



## Ai 기능 명세서

2026. 2. 13.

팀 프로젝트 주소 : [https://github.com/DOT-SOY/AI\\_HealthCare\\_Agent](https://github.com/DOT-SOY/AI_HealthCare_Agent)

팀장 | 한정연: <https://github.com/DOT-SOY>

팀원 | 박태오: <https://github.com/teomichaelpark-glitch>

팀원 | 김민식: <https://github.com/minsik321>

팀원 | 오인준: <https://github.com/01nJun>

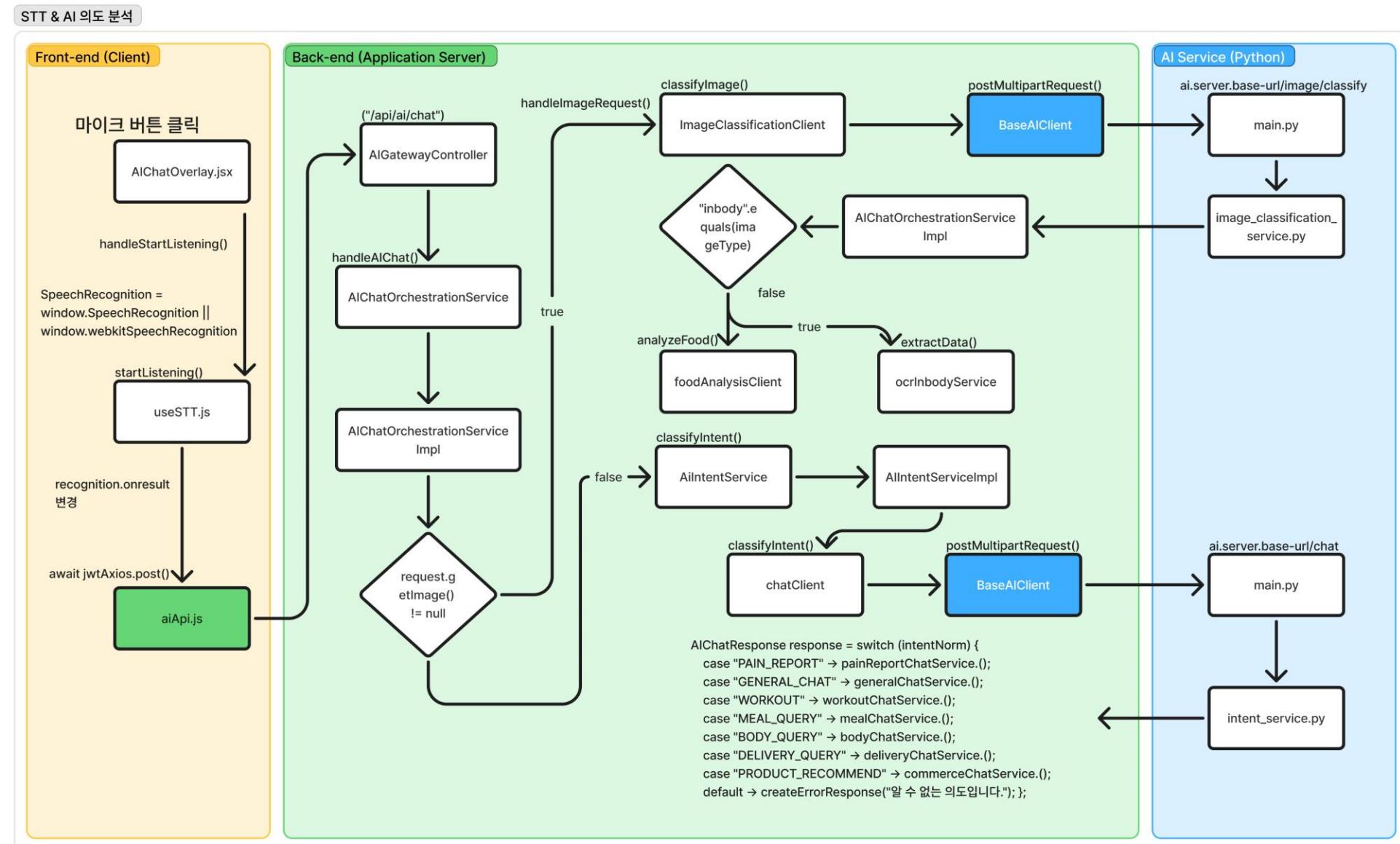
팀원 | 박건영: <https://github.com/keonyeong4550>

팀 노션 주소: <https://www.notion.so/2cce455ae480803d8925c84065cf4f0b>

## - 목 차 -

1. <a href="#">AI 음성 인식 (STT) → 의도인식</a>	p. 2
2. <a href="#">AI 의도 분류에 따른 logic 호출</a>	p. 5
3. <a href="#">AI 이미지 분류 (OCR 이미지/ 음식 사진 여부)</a>	p. 8
4. <a href="#">AI 음식 이미지 분석 → 칼로리, 음식명 등 반환</a>	p.11
5. <a href="#">AI 식단 생성 → 하루권장 kcal에 맞춰 식단 제공</a>	p.14
6. <a href="#">AI 운동 자세 인식 → 자세에 따른 운동 카운트</a>	p.17
7. <a href="#">AI 운동 피드백에 따른 실시간 알림</a>	p.20
8. <a href="#">AI 음성 합성 (TTS) → 실시간 음성 피드백</a>	p.23
9. <a href="#">AI 통증 조언에 따른 피드백</a>	p.26
10. <a href="#">AI 루틴 추천 → 하루 운동 스케줄 계획</a>	p.29
11. <a href="#">AI 상품 추천 - 채팅 입력에 따른 상품 추천</a>	p.32
12. <a href="#">OCR AI (인바디 이미지 인식)</a>	p.35

# 1. AI 음성 인식(STT) → 의도인식 **플로우 차트**



# 1. AI 음성 인식(STT) → 의도인식 구현



The screenshot shows the ALGORHYGYM app's main dashboard. At the top left is the logo and a 'Home' button. The top right displays a welcome message 'Welcome, 이사도님' and a profile icon. The central header says 'WELCOME BACK, 이사도'. Below the header are two main sections: a calendar and a graph. The calendar for February 2026 shows the 12th as the active day. To the right is a line graph titled '월별 주별' (Monthly Weekly) showing a single sharp peak at the 12th. Below these sections are 'TODAY'S FOCUS' and '운동 목록' (Exercise List) sections, both currently empty. A red vertical bar is on the far right.

```
useEffect(() => {
  if (!isListening && transcript && transcript.trim() && !loading) {
    const text = transcript.trim();
    if (lastSentTranscriptRef.current === text) return undefined;
  }

  const timer = setTimeout(async () => {
    lastSentTranscriptRef.current = text;
    setInputText('');
    await sendAIMessage(text);
  }, 300);

  return () => clearTimeout(timer);
}

return undefined;
}, [islistening, transcript, loading, sendAIMessage]);
```



The screenshot shows a mobile application interface. At the top, there is a code editor window displaying a portion of a React component. The component includes a button with a click handler that toggles listening status and a title for a text input field. Below the code editor is a dark-themed message list. The first message is from the user, stating they don't have a routine yet and asking if they want to add one. It includes a green button labeled "루틴에 운동 추가해줘". The second message is from the application, suggesting adding bench presses. The third message is from the user, asking if there's anything else. The bottom of the screen features a message input field with a microphone icon and a red square highlight, and a green send button labeled "전송".

#### ◆ AI 음성 인식 (STT - Speech-to-Text)

- **브라우저 기반 실시간 음성 입력음성 인식**: Web Speech API로 한국어 음성을 실시간 텍스트로 변환
  - **AI 코치 연동**: 음성으로 입력된 텍스트를 AI 코치 채팅으로 자동 전달하여 질문 처리
  - **연속 인식**: 음성 입력 종료 시 자동으로 AI 메시지 전송

## 실시간 STT 음성 요청



# 1. AI 음성 인식(STT) → 의도인식 가능 설명

## 1. 역할

- 사용자 음성을 텍스트로 변환해 AI 채팅 입력에 사용합니다. 마이크 버튼 클릭으로 음성입력을 활성화합니다.

## 2. 사용 기술

- Web Speech API (브라우저 내장)

## 3. 사용 시나리오

- AI 채팅 음성 입력: 마이크 버튼 클릭 → 음성 인식 시작 → 인식 결과를 입력 필드에 반영 → 300ms 디바운스 후 자동 전송

## 4. STT 토글

- 별도 토글 없음. 마이크 버튼이 녹음 시작/종료를 동시에 제어

## 5. 정리

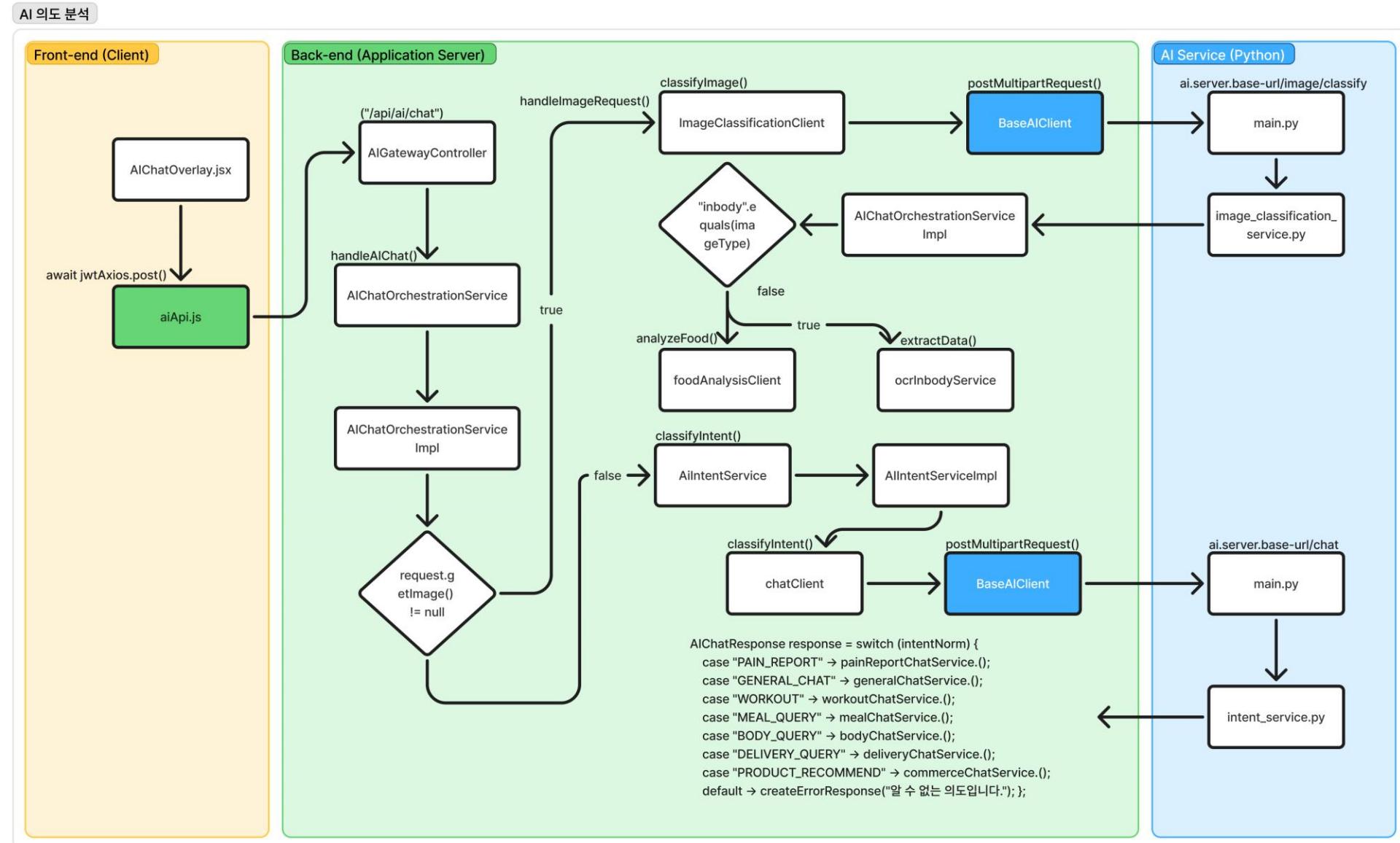
- continuous: false, interimResults: true, lang: 'ko-KR' 사용
- 지원하지 않는 브라우저에서는 콘솔 경고 출력

