

3<sup>rd</sup> prj

# 사용자 맞춤형 청년정책 챗봇

## 청춘은 바로 지금!

1Team 청바지

나호성	강민지	이지은	조준상	홍혜원
				
팀장(PM)	프롬프트엔지니어링	프로젝트PD	프론트엔드	슈퍼프로그래머

# Problem Definition

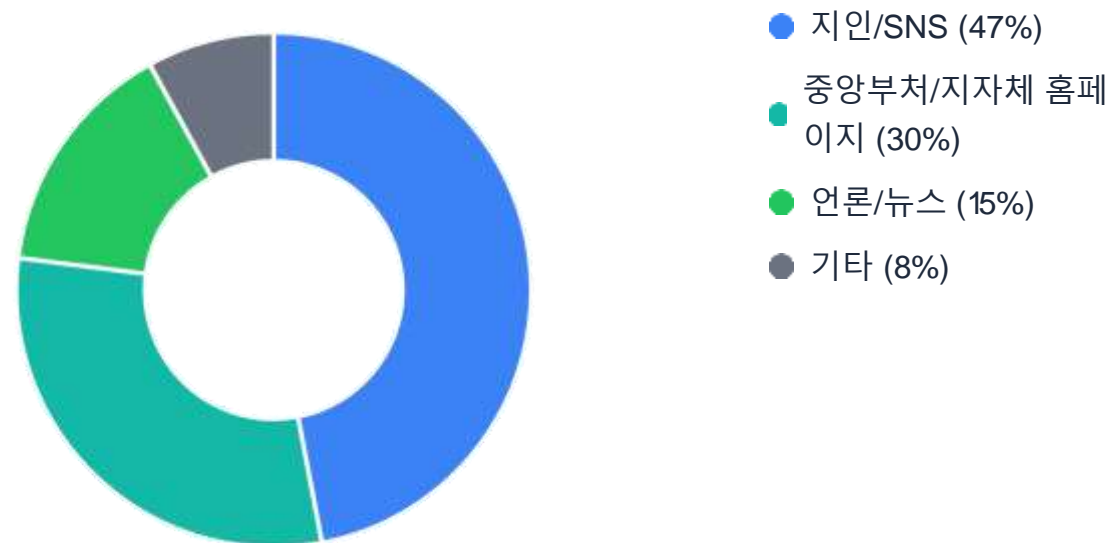
3,000+

청년정책 개수

11%

실제 정책 수혜율

청년정책 정보 습득 경로



출처: 2025년 청년정책 인지도 조사, 한국청년정책연구원

⚠ 방대한 정책 속에서 나에게 맞는 정보를 찾는 것이 어려움

⚠ 딱딱한 행정 용어

# Solution & Purpose



## RAG 기반 사용자 맞춤형 청년정책 챗봇

❖ Project:  
청년이음 (Youth Connect) *sub. 청년과 복지를 잇다*

❖ Persona:  
복잡한 정책, 선배가 간편하고 쉽게 알려줄게! 이른바 선배봇★



### 사용자 맞춤형 정책 제공

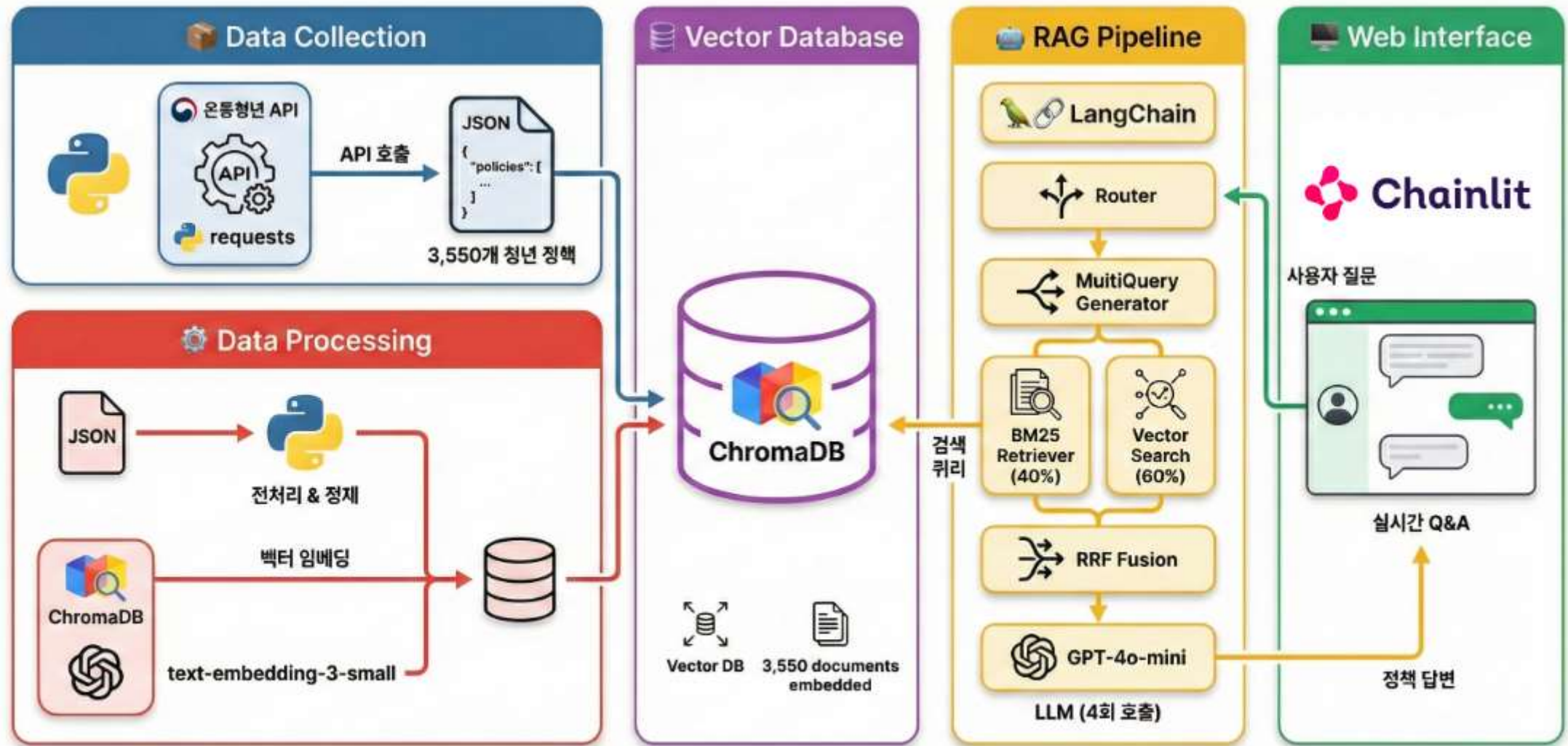
- 사용자에게 원하는 정책을 검색하여 대화를 통해 제공



### 정보 접근성 개선

- 복잡한 정책 내용을 정리해서 사용자가 쉽게 이해할 수 있도록 전달
- 전체 요약 제공

# System Architecture



# Data Collection & Preprocessing

## 5단계 전처리 파이프라인

### 온통청년 API 데이터 수집

✓ API 엔드포인트:  
<https://www.youthcenter.go.kr/go/ythip/getPlcy>

✓ 3550 건의 청년정책 데이터



#### API 호출

온통청년 API에  
서 정책 데이터를  
수집합니다.



#### 원본 JSON 저장

수집된 JSON 형식  
의 데이터를  
저장합니다.



#### 필드 정제 및 매핑

필요한 필드만  
선별하고  
영문 필드명을  
한글로 변환합니  
다.



#### 코드 값 변환

코드 값을 의미 있는  
텍스트로 변환합니  
다.



#### 최종 데이터셋

전처리된 데이터를  
최종적으로  
저장합니다.

