ChatGPT

QualiJournal 관리자 시스템 운영 가이드북

1. 작업 구분 (완료된 작업 vs. 남은 작업)

완료된 작업 목록: 현재 관리자 시스템에 구현되어 있는 주요 개선 작업들은 다음과 같습니다.

- **게이트 임계값 슬라이더 구현:** 관리자 UI 상단에 **게이트 임계값(GATE_REQUIRED)**을 조절할 수 있는 슬라이더를 추가하였고, 슬라이더 값 변경 시 백엔드의 `/api/config/gate_required` 엔드포인트에 PATCH 요청을 보내도록 구현되었습니다 [1]. 이를 통해 편집자는 **최소 승인 기사 수 등의 임계값**을 실시간으로 조정할 수 있습니다.
- **KPI 지표 자동 새로고침:** 관리자 대시보드의 KPI 통계(수집된 기사 수, 승인된 기사 수 등)를 일정 주기마다 자동 갱신하는 기능이 추가되었습니다 [2]. UI에서 **30초 간격으로** `/api/status`를 호출하도록 타이머를 설정하여, 별도로 "새로고침" 버튼을 누르지 않아도 최신 KPI 수치가 반영되도록 개선되었습니다.
- **관리자 인증 토큰 적용: ADMIN 토큰 기반 인증 미들웨어**가 서버에 도입되어 **모든 Admin API 요청에 토큰 확인이 필요**하도록 보안이 강화되었습니다. 토큰 값은 환경변수(`ADMIN_TOKEN`)로 설정하여 운영하며, 클라이언트는 API 호출 시 헤더에 `Authorization: Bearer <ADMIN_TOKEN>`를 포함해야 합니다 [3]. 서버는 요청 헤더의 토큰을 사전에 설정된 토큰과 비교하여 다를 경우 401 Unauthorized를 반환합니다. 이로써 **외부 비인가 접근을 차단**하고 관리자용 API만 보호합니다.
- **UI 테마 및 기타 개선:** 단기 과제로 제시되었던 **다크/라이트 모드 테마 통합**, **폴백 동작**(신규 비동기 API 미구현 시 구 버전 API로 처리), **텍스트 인코딩 오류 수정** 등의 작업이 이전 단계에서 완료되어 적용되었습니다 [4]. 예를 들어, 과거 API와의 **호환성 폴백**이 구현되어 일부 기능이 비동기로 동작하지 않을 경우 기존 동기 방식으로 자동 처리되도록 하였고, CSV 내보내기 시 **BOM(Byte Order Mark)**을 추가하는 등의 인코딩 문제 해결도 이루어졌습니다.
- **기타 완료 사항:** 관리자 시스템 전반의 코드 리팩토링 및 안정화 작업이 수행되어, **프로젝트 구조 개편**, **불필요한 코드 제거**, **성능 개선** 등이 이뤄졌습니다. 또한 **기사 아카이브(archive) 관리** 로직이 구축되어, 매일 발행되는 데이터가 `archive/` 폴더에 날짜별로 저장되고 있습니다.

미완료된 작업 목록: 아직 구현이 완료되지 않았거나 추가 개발이 필요한 기능들은 아래와 같이 정리됩니다. (괄호 안에 세부 항목으로 구분)

- **일일 보고서 생성 기능** – **백엔드 API** 및 **프론트엔드 UI** 미완료. 현재 백엔드에 일부 `GET /api/report` 엔드포인트만 존재할 뿐, **일간 보고서 생성**을 트리거하는 `POST /api/report` 엔드포인트와 해당 UI 버튼은 구현되지 않은 상태입니다 [5] [6]. 다시 말해 `tools/make_daily_report.py` 스크립트를 호출하여 Markdown 보고서를 생성하는 핵심 로직은 백엔드에 일부 구현되었으나, **클라이언트에서 이를 실행하거나 결과를 열람하는 UI가 없다**는 의미입니다.
- **뉴스 카드 요약·번역 기능** – **백엔드 API** 및 **UI 기능** 모두 미구현. **모든 수집 기사**에 대한 요약/번역 (`POST /api/enrich/keyword`)과 **최종 선정본 기사**에 대한 요약/번역 (`POST /api/enrich/selection`) 엔드포인트가 현재 서버에 없습니다 [6]. 해당 기능을 수행하는 `tools/enrich_cards.py` 스크립트는 존재하지만, 이를 호출하는 API와 UI 버튼이 마련되어 있지 않아 **요약/번역 작업을 수동으로 실행하거나 UI 상에서 트리거할 방법이 없는 상황**입니다 [7]. (예: **"키워드 요약/번역"**, **"선정본 요약/번역"** 버튼 미구현)
- **결과물 Export UI 연동** – **Markdown/CSV 내보내기 UI** 미완료. 서버에는 `GET /api/export/md`, `GET /api/export/csv` 엔드포인트가 존재하여 최종 발행본을 제공하도록 일부 구현되어 있으나, **프론트엔드에 해당 Export 버튼이나 메뉴가 없어** 실제 사용이 불가능합니다 [8]. 즉, **다운로드용 API는 있으나 이를 눌러 파일을 받는 UI 연결**이 누락된 상태입니다.
- **통합 테스트 및 문서화** – **테스트 코드 작성**과 **README/매뉴얼 갱신 작업**이 계획되었으나 아직 완비되지 않았습니다. 신규 기능들(보고서 생성, 요약, 내보내기 등)에 대한 **Pytest 기반 통합 테스트 스크립트 작성**이 필요하

며, **README**에는 비동기 처리 방식, 폴백 동작, 게이트 임계값 설정법, 환경 설정 등 최신 변경사항을 반영해야 합니다 [9] [10]. 현재 tests/ 디렉토리가 마련되어 있긴 하나, **모든 시나리오에 대한 테스트 케이스 및 문서 업데이트**는 향후 마무리되어야 할 과제로 남아있습니다.

- **추가 예정 및 장기 과제:** 이 외에도 일부 미구현 또는 향후 개선 예정인 기능들이 있습니다. **비동기 태스크 처리 엔드포인트** (예: `/api/tasks/...`)와 **실시간 진행률 표시** 기능은 아직 본격 도입되지 않았습니다. 현재는 모든 작업이 동기적으로 처리되며, UI에서는 단순 로딩 표시만 하고 있습니다. 또한 **DB 연동** 및 **관리자 다중 권한 관리**, **AI 요약 고도화**, **키워드 추천 알고리즘** 등의 기능은 장기 과제로 제시되었으나 아직 구현되지 않았습니다 [11]. 이러한 부분들은 현 시스템 안정화 후 **로드맵**에 따라 추후 단계적으로 추가 검토 및 개발할 사항들입니다.

2. 실행 중심 가이드 (실행 절차 및 설정)

미구현된 기능들을 포함하여, 관리자 시스템의 주요 기능들을 **운영/개발자가 실행**할 수 있도록 각 항목별 절차를 안내합니다. **Python 스크립트 경로**, **API 사용법**, **환경변수 설정**, **배포 방법** 등을 단계별로 다룹니다.

2.1 일일 보고서 생성 기능 실행

개요: 일일 보고서 생성은 편집자가 선정하여 승인한 당일 기사들을 모아 **Markdown 형식의 데일리 리포트**를 만들어 주는 기능입니다. 이 기능은 백엔드의 `POST /api/report` 엔드포인트로 제공되며, 내부적으로 `tools/make_daily_report.py` 스크립트를 실행합니다 [12]. 스크립트는 현재까지 승인된 기사 목록(`selected_articles.json`)을 취합해 템플릿에 따라 Markdown 파일을 생성합니다 [13].

