ChatGPT

퀄리저널(QualiJournal) 관리자 시스템 – 개발자 및 운영자 가이드북

본 가이드북은 퀄리저널 관리자(Admin) 시스템의 **개발자 가이드**와 **운영자 가이드**로 구성되어 있습니다. 개발자는 새로운 기능(API 및 UI)을 구현하고 배포하는 절차를 확인할 수 있고, 운영자는 관리자 UI를 통해 일일 뉴스 발행 작업을 수행하는 방법을 익힐 수 있습니다. 각 섹션에서는 환경 설정부터 API 사용 방법, Cloud Run 배포, 테스트/CI, UI 사용법, 버튼별 기능 등을 자세히 설명합니다.

개발자 가이드

환경 설정 및 .env 구성

퀄리저널 Admin 서버는 중요 설정값들을 환경변수로 관리합니다. **프로젝트 루트에** `.env` **파일**을 생성하여 다음과 같은 키들을 정의하세요 [1]:

```
# .env 예시 (민감정보는 예시값 대신 실제값으로 대체)
ADMIN_TOKEN="your-admin-token-12345"   # 관리자 인증 토큰 (임의의 복잡한 문자열)
API_KEY="sk-XXXXXXXXXXXXXXXXXXXXXXXX"  # 외부 요약/번역 API 키 (예: OpenAI 또는 Papago)
DB_URL="postgresql://user:pass@host:5432/dbname"  # (선택) DB 연결 URL (JSON 대신 DB 사용 시)
NAVER_CLIENT_ID="..."              # (선택) Papago API Client ID (번역에 Papago 사용 시)
NAVER_CLIENT_SECRET="..."          # (선택) Papago API Secret
```

- **ADMIN_TOKEN:** Admin API 호출에 필요한 비밀 토큰입니다. 추측이 어렵도록 **복잡한 난수 문자열**로 생성하는 것이 좋습니다 [2]. 클라이언트(JS) 코드나 브라우저에서 이 값이 드러날 수 있으므로 절대 노출되지 않게 주의하세요.
- **API_KEY:** OpenAI, Papago 등 **외부 요약/번역 API 키**를 설정합니다 [3]. 예를 들어 OpenAI를 사용할 경우 해당 발급 키를 입력하고, 코드에서 `openai.api_key = os.getenv("API_KEY")`와 같이 참조합니다. Papago API를 사용한다면 Client ID/Secret 쌍을 환경변수로 추가하고 API 호출 시 헤더에 포함하세요 [4].
- **DB_URL:** 현재 시스템은 기본적으로 **JSON 파일**로 데이터를 저장하지만, 향후 PostgreSQL 등 **데이터베이스로 전환**할 수 있습니다 [5]. DB를 사용할 경우 이 변수에 연결 문자열을 넣고 Cloud Run에 VPC 연결 등을 설정해야 합니다. (초기 도입 시에는 기존 데이터를 백업하고 신중히 마이그레이션해야 합니다.)
- **그 외:** 필요에 따라 추가 키들을 환경변수로 관리합니다. (.env에 정의하고 코드에서 `os.getenv`로 불러 쓰세요.)

`.env` 파일은 **절대로 Git 저장소에 커밋하지 말고**, `.gitignore`에 추가해야 합니다 [6]. 운영 환경(Cloud Run/CI)에서는 이 값을 직접 **환경변수 설정**이나 **시크릿**으로 관리합니다. 예를 들어 GCP Cloud Run 배포 시 `--update-env-vars` 옵션을 사용하거나, GitHub Actions의 Secrets로 ADMIN_TOKEN, API_KEY 등을 등록해 CI에서 적용합니다 [6].

로컬 개발 시에는 서버 실행 전에 `.env`의 변수를 로드해야 합니다. 일반적으로 `python-dotenv`를 이용하거나 터미널에서 수동으로 불러올 수 있습니다:

```
# .env 파일 로드 및 FastAPI 서버 실행 (개발모드)
source .env
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

위와 같이 하면 `.env`의 ADMIN_TOKEN, API_KEY 등이 `os.environ`에 설정되어 FastAPI 애플리케이션이 해당 값을 사용하게 됩니다 [7] [8]. (.env 로드 과정을 간편하게 하기 위해 `uvicorn` 실행 전에 자동으로 `.env`를 읽도록 코드를 구성할 수도 있습니다.)

> **보안 노트:** 운영 환경에서는 `.env` 파일을 사용하지 않고 Cloud Run 설정이나 CI/CD 시크릿으로 환경변수를 주입합니다. 모든 민감한 키는 절대로 코드에 하드코딩하지 말고 환경변수로 관리하세요 [9]. 또한 .env/시크릿 값들이 **로그나 오류 메시지에 노출되지 않도록** 각별히 유의합니다 (예: DEBUG 모드를 끄고, 오류 발생시 민감정보를 출력하지 않음).

주요 API 엔드포인트 및 사용법

쿼리저널 Admin 백엔드는 FastAPI로 구현되어 있으며, 일일 보고서 생성과 요약/번역, 결과물 출력 등을 위한 REST API 엔드포인트들을 제공합니다. **모든 Admin용 API는 인증용 헤더**(`Authorization: Bearer <ADMIN_TOKEN>`)가 필요하며, 올바른 토큰이 없으면 401 Unauthorized 응답을 반환합니다 [10] [11]. 아래에 각 주요 엔드포인트의 기능과 사용 예시를 정리합니다:

`POST /api/report` – 일일 보고서 생성

설명: 금일 키워드 뉴스 데이터를 모아 **일일 뉴스 보고서**(Markdown 포맷)를 생성하는 엔드포인트입니다. 내부적으로 `make_daily_report.py` 스크립트를 호출하여 현재 날짜(또는 지정된 날짜)의 뉴스를 취합하고 Markdown 형식의 보고서를 만듭니다 [12].

- **요청:** 본 엔드포인트는 JSON 본문을 필요로 하지 않으며, Admin 인증만 거치면 동작합니다 [13]. 즉, 헤더에 `Authorization: Bearer <ADMIN_TOKEN>`만 포함하여 POST 요청하면 됩니다.
- **응답:** 보고서 생성이 **성공**하면 결과를 JSON으로 반환합니다. 구현 방식에 따라 **Markdown 내용 자체**를 반환하거나, **서버에 저장된 보고서 파일 경로**를 반환할 수 있습니다 [12]. 예를 들어, 보고서를 파일로 저장하는 경우 다음과 같은 응답을 보낼 수 있습니다:

```
{ "success": true, "report_file": "2025-10-12_report.md" }
```

위 예시는 보고서 파일명이 `"2025-10-12_report.md"`임을 보여줍니다 [14]. 개발 시 파일명을 `YYYY-MM-DD_report.md` **형식**으로 날짜를 포함해 저장하도록 구현하여, 일자별로 파일이 누적되게 할 것을 권장합니다 [15].

- **동작:** Admin UI에서 운영자가 "일일 보고서 생성" 버튼을 클릭하면 이 API가 호출되고, 서버 측에서 당일 뉴스 데이터를 수집해 Markdown 문자열을 생성합니다 [12]. Markdown 문자열은 서버 `archive/` 폴더에 파일로 저장되며, 필요시 API 응답으로 내용을 직배송해 UI에 표시할 수도 있습니다 [12]. 현재 구현에서는 성공 메시지와 파일 경로만 응답하고, UI는 이를 받아 `"보고서 생성 완료!"` 등의 알림을 표시하거나 필요한 후속 조치를 할 수 있습니다 [16].

- **사용 예시 (cURL):**

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

※ 요청에는 **본문이 필요 없으며**, 서버는 이 호출을 받으면 자동으로 금일 보고서 생성을 수행합니다 [13]. 올바른 토큰을 포함하지 않으면 401 응답을 받게 됩니다. 응답 JSON에는 성공 여부와 생성된 보고서의 내용 또는 파일 경로가 담깁니다 [13].

- **사용 예시 (JS Fetch):**