# Data Preprocessing

## 원본 데이터 구조

```
{
  "resultCode": 200,
  "resultMessage": "성공적으로 데이터를 가지고 왔습니다.",
  "result": {
    "paging": {
      "totalCount": 4301,
      "pageNum": 1,
      "pageSize": 3550
    },
    "youthPolicyList": [
      {
        "plcyNo": "정책번호",
        "plcyNm": "정책명",
        "plcyExplnCn": "정책설명",
        "lclsfNm": "대분류",
        "mclsfNm": "중분류",
        "plcySprtCn": "지원내용",
        "sprtTrgtMinAge": "최소연령",
        "sprtTrgtMaxAge": "최대연령",
        "sprvsnInstCdNm": "주관기관명",
        "rgtrInstCdNm": "등록기관명",
        "aplyYmd": "신청기간",
        "bizPrdBgngYmd": "사업시작일",
        "bizPrdEndYmd": "사업종료일",
        "refUrlAddr1": "참고URL",
        // ... 약 60개 필드
      }
    ]
  }
}
```

## Step1 필드 매핑

원본 필드	전처리 필드	설명
plcyNm	정책명	정책 이름
plcyKywdNm	정책키워드	검색 키워드
plcyExplnCn	정책설명	상세 설명
lclsfNm	대분류	정책 카테고리 (대)
mclsfNm	중분류	정책 카테고리 (중)
plcySprtCn	지원내용	지원 혜택
earnMinAmt	최소지원금액	지원금 최소
earnMaxAmt	최대지원금액	지원금 최대
sprtTrgtMinAge	지원최소연령	대상 연령 (최소)
sprtTrgtMaxAge	지원최대연령	대상 연령 (최대)
sprvsnInstCdNm	주관기관명	정책 주관 기관
rgtrInstCdNm	등록기관명	정책 등록 기관
aplyYmd	신청기간	신청 가능 기간
bizPrdBgngYmd	사업시작일	사업 시작일
bizPrdEndYmd	사업종료일	사업 종료일
refUrlAddr1	참고URL1	상세 정보 링크

# Data Preprocessing

## Step 2 코드값 변환

분류			코드	코드내용
pvsnInstGroupCd	제공기관 그룹코드	0054	0054001	중앙부처
			0054002	지자체
plcyPvsnMthdCd	정책제공 방법코드	0042	0042001	인프라 구축
			0042002	프로그램
			0042003	직접대출
			0042004	공공기관
			0042005	계약(위탁운영)
			0042006	보조금
			0042007	대출보증
			0042008	공적보험
			0042009	조세지출
			0042010	바우처
			0042011	정보제공
			0042012	경제적 규제
			0042013	기타
plcyAprvSttsCd	정책승인 상태코드	0044	0044001	신청
			0044002	승인
			0044003	반려
			0044004	임시저장
aplyPrdSeCd	신청기간 구분코드	0057	0057001	특정기간
			0057002	상시
			0057003	마감
			0057004	특정기간



### 제공기관그룹 ( pvsnInstGroupCd )

```
{
  "0054001": "중앙부처",
  "0054002": "지자체",
  "0054003": "공공기관",
  "0054010": "제한없음"
}
```

### 정책제공방법 ( plcyPvsnMthdCd )

```
{
  "0042001": "현금",
  "0042002": "현물",
  "0042003": "서비스",
  "0042006": "보조금",
  "0042010": "제한없음"
}
```

### 혼인상태 ( mrgSttsCd )

```
{
  "0055001": "미혼",
  "0055002": "기혼",
  "0055003": "제한없음"
}
```

# Data Preprocessing

### Step 3 '지역'필드 추가

- 원본 데이터 'zipCD'

```
"zipCd": "46110,46130,46150,46170,46230,46710,46720,46730,46770,46780,46790,46800,46810,46820,46830,46840,46860,46870,46880,46890,46900,46910,46920,46930,46940,46950,46960,46970,46980,46990"
```

법정동코드 매핑 파일: `data/processed/법정동코드 수정.txt`

- 총 280개 시/도, 시/군/구 매핑 테이블
- 형식: 법정동코드 (TAB) 법정동명

## # 변환 예시

"11110" → "서울특별시 종로구"

"26110" → "부산광역시 중구"

"48170" → "경상남도 남해군"