- **백엔드 실행 절차:** 서버를 구동한 상태에서, `POST /api/report`를 호출하면 백엔드가 `make_daily_report.py`를 실행하여 `archive/reports/YYYY-MM-DD_report.md` 파일을 생성합니다 [13]. 파일명에는 당일 날짜와 "_report.md"가 포함되며, 키워드가 있다면 파일명에 키워드도 들어갑니다 (예: `2025-10-11_AI_report.md`) [13]. 생성 완료 후 **응답은 JSON** 형태로 반환되며, `ok` 필드에 성공 여부, `path` 필드에 생성된 파일 경로를 담습니다 [14] [15]. 예시 응답:

```
{
  "ok": true,
  "path": "archive/reports/2025-10-11_AI_report.md"
}
```

만약 보고서 생성 중 오류가 발생하면, 서버는 HTTP 200을 반환하되 `ok: false`와 `error` 메시지를 포함한 JSON을 응답합니다 [14]. (예: `"error": "NameError: 'XYZ' is not defined in make_daily_report.py"` 형태로 스크립트 에러 내용을 전달)

- **클라이언트(UI) 실행 절차:** 프론트엔드에는 **"일일 보고서 생성"** 버튼(또는 메뉴)을 추가해야 합니다 [16]. 사용자가 이 버튼을 클릭하면 자바스크립트를 통해 위 **POST API**를 호출합니다. **Authorization 헤더**에 `Bearer <ADMIN_TOKEN>`을 포함하는 것을 잊지 말아야 합니다 [17]. UI 코드 예시:

```
const res = await fetch('/api/report', {
  method: 'POST',
  headers: { 'Authorization': 'Bearer <토큰값>' }
});
const data = await res.json();
if (res.ok && data.path) {
  console.log("Report created:", data.path);
```

```
    // TODO: 화면에 "보고서 열기" 링크 표시 등 결과 처리
} else {
    alert('보고서 생성 실패: ' + (data.error || 'Unknown error'));
}
```

17 18

호출이 성공하면 응답 JSON의 `path`를 이용해 **"보고서 열기"** 링크나 다운로드 버튼을 UI에 표시합니다 19 . 예컨대, 미리 숨겨둔 `<a id="reportLink" href="" target="_blank">보고서 열기</a>` 태그에 `href`를 응답 경로로 설정하고 보이도록 하면 사용자가 클릭하여 Markdown 보고서를 열람할 수 있습니다 19 .

- **직접 스크립트 실행 (개발/운영용):** 서버 API를 통하지 않고 바로 리포트 생성이 필요할 경우, 관리자 서버 프로젝트 경로에서 다음처럼 **직접 Python 스크립트를 실행**할 수도 있습니다:

  ```
  $ source venv/bin/activate  # 가상환경 활성화 (예시)
  $ export ADMIN_TOKEN=<토큰값> # 환경변수 설정 (토큰 필요시)
  $ python tools/make_daily_report.py
  ```

  실행하면 현재 `selected_articles.json`을 기반으로 Markdown 파일을 생성합니다. CLI 실행의 경우 별도 출력은 없을 수 있으므로, `archive/reports/` 디렉토리에 새로운 파일이 생겼는지 확인합니다. **오류 발생 시** 터미널에 Traceback이 표시되므로 해당 에러를 토대로 코드를 수정하거나 데이터를 정비합니다 (예: JSON 포맷 오류 등은 8.2 절의 스크립트로 해결 가능 20).

- **REST API 호출 예시 (curl):** Cloud Run에 배포된 서비스에서 리포트를 생성하려면, Cloud Run 서비스 URL과 인증 헤더를 포함하여 curl로 요청할 수 있습니다. 예:

  ```
  curl -X POST "https://<CloudRun 서비스 URL>/api/report" \
    -H "Authorization: Bearer ${ADMIN_TOKEN}"
  ```

  정상 실행 시 위에서 설명한 JSON 응답이 반환되며, 에러 시 `ok: false`와 `error` 필드가 포함됩니다. 반환된 `path` 경로는 Cloud Run 컨테이너 내 경로이므로, **보고서 확인을 위해서는 별도의 UI 링크를 통해 열람하거나 Cloud Run에 저장된 파일을 다운로드**해야 합니다 (Cloud Run 컨테이너는 무상태(stateless)이므로 파일이 영구 저장되지는 않으나, session 내에서는 접근 가능).

2.2 뉴스 카드 요약 및 번역 기능 실행

개요: 뉴스 카드 요약/번역 기능은 **AI 요약 모델**을 사용해 수집된 기사들의 핵심 요약문(영어)을 생성하고, 필요에 따라 **한글 번역본**까지 자동으로 만들어주는 기능입니다. **키워드 전체 기사 대상**(`POST /api/enrich/keyword`)과 **선정된 기사 대상**(`POST /api/enrich/selection`) 두 가지 엔드포인트로 제공되며, 내부적으로 모두 `tools/enrich_cards.py` 스크립트를 호출합니다 21 .

- **백엔드 실행 절차:** 예를 들어 `POST /api/enrich/keyword`를 호출하면, 백엔드가 `enrich_cards.py`를 실행하여 `selected_keyword_articles.json` **내의 모든 기사**에 대해 요약을 수행합니다 22 . 기본 동작은 각 기사에 **영문 요약문(summary)**을 생성하고, 환경변수에 **번역 API 키(API_KEY)**가 설정되어 있으면 해당 요약을 **한국어로 번역**하여 `selected_keyword_articles.json`의 필드에 추가합니다 23 . 스크립트 실행 후, 백엔드는 업데이트된 JSON 데이터를 사용해 **모든 기사 카드의 제목, 원문 링크, 영문 요약, 국문 요약, 코멘트**를 포함한 **Markdown 파일**을 생성합니다 24 . 생성 파일명은

`archive/enriched/YYYY-MM-DD_<키워드>_ALL.md` 형식이며, 해당 키워드의 전체 기사 요약본을 담습니다 [25]. 응답 JSON은 보고서 생성과 동일한 구조로, `ok`와 `path`를 포함합니다 [26].

`POST /api/enrich/selection`의 동작도 유사하나 대상이 **승인된 기사들**로 한정됩니다 [27]. 선택된 기사들의 요약/번역을 수행한 뒤 `archive/enriched/YYYY-MM-DD_<키워드>_SELECTED.md` 파일을 만들고, JSON 응답에 경로를 반환합니다 [28]. 두 엔드포인트 모두 **스크립트 오류와 무관하게 Markdown 결과물을 생성하도록 설계**되었는데, 이는 만약 요약중 일부 기사에서 실패하더라도 그때까지 생성된 내용으로 파일을 만들기 위함입니다 [29] [30]. 실패 시에도 `ok: false`와 `error` 메시지를 함께 응답하지만, Markdown 파일은 (비록 일부 내용이 빠졌을 수 있으나) 남도록 **idempotent**하게 동작합니다 [29].