## 6. 설계 포인트

- 별도 서버/API 없이 브라우저 내장 API만 사용
- 한국어 채팅에 맞춰 ko-KR 사용
- interimResults로 말하는 동안 실시간 텍스트 표시

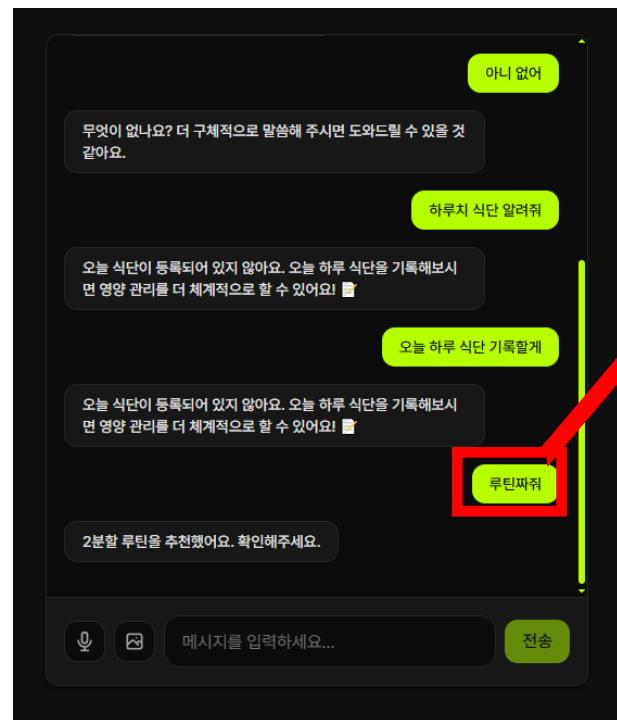


## 2. AI 의도 분류에 따른 logic 호출 플로우 차트





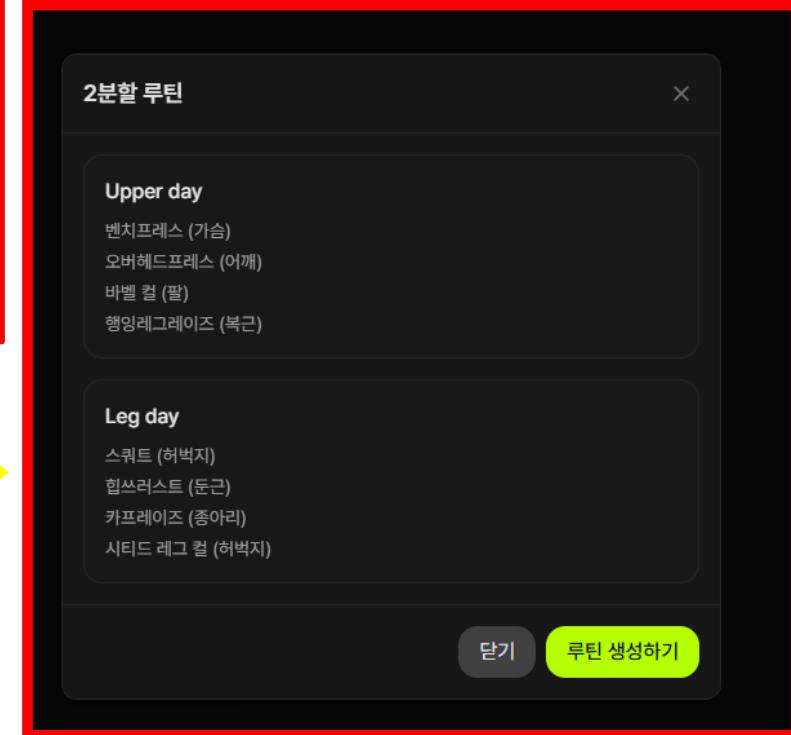
## 2. AI 의도 분류에 따른 logic 호출 구현



```
    }

    // 7. 의도에 따라 적절한 Service 호출
    AIChatResponse response = switch (intentNorm) {
        case "PAIN_REPORT" -> painReportChatService.handlePainReport(classification);
        case "GENERAL_CHAT" -> generalChatService.handleGeneralChat(classification);
        case "WORKOUT" -> workoutChatService.handleWorkout(classification);
        case "MEAL_QUERY" -> mealChatService.handleMeal(classification, request);
        case "BODY_QUERY" -> bodyChatService.handleBodyQuery(classification);
        case "DELIVERY_QUERY" -> deliveryChatService.handleDelivery(classification);
        case "PRODUCT_RECOMMEND" -> commerceChatService.handleCommerceRecommend(classification, request);
        default -> createErrorResponse(errorMessage: "알 수 없는 의도입니다.");
    };

    return response;
}
```



### ◆ AI 의도 분류 (Intent Classification) 및 조회(Search)

- 정확한 의도 파악을 위해 다단계 분류 프로세스 적용의도 분류: OpenAI GPT-4o-mini로 사용자 입력을 WORKOUT, MEAL\_QUERY, PAIN\_REPORT, PRODUCT\_RECOMMEND, GENERAL\_CHAT 등으로 분류
- 엔티티 추출: 발화에서 부위, 키워드, 예산, 카테고리 등 구조화된 정보 추출
- 라우팅: 분류 결과에 따라 적절한 처리 경로로 자동 라우팅 (운동 분석, 식단 처리, 통증 조언, 상품 추천 등)



## 2. AI 의도 분류에 따른 logic 호출 가능 설명

### 1. 역할

- 사용자 문장을 분석해 **의도(intent)**와 **액션(action)**을 구조화된 JSON으로 반환합니다. 이 결과로 후속 라우팅(운동/식단/통증/커머스 등)을 결정합니다.

### 2. 동작 방식

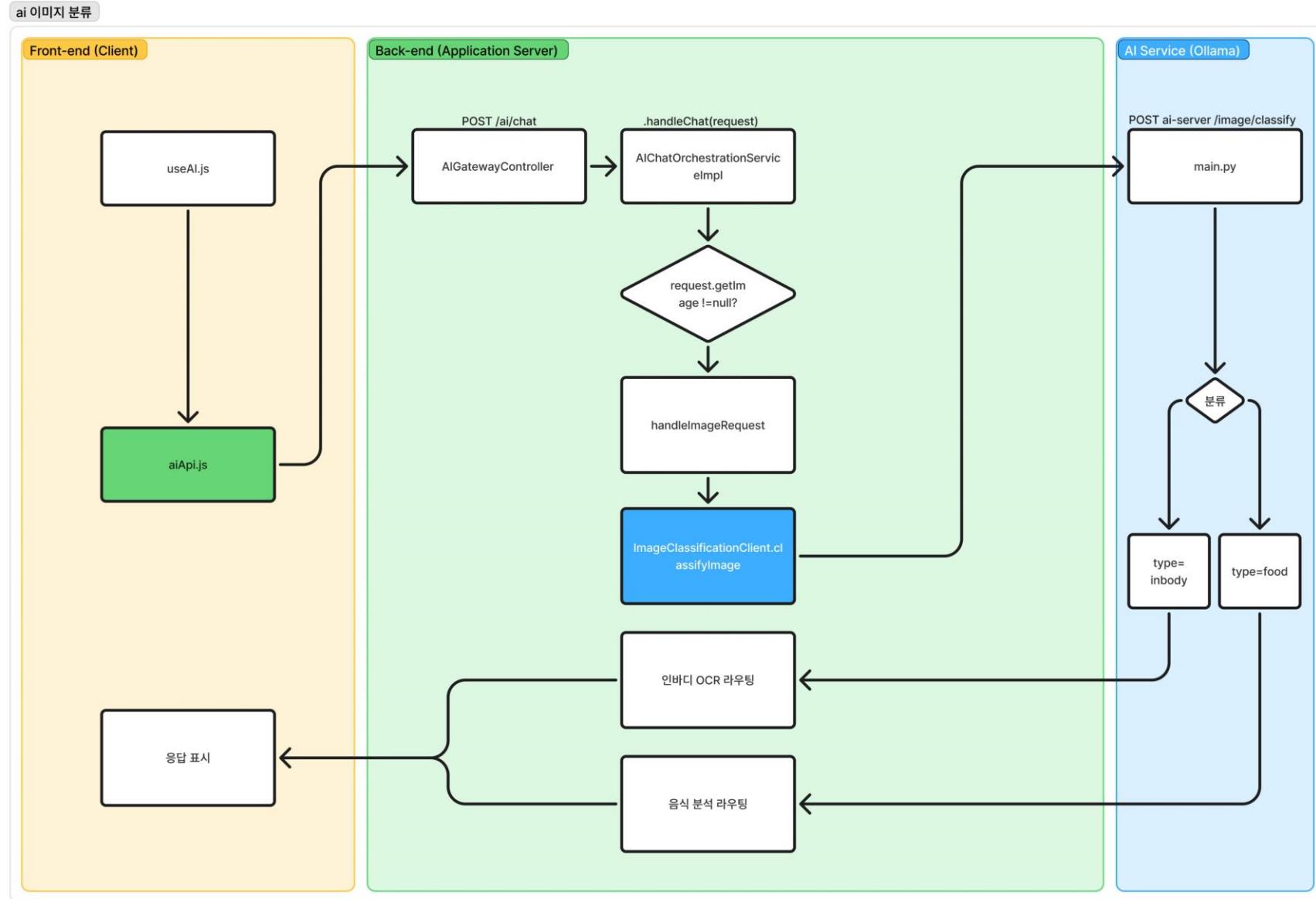
- 프롬프트에 **분류 규칙과 엔티티 규칙이 정의되어 있습니다.**
- WORKOUT**: "2분할 루틴 추천해줘" → split\_type=2, RECOMMEND
- PAIN\_REPORT**: "허리 아파요" → body\_part, REPORT
- MEAL\_QUERY**: "3일치 식단 짜줘" → periodDays, RECOMMEND
- BODY\_QUERY**: "체지방률 어제치" → date, body\_metric
- PRODUCT\_RECOMMEND**: "프로틴 추천해줘" → product\_name, RECOMMEND
- GENERAL\_CHAT**: 규칙에 해당하지 않으면 일상 대화로 처리
- 날짜는 {current\_date}를 치환해 "오늘", "어제" 등을 실제 날짜로 변환합니다.

### 3. 설계 포인트

- 규칙이 명확한 경우 **규칙 기반**으로 의도 결정
- 경계 케이스는 **예시(few-shot)**로 구분
- "근력운동 시작할건데 뭘 사야 할지 모르겠어" → GENERAL\_CHAT 등 모호한 문장 처리



### 3. AI 이미지 분류(OCR 이미지/ 음식 사진 여부) 플로우 차트





### 3. AI 이미지 분류(OCR 이미지/ 음식 사진 여부) 구현



#### ◆ AI 이미지 분류 (Image Classification)

- CLIP 기반 이미지 타입 자동 분류 이미지 매칭: CLIP 모델로 업로드된 이미지를 인바디 사진/음식 사진분류
- 유사도 계산: 인바디/음식 관련 텍스트 프롬프트와 이미지 간 코사인 유사도 계산하여 분류
- 자동 라우팅: 분류 결과에 따라 적절한 처리 앤드포인트로 자동 라우팅 (인바디 분석 또는 음식 분석)



### 3. AI 이미지 분류(OCR 이미지/ 음식 사진 여부) **기능 설명**

#### 1. 역할

- 업로드된 이미지가 **인바디 기기 화면인지 음식 사진인지** 분류합니다. 이후 인바디 OCR 또는 음식 분석 툴로 우로 분기합니다.

