ChatGPT

퀄리뉴스 "키워드 뉴스" 기능 설계 및 구현 전략

개요: 편집자가 하루에 하나의 키워드를 입력하면 그 키워드와 관련된 **공식 소스 기사**와 **커뮤니티 게시글**을 폭넓게 수집하고, **최소 15건 이상의 고품질 기사 카드**로 구성된 특별 호를 발행하는 기능입니다. 이 키워드 뉴스는 기존의 일일 뉴스 병합 발행과 별도로 동작하는 **새로운 발행 라인**으로서, 수집부터 편집, 발행까지 전 과정을 키워드 중심으로 처리합니다. 아래에서는 요구사항을 반영한 상세 설계와 구현 전략을 항목별로 정리합니다.

키워드 기반 기사 수집 흐름

1. 수집 모드 트리거: 편집자가 관리 화면이나 설정 파일을 통해 **키워드 문자열**을 입력하고 수집을 요청하면, 시스템은 키워드 뉴스 전용 수집 모드를 시작합니다. 이때 새로운 CLI 인자 (예: `--collect-keyword "<키워드>"`)나 관리자 UI 액션을 통해 수집기를 실행하며, 기존 정기 크롤링과는 별개로 **수동으로 한번만 실행**되도록 합니다.

2. 공식 소스 필터링: `smt_sources.json`, `supply_sources.json` 등 등록된 **공식 전문 소스** 전체를 대상으로 각 소스의 최신 목록을 가져와 크롤링합니다. 가져온 **기사 목록에서 제목(title)이나 요약(description)에 키워드가 포함된 항목만** 추려냅니다. 예를 들어 키워드가 `"리플로우"`라면 해당 단어가 기사 제목이나 소개에 등장하는지 검사하여, 포함된 기사만 상세 크롤링 대상으로 선정합니다. 이렇게 1차 필터를 적용함으로써 키워드와 무관한 최신 기사는 건너뛰어 효율을 높입니다. 다만 키워드가 본문에만 언급되고 제목에는 없을 수도 있으므로, **가능하면 본문 미리보기(요약)**에도 키워드가 있는지 확인합니다 (제목과 요약에 없으면 해당 기사 생략). 이 단계에서는 **대소문자 무시** 및 **한글/영문의 동의어 고려**도 필요합니다. 예컨대 편집자가 한글 키워드를 입력했을 경우 영문 동의어나 약어로도 검색하는 기능을 추가하여, 영문 기사도 포착할 수 있게 설계합니다.

3. 과거 기사 탐색 (필요시): 공식 소스 목록에 최근 키워드 관련 기사가 충분하지 않은 경우, **사이트 내부 검색**이나 **외부 뉴스 검색 엔진**을 활용해 해당 키워드의 과거 기사를 발굴합니다. 예를 들어 **Google News API나 Bing News API**로 `"키워드 + 뉴스"` 쿼리를 자동 실행하여 상위 결과 기사를 추가 수집할 수 있습니다 [1]. 이때 **신뢰도 낮은 도메인**은 제외하고, industryweek.com, electronicsweekly.com 등 신뢰할 만한 도메인 위주로 포함하는 **도메인 필터**를 적용합니다 [2]. 또한 **Google Alerts**를 활용해 해당 키워드의 RSS 피드를 미리 받아두고 읽는 방법도 고려합니다 [1]. (단, 이러한 외부 검색 통합은 초기 구현 복잡도가 높으므로 필요한 경우에 한해 2차 개발 단계로 검토합니다.)

4. 커뮤니티 소스 수집: 기존에 설정된 **커뮤니티 소스**(예: Reddit API 대상 서브레딧, 전자공학 포럼 등)에도 키워드 필터를 확장합니다. Reddit의 경우, 선택된 키워드를 포함하는 글을 검색하도록 **동적 쿼리**를 수행합니다. 예를 들어 `r/electronics`, `r/SMT` 등의 서브레딧에서 `"키워드"`로 검색하여 관련 포스트를 수집하거나, Reddit 전체 검색 API로 키워드 관련 인기 글을 조회합니다 [3]. 포럼의 경우도 현재 EEWorld 등에서 `"IPC"`, `"NASA"` 등의 키워드로 필터링하고 있다면 [3], 여기에 입력된 키워드 조건을 추가하여 **제목이나 본문에 해당 키워드가 포함된 게시글만 크롤링**합니다. 이를 위해 정규식 필터를 이용해 키워드가 텍스트에 등장하는지 검사하고, 부합하는 커뮤니티 글을 수집 목록에 추가합니다.

5. 기사 본문 수집: 위 단계들을 통해 수집 대상이 된 **모든 기사 URL에 대해 본문을 크롤링**합니다. 공식 뉴스 기사, 블로그 글, 포럼 글 등 다양한 형식이 섞일 수 있으므로, 기존 `_get()` 함수를 활용하여 HTML을 가져오고 **본문 주요 텍스트를 파싱**합니다 [4]. 본문 추출 로직은 기존 일일 수집과 동일하게 적용하되, **문서 유형별 파싱기**를 개선하여 기술 문서(PDF)나 제품 소개 페이지에서도 주요 내용 텍스트를 추출하도록 고려합니다 (예: <code><article></code> 태그 외에 블로그 특성 태그, 포럼 특성 태그 등 추가 처리).

6. 품질 게이트(QG) 적용: 크롤링된 각 기사에 대해 **품질 게이트 검사**를 수행하여 저품질/무의미한 기사를 걸러냅니다 [5]. 품질 게이트 기준은 기존과 동일하게 적용하되, 키워드 뉴스의 특성상 오래된 기사나 커뮤니티 글도 포함될 수 있으므로 **날짜에 의한 필터링은 하지 않습니다**. 주요 기준은 본문 길이 (예: 700자 이상), 문장 수 (6문장 이상), 하이퍼링

크 수 (2개 이상) 등이며, 기준 미달 시 "품질 미달(REJECT)"로 탈락시킵니다 [5]. 또한 **로그인 필요/구독 유도/오류 페이지** 등의 금지 요소가 감지되면 즉시 제외합니다 [6]. 이 단계에서 **신뢰도 높은 도메인**의 기사는 약간 점수를 가산하거나 (trust 값) 경계선의 경우 통과시켜줄 수도 있습니다 [7]. 키워드 검색으로 인해 맥락상 관계 없는 잡음 기사나 광고성 콘텐츠도 수집될 수 있으므로, 품질 게이트를 통해 1차적으로 자동 정화를 수행하는 것이 중요합니다 [8].

7. 팩트체크(FC) 모듈 적용: 품질 기준을 통과한 기사들에 대해 **팩트체크 스코어링**을 진행합니다. 기사 내용에 포함된 **객관적 근거 요소**(숫자 데이터, 날짜/연도, 표준 규격 코드, 외부 링크, PDF 첨부 등)를 탐지하여 **신뢰 점수**를 계산합니다 [9]. 예를 들어 숫자/통계의 존재, 권위 있는 기관의 외부링크나 공식 문서(PDF)의 언급 횟수 등에 가중치를 부여합니다 [9]. 일정 기준 이상 (예: 근거 종류 2가지 이상 & 점수 50점 이상)이면 **PASS**, 미달하면 **FAIL 또는 HOLD**로 표시합니다 [10]. 키워드 기반으로 수집된 기사 중에는 과장되거나 추측성 글도 있을 수 있으므로, 이 팩트체크 단계에서 **근거가 부족한 기사는 자동 보류/제외** 처리하여 신뢰성을 담보합니다 [8]. (엄격 모드일 경우 FAIL이면 리스트에서 제외하고, 완화 모드일 경우 HOLD로 둬 편집자가 추가 확인토록 합니다.)

