼 타입·제약(UNIQUE/INDEX/NOT NULL) 구체화와 마이그레이션 매핑 표 작성.
현재 고객의 확정하는 단계가 아니라 파일럿시스템을 만들고 설명 후 수정하는 형태로 진행할 계획이므로
최적화된 설계 보다는 일반적 설계를 거쳐 시스템을 개발

그래서 단계별로 진행할때 특이사항 아니면 진행하고 다큐먼트에 수록하고 go
파일럿 접근 원칙을 docs/pms/PMS_자원관리_설계.md 에 추가했습니다:
최적화보다 표준·단순 설계로 1 차 파일럿 구축 후 피드백 반영.
특이사항/리스크가 아니면 바로 개발 진행, 변경·결정 사항은 문서에 기록.
성능 최적화는 파일럿 검증 후 단계적 반영.
이 원칙에 따라 특이사항 없으면 계속 진행하고, 발견되는 변경은 문서에 즉시 업데이트하겠습니다. 다음으로 어떤 부분부터 구현/정리할지 알려주세요(예: ERD 세부화, API 스펙 상세, 초기 시드 작성 등).
진행
파일럿 스코프와 초기 API 스펙을 설계 문서에 추가했습니다.
MVP 범위: People/Employment CRUD, 자격/교육 관리, Work Type 카탈로그, /eligibility/check, Project 단위 Override, 기본 Audit, 만료 알림용 배치 쿼리.
API 초안: people/cert/training CRUD, work-types 관리, eligibility 체크, project override.
데이터 시드 방향: TG/WT 기본 세트, 핵심 CERT/TRN, 고위험 BLOCK·저위험 WARN 기본값.
특이사항 없으면 이 기준으로 개발 진행하며 변경 사항은 문서에 계속 반영하겠습니다. 다음으로 ERD 세부화나 시드 목록 구체화를 시작할까요?
ok
확인했습니다. 이대로 진행하겠습니다. 필요 시 언제든지 말씀 주세요.
진행
파일럿 시드 상세를 docs/pms/PMS_자원관리_설계.md 에 추가했습니다.
Trade Group/Work Type 목록을 기본 2~3 개씩 명시.
자격(CERT)·교육(TRN) 기본 세트 제안.
BLOCK/WARN 기본 가이드 반영.
특이사항 없으면 이 시드를 기준으로 초기 개발을 진행하고, 피드백 나오면 문서에 바로 업데이트하겠습니다. ERD 제약/인덱스 세부화도 곧 이어서 정리할까요?
진행
데이터 모델 제약·인덱스 초안을 docs/pms/PMS_자원관리_설계.md 에 추가했습니다.
파일럿 기준으로 PK/FK, UNIQUE, 주요 인덱스, soft delete 미도입 등을 명시했습니다.
특이사항 없으면 이 버전으로 개발 진행하며, 추후 피드백에 따라 조정하겠습니다.
진행
마이그레이션 매핑 초안과 파일럿 개발 체크리스트를 docs/pms/PMS_자원관리_설계.md 에 추가했습니다.
기존 자원관리 데이터 → 새 스키마(person/employment/qualification/person_cert·training/trade_group/work_type 등) 매핑 가이드.

파일럿 우선 작업: 스키마 생성, 시드(TG/WT/CERT/TRN, 고위험 BLOCK), API 1 차 세트, 룰/override 테스트, audit 로그.
특이사항 없으면 이 체크리스트 기준으로 개발을 이어가겠습니다. 필요하면 스키마 DDL 초안도 바로 추가할게요.
진행
스키마 DDL 초안(포스트그레스 예시)을 docs/pms/PMS_자원관리_설계.md 에 추가했습니다. 파일럿 기준으로 PK/FK, 제약, 인덱스가 포함돼 있어 바로 스키마 생성·적용 테스트에 활용할 수 있습니다. 특이사항 없으면 이 DDL 로 초기 DB 를 올리고 시드/개발을 이어가겠습니다.
진행
DDL 초안에 이어 파일럿 시드 INSERT 예시를 docs/pms/PMS_자원관리_설계.md 에 추가했습니다.
Trade Group/Work Type 기본값
자격(CERT)·교육(TRN) 마스터 기본 세트
고위험 WT 는 BLOCK, 일반 철거는 WARN 예시 포함
이대로 DB 올려서 초기 스키마+시드를 적용하고 개발을 진행하면 됩니다. 추가 조정이 필요하면 알려주세요.
진행
Eligibility 엔진 파일럿 구현 가이드를 설계 문서에 추가했습니다.
입력/단계별 처리, pending 자격 처리, BLOCK/WARN 적용 기준 명시.
min_role_counts·equipment 조건은 1 차 파일럿에서 미적용(추후 확장).
override merge 방식과 pseudo 예시 포함.
이대로 엔진 구현을 시작하고, 필요한 조정은 문서에 계속 반영하겠습니다. 다른 부분 보완이 필요하면 말씀 주세요.
진행
설계 문서에 파일럿 테스트 시나리오(핵심 케이스)까지 추가했습니다.
성공/누락/만료/WARN/Override/pending/다중 assignee 처리 시나리오를 명시했습니다.
이대로 엔진 구현 후 해당 시나리오로 검증 진행하면 됩니다. 추가로 보완할 부분 있으면 알려주세요.
진행
업데이트 완료 상태입니다. 현재 설계 문서에 스키마·시드·엔진 가이드·테스트 시나리오까지 포함되어 있으니 이 기준으로 구현을 시작하면 됩니다. 다음 단계: DB 스키마 생성 및 시드 적용 → /eligibility/check 와 기본 CRUD(API) 1 차 구현 → 시나리오 테스트. 필요한 추가 보완 요청 주시면 바로 반영하겠습니다.
구현진행
구현을 시작하겠습니다. 바로 착수 가능한 순서 제안입니다:

