API를 사용한
뉴스 수집 및 배포 자동화

팀명 : 취업시켜조

팀원 : 권 *
김*우
김*은
김*서

python

대한상공회의소
서울기술교육센터

0. 목차

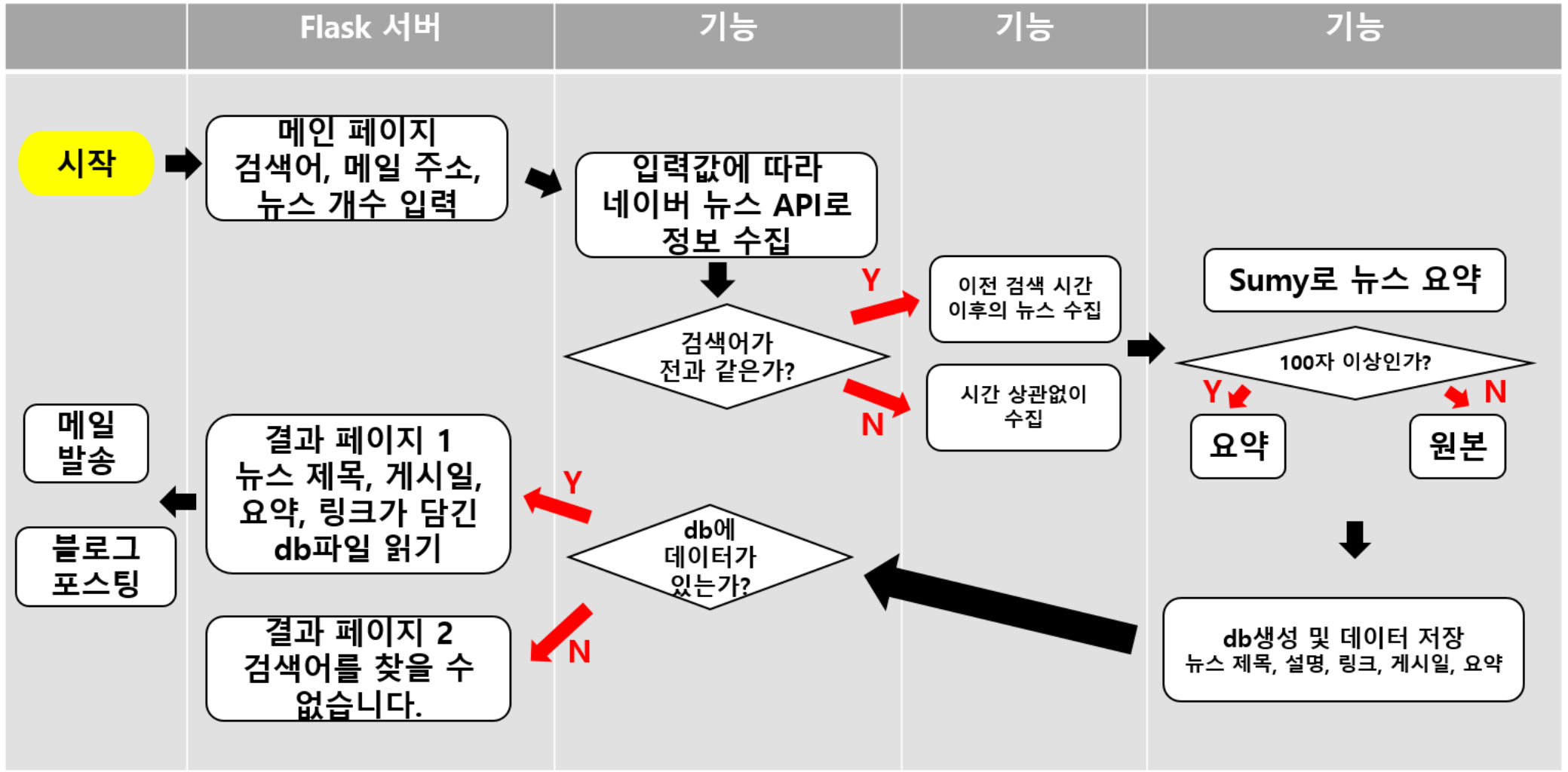
1. 기획 및 흐름도

1-1. 기획 의도

검색결과 4,926건

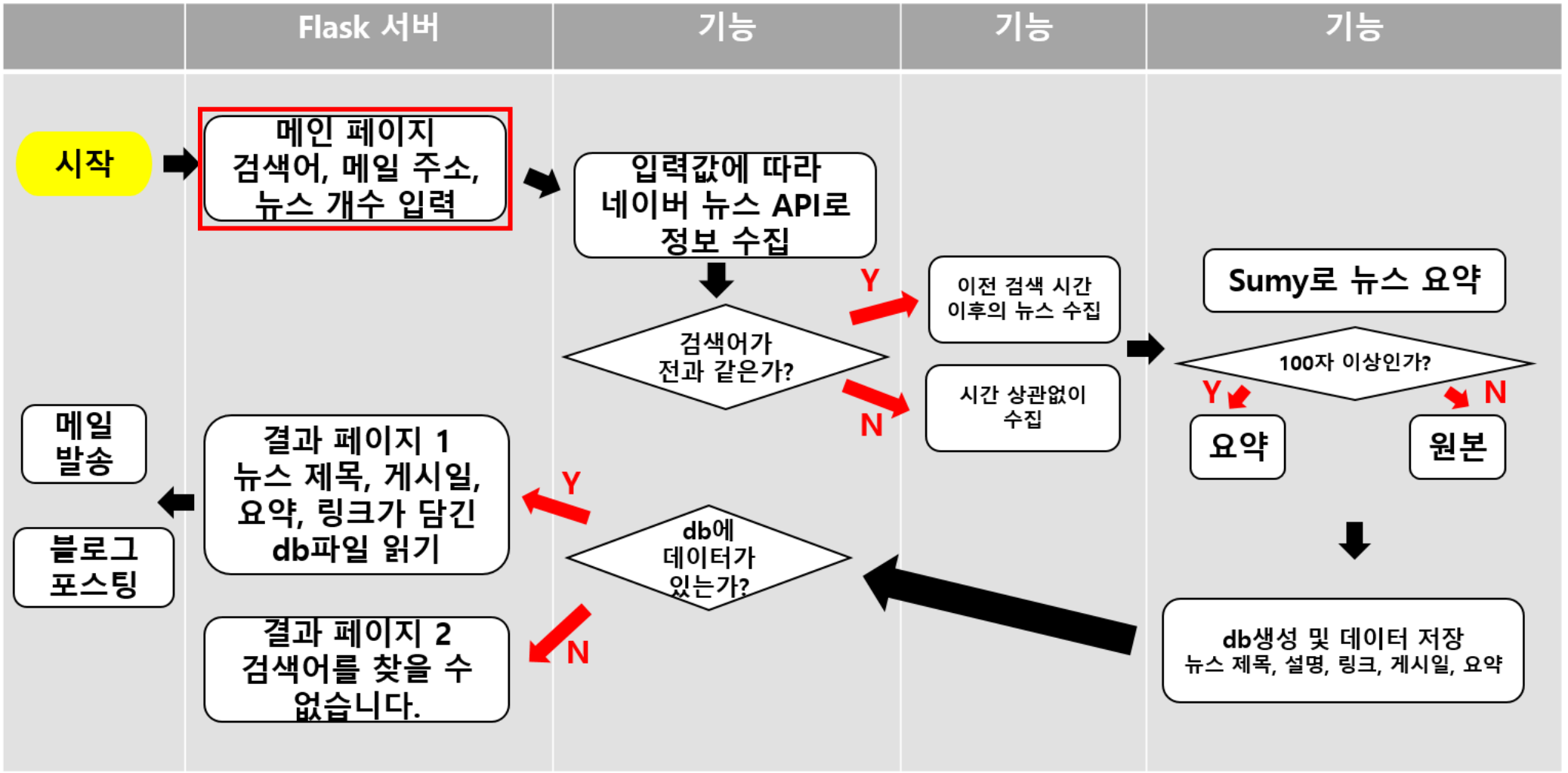
합격연도	기업명	직무명	분류	합격스펙/내용
2024 상반기	삼성에스원	경영지원	대기업, 신입	**인서울4년제 / 영어 / 학점 4.23 / 토익: 930, 토익스피킹: 레벨6 / 공기업 인턴 …** 최근 **사회이슈** 중 중요하다고 생각되는 한가지를 선택하고 이에 관한 자신의 견해를 기술해 주시기 바랍니다.[환경문제 속 비즈니스 기회 찾기] 사회·문화적 기대가 끊임없이 변화하면서 기업은 그에 맞는 사회적 책임(CSR)을 지고 있습니다.
2024 상반기	삼성에스원	경영지원	신입, 대기업	**인서울4년제 / 영어 / 학점 4.23 / 토익: 930, 토익스피킹: 레벨6 / 사회생활 경…** 최근 **사회이슈** 중 중요하다고 생각되는 한가지를 선택하고 이에 관한 자신의 견해를 기술해 주시기 바랍니다.[환경문제 속 비즈니스 기회 찾기] 사회·문화적 기대가 끊임없이 변화하면서 기업은 그에 맞는 사회적 책임(CSR)을 지고 있습니다.
2023 하반기	삼성전자DX	생활가전 사…	대기업, 신입	**서성한 / 사회과학전공 / 학점 4.1/4.5 / 토익: 955, 오픽: AL / 사회생활 경험: …** 최근 **사회이슈** 중 중요하다고 생각되는 한 가지를 선택하고, 이에 관한 자신의 견해를 기술해 주시기 바랍니다. (1000자) 최근 **사회이슈** 중 중요하게 생각하는 분야 중 하나는 AI(인공지능)의 발달입니다.