- **클라이언트(UI) 실행 절차:** 프론트엔드에 **"뉴스 요약/번역"** 버튼들을 추가합니다 [16]. 예를 들어 **"키워드 뉴스 요약"** 버튼은 `/api/enrich/keyword` POST 요청을 트리거하고, **"선정본 요약"** 버튼은 `/api/enrich/selection` 요청을 트리거하도록 구현합니다. 두 경우 모두 **Authorization 헤더에 토큰**을 포함해야 하며, 응답으로 돌아온 JSON의 `path`를 이용해 **"요약본 다운로드"** 링크를 제공하거나 새 창으로 결과 Markdown을 열람할 수 있게 합니다 [19]. UI 처리 방식은 보고서 생성의 경우와 비슷합니다. 즉, 호출 전 버튼 비활성화 및 로딩 표시 -> 완료 시 링크 표시 -> 실패 시 오류 경고를 하는 패턴을 적용합니다 [31] [32].

UI 예시 (선정본 요약 버튼):

```
const res = await fetch('/api/enrich/selection', {
  method: 'POST',
  headers: { 'Authorization': 'Bearer <토큰>' }
});
const result = await res.json();
if (res.ok && result.path) {
  document.getElementById('selSummaryLink').href = result.path;
  document.getElementById('selSummaryLink').style.display = 'inline';
} else {
  console.error(result.error || "Enrich failed");
  alert('요약 생성 실패: ' + (result.error || '원인 불명'));
}
```

주의: Markdown 결과물은 서버상의 파일 경로이므로, 브라우저에서 직접 열기 위해서는 해당 경로를 정적으로 서빙하거나, **별도의 API로 파일을 전송**하도록 해야 합니다. 간단한 방법으로, **요약/번역 API 호출 응답 시 파일 콘텐츠를 바로 보내도록** 구현할 수도 있지만 (현재는 경로만 넘김), 현 구조에서는 경로를 받고 **별도 GET 요청으로 파일을 받는** 형태가 더 안전합니다. 필요하다면 `/api/getfile?path=<...>` 같은 엔드포인트를 추가 구현하여 `archive/` 폴더의 파일을 다운로드시키는 방식도 고려할 수 있습니다 (보안상 해당 파일에 대한 접근도 ADMIN 토큰으로 제한 권장).

- **Python 스크립트 수동 실행:** 운영 중 긴급히 요약/번역을 다시 실행해야 할 경우, 서버를 거치지 않고 `tools/enrich_cards.py`를 직접 실행할 수 있습니다. 예시:

  ```
  $ python tools/enrich_cards.py --mode=keyword  # 키워드 전체 기사 요약 실행
  $ python tools/enrich_cards.py --mode=selection # 선정본(승인기사) 요약 실행
  ```

  (`--mode` 인자는 구현에 따라 다를 수 있습니다. 없는 경우 스크립트 내부 default로 키워드 전체 처리가 될 수 있음)
  이 스크립트는 실행 시 콘솔에 진행 상황이나 결과를 로그로 출력하도록 개발하는 것이 좋습니다. 만약

API_KEY 환경변수가 설정되지 않은 상태에서 실행하면, 영문 요약만 수행되고 번역은 생략됩니다 [33]. 번역까지 필요하다면 `.env` 파일이나 실행 환경에 `API_KEY`를 지정해야 합니다.

- **REST API 호출 예시 (curl):** Cloud Run 서비스에서 키워드 전체 기사 요약을 수행:

```
curl -X POST "https://<서비스 URL>/api/enrich/keyword" \
  -H "Authorization: Bearer ${ADMIN_TOKEN}"
```

응답 예:

```
{ "ok": true, "path": "archive/enriched/2025-10-11_AI_ALL.md" }
```

마찬가지로 이 경로의 파일에 최종 요약본이 저장되어 있으므로, 필요하면 추가로 해당 파일을 GET으로 다운로드 받는 절차가 필요합니다. (Cloud Run 컨테이너는 로컬스토리지를 영구 보존하지 않으므로, 생성 직후 다운로드하거나 외부 스토리지로 옮기는 것을 권장)

2.3 결과물 Export (Markdown/CSV) 기능 실행

개요: Export 기능은 현재 최종 선정된 기사 목록(final selection)을 **Markdown 파일**이나 **CSV 파일**로 다운로드하는 기능입니다. 백엔드에 `GET /api/export/md` (Markdown 출력)와 `GET /api/export/csv` (CSV 출력) 두 엔드포인트가 구현되어 있습니다 [34] [35]. 이들은 데이터베이스가 아닌 현재까지 **발행 준비된 JSON 데이터** (selected_articles.json 등에 저장된 최종 목록)를 기반으로 결과를 생성합니다.

- **백엔드 동작:** `/api/export/md`는 내부적으로 편집된 기사 목록을 Markdown 템플릿으로 직렬화하여 **파일 다운로드 응답**을 제공합니다 [36]. `/api/export/csv`는 기사 목록을 CSV 형식으로 변환하여 반환하며, **CSV 첫 바이트에 UTF-8 BOM (\ufeff)**을 추가하여 Excel에서 열 때 한글 인코딩 문제가 없도록 처리합니다 [37]. 두 엔드포인트 모두 **HTTP 응답으로 파일 내용을 직접 보내는 형태**로 설계되며, `Content-Disposition` 헤더를 통해 적절한 파일이름이 지정됩니다 (예: `attachment; filename="QualiJournal_20251011.csv"`).

- **클라이언트(UI) 연동:** 프론트엔드에 **"Export MD"**, **"Export CSV"** 버튼을 추가하고 해당 버튼 클릭 시 위 GET API를 호출하여 파일을 다운로드 받습니다 [38]. 구현 방법은 두 가지가 있습니다:

- **\<a> 태그 활용:** 버튼을 실제 `<a href="/api/export/csv" download>` 형태의 링크로 만들어 두고, 클릭 시 브라우저가 바로 파일을 받도록 합니다 [39]. 이 경우 **인증 토큰**이 헤더에 포함되지 않는 문제가 있으므로, 만약 Export 엔드포인트에도 토큰 검증이 걸려 있다면 동작하지 않습니다 [40]. (토큰을 쿼리스트링에 붙이는 방법은 보안상 권장되지 않습니다.)
- **JavaScript Fetch 활용:** `<button>` 클릭 이벤트에서 `fetch('/api/export/csv')`를 호출하여 응답 Blob 데이터를 수신한 후, JS로 임시 `<a>` 태그를 만들어 `URL.createObjectURL(blob)`을 사용해 다운로드를 트리거합니다 [41] [42]. 이 방법을 쓰면 **헤더에 Authorization 토큰을 포함**할 수 있어 보안에 맞게 구현할 수 있습니다.

구현 예시 (Fetch 사용):

```
async function exportCsv() {
  const res = await fetch('/api/export/csv', {
    headers: { Authorization: 'Bearer ' + ADMIN_TOKEN }
```

```
    });
    if (!res.ok) {
      alert('CSV export failed');
      return;
    }
    const blob = await res.blob();
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'QualiJournal_Today.csv';
    document.body.appendChild(a);
    a.click();
    a.remove();
}
```

43 42

위 예시처럼 구현하면 토큰 인증이 필요한 API도 안전하게 호출하여 파일 다운로드를 진행할 수 있습니다. 참고로, **Markdown 출력의 경우**에도 동일한 방식으로 `/api/export/md`를 호출해 `.md` 파일을 다운로드 받을 수 있습니다. Export 버튼을 누른 후에는 사용자가 받은 파일을 실제로 열어 **한글 깨짐 현상이 없는지** 등도 확인해야 합니다 44. (현재 CSV 응답에 BOM을 포함시키므로 Windows Excel에서 바로 열 때 인코딩이 올바르게 적용됩니다 44.)

