파이프라인 I/O (Input/Output) 정의

1. [입력] Audio Input

- **Input (User):** original.mp3 (또는 .m4a, .wav 등)
- **Output (to Preprocessing):**JSON
- {
- "filepath": "/temp/original.mp3",
- "mimetype": "audio/mp3"
- }

---

2. [모듈] Preprocessing (음성 전처리)

- **Input (from Step 1):** 원본 파일 경로
- **Output (to STT & Diarization):**JSON
    - 전처리가 완료된 **두 개의 파일 경로**를 생성하여 다음 단계로 전달합니다.
- {
- "stt_input_path": "/temp/vad_audio.wav", // VAD(음성구간감지)만 적용된 파일 (STT용)
- "diar_input_path": "/temp/nr_audio.wav" // VAD + 노이즈제거 적용된 파일 (화자분리용)
- }

---

3. [모듈 A] STT (Whisper)

- **Input (from Step 2):** stt_input_path
- **Output (to Tagger & Merge):**JSON
    - **단어(Word) 레벨**의 타임스탬프가 찍힌 텍스트 조각 리스트.
- [
- { "text": "네", "start": 3.6, "end": 3.8 },

- { "text": "민서씨", "start": 3.9, "end": 4.5 },
- { "text": "...", "start": null, "end": null }
-]

---

4. [모듈 B] Speaker Diarization (화자 분리)

- **Input (from Step 2):** diar_input_path
- **Output (to Tagger & Merge):**JSON
  - 각 화자(Label)의 고유한 **목소리 특징 벡터(Embedding)**와 해당 구간 리스트.
- {
- "turns": [
- { "speaker_label": "SPEAKER_00", "start": 0.4, "end": 3.5 },
- { "speaker_label": "SPEAKER_01", "start": 3.6, "end": 5.8 },
- { "speaker_label": "SPEAKER_00", "start": 5.9, "end": 8.1 }
-],
- "embeddings": {
- "SPEAKER_00": [0.12, -0.45, 0.88, ...], // (SPEAKER_00의 대표 임베딩)
- "SPEAKER_01": [-0.33, 0.76, -0.12, ...] // (SPEAKER_01의 대표 임베딩)
- }
- }

---

5. [핵심 모듈] Tagger & Merge Logic (태깅 및 병합)

- **Input 1 (from STT):** List[WordSegment] (3번 결과)
- **Input 2 (from Diarization):** DiarizationResult (4번 결과 - 임베딩 포함)

5a ~ 5c. 내부 처리 로직 (이름 감지, 목소리 비교)

1. **(이름 감지):** 3번(STT) 결과에서 "민서씨", "인서씨" 등 이름 후보를 추출합니다.
2. **(문맥/화자 매칭):** "민서씨"([3.9s~4.5s])라는 텍스트가 4번(Diarization)의 SPEAKER_01 구간에서 나왔음을 확인합니다. -> 대화 흐름상 SPEAKER_00이 "민서"일 것이라 추정합니다.
3. **(유사/동명 처리):** 만약 SPEAKER_00이 "인서"라고도 불렸다면, 4번의 embeddings 값을 가져와 두 목소리("민서"라 불린 SPEAKER_00과 "인서"라 불린 SPEAKER_00)의 **유사도를 내부적으로 비교**합니다.
4. **(내부 결론):** "유사도 95%. 동일인이다. 대표 제안 이름은 '민서'로 하자."

5d. UI로 전달 (사용자 검증 요청)

- **Output (to UI):**JSON
    - 복잡한 비교 로직(similarity_note)은 **제외**하고, **단순화된 제안**만 전달합니다.
- {
- "detected_names": ["민서", "인서"], // (참고용 전체 감지 이름)
- "suggested_mappings": [
- {
- "label": "SPEAKER_00",
- "suggested_name": "민서"
- // (백엔드가 '인서'와 동일인이라고 이미 판단 완료)
- },
- {
- "label": "SPEAKER_01",
- "suggested_name": null
- }
-]
- }

5e. UI로부터 입력 (사용자 확정)

- **Input (from UI):**JSON
    - 사용자가 제안을 검토하고 최종 수정한 매핑 정보.
- {
- "SPEAKER_00": "김민서", // (사용자가 '민서'를 '김민서'로 수정)
- "SPEAKER_01": "박철수" // (사용자가 'null'을 '박철수'로 입력)
- }

5f. 최종 결과물 (애플리케이션 전달)

- **Output (to Application):**JSON
    - 3번(STT), 4번(Diarization)의 결과를 5e(사용자 확정) 값 기준으로 **최종 병합**합니다.
- [
- {
- "speaker_name": "김민서",
- "start_time": 0.4,
- "end_time": 3.5,
- "text": "오늘 회의 안건은..."
- },
- {
- "speaker_name": "박철수",
- "start_time": 3.6,
- "end_time": 5.8,
- "text": "네, 민서씨 안건부터..."
- },
- {
- "speaker_name": "김민서",

- "start_time": 5.9,
- "end_time": 8.1,
- "text": "좋습니다. 그럼..."
- }
-]

---

6. [모듈] Application Router (응용)

- **Input (from Tagger & Merge):** List[FinalTranscript] (5f 결과)
- **Output (to User):**
  - **(요약):** 5f의 text를 모두 취합해 LLM에 전달 -> String (요약문)
  - **(RAG):** 5f의 각 항목을 Vector DB에 저장 (작업 수행)
  - **(자막):** 5f의 정보를 String (.srt 또는 .vtt 포맷)으로 변환