```
const adminToken = localStorage.getItem('adminToken'); // 저장된 관리자 토큰
fetch("/api/report", {
  method: "POST",
  headers: { "Authorization": "Bearer " + adminToken }
})
.then(res => res.json())
.then(data => console.log("Report Generation Result:", data));
```

※ 실제 UI에서는 위 응답 `data`를 확인하여 보고서가 정상 생성되었음을 알리고(`success:true`), `data.report_file` 경로를 이용해 파일 다운로드 링크를 제공하거나, 추가로 내용을 미리보기할 수도 있습니다 [17] [18]. (향후 개선으로 Markdown을 HTML로 렌더링해 UI상에서 바로 보여주는 기능을 고려할 수 있습니다.)

- **에러 처리:** 만약 뉴스 데이터가 부족하거나 기타 이유로 보고서 생성에 실패하면 API는 적절한 오류 응답(예: 500 서버 오류나 400 요청 오류)을 반환해야 합니다 [19]. 이 경우 클라이언트는 "보고서 생성 실패: 원인..." 등의 메시지를 사용자에게 보여주어야 합니다. 또한 동일한 날짜에 대해 보고서를 **중복 생성하는 것을 방지**해야 합니다 [20]. 예를 들어 운영자가 실수로 버튼을 여러 번 눌렀을 때, 이미 해당 날짜의 보고서가 존재하면 새 요청을 무시하거나 기존 파일을 덮어쓸지를 결정해야 합니다. 권장 구현은 **같은 날짜에 두 번 생성 시 파일명에 타임스탬프를 추가**하거나 기존 파일을 백업/압축 저장하여 충돌을 피하는 것입니다 [20]. (ex: `2025-10-12_report.md`, `2025-10-12_report-2.md` 식으로)

`POST /api/enrich/keyword` **– 뉴스 요약 및 번역 (전체 기사 대상)**

설명: 금일 수집된 모든 기사에 대해 AI 요약을 생성하고 (선택적으로) 번역을 수행하는 엔드포인트입니다. 요약 알고리즘으로 OpenAI GPT 등의 외부 API를 사용하며, 각 기사 본문을 요약한 뒤 한국어 등 **목표 언어로 번역**까지 수행합니다 (API_KEY가 설정된 경우) [21]. 이 기능은 일종의 "전체 뉴스 카드 요약/번역"으로, 운영자가 당일 모든 뉴스를 빠르게 훑어보거나 이후에 기사 선정을 도울 목적으로 사용됩니다.

- **요청/인증:** Admin 토큰이 필요하며, 기본적으로 요청 본문은 필요 없습니다 (구현에 따라 `lang`와 같은 옵션을 쿼리스트링으로 줄 수는 있음) [22]. 이 엔드포인트 역시 `Depends(verify_admin_token)`으로 보호되어 있어야 합니다 [23].

- **동작:** 호출 시 **현재까지 수집된 모든 기사 데이터**를 불러와, 각 기사별로 외부 요약 API를 호출하여 요약문을 생성합니다 [21]. API_KEY (예: OpenAI 키)가 제공된 경우 요약문을 대상 언어로 **자동 번역**하여 함께 저장합니다 [21]. 예를 들어 영문 기사라면 한국어 번역문을 생성해주는 식입니다.

- **데이터 처리:** 요약/번역 결과는 각 기사 데이터 구조에 `summary` 필드와 `translation` 필드로 추가되는 형태로 메모리에 유지되거나, 별도 파일에 저장될 수 있습니다 [24] [25]. 구현에 따라 즉시 결과를 파일로 저장하도록 할 수도 있습니다 (예: `archive/enriched/2025-10-12.json`에 요약 결과 저장) [25]. API 응

답 자체는 간략히 **성공 여부나 처리한 기사 개수** 등을 JSON으로 반환하고 끝낼 수 있습니다 [26] [27]. (대량의 요약문 텍스트를 응답으로 모두 보내지 않고 파일에 저장해두는 편이 효율적입니다.)

- **응답/후속처리:** 이 API 호출이 완료되면 **프론트엔드에서는 특별한 표시가 없을 수도 있지만**, 구현에 따라 요약된 내용을 UI에 표시할 수 있습니다 [28]. 예를 들어 관리자 대시보드의 기사 리스트에 각 기사 아래 작은 글씨로 요약문을 보여주거나, 모든 요약 결과를 모달 팝업에 나열하는 식으로 UX를 개선할 수 있습니다 [28]. 기본적으로는 "요약 완료"라는 메시지 정도로 성공을 알리되, 운영자가 원한다면 각 기사의 요약 내용을 확인할 수 있게 하는 것이 좋습니다 [29].

- **사용 예시 (cURL):**

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

(모든 수집된 기사들에 대한 요약/번역을 수행) [30]

- **사용 예시 (JS Fetch):**

```
fetch("/api/enrich/keyword", {
  method: "POST",
  headers: { "Authorization": "Bearer " + adminToken }
}).then(res => res.json())
 .then(data => console.log("Keyword enrich result:", data));
```

(이 때 `data`에는 요약 처리 건수나 성공 여부 등이 담길 수 있습니다.)

- **참고:** 요약 대상 언어를 변경해야 하는 경우, `/api/enrich/keyword?lang=ko`와 같이 구현할 수 있습니다 [31]. 현재 기본 설정은 API_KEY에 해당하는 서비스의 기본 또는 지정 언어로 번역되도록 되어 있습니다. 또한 이 작업은 기사 수에 비례해 **실행 시간이 수십 초 이상** 걸릴 수 있습니다. 필요한 경우 FastAPI의 `BackgroundTasks`나 비동기(async) 작업으로 처리하고, 프론트엔드는 진행 상태를 나타내는 것이 좋습니다 (예: "요약 진행중...") [32].

`POST /api/enrich/selection` – **뉴스 요약 및 번역 (선정 기사 대상)**

설명: 운영자가 최종 선정한 기사들만 모아 요약/번역을 수행하는 엔드포인트입니다. 위 `keyword` 요약과 동작은 거의 같지만, **대상 기사 목록을 제한**한다는 점이 다릅니다 [33]. 최종 데일리 뉴스레터에 포함될 기사들에 대한 요약과 번역문을 생성하며, 이 결과는 발행 콘텐츠에 직접 활용되므로 정확성이 특히 중요합니다 [33].

- **요청/인증:** 사용법은 `/api/enrich/keyword`와 동일하며, ADMIN_TOKEN이 필요합니다. 요청 본문 없이 POST로 호출합니다.

- **동작:** 백엔드는 현재 "승인된(선정된) 기사" 목록만 추려서 그 기사들의 본문을 요약/번역합니다 [33]. 운영자가 전체 기사 중에서 어떤 것들을 최종 발행할지 선택하면, 그 **선택된 기사들만 대상으로** AI 요약을 수행합니다.

- **데이터 처리:** 생성된 요약/번역문은 해당 기사 데이터에 추가되거나 별도 결과 파일에 저장됩니다. 이 또한 summary, translation 필드로 메모리에 보관하거나, Markdown으로 바로 변환해 둘 수도 있습니다 [25]. (예: 선정된 기사들의 요약을 포함한 Markdown 본문 조각 등을 미리 만들어 둘 수도 있습니다.)

- **응답:** 성공 시 간단한 JSON (예: `{"success": true, "count": 5}` 등)으로 처리 결과를 알려주고 종료합니다. 자세한 요약문은 파일/메모리에 저장하여 이후 `/api/export` 단계에서 활용할 수 있게 합니다 [27].

- **사용 예시 (cURL):**

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

(편집자가 승인한 기사들만 대상으로 요약/번역 수행) [34]

- **사용 예시 (JS Fetch):**

```
fetch("/api/enrich/selection", {
  method: "POST",
  headers: { "Authorization": "Bearer " + adminToken }
}).then(res => res.json())
 .then(data => console.log("Selection enrich result:", data));
```

- **후속 처리:** `selection` 요약이 완료되면, **최종 발행본에 번역문이 반영되는 것**이 목적입니다 [35]. 구현에 따라, 이 번역 결과를 바로 Markdown 보고서에 삽입하거나, Export시 포함하도록 할 수 있습니다 [35]. 예를 들어, 번역된 요약문을 최종 Markdown에 포함시켜 **영문 기사 + 한글 번역 요약** 형태로 발행하면 가독성이 높아집니다. 운영자는 번역된 최종 요약문을 검토하여 어색한 부분이 있다면 발행 전에 수정할 수도 있습니다 [36].

  참고: OpenAI나 Papago API의 **요금제/호출 제한**에 유의해야 합니다. 많은 기사에 대해 요약/번역을 빈번히 호출하면 한도 초과나 비용 이슈가 발생할 수 있으므로, 운영자는 하루에 필요한 만큼만 실행하고 결과를 재사용하는 것이 좋습니다. 또한 번역 API의 **무료 할당량 소진 여부**를 모니터링하고, 필요하면 API 키를 교체하거나 다음 날을 대비해야 합니다 [37].

`GET /api/export/md` – **Markdown 보고서 추출**

설명: 최종 발행 결과물인 **데일리 뉴스 보고서**를 Markdown 파일로 다운로드받는 엔드포인트입니다. 일반적으로 `/api/report`로 당일 보고서를 생성한 후 이 API를 호출하면, **생성된 Markdown 텍스트를 응답**으로 받습니다. 프론트엔드에서는 해당 응답을 받아 파일 다운로드를 트리거하거나 새 탭에 내용을 표시합니다.

- **요청/인증:** GET 요청이며, 반드시 `Authorization: Bearer <ADMIN_TOKEN>` 헤더가 필요합니다. 토큰이 없으면 다운로드 대신 401 오류가 발생해야 합니다 [38].

- **동작:** 서버는 **최근 생성된 발행본의 Markdown 문자열**을 반환합니다 [39]. 구현에 따라 `FileResponse` 등을 사용해 **파일로 응답**하거나, `text/markdown` 컨텐츠를 스트림으로 내보낼 수 있습니다. 또한 응답 헤더에 `Content-Disposition: attachment; filename="YYYY-MM-DD_report.md"` 와 같이 **파일명 정보를 포함**하면 브라우저가 자동으로 해당 이름으로 다운로드를 시도합니다 [40].

- **사용 예시 (프론트엔드):** Markdown은 텍스트 형식이므로, 단순히 `fetch` 후 `res.text()`로 받아 새 창에 열 수도 있고, Blob으로 받아 다운로드할 수도 있습니다 [41]. 예를 들어 새 탭에 Markdown을 표시하려면:

```
fetch("/api/export/md", { headers: { "Authorization": "Bearer " + adminToken } })
 .then(res => res.text())
 .then(text => {
   const mdWin = window.open("", "_blank");
   mdWin.document.write("<pre>" + text + "</pre>");
 });
```

이렇게 하면 새 탭에 Markdown 소스가 나타나긴 하지만, Markdown을 렌더링하지는 않으므로 가독성은 떨어집니다. 필요하다면 JS로 Markdown을 HTML로 변환해 보여줄 수 있습니다. **대안으로**, 바로 다운로드 받으려면 아래 CSV의 Blob 예시처럼 처리하거나, 응답 헤더의 Content-Disposition을 활용할 수 있습니다.

- **사용 예시 (cURL):**