#### 2. 사용 기술

- **CLIP** (openai/clip-vit-base-patch32)
- **PyTorch, Transformers**

#### 3. 동작 방식

- 인바디: "인바디 측정기 화면", "체성분 분석기 디스플레이" 등 8개 문장
- 음식: "음식 사진", "식사 사진", "food photo" 등 10개 문장

#### 4. 점수 계산: 최대 유사도 60% + 평균 유사도 40%로 가중 평균

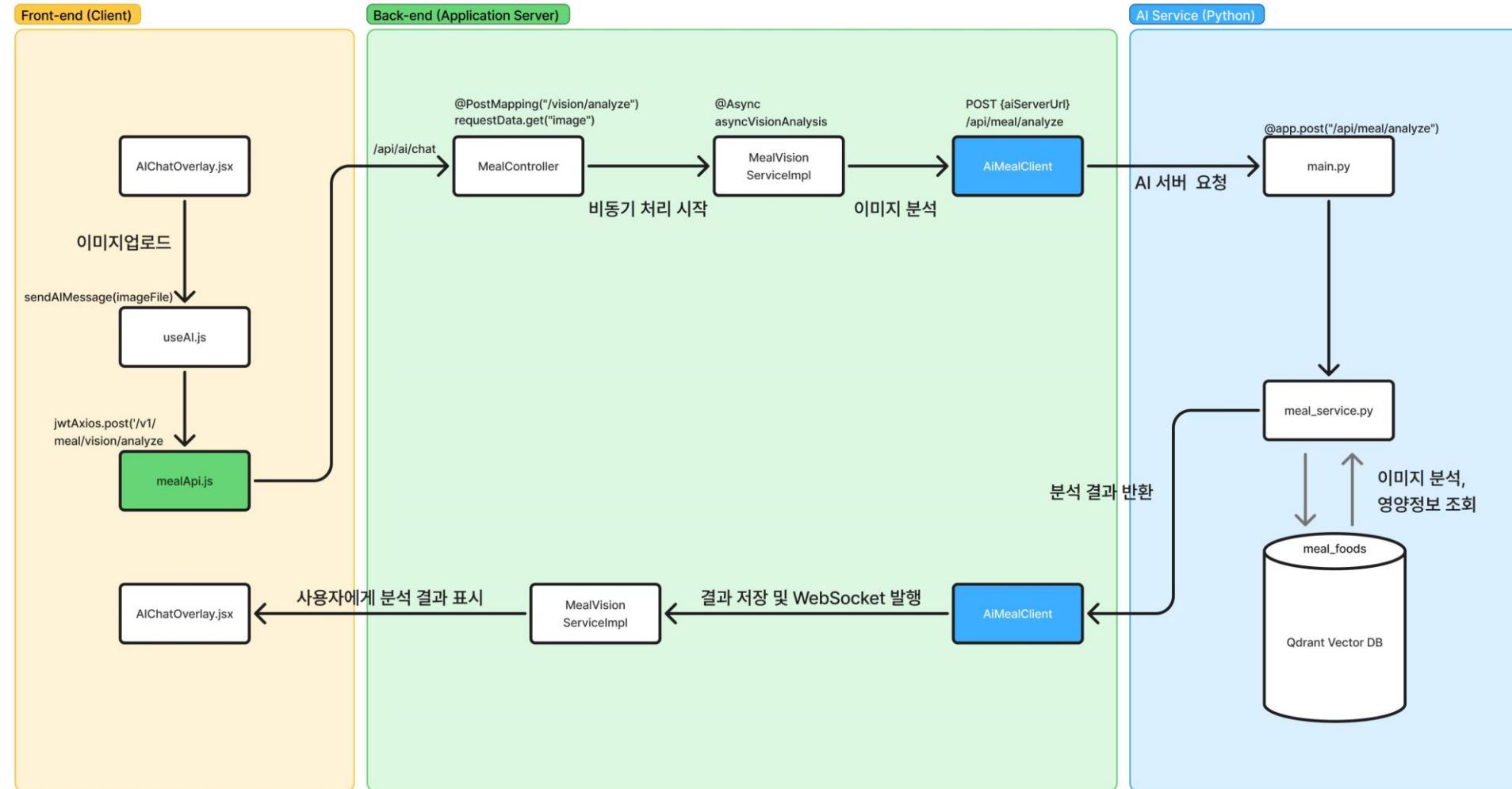
#### 5. 분류 규칙

- 인바디 점수  $\geq 0.3 \rightarrow$  인바디
- 인바디·음식 점수 모두  $< 0.2 \rightarrow$  음식(불확실 시 기본값)

#### 6. 설계 포인트

- 별도 API 호출 없이 로컬 CLIP으로 실시간 분류
- 인바디/음식 모두 다국어 텍스트로 판별력 향상

# 4. AI 음식 이미지 분석 → 칼로리, 음식명 등 반환 플로우 차트





# 4. AI 음식 이미지 분석 → 칼로리, 음식명 등 반환 구현



## ◆ AI 음식 이미지 분석 (Food Image Analysis)

- **Vision AI 기반 영양정보 자동 추출 이미지 분석:** Gemini 2.5 Pro Vision으로 음식 사진에서 LLM으로 음식명 분류 후 RAG 연결, 칼로리, 탄수화물, 단백질, 지방 자동 추출
- **영양정보 보정:** Qdrant 벡터 DB에서 음식명 정규화 및 영양정보 매칭으로 정확도 향상
- **비동기 처리:** 이미지 업로드 시 WebSocket으로 분석 진행률 실시간 전달, 완료 후 결과 반환



# 4. AI 음식 이미지 분석 → 칼로리, 음식명 등 반환 가능 설명

## 1. 역할

- 음식 사진을 보고 **음식명 후보**와 RAG 검색용 키워드만 추출합니다. 영양 수치는 LLM이 추론하지 않고, 이후 RAG·DB에서 조회합니다.

## 2. 사용 모델

- Gemini Vision** (gemini-2.5-pro)

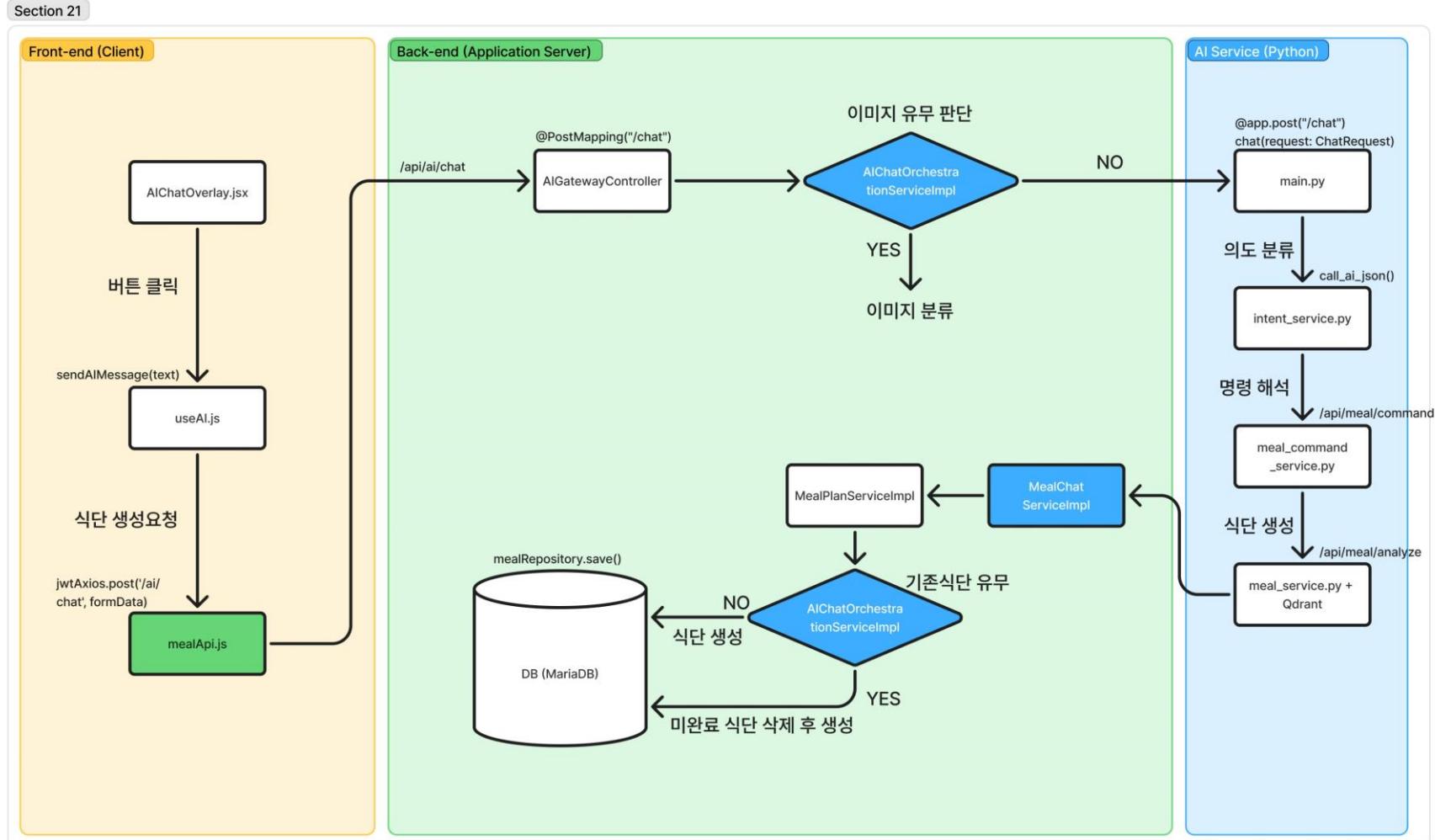
## 3. 프롬프트 규칙

- 영양 추론 금지**: 칼로리·탄수화물·단백질·지방 계산 금지
- 음식명 후보**: food\_candidates 배열 (name, confidence)
- RAG 쿼리**: rag\_queries 배열 (한글명·영문명·동의어)
- 확신이 낮으면 needs\_clarification, clarifying\_question 사용

## 4. 설계 포인트

- 영양은 RAG·DB 기반으로 일관성 확보
- LLM은 "무엇이 보이는지"와 검색 키워드에만 집중

# 5. AI 식단 생성 → 하루권장 kcal에 맞춰 식단 제공 플로우 차트





# 5. AI 식단 생성 → 하루권장 kcal에 맞춰 식단 제공 구현

def resolve\_meal\_command(text: str, context: Optional[Dict[str, Any]] = None) -> Dict[str, Any]:  
 ...  
 LLM + 사용자 질문으로 meal command를 생성합니다.  
 실패 시 ASK\_CLARIFY를 인식하게 fallback 합니다.  
 ...  
 try:  
 # Fast-path: pending 기반 질문은 단순으로 쿠션 처리(LLM 블록킹/기능적/빠른 응답)  
 fast = \_fast\_resolve\_with\_context(text, context)  
 if fast:  
 cmd = MealCommand.model\_validate(fast)  
 cmd = normalize\_command(cmd)  
 return cmd.model\_dump()  
 raw = call\_meal\_ai\_json(  
 system\_prompt=SYSTEM\_PROMPT,  
 user\_prompt=\_build\_user\_prompt\_with\_context(text, context),  
 temperature=0.0,  
 )  
 cmd = MealCommand.model\_validate(raw)  
 cmd = normalize\_command(cmd)  
 return cmd.model\_dump()

음식명: 비빔밥  
칼로리: 500 kcal  
탄수화물: 80 g  
단백질: 15 g  
지방: 12 g

이 메뉴를 오늘 식사에 반영할까요?  
예: "점심으로 바꿔줘", "추가로 기록해줘", "취소"

점심으로 바꿔줘

비빔밥으로 점심을 대체했어요.

