# 캡스톤 디자인과 창업 프로젝트 A 2차 보고서

청소년의 성 지식 탐구를 지원하는

OpenAI & Hybrid LangChain RAG 기반  성교육 챗봇 어플리케이션

27팀

2271107 이경선

2271106 박서현

# 목차


1. Team Information
    1.1. 과제명
    1.2. 팀 정보
    1.3. 팀 구성원

2. Project Summary
    2.1. 문제 정의
    2.2. 기존 연구와의 비교
    2.3. 제안 내용
    2.4. 기대 효과 및 의의
    2.5. 주요 기능 리스트

3. Project Design
    3.1. 요구사항 정의
    3.2. 전체 시스템 구성
    3.3. 주요 엔진 및 기능 설계
    3.4. 주요 기능의 구현
    3.5. 기타

# 1. Team Information

---

## 1.1 과제명

성큼성큼: 청소년의 성 지식 탐구를 지원하는 OpenAI & Hybrid LangChain RAG 기반 성교육 챗봇 어플리케이션

## 1.2 팀 정보

팀 번호: 27

팀 이름: 용용

## 1.3 팀 구성원

- 이경선(리더): RAG 기반 검색-생성 파이프라인 설계 및 구현, 프론트엔드 앱(UI)
- 박서현(팀원): 백엔드 및 서버 구현, MySQL 기반 DB 구축

# 2. Project Summary

## 2.1 문제 정의

(1) 배경

국내 청소년 대상 성교육은 교과 과정 내 일부 주제(예: 생식 구조, 피임법 등)에 한정되어 있어, 실제 청소년이 직면하는 성적 지향·정체성, 대인관계 등의 문제를 충분히 다루지 못한다. 그러나 청소년은 부모나 교사에게 직접 질문하기 어려워 인터넷에 의존하는 경우가 많고, 이 과정에서 자극적이거나 출처가 불분명한 정보에 노출될 위험이 크다. 실제 조사에 따르면, 청소년의 58.12%가 잘못된 성지식을 가지고 있으며(아하 서울시립 청소년 성문화센터, 2021), 평균 성지식 수준은 10점 만점에 3.73점으로 자기평가 점수(7.27점)와 큰 차이를 보였다(한국여성정책연구원). 이는 청소년이 충분한 성지식을 갖추었다고 생각하지만 실제 수준은 매우 낮음을 시사하며, 안전하고 신뢰할 수 있는 성교육 도구의 필요성을 뚜렷이 보여준다.

(2) Target Customer

본 과제의 주요 대상은 **만 13~18세 청소년(중·고등학생)**이다. 이들은 성적 호기심이 증가하는 시기에 있으나, 부모나 교사에게 직접 질문하기 어려운 환경에 놓여 있으며, 실제로는 인터넷과 친구를 통한 비공식적 채널에 의존하는 경향이 크다. 그러나 이러한 채널은 신뢰성이 낮아 왜곡된 성지식을 습득할 위험이 높다. 따라서 익명성과 안전성을 보장하면서, 연령·성별·관심사에 따라 맞춤형 학습을 제공하는 서비스가 필요하다.

(3) Pain Points

- **정보 접근성의 부족**: 교실 수업은 제한된 시간 안에 기초 지식만 다루며, 민감한 질문을 자유롭게 하기 어려운 환경이다. 청소년들은 부모나 교사에게 직접 질문하는 데 부담을 느끼고, 결국 스스로 인터넷 검색을 시도하게 된다.

- **온라인 정보의 신뢰성 문제**: 한국여성정책연구원(2018)의 조사에 따르면, 성 관련 고민 해결 경로로 “혼자 인터넷이나 책을 통해 찾는다”는 응답이 35.2%, “친구와 상의한다”는 응답이 30.8%로 나타났다. 이는 부모나 교사보다 비공식적 채널에 의존하는 경향이 높으며, 온라인의 왜곡된 정보를 무분별하게 습득할 가능성이 크다.

- **교육 방식의 비효율성**: 기존 성교육 자료는 형식적인 콘텐츠가 대부분이며, 청소년 눈높이에 맞지 않는 경우가 많다. 훈계조의 전달 방식은 오히려 거부감을 유발하고 학습 몰입을 방해한다.
- **개인 맞춤형 학습의 부재**: 청소년마다 이해 수준과 관심사가 다르지만, 현재 교육 체계에서는 개별화된 상담과 학습이 제공되지 않는다. 특히 "성적 동의(consent)"와 같은 실제 상황에서 필요한 주제는 충분히 다루어지지 않고 있다.

## 2.2 기존 연구와의 비교

기존 성교육 연구 및 서비스는 크게 두 가지 한계를 가진다.

- **교과·지침서 중심의 정보 제공**: 대부분의 자료는 학부모나 교사가 청소년을 지도하기 위한 형태로 작성되어 있으며, 학습자가 스스로 질문하고 대화하는 상호작용적 구조가 부족하다.
- **디지털 성교육 서비스의 한계**: 일부 웹 기반 상담이나 챗봇 서비스가 존재하나, 이들은 사전 정의된 FAQ나 단순 키워드 검색에 기반해 맞춤형 응답 생성이 불가능하다. 또한 청소년 친화적 언어가 부족하여 학습자와의 심리적 거리감을 형성한다.

이에 반해 본 프로젝트는 최신 NLP 기술인 **RAG(Retrieval-Augmented Generation)을 적용**하여 신뢰할 수 있는 성교육 문헌을 기반으로 답변을 생성한다. 더 나아가 GPT 기반 후처리를 통해 청소년 눈높이에 맞는 친화적 표현을 제공함으로써, 기존 연구 대비 정확성 · 상호작용성 · 공감성 측면에서 차별성을 확보한다.