8. 키워드 매칭 및 기타 스코어 계산: 마지막으로 각 후보 기사에 대해 **키워드 관련 점수와 종합 점수**를 계산합니다. 우선 **키워드 매칭 강도**를 정량화하여 점수화합니다. 예를 들어 **제목에 키워드가 직접 포함된 경우** 가점을 높게(+X점) 부여하고, 본문에 여러 차례 등장하면 빈도(kw_hits)에 따라 추가 가중치를 줍니다. 반대로 키워드가 한번 언급되거나 맥락과 관계 없으면 가점을 낮게 책정합니다. 동시에 **도메인 신뢰도 점수**를 부여합니다. 편집자가 사전에 정의한 신뢰 높은 공식 출처 (예: `ipc.org`, `openai.com`, `iconnect007.com` 등)에 대해서는 **+5~+10점 기본점수**를 주고, 불분명한 개인 블로그나 중복 가능성이 큰 출처는 0 또는 음수 가중치를 줄 수 있습니다 [11]. 이 도메인별 가중치는 `editor_rules.json`에 설정하여 사용합니다. 또한 **여러 매체에 등장한 동일 주제 기사**인지도 체크하여, 만약 수집된 후보 중 **제목이 유사한 기사가 2곳 이상** 발견되면 해당 주제가 업계 이슈로 판단되어 추가 가점을 주는 것도 고려합니다 [12]. 마지막으로 품질 게이트 통과 여부와 팩트체크 점수도 참고하여, **종합 점수(total_score)**를 계산합니다. (예시: `total_score = kw_score + domain_score + fc_score + bonus`). 이렇게 계산된 종합 점수를 각 기사에 부여하고, 추후 관리 화면에서 **우선순위 정렬** 및 **자동 추천**에 활용합니다 [13].

9. 결과 저장: 모든 후보 기사에 대한 메타데이터와 점수를 취합하여 **키워드 뉴스 임시 선정 파일**에 저장합니다. 이 단계에서는 아직 어떤 기사를 발행할지 최종 결정되지 않았으므로, **모든 통과 기사**를 리스트에 포함합니다. 이 파일에는 각 기사별 주요 정보(제목, 출처, URL, 요약 등)와 함께 앞서 계산한 QG/FC 상태, 키워드 매칭 횟수, 도메인 점수, 종합 점수 등을 기록합니다. 기본적으로 `selected_keyword_articles.json`이라는 JSON 파일을 사용하되, 날짜별로 별도 관리할 필요가 있다면 **예:** `selected_keyword_2025-10-03.json` 형태로 저장할 수도 있습니다. 이렇게 생성된 JSON은 다음 단계인 편집자 **승인 및 발행 단계**에서 활용됩니다.

> **참고:** 키워드 기반 수집은 일반 수집보다 많은 잡음과 옛날 기사가 섞일 가능성이 있으므로, **자동화된 품질/팩트 필터링과 편집자 확인 작업**이 필수적입니다 [8]. 적절한 키워드 선택과 튜닝, 그리고 위의 다중 단계 필터를 통해 **최소 15건 이상의 양질의 기사**를 확보하도록 설계합니다. 만약 최초 수집 결과 15건 미만이라면, 키워드 범위를 넓히거나(동의어 사용 등) 추가 수집(예: 외부 검색) 절차를 한 번 더 수행해 보강합니다.

기사 가치 판단 기준 (스코어링 기준)

키워드 뉴스의 후보 기사들을 평가하고 우선순위를 정하는 데에는 **키워드 매칭 정도 외에도 다양한 점수 요소**를 종합 고려합니다. 주요 판단 기준은 다음과 같습니다:

- **키워드 매칭 강도:** 입력한 키워드가 기사에 **얼마나 밀접히 언급**되었는지를 평가합니다. 제목에 키워드가 직접 포함되면 가장 높은 가중치를 주고, 본문에 여러 차례 등장하거나 키워드와 동의어/관련어가 반복되면 추가 점수를 부여합니다. 예를 들어 키워드 "AI 칩"의 경우 제목에 `"AI 칩"`이 있으면 +3점, 본문에 여러 번 나오면 횟수에 비례한 점수를 더합니다 [14]. 반대로 키워드가 한 번 지나가는 언급에 그치거나 본문과 큰 관련이 없으면 낮은 점수로 책정합니다. 이 **키워드 관련 점수(kw_score)**는 해당 기사의 키워드 적합도를 나타냅니다.

- **출처 도메인 신뢰도:** 기사 출처의 신뢰도를 점수에 반영합니다. `editor_rules.json`에 **주요 도메인별 기본점수 표**를 정의하여, 신뢰도 높은 공식 출처는 +5~+10점의 가중치를 부여하고, 신뢰도가 낮거나 광고성 우려가 있는 도메인은 0점 또는 음수로 페널티를 줍니다 [11]. 예를 들어 업계 표준단체나 유명 전문 매체(IPC, IEEE, iConnect007 등)의 기사는 기본점수를 높게 주어 우선 검토 대상으로 삼습니다. 이 도메인 점수(domain_score)는 품질 게이트 통과 여부와 별개로 해당 기사의 **선호도**를 결정하는 요소입니다.

- **품질 게이트(QG) 상태:** 품질 게이트에서 **PASS/HOLD**로 통과한 기사는 일정 점수를 확보하지만, **REJECT 판정**(예: 너무 짧음, 링크 없음 등)을 받은 기사는 애초에 리스트에 포함되지 않습니다 [5]. 다만 HOLD(보류)의 경우 기준에 살짝 못 미친 것이므로 약간 낮은 점수를 주어 후보에 남겨둘 수 있습니다. QG 관련 지표(본문 글자수, 문장 수 등)는 별도 필드로 저장하고, 종합 점수 산정 시 임계값 이상이면 가점을, 미달이면 제외하는 식으로 활용합니다. 이로써 **기사의 기본 완성도**를 반영합니다.

- **팩트체크(FC) 점수:** 팩트체크 모듈이 산출한 **신뢰도 점수**도 반영합니다. 객관적 근거가 풍부한 기사(숫자, 출처 링크, 공식 문서 언급 등 많음)는 높은 FC 점수를 받고, 추측성/홍보성 글은 낮은 점수를 받습니다 [9]. 기본적으로 FC 점수가 기준 이상(PASS)인 기사만 최종 후보로 남기지만, 만약 HOLD 처리된 경우 편집자 검토를 위해 리스트에 남겨둘 수도 있습니다. 종합 점수 계산 시 FC 점수를 가중 합산하여, **근거가 탄탄한 기사**가 상위에 오르도록 합니다. 예컨대 FC 점수 60점을 받은 기사는 60의 일부를 total_score에 반영하거나, PASS 여부를 이진 가산점으로 줄 수 있습니다.

- **중복 및 이슈성:** 동일 키워드에 대해 **여러 출처에서 유사한 뉴스가 나온 경우** 해당 토픽이 중요하거나 화제임을 의미하므로, 그런 기사는 추가 가점을 받을 수 있습니다 [12]. 구현상으로 후보 목록에서 **제목의 유사도**를 비교해 두 개 이상 매치되면 +N점을 부여합니다. (예: "신제품 A 출시" 관련 기사가 3곳에서 발견되면 각각에 +2점씩 부여). 이를 통해 업계 **핫토픽**이 키워드 뉴스에서 더 부각되도록 합니다.

