프로젝트 기획서: EXERCISE COACH (EC)

VLM 기반 실시간 AI 운동 코칭 솔루션

프로젝트 기간: 2025-12-02 ~ 2025-12-12 **문서 작성일:** 2025-12-05

1. 프로젝트 정의

1.1. 프로젝트명

EXERCISE COACH (EC)

1.2. 프로젝트 목표

본 프로젝트는 사용자가 웹캠을 통해 자신의 운동 자세를 실시간으로 분석하고, AI 기반의 개인화된 피드백을 받아 올바른 자세로 운동할 수 있도록 돕는 웹 애플리케이션을 개발하는 것을 목표로 합니다. 사용자의 운동 경험을 향상시키고 부상 위험을 줄이며, 운동 효과를 극대화하는 지능형 코칭 시스템 구축을 지향합니다.

1.3. 핵심 기능 및 특징

- ✓ **웹 기반 실시간 AI 코칭**

  FastAPI 백엔드 서버와 React 프론트엔드 UI를 결합하여 웹 브라우저에서 모든 기능이 동작하는 실시간 코칭 환경을 제공합니다. 별도의 프로그램 설치 없이 접근성을 극대화합니다.

- ✓ **정교한 모션 인식 및 분석**

Google의 MediaPipe Pose Landmarker를 활용하여 웹캠 영상에서 사용자의 관절 위치를 실시간으로 추적합니다. 스쿼트, 숄더프레스 등 특정 운동에 최적화된 상태 머신(State Machine)을 통해 정확한 횟수 카운팅 및 자세 분석을 수행합니다.

✓ 사용자 친화적 UX/UI

운동별 튜토리얼 영상을 YouTube API로 자동 검색하여 제공하고, AI가 분석한 자세 교정 피드백을 텍스트와 음성(TTS)으로 동시에 전달하여 사용자가 직관적으로 이해하고 즉각적으로 자세를 교정할 수 있도록 지원합니다.

✓ 체계적인 데이터 관리 파이프라인

사용자의 운동 기록(횟수, 시간)과 주요 자세의 캡처 영상(.webm)을 로컬 데이터베이스(SQLite) 및 파일 시스템에 자동으로 저장하여, 사용자가 자신의 운동 이력을 관리하고 성과를 추적할 수 있도록 합니다.

2. 팀 구성 및 역할

본 프로젝트는 총 4명의 팀원으로 구성되었으며, 각자의 전문 분야에 따라 역할을 분담하여 효율적인 협업을 진행했습니다.

구분	김채원	손지원	이주연	이진배
사진				
역할	팀장, PM	Frontend 개발	Backend 개발	Backend 개발
담당 모듈	데이터베이스(DB) 설계 및 관리	웹 UI/UX (React) 개발	컴퓨터 비전(CV), FastAPI 서버 구축	LLM 연동 및 Q&A, YouTube API 연동

3. 프로젝트 일정 및 WBS

3.1. 전체 일정 계획

프로젝트는 2025년 12월 2일부터 12월 12일까지 총 11일간 진행되었습니다. 각 단계별 주요 작업 항목과 일정은 다음과 같습니다.

작업 항목	시작 날짜	종료 날짜	기간 (일)	진행 상태	타임라인
아이디어 회의 및 기획	2025-12-02	2025-12-04	3	완료	
요구사항 정의 및 아키텍처 설계	2025-12-05	2025-12-05	1	진행중	
백엔드 REST API 개발 (exercise_server)	2025-12-06	2025-12-08	3		
프론트엔드 UI 및 Mediapipe 통합	2025-12-09	2025-12-10	2		
LLM/TTS 연동 및 통합 테스트	2025-12-11	2025-12-11	1		
프로젝트 최종 발표	2025-12-12	2025-12-12	1		

3.2. 작업 분할 구조 (WBS - Technical Detail)

프로젝트의 핵심 기능 구현을 위한 기술적 작업 분할 내역입니다.

