프로젝트 경과보고

# 1⃣ 지난주 프로젝트 진행 상황

## ✅ 요구 분석

- 주요 요구 사항 정리:

- 사용자/시스템 요구 명세서 작성

- 특이사항:

## ✅ 설계

### DB 모델링:

### UI/UX 설계:

### 데이터 수집:

## ✅ 구현

### Front-end

정윤

* 관리자 페이지구현:
* 반응형 사이드바 탐색 기능
* 사용자 분석결과 통계 확인 대시보드 구현
* 검색 및 삭제 기능이 있는 회원 관리 페이지
* 게시물 및 카테고리에 대한 콘텐츠 관리 추가
* 관리자 기능:
* 대시보드: 회원 통계, 성별 분석, 연령대 분석, 인기 있는 운동
* 회원 관리: 검색, 강제 계정 삭제
  + 강제탈퇴 - 이메일 알람 보내기
* 콘텐츠 관리: 카테고리 CRUD, 게시물 삭제
* 알림 기능 보완:
* API 응답 구조에 맞게 타입 정의 업데이트 (referenceId → postId, read → isRead)
* 알림 클릭 시 게시글로 정확한 이동 기능 구현
* 새로고침 후에도 읽음 상태 유지되도록 데이터 변환 로직 추가
* POST\_COMMENT, POST\_LIKE, COMMENT\_REPLY 등 새로운 알림 타입 지원
* API 호출 에러 처리 및 로깅 강화
* 루틴 count되는거, 회원가입창 좀 더 예쁘게 보완

지혜

* 분석결과 페이지
  + userid로 본인의 분석결과만 확인 가능
  + 비회원에게도 분석결과 공유가능(공유페이지는 읽기전용)
* 헤더 메뉴접근 권한
  + 커뮤니티, 운동목록, 자세분석 userRequireAuth를 통해 접근 권한 체크
* 로그인 페이지
  + 로고 클릭시 홈페이지 랜딩
* 관리자 페이지
  + 관리자 로그인 시 관리자 페이지로 랜딩
  + 관리자 페이지에서 사용자 페이지로 이동 메뉴 추가
  + 다크모드
  + 반응형에 따른 화면 및 그래프 조정
  + 회원 통계
    - 한달 기준 매주 사용자 증가 막대그래프
    - 1년 기준 매월 사용자 증가 막대그래프
    - 회원 중 성별 비율 원그래프
    - 회원 중 나이대 비율 원그래프
  + 콘텐츠 통계
    - 게시글 좋아요 수/ 조회 수/ 댓글 수
* 챗봇
  + 각 페이지마다 챗봇 아이콘 넣기
  + 페이지 랜더링 시 챗봇창 사이즈 줄이고, 입력창에 포커스 시 챗봇창 사이즈 커지도록 하기
  + 서로 다른 사용자의 대화내역이 보이는 문제 해결
    - 원인 : 결과분석 페이지를 더미로 고정된 페이지로 랜딩되면서 챗봇 대화 내역도 다른 사용자의 대화가 보임
    - 해결 : 결과분석 페이지를 userid별로 구분
* 뒤로가기 이벤트 처리
  + 로그인 되어있음에도 뒤로가기 했을 때 로그인 화면 나오는 문제 해결
* 다크모드
* 리팩토링
  + tsx에서 바로 주소요청하는 부분 api로 진행하기

### Back-end

정윤

* 관리자 기능
* 사용자 관리
* UserController: 전체 사용자 조회, 사용자 삭제(소프트 삭제) API 추가
* UserDTO: 비밀번호 필드에 [**@JsonIgnore**](https://github.com/JsonIgnore) 적용
* User 엔티티: 프로필 이미지 [**@lob**](https://github.com/lob) 및 LAZY 로딩 설정
* UserRepository: 활성 사용자 수, 날짜별 사용자 통계 메서드 추가
* UserServiceImpl: 사용자 삭제 시 소프트 삭제로 변경, 불필요한 import 제거
* 관리자 기능
* AdminController: 관리자 대시보드 및 회원 목록 조회 API 추가
* AdminDTO: 대시보드 응답 및 회원 정보 DTO 정의
* AdminService / AdminServiceImpl: 대시보드 통계, 성별·나이대별 분석 점수, 인기 운동 데이터 조회 구현
* 데이터 분석
* AnalysisHistoryRepository:
  + - 성별/나이대별 평균 분석 점수 조회 쿼리 추가
    - 성별 분석횟수 비교, 나이대별 분석 횟수 비교 쿼리 추가
* 게시글 관리
* PostServiceImpl: 게시글 삭제 시 소프트 삭제로 변경
* 권한 설정 SpringSecurity