## 2.3 제안 내용

본 프로젝트는 청소년의 성지식 격차와 잘못된 정보 노출 문제를 해결하기 위해, 최신 NLP 기술과 교육적 설계를 결합한 성교육 앱을 제안한다. 주요 내용은 다음과 같다.

1. 데이터 확보 및 정제
   정부·국제기구 성교육 가이드라인, 학부모 Q&A 등 신뢰할 수 있는 성교육 자료를 PDF

형태로 수집한다. 이를 OCR과 전처리 과정을 통해 학습 가능한 텍스트로 변환하고, 불필요한 기호 및 중복 표현을 제거하여 고품질 데이터셋을 구축한다.

2. 지식 검색 인프라 및 RAG 파이프라인
   OpenAI Embeddings를 활용하여 문장 단위 임베딩을 수행하고, FAISS와 BM25를 결합한 하이브리드 벡터 데이터베이스를 구성한다. 이를 통해 질문과 가장 관련성이 높은 자료를 고속으로 검색하며, GPT 모델이 검색된 문서를 근거로 응답을 생성하는 RAG(Retrieval-Augmented Generation) 구조를 적용한다.

3. LLM 기반 공감형 대화 기능
   GPT 모델을 활용하여 단순 사실 전달을 넘어, 청소년 눈높이에 맞춘 자연스럽고 공감적인 대화를 제공한다. 프롬프트 엔지니어링을 통해 "10대 친구처럼 친근하고 이해하기 쉬운 말투"로 답변을 후처리하여, 사용자가 안전하면서도 편안하게 질문할 수 있는 경험을 제공한다.

4. 시나리오형 퀴즈 시스템
   실제 청소년이 마주할 수 있는 상황(예: 데이트 중 동의 여부, 온라인 성적 괴롭힘 등)을 기반으로 선택형 문항을 제공한다. 이를 통해 단순한 지식 암기를 넘어, 상황 몰입형 학습과 자기주도적 성찰을 가능하게 하며, 사용자의 성지식 수준을 진단하고 피드백을 제공한다.

5. 모바일 최적화 및 익명성 보장
   React Native 기반으로 직관적이고 접근성 높은 모바일 앱을 구현한다. 회원가입 시 개인정보를 최소화하여 익명성을 보장하며, 민감한 질문도 부담 없이 입력할 수 있도록 익명 질문 기능을 제공한다.

## 2.4 기대 효과 및 의의

본 프로젝트는 신뢰성 있는 성교육 자료를 기반으로 AI 챗봇과 상황 몰입형 학습 기능을 제공하여, 청소년 성교육의 접근성과 효과를 크게 향상시킨다. 현재 청소년의 성지식 오답률이 58% 이상이고 실제 평균 점수가 3.73점에 불과한 상황에서, 본 시스템은 검증된 문헌 기반 응답을 통해 정보의 정확성과 학습 효과를 동시에 보장한다. 또한 기존의 "인터넷 검색(35.2%)"이나 "친구와의 상의(30.8%)"보다 안전하고 공신력 있는 학습 경로를 제공한다.

특히 GPT 기반 후처리를 통해 청소년 눈높이에 맞춘 친구 같은 대화를 구현하여 정서적 공감을 형성하고, 시나리오형 퀴즈를 통해 실제 상황 대응 능력을 강화한다. 결과적으로 본 프로젝트는 청소년의 건강한 성 가치관 형성과 사회적 성숙을 지원하는 데 기여할 수 있다.

**핵심 기대 효과**

- 정확성 확보: 신뢰성 있는 성교육 자료를 근거로 응답 제공
- 정서적 공감: GPT 후처리로 형식적 설명 대신 친구 같은 대화 제공
- 학습 효과: 시나리오형 퀴즈를 통한 실제 상황 대응 능력 강화
- 사회적 의의: 성교육 접근성과 질적 수준 향상 → 건강한 성 가치관 형성

## 2.5 주요 기능 리스트

본 프로젝트에서 제안한 솔루션을 실제 구현하기 위해 다음과 같은 주요 기능을 설계하였다.

1. **데이터 전처리 및 지식 관리 기능**
   - 수집된 성교육 자료(PDF)를 OCR과 전처리 과정을 거쳐 학습 가능한 텍스트로 변환한다.
   - 불필요한 기호 및 중복 표현을 제거하고, Embeddings를 통해 문장 단위 데이터베이스를 구축한다.

2. **AI 챗봇 상담 기능 (RAG 기반)**
   - 사용자가 입력한 질문을 벡터 검색(RAG)으로 처리하여 관련성이 높은 문서를 추출한다.
   - GPT 모델이 해당 문서를 근거로 응답을 생성해, 신뢰성과 정확성을 확보한다.

3. **공감형 대화 기능**
   - 프롬프트 엔지니어링을 통해 GPT 응답을 청소년 눈높이에 맞는 친근한 말투로 후처리한다.
   - 형식적인 설명 대신 정서적 공감을 제공하여 학습 몰입을 강화한다.

4. **시나리오형 퀴즈 기능**
   - 실제 청소년 상황(예: 데이트 중 동의, 온라인 괴롭힘 등)을 반영한 선택형 문제를 제공한다.
   - 응답 결과에 따른 피드백과 학습 가이드를 통해 자기주도적 학습과 문제 해결 능력을 강화한다.

5. **익명 질문 및 게시판 기능**
   - 민감한 고민도 부담 없이 입력할 수 있도록 익명 질문 기능을 제공한다.
   - 또래와 의견을 공유할 수 있는 게시판을 운영하며, Moderation 필터로 유해 콘텐츠를 차단한다.

6. **개인화 학습 및 추천 기능**
   - 연령, 성별, 관심사 기반으로 맞춤형 콘텐츠와 퀴즈를 추천한다.
   - 질문 기록과 퀴즈 응답 내역을 저장해 개별 학습 이력을 관리한다.

