퀄리저널 중기 개발 가이드북 (2025)

1. 계획 수립 현황 평가

중기과제 실행계획과 공식 포함 계획서, 그리고 전달해주신 프레젠테이션을 분석한 결과, 다음과 같은 개선 항목들이 제안되었고 현재 진행 상황은 아래와 같습니다.

계획서 제안	이행 현황	근거
비동기 작업 API 도입 – 기존 `/api/flow/*` 엔드포인트를 비동기화하여 `Flow API`(예: `/api/tasks/flow/daily`)로 작업을 실행하고 `job_id`를 반환한다【871876037888269†L29-L35】.	☒ 구현 완료. `server_quali.py`에 비동기 엔드포인트와 `TaskManager`를 추가했고, `orchestrator.py`는 JSON 형태의 단계별 로그를 출력하도록 수정하였다.	공식 포함 계획서에서는 `/api/tasks/flow/{kind}`, `/api/tasks/{job_id}`, `/api/tasks/{job_id}/stream` 등의 구현을 제안했다【871876037888269†L29-L35】.
SSE 스트리밍 및 폴백 – `StreamingResponse`로 상태를 실시간으로 전송하고, 지원되지 않는 브라우저는 주기적으로 `/api/tasks/{job_id}`를 폴링한다【871876037888269†L50-L53】.	☒ 구현 완료. `server_quali.py`는 SSE 스트림 엔드포인트를 제공하고 UI는 `EventSource`를 구독하며, 실패 시 폴백한다.	계획서는 SSE 스트리밍과 폴백을 명확히 설명하였다【871876037888269†L50-L54】.
프런트엔드 개선 – 진행률 모달 강화, KPI 자동 새로고침, 승인 게이트 슬라이더, 라이트/다크 테마 통합 등【871876037888269†L60-L70】.	☒ 대부분 구현 완료. `index.html`은 각 단계별 아이콘과 실행 시간을 표시하며, 게이트 슬라이더와 KPI 새로고침을 추가하였다.	계획서의 UI 요구사항을 반영해 진행률 모달과 테마를 수정하였다【871876037888269†L60-L70】.
통합 발행 플로우 – `/api/flow/merged` 엔드포인트와 원클릭 발행 버튼으로 공식·커뮤니티·승인·병합을 한 번에 처리한다【871876037888269†L6-L8】.	☒ 구현 완료. 버튼 클릭 한 번으로 일간 뉴스 발행이 가능하며, 자동 승인 기준을 설정 파일로 관리할 수 있다.	계획서에서는 통합 발행 플로우 도입을 장점으로 평가했다【871876037888269†L4-L8】.
품질·팩트 체크 플러그인 구조 – 엔진을 모듈화해 QG/FC/요약/번역을 플러그인으로 교체할 수 있게 한다【516276110013853†L210-L233】.	☒ 기본 구조 구현. `engine_core.py`에 플러그인 매니저를 도입하여 품질 게이트와 팩트체크 함수를 교체할 수 있게 했고, 스레드풀로 기사별 처리를 병렬화했다.	중기계획서에서 플러그인 아키텍처와 병렬 처리의 필요성을 강조했다【516276110013853†L210-L233】.
키워드 역사·트렌드 분석 – 키워드별 과거 발행량과 추세를 분석하는 기능 추가.	☒ 구현 완료. `orchestrator.py`에 `--collect-keyword-history` 플래그와 `keyword_history_analysis`를 추가하여 아카이브 데이터를 분석하고 UI에서 결과를 볼 수 있다.	보고서/트렌드 자료에서 키워드 큐레이션과 역사 분석을 제안했다.
보고서/요약/내보내기 API – `/api/report`, `/api/enrich/*`, `/api/export/md/csv` 등으로 일일 보고서 생성, 카드 요약·번역, 발행본 내보내기 기능을 제공한다【871876037888269†L68-L70】.	⚠☒ 미완료. 아직 API와 UI 버튼이 구현되지 않았다.	계획서는 도구 호출(`make_daily_report.py`, `enrich_cards.py`)과 파일 경로 반환을 요구한다【871876037888269†L68-L70】.
테스트 및 문서화 – Pytest 기반 통합·회귀 테스트, SSE 테스트, README/운영 가이드 업데이트【871876037888269†L85-L93】.	⚠☒ 부분 구현. 일부 로그 출력과 구조화가 이루어졌으나 자동화된 테스트 스위트와 문서 업데이트는 미흡하다.	계획서는 테스트와 문서화 없이는 품질 유지가 어렵다고 강조한다【871876037888269†L85-L93】.
보안 및 인증 – 토큰 기반 인증, HTTPS, CORS 정책 설정【871876037888269†L116-L124】.	⚠☒ 미완료. 현재 모든 엔드포인트가 인증 없이 호출 가능하고 HTTP로 서비스된다.	계획서는 인증과 보안 강화를 중기 과제로 제시한다【871876037888269†L116-L124】.