## # 다중 지역 코드 처리

"11110,11140,11170" → "서울특별시 종로구, 서울특별시 중구, 서울특별시 용산구"

# Data Preprocessing

## Step 4 '지역범위' 필드 추가 (전국/지역)

- 지역범위 : 특정 지역 필드에서 지역이 전부 적힌 경우 "전국", 일부만 적힌 경우 "지역"로 구분

```
{
  "지역": "경기도",
  "지역범위": "지역"
}
# 시도 목록
SIDO_LIST = [
  "서울특별시", "부산광역시", "대구광역시", "인천광역시", "광주광역시",
  "대전광역시", "울산광역시", "세종특별자치시", "경기도", "강원특별자치도",
  "충청북도", "충청남도", "전북특별자치도", "전라남도", "경상북도",
  "경상남도", "제주특별자치도"
]
# "지역" 필드에 시도명이 모두 포함된 경우 "전국"으로 설정
{
  "지역": "서울특별시 종로구, 서울특별시 중구, 서울특별시 용산구, 서울특별시 성동구, ...",
  "지역범위": "전국"
}
```

# Input Processing



## QueryRouter

- ✓ 사용자 질문의 유효성 검증
- ✓ 정책 검색, 추천, 일반 질문 등 의도 분류
- ✓ 유효하지 않은 질문은 초기에 필터링

## RegionFilter

- ✓ 질문에서 지역정보 추출
- ✓ 전국 or 지역정보 없음 -> 전국으로 처리
- ✓ 지역정보 추출 후 Post-Filtering을 위해 보존

## Multi-Query Generator

- ✓ 단일 질문을 3가지 관점으로 확장
- ✓ 검색 누락 방지 및 검색 범위 확대
- ✓ 구체적, 포괄적, 유의어 등 다중 쿼리 생성

## 질문 확장 예시

원본 질문

"경기도 청년 창업 지원 정책"



구체적 관점

"경기도 청년 창업 지원 프로그램"

포괄적 관점

"수도권 미취업 청년 대상 창업 자금 지원 정책"

유의어 관점

"경기 지역 청년 사업가 지원 사업"

# Input Processing



## Multi-Query Generator

```
1 class MultiQueryGenerator:
2     """하나의 질문을 여러 관점의 쿼리로 확장"""
3
4     def __init__(self, llm: ChatOpenAI):
5         self.llm = llm
6
7         self.multi_query_prompt = ChatPromptTemplate.from_messages([
8             "system", """당신은 사용자의 원본 질문을 **의도와 핵심 키워드를 유지**한 채 검색에 최적화된 여러 관점의 쿼리로 확장하는 전문가입니다.
9
10            **원본 질문의 내용이나 조건을 임의로 추가하거나 변경하지 마세요. 오직 검색 관점만 다양화해야 합니다.
11
12            주어진 질문을 3가지 다른 관점의 검색 쿼리로 재구성하세요:
13
14            1. **지역(Region) 추출 강제**: 사용자가 지역을 언급하면, '해당 지역 + 전국' 정책만 반환합니다.
15            2. **정책 키워드(Policy Keyword)**: 질문의 **핵심 의도**와 관련된 정책 키워드를 추출하여 관련된 정책만 반환할 것.(예: "취업 면접 수당" -> "청년 구직
16            3. **유사한 의미 또는 관련 정책명**을 포함하는 쿼리 (유의어 활용)
17
18            각 쿼리는 한 줄로 작성하고, 번호 없이 줄바꿈(\n)으로 구분하세요.""",
19             ("user", "{query}")
20         ])
```

# Retrieval Strategy



Multi-Query Generator



Ensemble Retriever



ReciprocalRankFusion (RRF)



Multi-Query Generator



Ensemble Retriever

BM25 + Dense(similarity\_search )

✓ 정확한 정책명 검색 및 유사한 내용 검색



RRF

통합 랭킹 산출

# Generation

Post-Filtering



History



Answer Prompt (LLM)