식단생성해줘

이틀

메시지를 입력하세요... 전송

TODAY'S MEAL PLAN

일일 목표 달성을  
FAIL 21% 80/369g 탄수화물  
FAIL 12% 15/124g 단백질  
FAIL 16% 12/73g 지방  
FAIL 19% 500/2629kcal 칼로리

아침 원료 생략  
1000 kcal  
• 안심 참스테이크 (고기+야채 소스)  
• 볶음밥 (마늘 후레이크)  
• 피클 (소량)  
• 오징어 젓갈 (소량)  
• 누룽지 (1그릇)

점심 원료 생략  
1320 kcal  
• 비빔밥 (완료)  
• 닭갈비 (고구마, 양배추 포함)  
• 볶음밥 (마무리)  
• 동치미 국물

저녁 원료 생략  
810 kcal  
• 짬뽕밥 (당면 포함, 밥 1공기)  
• 단무지  
• 흰수육 (깻잎, 5조각)

AI 식단 분석

## ◆ AI 식단 생성 (Meal Plan Generation)

- 사용자 프로필 기반 맞춤 식단 자동 생성프로필 분석: Gemini 2.5 Pro가 사용자의 키, 몸무게, 목표(다이어트/유지/벌크업)를 분석하여 목표 칼로리 및 영양소 배분 계산
- 식단 생성: 일/주/월 단위 식단 자동 생성, 식품 DB에서 목표에 맞는 음식 조합 추천
- 재분배 기능: 사용자 요청 시 기존 식단을 목표에 맞게 재조정

# 5. AI 식단 생성 → 하루권장 kcal에 맞춰 식단 제공 가능 설명



## 1. 역할

- 프로필·목표(칼로리·탄/단/지)·기간을 바탕으로 추천 식단을 생성하거나, 현재 식단을 재분배합니다.

## 2. 하이브리드 전략

- **목표치**: Gemini로 추론 (실패 시 공식 기반 fallback)
- **메뉴·영양**: Qdrant meal\_templates, meal\_foods payload 기반
- 템플릿이 없으면 LLM 생성으로 fallback (서비스 중단 방지)

## 3. RAG 구조

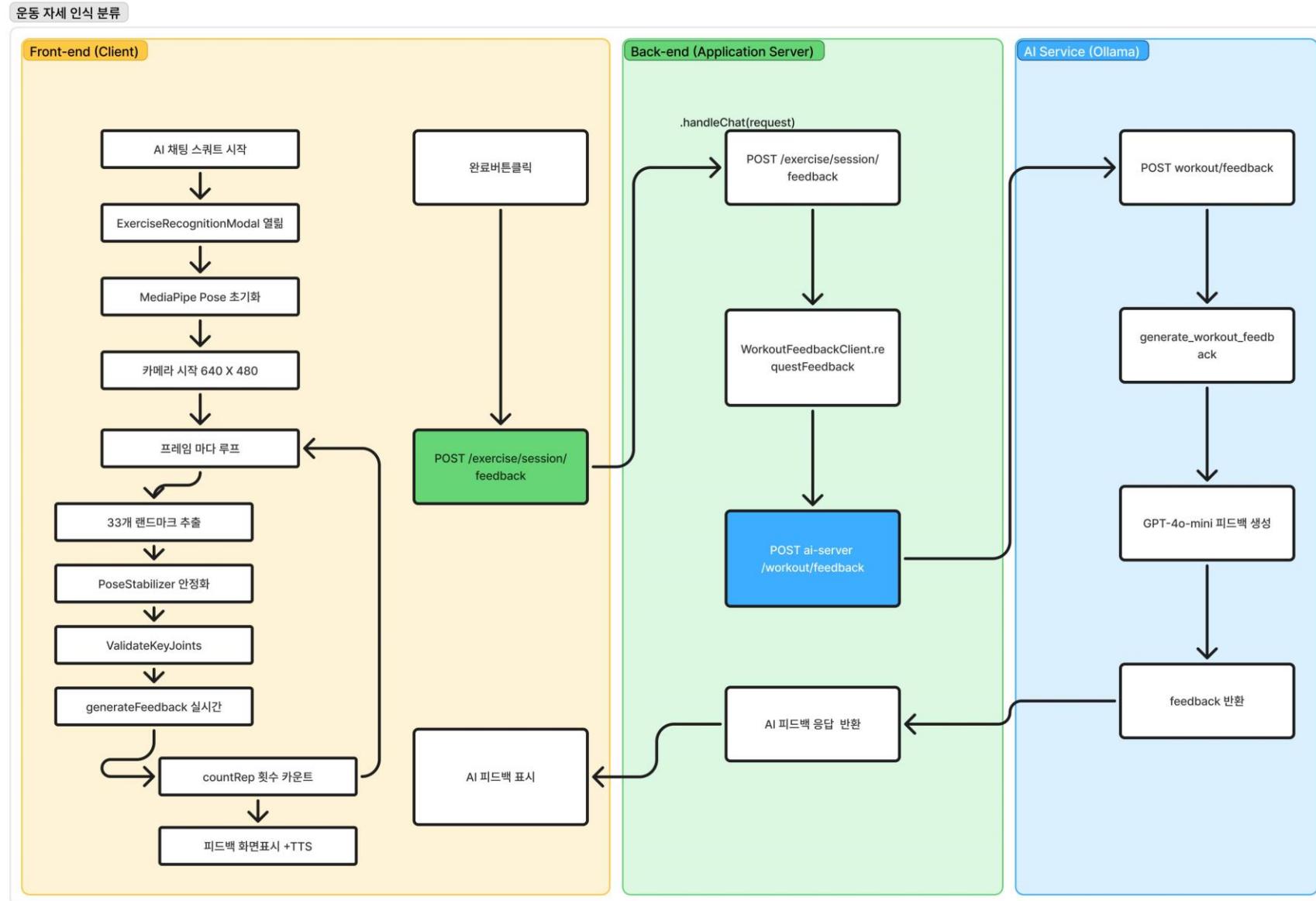
- **Gemini text-embedding-004** (768차원)
- **Qdrant**: meal\_foods, meal\_templates 컬렉션
- exact match 우선 → 문자열 유사도(fuzzy) fallback

## 4. 설계 포인트

- 반복·정형 영역은 **템플릿·DB**로 처리
- LLM은 목표치 추론 등 필수 구간에만 사용
- 비용·재현성·일관성 확보

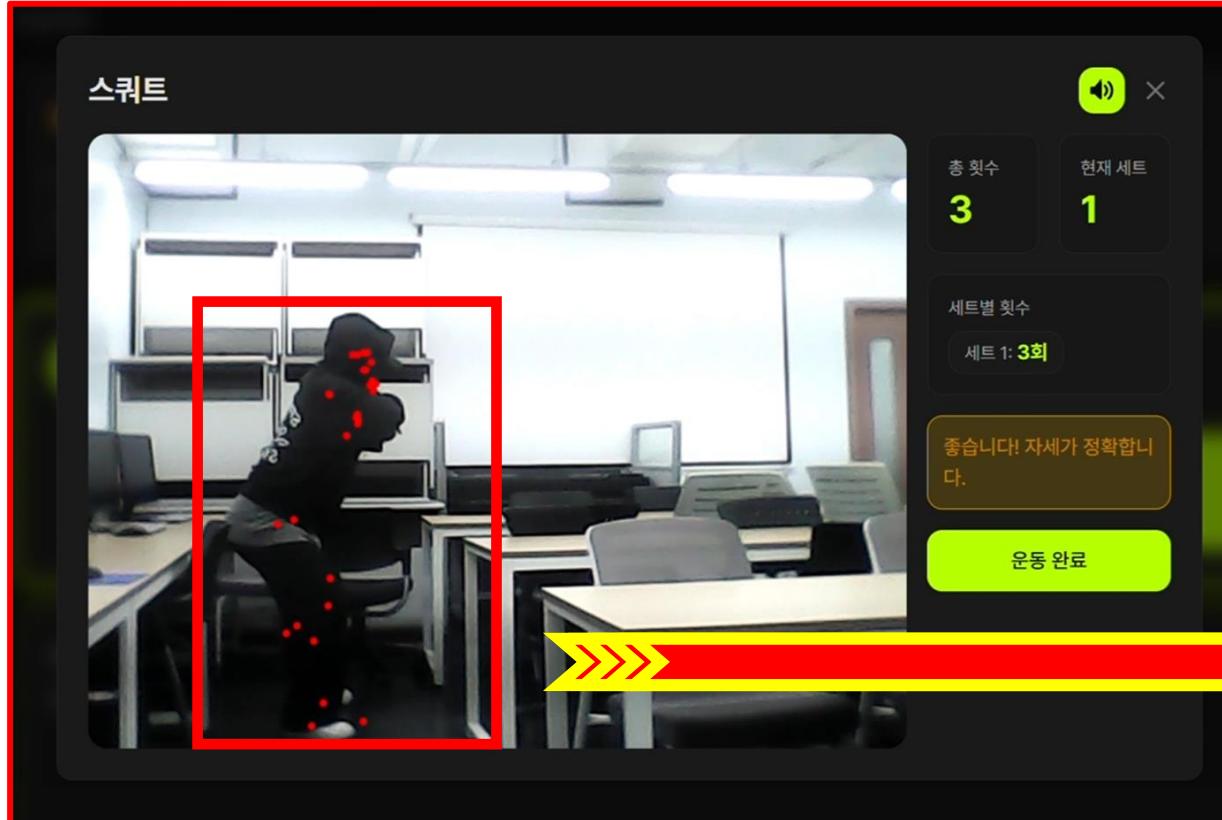


# 6. AI 운동 자세 인식 → 자세에 따른 운동 카운트 플로우 차트





# 6. AI 운동 자세 인식 → 자세에 따른 운동 카운트 구현



## ◆ AI 운동 자세 인식 (Exercise Pose Recognition)

- **실시간 관절 추적 기반 운동 분석**: MediaPipe Pose로 실시간 33개 관절 랜드마크 추출 (어깨, 팔꿈치, 손목, 무릎, 발목 등)
- **운동별 기준값 비교**: 운동별 정의된 기준값과 비교하여 자세 오류 비율 계산
- **횟수 카운팅**: 자세 변화 패턴 분석으로 운동 횟수 자동 카운팅

```
export default function ExerciseRecognitionModal(){  
  // MediaPipe 초기화  
  useEffect(() => {  
    if (!isOpen) return;  
  
    const pose = new Pose({  
      locateFile: (file) => {  
        return `https://cdn.jsdelivr.net/npm/@mediapipe/pose/${file}`;  
      }  
    });  
  
    pose.setOptions({  
      modelComplexity: 1,  
      smoothLandmarks: true,  
      enableSegmentation: false,  
      smoothSegmentation: false,  
      minDetectionConfidence: 0.6,  
      minTrackingConfidence: 0.6  
    });  
  
    pose.onResults((results) => {  
      if (!videoRef.current || !canvasRef.current) return;  
  
      const canvasCtx = canvasRef.current.getContext('2d');  
      canvasCtx.save();  
      canvasCtx.clearRect(0, 0, canvasRef.current.width, canvasRef.current.height);  
      canvasCtx.drawImage(results.image, 0, 0, canvasRef.current.width, canvasRef.current.height);  
  
      // 트레킹 안정화: null 프레임 발생 시 마지막 유효 랜드마크 유지  
      let stabilizedLandmarks = null;  
      if (poseStabilizerRef.current) {  
        stabilizedLandmarks = poseStabilizerRef.current.stabilizeLandmarks(results.poseLandmarks);  
      } else {  
        stabilizedLandmarks = results.poseLandmarks;  
      }  
    });  
  })  
}
```



# 6. AI 운동 자세 인식 → 자세에 따른 운동 카운트 기능 설명

## 1. 역할

- 카메라로 사용자 동작을 실시간 인식해 운동 횟수(count)를 세고, 자세 피드백을 실시간으로 제공합니다. 스쿼트·턱걸이·윗몸일으키기를 지원합니다.