1-2. 시스템 흐름도


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
검색어가
전과 같은가?
Y
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

2. 주요 기능

2-1. Flask 서버


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
Y
검색어가
전과 같은가?
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

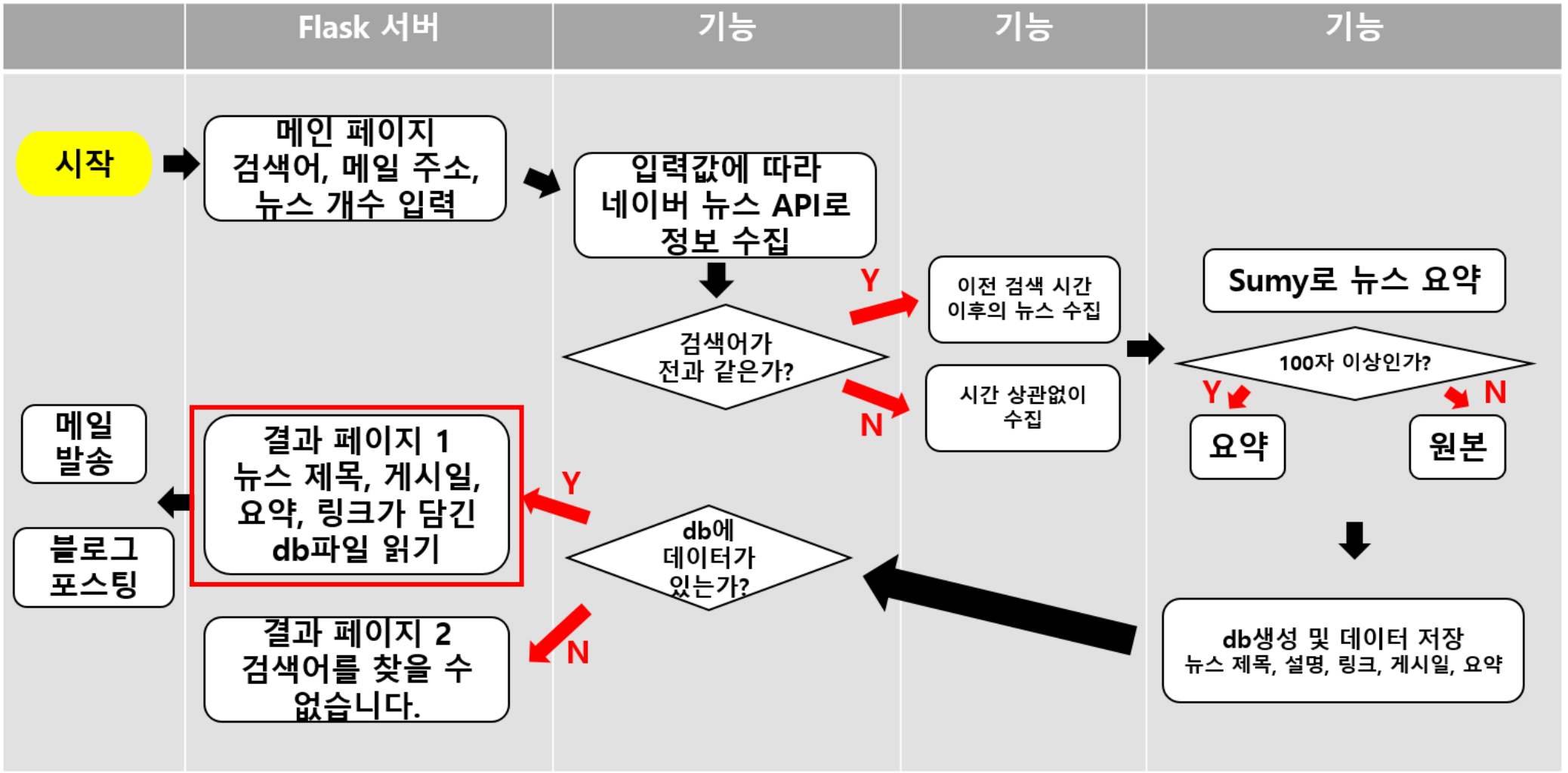
2-1. Flask 서버

```python
app = Flask(__name__)


@app.route("/")
def index():
    return render_template('index.html')


@app.route("/search", methods=["GET"])
def print_news():

    코드생략


    if news_token == 1:
        return render_template("no_results.html", search=searc

    return render_template("results.html", search=search, filtered_l

if __name__ == "__main__":
    app.run()
```

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        *{
            font-weight:bold;
            color:black;
            line-height: 20px;
        }
    </style>
</head>
<body align="center">
    <table align="center">
        <form action="/search" method="get">
            <img alt="대한상공회의소" src="https://i.ibb.co/RptmNbTH/image10000.jpg" height="30%" width="30%">
            <tr>
                <td style="font-size:20px" align="right">검색어를 입력하세요</td>
                <td><input type="text" name="search" required></td>
            </tr>
            <tr>
                <td style="font-size:20px" align="right">이메일을 입력하세요</td>
                <td><input type="email" name="mail" required></td>
            </tr>
            <tr>
                <td style="font-size:20px" align="right">뉴스의 개수를 입력하세요(1~20개)</td>
                <td><input type="number" name="display_in" min="1" max="20" required style="width:175px"></td>
            </tr>
            <td align="right">
            <td>
                <select name="sort" style="height:25px">
                    <option value="sim">정확도순</option>
                    <option value="date">날짜순</option>
                </select>
            <input type="submit" value="검색" style="height: 30px;width: 60px;"><br/>
            <br>
            <a style="color:green" href="https://news.naver.com/">네이버 뉴스 바로가기</a>
        </form>
    </table>
</body>
</html>
```

2-1. Flask 서버


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
검색어가
전과 같은가?
Y
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

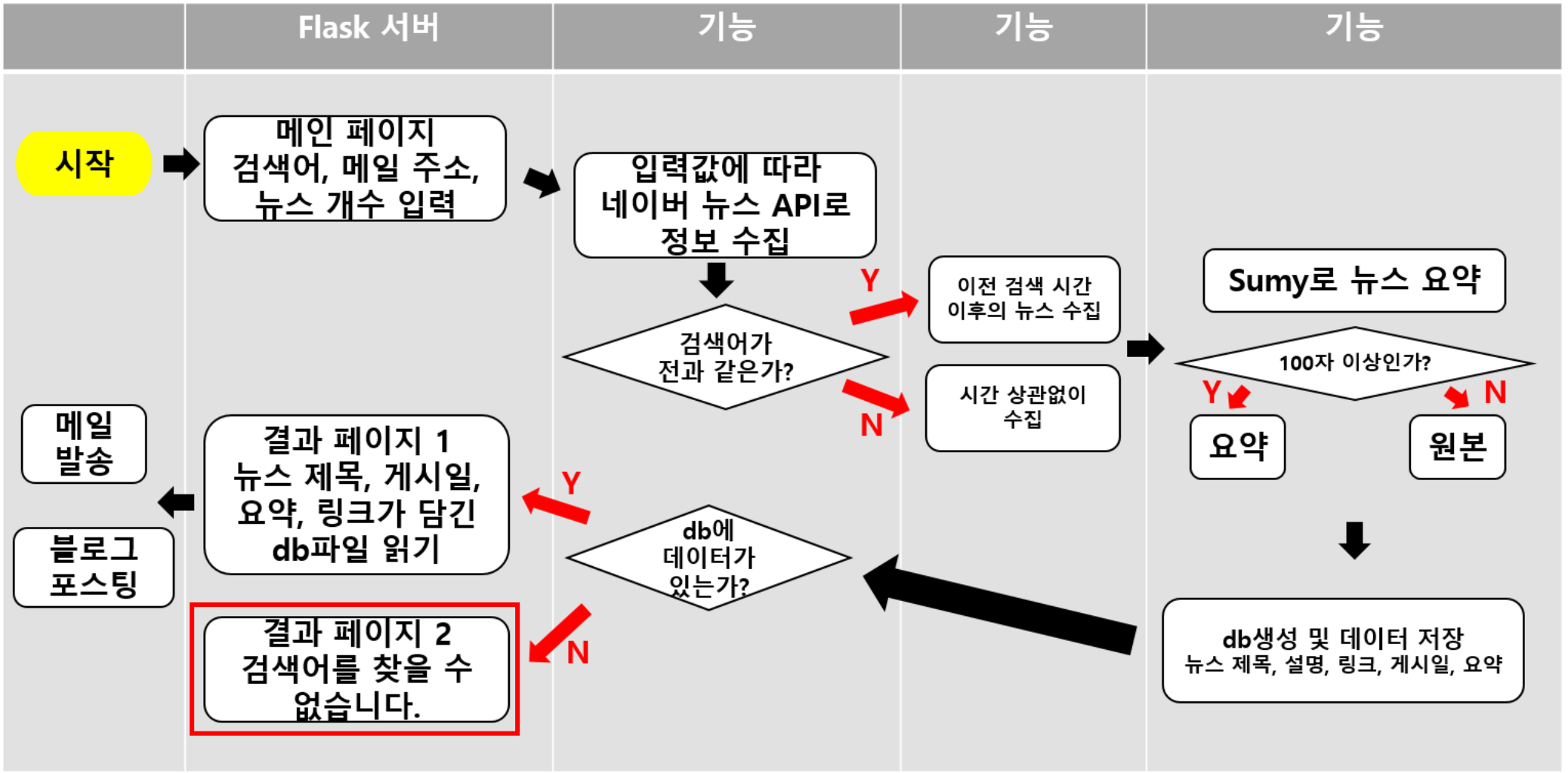
2-1. Flask 서버

```python
app = Flask(__name__)



@app.route("/")
def index():
    return render_template("index.html")



@app.route("/search", methods=["GET"])
def print_news():

    코드생략


    if news_token == 1:
        return render_template("no_results.html", search=searc

    return render_template('results.html', search=search, filtered_l

if __name__ == "__main__":
    app.run()
```

results.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>뉴스 검색 결과</title>
</head>
<body align="center">
    <h2>이메일 전송, 블로그 포스팅 완료</h2>
    <hr>
    <h2>"{{ search }}" 뉴스 검색 결과</h2>
    <table border="1" align="center">
        <tr>
            <th style="width:400px">제목</th>
            <th style="width:200px">게시일</th>
            <th style="width:400px">요약</th>
            <th style="width:100px">링크</th>
        </tr>
        {% for news in filtered_list%}
        <tr>
            <td>{{ news.title }}</td>
            <td>{{ news.pubDate }}</td>
            <td>{{ news.summary }}</td>
            <td><a href="{{ news.link }}" target="_blank">기사 보기</a></td>
        </tr>
        {% endfor %}
    </table>
    <br/>
    <a href="/">다시 검색</a>
</body>
</html>
```

2-1. Flask 서버


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
검색어가
전과 같은가?
Y
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