- **커뮤니티 반응도 (선택 요소):** 커뮤니티 출처(예: Reddit)의 글일 경우 **업보트 수, 댓글 수** 등이 그 글의 유용성 척도가 될 수 있습니다. 가능하다면 Reddit API 등을 통해 해당 글의 score나 comment count를 가져와 점수에 반영합니다. 예를 들어 업보트 100+인 토론 글이면 +2점, 댓글 활발하면 추가 +1점 등으로 가중치를 줄 수 있습니다. 이는 **커뮤니티에서의 관심도**를 반영하여, 단순 정보성 글보다 업계 종사자들이 주목한 글을 선별하는 데 도움을 줍니다.

以上의 기준으로 산출된 **종합 스코어(total_score)**는 각 기사 객체에 저장되며, 기본적으로 이 값을 통해 **후보 리스트를 정렬**합니다 (점수 내림차순). 다만 최종 발행 여부는 **편집자의 판단**에 따라 이루어지므로, 점수는 참고자료로 활용합니다. 편집자는 필요시 점수와 무관하게 중요하다고 생각되는 기사를 선택할 수 있으나, 스코어링 체계는 전반적으로 **키워드와 연관성 높고 신뢰성 있는 기사들이 상위에 위치**하도록 하여 편집 작업을 보조합니다 [13].

수집 결과 정리 방식 (selected_keyword_articles 구조)

키워드 뉴스 수집이 완료되면, 결과를 편집자가 검토하고 발행에 사용할 수 있도록 **후보 기사 모음 파일**을 생성합니다. 현행 퀄리뉴스에서는 `selected_articles.json`을 사용해 일일 기사 후보와 선택 여부를 관리하므로, 키워드 뉴스에는 별도의 파일(예: `selected_keyword_articles.json`)을 도입합니다. 이 파일의 구조 설계를 제안하면 다음과 같습니다:

- 최상위에 **키워드(keyword)**와 **수집 일자(date)** 필드를 명시하여, 해당 파일이 어떤 키워드의 어떤 날짜 결과인지를 식별할 수 있게 합니다.
- **articles** 배열 안에 각 후보 기사 객체를 포함합니다. 객체 구조는 기존 `selected_articles.json`의 **article** 항목과 유사하게 설계하되, 필요한 필드를 추가/조정합니다:
- `id`: 기사 고유 ID (해시 등으로 생성)
- `title`: 기사 원문 제목 (영문일 경우 영문 제목)
- `ko_title`: (선택) 한글 번역/편집 제목. 기본값은 원문 제목과 동일하게 두고, 추후 번역시 갱신하거나 편집자가 수정 가능.

- `source`: 출처 명 (예: Meta AI Blog, Reddit/r/electronics 등)
- `url`: 기사 원문 URL
- `summary_ko_text`: 본문 요약 (또는 첫 몇 문장) - 수집 시 원문에서 추출한 요약문. 영문인 경우 추후 발행 단계에서 한글 번역 예정.
- `full_ko`: (옵션) 기사 전체 한글 번역문. 키워드 뉴스에서는 보통 전체 번역까지는 하지 않을 가능성이 높으므로 기본 `null`로 두고 필요시 채웁니다.
- `section`: 기사 분류(섹션) - 기존 시스템에서 사용하던 필드이며, SMT 뉴스/AI 뉴스/커뮤니티 등으로 자동 분류되지만, 키워드 뉴스에서는 **단일 테마**이므로 임의로 `"Keyword"` 등으로 지정하거나 혹은 원 소스 기반 분류를 유지할 수 있습니다. 초기 구현에서는 크게 중요하지 않아 하나의 섹션으로 묶어 처리해도 무방합니다.
- **품질/팩트 관련:** `qg_status` (예: "PASS"/"HOLD"), `qg_score` (품질점수), `fc_status` ("PASS"/"FAIL"), `fc_score` (팩트체크 점수) 등을 포함해 자동 필터 결과를 표시합니다.
- **키워드/도메인 관련:** `kw_hits` (키워드 등장 횟수), `kw_score` (키워드 매칭 점수), `domain_score` (도메인 가중치 점수) 필드를 추가하여, 해당 기사에 대한 키워드 적합성과 출처 신뢰도를 수치로 명시합니다 [11].
- `total_score`: 위의 다양한 점수를 합산한 최종 점수.
- **편집선정 관련:** `selected` (기본값 `false`, 편집자가 발행 선택시 `true`로 변경), `editor_note` (편집자 한마디 코멘트, 기본 빈 문자열) [15], `pinned`/`pin_ts` (상단 고정 여부, 고정시각 등 기존 UI에서 사용 중이면 그대로 포함).

예시 JSON 구조 (간략화):

```
{
 "keyword": "리플로우 솔더링",
 "date": "2025-10-03",
 "articles": [
  {
   "id": "ab12cd34ef...",
   "title": "Reflow Soldering Process Innovations in 2025",
   "ko_title": "2025년 리플로우 솔더링 공정 혁신",
   "source": "SMT Today",
   "url": "https://smttoday.com/reflow-innovations-2025",
   "summary_ko_text": "Reflow soldering has seen several innovations in 2025, including ...",
   "qg_status": "PASS", "qg_score": 8.5,
   "fc_status": "PASS", "fc_score": 60,
   "kw_hits": 3, "kw_score": 5,
   "domain_score": 7,
   "total_score": 80,
   "selected": false,
   "editor_note": ""
  },
  {...}, ...
 ]
}
```

이런 구조를 사용하면, **편집자는 UI에서 이 JSON을 불러와** 각 기사들의 정보와 점수를 검토할 수 있고, `selected` 플래그와 `editor_note`를 편집하여 최종 발행 목록을 확정하게 됩니다. 추후 발행 시에는 이 파일을 참조하여 `selected: true`인 기사만 사용하게 됩니다 [16].

파일 관리: `selected_keyword_articles.json`은 **현재 진행 중인 키워드 뉴스의 작업본**으로 사용합니다. 만약 한 번에 하나의 키워드 뉴스만 운영한다면 이 한 파일로 족합니다. 새로운 키워드 뉴스 수집이 시작되면 이 파일을 덮어쓰거나 백업(아카이브) 처리합니다. 과거 키워드 뉴스 기록을 보관하려면, 발행이 완료될 때 해당 JSON을 별도 경로에 복사하여 **아카이브**할 수 있습니다. 예컨대 `archive/keyword_news/2025-10-03_리플로우.json` 형태로 저장해 두면 나중에 다시 참고하거나 재발행할 때 활용 가능합니다. 이때 파일명에 한글 키워드가 들어갈 수 있으므로 **파일 시스템 호환성을 고려한 슬러그 변환**을 적용합니다 (예: "리플로우 솔더링" -> "reflow-soldering" 등 영문 슬러그 사용).

또한, 키워드 뉴스의 Markdown 및 HTML 아카이브 파일과 JSON을 **동일한 식별자**로 관리하면 편리합니다. 예를 들어 `2025-10-03_리플로우.md` / `.html` / `.json` 세트를 같은 폴더에 보관하면, 특정 날짜/키워드의 모든 산출물을 쉽게 찾을 수 있습니다.