7. **포인트/배지 시스템**
   - 퀴즈 참여, 질문 작성, 게시판 활동에 포인트를 부여하고, 성취도에 따라 배지를 지급한다.
   - 이를 통해 학습 동기를 강화하고 지속적 참여를 유도한다.

8. **보안 및 안전성 강화 기능**
   - 개인정보를 최소 수집·보호하며, AI Moderation을 통해 부적절한 질문과 응답을 필터링한다.
   - 안전하고 신뢰할 수 있는 학습 환경을 보장한다.

# 3. Project Design

---

## 3.1 요구사항 정의

(1) 사용자 요구사항

- **자유로운 질문 입력**: 키워드 기반이 아닌 자연어 기반 질문 입력을 지원하며, 민감한 질문도 익명성을 보장해야 한다.
- **신뢰성 있는 답변 제공**: 챗봇은 인터넷 정보가 아니라 검증된 성교육 자료를 근거로 응답해야 한다.
- **친근한 대화체 응답**: 응답은 교과서식 설명이 아니라 청소년 눈높이에 맞춘 공감형 말투로 제공해야 한다.
- **시나리오형 퀴즈 제공**: 실제 상황을 반영한 선택형 문항을 통해 학습자의 성지식 수준을 확인하고 자율 학습을 유도해야 한다.
- **익명 게시판 활용**: 사용자가 고민이나 질문을 자유롭게 공유할 수 있어야 한다.
- **편리한 사용 환경**: 모바일 앱 기반 직관적 UI를 제공해 기술 숙련도가 낮은 사용자도 쉽게 접근 가능해야 한다.

(2) 시스템 요구사항

- **성능**: 질의 입력 후 응답까지 지연 시간은 3초 이내여야 하며, 벡터 검색 및 RAG 파이프라인을 최적화해야 한다.
- **안전성**: 민감하거나 부적절한 질문은 Moderation 필터로 차단 또는 안내해야 한다.
- **확장성**: 새로운 데이터셋, 임베딩 모델, 시나리오 퀴즈를 쉽게 추가할 수 있는 모듈형 구조를 갖추어야 한다.
- **사용성**: 모바일 환경에서 화면 전환이 빠르고 UI가 단순하여 학습 몰입을 방해하지 않아야 한다.
- **데이터 관리**: 질문 내역과 퀴즈 응답은 안전하게 저장·관리되어야 하며, 개인정보는 암호화 및 인증 절차를 통해 보호되어야 한다.

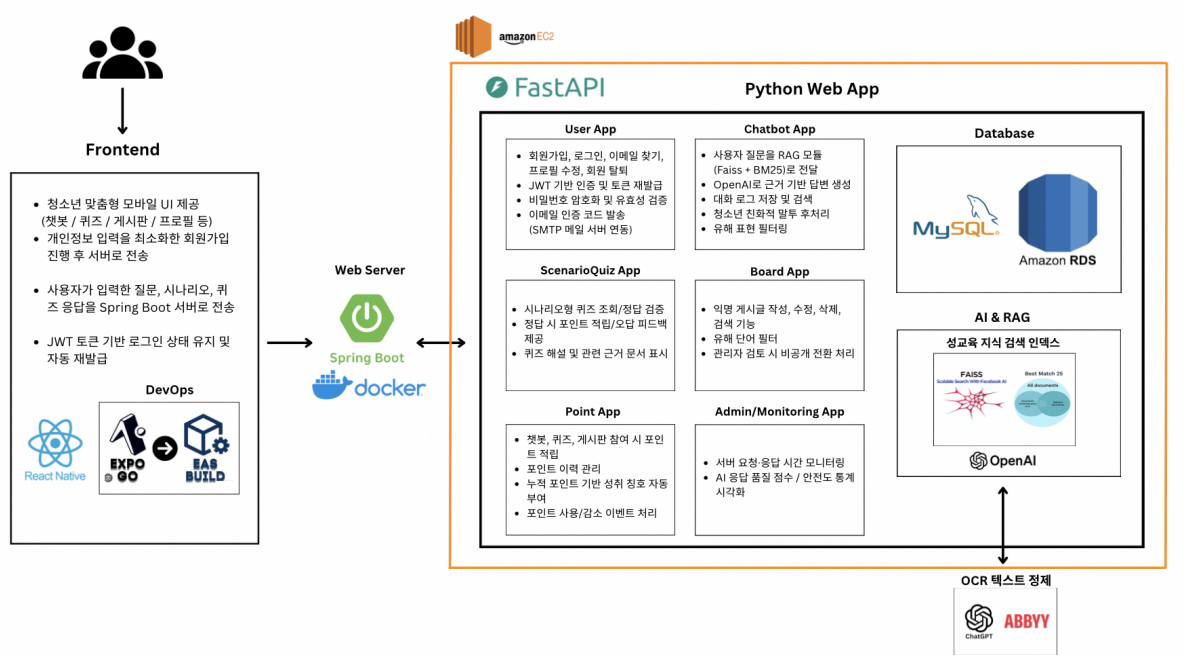
## 3.2 전체 시스템 구성

본 프로젝트의 전체 소프트웨어 구조는 **모바일 프론트엔드**(React Native), **웹 서버**(Spring Boot), **AI 서버**(FastAPI), **데이터베이스**(Amazon RDS MySQL + FAISS,bm25), 그리고 **외부 AI API**(OpenAI, LangChain, ABBYY OCR)로 구성된 **5계층 구조**이다.

아래의 각 구조도는 상이한 관점에서 설계되었다.