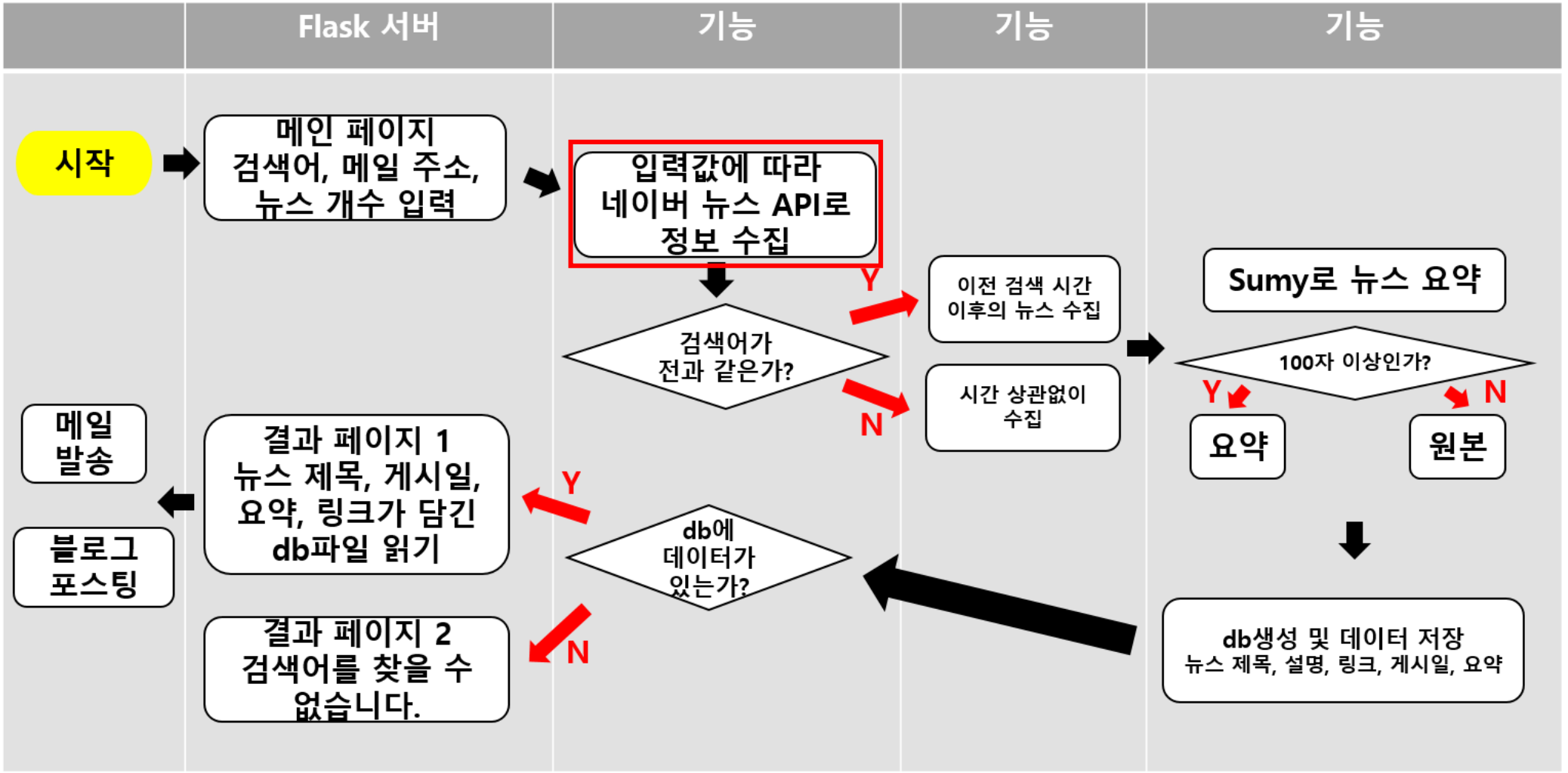
2-1. Flask 서버

```python
app = Flask(__name__)



@app.route("/")
def index():
    return render_template("index.html")



@app.route("/search", methods=["GET"])
def print_news():

    코드생략


    if news_token == 1:
        return render_template("no_results.html", search=search

    return render_template("results.html", search=search, filtered_l

if __name__ == "__main__":
    app.run()
```

no_results.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body  align="center">
    <table>
        <h2>"{{ search }}"로 검색한 새로나온 기사가 없습니다</h2><br>
        <a href="/">다시 검색</a>
    </table>
</body>
</html>
```

2-2. 검색 API 변수 저장 및 요청



2-2. 검색 API 변수 저장 및 요청

```python
# 네이버 검색 API 예제 - 블로그 검색
import os
import sys
import urllib.request
client_id = "YOUR_CLIENT_ID"
client_secret = "YOUR_CLIENT_SECRET"
encText = urllib.parse.quote("검색할 단어")
url = "https://openapi.naver.com/v1/search/blog?query=" + encText # JSON 결과
# url = "https://openapi.naver.com/v1/search/blog.xml?query=" + encText # XML 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
else:
    print("Error Code:" + rescode)
```

2-2. 검색 API 변수 저장 및 요청

```python
@app.route("/search", methods=["GET"])
def print_news():

    news_token = 0

    search =  request.args["search"]
    mail = request.args["mail"]
    display = request.args["display_in"]
    sort = request.args["sort"]

    # 네이버 developer에서 발급 "https://developers.na
    client_id = [redacted]
    client_secret = [redacted]

    # url = "https://openapi.naver.com/v1/search/{검색

    url = "https://openapi.naver.com/v1/search/news.j

    # API 인증을 위한 Id와 Secret을 headers에 입력
    headers = {
        "X-Naver-Client-Id" : client_id,
        "X-Naver-Client-Secret" : client_secret
    }

    # 검색어, 표시할 뉴스의 수, 정렬기준을 위한 파라미터
    params = {
        "query" : search,
        "display" : str(display),
        "sort" : sort
    }

    # 설정한 headers와 params를 get을 통해 url에 있는 사이트에 요청

    response = requests.get(url, headers=headers, params=params)

    if response.status_code == 200:
        response_dict = response.json()

        for item in response_dict["items"]:
    # Title과 Description 처리
            item["title"] = html.unescape(re.sub(r"<.*?>", "", item["title"]))
            item["description"] = html.unescape(re.sub(r"<.*?>", "", item["description"]))

            # Date 처리
            item["pubDate"] = datetime.strptime(item["pubDate"], '%a, %d %b %Y %H:%M:%S %z').strftime('%Y-%m-%d %H시 %M분')
```

Step 1. 사용자 입력변수 생성

```python
search =  request.args["search"]
mail = request.args["mail"]
display = request.args["display_in"]
sort = request.args["sort"]

# 네이버 developer에서 발급 "https://developers.naver.com/apps/#/myapps/5_EbG5C0vaQvkusHOlP5/overview"
client_id = "[redacted]"
client_secret = "[redacted]"

# url = "https://openapi.naver.com/v1/search/{검색}.json" 검색에 blog, news, 등에 따라 검색할 종목 변경

url = "https://openapi.naver.com/v1/search/news.json"
```

2-2. 검색 API 변수 저장 및 요청

```python
@app.route("/search", methods=["GET"])
def print_news():

    news_token = 0

    search =  request.args["search"]
    mail = request.args["mail"]
    display = request.args["display_in"]
    sort = request.args["sort"]

    # 네이버 developer에서 발급 "https://developers.naver.com/
    client_id = "NwwP4DrhiMMqv6I2uU8y"
    client_secret = "jclkobNj7a"

    # url = "https://openapi.naver.com/v1/search/{검색}.json"

    url = "https://openapi.naver.com/v1/search/news.json"

    # API 인증을 위한 Id와 Secret을 headers에 입력
    headers = {
        "X-Naver-Client-Id" : client_id,
        "X-Naver-Client-Secret" : client_secret
    }

    # 검색어, 표시할 뉴스의 수, 정렬기준을 위한 파라미터 설정(dis
    params = {
        "query" : search,
        "display" : str(display),
        "sort" : sort
    }

    # 설정한 headers와 params를 get을 통해 url에 있는 사이트에

    response = requests.get(url, headers=headers, params=par

    if response.status_code == 200:
        response_dict = response.json()

        for item in response_dict["items"]:
    # Title과 Description 처리
            item["title"] = html.unescape(re.sub(r"<.*?>", "", item["title"]))
            item["description"] = html.unescape(re.sub(r"<.*?>", "", item["description"]))

            # Date 처리
            item["pubDate"] = datetime.strptime(item["pubDate"], '%a, %d %b %Y %H:%M:%S %z').strftime('%Y-%m-%d %H시 %M분')
```

Step 2. url, header, parameter 생성 및 요청

```python
# API 인증을 위한 Id와 Secret을 headers에 입력
headers = {
    "X-Naver-Client-Id" : client_id,
    "X-Naver-Client-Secret" : client_secret
}

# 검색어, 표시할 뉴스의 수, 정렬기준을 위한 파라미터 설정(display의 경우 int형 숫자로 입력받기 때문에 string 형태로 변환)
params = {
    "query" : search,
    "display" : str(display),
    "sort" : sort
}