DB 스키마 적용: docs/pms/PMS_자원관리_설계.md 의 PostgreSQL DDL 로 스키마 생성 → 시드 INSERT 반영.
서비스 뼈대: People/Qualification/WorkType/Eligibility 모듈과 /eligibility/check, /eligibility/override, work-type/people/cert/training CRUD 라우트 골격 생성.
Eligibility 엔진: base rule 로드 → override 머지 → assignee 평가 → rule_trace 반환까지 파일럿 규칙(선택/누락/만료/경고) 구현.
테스트: 문서에 정의한 파일럿 시나리오(성공/누락/만료/WARN/override/pending/다중 assignee)로 단위 테스트 추가.
스택/프레임워크(예: Node/Express, Nest, Spring 등)와 DB 접속 정보만 알려주시면 바로 코드 생성에 착수하겠습니다.
구현진행
작업 완료했습니다. 주요 변경사항:
신규 PMS 자원/Eligibility 스키마 및 시드 추가: server/migrations/20251216_pms_resource.sql (확장/테이블/인덱스, Trade Group/Work Type, CERT/TRN 시드).
서버 부팅 시 자동 적용: server/index.js 에 PMS 리소스 마이그레이션 실행 함수 추가, 신규 라우터 등록.
PMS 리소스 API 라우터 신설: server/routes/pms_resource.js
GET /api/pms/resource/work-types : 활성 Work Type 조회
POST /api/pms/resource/people / PATCH /people/:id : 인력 CRUD(단순)
POST /people/:id/certs / /people/:id/trainings : 개인 자격/교육 upsert
POST /eligibility/override : 프로젝트 스코프 override 등록
POST /eligibility/check : 파일럿 Eligibility 엔진(기본 rule + project override merge, 자격/교육 만료/미보유 체크, pending 은 누락 처리)
다음 실행/검증 제안:
서버 재기동 시 자동으로 마이그레이션/시드가 적용됩니다. (Postgres 에 pgcrypto 확장 필요)
기본 시나리오 테스트: 성공/누락/만료/WARN/override/pending/다중 assignee 케이스로 /eligibility/check 호출 검증.
필요 시 추가 스코프(site/permit) override 및 role/min_role_counts 확장 반영.
지금까지 내용을 깃에 올려줘
요청하신 내용 깃에 반영했습니다.
커밋: Add PMS resource schema, seeds, and eligibility API
주요 포함 파일:
docs/pms/PMS_자원관리_설계.md (설계/DDL/시드/엔진 가이드/테스트 시나리오)
docs/pms/PMS_자원관리_요구사항정의.md, docs/pms/PMS_공종별 투입가능 판정_참고.md

서버: server/migrations/20251216_pms_resource.sql, server/routes/pms_resource.js, server/index.js (마이그레이션 적용 및 라우터 등록)
푸시: origin/main 완료 (faf109a5 → 37eb88a8)
참고: ems/node_modules/.package-lock.json, swms/node_modules/.package-lock.json 는 변경되었으나 커밋/푸시하지 않았습니다.
참고에 있는 파일들은 올라가면 안되는 파일인가?
지금 참고용 문서(PMS_자원관리_설계.md, PMS_자원관리_요구사항정의.md, PMS_공종별 투입가능 판정_참고.md)까지 origin/main 에 커밋·푸시한 상태입니다.
만약 외부 공유가 안 되는 파일이라면, 새 커밋에서 해당 파일을 삭제하거나 민감한 부분을 마스킹한 뒤 다시 푸시해야 합니다.
괜찮은 자료라면 추가 조치 없이 그대로 두셔도 됩니다.
필요 시 어떻게 처리할지 알려주시면 바로 작업하겠습니다.
사무실 서 다시 개발을 진행할 예정이라
사무실에서 이어서 하실 때 빠른 세팅 가이드입니다:
git pull origin main 으로 최신 가져오기.
DB 준비: Postgres 에 server/migrations/20251216_pms_resource.sql 적용(서버 재기동 시 자동 실행되지만, 처음엔 수동으로 한 번 확인 권장). pgcrypto 확장 필요.
환경변수
확인: server/.env 의 DB_HOST/DB_USER/DB_PASSWORD/DB_NAME/DB_PORT 맞추기.
서버 구동: npm run server (루트 package.json 의 server 스크립트 실행) → /api/pms/resource/* 엔드포인트 활성화.
프론트(PMS) 구동: npm run pms (루트에서) 또는 cd pms && npm run dev.
기본 동작 점검: /api/pms/resource/work-types 조회, /eligibility/check 파일럿 시나리오 테스트(문서에 있는 성공/누락/만료/WARN/override 케이스).
필요한 부분 있으면 말씀 주세요.
task_20251216.md 위 내용 포함해서 저장해 주고
docs/pms/task_20251216.md 에 오늘 진행 내용과 사무실 재개발 시 세팅 가이드를 정리해 저장했습니다.