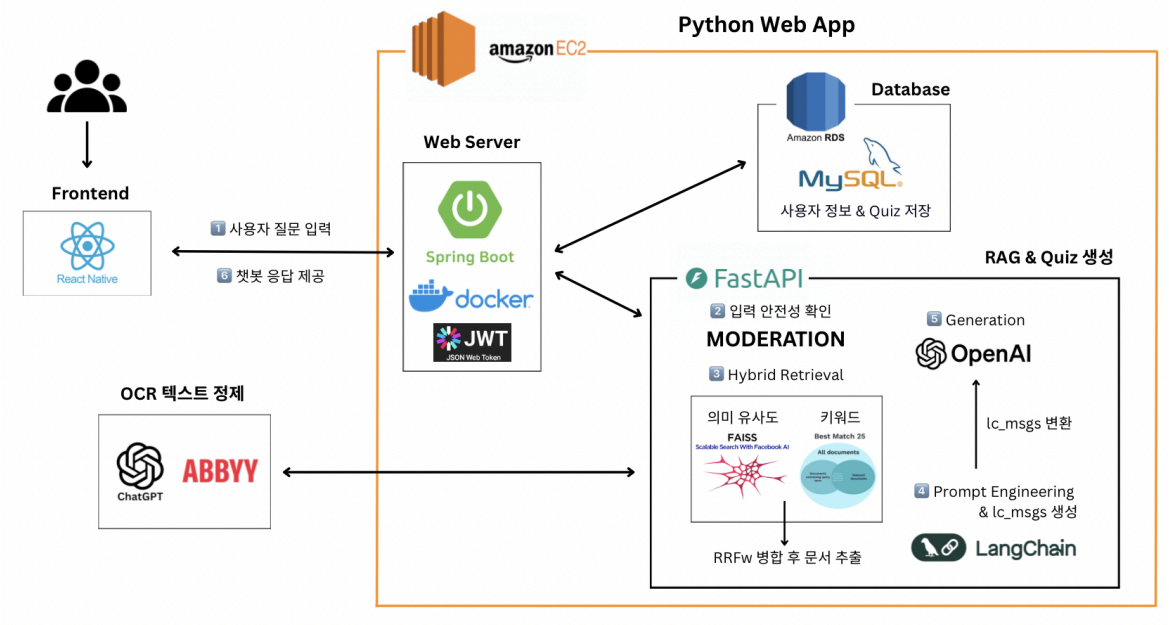
(1) 전체 시스템 구성 관점

: 프론트엔드-백엔드-AI-DB 계층의 책임과 역할을 구분하여 나타낸다.




(2) 데이터 및 제어 흐름 관점

: 사용자의 입력이 FastAPI 모듈을 거쳐 하이브리드 RAG 검색-생성 과정을 거쳐 다시 응답으로 반환되는 과정을 단계별로 설명한다.




1. **사용자(User)**

   최종 사용자(청소년)는 모바일 애플리케이션을 통해 시스템과 상호작용한다. 주요 활동은 자유 질문 입력과 시나리오형 퀴즈 참여이며, 이를 통해 성 지식을 탐색하고 학습한다.

2. **프론트엔드(Frontend)**

   React Native 기반으로 개발된 프론트엔드는 모바일 환경에 최적화된 UI를 제공한다. 주요 화면은 Splash, Home, Chat, Scenario로 구성되며, 사용자가 질문을 입력하거나 퀴즈에 참여하는 과정을 직관적으로 지원한다. 사용자는 JWT 기반 로그인을 수행한다.

   **데이터 흐름:**

   (1) 사용자가 질문을 입력하면 JSON 형식으로 Spring Boot 서버에 전송

   (2) 서버에서 FastAPI AI 모듈 호출 후 응답을 받아 앱 화면에 표시

   (3) 시나리오 퀴즈 결과와 포인트는 DB와 동기화되어 누적 관리

3. **Web Server(Spring Boot)**

   Spring Boot 기반의 백엔드는 프론트엔드·데이터베이스·외부 AI 모듈 간의 API Gateway 역할을 수행한다. REST API로 프론트엔드 요청을 수신하고 FastAPI, DB, RDS 간 데이터 라우팅 담당한다. 주요 기능은 JWT 토큰 기반 사용자 인증·인가 처리 및 세션 유지 관리, 사용자 요청 처리, 벡터 검색 및 LLM 응답 통합, 사용자 인증 및 계정 관리, 질문 로그 및 퀴즈 결과 관리이다. 또한 OpenAI API를 호출해 GPT 기반 응답을 생성하고, FAISS를 활용하여 의미 기반 검색을 지원한다.

**주요 기능:**

(1) UserController: 회원가입/로그인/토큰 재발급 관리

(2) QuizController: FastAPI로부터 생성된 퀴즈를 RDS에 저장

(3) ChatController: 사용자 질문을 AI 서버에 전달 및 응답 반환

## 4. AI Server (FastAPI 기반 Python Web App)

본 AI 서버는 FastAPI 위에 **하이브리드 RAG 검색-생성-검증**을 통합하여 제공한다. 핵심 앱은 다음과 같다.

① **Chatbot App**: 질의응답(Q&A)

② **ScenarioQuiz App**: 퀴즈 생성·검증

③ **Moderation App**: 입력/출력 안전성 보장

### 4-1. Chatbot App

#### 1) 기능 개요

- LangChain 기반 **하이브리드 RAG**로 동작한다.
- **FAISS(의미 검색)** 과 **BM25(키워드 검색)** 결과를 **RRF(Reciprocal Rank Fusion)** 으로 결합해 정확도와 재현율을 동시에 확보한다.

#### 2) 구현 방식

- **FAISS 어댑터, BM25 어댑터**를 도입한다.
- 두 검색기의 결과를 **하이브리드 리트리버**가 통합 조회하고, 순위 결합은 rrf.py에서 수행한다.
- 검색된 컨텍스트는 **응답 생성기**(AnswerGenerator)에서 프롬프트에 주입되어 LLM으로 답변을 생성한다.