- **REST API 호출 예시 (curl):** 토큰을 포함하여 CSV를 다운로드:

  ```
  curl -H "Authorization: Bearer ${ADMIN_TOKEN}" \
    -o "latest.csv" "https://<서비스 URL>/api/export/csv"
  ```

  위 명령은 `latest.csv` 이름으로 CSV 파일을 저장합니다. (Cloud Run 배포 시 ADMIN_TOKEN 검증이 활성화되어 있으므로 반드시 헤더에 토큰을 포함해야 함에 유의합니다.)

2.4 환경변수 설정 및 .env 구성

관리자 시스템의 여러 구성 값들은 **환경 변수(Environment Variables)**로 관리됩니다. 프로젝트 루트에 예시로 `.env` **파일**을 두고 사용하며, 실제 운영 환경에서는 이 값을 Cloud Run 등에 주입하여 사용합니다 3 45. 주요 환경변수와 설정 방법:

- **ADMIN_TOKEN:** 앞서 설명한 **관리자 인증 토큰**으로, **충분히 복잡한 랜덤 문자열**로 설정합니다 (예: `ADMIN_TOKEN=aklJE92nv2...`) 3. `.env` 파일 또는 Cloud Run 콘솔의 환경변수 설정에 추가하며, **절대로 Git 등에 노출되지 않도록** 합니다. FastAPI 서버는 기동 시 `os.getenv("ADMIN_TOKEN")`으로 이 값을 읽어 전역 변수로 저장하고, 모든 API 요청에서 인증 미들웨어가 사용합니다 3. 토큰을 변경한 경우 프론트엔드에서 사용하는 값도 반드시 업데이트해야 합니다 46.
- **API_KEY:** 뉴스 요약 결과를 **번역**하기 위한 외부 API 키입니다. 예를 들어 **Papago API**나 **Google Cloud Translate API** 키를 사용한다면 이 값을 설정합니다. `.env`에 `API_KEY=<your_translation_api_key>` 형식으로 넣고, 서버 실행 시 `enrich_cards.py`가 해당 키를 읽어 번역 기능을 수행합니다 33. 키가 없거나 값이 설정되지 않은 경우, **요약은 영문으로만 진행되고 번역은 생략**되도록 구현되어 있습니다 33.
- **DB_URL: 데이터베이스 연결 문자열**로, 만약 JSON 파일 저장 대신 DB를 사용하기로 전환할 경우 여기에 DB 접속 정보를 넣습니다 (예: `DB_URL=postgresql://user:pass@host:5432/dbname`) 47. 현재 시스템은 기본적으로 JSON 파일에 기사를 저장/관리하지만, 향후 DB로 마이그레이션하기 위한 준비로 코드에

DB 연결 로직이 일부 포함되어 있습니다. 운영 환경에서 DB를 사용한다면 이 값을 설정하고, Cloud Run 등에서 해당 DB에 접근 가능하도록 네트워크/VPC 및 권한 설정을 해야 합니다 [48]. DB_URL 설정 후 서버를 실행하면 **로그에 DB 연결 시도 메세지**가 뜨거나 `/api/status` 호출 시 DB 기반 통계를 반환하는 등 동작 변화가 있으니, 배포 시 이를 확인해야 합니다 [49].

예시 **.env 파일** (로컬 개발용):

```
ADMIN_TOKEN="your-admin-token-12345"
API_KEY="your-papago-or-google-api-key"
DB_URL="postgresql://postgres:Password123@localhost:5432/qualidb"
```

위 `.env` 파일은 `.gitignore`에 포함시켜 절대로 원격 저장소에 커밋하지 않습니다 [45].

- **로컬 환경 적용:** 로컬에서 서버를 실행할 때는 `.env` 파일이 자동으로 로드되도록 `.env` 지원 패키지를 사용하거나, 수동으로 `source .env` 명령으로 환경변수를 세팅한 뒤 `uvicorn` 등을 실행합니다.
- **Cloud Run 환경 변수 설정:** Cloud Run에 컨테이너를 배포할 때는 GCP 콘솔의 Variables & Secrets 설정 또는 GitHub Actions CI에서 secrets 주입을 통해 동일한 값들이 설정되어야 합니다 [50] [51]. 예를 들어 Cloud Run의 **Variables** 섹션에 `ADMIN_TOKEN`, `API_KEY`, `DB_URL` 키를 추가하고 값을 입력합니다 [50]. GitHub Actions를 사용하는 경우 repository secrets에 해당 키들을 등록해두고, 배포 Workflow에서 이를 환경변수로 전달합니다 (GitHub Actions 예시는 아래 Cloud Run 배포 절차에서 설명).
- **환경 설정 확인:** 배포 후 실제 컨테이너에서 **환경변수가 정상 적용되었는지** 확인해야 합니다. Cloud Run 콘솔의 환경변수 탭을 열어 변수들이 제대로 설정되었는지 확인하고 [52], 혹시 값이 누락되거나 잘못되진 않았는지 점검합니다. 또한 어플리케이션 로그나 `/api/status` 응답 등을 통해 필요한 기능이 활성화되었는지도 살펴봅니다. (예: API_KEY가 없는 상태로 enrich 요청 시 번역이 생략됨을 로그로 확인)

2.5 Cloud Run 배포 및 CI/CD 자동화

QualiJournal 관리자 서버는 **Docker 컨테이너**로 패키징되어 Google **Cloud Run**에서 운영됩니다 [53]. 초기에는 수동으로 도커 이미지 빌드/배포를 했으나, 현재는 **GitHub Actions 기반 CI/CD 파이프라인**을 구축하여 코드 변경 시 자동으로 빌드/배포가 이뤄지도록 개선되었습니다 [54]. 아래는 Cloud Run 배포 절차와 CI/CD 구성 요약입니다.

- **Docker 이미지 빌드:** 프로젝트에는 미리 설정된 `Dockerfile`이 존재하며, 해당 이미지를 빌드하면 FastAPI 서버와 관련 툴이 포함됩니다 [55]. 로컬에서 수동으로 빌드하려면:

  ```
  $ docker build -t asia.gcr.io/<GCP_PROJECT_ID>/qualijournal-admin:latest .
  $ docker push asia.gcr.io/<GCP_PROJECT_ID>/qualijournal-admin:latest
  ```

  와 같이 이미지를 빌드 후 Google Artifact Registry 등에 푸시할 수 있습니다. (※ 개발 단계에서만 수동 빌드를 사용하고, 실제 운영은 CI/CD에서 이 과정을 자동화합니다 [56].)

- **Cloud Run 서비스 설정:** GCP 콘솔에서 Cloud Run 서비스를 생성할 때, 위에서 빌드한 Docker 이미지 경로와 CPU/메모리, 영역(region) 등을 지정합니다. 예를 들어 **서비스 이름**을 `qualijournal-admin`으로 하고, **Region**은 `asia-northeast3` (서울)로 설정합니다. 동시요청 처리(concurrency)나 auto-scaling 최대/최소 인스턴스 수 등은 트래픽 패턴에 맞게 조정합니다. **환경변수**는 앞서 언급한 ADMIN_TOKEN, API_KEY 등을 추가합니다 [50]. 배포 후 서비스에는 `https://qualijournal-admin-<random-id>-`

`<region>.run.app` 형태의 기본 URL이 부여되며, 이 URL을 통해 API에 접근합니다 [57]. (사용자 도메인을 연결하여 커스텀 도메인으로 운영할 수도 있습니다.)