# 설정한 headers와 params를 get을 통해 url에 있는 사이트에 요청

response = requests.get(url, headers=headers, params=params)

if response.status_code == 200:
    response_dict = response.json()
```

2-2. 검색 API 변수 저장 및 요청

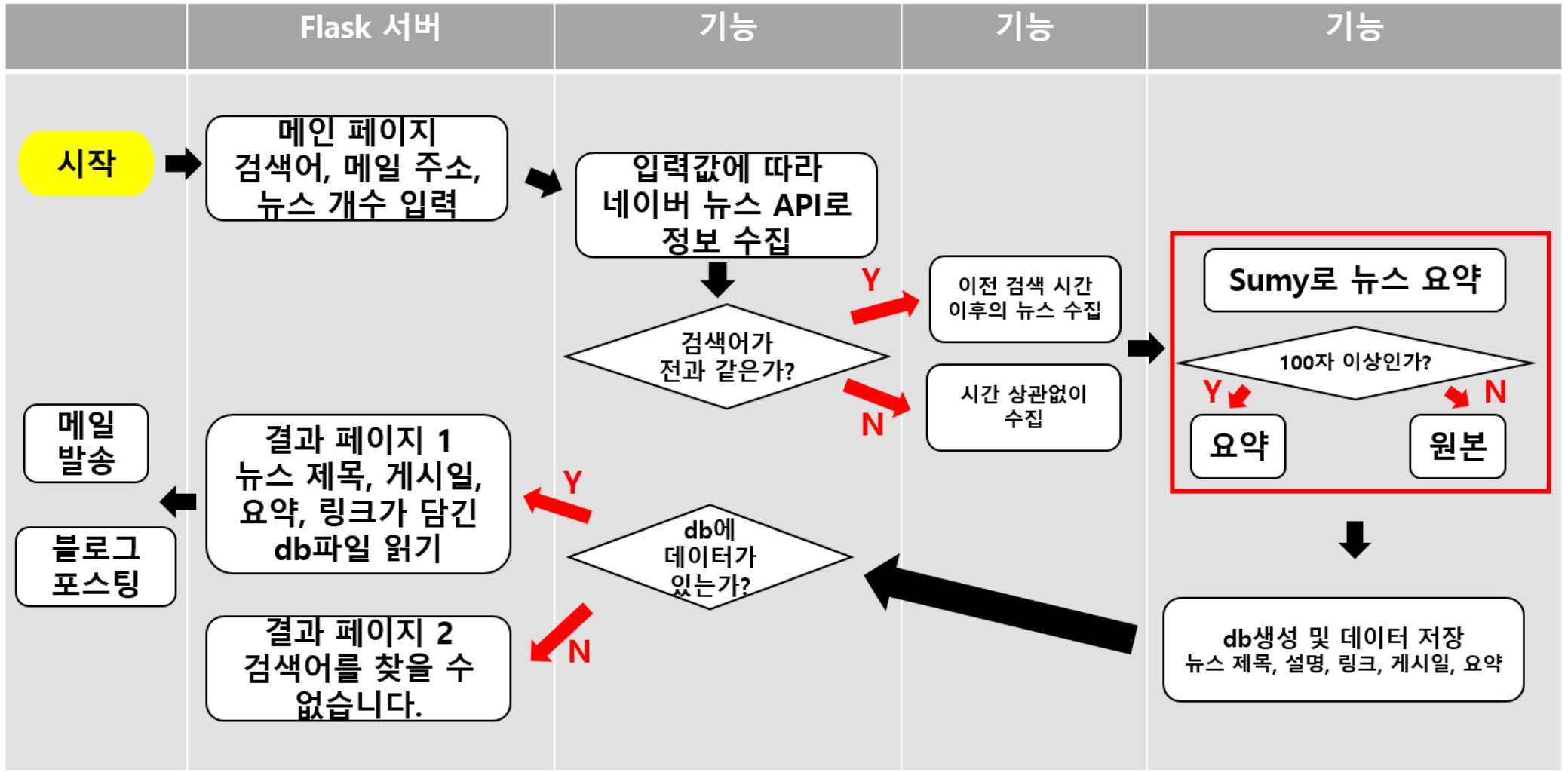
```python
@app.route("/search", methods=["GET"])
def print_news():

    news_token = 0

    search =  request.args["search"]
    mail = request.args["mail"]
    display = request.args["display_in"]
    sort = request.args["sort"]

    # 네이버 developer에서 발급 "https://developers.naver.com/apps/#/myapps/5_EbG5C0vaQvkusHOlP5/overview"
    client_id = "NwwP4DrhiMMqv6I2uU8y"
    client_secret = "jclkobNj7a"

    # url = "https://openapi.naver.com/v1/search/{검색}.jso

    url = "https://openapi.naver.com/v1/search/news.json"

    # API 인증을 위한 Id와 Secret을 headers에 입력
    headers = {
        "X-Naver-Client-Id" : client_id,
        "X-Naver-Client-Secret" : client_secret
    }

    # 검색어, 표시할 뉴스의 수, 정렬기준을 위한 파라미터 설정(d
    params = {
        "query" : search,
        "display" : str(display),
        "sort" : sort
    }

    # 설정한 headers와 params를 get을 통해 url에 있는 사이트에 요청

    response = requests.get(url, headers=headers, params=params)

    if response.status_code == 200:
        response_dict = response.json()

        for item in response_dict["items"]:
    # Title과 Description 처리
            item["title"] = html.unescape(re.sub(r"<.*?>", "", item["title"]))
            item["description"] = html.unescape(re.sub(r"<.*?>", "", item["description"]))

            # Date 처리
            item["pubDate"] = datetime.strptime(item["pubDate"], '%a, %d %b %Y %H:%M:%S %z').strftime('%Y-%m-%d %H시 %M분')
```

Step 3. html 태그 제거 및 날짜 양식 맞추기

ex) 2025-02-25 10시 25분

```python
    # Title과 Description 처리
            item["title"] = html.unescape(re.sub(r"<.*?>", "", item["title"]))
            item["description"] = html.unescape(re.sub(r"<.*?>", "", item["description"]))

            # Date 처리
            item["pubDate"] = datetime.strptime(item["pubDate"], '%a, %d %b %Y %H:%M:%S %z').strftime('%Y-%m-%d %H시 %M분')
```

2-3. 뉴스 요약 생성


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
Y
검색어가
전과 같은가?
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

2-3. 뉴스 요약 생성

```python
# Summary 생성
try:
    news_response = requests.get(item["link"], headers={'User-Agent': 'Mozilla/5.0'})
    news_soup = BeautifulSoup(news_response.text, "html.parser")

    # 다양한 선택자로 기사 본문 탐색 (뉴스 사이트마다 다를 수 있음)
    possible_selectors = [
        {"id": "newsct_article"},  # 기존 선택자
        {"class": "article_body"},  # 추가 선택자 1
        {"class": "news-body"},  # 추가 선택자 2
    ]

    article_text = ""
    for selector in possible_selectors:
        article_body = news_soup.find("div", selector)
        if article_body:
            article_text = article_body.get_text(strip=True)
            break  # 본문을 찾으면 반복 중단

    # 본문이 너무 짧으면 description 사용
    if not article_text or len(article_text) < 100:
        summary_text = item["description"]
    else:
        article_text = re.sub(r'\s+', ' ', article_text).strip()

        # Sumy 요약 적용
        parser = PlaintextParser.from_string(article_text, Tokeni
        summarizer = LexRankSummarizer()
        summary = summarizer(parser.document, 3)

        # 생성된 요약 확인 및 중복 방지
        if summary:
            summary_text = " ".join([str(sentence) for sentence i
        else:
            summary_text = item["description"]

except Exception as e:
    print(f"요약 생성 오류: {e}")
    summary_text = item["description"]  # 오류 발생 시 description 사용

# Summary를 item에 추가
item["summary"] = summary_text
```

Step 1. html 파싱 및 CSS 선택자로 본문 추출

```python
try:
    news_response = requests.get(item["link"], headers={'User-Agent': 'Mozilla/5.0'})
    news_soup = BeautifulSoup(news_response.text, "html.parser")
```

```python
article_text = ""
for selector in possible_selectors:
    article_body = news_soup.find("div", selector)
    if article_body:
        article_text = article_body.get_text(strip=True)
        break  # 본문을 찾으면 반복 중단
```

```
▼ <div id="newsct_article" class="newsct_article _article_body"> == $0
  ▶ <article id="dic_area" class="go_trans _article_content" style="-we
    it-tap-highlight-color: rgba(0,0,0,0)">...</article>
```

2-3. 뉴스 요약 생성

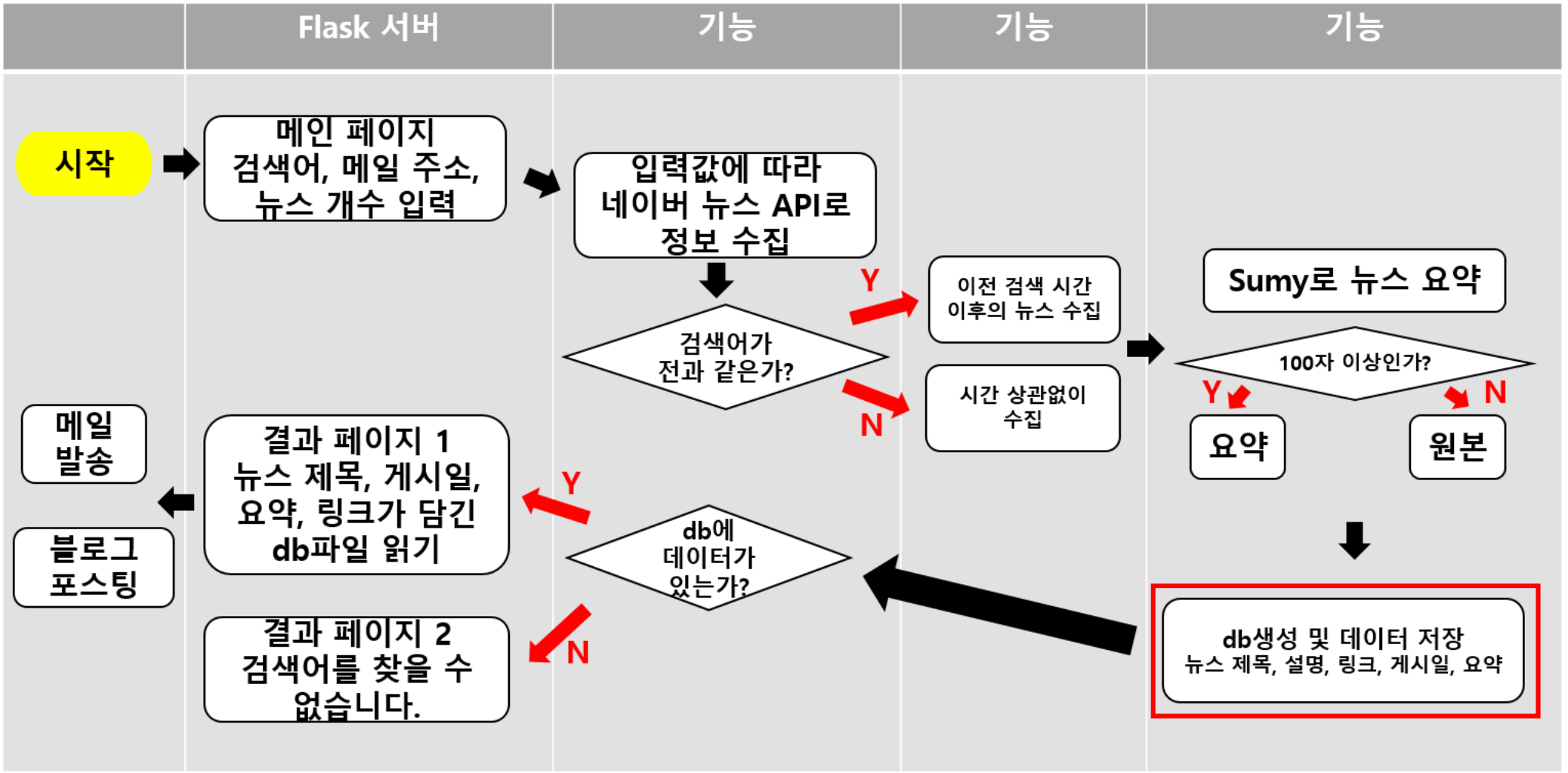
```python
# Summary 생성
try:
    news_response = requests.get(item["link"], headers={'User-Agent': 'Mozilla/5.0'})
    news_soup = BeautifulSoup(news_response.text, "html.parser")