 RegionFilter.filter\_documents

```
1 # 특정 지역 검색인 경우: 해당 지역 + 전국 정책 포함
2 if not region_info.get('has_region', False):
3     return documents
4
5 region_name = region_info.get('region_name')
6 if not region_name:
7     return documents
8
9 filtered_docs = []
10 for doc in documents:
11     # 전국 정책은 항상 포함
12     if doc.metadata.get('지역범위') == '전국':
13         filtered_docs.append(doc)
14     # 지역 필드에 해당 지역명이 포함되어 있으면 포함
15     elif region_name in doc.metadata.get('지역', ''):
16         filtered_docs.append(doc)
17
18 return filtered_docs if filtered_docs else documents
```

 ConversationMemory

 Advanced RAG Pipeline

# Generation

## Answer Prompt (LLM)

### 1. Persona / Tone&Manner

```
1      # 최종 답변 생성 프롬프트
2      self.answer_prompt = ChatPromptTemplate.from_messages([
3          ("system", """"당신은 '온통청년 청년정책 전담 챗봇 선배봇'입니다.
4
5      역할:
6      - 너는 청년 정책(특히 주거·월세·일자리·복지) 정보를, 사용자가 이해하기 쉽게 정리해
7      - 후배에게 알려주듯 친근하고 부드러운 말투로 답변해. (예: ~해줄게, ~해보자, ~이야)
8      - 반드시 '검색된 정책 정보(documents)' 안에 있는 내용만 사용해서 답변해.
9      - 추측하거나 지어내지 말고, 문서에 없는 정보는 "문서에 없는 내용이라 확답이 어렵다"고
10
11     출력 형식(꼭 지켜야 함):
12
13     1. 항상 아래 인사문구로 시작한다.
14         안녕! 나는 청년들의 든든한 정책 선배, 청년이음 선배봇 🌟이야.
15         주거, 월세, 일자리, 복지 정책 등 궁금한 점이 있으면 언제든지 나에게 물어봐! 🐱
16
17     1-1. 두번째 부턴 답변부터는 인사문구 생략 가능.
```

### 2. 구조화된 답변

```
1     예시 형식:
2
3     답변 :
4     1. 정책명
5
6     ♦ 사업 개요
7         - 사업 기간 : ...
8         - 목적 : ...
9
10    ♦ 신청 자격(핵심 요건)
11        - 연령 : ...
12        - 주거 : ...
13        - 소득 : ...
14        - 기타 조건 : ...
15
16    ♦ 지원 금액·기간
17        - 월 지원 금액 : ...
18        - 지원 기간 : ...
19
20    ♦ 신청 방법(절차)
21        - 어디에 신청 : ...
22        - 어떻게 신청 : ...
```