지혜

* 관리자 통계
  + 회원 통계
    - 한달 기준 매주 사용자 증가 막대그래프
    - 1년 기준 매월 사용자 증가 막대그래프
    - 회원 중 성별 비율 원그래프
    - 회원 중 나이대 비율 원그래프
  + 콘텐츠 통계
    - 게시글 좋아요 수/ 조회 수/ 댓글 수
* 챗봇 redis로 사용자별 대화 구분
  + DTO 생성
    - ChatMessage, ChatRequest, ChatResponse
  + ChatbotRedisService 생성
    - 현재 아이디와 채팅 세션은 24시간동안 저장되고 삭제되도록 설정
  + ChatbotController 생성
    - 새로운 세션 생성, 기존 활성 세션이 있으면 재사용, 챗봇 메시지 전송, 초기 메세지 생성 및 저장, 대화 기록 조회, 사용자별 활성 세션 조회, 더미 AI 응답

### Agent (예: AI Tool, RAG 등)

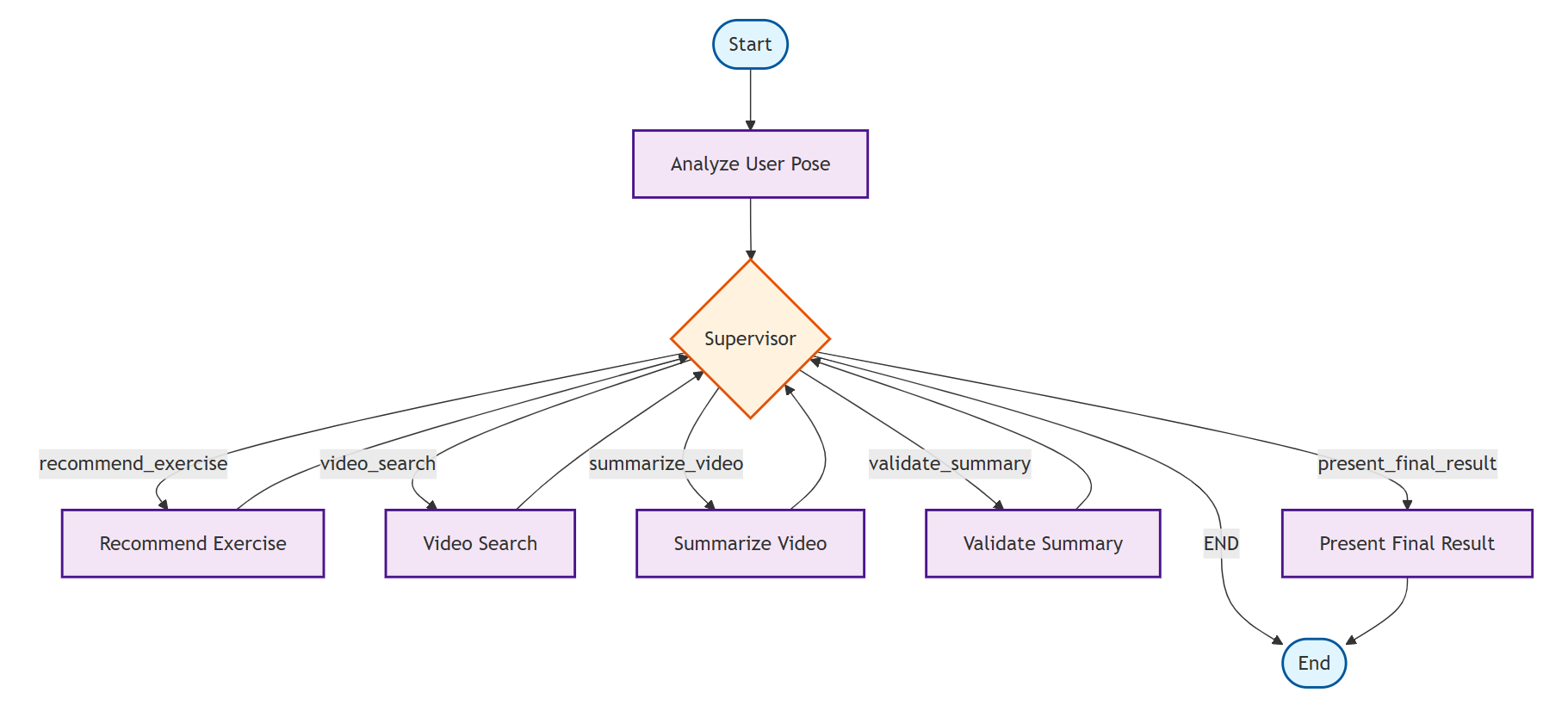
정윤

* fast-mcp에서 자세 점수 계산 로직 추가하기

| **항목** | **의미** | **등급별 최대 편차와 최소 점수 [(최대 편차, 최소 점수)]** |
| --- | --- | --- |