관리자 UI 확장 방안 (승인 및 한마디 입력 UI)

키워드 뉴스 기능을 위해 **관리자 대시보드(UI)**에 새로운 화면 또는 기능을 추가해야 합니다. 현재 퀄리뉴스 관리자 화면에서는 일일 수집된 기사 목록을 보고 **발행 승인 및 편집자 코멘트 입력**을 수행하고 있을 것입니다. 여기에 **키워드 뉴스 전용 UI**를 병행하도록 설계합니다:

- **키워드 뉴스 관리 화면 추가:** 관리자 대시보드에 “키워드 뉴스” 메뉴 또는 탭을 신설합니다. 편집자가 이 메뉴로 들어오면, 최근 수집된 `selected_keyword_articles.json`의 내용을 불러와 **후보 기사 리스트**를 표시합니다. 일일 뉴스 UI와 유사한 테이블/카드 뷰 형식으로, 기사 제목, 출처, 점수, (요약) 등을 한눈에 볼 수 있게 합니다. 동시에 각 행에는 **체크박스**나 토글로 발행 선택(`selected`) 여부를 설정할 수 있고, **한줄 코멘트 입력란**(editor_note)을 제공하여 편집자가 해당 기사에 대한 한마디를 입력할 수 있게 합니다.

- **한마디 입력 UI:** 기사별로 한줄 코멘트를 입력하는 필드는 **기존 일일 뉴스 UI 구성과 동일한 형태**로 제공합니다. 예컨대 테이블 한 열에 작은 입력란을 두거나, 기사 카드를 클릭하면 코멘트 작성 모달이 뜨는 식입니다. 편집자가 놓치지 않고 코멘트를 남기도록, 빈칸일 경우 경고하거나 기본 문구를 넣는 등의 UX를 고려합니다. 예: `editor_rules.json`에 미리 정의한 코멘트 가이드라인이나 placeholder를 활용할 수 있습니다. 실제로 시스템에 `"editor_note_placeholder": "한 문장으로 핵심 시사점 입력"`과 같은 설정을 두고 UI 입력필드에 힌트로 표시하면, 편집자가 일관된 스타일로 코멘트를 작성하는 데 도움이 됩니다 [15]. 또한 코멘트를 비워둔 채 발행하려 할 경우 기본 문구로 채워지지 않도록(공백 허용) 하는 등의 처리도 config로 제어 가능합니다 [15].

- **UI 상의 점수/정렬:** 키워드 뉴스 UI에서는 각 기사의 **종합 점수**나 키워드 적합도 등을 함께 보여줘 편집 판단을 돕습니다. 기본적으로 점수 순으로 정렬해주고, 필요한 경우 편집자가 점수 기준 정렬/필터를 바꿀 수 있게 옵션을 둡니다. 예를 들어 “점수순/출처순/제목순” 정렬 토글 등. 점수 상위 15개를 빠르게 선택하도록 돕되, 편집자가 임의 기사도 선택 가능해야 하므로 수동 조정도 허용합니다.

- **선택 관리:** 편집자가 최소 15건 이상을 선택하도록 **UI에서 선택한 기사 개수 카운트**를 표시하고, 15건 미만일 때는 발행 버튼을 비활성화하거나 경고를 줍니다. 반대로 너무 많은 기사를 선택하면 (예: 30건 이상) 읽는 이의 부담이 될 수 있으므로, 권장 범위를 안내합니다 (예: “15~20건 권장”). 하지만 시스템적으로 상한을 두지 않고 편집자의 판단에 맡깁니다.

- **동시 운영 방식: 기존 일일 뉴스 승인 UI와 병행**하여 동작할 수 있도록 구현합니다. 즉, 관리 화면에 “일일 뉴스” 탭과 “키워드 뉴스” 탭을 나란히 제공하여, 편집자는 둘 중 현재 작업하려는 것을 선택합니다. 이 둘은 각기 별개의 JSON 데이터를 기반으로 하므로 UI 코드는 대부분 공유하되, **데이터 소스만 분기**되게 합니다. 예를 들어:

- “일일 뉴스” 탭: `selected_articles.json` 로드, 기존 방식 그대로.
- “키워드 뉴스” 탭: `selected_keyword_articles.json` 로드, 키워드별 리스트 표시.

이렇게 **분리된 화면**으로 운영하면, 하루 중 일반 뉴스와 키워드 뉴스를 모두 발행하는 것도 가능해집니다 (예: 오전에는 일일 뉴스, 오후에 키워드 특별호 발행). 두 UI가 완전히 섞여 있지 않기 때문에 혼동을 줄이고, 각각의 발행 라인을 독립적으로 관리할 수 있습니다. 만약 초기에는 키워드 뉴스가 일일 뉴스를 대체하는 용도로 쓰인다고 하더라도, 기술적으로는 병행 운용이 가능하도록 설계해 두는 것이 유연합니다.

- **UI 재사용과 개발:** 기존 승인 UI의 컴포넌트(기사 리스트 테이블, 선택/코멘트 양식, 발행 버튼 등)를 최대한 재사용하여 개발 부담을 줄입니다. 구현 관점에서, React 등의 프론트엔드라면 상태 관리만 분기하면 되고, Flask 기반 서버라면 라우트만 다르게 해서 동일 템플릿을 호출하도록 할 수 있습니다. 따라서 **UI 확장은 중간 난이도**의 작업이며, 데이터 연동 부분만 조심하면 큰 어려움은 없습니다.

- **발행 트리거 버튼:** 각 탭(일일 뉴스/키워드 뉴스) 화면 내에 별도의 "발행" 버튼을 둡니다. 키워드 뉴스 탭에서는 "키워드 뉴스 발행" 버튼으로 명시하고, 누르면 해당 `selected_keyword_articles.json`에서 `selected: true`인 항목들을 모아 **발행 스크립트**를 실행합니다. 발행 과정은 뒤에 설명될 절차에 따라 Markdown/HTML을 생성하고, 최종 독자 공개를 완료합니다. 이때 일일 뉴스 탭의 발행 버튼은 기존 daily 발행만 수행하므로, **각 발행 라인별로 별도 버튼/프로세스**가 구분되어 동작합니다.

요약하면, 관리자 UI에는 키워드 뉴스용 **입력 및 승인 인터페이스**가 추가되며, 이는 기존 UI와 나란히 제공됩니다. 편집자는 키워드 입력 → 수집 실행 → 후보 확인 및 선택/코멘트 → 발행의 흐름을 **UI상에서 일관되게 처리**할 수 있게 됩니다.

(초기 구현 단계에서는 UI에서 키워드 입력/수집 트리거를 생략하고, 편집자가 별도 스크립트를 돌린 후 결과만 UI에서 승인하도록 할 수도 있습니다. 그러나 최종적으로는 UI 통합이 편집 효율을 높이므로, 가능하면 UI에서 키워드 입력부터 발행까지 지원하는 방향으로 개발합니다.)