- **의미 검색 (Vector Search, FAISS)**
  - 임베딩 모델: text-embedding-3-small
  - 인덱스: FAISS (물리적 벡터 공간)
  - 탐색: **L2 distance 기반 최근접 이웃 탐색**
  - 반환: {"text","score","id","meta"} 표준화 레코드
  - **절차**
    1. 질의 q → 임베딩 벡터 변환
    2. FAISS Top-k 검색
    3. 결과 표준화·스코어 부여

- **키워드 검색 (BM25 + Rank Fusion)**
  - 여러 BM25 인덱스를 병렬 로드하여 스코어링
  - 점수식: **Okapi BM25( TF, IDF, 길이 보정 )**
  - 결합: rrf.py에서 **1 / (K + rank)** 로 FAISS/BM25 결과를 합산하여 최종 순위 산출
  - **절차**
    1. 질의 토큰화 → 각 BM25 인덱스 스코어링
    2. FAISS/BM25 랭크 리스트 생성
    3. RRF로 결합 점수 계산 → 상위 n개 정렬

- **응답 생성 (AnswerGenerator)**
  - 모델: **gpt-4o-mini**, temperature ≤ 0.2
  - 입력: 하이브리드 상위 스니펫을 길이 한도 내로 압축·정제한 컨텍스트
  - 출력: 근거 기반 답변(필요 시 인용 메타 포함)

### 3) 프롬프트 엔지니어링

- 템플릿 소스: prompts.py, prompts_lc.py
- **PromptComposer**가 LLM 메시지 구조를 자동 조립한다.
  - **System**: 교육 목적, 연령 적합, 근거 우선, 부적절 표현 금지
  - **User**: RAG FACTS + 사용자 질문
  - **조립 절차**: FACTS 블록 생성 → 질문 삽입 → System+User+Grounding 순 결합 → LangChain 메시지 시퀀스 생성
- **ToneRephraser**: 1차 결과를 **친근·공감형 반말체**로 재구성(TONE_PRESETS 적용, 의미 불변성 확인).

## 4-2. ScenarioQuiz App

### 1) 기능 개요

- **RAG FACTS**를 근거로 **상황형/개념형** 퀴즈를 자동 생성하고, 형식·정답성을 검증한다.

### 2) 구현 방식

- 프롬프트 템플릿: SITUATION_PROMPT, CONCEPT_PROMPT(from prompts.py)
- 컨텍스트: 하이브리드 검색 결과 상위 FACTS

- **퀴즈 생성(QuizGenerator)**

- **상황형**: 실제 사례·대처 중심 4지선다
- **개념형**: 정의·용어 중심 정형 문항
- **절차**
  1. FACTS 수집 → 유형 결정
  2. 정답 위치 랜덤화
  3. gpt-4o-mini 호출로 문항·보기·해설 생성
  4. 결과를 표준 JSON으로 정리

- **검증(QuizValidator)**
  - 필수 필드(문항·정답·해설) 존재 여부, 정답 중복, 보기 분포 등을 점검
  - **부적합 시 자동 재생성 루프**로 품질 보장

### 4-3. Moderation App

#### 1) 기능 개요

- 사용자 입력과 모델 출력을 **이중으로 검열**하여 안전성을 보장한다.

#### 2) 구현 방식

- **외부 검수**: OpenAI Moderation API (moderation_public.py)
- **내부 검수**: 커스텀 가드 엔진(guard_engine.py, moderation_internal.py) - Regex/키워드/카테고리 스코어링
- **통합**: 두 결과를 병합하여 최종 판정(moderation.py)

- **처리 절차**
  1. 텍스트 입력 → 외부 Moderation 호출과 내부 필터 **동시 실행**
  2. 위험 카테고리 스코어 결합 및 임계치 판정
  3. 위험 시 **안전 안내 응답**으로 대체, 정상 시 파이프라인 지속

## 5. 데이터베이스(Database)

데이터베이스는 MySQL과 FAISS로 구성된다. MySQL은 사용자 계정, 질문 기록, 퀴즈 응답 결과를 저장·관리하고, FAISS는 임베딩된 문서 벡터를 저장하여 고속 유사도 검색을 수행한다. 이를 통해 안정적인 데이터 관리와 효율적인 정보 검색을 동시에 실현한다.

- **MySQL**: 사용자/세션/퀴즈/포인트·배지/로그 등 **구조화 데이터** 저장
- **FAISS Index**: 문서 임베딩 벡터 **비구조 데이터** 저장 및 고속 유사도 검색

종합적으로, 본 시스템은 사용자 입력을 프론트엔드에서 수집하고, 백엔드가 이를 인증 및 처리하여 OpenAI와 FAISS 모듈과 상호작용한다. 이후 결과는 MySQL과 연동되어 사용자에게 반환되는 **엔드투엔드 파이프라인 구조**를 갖춘다.

**엔드투엔드 흐름**

1. 사용자 입력 → FastAPI
2. 하이브리드 검색(FAISS+BM25) → **RRF 결합**
3. 프롬프트 조립 → **gpt-4o-mini** 생성
4. Moderation 통과 → 응답 및 메타 반환
5. 결과·로그를 **MySQL**에 저장, 필요한 인용은 **FAISS 메타**로 추적

**전체 시스템 구성 한줄 요약**

- 질의응답을 위해 **FAISS(의미)** 와 **BM25(키워드)** 를 병행하고 **RRF**로 결합한다.
- 결합 컨텍스트를 **프롬프트 엔지니어링**으로 안정화해 gpt-4o-mini로 생성하며,
- **이중 Moderation**으로 안전성을 보장한다.

## 3.3 주요 엔진 및 기능 설계

### 3.3.0 Hybrid LangChain RAG 구조

초기 개발 단계에서는 **직접 구현한 RAG 구조(Custom RAG Pipeline)** 를 사용하였다. 이 구조는 문서 임베딩(FAISS)과 키워드 검색(BM25)을 개별적으로 수행하고, 결과를 수동으로 결합하여 OpenAI 모델에 프롬프트로 전달하는 방식이었다. 그러나 다음과 같은 문제점이 발생하였다.