## 2. 사용 기술

- **MediaPipe Pose** (Google ML 포즈 추정)
- **exerciseRules.json** (운동별 규칙 정의)
- **PoseStabilizer** (트래킹 끊김 시 최대 5프레임 보간)

## 3. 사용 시나리오

- 카메라 ON → MediaPipe로 33개 랜드마크 추출 → validateKeyJoints()로 핵심 관절 검증 → countRep()로 횟수 카운트 → generateFeedback()로 자세 피드백 생성 → 화면 표시 + TTS 음성 출력

## 4. 정리

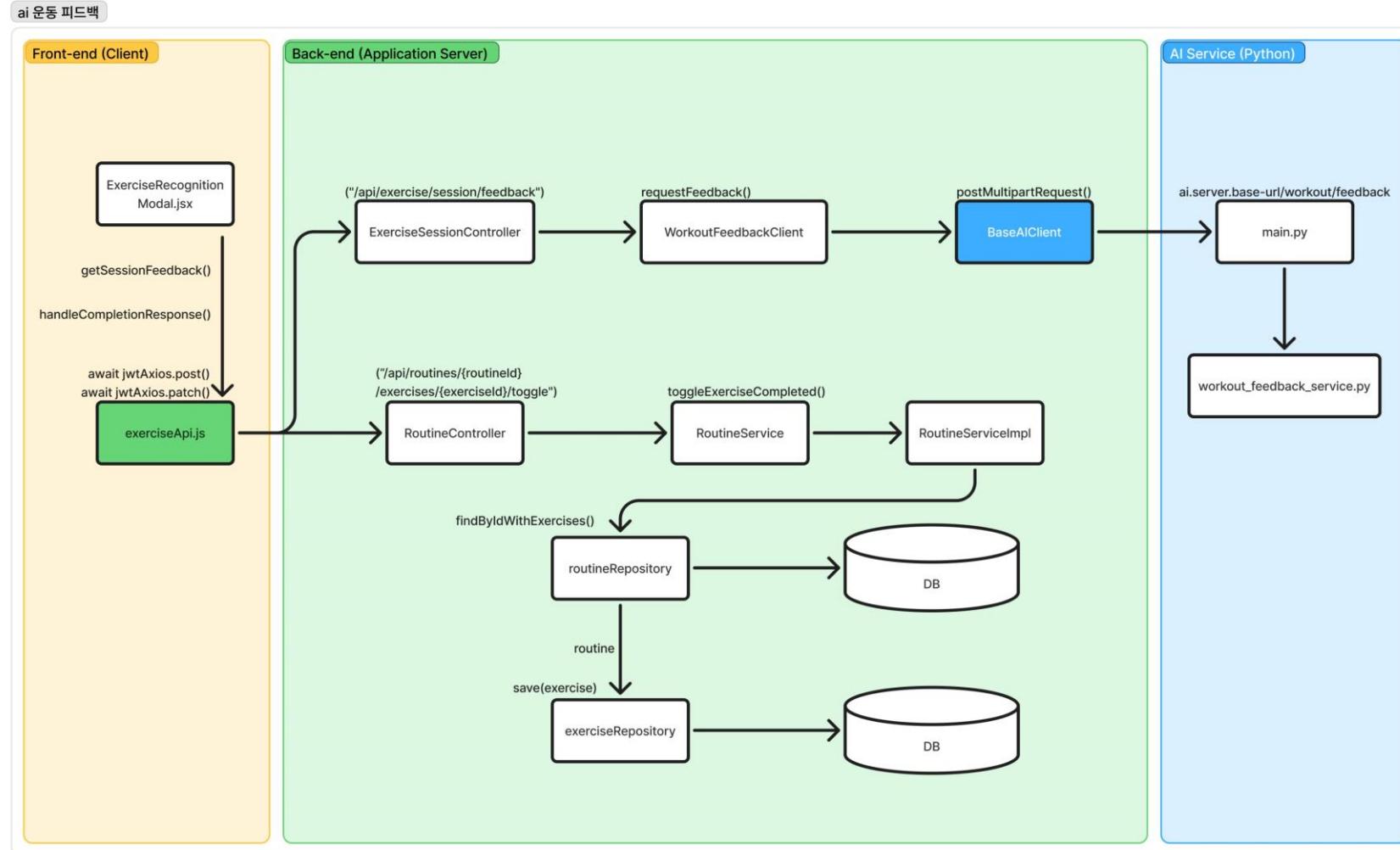
- **자세 인식:** MediaPipe Pose (ML 모델)
- **세션 피드백:** 운동 완료 후 별도 AI 서버 호출 (운동 피드백 항목 참고)

## 5. 설계 포인트

- 운동 중 시선을 화면에 두고 TTS로 피드백을 들을 수 있도록 구성
- 규칙 기반 피드백으로 자연 최소화



# 7. AI 운동 피드백에 따른 실시간 알림 플로우 차트





# 7. AI 운동 피드백에 따른 실시간 알림 구현

스쿼트

총 횟수 5      현재 세트 1

세트별 횟수  
세트 1: 5회

좋습니다! 자세가 정확합니다.

운동 완료

```
/*
 * 실시간 피드백 생성
 */
export function generateFeedback(landmarks, exerciseName) {
  if (!landmarks || landmarks.length < 33) {
    return null;
  }

  const rules = exerciseRules[exerciseName];
  if (!rules || !rules.feedback) {
    return null;
  }

  const feedbacks = [];

  if (exerciseName === '스쿼트') {
    const leftHip = landmarks[LANDMARKS.LEFT_HIP];
    const rightHip = landmarks[LANDMARKS.RIGHT_HIP];
    const leftKnee = landmarks[LANDMARKS.LEFT_KNEE];
    const rightKnee = landmarks[LANDMARKS.RIGHT_KNEE];
    const leftAnkle = landmarks[LANDMARKS.LEFT_ANKLE];
    const rightAnkle = landmarks[LANDMARKS.RIGHT_ANKLE];
    const leftShoulder = landmarks[LANDMARKS.LEFT_SHOULDER];
    const rightShoulder = landmarks[LANDMARKS.RIGHT_SHOULDER];

    // 필요한 관절이 모두 유효한지 확인
    if (!leftHip || !rightHip || !leftKnee || !rightKnee ||
        !leftAnkle || !rightAnkle || !leftShoulder || !rightShoulder) {
      return null;
    }

    // 허리 숙여짐 체크
    const backAngle = calculateAngle(leftShoulder, leftHip, leftKnee);
    if (backAngle > rules.feedback.back_leaning.threshold) {
      feedbacks.push(`허리가 너무 숙여져 있습니다.`);
    }
  }
}
```

## ◆ AI 운동 피드백 (Exercise Feedback Generation)

- **분석 결과 기반 맞춤 피드백 제공**: MediaPipe 분석 결과(운동 종류, 총 횟수, 운동 시간, 자세 오류 비율)를 OpenAI GPT-4o-mini에 전달
- **피드백 생성**: 분석 데이터를 바탕으로 자세 개선 방안 및 다음 운동 가이드 생성
- **TTS 음성 안내**: 분석 결과를 Web Speech API로 음성 변환하여 실시간 피드백 제공



# 7. AI 운동 피드백에 따른 실시간 알림 기능 설명

## 1. 역할

- 운동 세션 종료 시점에, 그동안 쌓인 세션 데이터를 바탕으로 AI가 최종 피드백 문구를 생성합니다.

## 2. 사용 시나리오

- 운동 완료 버튼 클릭 → feedbackHistory에서 main\_issue, bad\_posture\_ratio 계산 → POST /api/exercise/session/feedback → AI 서버 POST /workout/feedback → 피드백 문구 생성 → 모달 표시

## 3. 세션 데이터

- exercise\_type**: 운동명 (예: 스쿼트)
- total\_reps**: 총 횟수
- main\_issue**: 가장 많이 발생한 부정 피드백
- bad\_posture\_ratio**: (부정 피드백 수 / 전체 피드백 수) × 100

## 4. 정리

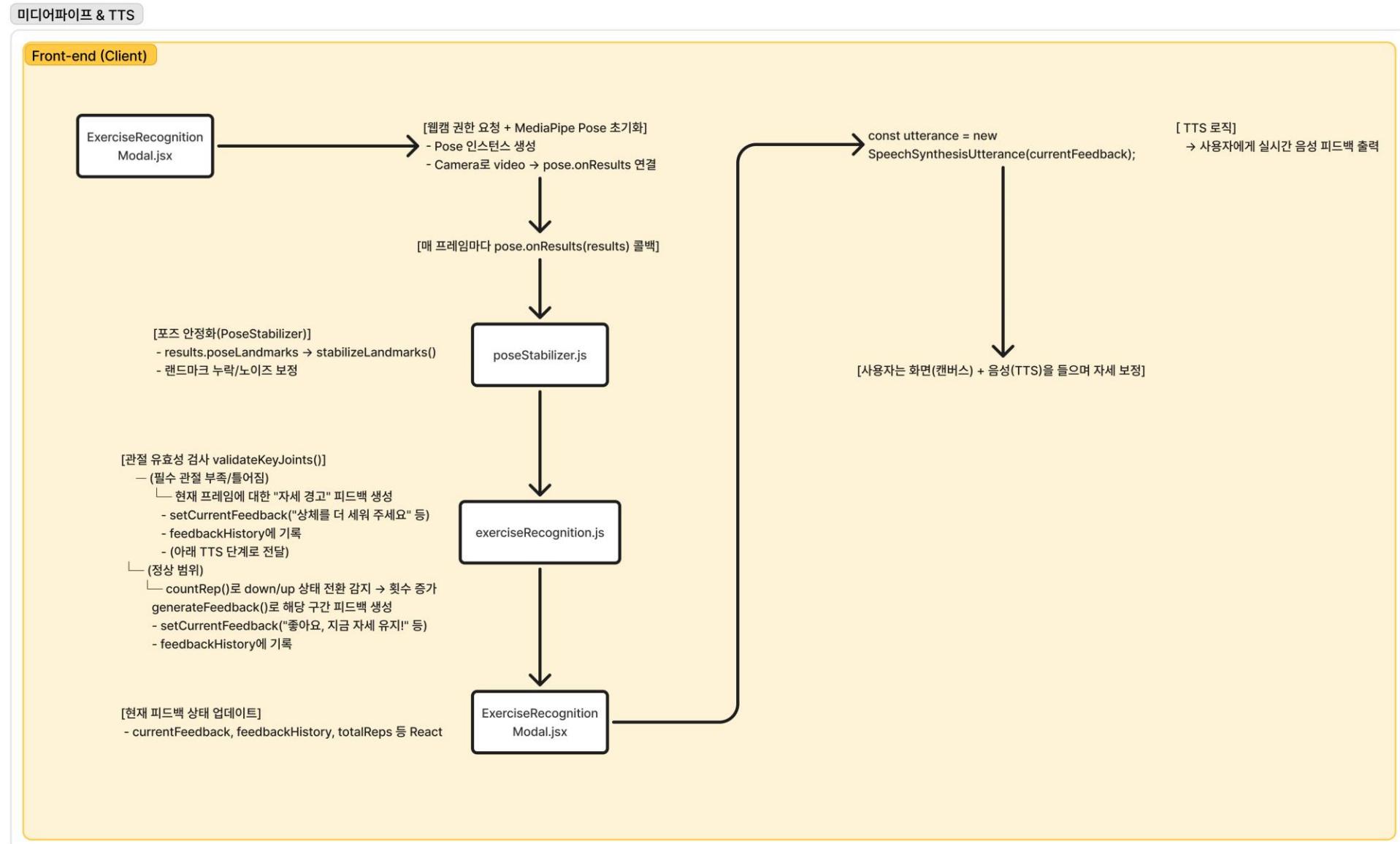
- 세션 종료 시점에만 호출 (실시간 피드백과 구분)
- 실시간 피드백은 규칙 기반, 세션 피드백은 AI 기반

## 5. 설계 포인트

- 최대 2문장 구조**: 첫 문장은 요약·개선점, 두 번째 문장은 완료 여부 질문 (STT 대응)
- 과도한 칭찬·단정적 평가 금지, 중립적 톤 유지



# 8. AI 음성 합성(TTS) → 실시간 음성 피드백 플로우 차트





## 8. AI 음성 합성(TTS) → 실시간 음성 피드백 구현

스쿼트



총 횟수 2      현재 세트 1  
세트별 횟수  
세트 1: 2회

허리가 너무 숙여졌습니다.  
등을 곧게 펴주세요.