```
curl -H "Authorization: Bearer ${ADMIN_TOKEN}" \
  -o "latest_report.md" "https://<배포도메인>/api/export/md"
```

(성공 시 `latest_report.md` 이름으로 파일이 저장됩니다) [42].

`GET /api/export/csv` – **CSV 데이터 추출**

설명: 최종 발행본 기사들을 CSV 형식으로 추출하는 엔드포인트입니다. CSV 내에는 키워드, 제목, 본문 요약, 번역문, 원문 링크 등의 필드를 포함하도록 설계할 수 있습니다. 이 기능은 발행 결과를 **엑셀 등으로 분석하거나 아카이빙**하기 위함입니다 [43].

- **요청/인증:** `/api/export/md`와 동일하게 GET 요청 + 인증 헤더가 필요합니다.
- **동작:** 서버는 **최신 발행본의 CSV 문자열**을 생성하여 응답합니다 [39]. (이미 구현된 경우 백엔드에서 Markdown 생성 시 함께 CSV도 생성해 두었을 것입니다. 구현되어 있지 않다면, 현재 선택된 기사 리스트를 불러와 각 필드를 CSV로 포맷팅하여 반환하도록 해야 합니다 [44].) CSV 다운로드의 경우 브라우저는 바이너리/octet-stream으로 처리하므로, 프론트엔드에서 Blob을 이용해 수동으로 파일 저장을 트리거해야 합니다 [45] [46].
- **사용 예시 (프론트엔드 Blob):** CSV 다운로드를 위해 fetch로 응답을 받아 Blob으로 변환하고, 가짜 링크를 클릭하여 파일을 저장할 수 있습니다:

```
document.getElementById('btn-export-csv').onclick = function() {
  fetch("/api/export/csv", { headers: { "Authorization": "Bearer " + adminToken } })
  .then(res => res.blob())
  .then(blob => {
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "latest_report.csv";  // 다운로드될 파일명 지정
    document.body.appendChild(a);
    a.click();
    a.remove();
    window.URL.revokeObjectURL(url);
  })
```

```
    .catch(err => alert("CSV Export 실패: " + err));
};
```

위 예시는 CSV Blob을 생성한 뒤 임시 `<a>` 엘리먼트를 클릭하여 `"latest_report.csv"` 파일을 다운로드하는 코드입니다 [47] [48].
Markdown의 경우 text로 바로 받을 수 있으므로 Blob 처리 없이도 가능하지만, 보안상 인증 헤더를 포함하려면 fetch를 써야 합니다. `<a href="/api/export/md" download>`처럼 직접 링크를 제공하면 Authorization 헤더를 첨부할 수 없고, 토큰을 URL에 노출시키는 것은 위험하므로 피해야 합니다 [46] [49]. 따라서 **반드시 fetch로 호출**하여 다운로드를 처리하세요.

- **사용 예시 (cURL):**

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

(성공 시 `latest_report.csv` 파일로 저장됩니다) [50].

- **주의:** 백엔드가 응답을 보낼 때 `Content-Type: text/csv; charset=utf-8` 및 `Content-Disposition: attachment; filename="YYYY-MM-DD_report.csv"` 등의 헤더를 설정하면 브라우저에서 다운받는 파일명이 지정될 수 있습니다 [40]. 구현 시 이러한 헤더를 추가해두는 것을 권장합니다. 프론트엔드 테스트 시 Cloud Run 환경에서 CORS 문제가 없는지도 함께 확인하세요 [51] (동일 오리진에서 Admin UI를 제공하면 큰 문제는 없습니다).

CLI 스크립트 실행 (보고서 생성/요약)

Admin API는 위와 같이 HTTP 엔드포인트로 제공되지만, **개발/디버깅 목적**으로 백엔드 로직을 직접 실행해볼 수 있습니다.

- `tools/make_daily_report.py`: 일일 보고서 생성 로직이 구현된 Python 스크립트입니다 [52]. API를 경유하지 않고 이 스크립트를 직접 실행하면 (예: `python tools/make_daily_report.py`) 당일 보고서 Markdown을 생성하여 파일로 저장하거나 표준출력으로 출력할 수 있습니다. 로컬에서 이 스크립트를 테스트할 때도 `.env`의 ADMIN_TOKEN, API_KEY 등이 로드된 상태여야 외부 API 호출 등 내부 로직이 정상 동작합니다.

- `tools/enrich_cards.py`: (가정) 뉴스 요약/번역 관련 함수들이 들어있는 모듈입니다. 직접 실행 스크립트는 없을 수 있으나, 해당 모듈의 함수를 대화형 환경에서 호출하여 요약 결과를 얻어볼 수 있습니다. 예를 들어 `python` REPL이나 Jupyter 노트북에서 `from tools.enrich_cards import summarize_all, summarize_selection` 등을 임포트해 호출하면 각 기사 리스트에 대한 요약을 수행할 것입니다. 이때도 실제 API_KEY를 세팅하거나, 테스트를 위해 AI API 호출 부분을 목업(mock)하여 실행해야 과금이나 오래 기다리는 것을 피할 수 있습니다.

- **관리자 기능 CLI 대안:** 만약 운영 환경에서 **API 서버를 거치지 않고 수동으로 보고서 생성을 트리거**해야 하는 경우(예: 긴급 상황에서 CI/CD 없이 수동 발행), 개발자는 서버 코드 없이도 동작하는 스크립트를 준비해둘 수 있습니다. 예를 들어 `python tools/make_daily_report.py --date 2025-10-15` 같이 특정 날짜 인자를 받아 동작하도록 해두면 활용도가 높아집니다. 다만 일반 시나리오에서는 Admin UI 또는 API 호출로 처리하므로, CLI는 보조 수단으로 생각하십시오.

로컬 개발 환경에서 서버 실행 & 디버깅

개발 시에는 로컬에서 FastAPI 서버를 구동하여 UI와 API를 테스트할 수 있습니다. 위에서 언급한 `.env`를 로드한 뒤 다음을 실행합니다:

```
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

이렇게 하면 개발용 서버가 `http://localhost:8000`에서 실행되며, Admin UI (정적 파일)와 API 엔드포인트를 사용할 수 있습니다. **정적 파일**(Admin 프론트엔드 HTML/JS/CSS)은 FastAPI에 `/` 경로나 `/static` 경로로 서빙되도록 설정되어 있어야 합니다. 예를 들어 `app.main`에서 `app.mount("/", StaticFiles(directory="admin-ui", ...))` 또는 Jinja2 템플릿 방식으로 Admin 페이지를 제공한다면, 로컬 서버에서도 브라우저로 해당 페이지에 접근 가능합니다. UI가 뜨면, 개발용 ADMIN_TOKEN을 세팅하여 (예: 브라우저 콘솔에서 `localStorage.setItem('adminToken', '<token>')`) 각 기능을 테스트합니다 [53].

API 테스트: 로컬 서버에서 API를 테스트할 때는 Swagger UI나 curl을 이용하면 편리합니다. `http://localhost:8000/docs`로 접속하면 자동 문서(Swagger)가 제공되는 경우 거기서 토큰을 넣어 직접 호출해볼 수 있습니다. 또는 curl을 사용해 예시처럼 호출합니다:

```
# 로컬에서 토큰 없이 호출 (401 응답 확인)
curl -i "http://localhost:8000/api/report"
# → HTTP/1.1 401 Unauthorized (인증 실패)

# 로컬에서 토큰 포함 호출 (200 응답 및 결과 확인)
curl -i -H "Authorization: Bearer ADMIN_SECRET_TOKEN" "http://localhost:8000/api/report"
# → HTTP/1.1 200 OK (응답 JSON 출력)
```

이 때 ADMIN_SECRET_TOKEN 자리에 `.env`에 설정한 토큰값을 넣어주세요 [10]. 이러한 테스트를 통해 **인증 미들웨어**가 잘 적용되었는지 (토큰 없을 때 401 발생) 확인하고, 실제 보고서 생성 로직이 예Expected 대로 Markdown을 만들어내는지 볼 수 있습니다.

디버깅: 오류가 발생하면 콘솔 로그나 응답 메시지를 확인하세요. FastAPI의 디폴트 에러 페이지에는 스택트레이스를 보여주므로 개발 모드에서는 참고가 됩니다. 단, 운영 모드에서는 **DEBUG=False**로 설정해 민감정보가 노출되지 않도록 합니다 [54].

또한, 외부 API 호출 (OpenAI 등)이 연쇄적으로 일어나는 기능의 경우, 로컬에서 테스트할 때 실제 API를 호출하면 지연이 길고 비용이 발생할 수 있으므로, **테스트/디버깅 시에는 모의 응답(mock)** 을 사용하는 것이 좋습니다. 예를 들어, Python 단위테스트에서 `monkeypatch`를 이용해 `openai.Completion.create` 메소드를 가짜로 대체하여 항상 동일한 문자열을 반환하게 한 후 요약 함수를 호출하면, 외부 API 없이도 동작 검증이 가능합니다 [55].

Cloud Run에 컨테이너 배포하기

쿼리저널 Admin 시스템은 **Docker 컨테이너**로 패키징하여 Google Cloud Run 등에 배포할 수 있습니다. 이하 절차는 GCP Artifact Registry와 Cloud Run을 사용하는 경우를 예로 듭니다.