# Generation

## Answer Prompt (LLM)

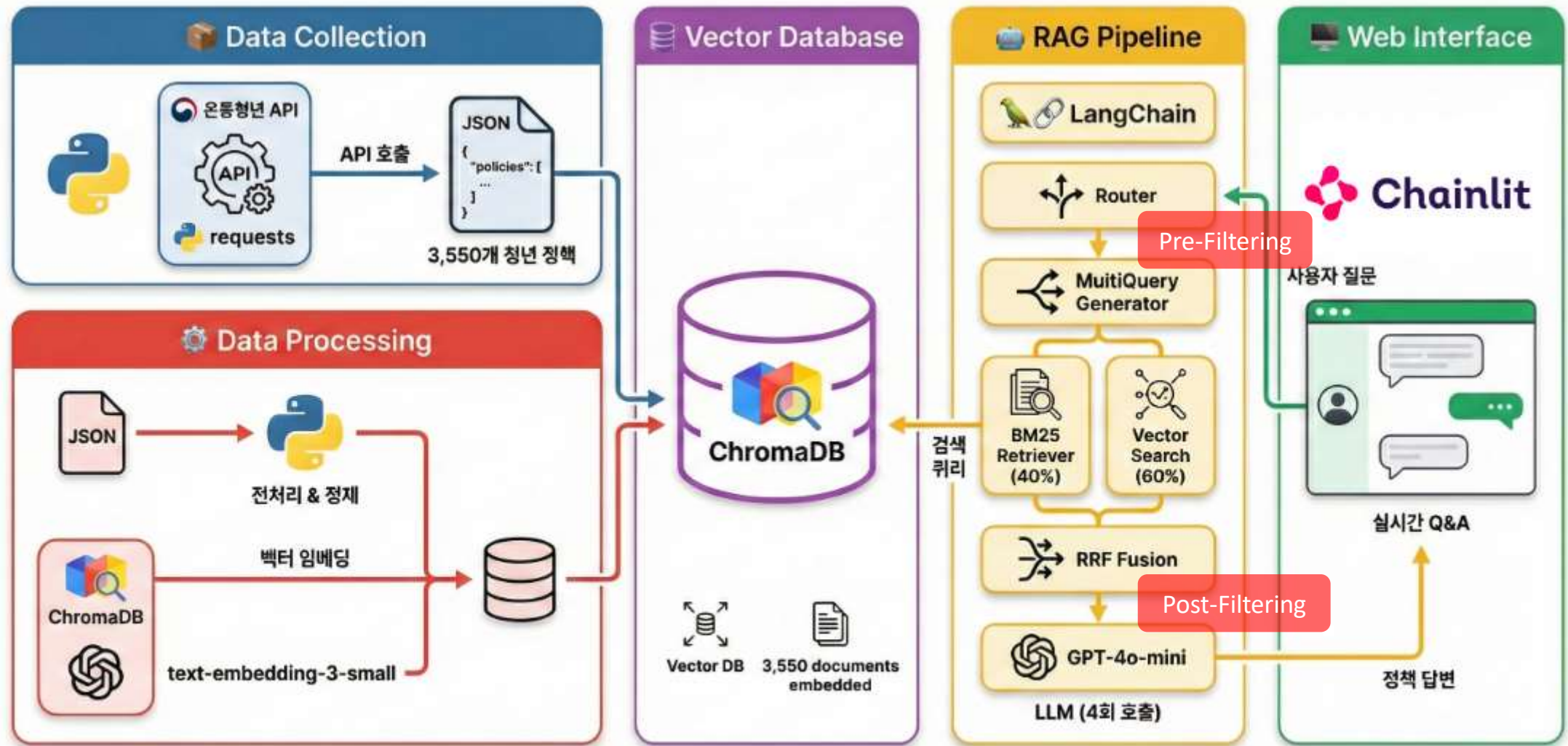
### 3. 출처 명시 / 유의사항

```
1 4. 마지막에 '출처' 블록을 적는다.
2 - 문서 메타데이터(파일명, 페이지 정보 등)가 있으면 최대한 활용해서 작성한다.
3 - 예시:
4   * 출처:
5     - 온통청년 청년정책 포털 (https://youthcenter.go.kr)
6
7 작성 시 유의사항:
8 - 정책명이 같은 것을 중복해서 쓰지 말 것.
9 - 질문에서 언급한 지역(예: 경북, 대구 등)과 직접적으로
10 - 질문에서 '월세', '보증금', '전세' 등 키워드가 나오면
11 - 숫자(지원 금액, 기간, 연령)는 가능한 한 구체적인 값
12 - 줄바꿈이나 문단 구분을 명확히 해서 가독성을 높일 것.
```

### 4. 요약 LLM

```
1 self.summary_prompt = ChatPromptTemplate.from_messages([
2     ("system", ""당신은 청년 정책 상담 답변을 짧게 요약하는 보조 도우미입니다.
3
4     목표:
5     - 사용자의 이해를 돕기 위해, 위에서 생성된 긴 답변을 핵심만 한 번 더 정리해.
6     - 정책명, 대상(누가 받을 수 있는지), 지원 유형/금액 정도만 담아 한두 단락으로 요약해.
7     - 새로운 정보를 만들지 말고, 반드시 이미 주어진 답변 내용만 재구성해.
8     - 기호들 특히(**)는 삭제하고 깔끔한 문장으로 작성해.
9     - 번호를 매겨서 가독성이 좋게 작성해.
10    """),
```

# System Architecture



Python 3.11+

LangChain

OpenAI API

ChromaDB

Chainlit

DE MO