📸 CV & 실시간 처리

- ✓ **Pose Detection**: MediaPipe `TasksVision` 라이브러리(Wasm)를 로드하여 실시간 웹캠 스트림에서 사용자의 자세를 분석합니다.
- ✓ **좌우 반전(Mirroring)**: 사용자가 거울을 보는 것처럼 느낄 수 있도록 비디오 화면과 그 위에 그려지는 Canvas 오버레이를 좌우 반전 처리하여 직관성을 높입니다.
- ✓ **운동별 로직 구현**:
  - ✓ **스쿼트(Squat)**: 힙-무릎-발목 관절의 각도를 계산하여 자세를 판단합니다. `checkFullBodyVisibility` 함수를 통해 하체 랜드마크가 모두 보이는지 확인하며, 무릎 각도가 임계값(`SQUAT_THRESHOLD`: 110도) 미만으로 내려갔을 때 최하단 자세로 인식하고 이미지를 캡처합니다.
  - ✓ **숄더프레스(Shoulder Press)**: 어깨-팔꿈치-손목 관절의 각도를 계산합니다. 상체(어깨/골반) 및 팔(팔꿈치/손목) 랜드마크가 모두 인식되어야 동작하며, 팔꿈치 각도가 150도 이상으로 펴진 `UP` 상태에서 최상단 자세를 캡처합니다.
- ✓ **이미지 전송 및 저장**: 각 운동 로직에 따라 캡처된 이미지는 분석을 위해 백엔드 REST API(`/analyze-image`)로 전송되며, 프론트엔드 화면에도 오버레이로 표시됩니다.

💬 LLM 코칭 & 리포트

- ✓ **운동별 피드백 생성**: 프론트엔드에서 전송된 Base64 인코딩 이미지, 운동 종류, 현재 Rep 횟수 정보를 받아 GPT-4o-mini Vision 모델을 통해 자세 분석 및 피드백을 생성합니다.
- ✓ **Prompt Engineering**: 스쿼트의 무릎 각도, 숄더프레스의 팔꿈치 벌어짐 등 운동별 핵심 체크포인트를 System Prompt에 명시하여, 모델이 10자 내외의 짧고 직관적인 한국어 피드백을 생성하도록 유도합니다.
- ✓ **오류 처리**: OpenAI API 호출 실패 또는 분석 불가 시, 사전에 정의된 기본 안내 메시지(Fallback Message)를 사용자에게 제공하여 서비스 안정성을 확보합니다.

🗣️ TTS (Text-to-Speech)

- ✓ **Web Speech API 활용**: 브라우저에 내장된 `window.speechSynthesis` API를 사용하여 별도의 서버 없이 클라이언트 측에서 텍스트 피드백을 음성으로 변환합니다.

- ✓ **사용자 경험 최적화**: 동일한 피드백이 반복적으로 발화되는 것을 방지하기 위해 `lastSpokenRef`로 마지막 발화 내용을 추적하고, 4초의 쿨다운(Cooldown)을 적용하여 과도한 음성 안내를 제어합니다.

🎬 튜토리얼/검색

- ✓ **YouTube Data API 연동**: 백엔드의 `/search-youtube` 엔드포인트를 통해 사용자가 선택한 운동 종목에 맞는 튜토리얼 영상을 자동으로 검색하고, 결과를 프론트엔드 iframe에 임베드하여 보여줍니다.
- ✓ **Fallback 메커니즘**: API Quota 초과 또는 API 키 누락과 같은 예외 상황 발생 시, 사전에 정의된 기본 튜토리얼 영상(`FALLBACK_YT`)을 재생합니다. 또한, 사용자가 직접 YouTube 영상 ID나 링크를 입력할 수 있는 기능도 제공합니다.