1. **Docker 이미지 빌드:** 현재 프로젝트 디렉토리에 Dockerfile이 있다고 가정합니다. 코드 수정 후 **Dockerfile을 최신 코드와 종속성에 맞게 업데이트**했는지 확인하세요 (새로운 Python 패키지를 썼다면 Dockerfile에도 추가해야 함). 그런 다음 도커 이미지를 빌드합니다:

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

위 명령에서 `<GCP_PROJECT_ID>`는 실제 GCP 프로젝트 ID로 바꾸십시오. Docker 이미지 태그를 GCP Artifact Registry 경로로 지정하여 바로 클라우드에 푸시할 수 있게 합니다 [56]. (이미 Artifact Registry에 리포지토리를 만들었다면 그 경로를 사용)

2. **이미지 레지스트리에 Push:** 빌드한 이미지를 GCP Artifact Registry에 푸시합니다:

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

푸시가 성공하면 해당 이미지가 원격 레지스트리에 저장됩니다 [56].

3. **Cloud Run 서비스에 배포:** 이제 최신 이미지를 Cloud Run에 배포합니다. gcloud CLI를 사용하면 편리합니다:

```
gcloud run deploy qualijournal-admin \
  --image asia.gcr.io/<GCP_PROJECT_ID>/qualijournal-admin:latest \
  --region=asia-northeast3 --platform=managed \
  --update-env-vars ADMIN_TOKEN=${ADMIN_TOKEN},API_KEY=${API_KEY},DB_URL=${DB_URL}
```

위 명령에서 서비스 이름(`qualijournal-admin`), 리전(예: asia-northeast3) 등을 실제 환경에 맞게 변경하세요 [57]. `--update-env-vars` 플래그를 통해 배포 시 Admin 토큰, API 키, DB URL 등의 환경변수를 세팅했습니다 [58]. (`${ADMIN_TOKEN}` 등의 값은 현재 쉘에 export되어 있다고 가정하거나 직접 리터럴로 넣을 수 있습니다.) 배포에 성공하면 Cloud Run 서비스 URL과 리비전 정보가 출력됩니다 [59].

배포가 완료되면 Cloud Run 콘솔에서 **Revision History**에 새 이미지가 기록되고 트래픽이 최신 리비전에 할당되었는지 확인합니다 [60]. Cloud Run 서비스 URL로 접속하여 Admin UI가 정상 로드되고 API 호출이 동작하는지도 직접 확인하세요 (예: 브라우저에서 Cloud Run URL을 열어 관리자 페이지에 접속, 버튼들을 눌러보기).

> **환경변수 설정:** 위에서 `--update-env-vars`로 준 값들은 Cloud Run 서비스 설정에 반영됩니다. 이후 새로운 이미지를 배포할 때 특별히 env를 변경하지 않으면 기존 설정이 유지되니, 굳이 매번 넣지 않아도 됩니다 [61] [62]. 다만 **ADMIN_TOKEN을 교체**해야 할 경우에는 해당 옵션으로 새 토큰을 넣어 배포하거나, Cloud Run 콘솔에서 수동으로 환경변수를 수정할 수 있습니다.
>
> **수동 배포 vs CI/CD:** 초기에는 위와 같이 수동으로 build/push/deploy 하지만, CI/CD 파이프라인을 구축해 **git push 시 자동으로 배포**하도록 설정할 수 있습니다. 아래 CI/CD 설정 내용을 참고하세요.

CI/CD 설정 및 자동 배포

지속적 통합/배포(CI/CD)를 통해 코드 변경 시 자동으로 테스트를 돌리고 배포까지 수행할 수 있습니다. QualiJournal Admin의 예시로 **GitHub Actions**를 사용한 Workflow를 설정할 수 있습니다. 주요 아이디어는: **main 브랜치에 코드**

푸시 -> **CI가 pytest 등 테스트 실행** -> **도커 이미지 빌드 & Artifact Registry에 push** -> **Cloud Run에 배포** 순서로 진행하는 것입니다 [63] .

- **GitHub Actions 워크플로우:** `.github/workflows/deploy.yml` 등의 파일에 GCP 인증 및 Cloud Run 배포 job을 정의합니다. 예를 들어 Google 제공 액션인 `google-github-actions/deploy-cloudrun@v0`를 사용하여 Cloud Run에 배포할 수 있습니다 [63] . 이때 GitHub Actions에 GCP 서비스 계정 JSON 키를 시크릿으로 등록하거나, Workload Identity Federation(OIDC)를 설정하여 권한을 부여해야 합니다 [64] . 또한 ADMIN_TOKEN, API_KEY 등도 Actions 시크릿에 저장해 workflow에서 환경변수로 주입합니다 [65] .

- **예시 워크플로우 개요:**

  1. 코드 체크아웃
  2. Python 테스트 실행: `pytest` 명령으로 모든 테스트 통과 확인 [66] (테스트에 필요한 ADMIN_TOKEN 등은 dummy 값으로 시크릿에 설정하여 사용).
  3. 도커 빌드 및 Artifact Registry 푸시
  4. Cloud Run 배포 (이미지 경로 및 환경변수 지정)
  5. 성공시 알림 또는 후속 작업

GitHub Actions YML 내 예시:

```
jobs:
  build-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: google-github-actions/setup-gcloud@v1
        with:
          service_account_key: ${{ secrets.GCP_SA_KEY }}
          project_id: ${{ secrets.GCP_PROJECT }}
      - run: gcloud builds submit --tag asia.gcr.io/$GCP_PROJECT/qualijournal-admin:${{ github.sha }}
      - run: gcloud run deploy qualijournal-admin --image asia.gcr.io/$GCP_PROJECT/qualijournal-
admin:${{ github.sha }} --region=asia-northeast3 --platform=managed
```

위 예시는 단순화한 것이며, 실제론 test 단계, env var 설정 등이 추가되어야 합니다. (예를 들어 `--update-env-vars ADMIN_TOKEN=${{ secrets.ADMIN_TOKEN }},API_KEY=${{ secrets.API_KEY }}`를 배포 명령에 넣어주는 등.)

- **CI 설정 확인:** CI/CD 파이프라인이 설정되었다면, 코드 푸시로 자동 배포가 잘 되는지 검증합니다. main 브랜치에 푸시하면 Actions가 트리거되어 **테스트 -> 빌드 -> 배포** 단계가 순차적으로 실행되는지 확인하세요 [67] . 만약 테스트가 실패하면 배포 단계가 스킵되는지, 테스트 성공 후에는 Cloud Run에 새로운 리비전이 배포되는지 살펴봅니다 [67] [68] . Actions 로그에 민감한 환경변수 값이 노출되지 않았는지도 점검합니다 [69] .

- **Cloud Run 배포 모니터링:** CI/CD를 통해 배포된 경우에도, Cloud Run 콘솔의 "Revisions" 탭에서 최신 이미지와 배포 시간이 반영되었는지 확인합니다 [70] . 그리고 **트래픽 할당이 100% 새 리비전에 되어있는지** 확인합니다 (수동으로 이전 리비전에 일부 트래픽을 남겨두지 않았는지) [71] .

- **배포 결과 검증:** 새로운 버전이 올라간 후 Cloud Run 서비스 URL (또는 설정한 커스텀 도메인)로 접속하여 **관리자 UI의 모든 기능을 실제로 테스트**합니다 [72] . 예를 들어:

- 관리자 페이지가 열리는지
- "일일 보고서 생성" 버튼 클릭 -> 성공 메시지 표시 및 로그 확인
- "전체 요약", "선정본 요약" 클릭 -> 요약 완료 메시지 및 UI 반영 확인
- "Export CSV/MD" 클릭 -> 파일 다운로드 확인 (내용 열어보기 포함)
  이러한 시나리오를 운영자 입장에서 한 번 쭉 수행하여 문제없는지 확인합니다 [72] [73] . 동시에 Cloud Run의 Logs에서 해당 API 호출들이 200 OK로 처리되었는지, 예외(traceback) 로그는 없는지 살펴보세요 [74] .

테스트 자동화 (pytest 등)

코드에 대한 **단위테스트/통합테스트**를 작성하여 주요 기능이 잘 동작하는지 검증해야 합니다. 특히 Admin 시스템에는 인증과 외부 API 호출이 포함되므로, 다음과 같은 테스트 시나리오를 고려합니다:

- **인증 관련 테스트:** 잘못된 토큰 또는 토큰 없이 각 Admin API를 호출할 경우 401 또는 403 응답을 반환하는지 확인합니다. 올바른 토큰으로 호출하면 200 OK와 기대되는 데이터를 반환해야 합니다 [10] .
- **일일 보고서 생성 테스트:** `/api/report` 엔드포인트를 테스트합니다. 이때 실제 뉴스를 모으는 로직은 I/O가 있으므로, 테스트에서는 해당 부분을 stub/mock 처리하거나, 테스트용 작은 JSON 데이터를 준비해놓고 수행합니다. 예를 들어 `make_daily_report()` 함수를 호출했을 때 Markdown 문자열이 특정 키워드(제목 등)를 포함하는지, 혹은 리턴된 파일명이 날짜 형식인지 등을 확인합니다.
- **요약/번역 기능 테스트:** `/api/enrich/keyword`와 `/api/enrich/selection` 호출 시 외부 API 호출을 실제로 하지 않고도 테스트할 수 있어야 합니다. **외부 API 모킹**이 중요한데, pytest에서는 `monkeypatch`를 이용해 openai나 Papago 호출 함수를 가짜 함수로 치환하여 항상 고정된 요약문을 리턴하게 설정합니다 [55] . 그런 다음 엔드포인트를 호출하거나, 내부 함수를 직접 호출해 summary 필드가 잘 추가되었는지 검사합니다. 또한 토큰 인증이 없으면 401을 주는지, 토큰 있을 때 200과 JSON 응답을 주는지도 테스트합니다 [55] .
- **Export 기능 테스트:** `/api/export/md`와 `/api/export/csv`를 호출하면 Content-Type과 응답 내용이 올바른지 확인합니다. pytest로 파일 응답을 다루는 경우 `testclient.get("/api/export/md", headers=auth)` 호출 후 `response.status_code == 200`과 `response.text` 또는 `response.content` 내용에 특정 문자열(예: 보고서 제목)이 들어있는지 확인할 수 있습니다. CSV의 경우 `response.text`를 CSV 파싱하여 컬럼 수, 행 수 등을 검사합니다. 또한 인증 없을 때 401 나오는지도 테스트합니다.
- **동시성 및 중복 방지:** 동일한 리소스에 대해 여러 번 호출했을 때 문제없는지 테스트합니다. 예를 들어 `/api/report`를 연속 두 번 호출하면 두 번째 호출에서 적절히 처리되는지 (덮어쓰기 또는 무시) 확인합니다. 또는 pytest를 이용해 멀티스레드로 동시에 호출 시에도 문제가 없도록 설계되었다면 그런 테스트도 고려합니다 (다만 일반적으로는 필요 없을 수도 있습니다).