발행 구조 설계 (산출물 생성 및 독자용 UI)

키워드 뉴스 발행은 **하나의 키워드에 대한 독립된 뉴스 카드 묶음**을 만드는 과정입니다. 기존 퀄리뉴스 일일 발행 구조와 유사하나, 몇 가지 차이가 있습니다:

- **아카이브 Markdown 작성:** 발행 스크립트는 편집 완료된 `selected_keyword_articles.json`을 읽어, `selected: true`로 표시된 기사들만 골라냅니다[16]. 그리고 미리 준비된 **카드 템플릿(Markdown)**을 이용해 전체 Markdown 파일을 생성합니다. 이 파일은 해당 키워드 뉴스의 원본으로, 예컨대 `퀄리뉴스_키워드_2025-10-03_리플로우.md` 형태로 저장합니다. Markdown 상에는 각 기사를 카드 형태로 구성합니다:
- **헤더:** 키워드 뉴스임을 표시하는 머리글을 추가합니다. 예: `# 퀄리뉴스 키워드 뉴스 – 2025-10-03 「리플로우 솔더링」`. 이로써 독자가 보았을 때 오늘의 키워드와 날짜를 바로 알 수 있게 합니다.

- **기사 카드 목록:** 각 선택된 기사를 Markdown으로 기술합니다. 기존에 정의된 카드 템플릿을 재사용하되, 필요에 따라 키워드 표시를 추가합니다. 일반적으로 카드에는 제목 (한글 번역 제목), 출처, 요약 (국영문 번역 교차), 편집자 코멘트 등이 포함됩니다. 예컨대:

> ****[메타 AI 블로그] DINOv3: Self-supervised learning for vision at unprecedented scale****
> *Self-supervised learning(SSL) … (중략) … new SOTA results.*
> _(메타AI가 자체 지도 학습 기반 비전 모델 DINOv3를 공개했다. …)_
> – 편집자 한마디: 대규모 비전 모델의 성능을 SSL로 끌어올린 사례로, 향후 상용화 가능성이 주목된다.

위 예시는 일반 뉴스 포맷이지만, 키워드 뉴스도 이와 동일한 구성으로 각 항목을 나열합니다. **섹션 구분:** 만약 선택된 기사들이 출처 성격에 따라 다양하다면, 필요시 그룹화를 고려할 수 있습니다. 예를 들어 "공식 뉴스" vs "커뮤니티 의견"으로 나눠보는 방법이나, 아니면 **모든 기사를 한 묶음으로 표시**하는 방법이 있습니다. 키워드 자체가 이미 주제 구분 역할을 하므로, 초기 구현에서는 따로 섹션 헤더 없이 **하나의 목록**으로 15여개 카드를 쭉 나열하는 것을 제안합니다. 추후 독자 피드백에 따라 "토픽 기사"와 "관련 토론" 식으로 섹션을 넣는 개선을 할 수 있습니다.

- **HTML 페이지 생성:** Markdown 파일이 준비되면, 이를 기존과 같은 방법으로 **HTML로 변환**합니다. 일반적으로 정적 사이트 생성 방식이나 서버사이드 렌더링을 활용할 수 있습니다. 현 시스템에서는 `--publish` 모드에서 Markdown을 불러와 HTML 템플릿에 끼워넣는 것으로 보입니다[16]. 동일하게, 키워드 뉴스 Markdown을 처리하는 `--publish-keyword` 모드 또는 분기 로직을 추가합니다. 변환 과정에서 **번역 처리**를 수행합니다. 즉, 요약문의 영문 부분을 OpenAI API 등을 이용해 **자연스러운 한글로 번역**하여 교체합니다[17]. (수집 단계에서는 시간절약을 위해 번역을 미리 하지 않으므로, 발행 시점에 필요한 부분만 번역하는 구조를 유지합니다.) 필요한 경우 **전체 본문 번역(full_ko)**도 이 시점에 생성할 수 있으나, 일반적으로는 요약과 핵심만 제공하면 충분하므로 옵션으로 둡니다[18]. HTML 생성 시 편집자의 `editor_note`도 각 카드 하단에 함께 삽입하고, 글꼴/스타일 등은 기존 일일 뉴스 카드와 통일합니다. 최종 산출된 HTML 파일은 예: `2025-10-03-리플로우.html`로 저장되고, 배포 준비를 마칩니다.

- **발행 및 배포:** 완성된 HTML를 **정해진 경로에 업로드 또는 링크**시킵니다. 만약 퀄리뉴스가 정적 페이지로 매일 발행된다면, 키워드 뉴스도 유사하게 정적 페이지로 배포합니다. 이때 **기존 일일 뉴스와의 관계**를 정해야 합니다. "키워드 뉴스가 병합 발행을 대체"한다고 명시되었지만, 실제 운영에서는 **병행 가능성**도 고려됩니다. 예를 들어 키워드 뉴스 발행일에는 일일 뉴스를 쉬고 키워드 특별호만 내보낼 수도 있고, 또는 일일 뉴스는 그대로 내되 추가로 키워드 뉴스를 냈을 수도 있습니다. 시스템적으로는 둘 다 지원하도록 준비하되, 편집 운영방침에 따라 당일의 어떤 콘텐츠를 독자에게 노출할지 결정하면 됩니다.

- 만약 **대체** 운영을 한다면: 해당 일에는 키워드 뉴스 HTML만 배포하고, 메인 페이지나 이메일 등에 그것만 실어 보냅니다. 일일 뉴스 HTML은 그날 생성하지 않거나, 생성하더라도 링크를 노출하지 않습니다.

- **병행**한다면: 키워드 뉴스는 별도의 링크로 추가 제공됩니다. 예를 들어 메인 홈페이지에 "오늘의 키워드: 리플로우 솔더링" 섹션을 만들어 키워드 뉴스를 소개하고, 일일 뉴스는 평소대로 제공할 수 있습니다.

- **독자용 UI/UX 제안:** 독자가 키워드 뉴스를 소비할 수 있도록, 프런트엔드에서 약간의 **UI 개선**이 필요합니다. 몇 가지 아이디어는 다음과 같습니다:

- **홈/메인 표시:** 퀄리뉴스 홈페이지 상단에 "🔍 키워드 뉴스" 메뉴를 추가하여, 키워드 뉴스 아카이브 전용 페이지로 이동할 수 있게 합니다. 또는 메인에 **오늘의 키워드**를 배너나 카드 형식으로 노출하여, 클릭 시 해당 키워드 뉴스 페이지로 들어가도록 구성합니다.
- **아카이브 페이지:** 키워드 뉴스 발행 목록을 모아볼 수 있는 페이지를 만듭니다. 날짜순으로 키워드와 함께 목록화하여 ("2025-10-03 리플로우 솔더링", "2025-10-10 AI 엣지컴퓨팅" 등) 링크를 제공합니다. 독자는 과거에 어떤 키워드들이 다뤄졌는지 한눈에 볼 수 있고, 관심 키워드의 호를 찾아볼 수 있습니다.
- **페이지 구조:** 키워드 뉴스 HTML 페이지 자체는 일일 뉴스 페이지와 유사한 레이아웃으로 하되, **상단에 해당 키워드에 대한 간략 소개나 선정 이유** 등을 추가하는 것을 고려합니다. 예를 들어 편집장이 이 키워드를 선정한 배경이나, 키워드의 간단한 정의를 한두 줄 써주면 독자가 이해하기 쉽습니다. (이 부분은 편집자의 선택사항이지만, UI에 한줄짜리 키워드 소개 필드를 넣어둘 수 있음.)
- **모바일/알림:** 만약 모바일 앱이나 알림 시스템이 있다면, 키워드 뉴스 발행 시 별도의 푸시 알림 ("퀄리뉴스 키워드 특집: 오늘의 키워드 - 리플로우 솔더링")을 보내어 관심을 유도합니다.