| shoulder\_tilt\_angle | 어깨 기울기 각도 | [(1.5°, 90), (3.5°, 70), (8.5°, 40), (20°, 0)] |
| hip\_tilt\_angle | 골반 기울기 각도 | [(1.5°, 90), (3.5°, 70), (8.5°, 40), (20°, 0)] |
| torso\_tilt\_angle | 몸통 기울기 각도 | [(2°, 90), (4°, 70), (9.5°, 40), (22°, 0)] |
| neck\_forward\_angle | 거북목 각도 | [(8°, 90), (15°, 70), (25°, 40), (35°, 0)] |
| back\_angle (수직선 편차) | 척추 굽음(등 허리 굴곡) | [(4°, 90), (10°, 70), (20°, 40), (30°, 0)] |

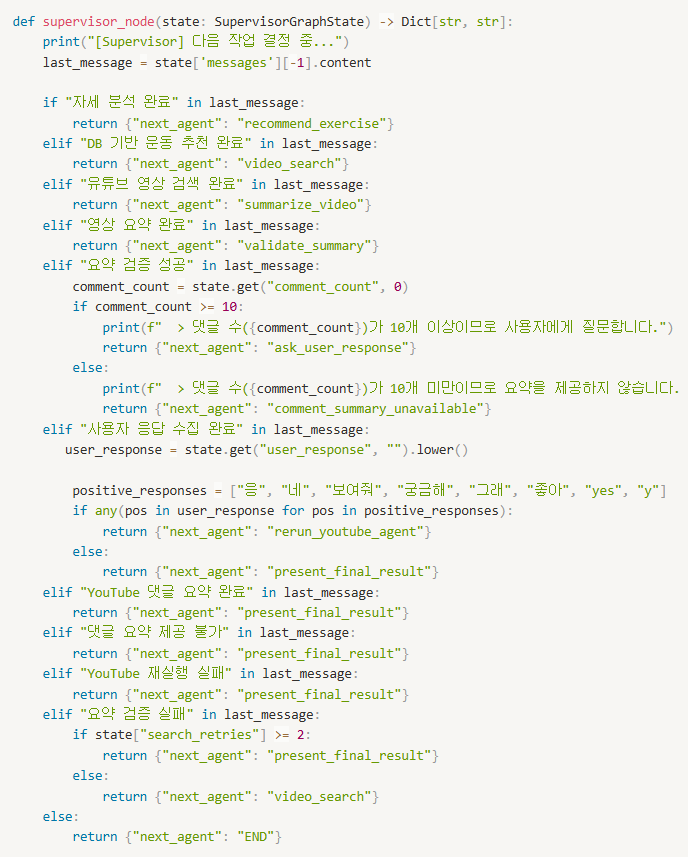
* 한사람만 자세분석 할 수 있도록 하기
* 자세진단 llm 붙이기 (ollama 시도해보고 안되면 gpt)
  + Ollama API를 활용한 자세 진단 메시지 생성 및 번역 기능 추가.
  + Streamlit 앱에서 Ollama 진단 메시지를 UI에 표시.
* 랭그래프 작업 내용 - Supervisor