운동 완료

실시간 TTS 재생 화면

```
// 실시간 피드백 TTS 재생
useEffect(() => {
  if (!currentFeedback || !ttsEnabled || isCompleted) return;

  // TTS 재생
  if ('speechSynthesis' in window) {
    // 이전 음성이 있으면 취소
    window.speechSynthesis.cancel();

    const utterance = new SpeechSynthesisUtterance(currentFeedback);
    utterance.lang = 'ko-KR';
    utterance.rate = 1.0; // 읽기 속도
    utterance.pitch = 1.0; // 음성 높이
    utterance.volume = 0.8; // 볼륨

    utterance.onend = () => {
      // 재생 완료 후 정리
    };

    utterance.onerror = (error) => {
      console.error('TTS 재생 오류:', error);
    };
  }

  window.speechSynthesis.speak(utterance);
}
```

### ◆ AI 음성 합성 (TTS - Text-to-Speech)

- **실시간 음성 피드백 텍스트 음성 변환:** Web Speech API로 운동 분석 결과 및 자세 가이드를 음성으로 변환
- **실시간 안내:** 운동 중 시선 분리 없이 음성으로 피드백 제공
- **가이드 음성:** 자세가 화면 밖으로 벗어날 때 "자세를 화면 중앙에 맞춰주세요" 음성 안내



# 8. AI 음성 합성(TTS) → 실시간 음성 피드백 가능 설명

## 1. 역할

- 텍스트를 음성으로 변환해 재생합니다. 운동 자세 인식 모달에서 실시간 피드백을 음성으로 전달합니다.

## 2. 사용 기술

- Web Speech API (브라우저 내장)

## 3. 사용 시나리오

- 실시간 자세 피드백: MediaPipe로 자세 분석 → generateFeedback() 반환 문자열 → TTS로 읽어줌

## 4. TTS 토글:

- “음성 피드백 끄기/켜기” 버튼으로 ttsEnabled 상태 제어
- 비활성화 시 speechSynthesis.cancel()로 재생 중지

## 5. 정리:

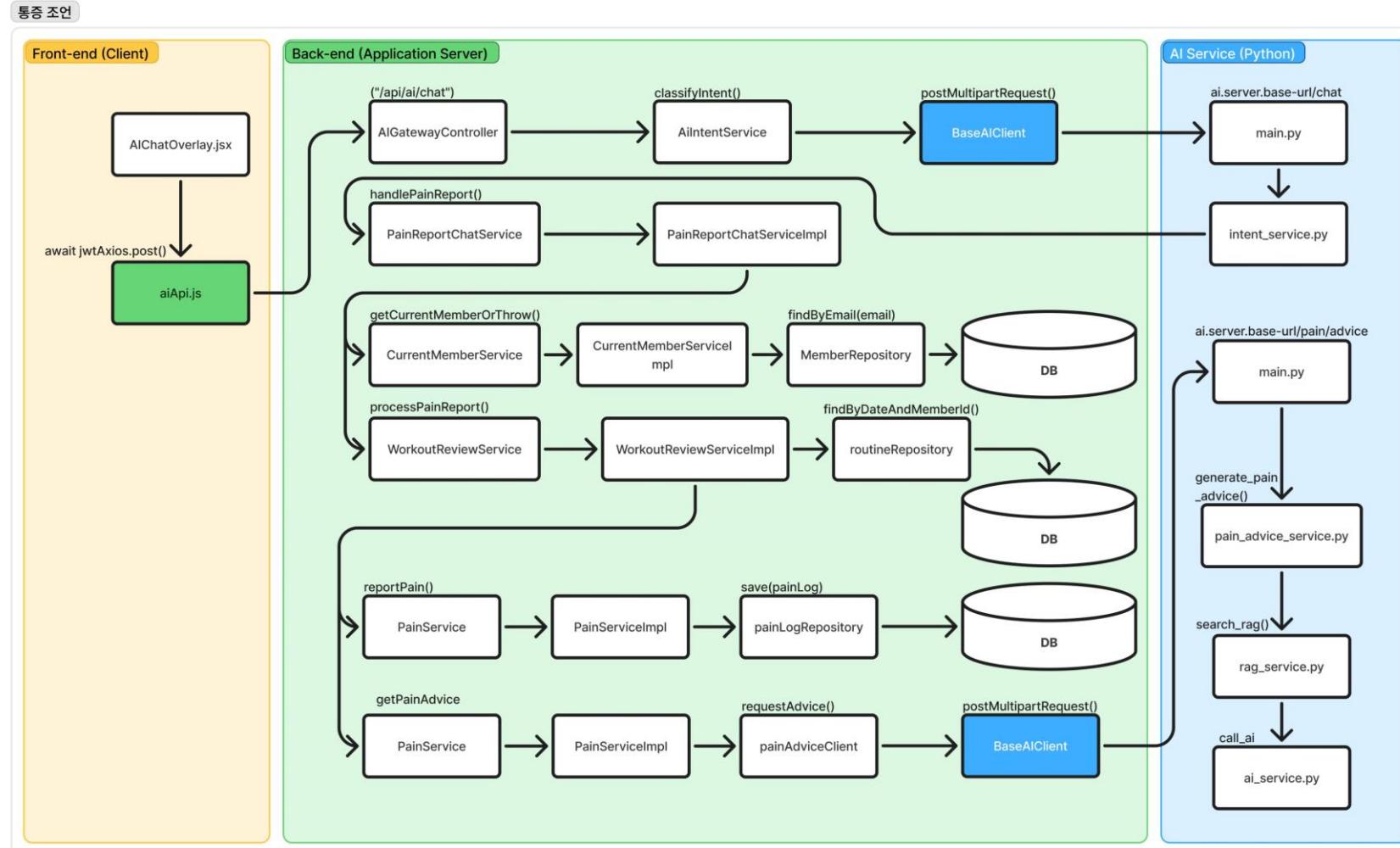
- 모달 닫기·언마운트 시 speechSynthesis.cancel()로 음성 중지

## 6. 설계 포인트

- 운동 중 시선을 화면에 두고 피드백을 들을 수 있도록 음성 피드백 제공
- 사용자가 원할 때 끌 수 있도록 토글 제공
- AI 서버 응답 TTS는 구현되지 않음, exercise 모달 실시간 피드백에만 사용



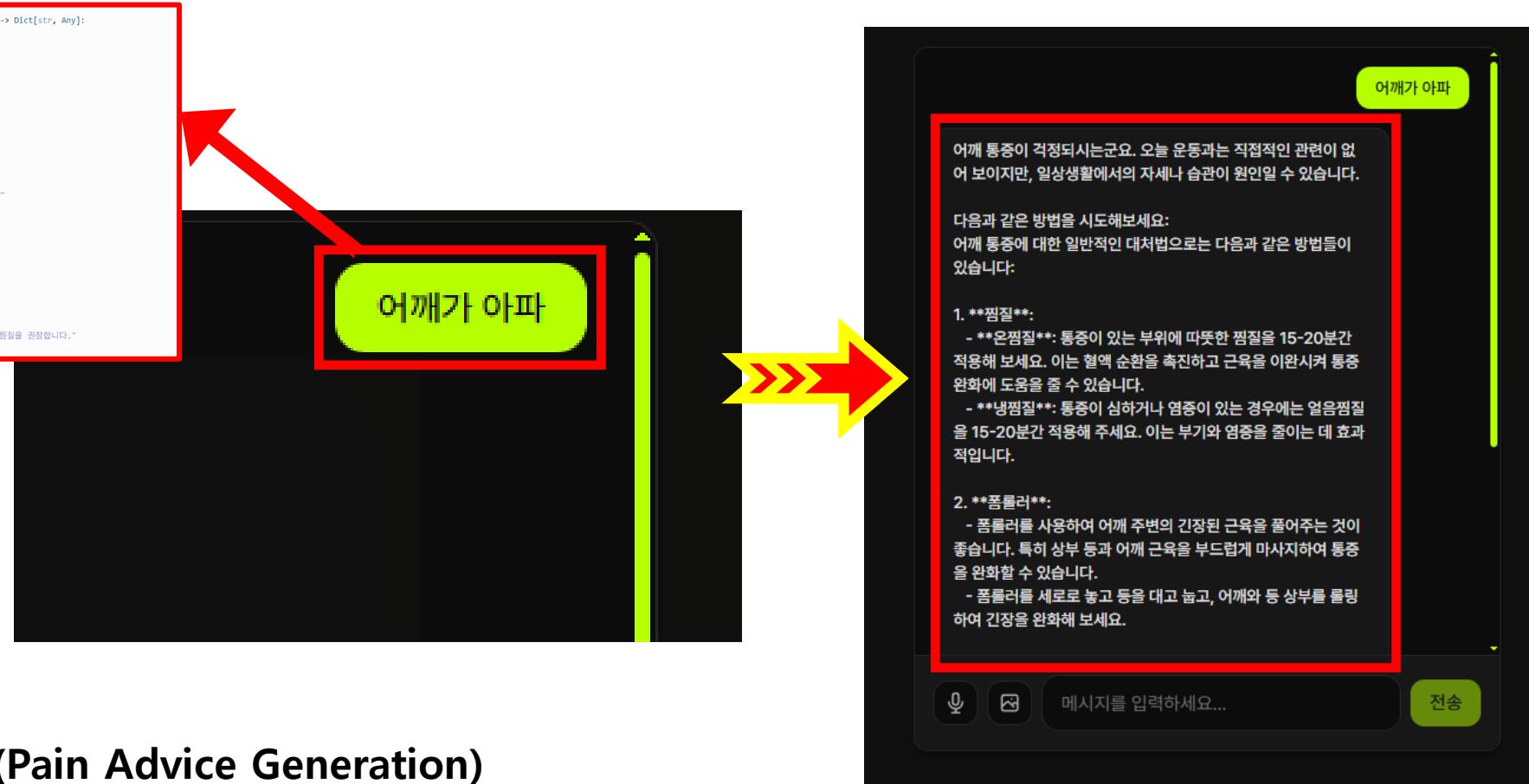
# 9. AI 통증 조언에 따른 실시간 피드백 플로우 차트





# 9. AI 통증 조언에 따른 실시간 피드백 구현

```
def generate_pain_advice(body_part: str, count: int, note: str = None) -> Dict[str, Any]:  
    """통증 조언 생성"""  
    # RAG 검색  
    query = f"(body_part) 통증 {note or ''}"  
    rag_results = search_rag(query, limit=5)  
  
    # 통증 레벨 결정  
    level = "LOW" if count <= 2 else "HIGH"  
    advice_prompt = get_advice_prompt(body_part, count)  
  
    # RAG 컨텍스트 포함  
    rag_context = ""  
    if rag_results:  
        rag_context = "\n\n참고 지식:\n"  
        for result in rag_results:  
            rag_context += f"- {result['title']}: {result['content']}\n"  
  
    # 조언 생성 (통증 조언용 모델 사용)  
    try:  
        advice = call_ai(  
            system_prompt=SYSTEM_PROMPT,  
            user_prompt=f"(advice_prompt)\n{rag_context}",  
            temperature=0.5,  
            model=PAIN_ADVICE_MODEL  
        )  
    except Exception as e:  
        print(f"통증 조언 생성 실패: {e}")  
        advice = f"(body_part) 통증이 {count}회 발생했습니다. 충분한 휴식과 첨성을 권장합니다."  
    return {"level": level, "advice": advice}
```



## ◆ AI 통증 조언 (Pain Advice Generation)

- **RAG 기반 맞춤형 통증 상담지식 검색**: Qdrant RAG로 부위별 통증 관련 지식 검색 (운동 부상, 원인, 대처법 등)
- **조언 생성**: OpenAI GPT-4o-mini가 통증 발생 횟수와 검색된 자료를 결합하여 맞춤형 조언 생성
- **통증 레벨 판단**: 발생 횟수 기반 LOW/HIGH 레벨 자동 판단



# 9. AI 통증 조언에 따른 실시간 피드백 기능 설명

## 1. 역할

- 사용자가 보고한 **통증 부위**와 **횟수**를 기반으로 운동·통증 관리 관련 조언을 제공합니다.