- **URL 체계:** 각 키워드 뉴스 페이지에 고유 URL을 부여합니다. 예: `/keyword/20251003-리플로우` 또는 `/keyword/reflow-soldering-20251003`. URL에 키워드를 표시하면 SEO 측면에서도 검색 노출에 유리합니다. 다만 한글 키워드는 퍼센트 인코딩되므로, 영문 슬러그 병기도 고려합니다.

- **일일 뉴스와 분리 운영:** 독자 UI에서는 키워드 뉴스와 일일 뉴스를 **명확히 구분**해줍니다. 예컨대 사이트 메뉴에서 “큐레이션 뉴스” (일일 뉴스)와 “키워드 특집”을 별도 섹션으로 분리하고, 서로 다른 피드로 취급합니다. 이렇게 하면 독자가 둘을 다른 콘텐츠로 인식하기 쉬워집니다. 반면, 만약 키워드 뉴스가 특정 기간 기존 뉴스를 완전히 대체한다면 (예: 실험적으로 일주일간 키워드 뉴스만 발행), 독자에게 공지하여 혼란을 줄여야 할 것입니다. 기술적 구현은 유연하게 해두고, **운영 정책에 따라 노출 제어**가 가능하도록 하는 것이 바람직합니다.

요약하면, 발행 단계에서는 **키워드별 MD/HTML/JSON 산출물**을 생성하여 아카이브하고, 독자에게는 별도의 키워드 뉴스 페이지/메뉴를 통해 제공하도록 구성합니다. 기존 일일 뉴스 발행과는 분리되어 있되, UI 상에서는 서로 **연계되어 탐색**할 수 있게 해주는 것이 좋습니다 (예: “다른 날의 키워드 보기” 혹은 “일일 뉴스 보러가기” 링크 교차 제공). 이렇게 함으로써 키워드 뉴스가 퀄리뉴스의 새로운 카테고리로 인식되고, 독자들은 관심 분야의 특별 큐레이션을 손쉽게 접할 수 있을 것입니다.

운영 방식 (키워드 입력 및 발행 트리거)

키워드 뉴스 기능은 **전적으로 수동 요청으로만 실행**됩니다. 이를 위해 운영 프로세스 및 도구를 정비합니다:

- **키워드 입력 방식:** 편집자가 하루 한 번 입력할 키워드를 시스템에 전달하는 방법은 두 가지 옵션이 있습니다. (1) **관리자 UI에 입력 폼 추가:** “키워드 뉴스 수집” 섹션에 텍스트 입력칸을 만들고, 편집장이 키워드를 기입한 뒤 “수집 시작” 버튼을 누르면 해당 키워드로 수집 프로세스가 트리거됩니다. (2) **설정 파일/명령행 사용:** 초기 간단히 구현할 때는 `config.json`에 `keyword` 필드를 넣고 수집 스크립트를 수동 실행하거나, CLI 인자로 `--keyword "xxx"`를 주어 실행하는 방법을 사용할 수 있습니다. 예를 들어 개발/테스트 단계에서는 다음처럼 사용 가능합니다:

  ```
  $ python jjippa_news_scheduler.py --collect-keyword "리플로우 솔더링"
  ```

  이 명령이 실행되면 내부에서 `config["keyword"] = "리플로우 솔더링"`과 같은 설정을 읽어 위에서 설명한 키워드 필터 흐름을 따라 동작하게 합니다. **UI 연동이 완료되기 전까지는 임시로 편집자가 SSH 등으로 명령을 실행**할 수 있지만, 최종적으로는 UI 버튼으로 대체합니다. UI에서 키워드 입력을 받으면 백엔드에서 해당 프로세스를 비동기로 실행하고, 일정 시간 후 결과를 UI에 반영해주는 형태가 될 것입니다 (예: “수집 중...완료” 상태 표시 후 리스트 로딩).

- **발행 트리거 및 자동화 제한:** 키워드 뉴스 수집/발행은 **자동 스케줄에 올라있지 않도록** 해야 합니다. 즉, cron이나 스케줄러에 등록하지 않고, **편집자 액션이 있을 때만 실행**되게 합니다. 이를 보장하기 위해:

- 키워드 뉴스 수집 함수는 기본 비활성화 상태이며, 인자가 전달되거나 UI 액션이 없는 한 작동하지 않습니다.
- 일일 수집과 완전히 별개의 함수로 구현하거나, 하나의 스크립트 내에 `if args.keyword: ...` 분기로 구현하되 기본 스케줄 모드에서는 키워드 관련 코드를 건너뛰도록 합니다.
- 만약 실수로 키워드 입력 없이 키워드 모드가 실행되는 것을 막기 위해, **빈 키워드에 대한 예외 처리**를 합니다. (예: `if keyword.strip() == "": abort`).

- 또한 하루에 하나만 처리하도록 설계했으므로, 한 번 실행이 끝나면 `selected_keyword_articles.json`에 기록된 date를 확인하거나 전역 플래그를 사용해 **이미 처리됨**을 표시해둘 수 있습니다. 하지만 수동 실행인 이상, 하루에 여러 번 돌릴 가능성은 낮으므로 엄격한 제약은 필요 없고, 편집자 재요청 시 덮어쓸 수 있도록 유연하게 둡니다.

- **수집 및 발행 절차:** 운영 프로세스는 다음과 같이 정리됩니다:

- **키워드 선정:** 편집자 내부 회의나 판단으로 그날의 키워드를 결정.
- **키워드 입력:** 관리 도구를 통해 키워드 문자열 입력 (예: "리플로우 솔더링").

- **수집 실행:** 키워드 입력과 함께 수집 프로세스 트리거 → 자동 크롤링 및 후보 JSON 생성 (약 수분 소요).
- **편집 검토:** 관리자 UI에서 후보 기사/게시글 확인 → 적합한 기사 15~20개 **선택** 및 **코멘트 입력**.
- **발행 실행:** UI에서 "발행" 실행 → Markdown/HTML 생성 및 자동 번역 처리 → 결과 페이지 생성.
- **배포 확인:** 생성된 페이지를 독자에게 노출 (웹 게시 또는 이메일 전송 등)하고 링크 확인.

모든 과정은 수동 개입 (키워드 입력, 기사 선택, 발행 버튼 클릭)이 필요하며, **자동으로 주기 실행되지 않습니다.** 이로써 편집자는 **원하는 날에만** 키워드 뉴스를 발행할 수 있고, 원치 않는 날에는 아무 작업도 일어나지 않습니다.

- **자동화 제한 근거:** 이렇게 수동으로 운영함으로써, **키워드 선정의 신중함**을 유지하고 콘텐츠 품질을 관리할 수 있습니다. 키워드 뉴스는 성격상 특정 주제에 대한 깊이있는 큐레이션이므로, 매일 기계적으로 돌리기보다는 **필요할 때**(예: 큰 이슈가 있을 때, 주1~2회 등) 발행하는 것이 바람직합니다. 추후 만약 운영상 매일 정례화한다면, 그때는 일일 자동화로 전환할 수도 있으나, 초기엔 편집자 판단에 맡기는 것이 좋습니다.

- **예외 상황:** 만약 키워드 뉴스 수집을 돌렸는데 15건 미만으로 나왔다면, 편집자는 키워드를 변경하거나 유사 키워드를 추가하여 다시 수집할 수 있습니다. 시스템적으로 이를 돕기 위해, **최근 결과를 초기화하고 재수집**할 수 있는 기능을 UI에 넣는 것을 고려합니다 (예: "결과 부족 – 키워드 수정 후 재수집" 안내). 그러나 일단 첫 번째 키워드 입력에만 집중하도록 구현하고, 편집자 판단으로 스크립트를 다시 실행하는 것으로 커버해도 무방합니다.