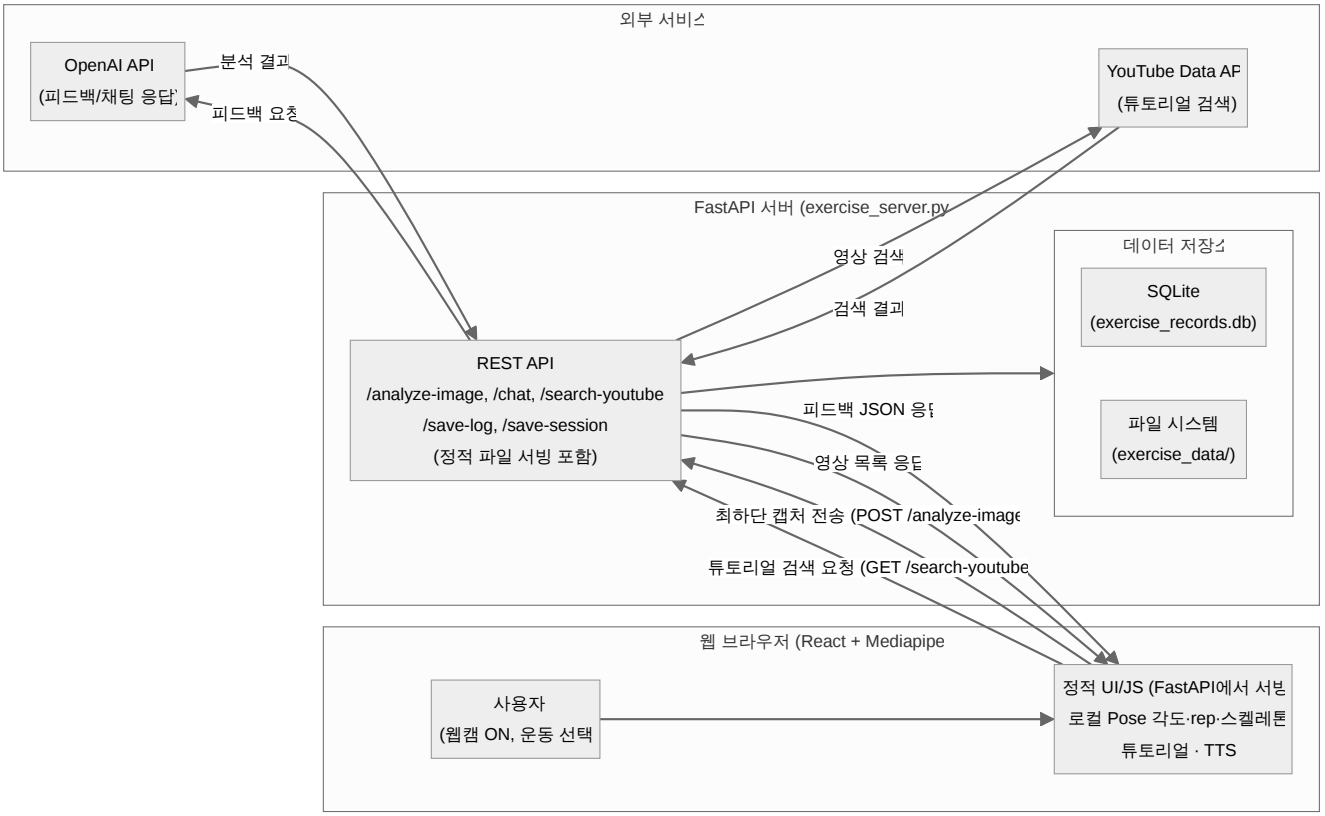
💻 프론트엔드 UX/UI

- ✓ **React (Vanilla Style)**: 복잡한 빌드 과정 없이 `index.html`에 CDN으로 React와 Babel을 로드하는 방식을 채택했습니다. 이를 통해 빠른 프로토타이핑과 수정 용이성을 극대화했습니다.
- ✓ **효율적인 상태 관리**: `useState`와 `useRef` Hook을 중심으로 상태를 관리하며, 특히 `requestAnimationFrame`을 사용한 Canvas 드로잉 루프에서는 불필요한 리렌더링을 최소화하여 성능을 최적화했습니다.
- ✓ **직관적인 UI 구성**: 분석 뷰, 튜토리얼 뷰, LLM 채팅 뷰를 명확히 구분하고, 운동 종목 변경 시 관련 상태(카운트, 로그 등)를 초기화하는 로직을 구현했습니다. API 키 부재나 통신 실패 시 사용자에게 명확한 안내 메시지를 표시합니다.

4. 시스템 설계

4.1. 시스템 아키텍처

본 시스템은 클라이언트(웹 브라우저)와 서버(FastAPI)가 명확히 분리된 구조를 가집니다. 클라이언트는 실시간 영상 처리와 UI를 담당하고, 서버는 외부 API 연동 및 데이터 처리를 담당하여 역할을 분담합니다.



아키텍처 흐름:

1. **사용자 상호작용**: 사용자가 브라우저에서 웹캠을 켜고 운동을 선택합니다.
2. **프론트엔드 (클라이언트)**: FastAPI 서버로부터 정적 파일(HTML, JS, CSS)을 받아 UI를 렌더링합니다. 로컬에서 MediaPipe를 이용해 웹캠 스트림을 분석하고, 관절 각도 계산, Rep 카운팅, 스켈레톤 그리기를 수행합니다.
3. **API 요청**: 운동 중 특정 조건(예: 스쿼트 최하단)이 충족되면, 캡처된 이미지를 백엔드 FastAPI 서버의 `/analyze-image` 엔드포인트로 전송합니다.
4. **백엔드 (서버)**: FastAPI 서버는 요청을 받아 OpenAI Vision API로 이미지를 전달하여 분석을 요청합니다.
5. **외부 API 연동**: OpenAI API는 이미지 분석 후 자세에 대한 피드백 텍스트를 생성하여 FastAPI 서버로 반환합니다. YouTube API 연동도 유사한 방식으로 서버를 통해 이루어집니다.