테스트를 실행하려면 프로젝트 폴더에서 `pytest` 명령을 사용합니다. CI에서도 이 명령을 사용하여 모든 테스트가 통과했는지 확인합니다 [66] .

테스트 팁: CI 환경에서는 실제 운영 토큰/키를 쓰지 않고 **테스트용 값**을 사용해야 합니다 [75] . 예컨대 GitHub Actions에 `ADMIN_TOKEN_TEST` 시크릿을 "test-token" 같은 값으로 넣고, 코드에서 테스트 모드일 때 해당 토큰으로 대체하거나, pytest에서 환경변수를 오버라이드하는 방식으로 처리할 수 있습니다 [76] . 외부 API 호출 테스트에서는 비용이나 시간 문제로 실제 호출을 하면 곤란하므로 반드시 mocking 처리하세요. 이를 위해 함수에 의존성 주입을 사용하거나, monkeypatch fixture를 활용하면 좋습니다 [55] .

또한 테스트 간 **상호 간섭**이 없도록 주의합니다. 예를 들어 어떤 테스트에서 임시 파일을 생성하면 테스트 종료 후 정리하고, 여러 테스트를 병렬 실행할 경우에도 각각 별개의 데이터를 사용하도록 합니다 [65] . (`/tmp` 디렉토리를 활용하거나, Fixture의 scope를 조정하여 매 테스트마다 새로운 인스턴스를 사용).

보안, 로그 및 백업 관리

마지막으로, Admin 시스템을 안정적으로 운영하기 위해 **보안 설정과 데이터 백업/관리**에 유의해야 합니다.

- **토큰 보안:** Admin 인증 토큰(ADMIN_TOKEN)은 유출 시 누구든지 API를 호출할 수 있으므로 **절대 노출되지 않게** 관리합니다. 토큰 값이 코드나 공개 저장소에 담기지 않도록 하고, Frontend 배포 시 소스 맵 제거 등을 통해 토큰 문자열이 검색되지 않도록 합니다 [77]. 운영 중에는 **정기적으로 토큰을 변경**하는 것도 고려합니다 [78]. 토큰을 변경할 경우 Cloud Run 환경변수와 프론트엔드 설정(예: localStorage 저장값)을 동시에 업데이트하고, 이전 토큰은 즉시 폐기합니다 [78].

- **접근 통제:** Cloud Run 서비스를 deploy할 때 `--allow-unauthenticated` 설정을 사용하면 토큰이 없더라도 엔드포인트에 접근 시도는 가능합니다. 현재는 모든 유효 요청에 토큰 검사를 하므로 큰 문제는 없으나, 보다 안전하게 하려면 Cloud Run 자체를 **인증된 호출만 허용**하도록 설정하거나, IP 제한을 추가로 거는 방법도 있습니다 [79] [80]. (다만 이런 설정을 하면 프론트엔드도 인증을 거쳐야 하므로 현재 구조에서는 복잡해질 수 있습니다.)

- **로그 관리: 민감정보를 로그에 남기지 않도록** 합니다. 예를 들어 요청을 로깅할 때 Authorization 헤더 전체를 찍지 말고, 토큰이 일부라도 노출되지 않게 해야 합니다 [54]. 또한 DEBUG 모드를 끄고, error 로그에도 환경변수값이 출력되지 않는지 점검합니다 [81]. Cloud Run의 로그를 주기적으로 모니터링하여, **보안 이상징후**(예: 토큰 없이 접근 시도 반복 등)가 있으면 ADMIN_TOKEN 교체나 IP 차단 등의 조치를 검토합니다 [82].

- **아카이브(Archive) 관리:** 보고서와 요약 결과 등을 파일로 `archive/` 폴더에 저장해두는데, **Cloud Run 컨테이너는 스테이트리스(stateless)**이므로 컨테이너가 재시작되면 해당 파일들이 사라질 수 있습니다 [83]. Cloud Run에서는 컨테이너 인스턴스가 언제든 초기화될 수 있으므로, **중요한 발행 결과는 반드시 외부에 백업**해두어야 합니다 [83] [84]. 구체적으로, **운영자는 매일 발행 후 Export 기능으로 Markdown과 CSV 파일을 로컬에 다운로드 받아** 사내 서버나 클라우드 스토리지(예: 구글 드라이브)에 업로드해 보관하는 절차를 따르는 것이 좋습니다 [83] [84]. 이렇게 하면 나중에 과거 뉴스를 조회하거나 문제가 발생했을 때 복구할 수 있습니다.

- 만약 별도의 영구 스토리지(예: Cloud Storage 버킷)를 연결하여 archive 폴더를 유지하도록 했다면, **파일이 지속적으로 누적**되므로 주기적으로 오래된 파일을 압축 보관하거나 삭제하여 용량을 관리합니다 [84]. 예를 들어 3개월 지난 보고서는 ZIP으로 묶어 별도 저장소로 옮긴다든지 정책을 정합니다.

- **의존성 및 취약점 관리:** Admin 시스템이 사용하는 Python 패키지 등의 **의존성을 정기적으로 업데이트**하고, `pip audit`이나 `safety` 같은 도구로 알려진 취약점이 있는 패키지가 없는지 검사합니다 [85]. 중요 보안 패치가 나오면 즉시 적용하고 이미 배포된 컨테이너를 재배포합니다. Docker 이미지에도 오래된 라이브러리가 남지 않게 하고, 필요없는 레이어는 제거하여 빌드합니다 [86].

- **모니터링:** Cloud Run의 **로그와 모니터링 대시보드**를 활용해 CPU, 메모리 사용량, 에러 발생률 등을 꾸준히 관찰합니다 [87] [88]. 새로운 기능 추가 후 자원 사용량이 증가할 수 있으니, 필요하면 Cloud Run 서비스의 메모리/CPU 할당량을 늘립니다 [87]. 또한 GCP Error Reporting 등을 사용하면 미처 처리하지 못한 예외가 쌓일 경우 바로 인지할 수 있으므로 적극 활용합니다 [87].

- **긴급 대응:** 배포 후 심각한 문제가 발견되면 **롤백 전략**을 준비합니다. Cloud Run에서는 이전 리비전으로 트래픽을 돌리는 것이 가능하므로, 문제 시 신속히 이전 버전으로 복구하는 절차를 마련해 두세요 [89]. 또한 CI/CD 상에서도 배포 중단이나 롤백 워크플로우를 구성해둘 수 있습니다 [89]. (예: "deploy" job 실행 전에 수동 승인 단계를 넣거나, 별도 `rollback.yml` 워크플로우를 만들어 필요 시 실행)

최종 점검 체크리스트 (개발자용)

개발 작업이 완료되고 배포 전에, 아래 **체크리스트**를 통해 빠진 부분이 없는지 확인하세요:

- [ ] **환경 변수 구성:** `.env` 에 ADMIN_TOKEN, API_KEY 등이 올바르게 설정되어 있고, 해당 파일은 git에 커밋되지 않았는지 확인합니다. Cloud Run 환경변수에도 최신 값이 반영되었는지 (`gcloud run services describe` 명령으로) 점검합니다 [90] [91] . 프론트엔드 JS에 하드코딩된 토큰이 없다면 `localStorage` 등에서 토큰을 불러오는지 재확인합니다.
- [ ] **인증 및 보안:** 모든 Admin API 엔드포인트가 토큰 인증을 요구하며, 잘못된 토큰에 401 응답을 주는지 테스트했습니다 [10] . 프론트엔드에 저장된 토큰과 백엔드 환경변수의 토큰이 **일치**하는지 확인합니다 [82] . (배포 후 401이 연속 발생하면 토큰 불일치를 의심해야 합니다.) 로그에 토큰이나 API 키 등 민감정보가 남지 않도록 로그 설정을 최종 점검했습니다 [81] .
- [ ] **기능 구현 완료:** 신규 엔드포인트들 (`/api/report` , `/api/enrich/*` , `/api/export/*`)이 모두 FastAPI에 추가 구현되었고 [92] , 프론트엔드의 버튼 클릭과 연결되어 있습니다. 버튼 ID, 경로 등이 정확히 맞는지 (예: `#btn-export-md` -> `/api/export/md`) 확인하세요. 또, 기능 동작에 필요한 파일(예: tools 스크립트, JSON 데이터)이 누락되지 않았는지 확인합니다.
- [ ] **테스트 통과:** 모든 **자동화 테스트(Pytest)**가 통과했습니다. 특히, 보고서 생성/요약/익스포트 관련 테스트에서 예상한 응답이 나오고, edge case에 대한 예외처리도 문제없음을 확인합니다 [66] . 토큰 없을 때 401, 있을 때 200 및 내용 검증, 외부 API 호출부는 monkeypatch로 잘 대체하여 테스트하는지 등의 항목을 점검합니다.
- [ ] **UI 연동 확인:** 로컬 환경에서 관리자 HTML/JS와 API를 함께 실행해 **통합 테스트**를 거쳤습니다. 버튼들을 실제로 눌러보고 UI에 예상대로 표시되는지, 네트워크 탭에 요청/응답이 올바른지 확인합니다. 특히 "일일 보고서 생성" 후 성공 알림 또는 파일 링크 노출, "요약/번역" 실행 후 요약결과 표시, "Export" 클릭 시 파일 다운로드가 이루어지는지를 확인합니다 [93] [17] .
- [ ] **CI/CD 파이프라인:** GitHub Actions 등 CI 설정이 구성되어 있다면, **의도적으로 코드 변경을 커밋/푸시**하여 CI가 동작하는지 확인합니다 [67] . Actions 로그에서 테스트 -> 빌드 -> 배포 순으로 진행되고 있는지, 비밀값이 노출되지 않았는지 [69] , 배포 완료 후 Cloud Run에 새 리비전이 올라왔는지 확인합니다.
- [ ] **최종 배포 및 기능 점검:** 실제 Cloud Run에 최신 코드를 배포한 후, **운영자 시나리오로 기능 점검**을 수행합니다 [73] . 운영자 가이드의 절차에 따라 순서대로 버튼을 눌러 보고서 생성 -> 요약 -> Export까지 한 사이클을 돌려봅니다. 이 과정에서 발생하는 **모든 API 호출이 2xx 성공**이며, UI에도 이상이 없음이 확인되어야 합니다 [72] . Cloud Run 서비스 로그에도 Error-level 로그가 없는지 확인합니다 [94] .
- [ ] **문서 최신화:** README.md 등의 **개발 문서**에 환경변수 설정법(.env 예시 포함)과 로컬 실행 방법, Cloud Run 배포 방법, 주요 REST API 목록 및 사용 예시 등을 업데이트했습니다 [95] . 또한 **운영 가이드 문서**(본 가이드북)에도 새로 추가된 기능 (보고서 생성, 요약/번역, Export)에 대한 사용법과 주의사항을 모두 반영했습니다 [96] . 필요하다면 UI 스크린샷을 첨부하거나, 운영 절차를 순서대로 정리하여 **운영팀에 인계**합니다.
- [ ] **향후 개선 목록 정리:** 이번 개선에서 미처 다 하지 못한 **추가 기능이나 개선 아이디어**가 있다면 별도로 기록해두었습니다 (예: 다중 사용자 인증 도입, 추천 기사 자동선정 알고리즘, 검색 기능 등) [97] . 프로젝트 로드맵이나 TODO 리스트를 업데이트하여 추후 작업에 참고토록 합니다.

위 체크리스트를 모두 만족하면, 개발 및 배포 준비가 완료된 것입니다. 이제 운영자는 아래 운영자 가이드를 참고하여 시스템을 활용하면 됩니다.

운영자 가이드

퀄리저널 관리자 시스템의 운영자용 가이드는 **관리자 웹 UI**를 통해 일일 뉴스를 발행하고 관리하는 방법을 다룹니다. 이 가이드는 데일리 뉴스 작업을 수행하는 에디터/운영자분들을 위한 것으로, 관리자 페이지 접속부터 각 버튼의 역할, 그리고 산출물(보고서)을 활용/보관하는 방법까지 설명합니다.

시스템 접속 및 인증

퀄리저널 관리자 시스템 UI는 **웹 브라우저**를 통해 접근합니다. 배포된 Cloud Run 서비스의 URL (예: `https://qualijournal-admin-xxxx.run.app`)로 접속하면 관리자 화면이 로드됩니다. 이 페이지를 열 때 **인증 토큰** 확인 절차가 있을 수 있습니다. 현재 시스템은 로그인 폼이 없는 대신 **사전 공유된 ADMIN 토큰**으로 인증하므로, 브라우저가 해당 토큰을 알고 있어야 합니다 [53].

- 처음 접속하거나 토큰이 설정되지 않은 경우, 개발팀으로부터 전달받은 ADMIN_TOKEN 값을 브라우저 콘솔에서 설정해야 할 수 있습니다. 예를 들어, 크롬에서 F12를 눌러 콘솔 창에 `localStorage.setItem('adminToken', '<발급받은토큰>');` 을 입력하고 페이지를 새로고침하면 인증이 적용됩니다 [98]. (향후 UI 개선으로 토큰을 입력하는 창이 제공될 수 있습니다 [99].)
- 이후부터는 **모든 API 요청**에 이 토큰이 포함되어 전송되므로, 별도 로그인이 없어도 관리자 기능을 사용할 수 있습니다. 토큰은 매우 민감한 값이므로 다른 사람과 공유하지 말고, 노출되지 않도록 주의하세요. 만약 토큰이 유출되었거나 의심되면 즉시 개발팀에 알려 **새 토큰으로 교체**해야 합니다.

정상적으로 토큰이 설정되면, 관리자 페이지의 각종 버튼과 기능을 제약 없이 사용할 수 있습니다. 다음은 일일 발행 작업의 일반적인 흐름과 각 기능에 대한 설명입니다.

일일 보고서 생성 (Daily Report 생성)

역할: 금일의 뉴스 키워드 보고서를 생성하는 기능입니다. 운영자는 하루에 한 번, 해당 버튼을 눌러 그 날 발행할 뉴스 요약 리포트를 만듭니다.

사용 방법: 관리자 대시보드 화면에서 **「일일 보고서 생성」** 버튼을 클릭합니다. 그러면 백엔드에서 금일자 뉴스를 정리하여 Markdown 형태의 **데일리 리포트**를 생성합니다 [12]. 버튼 클릭 시 즉각 화면에 큰 변화는 없지만, 백그라운드로 작업이 진행되며 보통 **2~5초 내에 완료**됩니다 (뉴스 기사 수에 따라 다름).

- UI 상에서는 **"보고서 생성 중..."**과 같은 로딩 표시가 나타날 수 있습니다. 로딩 표시는 수 초 이상 걸리는 작업에 대해 사용자에게 제공되는 피드백입니다 [100]. 만약 UI에 로딩 스피너나 진행 표시가 없다면, 클릭 후 조금 기다리면 결과 알림이 뜹니다.
- 완료되면 **"보고서 생성 완료"** 등의 알림 팝업이나 메시지가 화면에 표시됩니다 [16]. 이 메시지로 생성 성공을 확인할 수 있습니다.
- 일부 구현에서는 생성된 Markdown 콘텐츠를 바로 새 창으로 열어 보여주거나, **파일 다운로드 링크**를 제공하기도 합니다 [18]. 예를 들어, "보고서가 생성되었습니다. (파일: 2025-10-12_report.md)" 식으로 파일명이 표시될 수 있습니다 [101] [102]. 파일명이 나온다면 다음 단계인 Export 없이도 해당 Markdown을 다운로드할 수도 있습니다. (하지만 보통은 Export 버튼을 통해 받게 됩니다.)

주의사항: - 하루에 **한 번만 생성**하면 됩니다. 같은 날에 여러 번 "일일 보고서 생성"을 누르면 중복된 보고서 파일이 만들어질 수 있으므로 유의하세요. 만약 실수로 두 번 눌렀다면, 시스템 설정에 따라 두 번째 시도를 무시하거나 기존 파일을 덮어썼을 수 있습니다 [20]. 가능하면 한 번 눌러 성공 메시지를 받았으면 다시 누르지 않도록 합니다. - 보고서 생성이 실패하는 경우(예: "보고서 생성 실패: 데이터 부족" 등의 경고가 뜨는 경우), 이는 **해당 날짜에 수집된 뉴스 데이터가 충분하지 않거나 오류가 발생**했음을 뜻합니다 [19]. 이때는 크롤러나 데이터 수집 부분을 확인해야 합니다. 운영자 입장에서는 우선 **뉴스 수집 작업이 완료되었는지** 확인해야 합니다. (예: 수집된 기사 수가 0이면 실패합니다.) 필요하면 개발팀에 연락하여 원인을 파악하고 다음 조치를 받습니다. - 보고서가 성공 생성되면, 내부적으로 서버 `archive/` 폴더에 `YYYY-MM-DD_report.md` 이름의 파일이 저장됩니다 [12]. 이 파일은 Export 전에 임시로 저장된 것입니다. Cloud Run 환경에서는 이 파일이 영구 저장되지 않을 수 있으므로, **반드시 Export 단계에서 파일을 내려받아 보관**해야 합니다 (뒤에서 설명).

뉴스 요약 및 번역 – “전체 요약” 기능

역할: 모든 수집된 기사에 대한 요약문과 번역문을 생성하는 기능입니다. 관리자 UI에서는 보통 **「전체 요약」** 또는 **「키워드 전체 요약」** 등의 버튼으로 제공되며, 누르면 AI가 해당 키워드의 모든 기사 콘텐츠를 요약해 줍니다. 이 기능을 통해 운영자는 각 기사들의 핵심을 빠르게 파악하고, 이후 어떤 기사를 최종 발행할지 결정하는 데 도움을 받을 수 있습니다.

