QualiJournal 중기 과제 실행 계획서

목적

이 계획서는 퀄리저널 관리자 시스템의 중기 과제(2–4 주)에서 달성해야 할 목표와 세부 작업을 구체적으로 제시합니다. 기본적인 인코딩 오류 수정과 UI/서버 통합 등 단기 과제를 마친 이후, 비동기 처리 도입과 기능 확장, 사용자 경험 개선을 중심으로 추진합니다. 계획에 포함된 각 단계와 산출물은 목표를 잃지 않도록 명확하게 정의되어 있습니다.

목표

- **비동기 워크플로 도입**: 일일·커뮤니티·키워드 플로우를 서버에서 비동기로 실행하여 UI 응답성을 높입니다.
- **실시간 진행 상황 전송**: SSE(Stream-Sent Events)와 폴링을 통해 작업 상태를 프런트에 실시간으로 전송합니다.
- **프런트엔드 개선**: 진행률 모달, KPI 자동 새로고침, 게이트 임계값 슬라이더 등을 추가하여 UX 를 향상합니다.
- **보고서·내보내기 기능 추가**: 일일 보고서, 요약·번역, 마크다운/CSV 내보내기 기능을 API 로 제공합니다.
- **문서화 및 테스트**: 모든 신규 기능을 문서화하고 테스트를 작성하여 품질을 유지합니다.

세부 작업

1. 비동기 API 구현

1. /api/tasks/flow/{kind} (POST)
   - kind 는 daily, community, keyword 중 하나입니다.
   - 요청 시 키워드나 사용 옵션을 포함할 수 있습니다.
   - **동작**: asyncio.create_subprocess_exec 로 오케스트레이터를 비동기 실행하고 고유한 job_id 를 발급합니다. 작업 상태(status, steps, stdout, stderr)를 메모리 또는 Redis 에 저장합니다.
   - **반환**: { "job_id": "..." } JSON
2. /api/tasks/{job_id} (GET)
   - **동작**: 지정된 작업의 현재 상태와 로그를 JSON 으로 반환합니다.
   - **필드**: status(running/done/error/canceled), steps(리스트), started_at, ended_at, messages 등.
3. /api/tasks/{job_id}/stream (GET)
   - **동작**: SSE 를 통해 주기적으로 상태 업데이트를 전송합니다.
   - **구현**: FastAPI 에서 async def event_generator()를 정의하여 `yield f"data: {json.dumps(state)}

"형식으로 전송합니다.Content-Type 은 text/event-stream`입니다.

4. /api/tasks/{job_id}/cancel (POST)

- **동작**: 실행 중인 하위 프로세스를 종료하고 상태를 canceled 로 설정합니다.

5. /api/tasks/recent (GET)
   - **동작**: 최근 N 개의 작업 요약을 반환합니다. UI 에서 “최근 작업” 목록에 사용됩니다.
6. **데이터 구조 및 저장**
   - 초기에 간단한 dict 를 사용하여 메모리에 저장하되, 장기적으로는 Redis 같은 인메모리 저장소로 교체할 수 있도록 인터페이스(TaskManager)를 설계합니다.
   - 작업 상태와 로그는 시간순으로 정렬된 단계(cmd, ok, stdout, stderr, t_start, t_end)를 포함합니다.

2. SSE 스트리밍

- /api/tasks/{job_id}/stream 엔드포인트는 SSE 를 사용하여 프런트엔드에 작업 상태를 실시간 전송합니다. FastAPI 에서는 아래와 같이 구현합니다.

  ```python from fastapi import FastAPI, Request from fastapi.responses import StreamingResponse import json, asyncio

  async def event_generator(job_id): while True: state = task_manager.get_state(job_id) yield f”data: {json.dumps(state)}

” if state.get(“status”) in {“done”, “error”, “canceled”}: break await asyncio.sleep(0.5)

@app.get(“/api/tasks/{job_id}/stream”) async def stream(job_id: str): return StreamingResponse(event_generator(job_id), media_type=“text/event-stream”) ```

- 프런트엔드에서는 EventSource 객체로 스트림을 구독하고, 메시지 수신 시 진행률 모달을 갱신합니다. SSE 가 지원되지 않거나 연결 실패 시에는 /api/tasks/{job_id}를 폴링합니다.

## 3. 프런트엔드 개선

1. **진행률 모달 강화**
   - 각 단계에 아이콘(✅, ⏳, ❌)과 실행 시간(ms)을 표시합니다.
   - Canvas 또는 SVG 로 타임라인 차트를 그립니다.
   - ‘작업 취소’ 버튼과 로그 표시 영역을 유지합니다.
2. **HAS_TASKS 및 폴백 로직**
   - 서버에서 has_tasks 플래그를 제공하면 UI 는 비동기 엔드포인트를 사용하고, 그렇지 않으면 /api/flow/{kind} 동기 엔드포인트로 폴백합니다. 이렇게 하면 404 오류가 발생해도 하나의 버튼으로 플로우를 실행할 수 있습니다 【713135244285786†L38-L67】.
3. **게이트 임계값 슬라이더**
   - 승인 기준(예: 15)을 UI 에서 조정할 수 있도록 슬라이더를 추가합니다.
   - 변경 시 /api/config/gate_required(PATCH) API 를 호출하여 config.json 을 업데이트합니다.
4. **KPI 자동 새로고침**
  ```

- 30 초 간격으로 GET /api/status 를 호출해 '선정본 총량', '선정본 승인', '커뮤니티 후보', '키워드 특집' 등을 갱신합니다.