정리하면, 운영상 **편집장의 키워드 입력 → 수동 트리거**로만 동작하며, 자동 스케줄은 투입하지 않습니다. 이로써 잘못된 키워드로 인한 오동작이나, 키워드 미입력시의 불필요한 실행을 방지합니다. 또한 이러한 수동 운영 방식은, 향후 **키워드 뉴스 발행 빈도나 시기를 유연하게 조절**할 수 있게 해줍니다 (예: 독자 반응에 따라 늘리거나 줄이는 등).

구현 난이도 및 우선순위 (개발 단계별 전략)

마지막으로, 앞서 제시한 기능들을 실제로 개발할 때의 난이도 평가와 순차적 추진계획을 제안합니다. **주요 작업 항목과 우선순위**는 다음과 같습니다:

1. **키워드 필터 수집 로직 구현 (1순위, 난이도: 보통):** 가장 먼저, 백엔드 수집기에 키워드 필터링 기능을 추가합니다. 이는 Python 크롤러(`jjippa_news_scheduler.py` 등)에 비교적 간단한 분기와 문자열 매칭 로직을 넣는 것으로 실현 가능합니다. 공식 소스 리스트를 순회하며 제목/요약에 `keyword in text` 체크를 추가하고, 커뮤니티 소스는 Reddit API 검색 쿼리를 구성하는 식입니다. 기본 구현은 용이하나, **각 소스별 커스텀 처리**(예: 사이트별 검색 URL 존재 여부 등)를 해야 한다면 약간의 조사와 코드 추가가 필요할 수 있습니다. 우선은 **RSS/리스트 기반 필터링**으로 충분히 효과를 볼 수 있으므로, 이 부분을 빠르게 완성합니다. (신규 소스나 키워드 리스트 확장은 설정 변경만으로 가능하므로 손쉽게 커버할 수 있습니다 [19].)

2. **스코어링 체계 및 editor_rules 설정 (1순위, 난이도: 낮음):** 다음으로 기사 가치 평가를 위한 **스코어 계산 로직**을 구현합니다. 대부분의 요소 (도메인 가중치, 키워드 매칭, 팩트체크 점수 등)는 이미 데이터로 존재하거나 쉽게 뽑아낼 수 있으므로, 이를 합산하는 함수를 작성하면 됩니다. `editor_rules.json`에 **도메인별 점수표와 키워드별 점수표**를 정의하고 [11], 코드에서 이를 불러와 적용하도록 합니다. 예를 들어:

```
domain_score = editor_rules["domain_weights"].get(domain, 0)
kw_score = 0
for kw, w in editor_rules["keyword_weights"].items():
    if kw.lower() in article_text.lower():
        kw_score += w
total_score = domain_score + kw_score + fc_score + ...
```

이런 형태로 구현하고, 결과를 `selected_keyword_articles.json` 생성 시 함께 저장합니다. 이 작업은 복잡한 알고리즘이 아닌 상수 가중치 적용이므로 난이도가 낮습니다. 단, 적절한 가중치 값은 초기에 **편집자와 협의하여 설정**해야 합니다. (ex: IPC 등의 핵심어 +3, 주요 도메인 +5 등 [14]) 이 부분은 1순위에 포함시키는 이유가, 점수체계가 갖춰져야 이후 UI에서 편집 선정을 효율화할 수 있기 때문입니다.

3. **JSON 결과 구조 및 파일 출력 (1순위, 난이도: 낮음):** 수집이 끝난 후 결과를 JSON으로 저장하는 기능을 구현합니다. Python에서 딕셔너리를 JSON으로 dump하면 되므로 간단합니다. 기존 `selected_articles.json` 생성로직을 참고하여, 키워드 모드일 때는 약간 다른 필드구조로 출력하게 합니다. 예를 들어 `if mode == "keyword": output_data = {"keyword": kw, "date": today, "articles": [...]}` 식으로 구성합니다. 이 단계에서는 `selected_keyword_articles.json`을 덮어쓸지, 날짜별로 새 파일 생성할지 결정해야 합니다. 초기에는 **하나의 파일 재사용 방식**이 구현이 쉬우므로 그렇게 하고, 이후 운영 중 필요 느껴지면 아카이브 기능 추가합니다. JSON 출력은 1순위 작업들과 함께 진행해 바로바로 결과를 살펴볼 수 있게 합니다.

4. **Markdown/HTML 템플릿 제작 (2순위, 난이도: 낮음):** 키워드 뉴스용 Markdown 템플릿은 기존 일일 뉴스 템플릿을 복사하여 약간 수정하면 됩니다. 큰 수정 없이, 제목 부분에 키워드를 표시하도록 하고, 만약 섹션을 나누지 않는다면 루프 돌며 카드 나열만 하면 됩니다. Jinja2 템플릿 등을 사용 중이면 새 템플릿 파일을 만들거나, 기존 템플릿에 조건 분기를 넣어 재활용할 수도 있습니다. 난이도가 낮고 비교적 빠르게 완성 가능하므로 2순위로 설정합니다. HTML 변환 및 번역 삽입도 기존 코드 활용으로 쉽게 구현됩니다. (OpenAI API 호출 부분 등은 이미 있으므로, 키워드 뉴스 발행 함수에서 동일 호출하면 됩니다 [17] .) 이때 **발행 스크립트 분리 여부**를 결정해야 합니다. 옵션은 a) 하나의 `--publish`에서 내부적으로 키워드/일반을 구분, b) 별도 `--publish-keyword` 인자를 만드는 것 입니다. 구현과 사용의 명확성을 위해 **별도 인자/명령**을 만드는 편이 좋습니다. 그래서 수집은 `--collect-keyword`, 발행은 `--publish-keyword`로 CLI를 제공하면, UI에서도 각각 별도 호출하기 쉽습니다.

5. **관리자 UI 기능 추가 (2순위, 난이도: 중간):** 백엔드와 템플릿 쪽 윤곽이 잡히면, 관리자 대시보드에 키워드 뉴스 기능을 붙입니다. 프론트엔드(React/Vue 등)와 백엔드(Flask/FastAPI 등) 양쪽 수정이 필요합니다. 구체 작업으로는:

6. 새 메뉴/라우트 생성: `/admin/keyword` 페이지를 만들고, 진입 시 서버에서 `selected_keyword_articles.json` 데이터를 로드하여 전달.
7. 프론트엔드 컴포넌트: 기존 기사 리스트 테이블 컴포넌트를 재사용하거나, 복사해서 키워드 전용으로 사용. 차이는 데이터 필드(예: keyword 표시 등) 정도이므로, **동일 컴포넌트를 매개변수만 달리하여** 쓸 수 있을 것입니다.
8. 키워드 입력/수집 트리거: UI에서 키워드 입력 폼과 "수집" 버튼을 구현하고, 이를 누르면 AJAX로 백엔드 `/admin/keyword_collect?kw=...` 등의 API를 호출하여 수집기 실행을 요청. 백엔드는 subprocess로 `--collect-keyword` 스크립트를 실행하거나, 쓰레드/비동기로 해당 함수를 호출합니다. 수집이 완료되면 JSON이 생성되므로, UI에서 일정 시간 딜레이 후 목록을 갱신하거나 수집 완료 신호를 받을 수 있게 합니다 (예: 폴링 혹은 WebSocket 등 구현).
9. 선택/코멘트 입력: 이 부분은 기존 기능과 동일하게 구현. JSON의 `selected`와 `editor_note` 필드를 UI에서 편집 가능케 하고, 변경 시 바로 Ajax로 저장하거나 temporary state에 두었다가 발행 시 한꺼번에 PATCH. (현재 시스템에서 편집 완료 후 수동으로 발행 스크립트를 돌리는 구조라면, UI에서 JSON 저장만 하고 발행은 서버 CLI에서 separate하게 할 수도 있습니다.)
10. 발행 버튼: 프론트에서 "발행" 클릭 시 `/admin/publish_keyword` API 호출 → 백엔드에서 `--publish-keyword` 실행 → Markdown/HTML 생성. 성공 시 UI에 성공 알림.
11. 오류/예외 처리: 수집 중 두 번 누르지 않도록 버튼 비활성화, 수집 실패 시 메시지 표시, 발행 완료되면 URL 표시 등 사용자 안내 요소 구현.