사용 방법: 「전체 요약」 버튼을 클릭하면 백엔드에서 금일 키워드의 모든 기사 본문을 순차적으로 AI에게 보내 요약을 받습니다 [21] . 또한 영어 기사 등은 한국어로 번역문도 생성합니다 (외부 번역 API 키가 설정된 경우) [21] . 이 작업은 기사 개수에 따라 **몇 초에서 수십 초**까지 걸릴 수 있습니다. UI는 이 동안 **"요약 진행중..."**과 같은 표시를 해줄 수 있으며, 작업 완료 전까지 다른 작업을 병행하지 않는 것이 좋습니다. 완료되면 **"요약 완료"** 혹은 **"전체 기사 요약이 완료되었습니다"** 같은 메시지가 표시됩니다.

결과 확인: - 전체 요약이 완료되면, **기사 리스트에 요약 결과가 표시**될 수 있습니다 [28] . 예를 들어 관리자 대시보드에 각 기사가 카드 형태로 나열돼 있다면, 제목 아래 작은 글씨로 요약문이 나타나도록 개발되었습니다. 혹은 별도의 팝업/모달을 통해 전체 요약 결과를 한꺼번에 볼 수 있도록 구현되었을 수도 있습니다. (만약 현재 UI에 이런 기능이 없다면, 요약 결과가 바로 눈에 보이지 않을 수 있는데, 이는 개선 예정 사항입니다.) - 요약 결과에는 자동 번역된 텍스트가 포함될 수 있습니다. 즉, 원문이 영어 기사라면 요약문은 영어로 나온 뒤 추가로 한국어 번역문이 붙어 있을 수도 있습니다. 개발 설정에 따라, 요약문 뒤에 괄호로 번역문을 붙여준다거나 하는 방식일 수 있습니다.

사용 의도: 전체 요약 기능은 **당일 수집된 모든 기사**의 내용을 개괄적으로 파악하기 위한 것입니다. 운영자는 이를 통해 기사들의 중요도나 신선도를 판단하고, 어떤 것을 **발행할지 선별(selection)**할지 결정할 수 있습니다. 또한 한눈에 여러 기사 내용을 비교함으로써 중복되는 소식은 제외하고, 핵심만 다룰 수 있습니다.

주의사항: - **API 키 한도:** 이 기능은 OpenAI나 Papago API 호출을 기사 수 만큼 반복 수행하므로, API 호출량이 많습니다. 만약 누르셨을 때 실패한다면, API 키의 호출 제한에 도달했을 수 있습니다. 예를 들어 Papago 무료 사용량 초과 시 번역이 안 되거나 오류가 발생할 수 있습니다. 이 경우 요약문은 원어로 남고 번역은 빈 상태일 수 있습니다 [103] . 가능하면 월간 할당량을 확인하면서 사용하고, 필요시 개발팀에 이야기하여 API 키를 교체하거나 유료 플랜으로 업그레이드합니다. - **실행 빈도:** 하루에 여러 번 누를 필요는 없습니다. 오전에 뉴스 수집 후 한 번 전체 요약을 실행하고, 그 날 추가로 더 많은 뉴스가 들어오지 않는다면 재실행하지 않아도 됩니다. (재실행하면 새로운 요약문으로 덮어쓸 뿐입니다.) - **요약 정확도:** AI가 만들어준 요약이므로, 내용상 부정확하거나 어색한 번역이 있을 수 있습니다. **최종 발행 전 반드시 사람이 검토**해야 하며, 잘못된 부분이 있으면 수정하거나 해당 기사를 제외하는 판단을 내려야 합니다.

뉴스 요약 및 번역 – “선정본 요약” 기능

역할: 최종 선정된 기사들에 대한 요약/번역문을 생성하는 기능입니다. UI에서는 **「선정본 요약」** 또는 **「선정 기사 요약」** 버튼으로 표시됩니다. 이 버튼은 운영자가 여러 기사 중 최종 발행할 기사들을 **선정(mark as selected)**한 뒤에 누르게 되어 있습니다. 선택된 기사들만을 대상으로 다시 한 번 요약/번역을 실행하여, **발행에 사용할 최종 콘텐츠를 다듬는 단계**입니다.

사용 방법: 먼저 관리자 화면에서 오늘 발행할 기사들을 선택해야 합니다. 일반적으로 기사 리스트 옆에 체크박스나 "승인" 버튼이 있어, 운영자가 읽어보고 괜찮은 기사들을 **승인/선정** 표시합니다. 이렇게 선택된 기사들이 정해지면, **「선정본 요약」** 버튼을 클릭합니다. 그러면 백엔드에서 선택된 기사들의 본문만 골라 AI 요약/번역을 수행합니다 [33] .

- 이 과정도 "전체 요약"과 유사하게 시간이 걸리지만, 기사 수가 더 적으므로 비교적 빠르게 완료됩니다. (예: 5개 기사 선정 시 5건 요약/번역 실행)
- 완료되면 **"선정된 기사 요약 완료"** 등의 메시지가 표시됩니다. UI에서 특별한 변화를 못 느낄 수도 있으나, 사실 백엔드 데이터에 각 선정 기사들의 `summary` 와 `translation` 필드가 채워지게 됩니다.

결과 활용: 선정본 요약의 주요 목적은 **최종 발행본에 번역문을 포함시키는 것**입니다 [35] . 예를 들어, 최종 Markdown 보고서를 작성할 때 영문 기사 제목 아래에 한국어 번역 요약문을 덧붙이는 식으로 콘텐츠를 구성할 수 있습니다. 실제

Admin 시스템에서는 선정본 요약이 완료된 후: - 곧바로 **Markdown 보고서 생성**을 하면, 해당 번역문이 그 보고서에 포함되도록 구현되었을 수 있습니다 [35]. (개발 설정에 따라, `make_daily_report`가 selection 요약 결과를 참조하여 Markdown을 작성하도록 되어 있을 수 있습니다.) - 또는 운영자가 Export로 Markdown을 내려받은 후 그 안에 번역문이 포함되어 나올 것입니다 [35].

그러므로 **이 버튼은 보고서 생성 직전에 누르는 것이 좋습니다.** 전체 요약을 통해 윤곽을 잡고 기사들을 선택한 다음, 선정본 요약을 실행하여 최종적으로 번역문까지 준비해두는 것입니다. 이후 보고서를 생성하거나 Export하면 번역이 반영된 완성본이 나오게 됩니다 [104].

주의사항: - **선정 단계 필요:** 이 기능은 어디까지나 "선택된 기사들만" 대상으로 동작하므로, 사전에 어떤 기사를 발행할지 결정하여 체크해두는 과정이 필요합니다. 만약 아무 기사도 선택 안 한 상태에서 누르면 아무 일도 일어나지 않거나 오류가 날 수 있으니, 반드시 대상 기사를 선택하세요. - **번역 검토:** 번역문이 자동 생성되므로, 자연스럽지 않은 문장이 있을 수 있습니다. 운영자는 Export 단계에서 Markdown을 열어보거나, UI상에서 번역 결과를 확인할 수 있다면 확인하여 **부자연스러운 문장은 편집**하는 것이 좋습니다 [36]. (예: 어미가 이상하다면 고치거나, 중요한 뉘앙스가 제대로 번역되지 않았으면 수동 수정.) - **API 동일 처리:** 내부적으로 selection 요약도 전체 요약과 같은 AI 엔진을 사용하므로, 앞서 언급한 API 사용량 문제에 유의합니다. 다만 기사 수가 적어 부담은 크지 않습니다.

결과물 Export (Markdown/CSV 내보내기)

역할: 생성된 데일리 뉴스 보고서를 **파일로 다운로드하거나 외부로 내보내는 기능**입니다. 관리자 UI에서 **「Export Markdown」** 버튼과 **「Export CSV」** 버튼 두 가지로 제공됩니다. Markdown 파일은 사람이 읽거나 그대로 뉴스레터로 발송할 수 있는 형식이고, CSV 파일은 정형화된 표 형식 데이터로서 사내 백업이나 분석 용도로 사용할 수 있습니다 [43].

사용 방법: - **Markdown 내보내기: 「Export Markdown」** 버튼을 클릭하면, 브라우저가 새 탭을 열어 Markdown 텍스트를 표시하거나 바로 `.md` 파일 다운로드를 시작합니다 [44] [45]. 구현에 따라 둘 중 하나로 동작할 수 있는데, 만약 새 탭으로 열리는 경우 사용자는 해당 탭에서 내용을 확인한 후 직접 저장할 수 있고, 자동 다운로드인 경우 곧바로 PC에 파일이 저장됩니다. 파일명은 보통 `YYYY-MM-DD_report.md` 형태로 내려받을 것입니다 (예: `2025-10-12_report.md`) [40]. - **CSV 내보내기: 「Export CSV」** 버튼을 누르면, 즉시 `.csv` 파일 다운로드가 시작됩니다 [45]. CSV는 일반적으로 Excel로 열 수 있는 콤마구분값 파일이며, 파일명은 `latest_report.csv` 혹은 `YYYY-MM-DD_report.csv`로 지정될 것입니다. 다운로드가 완료되면 PC의 다운로드 폴더에서 해당 파일을 확인할 수 있습니다.

Export 내용: - Markdown 파일에는 당일 발행 뉴스 콘텐츠가 Markdown 문법으로 작성되어 있습니다. 예컨대, 제목, 기사 리스트(번호 매겨진 목록), 각 기사 제목 및 링크, 요약문 등이 포함된 구조입니다. 만약 번역이 적용되었다면 요약문 아래에 번역문이 포함되어 있을 수 있습니다. 이 파일은 그대로 **사내 이메일이나 슬랙 등에 공유**하거나, Markdown을 지원하는 뷰어로 열어볼 수 있습니다. - CSV 파일에는 일반적으로 기사들의 속성이 표 형태로 들어갑니다. 각 행이 한 기사에 대응되고, 컬럼으로 날짜, 키워드, 기사제목, 요약문, 번역문, 원문 URL 등이 포함될 수 있습니다. 이 포맷은 내부 분석(예: 어떤 키워드에 어떤 기사가 많았는지 통계)이나 DB 적재 등의 용도로 활용 가능합니다 [43]. 정확한 컬럼 구성은 개발팀과 합의된 스키마에 따릅니다.