## 2. RAG + LLM 구조

**3.RAG 검색:** 쿼리 "{부위} 통증 {메모}"로 Qdrant 벡터 검색

**4.임베딩:** Sentence-Transformers paraphrase-multilingual-MiniLM-L12-v2 (384차원)

**5.검색 결과:** 상위 5개 문서를 LLM 컨텍스트로 사용

**6.LLM 생성:** 검색된 지식과 함께 통증 조언 문장 출력

## 7. 프롬프트 전략

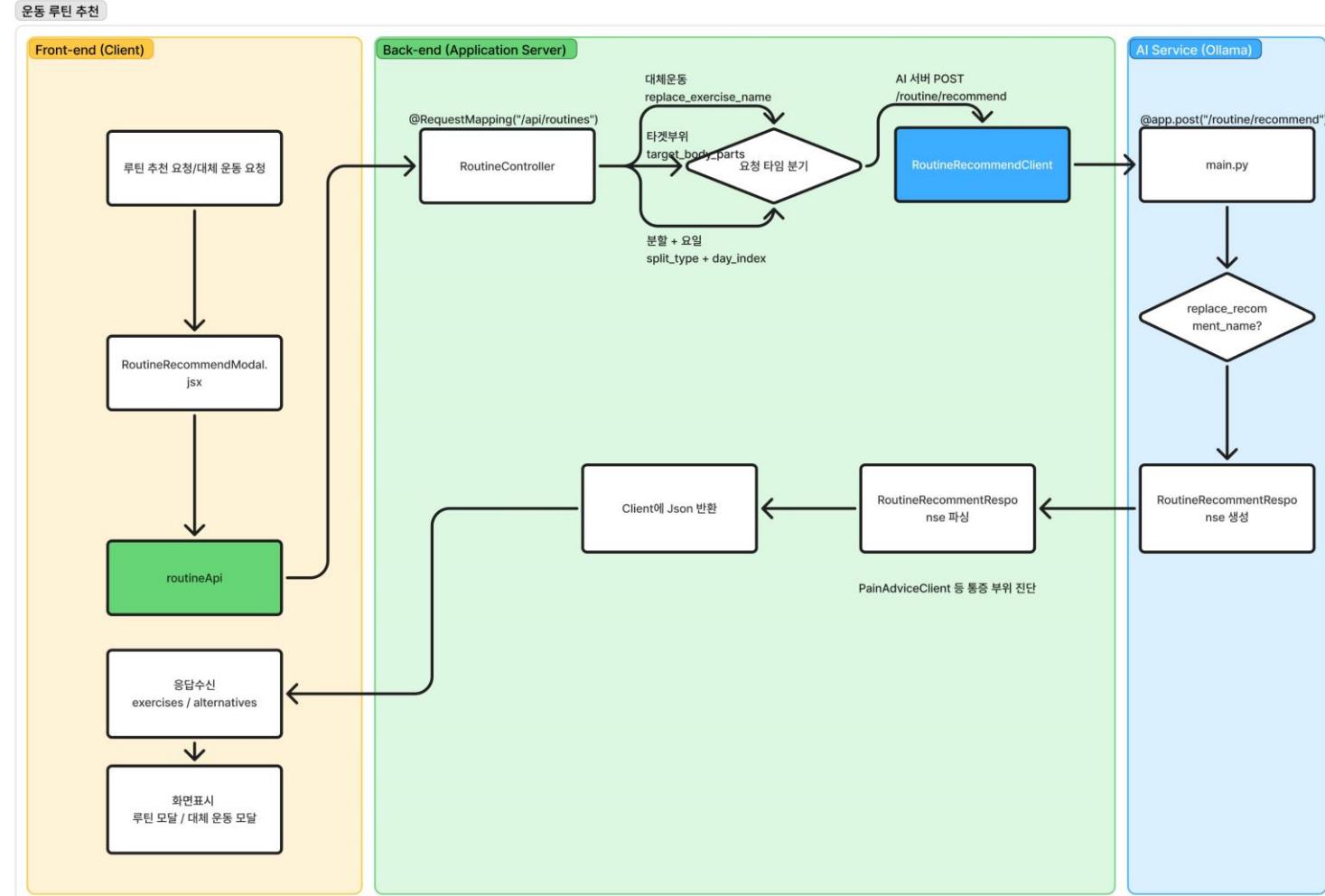
- **횟수 2회 이하:** 짐질, 폼롤러, 스트레칭 등 일반 대처법
- **횟수 3회 이상:** 병원 방문 권고, 해당 부위를 피한 대체 운동 제안

## 8.설계 포인트

- LLM만 사용하지 않고 **RAG 검색**으로 근거 있는 조언 제공
- 한국어·영어 모두 지원하는 **멀티링구얼 임베딩** 사용



# 10. AI 루틴 추천 → 하루 운동 스케줄 계획 플로우 차트





# 10. AI 루틴 추천 → 하루 운동 스케줄 계획 구현

The image shows a sequence of three screenshots from the ALGORHYGYM mobile application:

- Left Screenshot:** A code editor window displaying Python code for generating exercise routines based on target body parts, exclude body parts, and exclude exercise names. A red box highlights the code area.
- Middle Screenshot:** A modal window titled "2분할 루틴" (2-minute split routine) showing two sections: "Upper day" and "Leg day". The "Upper day" section lists exercises like "벤치프레스 (가슴)", "오버헤드프레스 (어깨)", "비벨 컬 (팔)", and "행잉레그레이즈 (복근)". The "Leg day" section lists exercises like "스쿼트 (허벅지)", "힙쓰러스트 (둔근)", "카프레이즈 (종아리)", and "시티드 레그 컬 (허벅지)". A large red arrow points from this screen to the right.
- Right Screenshot:** The main application interface titled "TODAY'S ROUTINE". It shows a sidebar with navigation options: Home, Routine (highlighted), Log, Diet, Shopping, Ranking, and Profile. The main content area displays "AI 코칭 요약" (AI coaching summary) with tips about warming up and stretching. Below it are two sections: "스크루드" (Squats) and "힙쓰러스트" (Hip Thrusts), each with a set of 3 reps at 10 weight. Each section has four buttons: "▶ 세트 시작" (Start Set), "✓ 완료됨" (Completed), "수정" (Edit), and "삭제" (Delete). A green button at the bottom right says "다음 운동" (Next Exercise).

## ◆ AI 루틴 추천 (Routine Recommendation)

- **안전성 고려한 맞춤 운동 추천지식 기반 검색:** Qdrant RAG로 운동 지식 검색 (운동 방법, 타겟 부위, 부상 위험 부위 등)
- **부상방지 필터링:** 타겟 부위 유지 + 부상 위험 부위 배제로직으로 안전한 운동만 추천
- **다양한 루틴 추천:** 요일별 타겟 부위에 맞는 운동 추천

# 10. AI 루틴 추천 → 하루 운동 스케줄 계획 가능 설명



## 1. 역할

- 타겟 부위·배제 부위를 기준으로 **운동 목록**을 추천하고, 특정 운동의 **대체 운동**을 제안합니다.

## 2. 사용 기술

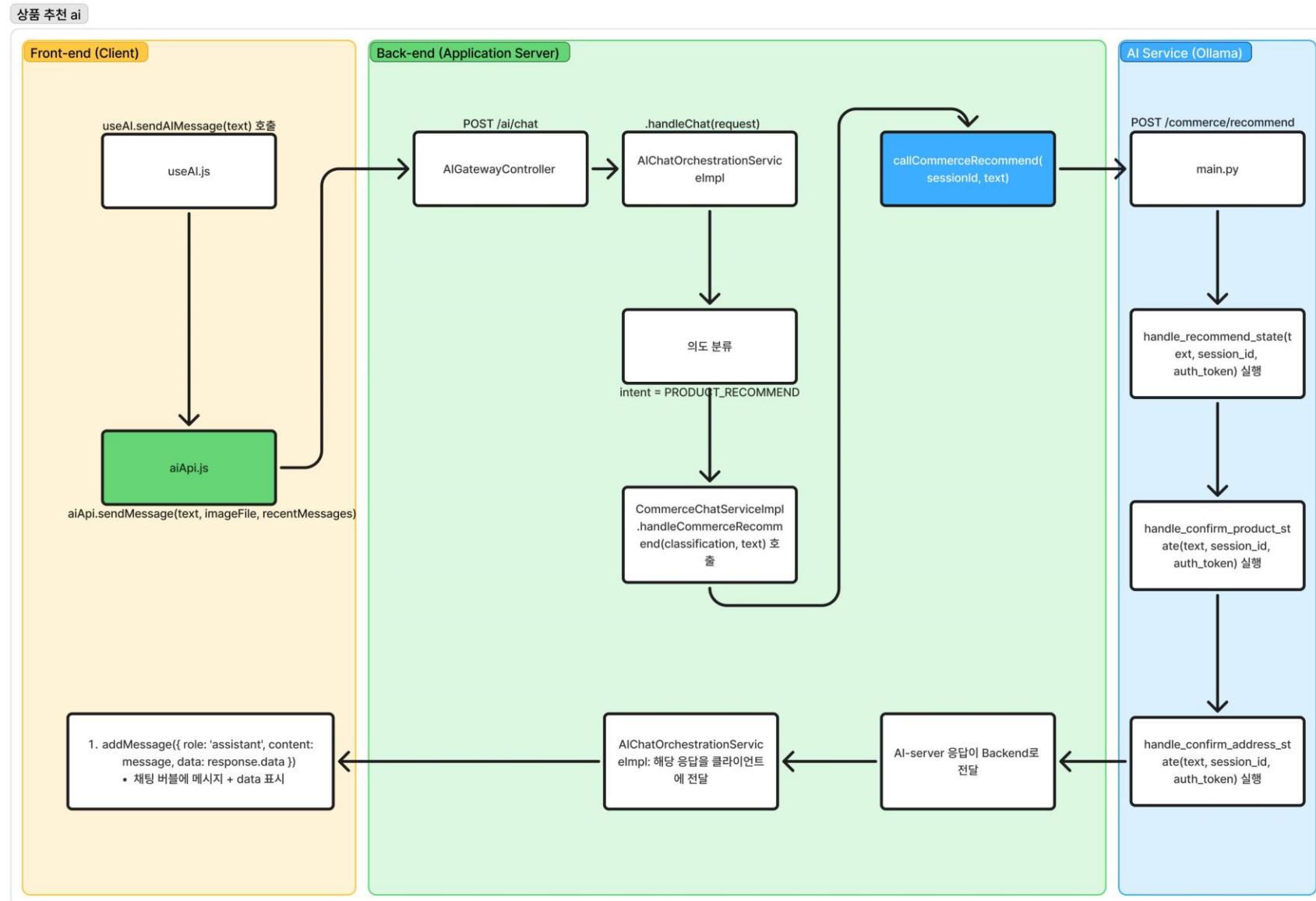
- **rag\_data.json**: exercise\_routine, exercise\_injury 등 정적 데이터
- 별도 벡터 DB·LLM 없이 **필터·매칭 로직**으로 추천

## 3. 동작 방식

- exercise\_routine에서 해당 일의 타겟 부위와 맞는 운동만 선택
- risk\_factors와 exclude\_body\_parts가 겹치면 제외
- 부상 위험 부위를 배제한 추천