6. **응답 및 UI 업데이트**: 서버는 받은 피드백을 JSON 형태로 프론트엔드에 응답합니다. 프론트엔드는 이 데이터를 받아 화면에 텍스트로 표시하고, Web Speech API를 통해 음성으로 출력합니다.
7. **데이터 저장**: 운동 기록은 서버의 `/save-log` 등의 엔드포인트를 통해 SQLite DB 및 로컬 파일 시스템에 저장됩니다.

4.2. 기술 스택 (Technology Stack)

프로젝트 구현에 사용된 주요 기술 및 라이브러리는 다음과 같습니다.

Frontend

Backend

CV / 분석

운영 / 도구

4.3. 프로젝트 폴더 구조

프로젝트의 소스 코드는 다음과 같은 구조로 구성되어 있습니다.

```
ai01-2nd-2team-KSLNOVA/
├── README.md                    # 프로젝트 개요 및 문서
├── HOWTORUN.md                  # 상세 실행 가이드
├── requirements.txt             # 백엔드 Python 의존성 파일
├── .gitignore
├── frontend/
│   ├── index.html               # React CDN을 사용하는 정적 엔트리 파일
│   ├── env.js                   # .env 파일 기반으로 생성되는 프론트엔드 런
│   ├── .env                     # 프론트엔드용 API 키/엔드포인트 (Git 추적
│   ├── generate_env_js.py       # .env 파일을 env.js로 변환하는 스크립트
│   └── src/
│       ├── App.jsx              # 메인 UI 컴포넌트 및 핵심 로직
│       ├── main.jsx             # ReactDOM 렌더링을 위한 부트스트랩 파일
│       └── styles/main.css      # 애플리케이션 전역 스타일시트
└── backend/
    ├── exercise_server.py       # FastAPI 메인 서버 (REST API, 정적 파일
    ├── db.py                    # SQLite 데이터베이스 세션 관리 유틸리티
    ├── exercise_records.db      # 로컬 SQLite 데이터베이스 파일 (실행 중
    └── exercise_data/           # 운동 로그, 캡처 이미지/영상 저장 폴더 (필
```

4.4. 클라우드 아키텍처 설계 (확장안)

현재는 로컬 데모 환경을 기준으로 설계되었으나, 향후 클라우드 배포를 고려한 확장 아키텍처는 다음과 같이 구상할 수 있습니다.

- ✓ **프론트엔드 배포**: React 애플리케이션을 빌드하여 정적 파일로 만든 후, AWS S3와 CloudFront, 또는 Vercel, Netlify와 같은 정적 호스팅 서비스를 통해 배포합니다. 이를 통해 전 세계 사용자에게 빠른 속도로 UI를 제공할 수 있습니다.
- ✓ **백엔드 배포**: FastAPI 애플리케이션을 Dockerize하여 AWS EC2, ECS, 또는 Google Cloud Run과 같은 컨테이너 기반 서비스에 배포합니다. Auto Scaling을 구성하여 트래픽에 따라 유연하게 서버 자원을 조절할 수 있습니다.
- ✓ **데이터베이스 및 스토리지**: 로컬 SQLite는 AWS RDS, Google Cloud SQL과 같은 관리형 데이터베이스 서비스로 마이그레이션하여 안정성과 확장성을 확보합니다. 운동 영

상 및 캡처 파일은 AWS S3나 Google Cloud Storage 같은 오브젝트 스토리지에 저장합니다.

- ✓ **보안 강화**: API 키와 같은 민감 정보는 AWS Secrets Manager나 Google Secret Manager를 통해 안전하게 관리하고, 서버 환경변수로 주입합니다. 프론트엔드에 필요한 설정은 CI/CD 파이프라인에서 빌드 시점에 주입하여 코드 노출을 방지합니다.