- **GitHub Actions CI/CD 파이프라인:** CI 설정 파일 `.github/workflows/ci.yml` (또는 `deploy.yml`)에서 **main 브랜치에 대한 push 이벤트**를 트리거로 워크플로우가 실행되도록 구성합니다 [58] [59]. 주요 단계는 다음과 같습니다:

- **GCP 인증 세팅:** Actions에서 GCP에 접속하기 위해 **서비스 계정 키** JSON을 시크릿으로 등록하고, 워크플로우에서 `google-github-actions/setup-gcloud` 액션을 사용해 인증합니다 [60] [61]. 예를 들어 `secrets.GCP_SA_KEY`에 서비스계정 키(JSON)를 저장하고, `secrets.GCP_PROJECT`에 GCP 프로젝트 ID를 저장해 사용합니다.
- **도커 이미지 빌드 및 푸시:** `gcloud builds submit` 명령 또는 Docker CLI를 이용해 Artifact Registry에 이미지를 빌드/푸시하는 step을 추가합니다 [62]. 예시:

```
- name: Build and Push
  run: |
    gcloud builds submit --tag asia.gcr.io/$GCP_PROJECT/qualijournal-admin:${GITHUB_SHA}
```

위 명령은 현재 커밋의 해시(${GITHUB_SHA})를 태그로 하여 이미지를 빌드/등록합니다 [62].

- **Cloud Run 배포:** Google이 제공하는 Action인 `google-github-actions/deploy-cloudrun@v1`을 사용하여, 빌드된 이미지로 Cloud Run을 업데이트합니다 [63] [64]. 필요한 인자로 **이미지 경로**, **서비스 이름**, **프로젝트 ID**, **리전(region)** 등을 전달합니다. 또한 이 단계에서 **환경변수(env_vars)**를 함께 설정할 수 있습니다 [65]. 예시 YAML 스니펫:

```
- uses: google-github-actions/deploy-cloudrun@v1
  with:
    image: asia.gcr.io/${{ secrets.GCP_PROJECT }}/qualijournal-admin:${{ github.sha }}
    service: qualijournal-admin
    project_id: ${{ secrets.GCP_PROJECT }}
    region: asia-northeast3
    env_vars: >
      ADMIN_TOKEN=${{ secrets.ADMIN_TOKEN }},
      API_KEY=${{ secrets.API_KEY }},
      DB_URL=${{ secrets.DB_URL }}
```

[66]

위와 같이 **GitHub Secrets**에 저장된 ADMIN_TOKEN, API_KEY, DB_URL 값을 배포 시 환경변수로 주입할 수 있습니다. (만약 Cloud Run 콘솔에서 이미 환경변수를 설정했다면 `env_vars`는 생략 가능합니다 [67].)

- **테스트 단계 추가(선택):** 배포 전에 **Pytest 테스트를 자동 실행**하도록 job을 추가하면 좋습니다 [68]. `pytest -q` 명령을 실행하고, 만약 한 개라도 테스트가 실패하면 배포 job을 중단시켜 **문제 있는 코드가 배포되지 않도록** 막을 수 있습니다 [69] [70]. (구체적인 테스트 작성에 대해서는 다음 절에서 다룹니다.)

- **Slack 알림 등 (선택):** Workflow 마지막에 배포 성공 시 Slack이나 이메일로 알림을 보내는 단계도 추가 가능합니다. 이때 Cloud Run 서비스 URL이나 배포된 커밋 정보를 포함하면 운영자가 바로 확인하기 편리합니다 [71].

- **CI/CD 동작 확인:** 설정 후 실제로 main 브랜치에 코드를 push하여 **워크플로우가 정상 동작**하는지 검증합니다. GitHub **Actions 로그**에서 도커 이미지 빌드 및 Cloud Run 배포 성공 메시지를 확인하고, 오류가 없는지 살핍니다 [72]. 배포가 완료되면 **Cloud Run 서비스 URL**이 출력되도록 워크플로우를 구성할 수 있는데, 해당 URL로 **curl이나 브라우저**를 통해 접근하여 응답이 오는지 검사합니다 [73]. 특히 `/api/status` 엔드포인트를 호출했을 때 정상적으로 KPI 지표 JSON이 반환되는지 확인하면 서비스가 잘 작동했는지 알 수 있습니다 [74].

- **CI/CD 보안 유의사항:** CI 파이프라인에서 사용하는 **GCP 서비스 계정 키, ADMIN_TOKEN 등 비밀값은 GitHub Secrets**에 저장하여 참조하고, 워크플로우 YAML 내에 노출되지 않도록 주의해야 합니다 [75]. 만약 워크플로우 실행 로그에 민감 정보가 찍히지 않도록 `set -x` 비활성화 등도 고려합니다. 또한 Cloud Run 배포 시 **Google Cloud Secrets Manager**를 활용하면 환경변수를 CI에서 넘기지 않고 Cloud Run이 시크릿을 참조하도록 설정할 수도 있습니다 (민감정보 유출 위험 감소).

3. 체크리스트 기반 운영 지침

시스템 운영자는 아래 **체크리스트**를 토대로 일상적인 점검을 수행하고, 배포 이후나 예외 상황 발생 시 대응할 수 있습니다. 각 기능별 확인 항목과 **배포 후 검증 사항**, **예외 상황별 테스트 및 UI 반응** 확인 절차를 정리합니다.

3.1 일상 점검 체크리스트 (데일리 운영)

- **데일리 발행 전 기사 수 확인:** 매일 새로운 키워드 뉴스를 발행하기 전에 **현재 승인된 기사 수(selection_approved)**가 **최소 요구 개수(gate_required)**를 충족하는지 확인합니다 [76]. 이는 `/api/status` 응답의 `selection_approved`와 `gate_required` 값을 비교하거나, UI 대시보드에서 해당 수치를 확인함으로써 가능합니다 [76]. 기본 임계값은 15개이며, 부족할 경우 발행을 강행하지 말고 **추가 기사를 승인**하거나 임계값을 일시 조정해야 합니다.
- **품질 게이트 통과 여부:** `/api/status`의 `gate_pass` 값이 `false`이면 품질 게이트에 미달된 상태이므로 **발행을 보류**하는 것이 원칙입니다 [77]. 부득이하게 발행이 필요하면 **임계값을 현재 승인 기사 수 이하로 낮추는 방안**을 취합니다 [78]. 예를 들어 승인된 기사가 10개인데 gate_required=15이면, `/api/config/gate_required`에 PATCH 요청으로 값을 10으로 낮춰 `gate_pass: true`로 만든 후 진행합니다 [78]. (사후에 반드시 gate_required 값을 원상 복구할 것)
- **소스 수집 상태 점검:** 해당 날짜의 **뉴스 수집 작업**이 제대로 이루어졌는지 확인합니다. 예를 들어, `selected_keyword_articles.json` 파일의 타임스탬프가 최신인지, 또는 `/api/status`의 `keyword_total` (수집된 기사 총수)이 평소 평균 수준인지 살펴봅니다 [79]. 만약 특정 뉴스 소스에서 기사 수가 **비정상적으로 0개**인 경우 크롤러 오류나 소스 설정 문제를 의심하고 조치합니다.
- **UI 대시보드 KPI 동기화 확인:** 관리자 UI에 표시되는 **KPI 지표들**(누적 수집, 승인 건수, 공식/커뮤니티 기사 비율 등)이 백엔드 `/api/status` 값과 일치하는지 확인합니다 [80]. **자동 새로고침 기능**이 있다면 주기적으로 값이 갱신되는지, 없다면 발행 직전에 수동 새로고침을 해서 최신 상태로 만든 후 진행합니다 [80]. (예: 30초 자동 갱신이 동작 중이라면 별도 조치 불필요, 수동 새로고침 버튼이 있는 경우 눌러서 최신화)
- **Export 데이터 백업:** 운영 편의 및 데이터 보존을 위해 주기적으로 **CSV/Markdown Export**를 수행해 파일을 백업합니다 [81]. 예를 들어, 매일 발행 후 **"Export CSV"** 버튼을 눌러 해당 CSV를 로컬 PC나 구글 드라이브 등에 저장해두면, 추후 통계나 분석에 활용할 수 있습니다 [81]. (이때 CSV 파일을 열어 한글이 깨지지 않는지 확인하여 인코딩 이슈가 없음을 재확인합니다 [82].)
- **아카이브 용량 관리:** 서버 컨테이너 내부의 `archive/` 폴더에 일별로 Markdown 리포트와 enriched 요약본 등이 계속 쌓이므로, **디스크 용량**을 모니터링해야 합니다 [83]. Cloud Run 자체는 스테이트리스라 컨테이너 재시작 시 파일이 초기화될 수 있지만, 만약 NAS나 외부 스토리지를 마운트하여 archive를 유지하는 설정이라면 용량이 초과되지 않도록 관리합니다 [83]. 오래된 파일은 주기적으로 압축하여 백업하거나 외부 스토리지로 이전한 뒤 삭제하는 정책을 세웁니다.
- **Cloud Run 리소스 모니터링:** GCP Cloud Run의 **모니터링 탭**에서 CPU 사용률, 메모리 사용량, 요청 성공률 및 지연 시간 등을 정기적으로 확인합니다 [84]. 메모리 부족으로 인한 **OOM**이나 CPU과부하로 인한 **스로틀링** 징후가 발견되면 Cloud Run 서비스의 **메모리/CPU 한도를 상향 조정**을 검토합니다 [84]. 또한 **동시 처리**