5. **테마 및 접근성 개선**
   - 라이트/다크 테마를 하나의 HTML 파일로 통합하고, CSS 변수로 색상을 정의합니다. WCAG 2.1 대비 기준을 만족하는 색상 조합을 유지합니다【713135244285786†L10-L18】.
6. **최근 작업 목록**
   - '최근 작업' 버튼을 추가하여 /api/tasks/recent 결과를 모달 또는 로그 영역에 표시합니다.

4. 보고서 및 내보내기 엔드포인트

1. **/api/report** (POST)
   - 파라미터로 날짜와 키워드를 받아 tools/make_daily_report.py 를 호출합니다.
   - 성공 시 archive/reports/{date}_{keyword}.md 경로를 반환합니다.
2. **/api/enrich/keyword** and **/api/enrich/selection** (POST)
   - 선택본과 키워드별 카드에 대해 tools/enrich_cards.py 를 호출하여 요약·번역 작업을 수행합니다.
   - 완료된 파일의 경로를 JSON 으로 반환합니다.
3. **/api/export/md** and **/api/export/csv** (GET)
   - 현재 발행본을 마크다운과 CSV 로 내보냅니다.
   - CSV 는 UTF-8 BOM 을 포함하여 Excel 호환성을 보장합니다.
   - 내보낸 파일 경로를 JSON 으로 반환합니다.
4. **프런트엔드 연계**
   - 보고서·요약·내보내기 기능을 호출하는 버튼을 UI 에 추가하고, 성공 시 결과 파일 다운로드 링크를 표시합니다.

5. 테스트 및 문서화

1. **통합 테스트 작성**
   - Pytest 를 사용해 각 API 엔드포인트의 정상 동작과 에러 처리, 취소 기능을 검증합니다.
   - SSE 스트리밍을 테스트하기 위해 Starlette TestClient 로 스트림을 구독하고 상태 변화를 확인합니다.
2. **회귀 테스트 자동화**
   - tests/d3_regression_check.py 를 CI 파이프라인에 연결하여 도메인·키워드 규칙 변경 시 결과가 기대 범위에 있는지 확인합니다.
3. **문서 업데이트**
   - README.txt 와 README_quickstart.txt 에 비동기 실행 방법, 폴백 로직, 새 엔드포인트, UI 기능, 게이트 조정 방법을 추가합니다.
   - 관리자용 가이드에 승인 임계값 조정과 KPI 확인 절차를 명확히 설명합니다.

6. 일정 (예상 2–4 주)

주차	주요 작업	산출물
1 주차	비동기 API 스켈레톤 구현;	구현된 비동기 엔드포인트,

주차	주요 작업	산출물
	TaskManager 설계; /api/tasks/flow/* 및 /api/tasks/{job_id} 구현; config.json 게이트 설정 API 추가; 문서 초안 작성	TaskManager 모듈, 업데이트된 config API
2 주차	SSE 스트리밍 구현; /api/tasks/{job_id}/stream 추가; 프런트엔드에서 EventSource 통합; 진행률 모달 UI 개선; KPI 자동 새로고침 및 슬라이더 구현	SSE 스트림 동작, 개선된 모달 UI, 자동 새로고침 기능
3 주차	보고서·요약·내보내기 API 구현 및 UI 연동; 최근 작업 목록 기능 추가; 테마 통합 및 접근성 개선	새로운 API 엔드포인트와 UI 버튼, WCAG 준수 테마
4 주차	테스트 케이스 작성 및 회귀 테스트 통합; 문서 완성; 버그 수정 및 코드 정리	Pytest 스위트, 업데이트된 문서, 안정화된 코드

7. 위험 및 대응

- **비동기 처리 오류**: 작업 중간에 예외가 발생해도 상태를 error 로 저장하고 SSE 스트림을 종료해야 합니다. try/except 블록으로 모든 하위 프로세스 호출을 감싸고, 오류 메시지를 로그에 저장합니다.
- **메모리 누수**: 작업 상태를 메모리에만 저장하면 서버 재시작 시 기록이 사라집니다. 추후 Redis 또는 데이터베이스 저장소를 도입하도록 설계 단계에서 인터페이스를 분리합니다.
- **SSE 미지원 브라우저**: SSE 를 지원하지 않는 브라우저에서는 자동으로 폴링으로 전환합니다.
- **보안**: 앞으로 토큰 기반 인증과 HTTPS 적용을 고려해야 합니다. 엔드포인트가 인증 없이 호출되지 않도록 프레임워크의 의존성 주입(Depends)을 사용해 인증 로직을 삽입합니다【713135244285786†L10-L18】.

마무리

중기 과제의 성공은 비동기 처리와 사용자 경험 개선에 달려 있습니다. 위 계획서에 따라 각 기능을 차근차근 구현하고, 문서화와 테스트를 병행하면 향후 고도화 단계와 마이크로서비스 분리에 대비할 수 있습니다. 목표를 명확하게 유지하고 산출물을 검증하는 것이 핵심입니다.