#### main\_graph



##### 핵심 구성 요소

###### 상태 관리 (SupervisorGraphState)



###### 서비스 초기화

* **PostureAnalyzer**: 자세 분석 (YOLO 모델)
* **ExerciseVectorDB**: 운동 추천 (벡터 데이터베이스)
* **ChatbotActionNode**: 유튜브 영상 검색
* **youtube\_summary\_agent**: 영상 요약

##### 워크플로우 실행 순서

###### 1단계: 자세 분석 (analyze\_user\_pose\_node)

* 입력 이미지에서 자세를 분석
* YOLO 모델로 포즈 데이터 추출
* Ollama를 통해 진단 텍스트 생성

###### 2단계: 운동 추천 (recommend\_exercise\_node)

* 진단 내용을 기반으로 LLM이 검색 쿼리 생성
* VectorDB에서 가장 적합한 운동 1개 검색
* 예: "중급 난이도의 유연성을 높이는 효과를 가진 스트레칭 운동"

###### 3단계: 영상 검색 (video\_search\_node)

* 추천된 운동명을 기반으로 유튜브 검색
* 운동명에 "하는 법" 또는 "운동 자세" 키워드 추가
* 재검색 횟수 추적 (최대 2회)

###### 4단계: 영상 요약 (summarize\_video\_node)

* 찾은 유튜브 영상의 내용을 요약
* youtube\_summary\_agent 사용

###### 5단계: 요약 검증 (validate\_summary\_node)

* LLM으로 요약 내용이 진단과 관련 있는지 검증
* 구조화된 출력 사용 (ValidationResult)
* 검증 실패 시 재검색 또는 종료 결정

###### 6단계: 최종 결과 제시 (present\_final\_result\_node)

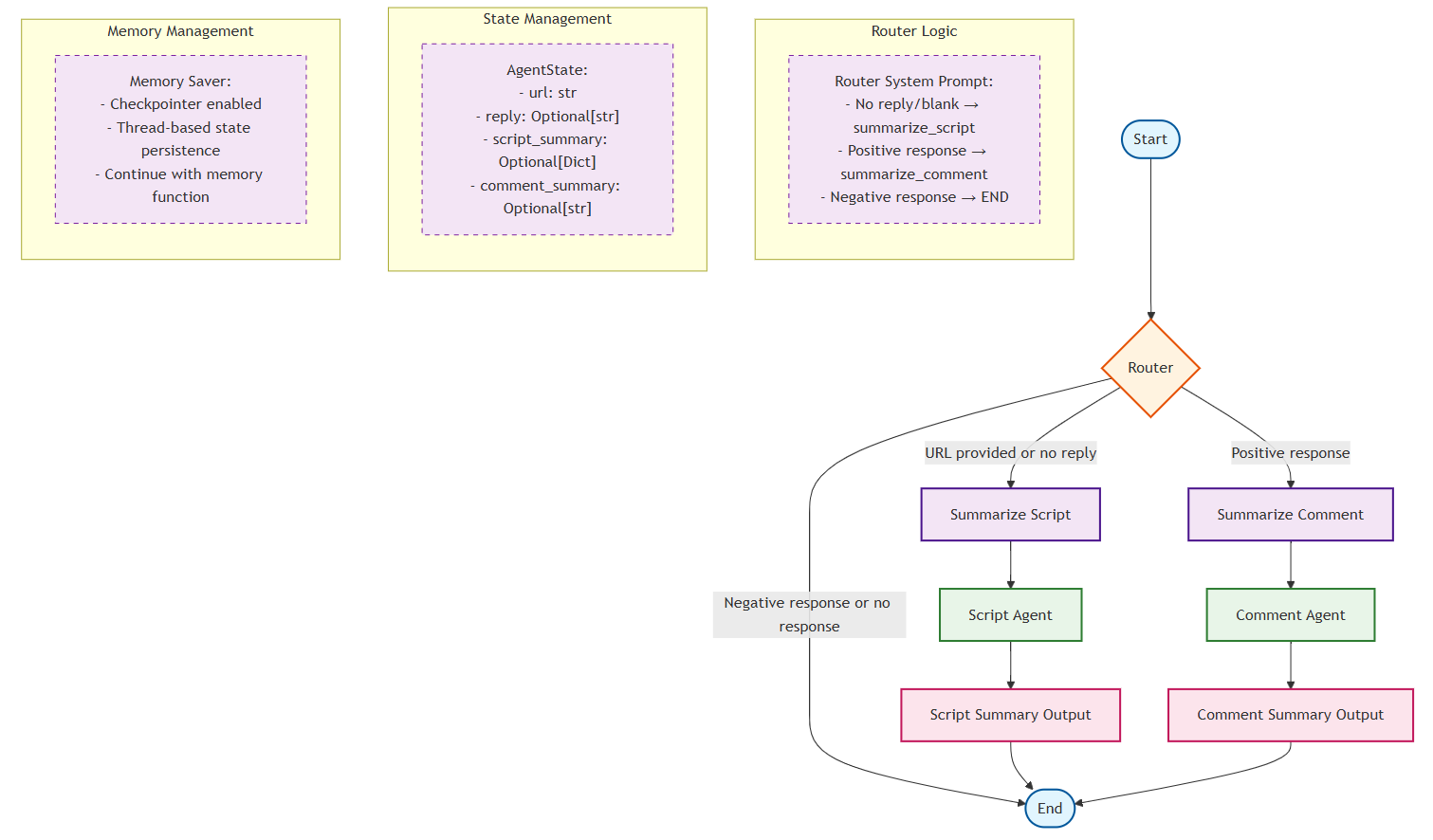
* 모든 결과를 JSON 형태로 통합
* 성공/실패 여부와 함께 완전한 추천 정보 제공