(concurrency) 설정과 **자동 스케일링** 동향도 관찰하여, 너무 빈번한 scale-out이 발생하면 최대 인스턴스 수를 제한하거나 코드 최적화를 고려합니다 [84].

3.2 배포 후 확인 및 주요 체크포인트

새 버전을 배포한 직후나 설정 변경 후에는 다음 항목들을 확인하여 **정상 반영 여부**를 점검합니다:

- **CI/CD 배포 로그 확인:** GitHub Actions에서 **CI 파이프라인 결과**를 확인합니다. **이미지 빌드 및 푸시 단계**, **Cloud Run 배포 단계**가 성공적으로 완료됐는지 로그를 검토하고, 에러가 없는지 확인합니다 [72]. 만약 Actions 단계 중 실패가 있다면 해당 단계의 로그를 열어 원인을 파악하고 재시도 또는 수정합니다.
- **Cloud Run 서비스 상태:** Cloud Run 콘솔에서 새 Revision이 배포된 것을 확인하고, **Revision 상태가 Active(활성)**인지 봅니다. **서비스 URL**로 접속하여 응답을 체크합니다 (브라우저에서 기본 URL 열기 또는 `curl <서비스URL>/api/status` 호출) [74]. 응답으로 status JSON이 제대로 오거나, 최소한 HTTP 200 응답이 오면 컨테이너가 정상 실행 중임을 의미합니다. **만약 응답이 없거나 502/503 에러**가 발생하면 배포된 컨테이너에 문제가 있는 것이므로 Cloud Run **Logs**에서 에러 메시지를 확인합니다 [85].
- **환경변수 적용 확인:** Cloud Run **Variables & Secrets** 설정에 입력한 환경변수들이 **정확히 반영**되었는지 확인합니다 [50] [52]. 예를 들어 ADMIN_TOKEN, API_KEY 값이 제대로 설정되었는지, 철자가 틀리거나 누락된 값은 없는지 살핍니다. 환경변수가 잘못되면 기능 일부가 동작하지 않으므로 (`ADMIN_TOKEN`이 없으면 모든 API가 401 반환 등), 필요한 경우 수정 후 재배포합니다.
- **기능 동작 검증:** 배포된 새 버전에서 **주요 기능을 한 바퀴 테스트**합니다. 예를 들어, **기사 수집 -> 승인 -> 보고서 생성 -> 요약 생성 -> Export**까지의 과정을 짧게 시뮬레이션해 봅니다 [86]. 간단히는 `/api/status`로 수집/승인 건수 확인 -> (필요시 임의로 selected_articles.json 채워넣기) -> `/api/report` 호출 -> `/api/export/csv` 호출 등으로 E2E 흐름을 점검해볼 수 있습니다 [86]. UI 상에서도 버튼들을 한 번씩 눌러보며 **버튼 비활성화/로딩/성공/실패 시 UI 반응**이 올바른지 확인합니다.
- **토큰 인증 확인: Authorization 토큰 없이 API 호출 시 차단되는지** 확인합니다. 예를 들어 배포 후 `curl -X POST https://.../api/report` (토큰 미첨부)로 호출하면 401 Unauthorized가 오는지 테스트하고, 올바른 토큰을 첨부하면 200 OK가 오는지 비교합니다 [87]. 이를 통해 **토큰 인증 미들웨어가 제대로 작동**하는지 재확인합니다. 만약 401 대신 200이 나온다면 토큰 검증이 누락된 치명적 보안 버그이므로 즉시 수정해야 합니다.
- **오류 로그 모니터링:** 배포 직후부터 Cloud Run **로그 스트림**을 실시간 모니터링하며 **예기치 않은 오류(traceback)**나 경고가 출력되지 않는지 살펴봅니다 [88]. 특히 새로 추가한 기능 경로(`/api/report`, `/api/enrich/...` 등) 호출 시 에러 로그가 없는지 확인합니다. 오류가 발견되면 해당 리비전을 **일시 롤백**(이전 버전을 100% 트래픽으로)하고 문제를 해결한 뒤 재배포합니다 [85].

3.3 예외 상황별 대응 및 테스트 지침

운영 중 발생할 수 있는 다양한 예외 상황에 대비하여, 다음 대응 방법과 테스트 절차를 숙지합니다.

- **토큰 인증 오류 발생:** 모든 API 호출이 401 또는 403 에러를 반환한다면 **ADMIN_TOKEN 불일치**를 의심합니다. 이 경우 현재 UI/스크립트에서 사용하는 토큰 값이 `.env` 또는 Cloud Run 설정의 값과 같은지 확인합니다 [46]. 최근에 토큰을 변경하고 프론트엔드에 반영하지 않았다면 새로운 토큰으로 업데이트해야 합니다. 또한 Cloud Run 로그에 401 에러 기록이 반복된다면 **외부에서 토큰 없이 접속 시도가 있지는 않은지** 살펴보고, 보안상 필요 시 토큰을 교체하는 것도 고려합니다 [89].
- **API 실패 (보고서/요약 생성 오류):** 사용자 측면에서 **"실패" 알림**이 뜨거나 응답 JSON에 `ok: false`가 온 경우, 우선 **응답의** `error` **필드**를 확인합니다 [90]. 예를 들어 `"NameError: 'XYZ' is not defined in make_daily_report.py"` 라는 에러 메시지가 담겨 있다면 해당 스크립트 코드에 버그가 있는 것이므로 신속히 코드를 수정해야 합니다 [91]. 요약 기능의 경우 `"Translation API quota exceeded"`와 같은 메시지가 올 수 있으므로, 이때는 **외부 번역 API의 사용량/쿼터 상태**를 점검하고 필요 시 키를 교체하거나 쿼터를 상향합니다 [92]. 한편, **부분 실패 시에도**

Markdown 결과 파일은 일부 생성되었을 수 있으므로, 서버에 접속해 `archive/reports/` 또는 `archive/enriched/` 폴더를 살펴보고 생성된 파일이 있다면 내용을 보완하여 활용할 수도 있습니다 [93].