    # 다양한 선택자로 기사 본문 탐색 (뉴스 사이트마다 다를 수 있음)
    possible_selectors = [
        {"id": "newsct_article"},  # 기존 선택자
        {"class": "article_body"},  # 추가 선택자 1
        {"class": "news-body"},  # 추가 선택자 2
    ]

    article_text = ""
    for selector in possible_selectors:
        article_body = news_soup.find("div", selector)
        if article_body:
            article_text = article_body.get_text(strip=True)
            break  # 본문을 찾으면 반복 중단

    # 본문이 너무 짧으면 description 사용
    if not article_text or len(article_text) < 100:
        summary_text = item["description"]
    else:
        article_text = re.sub(r'\s+', ' ', article_text).strip()

        # Sumy 요약 적용
        parser = PlaintextParser.from_string(article_text, Tokenizer("korean"))
        summarizer = LexRankSummarizer()
        summary = summarizer(parser.document, 3)

        # 생성된 요약 확인 및 중복 방지
        if summary:
            summary_text = " ".join([str(sentence) for sentence in summary])
        else:
            summary_text = item["description"]

except Exception as e:
    print(f"요약 생성 오류: {e}")
    summary_text = item["description"]  # 오류 발생 시 description 사용

# Summary를 item에 추가
item["summary"] = summary_text
```

Step 2. 본문 길이에 따라 요약 선택 및 저장

```python
# 본문이 너무 짧으면 description 사용
if not article_text or len(article_text) < 100:
    summary_text = item["description"]
else:
    article_text = re.sub(r'\s+', ' ', article_text).strip()

    # Sumy 요약 적용
    parser = PlaintextParser.from_string(article_text, Tokenizer("korean"))
    summarizer = LexRankSummarizer()
    summary = summarizer(parser.document, 3)
```

```python
# Summary를 item에 추가
item["summary"] = summary_text
```

2-4. 데이터베이스


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
Y
검색어가
전과 같은가?
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

2-4. 데이터베이스

```
# DB 생성 및 저장
os.makedirs("/", exist_ok=True)
conn = sqlite3.connect("news.db")

curs = conn.cursor()

# Summary 필드 추가
sql = """
CREATE TABLE IF NOT EXISTS news(
    title TEXT,
    detail TEXT,
    link TEXT,
    date TEXT,
    keyword TEXT,
    summary TEXT
)
"""

curs.execute(sql)
conn.commit()

# 데이터를 DB에 저장
for item in filtered_list:
    sql = """
    INSERT INTO news(title, detail, link, date, keyword, summary)
    VALUES (?, ?, ?, ?, ?, ?)
    """
    curs.execute(sql, (item["title"], item["description"], item["link"], item["pubDate"], search, item["summary"]))

conn.commit()
conn.close()

# R(Read)
# Sensing.db안에 있는 temp table데이터를
# 전부 SELECT하는 셀
conn_sensing = sqlite3.connect("news.db")

sql_sensing = """
SELECT *
FROM news
"""
df_sensing = pd.read_sql(sql_sensing, conn_sensing)
conn_sensing.close()
```

Step 1. DB 생성

```
# DB 생성 및 저장
os.makedirs("/", exist_ok=True)
conn = sqlite3.connect("news.db")

curs = conn.cursor()

# Summary 필드 추가
sql = """
CREATE TABLE IF NOT EXISTS news(
    title TEXT,
    detail TEXT,
    link TEXT,
    date TEXT,
    keyword TEXT,
    summary TEXT
)
"""

curs.execute(sql)
conn.commit()
```

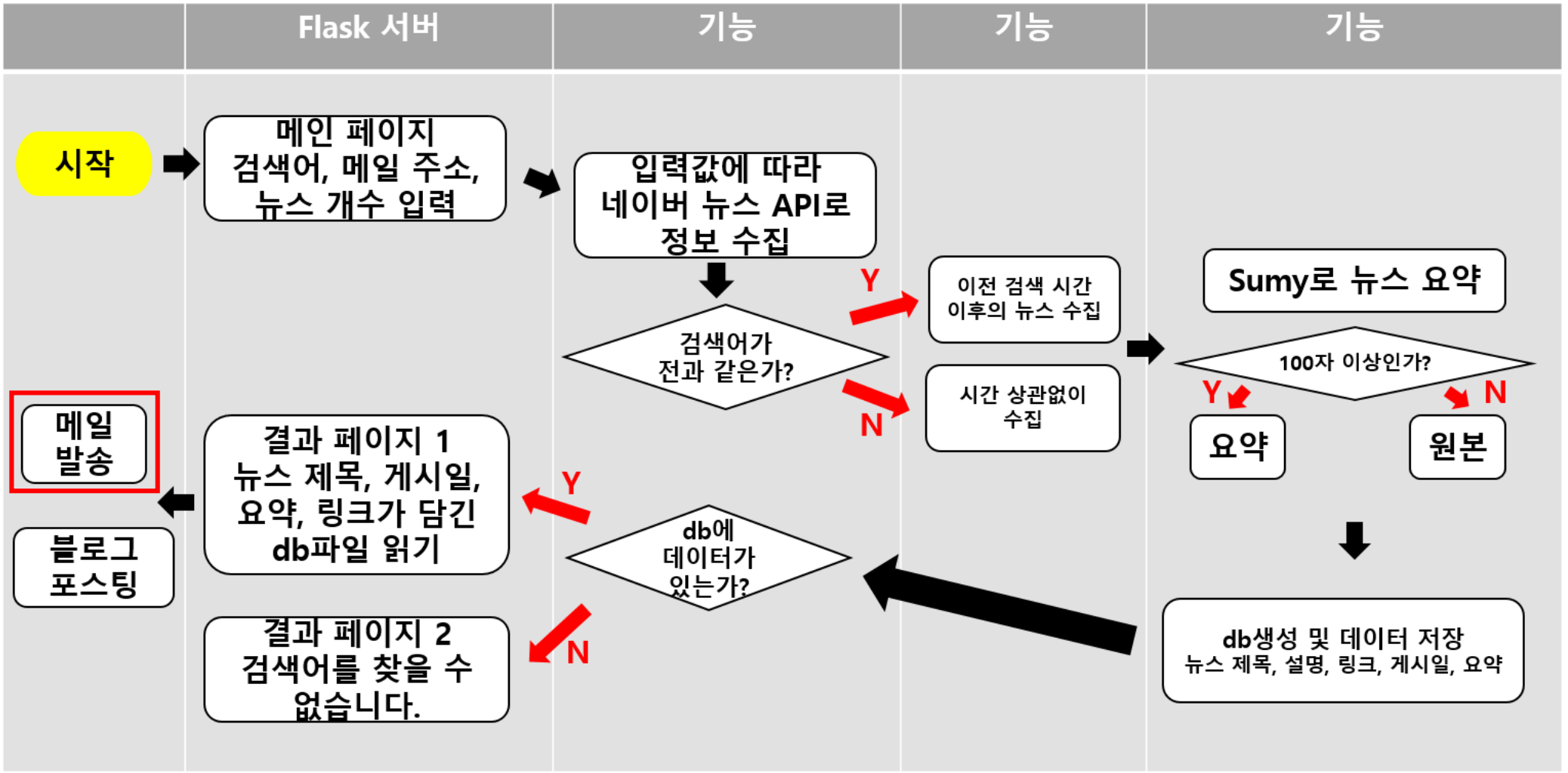
2-4. 데이터베이스

Step 2. DB에 데이터 저장 및 읽기

```python
# DB 생성 및 저장
os.makedirs("/", exist_ok=True)
conn = sqlite3.connect("news.db")

curs = conn.cursor()

# Summary 필드 추가
sql = """
CREATE TABLE IF NOT EXISTS news(
    title TEXT,
    detail TEXT,
    link TEXT,
    date TEXT,
    keyword TEXT,
    summary TEXT
)
"""

curs.execute(sql)
conn.commit()

# 데이터를 DB에 저장
for item in filtered_list:
    sql = """
    INSERT INTO news(title, detail, link, date, keyword, summary)
    VALUES (?, ?, ?, ?, ?, ?)
    """
    curs.execute(sql, (item["title"], item["description"], item["link"],

conn.commit()
conn.close()

# R(Read)
# Sensing.db안에 있는 temp table데이터를
# 전부 SELECT하는 셀
conn_sensing = sqlite3.connect("news.db")

sql_sensing = """
SELECT *
FROM news
"""
df_sensing = pd.read_sql(sql_sensing, conn_sensing)
conn_sensing.close()
```

```python
# 데이터를 DB에 저장
for item in filtered_list:
    sql = """
    INSERT INTO news(title, detail, link, date, keyword, summary)
    VALUES (?, ?, ?, ?, ?, ?)
    """
    curs.execute(sql, (item["title"], item["description"], item["link"], item["pubDate"], search, item["summary"]))

conn.commit()
conn.close()

# R(Read)
# Sensing.db안에 있는 temp table데이터를
# 전부 SELECT하는 셀
conn_sensing = sqlite3.connect("news.db")

sql_sensing = """
SELECT *
FROM news
"""
df_sensing = pd.read_sql(sql_sensing, conn_sensing)
conn_sensing.close()
```

2-5. 이메일 전송


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
검색어가
전과 같은가?
Y
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

2-5. 이메일 전송

```
# 3. 메일 보내는 단계
if filtered_list != []:

    def sendEmail(addr):
        reg = "^[a-zA-Z0-9.+_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$"  # 유효성 검사를 위한 정규표현식
        if re.match(reg, addr):
            smtp.sendmail(my_account, mail, msg.as_string())
            print("정상적으로 메일이 발송되었습니다.")
        else:
            print("받으실 메일 주소를 정확히 입력하십시오.")