1. **모듈 간 결합도 증가 및 유지보수 어려움**
   검색기, 프롬프트, 생성기를 별도로 관리해야 하여, 코드 수정 시 전체 파이프라인을 반복적으로 갱신해야 했다.

2. **검색·생성 단계 간 데이터 일관성 부족**
   FAISS와 BM25 결과의 형식, 스코어 기준, 순위 결합 규칙이 서로 달라 응답 품질이 요청마다 달라지는 현상이 있었다.

3. **확장성 제약 및 안전성 체계 부재**

자체 구조에서는 Moderation(검열) 체인이나 Tone 조정 체인을 유연하게 붙이기 어려워, 청소년 대상 콘텐츠 서비스의 신뢰성 확보에 한계가 있었다.

이러한 한계를 해결하기 위해, 본 프로젝트는 **LangChain 프레임워크 기반의 하이브리드 RAG 구조**로 전환하였다. LangChain은 Retrieval, Prompt, LLM, Moderation 등 각 단계를 체인(Chain) 형태로 모듈화하여 '검색-생성-검증' 과정을 일관되게 관리할 수 있게 해준다. 이에 따라 유지보수성이 향상되고, 프롬프트 관리 및 안전성 제어가 체계화되었다.

또한 기존 RAG의 단일 검색 방식을 개선하여, **FAISS(의미 검색)** 과 **BM25(키워드 검색)** 의 결과를**RRFw(Reciprocal Rank Fusion Weight)** 으로 융합하는 **하이브리드 검색 구조**를 적용하였다. 이를 통해 정확한 키워드 일치와 의미적 유사성 이해를 동시에 확보할 수 있었다.

결과적으로 직접 구현한 RAG 구조는 관리 효율성과 검색 일관성의 한계를 가졌으나, LangChain 기반 하이브리드 RAG 구조는 체계적 모듈화, 품질 안정성, 안전성 측면에서 더 높은 완성도의 시스템을 구현할 수 있게 했다.

### 3.3.1 하이브리드 검색 엔진(Hybrid Retriever)

#### 구성 요소

- **FAISS 어댑터**: 벡터 인덱스 검색(L2)
- **BM25 어댑터**: 역색인 기반 Okapi BM25 스코어링
- **RRFw 결합기**: score += 1/(K+rank)로 결합, 임계값 이하 컷오프
- **스니펫 컴포저**: 길이 한도/문장 경계 유지, 출처 메타 부착

#### 핵심 알고리즘

```
q_embed = embed(q)
R_faiss = topk_faiss(q_embed)
R_bm25  = topk_bm25(tokenize(q))

for result in (R_faiss ∪ R_bm25):
    rrf_score = Σ (1/(K + rank_i))
ranked = sort_by(rrf_score)
context = compact(ranked[:n], max_chars)
```

#### 설계 포인트

- 의미 일반화(FAISS) + 정확 키워드(BM25)의 **상보성 확보**
- RRF로 특정 검색기의 편향을 최소화하여 **안정적 랭킹** 유지
- 결과 레코드 스키마 통일(text, score, id, meta)로 **상위 레이어 단순화**

### 3.3.2 프롬프트 엔진(PromptComposer / ToneRephraser)

#### 구성 요소

- **System 규칙 블록**: 교육 목적, 연령 적합, 근거 우선, 금지 표현 포함
- **User 블록**: 질문 + FACTS(출처 메타 포함)
- **Tone 프리셋**: 친근·공감형 반말체

#### 조립 규칙

- FACTS는 항상 System보다 아래 단계에 위치하여 모델이 근거로 인식
- 길이 초과 시 FACTS를 우선 압축(핵심 문장 유지, 수치·기관명 보존)
- 생성 후 ToneRephraser 2차 호출(의미 동일성 검사 통과 시만 적용)

### 3.3.3 응답 생성기(AnswerGenerator)

#### 모델 및 파라미터

- 모델: gpt-4o-mini
- 온도: ≤ 0.2, Top-p: 기본값, 최대 토큰: 서비스 한도 내 설정

#### 출력 정책

- 확실한 부분만 서술 + 모르면 '모른다' 명시
- 근거 인용(문서ID/청크ID) 가능, 민감 항목은 완곡 표현 또는 안내 문장으로 대체

### 3.3.4 시나리오/개념 퀴즈 엔진(QuizGenerator / Validator)

#### 템플릿

- **SITUATION_PROMPT**: 상황 제시 → 선택지 4개(정답 1) → 근거 해설
- **CONCEPT_PROMPT**: 정의·원리 질문형 → 선택지 4개 → 근거 해설

#### 생성 절차

1. 하이브리드 검색으로 FACTS 수집(주제/키워드/학년(나이) 기준)
2. 정답 위치 랜덤화(편향 방지)
3. LLM(gpt-4o-mini) 생성 → Validator 스키마 검증 → 실패 시 자동 재생성

#### 검증 규칙(핵심)

- 필수 필드 존재(문항/정답/해설)
- 오답 매력도 규칙: 근거 대비 그럴듯하지만 핵심 포인트에서 틀리도록 구성
- 동일/중복 보기 금지, 부적절 표현 필터링

### 3.3.5 Moderation 엔진(외부 + 내부 이중화)

#### 구성

- **External**: OpenAI Moderation (카테고리/점수 기반 검수)
- **Internal**: guard_engine.py (Regex/키워드/나이 민감 정책 맵)
- **Aggregator**: 위험 레벨 병합, 임계값 정책 적용