- **데이터 불일치 또는 JSON 무결성 문제:** 편집 도중 수동으로 JSON을 편집하다가 포맷이 깨지거나 (예: 따옴표나 쉼표 누락) 데이터 불일치가 발생할 수 있습니다. 이때 일부 스크립트가 KeyError 등 오류를 낼 수 있는데, **복구 스크립트**를 활용합니다. 프로젝트의 `tools/repair_selection_files.py` 스크립트를 실행하면 `selected_articles.json` 등 주요 JSON의 포맷을 교정할 수 있고, 이어서 `tools/sync_selected_for_publish.py`를 실행하여 필요한 JSON을 재생성합니다 [20] [94]. 이러한 응급처치용 스크립트들의 사용법 (`python tools/repair_selection_files.py` 등)은 숙지해 두는 것이 좋습니다 [94] [95]. 운영 tip: 중요 JSON은 변경 전에 백업본을 남겨두고, 문제가 해결되면 가능한 한 근본적인 버그(왜 깨졌는지)를 개발 측면에서 수정합니다.
- **품질 게이트 미충족 긴급 발행:** 당일 뉴스를 발행해야 하는데 승인된 기사가 부족하여 `gate_pass=false`인 경우가 있습니다. 이럴 땐 **두 가지 대안**이 있습니다: (1) 앞서 언급한 대로 `/api/config/gate_required`로 임계값을 낮춰 **강제로 gate_pass를 true**로 만든다 [78]. (2) 또는 **공식 기사 풀 보충 스크립트**(예: `tools/augment_from_official_pool.py`)가 있다면 이를 실행해 부족한 공식 기사를 빠르게 추가 수집/승인하는 방법입니다 [96]. 첫 번째 방법은 편의상 즉각적이지만 사후에 임계값을 원복해야 함을 잊지 말고, 두 번째 방법은 시간이 들지만 기사 수를 실제로 채워 품질 기준을 충족시키는 정석입니다. 상황에 따라 선택하되, **발행 품질에 영향을 주지 않는 범위**에서 판단합니다.
- **Cloud Run 장애 및 롤백:** 새 버전 배포 후 **API 응답이 비정상**이거나 주요 기능 호출시 오류가 난다면, 즉시 Cloud Run **로그에서 오류 원인**을 찾습니다 [85]. 흔한 원인으로 **모듈 누락**(ModuleNotFound) 오류가 있어 기능이 죽는 경우가 있는데, 이는 `requirements.txt`에 해당 모듈을 추가하고 재배포해야 합니다 [85]. 만약 긴급히 이전 상태로 돌아가야 하면 Cloud Run 콘솔에서 **지난 Revision으로 트래픽을 롤백**하거나 `gcloud run services rollback` 명령어를 사용해 이전 버전을 100%로 돌립니다 [85]. 배포 파이프라인 자체에 문제가 발생하여 배포가 중단된 경우(예: GCP 권한 오류), GitHub Actions 로그를 보고 **인증/권한 설정을 재점검**한 후 재시도합니다 [85].
- **데이터베이스 관련 이슈:** 현재는 DB를 적극 사용하고 있지 않지만, 만약 `.env`에 `DB_URL`을 설정하고 서버에서 DB 초기화 코드가 실행된 경우 **DB 연결 오류**가 발생할 수 있습니다 [97]. 이때는 우선 `.env`의 DB_URL이 올바른지, DB 인스턴스가 외부에서 접속 가능하게 설정되어 있는지(VPC 및 방화벽 규칙) 확인합니다 [48]. 연결은 되지만 마이그레이션 오류나 쿼리 실패가 있다면, Cloud Run 로그와 DB 자체의 로그를 대조하여 문제를 파악합니다 [97]. **DB 마이그레이션 작업**은 서비스 중지 또는 트래픽 차단 후 진행하는 것을 권장하며, JSON에서 DB로 데이터를 옮기는 경우 충분한 테스트와 백업 후 수행합니다 [97].

4. 테스트 및 문서화

새로 구현되거나 변경된 기능들에 대해서는 반드시 **테스트 케이스 작성**과 **문서 업데이트**를 진행해야 합니다. 이는 시스템의 안정성과 **회귀 방지**를 위해 중요합니다.

4.1 Pytest 기반 통합 테스트 예시

관리자 시스템은 FastAPI로 구현되어 있으므로, **FastAPI의** `TestClient`를 활용하여 주요 API를 호출해보는 통합 테스트를 작성할 수 있습니다 [98]. 모든 테스트 코드는 프로젝트의 `tests/` 디렉토리에 모아 관리하며, pytest 명령으로 실행됩니다 [55].

테스트 시나리오 예시:

- **보고서 생성 API 테스트:** `/api/report` 호출 시 Markdown 보고서 파일이 생성되고, 응답 JSON에 해당 파일 경로(`path`)가 포함되는지 확인합니다 [99]. 또한 ADMIN 토큰이 없는 상태로 호출하면 401/403을 반환하는지, 올바른 토큰을 주면 200을 반환하는지 테스트하여 **인증 로직**을 검증합니다 [87].
- **요약 생성 API 테스트:** `/api/enrich/keyword` 및 `/api/enrich/selection` 호출 시 **요약 결과 Markdown 파일이 생성**되는지를 확인합니다. 예를 들어 임의로 몇 개의 기사 데이터를 fixture로 넣어둔 후

API를 호출하고, 응답의 `path`에 해당하는 파일 내용을 검사합니다 (각 기사 제목, 영문 요약이 포함됐는지 등).

- **Export API 테스트:** `/api/export/csv` 응답으로 받은 텍스트에 **CSV 헤더 행이나 기사 제목 등이 포함**되어 있는지 확인합니다. 또한 **응답 시작이** `\ufeff` **(BOM)**로 시작하는지 체크하여 Excel 호환성을 테스트합니다 [100]. Markdown 출력도 유사하게 주요 Markdown 문법 (예: 리스트 `- [ ]` 표기 등)과 콘텐츠가 포함됐는지 검사합니다.
- **인증 및 권한 테스트:** ADMIN 토큰 미제공 시 **모든 보호된 엔드포인트가 접근 거부**되는지 케이스별로 확인합니다. 아래는 **토큰 인증 테스트**의 예시입니다:

```
def test_auth_token_required():
    res = client.post("/api/report")  # 토큰 없이 호출
    assert res.status_code in (401, 403)
    # 올바른 토큰으로 호출 시 200 응답
    res2 = client.post("/api/report", headers={"Authorization": "Bearer test-token"})
    assert res2.status_code == 200
}
```

[87]

위 테스트에서 `test-token`은 가상의 값으로, 테스트 시작 시 `os.environ['ADMIN_TOKEN'] = "test-token"` 식으로 환경변수를 주입하거나 TestClient를 생성할 때 해당 설정을 반영합니다 [101]. 이를 통해 실제 서버와 동일한 인증 흐름을 재현합니다.