요약하면, 비동기 처리·SSE 스트리밍·프런트엔드 개선·플러그인 구조·키워드 역사 분석과 같은 핵심 기능은 이미 구현되었다. 그러나 **보고서/내보내기 기능**, **테스트·문서화**, **보안·인증** 등 일부 핵심 항목은 아직 개발되지 않았으며, 국제화·모바일 최적화·모니터링 도구 도입과 같은 장기 과제도 남아 있다.

2. 남아 있는 중기 개발 계획 – 단계별 로드맵

아래 단계는 현재까지 구현된 기능을 바탕으로 남은 중기 과제를 4단계로 나누어 정리한 것이다. 각 단계는 1주 정도의 작업을 예상하며, 병행 가능한 작업은 병렬로 진행해 개발 기간을 단축할 수 있다.

단계 1 – 보고서·요약·내보내기 기능 구현

1. **API 엔드포인트 추가**
   - `/api/report` (POST): 날짜와 키워드를 받아 `tools/make_daily_report.py`를 호출해 Markdown 형태의 일간 보고서를 생성하고 `archive/reports/{date}_{kw}.md` 경로를 반환한다【871876037888269†L68-L70】.
   - `/api/enrich/keyword`, `/api/enrich/selection` (POST): 선택본 또는 키워드 카드에 대해 `tools/enrich_cards.py`를 호출하여 기사 요약과 번역을 수행한다【516276110013853†L210-L233】.
   - `/api/export/md`, `/api/export/csv` (GET): 현재 발행본을 Markdown과 CSV(BOM 포함)로 내보내고 파일 경로를 반환한다【516276110013853†L330-L378】.

  - 생성된 보고서 파일의 구조와 내용(헤드라인, 기사 요약, 승인 여부, KPI)을 검토하여 템플릿을 개선한다.
  - Markdown → PDF 변환 스크립트(`md_to_pdf.py`)를 도입해 자동 출력할 수 있게 한다.

단계 2 – 자동화 테스트 및 모니터링 도구 도입

1. **통합·회귀 테스트 작성**
  - `pytest`를 사용하여 비동기 엔드포인트의 정상/에러 처리, 취소 기능을 검증한다【871876037888269†L85-L91】.
  - `Starlette TestClient`로 SSE 스트리밍을 구독하고 상태 변화를 검증한다【871876037888269†L85-L93】.
  - `tests/d3_regression_check.py`를 CI 파이프라인에 통합하여 도메인·키워드 규칙 변경 시 기대 범위를 확인한다【871876037888269†L87-L91】.

2. **문서 업데이트**
  - `README` 및 운영자 가이드에 비동기 사용법, 폴백 로직, 새 API 설명, 승인 임계값 조정 방법을 추가한다【871876037888269†L85-L93】.
  - 각 기능에 대한 사용 예제와 흔히 발생하는 오류 해결 방법을 정리한다.

3. **모니터링 도구 도입**
  - `Prometheus`와 `Grafana`를 연동하여 API 응답 시간, 작업 실패율, 승인률, 큐 길이 등을 시각화한다【871876037888269†L92-L93】.
  - 알림 규칙을 설정하여 비정상적인 지표 변화가 감지되면 Slack/Email로 알림을 보내도록 한다.

4. **로그 구조화 및 회전 설정**
  - Python `logging` 모듈로 JSON 형식 로그를 작성하고, 심각도별 로그 레벨을 정의한다【871876037888269†L114-L116】.
  - `logrotate` 설정으로 로그 파일 크기에 따라 회전하도록 구성한다.

단계 3 – 보안·국제화·모바일 개선

1. **인증과 권한 관리**
  - FastAPI의 `Depends`를 활용하여 모든 API에 **토큰 기반 인증**을 적용한다【871876037888269†L116-L124】. 관리자는 승인·발행 권한을 가지며, 일반 사용자와 게스트는 조회만 가능하도록 권한을 분리한다.
  - OAuth2 또는 JWT 기반 인증 방식을 적용하고, 토큰 발급·검증 로직을 구현한다.

2. **HTTPS 및 CORS**
  - 개발·스테이징·운영 환경에 대한 HTTPS 인증서를 구성하고 Nginx/Traefik 등 리버스 프록시에서 TLS를 종료한다【871876037888269†L119-L124】.
  - `Access-Control-Allow-Origin` 헤더를 적절히 설정하여 허용된 도메인에서만 API를 호출할 수 있도록 한다【871876037888269†L121-L123】.

3. **국제화(I18N) 지원**
  - UI 텍스트를 JSON 리소스 파일로 분리하고, 날짜·숫자 포맷을 로캘에 따라 변환하는 함수를 작성한다【871876037888269†L71-L72】.
  - 관리자 페이지에 언어 선택 메뉴를 추가하고 영어·한국어를 우선 지원한다.

