

# 2차(IIIm 작동 방식 예시)

## 1. 사용자 정보 입력

- 필수: 성별, 나이
- 선택/추가: 증상 개선 목표 (피로도, 면역력, 피부 등), 알레르기, 복용 중인 영양제

## 2. 영양제 데이터 관리

- Supplements 테이블: 영양제 기본 정보
- Ingredients 테이블: 성분 정보 + 권장량
- Supplement\_Ingredients: 성분-영양제 관계

## 3. 추천 로직

- 하루 권장량(RDA) 초과하지 않도록 성분 계산
- 중복 성분 제거 (여러 영양제에 동일 성분 포함 시 합산)
- 사용자의 개선 목표에 맞는 성분 우선 추천

## 4. LangGraph 활용

- RAG 기반 질의 응답: "사용자가 면역력 강화를 원할 때 어떤 영양제를 추천할까?"
- SQL DB와 연동: 필요한 성분, 권장량 체크
- 추천 이유/설명 생성: "왜 이 영양제가 적합한지" LLM이 설명

## 5. 결과 제공

- 추천 영양제 리스트 + 성분별 섭취량
- 하루 권장량 대비 섭취 비율 표시
- 중복 피하기 위한 경고/대안 표시

좋아요! 😊 그러면 8일 × 8시간 = 총 64시간 기준으로, 단계별 To-Do 리스트 + 파이썬/SQL/LangGraph 구조를 만들어서 바로 개발 로드맵으로 쓸 수 있게 정리해드릴게요.



## 8일간 작업 로드맵

### Day 1 (8h) — 데이터 확인 & 설계

- **목표:** 데이터 구조 이해, SQL 스키마 설계
  - **세부 작업**
    1. JSON 데이터 로드 및 컬럼 확인 (pandas)
    2. 결측치, 중복 데이터 확인 및 정제
    3. SQL DB 스키마 설계
      - Tables: Supplements, Ingredients, Supplement\_Ingredients, Symptoms\_Ingredients
    4. 증상-성분 매핑 테이블 초안 작성
  - **산출물:** SQL 스키마 + 초기 DB 생성 스크립트
- 

## Day 2 (8h) — SQL DB 구축 & 데이터 적재

- **목표:** 정규화 DB 구축 및 데이터 삽입
  - **세부 작업**
    1. SQL DB 생성 (MySQL/PostgreSQL 등 선택)
    2. JSON → SQL 변환 코드 작성 (Python pandas.to\_sql)
    3. 권장량(RDA) 컬럼 포함하여 성분 테이블 채우기
    4. Supplement\_Ingredients M:N 관계 테이블 완성
  - **산출물:** SQL DB 초기 데이터 삽입 완료
- 

## Day 3 (8h) — 추천 로직 파이썬 구현

- **목표:** RDA 기반, 중복 성분 제거, 증상별 추천 로직
- **세부 작업**
  1. 사용자 입력 파이썬 클래스 설계 (User)
  2. 증상 → 필요한 성분 추출
  3. 영양제 후보 필터링
    - 하루 권장량 초과 방지
    - 중복 성분 제거
  4. 추천 점수 계산 (예: 증상과 성분 매칭 정도)
- **산출물:** 추천 알고리즘 Python 코드 + 테스트 케이스

---

## Day 4 (8h) — 추천 로직 검증 & 최적화

- **목표:** 추천 정확도 향상, 권장량 체크 검증
  - **세부 작업**
    1. 다양한 사용자 케이스 테스트 (성별, 나이, 증상 조합)
    2. RDA 초과 여부, 중복 성분 체크 로직 검증
    3. 추천 우선순위 알고리즘 개선
      - 예: 증상 관련 성분 가중치 부여
  - **산출물:** 검증 완료된 추천 로직, 알고리즘 문서화
- 

## Day 5 (8h) — LangGraph 통합 준비

- **목표:** RAG용 문서 준비 및 LangGraph 구조 설계
  - **세부 작업**
    1. SQL DB → RAG용 Document 생성
      - 영양제 이름, 성분, 권장량, 증상 매핑
    2. LangGraph 노드/엣지 설계
      - Node: Supplement, Ingredient, Symptom
      - Edge: 포함관계, 개선관계
    3. LangGraph 기본 연결 테스트
  - **산출물:** LangGraph 초기 구조 + 문서 데이터
- 

## Day 6 (8h) — LangGraph RAG 연동

- **목표:** 사용자 질의 → 추천 결과 반환
  - **세부 작업**
    1. 사용자 입력 (성별, 나이, 증상) → LangGraph 쿼리
    2. 추천 알고리즘 결과와 RAG 결과 결합
    3. 추천 이유 생성 프롬프트 작성
  - **산출물:** LangGraph RAG 추천 시나리오 완성
-

## Day 7 (8h) — UI/UX 구현 & 결과 시각화

- **목표:** 사용자 친화적 입력/출력 인터페이스
- **세부 작업**
  1. 입력 폼 구현 (웹/CLI)
    - 성별, 나이, 증상, 기존 복용 여부
  2. 추천 결과 테이블/그래프 표시
    - 성분별 권장량 대비 섭취량
    - 초과 시 경고 표시
  3. LangGraph 결과와 파이썬 추천 로직 연동
- **산출물:** 기본 UI/UX + 추천 결과 시각화

## Day 8 (8h) — 최종 검증 & 배포 준비

- **목표:** 전체 시스템 통합 테스트, 안정화
- **세부 작업**
  1. 추천 로직 + LangGraph + UI 통합 테스트
  2. 다양한 사용자 케이스 테스트
    - 피로 개선, 면역 개선, 중복 성분, 권장량 초과
  3. 코드 정리, 문서화
  4. 최종 배포/데모 준비
- **산출물:** 완성된 에이전트 + 테스트 보고서 + 배포 자료

## ◆ 전체 구조 요약

[사용자 입력] → Python 추천 로직 → SQL DB 조회  
↳  
LangGraph RAG → 추천 이유 생성  
→ [추천 영양제 리스트 + 섭취량/경고 표시] → UI/UX

- **Python:** 추천 알고리즘, 중복 성분 제거, 권장량 계산
- **SQL:** 영양제/성분/증상 데이터 관리, 필터링

- **LangGraph**: RAG 기반 추천 이유 생성, 증상-성분 관계 탐색
  - **UI/UX**: 사용자 입력 + 결과 표시
- 

원하면 내가 이 일정표를 **시간 단위 To-Do 리스트 + 각 단계별 코드 템플릿**까지 만들어서, 바로 코딩하면서 체크리스트로 쓸 수 있게 만들어줄 수 있어요.

그거 만들어줄까?