#### 판정 정책

- 우선순위: 차단 > 대체 > 경고 > 통과
- 대체 응답 템플릿: 도움 자원·상담 안내·사실 정보 위주

### 3.3.6 오케스트레이션 및 신뢰성

#### 요청 제어

- Spring Boot: 타임아웃·재시도(지수 백오프), 표준 에러 스키마 처리
- FastAPI: 서킷 브레이커(옵션), 슬로우 쿼리 로깅

#### 인덱스 운영

- 버전 메타, 무결성 해시, 백업/롤백 스크립트 관리
- 기동 시 헬스체크(FAISS/BM25 로드 상태, 문서 카운트, 샘플 질의) 자동 점검

### 3.3.7 데이터 스키마(요점)

- **퀴즈**:
  - quiz_scenario(id, title, created_at, status)
  - quiz_question(id, stem, scenario_id)
  - quiz_option(id, question_id, label, text, is_correct)
- **대화**:
  - chat_session, chat_message(user/assistant, source_meta)
- **포인트/배지**:
  - 이벤트 트리거 기반 누적 테이블

본 프로젝트의 AI 서버는 **LangChain 기반 Hybrid RAG 구조**를 통해 의미 검색(FAISS) + 키워드 검색(BM25)을 RRFw로 융합하고, 프롬프트·생성·검증·안전성 체인을 통합하여 **정확성·안전성·확장성**을 모두 갖춘 청소년 맞춤형 AI 성교육 시스템을 구현하였다.

## 3.4 주요 기능의 구현

### 3.4.1 Q&A 챗봇 기능 (Hybrid RAG → LLM → Moderation)

**Flow**

1. 사용자가 질문 입력(React Native) → Spring Boot 게이트웨이로 전달(JWT 검증)
2. FastAPI /chat 호출 → 입력 **사전 Moderation**(내부 Regex + OpenAI Moderation)
3. **하이브리드 검색**:
   - FAISS 의미 검색(임베딩 쿼리 → Top-k)
   - BM25 키워드 검색(토큰화 → Top-k)
   - RRF로 순위 결합 → 상위 n 스니펫 선택(길이 한도 내 압축/정제)
4. **프롬프트 조립**: System(교육·안전 규칙) + User(질문) + FACTS(근거)
5. **생성**: gpt-4o-mini(temperature ≤ 0.2)로 근거 기반 답변 생성
6. **사후 Moderation** → 안전 안내로 대체/완곡 처리
7. 응답·근거 메타(문서ID/청크ID/스코어) 및 로그를 MySQL에 기록, FE에 반환

**예외/폴백**

- 하이브리드 결과 0건 → BM25 only 재시도 → 그래도 0건이면 "모름 + 탐색 경로 안내" 템플릿 반환
- LLM 타임아웃 → 재시도(지수 백오프) 후 실패 시 안전 폴백 응답

### 3.4.2 시나리오/개념 퀴즈 자동 생성 기능

**Flow**

1. 사용자/관리자 요청 → Spring Boot → FastAPI /quiz/generate
2. 하이브리드 검색으로 **FACTS** 수집(주제/키워드 기준)
3. 템플릿 선택: SITUATION_PROMPT 또는 CONCEPT_PROMPT
4. 정답 위치 랜덤화 → gpt-4o-mini 호출(문항·선지·해설)
5. **검증**: JSON 스키마/정답 단일성/오답 분포 확인 → 실패 시 자동 재생성 루프
6. 결과 저장: quiz_scenario, quiz_question, quiz_option(RDS)
7. FE 시나리오 화면에 배포(미리보기/승인 플래그 지원)

**예외/폴백**

- FACTS 부족 → Top-k 확장 or 주제 유사어 재검색
- 검증 실패 3회 이상 → "인적 검토 필요" 상태로 저장(비공개)

### 3.4.3 안전성·콘텐츠 검증 기능(Moderation)

**Flow**

1. 입력 단계: 내부 가드(Regex/키워드/나이민감) + OpenAI Moderation 병렬 실행
2. 생성 단계: 동일 이중 검사 → 위험 레벨 통합
3. 위험 시: 차단 또는 **안전 대체 응답**(도움 자원/상담 안내/사실 정보 위주)
4. 모든 판정/카테고리 스코어 MySQL 로깅(감사 추적)

## 3.5 기타

**<Hybrid LangChain RAG 구조>**

**1) 전체 아키텍처**

- **데이터 레이어**
  - 문서 정제 → 청크 분할 → 임베딩 → **FAISS 벡터 인덱스** + 메타(meta.json) 생성
  - 동일 문서셋으로 **BM25 인덱스**도 구축(bm25_store.py)
- **검색 레이어**
  - 질문 q에 대해 **FAISS(의미)**, **BM25(키워드)** 각각 Top-k 검색 → **RRFw(Reciprocal Rank Fusion weight)** 로 재정렬
- **생성 레이어**
  - 1차: **SYSTEM 프롬프트**로 사실 기반 답변 생성
  - 2차: **친근한 말투로 후처리**
  - 생성물은 **Moderation 체인**(금칙어/부적절 표현 필터) 통과
- **서빙 레이어**
  - FastAPI(AI) ↔ Spring Boot(BE) ↔ React Native 앱(FE)
  - 필요시 소스 스니펫/인용 정보 함께 반환

**2) 물리 결합 FAISS + 논리 결합 BM25**

- **FAISS (물리 결합, Vector Store)**
  - 임베딩 벡터를 **물리적으로** 인덱스 파일(index.faiss)에 저장하고, 검색 시 벡터 유사도(코사인/IP 또는 L2)로 직접 질의
  - **FAISS**는 *임베딩이라는 물리적 벡터 공간*에서 **최근접 탐색**을 수행
  - 특징: 고차원 의미 검색, 장점은 **의미적 동의어·재표현**을 잘 잡는다.
  - 구현: faiss_store.py

        - 빌드/저장: build_empty_index() → add_to_index() → save_index()
        - 로드/검색: FaissStore.load(...) → .search(query, top_k)