##### Supervisor 제어 로직

각 노드가 실행 결과를 HumanMessage로 supervisor에게 보고하고, supervisor는 이를 바탕으로 다음 실행할 노드를 결정



#### Youtube\_agent

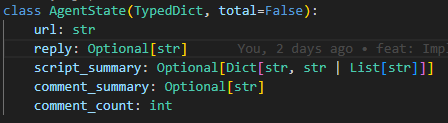


##### 라우터 기반 조건부 분기 패턴

: 사용자의 입력과 상태에 따라 두가지 경로로 분기

→ Route Pydantic 모델로 구조화된 라우팅 결정

##### 상태관리



##### 라우터 로직

* **summarize\_script**: URL이 있거나 reply가 없는 경우
* **summarize\_comment**: 긍정적 응답("응", "네", "보여줘" 등)
* **END**: 부정적 응답 또는 무응답

##### 메모리 관리

* MemorySaver로 상태 지속성 제공
* Thread 기반 대화 흐름 관리
* continue\_with\_memory 함수로 이전 상태 유지하며 새로운 상태 업데이트

##### 워크플로우 실행 흐름

###### 1단계: 초기 URL입력 → 스크립트 요약

initial\_state = {'url': url}

step1\_state = graph.invoke(initial\_state, config=config)

###### 2단계: 사용자 응답에 따라 댓글 요약 실행

step2\_state = continue\_with\_memory(graph, graph\_memory, config,

{"reply": "응", "url": step1\_state.get("url")})

인우

* Yolo 모델 스트림릿 테스트 페이지
* Redis PC에서 외부접속 열기
* 감정분석 자연어 데이터(AI\_HUB) 2개 전처리
* luxia 파인튜닝 중

# 2⃣ 금주 진행 계획

## 분석/설계 수정 기능:

해당 사항 없음

## 개발/디버깅 항목(팀원별):

인우

* 랭그래프 Fast API 연결
* 가능하면 MCP
* Luxia 학습

정윤

* 감정분석 리포트 페이지 FE, BE 구현

지혜

* react-springboot-fastapi-model-socket 연결

## 주간 목표:

* 감정분석 구현
* 챗봇