    # smpt 서버와 연결
    gmail_smtp = "smtp.gmail.com"  # gmail smtp 주소
    gmail_port = [redacted]  # gmail smtp 포트번호. 고정(변경 불가)
    smtp = smtplib.SMTP_SSL(gmail_smtp, gmail_port)

    # 로그인
    my_account = "test.mail.news12@gmail.com"
    my_password = "pnoa bnce pwpz hkvy"
    smtp.login(my_account, my_password)

    # 메일을 받을 계정
    # mail = "gnada7529@gmail.com"

    # 메일 기본 정보 설정
    now = datetime.now()
    today_date = now.strftime("%Y-%m-%d")
    msg = MIMEMultipart()
    msg["Subject"] = f"{today_date} 뉴스 입니다."  # 메일 제목
    msg["From"] = my_account
    msg["To"] = mail
```

Step 1. smtp 서버와 연결

```
# smpt 서버와 연결
gmail_smtp = "smtp.gmail.com"  # gmail smtp 주소
gmail_port = [redacted]  # gmail smtp 포트번호. 고정(변경 불가)
smtp = smtplib.SMTP_SSL(gmail_smtp, gmail_port)
```

2-5. 이메일 전송

```python
# 3. 메일 보내는 단계
if filtered_list != []:

    def sendEmail(addr):
        reg = "^[a-zA-Z0-9.+_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$"  # 유효성 검사를 위한 정규표현식
        if re.match(reg, addr):
            smtp.sendmail(my_account, mail, msg.as_string())
            print("정상적으로 메일이 발송되었습니다.")
        else:
            print("받으실 메일 주소를 정확히 입력하십시오.")

    # smpt 서버와 연결
    gmail_smtp = "smtp.gmail.com"  # gmail smtp 주소
    gmail_port = 465  # gmail smtp 포트번호. 고정(변경 불가)
    smtp = smtplib.SMTP_SSL(gmail_smtp, gmail_port)

    # 로그인
    my_account = "test.mail.news12@gmail.com"
    my_password = "[redacted]"
    smtp.login(my_account, my_password)

    # 메일을 받을 계정
    # mail = "gnada7529@gmail.com"

    # 메일 기본 정보 설정
    now = datetime.now()
    today_date = now.strftime("%Y-%m-%d")
    msg = MIMEMultipart()
    msg["Subject"] = f"{today_date} 뉴스 입니다."  # 메일 제목
    msg["From"] = my_account
    msg["To"] = mail
```

Step 2. 로그인

```python
# 로그인
my_account = "test.mail.news12@gmail.com"
my_password = "[redacted]"
smtp.login(my_account, my_password)
```

2-5. 이메일 전송

```
# 3. 메일 보내는 단계
if filtered_list != []:

    def sendEmail(addr):
        reg = "^[a-zA-Z0-9.+_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$"  # 유효성 검사를 위한 정규표현식
        if re.match(reg, addr):
            smtp.sendmail(my_account, mail, msg.as_string())
            print("정상적으로 메일이 발송되었습니다.")
        else:
            print("받으실 메일 주소를 정확히 입력하십시오.")

    # smpt 서버와 연결
    gmail_smtp = "smtp.gmail.com"  # gmail smtp 주소
    gmail_port = 465  # gmail smtp 포트번호. 고정(변경 불가)
    smtp = smtplib.SMTP_SSL(gmail_smtp, gmail_port)

    # 로그인
    my_account = "test.mail.news12@gmail.com"
    my_password = "[redacted]"
    smtp.login(my_account, my_password)

    # 메일을 받을 계정
    # mail = "gnada7529@gmail.com"

    # 메일 기본 정보 설정
    now = datetime.now()
    today_date = now.strftime("%Y-%m-%d")
    msg = MIMEMultipart()
    msg["Subject"] = f"{today_date} 뉴스 입니다."  # 메일 제목
    msg["From"] = my_account
    msg["To"] = mail
```

Step 3. 메일 기본 정보 설정

```
# 메일 기본 정보 설정
now = datetime.now()
today_date = now.strftime("%Y-%m-%d")
msg = MIMEMultipart()
msg["Subject"] = f"{today_date} 뉴스 입니다."  # 메일 제목
msg["From"] = my_account
msg["To"] = mail
```

2-5. 이메일 전송

Step 4. 메일 내용 생성

```
# 메일 본문 내용
for m in range(int(display)):
    mail_title = df_sensing["title"].iloc[-int(display)+m]
    mail_description = df_sensing["detail"].iloc[-int(display)+m]
    mail_link = df_sensing["link"].iloc[-int(display)+m]
    mail_pubDate = df_sensing["date"].iloc[-int(display)+m]
    content = f"{m+1}\n\
    뉴스 제목 : {mail_title}\n\n\
    뉴스 설명 : {mail_description}\n\n\
    뉴스 링크 : {mail_link}\n\n\
    뉴스 날짜 : {mail_pubDate}\n\n\
    키워드  : {search} \n\n\
    ------------------------------------------------------------------------

    # MIMEText로 최종 본문 한 번만 추가
    content_part = MIMEText(content, "plain")
    msg.attach(content_part)

# 받는 메일 유효성 검사 거친 후 메일 전송
sendEmail(mail)

# smtp 서버 연결 해제
smtp.quit()
```


```
# 메일 본문 내용
for m in range(int(display)):
    mail_title = df_sensing["title"].iloc[-int(display)+m]
    mail_description = df_sensing["detail"].iloc[-int(display)+m]
    mail_link = df_sensing["link"].iloc[-int(display)+m]
    mail_pubDate = df_sensing["date"].iloc[-int(display)+m]
    content = f"{m+1}\n\
    뉴스 제목 : {mail_title}\n\n\
    뉴스 설명 : {mail_description}\n\n\
    뉴스 링크 : {mail_link}\n\n\
    뉴스 날짜 : {mail_pubDate}\n\n\
    키워드  : {search} \n\n\
    ------------------------------------------------------------------------

    # MIMEText로 최종 본문 한 번만 추가
    content_part = MIMEText(content, "plain")
    msg.attach(content_part)
```

2-5. 이메일 전송

```
    # 메일 본문 내용
    for m in range(int(display)):
        mail_title = df_sensing["title"].iloc[-int(display)+m]
        mail_description = df_sensing["detail"].iloc[-int(display)+m]
        mail_link = df_sensing["link"].iloc[-int(display)+m]
        mail_pubDate = df_sensing["date"].iloc[-int(display)+m]
        content = f"{m+1}\n\
        뉴스 제목 : {mail_title}\n\n\
        뉴스 설명 : {mail_description}\n\n\
        뉴스 링크 : {mail_link}\n\n\
        뉴스 날짜 : {mail_pubDate}\n\n\
        키워드  : {search} \n\n\
        ----------------------------------------------------------------------------------------------------\n\n\n\n"