UI 작업은 기능단위로 보면 어렵지 않으나, **수집 트리거의 비동기 처리** 등 약간의 개발 난도가 있습니다. 한 번에 다 구현하기보다, 초기에는 **키워드 입력->수집은 수동(CLI)**, UI에서는 결과 승인/발행만 하도록 단계를 밟아도 됩니다. 최종

적으로는 UI에서도 수집까지 되도록 할 것이므로, 전체 구현 난이도는 중간 정도로 평가합니다. 이 작업을 2순위로 둔 이유는, 백엔드 작업이 어느 정도 완료된 뒤 UI를 붙이는 게 자연스럽기 때문입니다. (UI 개발은 병렬로도 가능하지만, 데이터 포맷이 완성되어야 수월합니다.)

1. **독자용 프론트엔드 개편 (3순위, 난이도: 중간):** 키워드 뉴스가 실제 발행되면, 독자들이 이를 찾아보고 읽을 수 있는 UI를 만들어야 합니다. 이는 최종 마무리 단계로 둘 수 있습니다. 작업 내역:
2. 메인 페이지 수정: 키워드 뉴스 항목/배너 추가. 디자인팀과 협의하여 눈에 띄게 배치.
3. 키워드 뉴스 목록 페이지: 새로운 페이지 개발. 기존 뉴스 아카이브 페이지를 참고해 유사하게 구현. SEO 위해 제목에 키워드 포함, 메타태그 설정 등 챙기기.
4. 페이지 라우팅: `/keyword/:slug` 형태로 라우팅 추가. static 파일일 경우 Nginx 설정, 아니면 SSR일 경우 서버 라우트 추가.
5. 스타일 조정: 키워드 뉴스 페이지 내 상단에 키워드 제목과 날짜를 큰 글씨로 표시하고, 이하 카드 리스트는 기존과 동일 CSS 적용. 필요하면 키워드 강조 색상 등을 넣어줄 수도 있음.
6. 테스트: 다양한 기기에서 UI 확인, 기존 일일 뉴스와 충돌 없는지 체크.

이 프론트엔드 개편은 **필수이지만 시급하지는 않아서 3순위**로 둡니다. 키워드 뉴스 자체 기능 구현이 먼저이고, 내부 테스트를 거쳐 발행 준비가 되면 독자 UI를 배포하면 됩니다. 난이도는 현재 웹사이트 구조에 따라 다르나, 메뉴 하나 추가하고 목록/상세 페이지 더하는 정도라면 중간 난이도로 볼 수 있습니다.

1. **고급 기능 및 튜닝 (장기 과제):** 위의 핵심 개발이 완료되고 운영을 시작하면, 실사용을 통해 부족한 점이나 추가 요구가 나올 수 있습니다. 예컨대:
2. **외부 검색 통합 개선:** 초기에는 키워드 관련 기사를 기존 소스 위주로 모으지만, 필요시 Google/Bing API를 본격 연동하여 **커버리지 확대**를 도모할 수 있습니다. 이 작업은 API 키 발급, 쿼리 튜닝, 응답 파싱 등의 노력이 필요하고, 잘못하면 잡음이 많이 끼므로 신중히 접근해야 합니다 [20]. 구현 난이도는 다소 높고 (API 이용, 비용 고려), 운영 위험은 잡음 제어 측면에서 존재합니다. 따라서 실 운영에서 특정 키워드에 기사 수가 부족하거나, 더 많은 글로벌 뉴스를 원할 때 **2차 개선사항**으로 추진합니다.
3. **소셜 모니터링/수동 입력:** LinkedIn, Twitter 등의 키워드 관련 언급을 수집하거나 Slack 알림으로 편집자에게 참고 자료를 주는 등의 기능도 생각해볼 수 있습니다 [21]. 이는 자동화보다 편집자의 리서치 도구로 활용될 가능성이 높고, 구현은 복잡하므로 추후 검토합니다.
4. **AI 기반 추천/요약:** 키워드 뉴스 적합 기사를 AI 모델이 추천하거나, 편집자의 코멘트 작성에 AI가 보조하는 등도 장기적으로 가능성 있습니다. 예를 들어 점수 상위 기사들을 자동으로 선택해주고 편집자가 수정만 한다든지, 또는 코멘트 초안을 GPT로 생성해준다든지 하는 방식입니다. 이러한 AI Assistant 도입은 **중장기 로드맵**으로 고려하고, 먼저 키워드 뉴스 기능이 안정적으로 자리잡으면 추진합니다 [22] [13].

우선순위 요약:

- **1단계 (핵심)**: 키워드 기반 수집 백엔드 (+스코어링) 구현 → 결과 JSON 출력 → 기본 발행(MD/HTML) 기능 → (내부 점검 및 튜닝) [19] [11].
- **2단계 (운영 도구)**: 관리자 UI에서 키워드 뉴스 승인/코멘트 기능 추가 → UI에서 발행 트리거 지원 [15].
- **3단계 (출시 준비)**: 독자용 프론트 개편 → 키워드 뉴스 정식 발행 개시 (베타 운영).
- **4단계 (고도화)**: 추가 소스/검색 통합, AI 보조 등 품질 및 커버리지 향상 작업.

이러한 순서로 진행하면, **위험도가 낮은 부분부터 차근차근 완성**하여 빠르게 기본 기능을 선보이고, 이후 개선을 통해 완성도를 높일 수 있습니다 [23] [24]. 특히 1단계의 백엔드 구현은 비교적 직선적인 작업이고, 2단계 UI는 조금 복잡하지만 사용자 가치에 직접 영향하므로 신속히 착수합니다. 3단계는 내부적으로 키워드 뉴스 콘텐츠 품질이 확인되고 나면 바로 이어서 진행하면 됩니다. 최종적으로 키워드 뉴스 기능이 성공적으로 정착하면, 퀄리뉴스는 **"하루 한 키워드로 업계 핵심 정보를 손쉽게 파악"**할 수 있는 새로운 서비스를 독자들에게 제공하게 될 것입니다. 이는 편집 효율 향상과 콘텐츠 전문성 강화를 모두 달성하는 방향으로, 기존 일일 뉴스와 상호보완적으로 플랫폼 가치를 높여줄 것으로 기대됩니다. [8] [25]

---

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 0930퀄리뉴스 SMT_AI 기사 수집 기술 고도화 가이드.pdf
file://file-SsvqZEz4mindUznsFZjfUv