일일 백업 및 활용: Export 기능은 운영자가 **매일의 발행 결과물을 외부에 보관**하기 위한 핵심 절차입니다. Cloud Run 서버 자체에는 파일이 영구 보관되지 않으므로, **반드시 Markdown과 CSV를 로컬에 다운로드 받아 사내 보관하세요** [83] [84]. 예를 들어 다운로드 받은 파일들을 사내 공유 폴더나 Google Drive 등에 날짜별로 정리해두면, 나중에 과거 뉴스를 찾아보거나 문제 발생 시 복구하는 데 도움이 됩니다 [83].

또한 Markdown 파일은 그대로 뉴스레터 발송에 쓸 수도 있습니다. 필요한 경우 Markdown을 HTML로 변환하여 웹에 게시하거나, 일부 편집을 거쳐 최종 컨텐츠로 만들 수 있습니다. CSV는 Excel로 열어서 빠르게 내용 확인하거나 수치 분석을 하는 용도로 사용할 수 있습니다.

주의사항: - Export 버튼을 눌렀는데 다운로드가 안 될 경우, **브라우저가 팝업/다운로드를 차단**한 것인지 확인해주세요. 새 탭을 열려고 할 때 브라우저가 막으면 아무런 반응이 없을 수 있습니다. 이때 주소창 오른쪽에 뜨는 팝업 차단 아이콘을 눌러 허용하면 다시 시도할 수 있습니다. - 파일이 다운로드 되었지만 **열 때 글자 깨짐** 등이 발생하면, 인코딩 문제일 수 있습니다. 개발은 UTF-8로 되어 있으므로 Excel에서 CSV를 불러올 때 UTF-8로 인코딩 설정을 하거나, 메모장으로 연 후 다른 이름 저장 시 ANSI->UTF-8로 변경하는 방법 등이 있습니다. Markdown은 일반 텍스트 파일이므로 메모장이나 VSCode 등으로 열면 깨짐 없이 보일 것입니다. - Export 실행 전에 반드시 **보고서 생성과 요약 작업이 완료**되었는지 확인하세요. 그렇지 않으면 Markdown에 내용이 비거나 최신이 아닐 수 있습니다. 일반적인 순서는 보고서 생성 -> (전체 요약/선정 요약) -> Export 입니다. 만약 순서를 바꾸면 Export에 반영되지 않을 수 있으니 유의합니다.

운영 팁 및 주의사항

마지막으로, 관리자 시스템을 운영하면서 알아두면 좋은 팁과 유의사항을 정리합니다:

- **데이터 수집 확인:** 보고서 생성이나 요약을 하기 전에, **해당 날짜 키워드의 뉴스 수집 작업이 완료**되었는지 확인하세요 [105] . 보통 크롤러가 자동으로 기사를 모아둘 텐데, 간혹 수집이 실패하면 기사 수가 0일 수 있습니다. 이때 바로 보고서 생성하면 실패하므로, 우선 수집된 기사 개수를 점검합니다 (관리자 대시보드에 오늘 기사 수가 표시되거나 `/api/status`로 확인 가능). 최소 게이트 기준(예: 15건)이 충족되지 않으면 보고서 퀄리티가 떨어지니, 필요한 경우 추가 수집이나 키워드 변경을 검토합니다 [106] .

- **기능별 진행 상황:** "전체 요약"이나 "선정본 요약"은 누르면 UI가 바로 응답하지 않고 **수 초에서 수십 초 후 완료**됩니다. 따라서 **한 번 누른 후 재차 클릭하지 말고 기다립니다.** 중복 클릭한다고 빨라지지 않고 오히려 서버에 부하만 줄 수 있습니다. 진행 도중 다른 버튼을 누르면 내부 데이터가 일관되지 않을 수 있으니, 가능하면 순차적으로 한 작업씩 처리하세요 [107] . (예: 전체 요약이 끝나기 전에 Export를 누르면 요약반영이 안 된 내용을 받을 수 있습니다.)

- **출력물 검토:** AI가 생성한 **요약문과 번역문을 사람이 최종 검토**하는 것은 필수입니다. 관리자는 Export한 Markdown 파일을 열어 **번역이 자연스러운지, 중요한 내용 누락은 없는지** 등을 확인해야 합니다 [36] . 필요하다면 Markdown 파일을 편집기에서 열어 직접 문구를 다듬습니다. (Markdown은 텍스트이므로 바로 수정 후 저장하면 됩니다.) 특히 역사 관련 콘텐츠라면 사실 관계가 맞는지, 번역 어투가 적절한지 살펴보세요.

- **토큰 유지 및 재발급:** Admin Token은 현재 브라우저 로컬스토리지 등에 저장되어 인증에 쓰입니다. **브라우저 캐시/저장소를 지우거나 다른 브라우저를 쓰면 토큰을 다시 설정해야** 합니다. 토큰을 분실했거나 노출되었다면 개발팀에 요청해 새 토큰을 발급받으세요. 새 토큰이 나오면 .env와 Cloud Run 환경에 업데이트한 뒤 운영자 브라우저에도 갱신해야 합니다 [82] . 한편, **401 Unauthorized 에러가 계속 발생**한다면 토큰 불일치가 의심되므로 이 역시 개발팀과 상의해 해결합니다 [82] .

- **로그 모니터링:** 운영 중 특별히 UI에 문제는 없어 보여도, 백엔드에서는 에러가 발생했을 수 있습니다. **Cloud Run의 Logs 탭**에서 시간별 로그를 확인하여 ERROR 수준의 로그가 있는지 수시로 봐주세요 [94] . (예: 번역 API 오류, 파일쓰기 오류 등) 문제가 누적되면 향후 기능에 지장 줄 수 있으므로, 발견 즉시 개발팀과 공유합니다.

- **백업 철칙:** 내보낸 Markdown과 CSV 파일은 반드시 **안전한 곳에 보관**합니다 [83] . 로컬 PC에만 두지 말고 사내 공용 드라이브나 클라우드 스토리지에 날짜별 폴더를 만들어 업로드해두면 좋습니다. 이렇게 하면 Cloud Run 로그나 데이터베이스 없이도 과거 기록을 쉽게 찾을 수 있고, 혹시 모를 분실에 대비할 수 있습니다 [83] [84] . (Cloud Run 컨테이너는 재시작하면 archive 폴더 파일을 잃으므로, 남아있지 않을 수 있음을 기억하세요.)

- **시스템 업데이트 공지:** 개발팀이 새로운 기능을 추가하거나 수정사항을 배포할 때는 릴리스 노트를 공유할 것입니다. 운영자는 이를 숙지하고 **UI 변동사항이나 새로운 버튼의 사용법**을 파악해야 합니다. 또한 혹시 문서나 안내에 없는 버튼/기능이 보이면, 업데이트된 가이드가 있는지 확인하거나 개발팀에 문의하여 교육을 받습니다

96 . 예컨대 "임계값 조정" 슬라이더나 "재발행" 버튼 등이 새로 생길 수 있으니, 메뉴얼 최신 버전을 항상 참고합니다.

주요 버튼 기능 요약

마지막으로, 관리자 페이지에서 사용하는 **주요 버튼들과 그 역할**을 한눈에 정리하면 다음과 같습니다:

- **일일 보고서 생성:** 당일 수집된 뉴스로 Markdown 형식의 데일리 보고서를 생성합니다 12 . (하루 한 번 실행)
- **전체 요약:** 오늘 수집된 **모든 기사**의 요약과 번역문을 생성합니다 21 . (기사 선별 전에 실행하여 개략을 파악)
- **선정본 요약:** 운영자가 **선택(승인)한 기사들만** 모아 요약/번역을 생성합니다 33 . (최종 발행 직전에 실행하여 번역문 준비)
- **Export Markdown:** 최신 보고서의 내용을 Markdown 파일로 다운로드합니다 108 . (파일명에 날짜 포함, 발행본 백업/공유 용도)
- **Export CSV:** 최신 보고서의 데이터를 CSV 파일로 다운로드합니다 108 . (테이블 형식 데이터 백업/분석 용도)

以上의 기능들을 순서대로 사용하면, **“뉴스 수집 완료 -> 전체 요약 -> 기사 선택 -> 선정본 요약 -> 보고서 생성 -> Export”**의 흐름으로 하루 발행 작업이 마무리됩니다. 각 단계에서 의문사항이나 문제가 발생하면 이 가이드북을 참고하거나 개발팀에 문의하세요. 퀄리저널 관리자 시스템을 통해 효율적이고 체계적인 뉴스 큐레이션 작업을 수행하시길 바랍니다. 43 96

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 55 56 58 59 62 64 65 67 68 69 71 72 73 74 75 76 82 83 84 86 87 88 89 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 1013_1QualiJournal 관리자 시스템 최종 통합 작업 가이드북.pdf

file://file-6DfuAdPY3q666RS3cV5ydM

54 57 60 61 63 66 70 77 78 79 80 81 85 90 91 1012_1QualiJournal 관리자 시스템 최종 단계별 작업 가이드북.pdf

file://file-8wm4iV9GGuoV7FL4ABABAZ