        # MIMEText로 컨텐츠 본문 한 번만 추가
        content_part = MIMEText(content, "plain")
        msg.attach(content_part)

    # 받는 메일 유효성 검사 거친 후 메일 전송
    sendEmail(mail)

    # smtp 서버 연결 해제
    smtp.quit()
```

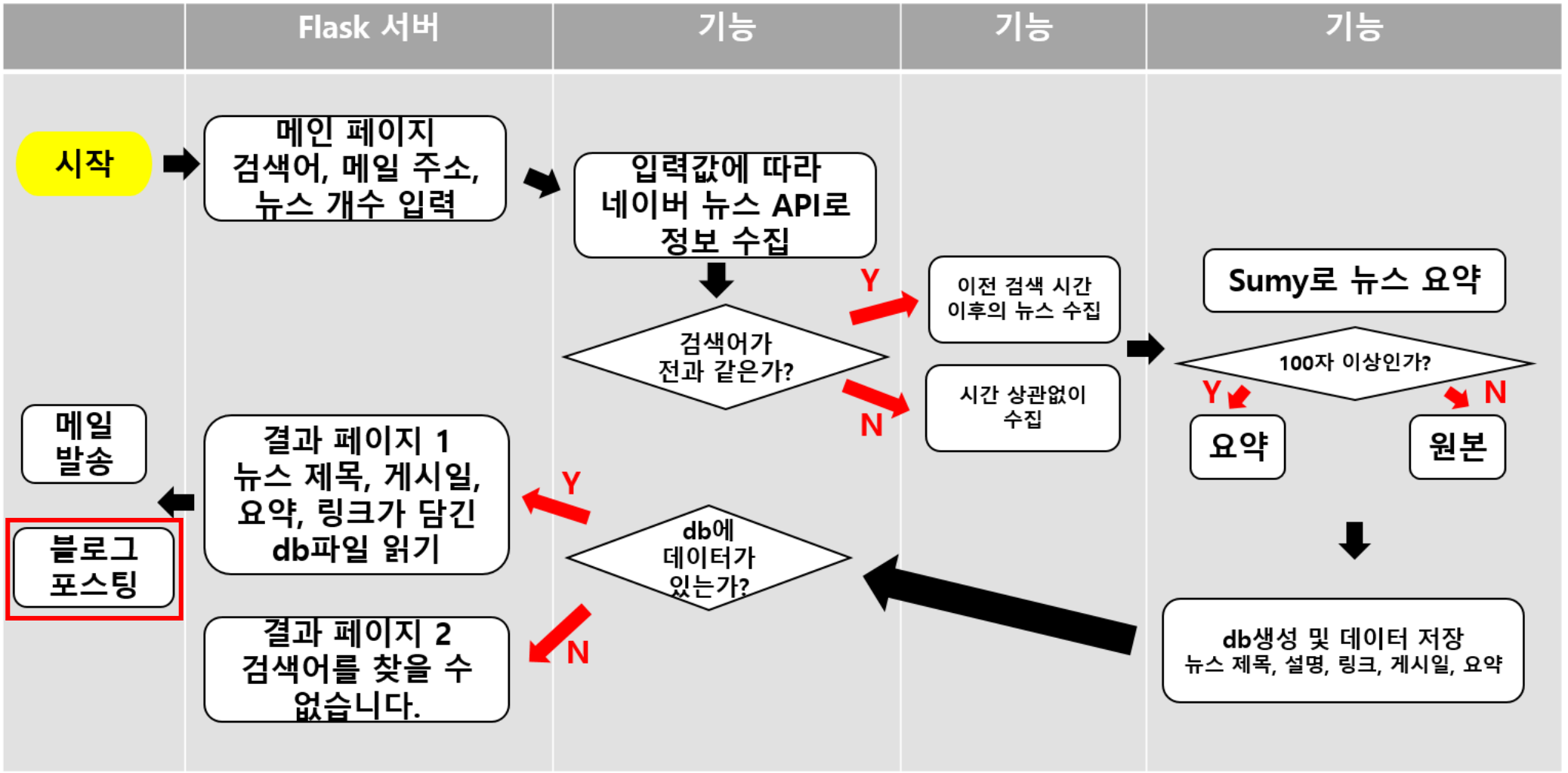
Step 5. def sendEmail(addr) 메일 주소 유효성 판단

```
# 받는 메일 유효성 검사 거친 후 메일 전송
sendEmail(mail)
```

Step 6. smtp 서버 종료

```
# smtp 서버 연결 해제
smtp.quit()
```

2-6. 블로그 포스팅


Flask 서버
기능
기능
기능
시작
메인 페이지
검색어, 메일 주소,
뉴스 개수 입력
입력값에 따라
네이버 뉴스 API로
정보 수집
Y
검색어가
전과 같은가?
N
이전 검색 시간
이후의 뉴스 수집
시간 상관없이
수집
Sumy로 뉴스 요약
100자 이상인가?
Y
N
요약
원본
db생성 및 데이터 저장
뉴스 제목, 설명, 링크, 게시일, 요약
db에
데이터가
있는가?
Y
N
결과 페이지 1
뉴스 제목, 게시일,
요약, 링크가 담긴
db파일 읽기
결과 페이지 2
검색어를 찾을 수
없습니다.
메일
발송
블로그
포스팅

2-6. 블로그 포스팅

-token : Google API를 사용하기 위해서는 '키' 역할을 하는 token이 필요(유효기간이 있음)

-creds : ‘키’인 'token'을 담는 그릇, token을 갱신할 때 필요

2-6. 블로그 포스팅

```
# 4. 블로그에 게시
BLOG_ID = "3244202782305609248" #블로거ID 입력
SCOPES = ['https://www.googleapis.com/auth/blogger', 'https://www.googleapis.com/auth/drive.file']

# API에 요청할 **권한 범위(Scope)**를 정의
# SCOPES = ['https://www.googleapis.com/auth/blogger']

def get_blogger_service_obj():
    creds = None
    # creds는 인증정보를 저장할 변수 선언
    if os.path.exists('auto_token.pickle'): # 토큰 파일이 있으면
        with open('auto_token.pickle', 'rb') as token:
            # *pickle.load()**로 파일에서 인증 정보(creds)를 읽어옵니다
            creds = pickle.load(token)

    if not creds or not creds.valid: # creds가 없거나 유효하지 않다면 실행
        if creds and creds.expired and creds.refresh_token: # creds가 있고 만료됐고 refresh token이 있는 경우
            creds.refresh(Request()) # 호출을 통해 새로운 Access Token을 발급받습니다.

        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'c:Users/kccistc/Desktop/workspace/client_secret.json', SCOPES)
            creds = flow.run_local_server(port=0)

        with open('auto_token.pickle', 'wb') as token:
            # pickle.dump()를 사용해 creds 객체를 바이너리 파일로 저장
            pickle.dump(creds, token)

    blog_service = build('blogger', 'v3', credentials=creds)
    drive_service = build('drive', 'v3', credentials=creds)

    return drive_service,blog_service

drive_handler, blog_handler = get_blogger_service_obj()
content = ""
posting_count = int(display)


for i in range(posting_count):
    keyword = search #태그
    blogger_title = "오늘의 기사" #글 제목
    content += (
                f'<h2>{df_sensing["title"].iloc[-(posting_count - i)]}</h2><br>'
                f'기사 내용 : {df_sensing["detail"].iloc[-(posting_count - i)]}<br><br>'
                f'기사 날짜 : {df_sensing["date"].iloc[-(posting_count - i)]}<br><br>'
                f'원문 기사 링크 : <a href="{df_sensing["link"].iloc[-(posting_count - i)]}">{df_sensing["link"]
                f'<hr>'  # 구분선 추가
                )
data = {
    'content': content,
    'title': blogger_title,
    'labels' : keyword,
    'blog': {
        'id': BLOG_ID, # The identifier of the Blog that contains this Post.
        },}
posts = blog_handler.posts()
res = posts.insert(blogId=BLOG_ID, body=data, isDraft=False, fetchImages=True).execute()
time.sleep(3)
```

Step1.
Token의 상태를 확인하고 token을 갱신, 새로 발급 등을 하여 creds에 담기

```
if os.path.exists('auto_token.pickle')
```

```
if not creds or not creds.valid:
```

```
if creds and creds.expired and creds.refresh_token:
    creds.refresh(Request()) # 호출을 통해 새로운 Acc
```

```
with open('auto_token.pickle', 'rb') as token:
    # *pickle.load()**로 파일에서 인증 정보(cred
    creds = pickle.load(token)
```

```
else:
    flow = InstalledAppFlow.from_client_secrets_file(
        'c:Users/kccistc/Desktop/workspace/client_secret.json', SCOPES)
    creds = flow.run_local_server(port=0)
```

```
with open('auto_token.pickle', 'wb') as token:
    # pickle.dump()를 사용해 creds 객체를 바이너리
    pickle.dump(creds, token)
```

2-6. 블로그 포스팅

```
BLOG_ID = "3244202782305609248" #블로그ID 입력
SCOPES = ['https://www.googleapis.com/auth/blogger', 'https://www.googleapis.com/auth/drive.fi

# API에 요청할 **권한 범위(Scope)**를 정의
# SCOPES = ['https://www.googleapis.com/auth/blogger']

def get_blogger_service_obj():
    creds = None
    if os.path.exists('auto_token.pickle'): # 토큰 파일이 있으면
        with open('auto_token.pickle', 'rb') as token:

            # *pickle.load()**로 파일에서 인증 정보(creds)를 읽어옵니다
            creds = pickle.load(token)

    if not creds or not creds.valid: # creds가 없거나 유효하지 않는다면 실행
        if creds and creds.expired and creds.refresh_token: # creds가 있고 만료됐고 refresh tok
            creds.refresh(Request()) # 호출을 통해 새로운 Access Token을 발급받습니다.

        # 새 인증정보 생성
        # 기존 토큰이 없거나 만료된 상태에서 Refresh Token도 없을 때 실행됩니다.
        else:
            # json file과 scope(url)을 통해 권한을 받는 과정
            # from_client_secrets_file(): Google Cloud Console에서 다운로드한 JSON 파일(client_s
            flow = InstalledAppFlow.from_client_secrets_file(
                'c:Users/kccistc/Desktop/workspace/client_secret.json', SCOPES)

            # run_local_server(): 로컬 서버를 열어 브라우저에서 Google 계정으로 로그인 및 권한 부0
            creds = flow.run_local_server(port=0)

        with open('auto_token.pickle', 'wb') as token:
            # pickle.dump()를 사용해 creds 객체를 바이너리 파일로 저장
            pickle.dump(creds, token)

    blog_service = build('blogger', 'v3', credentials=creds)

    return blog_service

drive_handler, blog_handler = get_blogger_service_obj()
content = ""
posting_count = int(display)

for i in range(posting_count):
    keyword = search #태그
    blogger_title = "오늘의 기사" #글 제목
    content += (
                f'<h2>{df_sensing["title"].iloc[-(posting_count - i)]}</h2><br>'
                f'기사 내용 : {df_sensing["detail"].iloc[-(posting_count - i)]}<br><br>'
                f'기사 날짜 : {df_sensing["date"].iloc[-(posting_count - i)]}<br><br>'
                f'원문 기사 링크 : <a href="{df_sensing["link"].iloc[-(posting_count - i)]}">{d
                f'<hr>'  # 구분선 추가
            )
data = {
    'content': content,
    'title': blogger_title,
    'labels' : keyword,
    'blog': {
        'id': BLOG_ID, # The identifier of the Blog that contains this Post.
        },}
posts = blog_handler.posts()
res = posts.insert(blogId=BLOG_ID, body=data, isDraft=False, fetchImages=True).execute()
time.sleep(3)
```

Step 2. Blogger 서비스 객체 생성

```
blog_service = build('blogger', 'v3', credentials=creds)

return blog_service
```

2-6. 블로그 포스팅

```python
# 4. 블로그에 게시
BLOG_ID = "3244202782305609248" #블로거ID 입력
SCOPES = ['https://www.googleapis.com/auth/blogger', 'https://www.googleapis.com/auth/drive.file']

# API에 요청할 **권한 범위(Scope)**를 정의
# SCOPES = ['https://www.googleapis.com/auth/blogger']

def get_blogger_service_obj():
    creds = None
    # creds는 인증정보를 저장할 변수 선언
    if os.path.exists('auto_token.pickle'): # 토큰 파일이 있으면
        with open('auto_token.pickle', 'rb') as token:
            # *pickle.load()**로 파일에서 인증 정보(creds)를 읽어옵니다
            creds = pickle.load(token)

    if not creds or not creds.valid: # creds가 없거나 유효하지 않다면 실행
        if creds and creds.expired and creds.refresh_token: # creds가 있고 만료됐고 refresh token이 있는 경우
            creds.refresh(Request()) # 호출을 통해 새로운 Access Token을 발급받습니다.