- **BM25 (논리 결합, Inverted Index)**
    - 원문 토큰을 바탕으로 **역색인/빈도(DF, TF)** 를 계산해 **점수(Okapi BM25)** 로 랭킹
    - **BM25**는 원문 토큰과 **논리적 매칭 규칙**(TF-IDF/길이 보정)으로 점수화
    - 특징: **정확한 키워드 매칭**, 숫자·고유명사·전문용어에 강함
    - 구현: bm25_store.py
        - 간단 토크나이저(한/영/숫자), BM25.topk()로 스코어링
        - .search(query, top_k)가 표준화된 레코드(text, score, id, meta)를 반환

### 3) 하이브리드 검색 & RRF 결합

- generator._faiss_search(q, k) + _bm25_search(q, k) 결과를 합쳐 **RRF 스코어**를 계산해 단일 랭킹으로 통합
- **RRF 핵심: 각 검색기 랭크 r에 대해 점수 1 / (k + r)을 합산해 편향을 줄이고 두 검색기의 강점을 동시에 살림**
- 임계치 threshold가 설정되면 _rrf가 낮은 스니펫은 컷오프(잡음 제거) 수행
- 검색 실패 시 폴백: “하이브리드 0건 → BM25 only” 로 다시 시도

### 4) 컨텍스트 빌드 & 안전화

- 상위 스니펫을 **문자 수 한도**를 두고 붙여서 컨텍스트 생성
- 민감 표현/연령 부적합 표현은 **sanitize** 처리 후 프롬프트에 투입
- LangChain은  prompts_lc.py에서 Document 리스트를 받아 **문맥 문자열 + citation 메타**를 동시에 산출

### 5) 프롬프트 구조

#### A. 챗봇 RAG (일반 Q&A)

- **System 프롬프트(고정 규칙)**
    - 역할: 교육/보건 목적, 연령 적합, 사실 우선, 모르면 ‘모른다’ + 안전한 대안 제시등의 규칙을 명시
    - 톤/안전/근거 지시가 통합된 **S1_SYSTEM_PROMPT** 사용
- **User 메시지**
    - “검색된 컨텍스트 + 질문 + 요구사항 을 하나로 전달(근거 기반 답변 유도)
- **Assistant 생성**
    - model=gpt-4o-mini, temperature=0.0~0.2로 **일관·사실 중심** 답변

- **친근한 말투로 후처리**
  - 2차 프롬프트로 말투를 친근반말 등 프리셋에 맞춰 재가공
- **Moderation**
  - OpenAI Moderation + 커스텀 키워드 필터로 부적절 표현 제거/완곡 대체

**B. 퀴즈 생성 RAG**

- **컨텍스트**: RAG로 선정된 **사실**만 주입
- **프롬프트 세트(prompts.py)**
  - SITUATION_PROMPT: 상황형(연령 적합 가이드, 문제유형 템플릿, 정답 랜덤 위치 규칙 포함)
  - CONCEPT_PROMPT: 개념형("옳은 것/틀린 것/적절한 것…"의 **질문형식 리스트**와 함께 사용)
  - get_prompt(qtype, keyword, topic, context, correct_position, question_type)로 **유형별 템플릿 완성**
- **출력 규격**
  - 문제 본문, 선택지 4개(정답 1개 + 오답 패턴 3개), **해설 필수**, 정답 위치 랜덤화

**C. LangChain 스타일 프롬프트(prompts_lc.py)**

- **TONE_PRESETS**(예: 친근한) + **안전/가이드 블록** + **컨텍스트/질문 슬롯**을 합친 **ChatPromptTemplate**
- 입력: List[Document] → 내부에서 길이 제한을 적용하여 **컨텍스트 문자열**과 **citation 메타** 동시 구성

**6) 사용 모델(환경 변수와 기본값)**

- **생성 모델**: gpt-4o-mini
  - 코드 기본값: SCSC_CHAT_MODEL 미설정 시 gpt-4o-mini 사용(generator.py)
  - 이유: **고품질 생성 + 낮은 지연**의 균형(시나리오/해설/대화 모두 안정)
- **임베딩 모델**: text-embedding-3-small
  - 코드 기본값: EMBED_MODEL 또는 SCSC_EMBED_MODEL 미설정 시 해당 모델
  - 이유: **경량·저비용** 대비 우수한 의미 유사도 성능(대규모 인덱스 운영에 적합)
- **재랭크/모더레이션**: OpenAI API 사용

### 7) 요청-응답 흐름(end-to-end)

1. 사용자가 질문 입력 → BE가 AI 서버 /chat 호출
2. AI 서버:
   - **FAISS + BM25** 검색 → **RRF** 결합 → LLM 재랭크
   - 스니펫 압축/클린 → **SYSTEM 프롬프트** 조합 → **gpt-4o-mini**로 답변 생성
   - 친근한 말투로 후처리 → Moderation 필터
3. 응답과 함께 **근거 스니펫 메타**(source, chunk_id, rrf) 반환
4. BE가 로깅/정책 처리 후 FE로 전달

### 8) 설계 포인트

- **하이브리드 = 정확도 + 재현율**: BM25(숫자/고유명사 강점)와 FAISS(의미 일반화)를 **RRF**로 안정 결합 → 고정 파라미터로도 일관 성능
- **프롬프트 분리**: 규칙·톤·근거를 **System**에 고정, 컨텍스트·질문을 **User**로 분리 → 모델 교체/튜닝 시에도 안전성 유지
- **두 단계 생성**: 1차(정확) + 2차(말투) 분리로 **품질·일관성·안전**을 동시에 달성
- **환경 변수화**: SCSC_CHAT_MODEL, SCSC_EMBED_MODEL, .env로 **운영비/성능**을 배포 환경별로 조절 가능