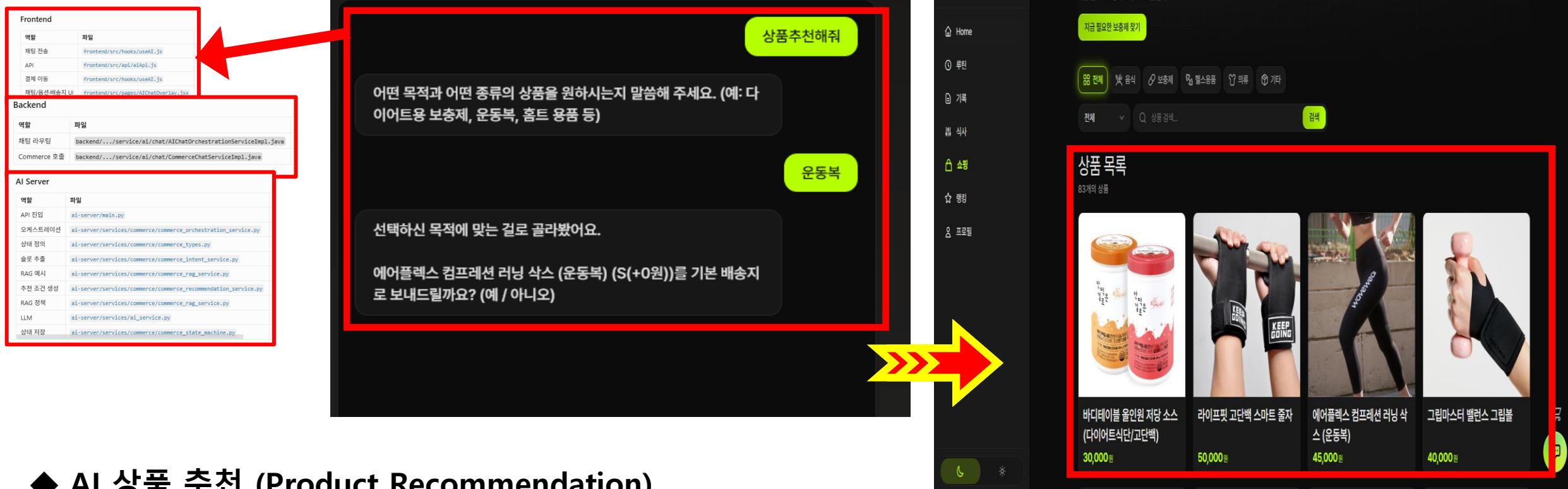
# 11. AI 상품 추천 - 채팅 입력에 따른 상품 추천 플로우 차트





# 11. AI 상품 추천 - 채팅 입력에 따른 상품 추천 구현

Welcome, 오재현님



## ◆ AI 상품 추천 (Product Recommendation)

- 대화형 상태머신 기반 상품 추천의도/슬롯 추출:** OpenAI GPT-4o-mini로 사용자 발화에서 예산, 카테고리, 키워드, 배송지 등 추출
- 정책 문서 검색:** Qdrant RAG로 추천 정책 문서 검색하여 추천 기준 생성
- 상태머신 플로우:** RECOMMEND → CONFIRM → PAYMENT 상태 전이로 자연스러운 대화형 추천 진행

[TOP](#)

# 11. AI 상품 추천 - 채팅 입력에 따른 상품 추천 가능 설명



## 1. 역할

- 사용자 자연어로 상품 추천을 요청하면, **의도·슬롯 추출** → **RAG 기반 추천 조건 생성** → **백엔드 추천 API** → **상품·옵션 선택** → **order draft**까지 대화형으로 처리합니다.

## 2. 사용 기술

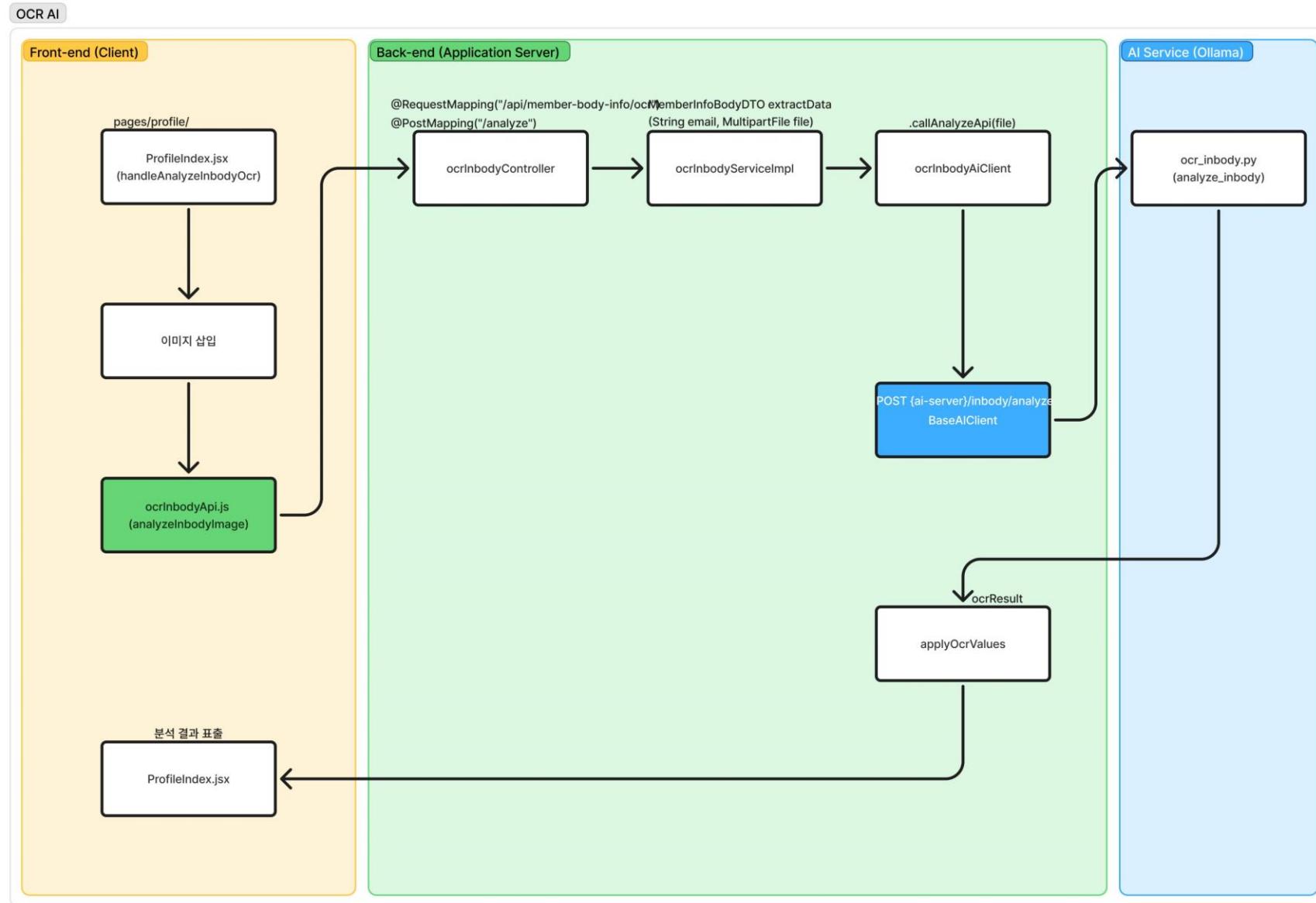
- RAG**: Qdrant + Sentence-Transformers (정책 문서)
- LLM**: OpenAI gpt-4o-mini (의도·조건·대화 생성)
- Redis**: 대화 세션·상태머신 저장

## 3. 동작 방식

- 의도/슬롯**: RAG에서 유사 발화 예시 검색 → few-shot → LLM으로 intent·슬롯 추출
- 추천 조건**: 정책 RAG 검색 → LLM으로 RecommendationCondition 생성
- 상태머신**: CONFIRM\_PRODUCT → ADD\_TO\_CART → CONFIRM\_ADDRESS → PAYMENT\_READY 등 단계 관리



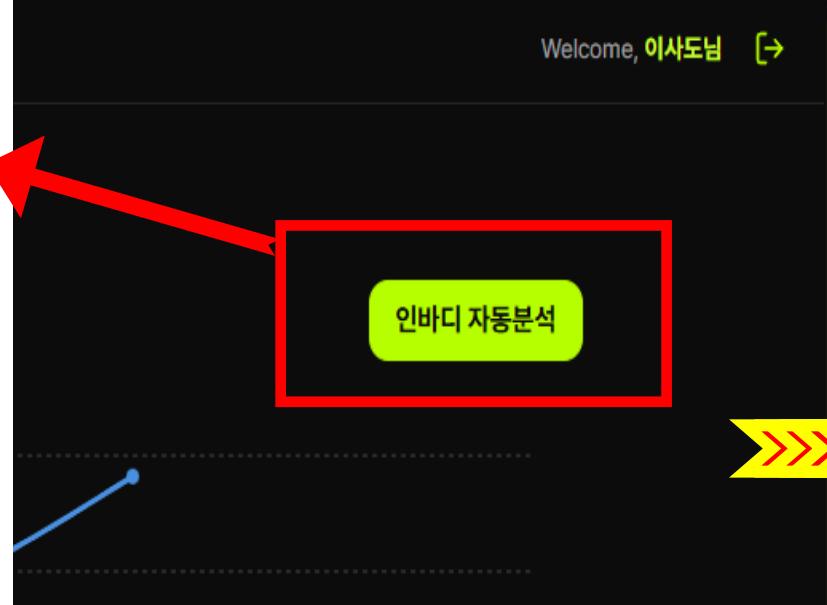
# 12. OCR AI (인바디 이미지 인식) 플로우 차트





# 12. OCR AI (인바디 이미지 인식) 구현

```
public <T> T postRequest(String endpoint, Map<String, Object> requestBody, Class<T> responseType) {  
    try {  
        String url = aiServerBaseUrl + endpoint;  
  
        log.info(format("AI 서버 호출: url=%s, endpoint=%s", url, endpoint));  
  
        HttpHeaders headers = new HttpHeaders();  
        headers.setContentType(MediaType.APPLICATION_JSON);  
  
        HttpEntity<Map<String, Object>> request = new HttpEntity<>(requestBody, headers);  
  
        ResponseEntity<T> response = restTemplate.postForEntity(  
            url,  
            request,  
            responseType);  
    } catch (Exception e) {  
        log.error("AI 서버 호출 중 예외 발생: " + e.getMessage());  
        throw new RuntimeException(e);  
    }  
}
```



## ◆ OCR AI (인바디 이미지 인식)

### - 이미지 → 구조화 데이터 추출: OpenAI GPT-4o-mini

Vision(백엔드) 또는 Paddle OCR(ai-server)로 인바디 보고서 이미지에서 텍스트 인식 후, 체중·키·골격근량 등 필드 추출

### - 라벨-값 매칭·검증: 박스 좌표 기반 라벨-숫자 근접 매칭(같은 행, 라벨 오른쪽 값),

키별 합리 범위 필터(in\_range), 체성분 관계(체수분+단백질+무기질+체지방량 = 체중)로 적정체중·체성분 값 교정

처리 플로우: 이미지 업로드→(선택) 인바디/음식 이미지 분류(CLIP)→OCR 추출→파싱·범위·관계 검증→프론트 검증 모달 표시→사용자 확인 후 저장



# 12. OCR AI (인바디 이미지 인식) 기능 설명

## 1. 역할

- 인바디 기기 결과 화면 이미지에서 **체수분·단백질·무기질·체지방량·골격근량·체중·체지방률·적정체중·체중조절** 등을 구조화된 JSON으로 추출합니다.

## 2. 사용 모델

- OpenAI gpt-4o-mini Vision

## 3. 정확도 향상 전략

### A) 이미지 분할·크롭

- 상단(1~3): 헤더, 체성분 표, 골격근·지방 분석
- 하단(4~5): 비만 진단, 체중조절
- 체중조절 전용: 우측 하단 영역만 타이트 크롭
- 단백질·무기질: 표 영역만 별도 크롭 (fallback용)

### B) 체수분(bodyWater) 보강

- 표에서 체수분 숫자만 잘라내는 초타이트 크롭
- PNG 무손실로 전달해 2↔3 같은 오독 감소
- 대비·샤프닝 과적용을 피하기 위해 최소 전처리만 적용

### C) Fallback

- 단백질·무기질이 0 또는 null → 미세 크롭 fallback
- 골격근량·체지방량·체지방률 누락 → 마이크로 fallback
- 체중조절 누락 → 전용 크롭 fallback

### D) 프롬프트 규칙

- 괄호 안 표준범위는 사용하지 않음
- 신체변화 그래프·추세선 숫자는 사용하지 않음
- 표 내 측정치만 추출**

## 4. 병렬 처리

- 상단·하단·체중조절 크롭을 `asyncio.gather`로 병렬 Vision 호출
- 응답 시간 단축

## 5. 저장 전 사용자 검증

- OCR 결과는 **바로 저장하지 않고** 프론트로 반환
- 사용자가 검증 모달에서 확인·수정 후에만 최종 저장
- OCR 오독 가능성을 줄이고 데이터 품질을 확보

TOP

감사합니다