4. **모바일/접근성 개선**
  - 관리자 UI를 반응형으로 설계하여 태블릿과 휴대폰에서도 사용할 수 있도록 미디어 쿼리를 적용한다.
  - WCAG 2.1 대비 기준을 만족하는 색상 대비를 유지하고, 키보드 네비게이션과 스크린리더 호환성을 검증한다【871876037888269†L66-L67】.

단계 4 – 장기적 확장과 최적화

1. **상태 저장소 확장**
  - 현재는 메모리 기반 `TaskManager`를 사용하지만, 작업이 늘어나면 상태가 서버 재시작 시 사라지는 문제가 있다. Redis나 PostgreSQL 등 외부 저장소를 사용하도록 `TaskManager` 인터페이스를 확장한다【871876037888269†L34-L36】.
  - 장기적으로는 Celery/RQ와 같은 작업 큐를 도입하여 대규모 비동기 작업을 분산 처리한다【871876037888269†L125-L126】.

2. **AI 요약 및 번역 자동화**
  - ChatGPT API나 공개 모델을 활용하여 기사 요약·번역을 자동화하고 인간 검수를 최소화한다【871876037888269†L169-L170】.
  - 플러그인 방식으로 요약·번역 엔진을 교체할 수 있도록 `engine_core`의 플러그인 매니저를 확장한다【516276110013853†L210-L233】.

3. **마이크로서비스/모듈 통합 검토**
  - 현재 두 수집기(`engine_core.py`와 `orchestrator.py`)를 하나의 모듈로 통합하거나, 반대로 수집기별로 마이크로서비스를 분리하여 서버를 경량화하는 방안을 검토한다【871876037888269†L167-L173】.
  - 서비스가 성장하면 Kubernetes 등의 오케스트레이션 도구를 적용해 확장성과 안정성을 확보한다【871876037888269†L125-L126】.

4. **키워드 트렌드 및 히스토리 고도화**
  - 현재 구현된 키워드 역사 분석을 확장하여, 외부 데이터(표준 변경, 특허 트렌드 등)를 분석하고 가중치와 추천 알고리즘에 반영한다.
  - `keyword_synonyms.json`과 `editor_rules.json`을 주기적으로 업데이트하고, 기사 메타데이터에 원본 소스 URL·수집 시간·품질/팩트/도메인 점수를 저장하여 회귀 테스트와 큐레이션 품질을 향상한다【871876037888269†L83-L84】.

3. 성공적인 개발을 위한 팁과 권장 사항

- **점진적 구현과 리뷰**: 각 단계가 완성될 때마다 코드 리뷰와 사용자 피드백을 통해 개선점을 반영한다. 작은 단위로 배포하면 문제를 빠르게 발견하고 수정할 수 있다.
- **표준화된 로그와 모니터링**: JSON 구조의 로그를 사용하고 Prometheus/Grafana를 통해 지표를 지속적으로 확인한다. 이상 징후를 조기에 발견하고 해결할 수 있다【871876037888269†L114-L116】.
- **테스트 주도 개발**: 신규 API를 작성할 때 테스트를 함께 작성하면 회귀를 예방하고 안정성을 높일 수 있다. SSE와 비동기 흐름은 Starlette TestClient로 테스트한다【871876037888269†L85-L93】.
- **보안과 개인정보 보호**: 초기에는 개발 편의를 위해 인증을 생략하더라도, 운영 환경에서는 반드시 토큰 기반 인증과 HTTPS를 적용한다. API 키와 토큰은 `.env` 또는 안전한 비밀 저장소에 저장한다【871876037888269†L116-L124】.
- **협업과 문서화**: 개발팀과 운영자가 동일한 가이드와 계획을 공유할 수 있도록 README와 관리자 가이드를 지속적으로 업데이트한다. 회의에서 변경 사항과 일정 조정을 논의한다.
- **사용자 경험 우선**: 비동기 처리와 실시간 스트리밍은 관리자와 사용자의 생산성을 크게 높인다. 진행률 모달, KPI 새로고침, 테마 통합 등 UI 개선을 통해 긍정적 경험을 제공해야 한다【871876037888269†L60-L70】.

4. 맺음말

중기 개발은 이미 상당 부분 진행되었으며, 남은 과제들은 보고서·내보내기 기능과 테스트·보안 강화, 문서화에 집중되어 있다. 위 로드맵을 따라 단계별로 기능을 구현하고, 지속적인 테스트와 문서화를 병행한다면 쿼리저널 시스템은 향후 고도화와 마이크로서비스 분리에 대비할 수 있을 것이다.

향후 과제까지 염두에 두고 플러그인 구조와 스케일링 전략을 마련함으로써, 기사 큐레이션 서비스의 신뢰성과 확장성을 지속적으로 높여가길 바란다.