        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'c:Users/kccistc/Desktop/workspace/client_secret.json', SCOPES)
            creds = flow.run_local_server(port=0)

        with open('auto_token.pickle', 'wb') as token:
            # pickle.dump()를 사용해 creds 객체를 바이너리 파일로 저장
            pickle.dump(creds, token)

    blog_service = build('blogger', 'v3', credentials=creds)
    drive_service = build('drive', 'v3', credentials=creds)

    return drive_service,blog_service

drive_handler, blog_handler = get_blogger_service_obj()
content = ""
posting_count = int(display)


for i in range(posting_count):
    keyword = search #태그
    blogger_title = "오늘의 기사" #글 제목
    content += (
                f'<h2>{df_sensing["title"].iloc[-(posting_count - i)]}</h2><br>'
                f'기사 내용 : {df_sensing["detail"].iloc[-(posting_count - i)]}<br><br>'
                f'기사 날짜 : {df_sensing["date"].iloc[-(posting_count - i)]}<br><br>'
                f'원문 기사 링크 : <a href="{df_sensing["link"].iloc[-(posting_count - i)]}">{df_sensing["link"]
                f'<hr>'  # 구분선 추가
            )
data = {
    'content': content,
    'title': blogger_title,
    'labels' : keyword,
    'blog': {
        'id': BLOG_ID, # The identifier of the Blog that contains this Post.
        },}
posts = blog_handler.posts()
res = posts.insert(blogId=BLOG_ID, body=data, isDraft=False, fetchImages=True).execute()
time.sleep(3)
```

Step 3. 포스팅 내용 작성 및 자동 포스팅

```python
content = ""
posting_count = int(display)
for i in range(posting_count):
    keyword = search #태그
    blogger_title = "오늘의 기사" #글 제목
    content += (
                f'<h2>{df_sensing["title"].iloc[-(posting_count - i)]}</h2><br>'
                f'기사 내용 : {df_sensing["detail"].iloc[-(posting_count - i)]}<br><br>'
                f'기사 날짜 : {df_sensing["date"].iloc[-(posting_count - i)]}<br><br>'
                f'원문 기사 링크 : <a href="{df_sensing["link"].iloc[-(posting_count - i)]
                f'<hr>'  # 구분선 추가
            )
data = {
    'content': content,
    'title': blogger_title,
    'labels' : keyword,
    'blog': {
        'id': BLOG_ID, # The identifier of the Blog that contains this Post.
        },}
posts = blog_handler.posts()
res = posts.insert(blogId=BLOG_ID, body=data, isDraft=False, fetchImages=True).execute()
```

3. 문제점 & 해결

3-1. 문제점

짧은 시간동안 같은 키워드로 검색을 하면 이전에 검색했던 내용이

중복으로 DB에 들어가는 문제 발생

중복 →

101	SKT, K-AI 얼라이언스와 'MWC 2025' 참가...AI 혁신 기술 선보여	리벨리온(Rebellions)은 AI반도체 스타트업으로, 지난해 12월 사피온코리아...	http://www.sisacast.kr/news/articleView.html?i...	2025-02-24 12시 38분

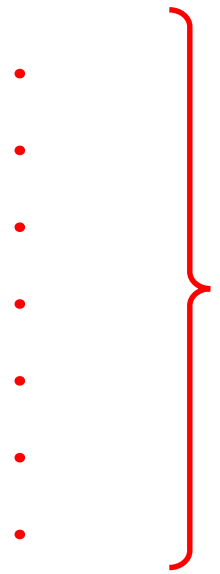
중복 →

102	SKT, K-AI 얼라이언스와 'MWC 2025' 참가...AI 혁신 기술 선보여	리벨리온(Rebellions)은 AI반도체 스타트업으로, 지난해 12월 사피온코리아...	http://www.sisacast.kr/news/articleView.html?i...	2025-02-24 12시 38분

3-2. 해결 과정

1. 이전에 검색한 시간 저장, 새롭게 검색한 시간을 측정

```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```



각종 코드(블로그, 이메일 작성, DB 저장 등)

```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

3-2. 해결 과정

1. 이전에 검색한 시간 저장, 새롭게 검색한 시간을 측정

```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```

- 각종 코드(블로그, 이메일 작성, DB 저장 등)


```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

3-2. 해결 과정

```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```

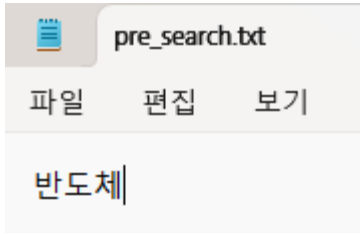
각종 코드(블로그, 이메일 작성, DB 저장 등)

2. 이전에 검색한 키워드를 text 파일로 저장

```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

```
    if pre_search == search:
        with open("END_TIME_FILE.txt", "r") as file:
            end_time = file.read().strip()
            end_time = datetime.strptime(end_time, "%Y-%m-%d %H:%M:%S")

            for i in range(int(display)):
                if (response_dict["items"][i]["pubDate"] >= end_time) &
                    filtered_list.append(response_dict["items"][i])
    else:
        for i in range(int(display)):
            filtered_list.append(response_dict["items"][i])

else:
    for i in range(int(display)):
        filtered_list.append(response_dict["items"][i])

    print("필터가되지않았습니다")

pre_search = "pre_search.txt"

with open(pre_search, "w") as file:
    file.write(search)
```

이전 검색 키워드 저장




```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```

각종 코드(블로그, 이메일 작성, DB 저장 등)

```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

```
    if pre_search == search:
        with open("END_TIME_FILE.txt", "r") as file:
            e  time = file.read().strip()
            e  time = datetime.strptime(end_time, "%Y-%m-%d %H:%M:%S")

            fo  in range(int(display)):
                 (response_dict["items"][i]["pubDate"] >= end_time) &
                  filtered_list.append(response_dict["items"][i])
    else:
        for i in  nge(int(display)):
            filte  list.append(response_dict["i  ms"][i])

else:
    for i in range(i  (display)):
        filtered_lis  ppend(response_dict["item  ][i])

    print("필터가되지않  습니다")

pre_search = "pre_searc  txt"

with open(pre_search, "w  as file:
    file.write(search)
```

이전 검색 키워드와 같으면 필터링

```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```

각종 코드(블로그, 이메일 작성, DB 저장 등)

```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

```
    if pre_search == search:
        with open("END_TIME_FILE.txt", "r") as file:
            end_time = file.read().strip()
            end_time = datetime.strptime(end_time, "%Y-%m-%d %H:%M:%S")

            for i in range(int(display)):
                if (response_dict["items"][i]["pubDate"] >= end_time) &
                    filtered_list.append(response_dict["items"][i])
    else:
        for i in range(int(display)):
            filtered_list.append(response_dict["items"][i])

else:
    for i in  ge(int(display)):
        filter  list.append(response_dict["items"][i])

    print("필터가 적용되지 않았습니다")

pre_search = "pre_se  h.txt"

with open(pre_search,  ) as file:
    file.write(search)
```

그렇지 않다면 전부 저장

```
start_time = datetime.now()
start_time_str = start_time.strftime('%Y-%m-%d %H:%M:%S') #문자열로
start_time = datetime.strptime(start_time_str, '%Y-%m-%d %H:%M:%S')
```

각종 코드(블로그, 이메일 작성, DB 저장 등)

```
END_TIME_FILE = "END_TIME_FILE.txt"
end_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
with open(END_TIME_FILE, "w") as file:
    file.write(end_time)
```

3-3. 추가 개선 사항

1. 뉴스 기사 필터링 문제
   ex) “Python” -> “반도체” -> “Python” 검색 시 “Python” 뉴스 중복!

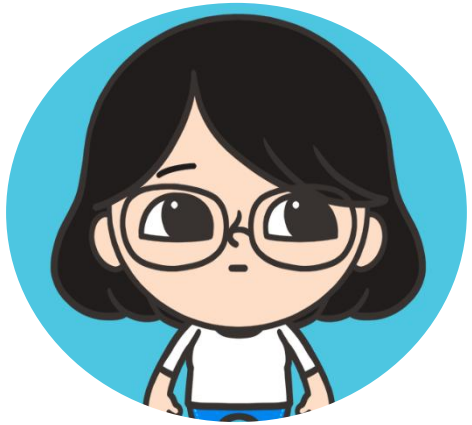
2. 뉴스 요약하는 Sumy 라이브러리 한계
   자바 설치 필수!

4. 구현 영상


127.0.0.1:5000
대한상공회의소
서울기술교육센터
검색어를 입력하세요
이메일을 입력하세요
뉴스의 개수를 입력하세요(1~20개)
정확도순
검색
네이버 뉴스 바로가기

5. 팀원소개

5. 팀원소개



권*

이메일 구현
PPT 제작(맡은부분)

김*우

Sumy 활용 뉴스 요약
PPT 제작(맡은부분)

김*은

블로그 포스팅
DB 생성 및 저장
PPT 제작(맡은부분)

김*서

API 구현
발표
PPT 제작(맡은부분)

6. 부록

6-1. 사용된 설치 파일 및 라이브러리

```
# 설치 파일 정리
!pip3 install requests

!pip install --upgrade jpype1

# sumy 라이브러리는 내부적으로 jpype를 사용하여 java 를 실행하므로 설치 필요
!pip install konlpy

# 한국어를 사용하기 위한 형태소분석기
!pip install sumy

# sumy 라이브러리 설치
!pip install google-auth google-auth-oauthlib
google-auth-httplib2 google-api-python-client

!pip install --upgrade oauth2client
```

```
import time
import os
import json
import requests

# html entity 변환을 위한 라이브러리(html.unescape)
import html

# html tag 제거를 위한 라이브러리(re.sub())
import re

# 게시일 표기 변경을 위한 라이브러리
from datetime import datetime
import sqlite3
import pandas as pd

### 메일쓰기 위한 모듈 ####
import smtplib  # SMTP 사용을 위한 모듈
from email.mime.multipart import MIMEMultipart  # 메일의 Data 영역의 메시지를 만드는 모듈
from email.mime.text import MIMEText  # 메일의 본문 내용을 만드는 모듈

# sumy 라이브러리를 위한 import
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lex_rank import LexRankSummarizer

# 한국어 문장 분리용 추가
from konlpy.tag import Kkma

from bs4 import BeautifulSoup

import sys
import pickle
from oauth2client import client
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
from flask import Flask, request, render_template
```