- **게이트 임계값 조정 테스트:** (고급) `/api/config/gate_required`로 임계값을 변경했을 때 `/api/status`의 `gate_pass` 값이 올바르게 바뀌는지 확인합니다. 예를 들어, 현재 승인된 기사 수를 10개로 세팅한 뒤 gate_required를 15에서 10으로 PATCH 요청하여 **gate_pass가 False에서 True로 바뀌는지** 확인하는 식입니다 [102] [103]. 이 테스트는 monkeypatch 등을 활용해 서버의 내부 상태를 가정하고 시나리오를 구성합니다.

위와 같은 테스트 케이스들을 작성하여 **pytest를 실행**, 모든 테스트가 통과하는지 확인합니다 [104]. 테스트 파일들은 가능하면 독립적으로 실행 가능하도록 하고, 공통 준비 로직은 fixture로 분리하여 작성합니다. 특히 **신규 기능 관련 테스트(보고서 생성, 요약, 내보내기 등)**뿐 아니라 **기존 기능(기사 크롤링, 승인 처리 등)의 회귀 테스트**도 포함하여 최근 코드 변경이 다른 부분에 영향을 주지 않았음을 보장해야 합니다 [104].

추가로, **coverage 툴**(coverage.py 등)을 사용하여 테스트 커버리지를 확인하는 것을 권장합니다 [105]. 예를 들어 보고서 생성 함수, 요약 생성 함수 등이 테스트에서 한 번 이상 호출되었는지 커버리지 리포트를 통해 점검하고, 빠진 부분이 있으면 테스트를 보완합니다 [105].

마지막으로, 가능하다면 **엔드투엔드(E2E) 시나리오 테스트**도 고려합니다 [106]. 예를 들어: 가상의 기사 20개 수집 -> 15개 승인 -> `/api/report` 호출 -> `/api/export/csv` 호출까지 일련의 과정을 자동화된 테스트로 시뮬레이션하여 **전체 파이프라인**이 문제없이 동작하는지 검증합니다 [106]. 이는 실행 시간이 오래 걸릴 수 있으므로 필수는 아니지만, 주기적으로 또는 주요 변경 시에 수행하면 안정성 확보에 도움이 됩니다.

4.2 문서화 및 매뉴얼 업데이트

모든 기능 구현과 테스트가 완료된 후에는 **프로젝트 문서(README 등)**를 최신 상태로 업데이트해야 합니다 [10] . 운영자와 개발자가 시스템을 이해하고 사용할 수 있도록 아래 항목들을 문서에 반영합니다:

- **신규 기능 사용법:** 이번에 추가된 **보고서 생성, 요약/번역, Export 기능**의 사용 방법을 README에 상세히 기술합니다 [10] . 각 API 엔드포인트와 대응되는 UI 조작 방법, 예시 스크린샷 등을 포함하면 좋습니다. (예: "일일 보고서 생성 버튼을 누르면 `/api/report`를 호출하여 ... 결과는 archive/reports 폴더에 저장되고 UI에 링크가 나타남")
- **비동기 처리 및 폴백 설명:** 설계 문서에 언급된 **비동기 태스크 엔드포인트(`/api/tasks/...`)**가 현재 구현되어 있지 않고, 시스템이 **동기 처리**로 동작하는 부분에 대한 설명을 추가합니다 [10] . 예를 들어, "현재 버전에서는 모든 작업이 동기적으로 처리되며, 추후 비동기 엔드포인트가 구현되면 자동으로 해당 방식을 사용할 수 있도록 폴백 구조가 마련돼 있습니다" 등의 내용을 기술합니다. 폴백 동작의 내부 로직이나 설정(`USE_ASYNC_TASK=False` 등 플래그가 있다면)도 함께 문서화합니다 [10] .
- **게이트 임계값 및 KPI 새로고침 안내: 게이트 임계값 조정 방법**(관리자 UI 슬라이더 사용 및 config.json 직접 편집 방법)과 **KPI 자동 갱신 기능**에 대해 README에 안내합니다 [107] . 운영자가 임계값의 의미를 이해하고 필요시 조정하는 방법, KPI 지표가 자동으로 업데이트됨을 알고 수동 새로고침이 불필요함 등을 명시합니다.
- **환경 설정 및 배포:** README에 **환경변수(.env) 설정 예시**, **Cloud Run 배포 방법 요약** 등을 추가합니다 [108] . 특히 ADMIN_TOKEN 설정 방법, API_KEY 준비 방법, DB_URL 사용 시 주의사항 등을 bullet point로 정리합니다. 또한 GitHub Actions를 통한 CI/CD 파이프라인 구성 및 배포 절차도 간략히 소개하여, 새로운 개발자가 배포 과정을 쉽게 따라할 수 있도록 합니다.
- **프로젝트 구조 및 의존성:** 최근에 변경된 **폴더 구조**(예: `admin/` 폴더 신설 여부, `tools/` 폴더 내용 등)와 **requirements.txt 의존 패키지 목록**을 업데이트합니다 [108] . 불필요한 패키지를 제거하고 추가된 패키지(pytz, coverage 등 예시)가 있다면 반영합니다.
- **운영 매뉴얼 링크 및 요약:** 본 운영 가이드북의 주요 체크리스트와 대응 방법을 README 또는 별도의 운영 매뉴얼 문서에 요약하여 첨부합니다. 예를 들어 "운영 체크리스트" 섹션을 표 형태로 정리하거나, 3장에서 다룬 예외 대응 방법을 Q&A 형태로 재구성하여 쉽게 찾을 수 있도록 합니다.
- **기타 참고 자료:** API 명세 표나 주요 기능별 시연 GIF, 또는 성능 튜닝 팁 등이 있으면 추가합니다.

문서화의 궁극적인 목표는 **개발자 및 운영자 누구나 이 시스템의 작동과 개선 사항을 이해하고, 문제 발생 시 이 가이드를 따라 해결할 수 있도록 하는 것**입니다 [109] . 따라서 문서를 최신으로 유지하고, 새로운 기능 추가 시 이 가이드북과 README를 함께 업데이트하는 프로세스를 갖추는 것이 바람직합니다.

以上의 내용에 따라 QualiJournal 관리자 시스템의 운영 및 유지보수에 필요한 가이드 작업을 완료합니다. 이 가이드북을 참고하여 순차적으로 운영 절차를 수행하면 무리 없이 시스템을 관리할 수 있을 것이며, 향후 기능 추가나 변경 시에도 본 문서를 기반으로 점진적으로 확장해 나갈 수 있을 것입니다 [109] . 성공적인 운영을 기원합니다! [109]

Sources:

1. QualiJournal 관리자 시스템 최종 가이드북 [110] [111] [22] [25] [29] [15] [31] [19] [32] [46] [90] [78] [85] [76] [80] [83] [84] [3] [33] [50] [52] [112] [66] [87] [102] [104]

2. QualiJournal 관리자 시스템 개선 작업 인수인계 보고서 (1010_1) [5] [113] [7] [8] [1] [2] [9] [6] [4] [16]

---

[1] [2] [4] [5] [6] [7] [8] [9] [10] [16] [37] [38] [107] [108] [113] 1010_1_나머지 계획_퀄리저널 관리자 시스템 개선 작업 인수인계 보고서.pdf
file://file-XBYbxoGjLn4pSrN6QHLGAd

3 11 12 13 14 15 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 39 40 41
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 109 110 111 112 1011_3QualiJournal 관리자 시스템 최종 가이드북.pdf
file://file-6Hu18vVrjQweSb5SZz6a6M