5. 요구사항 정의

5.1. 기능 요구사항 (Functional Requirements)

- ✓ [FR-01] 사용자는 브라우저에서 웹캠을 활성화하고, 클라이언트 측 MediaPipe를 통해 실시간으로 관절 좌표와 각도를 계산할 수 있어야 한다.
- ✓ [FR-02] 시스템은 스쿼트와 숄더프레스 운동 규칙에 따라 Rep(횟수)을 카운트하고, 핵심 자세(최하단/최상단)를 자동으로 캡처하여 UI에 스켈레톤과 함께 표시해야 한다.
- ✓ [FR-03] 캡처된 이미지는 FastAPI 백엔드의 `/analyze-image` API로 전송되어 OpenAI Vision API 기반의 자세 분석 피드백을 받아야 한다.
- ✓ [FR-04] 시스템은 수신된 텍스트 피드백을 브라우저의 Web Speech API를 사용하여 한국어 음성(TTS)으로 사용자에게 안내해야 한다.
- ✓ [FR-05] 사용자는 YouTube Data API를 통해 추천된 튜토리얼 영상을 시청하거나, 직접 영상 ID/링크를 입력하여 원하는 영상을 임베드할 수 있어야 한다. API 실패 시에는 기본 영상으로 대체되어야 한다.
- ✓ [FR-06] 사용자가 운동 종목을 변경할 경우, 현재 진행 중인 세션 상태(카운트, 로그, 피드백 내역)는 모두 초기화되어야 한다.
- ✓ [FR-07] OpenAI, YouTube API 키 등 민감한 정보는 백엔드 서버의 환경변수를 통해 주입되어야 하며, Git 저장소에 포함되지 않아야 한다.
- ✓ [FR-08] 사용자는 언제든지 '운동 종료' 버튼을 통해 카메라 스트림을 중지하고, 모든 카운트와 로그를 리셋할 수 있어야 한다.

5.2. 비기능 요구사항 (Non-Functional Requirements)

- ✓ [NFR-01] **성능**: 로컬 데모 환경에서 사용자가 자세를 취한 후 피드백을 받기까지의 지연 시간(Latency)은 수백 ms에서 최대 1초 이내를 목표로 한다.
- ✓ [NFR-02] **안정성 및 예외 처리**: 외부 API(OpenAI, YouTube)의 키가 누락되거나 할당량(Quota)이 초과될 경우, 시스템이 중단되지 않고 사용자에게 명확한 안내와 함께 대체 기능(기본 영상, 기본 안내 메시지)을 제공해야 한다.
- ✓ [NFR-03] **보안**: API 키가 소스 코드에 하드코딩되는 것을 방지하고, `.env` 파일을 사용하여 관리하며, 이 파일은 Git 저장소에 포함되지 않도록 `.gitignore`에 명시해야 한다.
- ✓ [NFR-04] **가용성**: 외부 네트워크(OpenAI, YouTube) 연결에 실패하더라도, 웹캠 스트리밍, 로컬 자세 분석, 기본 튜토리얼 영상 재생 등 핵심 오프라인 기능은 정상적으로 동작해야 한다.

6. 데이터 및 연동 정의

6.1. 데이터 흐름 정의

- ✓ **입력 데이터**:
  - ✓ 실시간 웹캠 비디오 프레임 (클라이언트 로컬 처리)
  - ✓ 자세 분석용 캡처 이미지 (Base64 인코딩된 JPEG 형식)
  - ✓ 사용자 선택 운동 종류 (e.g., 'squat', 'shoulder_press')
  - ✓ LLM 채팅을 위한 사용자 질문 텍스트
- ✓ **출력 데이터**:
  - ✓ AI 코칭 피드백 텍스트 (GPT-4o-mini 생성)
  - ✓ 운동 Rep 카운트 (숫자)

- ✓ 분석된 자세 캡처 이미지 (UI 표시용)
- ✓ 튜토리얼 영상 정보 (YouTube 영상 ID 및 제목)

✓ 관리 데이터:

- ✓ 외부 API 키: `OPENAI_API_KEY`, `YOUTUBE_API_KEY` (백엔드 환경변수)
- ✓ 로컬 데이터베이스: `exercise_records.db` (SQLite) - 운동 세션 기록 저장
- ✓ 로컬 파일 시스템: `exercise_data/` 폴더 - 상세 로그 및 영상 파일 저장

6.2. 시스템 연동 방식

✓ 프론트엔드 ↔ 백엔드 (FastAPI):

- ✓ **정적 파일 서빙**: 최초 접속 시, FastAPI 서버가 `frontend/` 폴더의 `index.html` 및 관련 리소스(JS, CSS)를 클라이언트에 제공합니다.
- ✓ **REST API 통신**: 모든 동적 데이터 교환은 비동기 REST API 호출을 통해 이루어집니다.
  - ✓ `POST /analyze-image`: 캡처된 이미지를 전송하고 AI 피드백을 요청합니다.
  - ✓ `POST /chat`: LLM 기반 채팅 응답을 요청합니다.
  - ✓ `GET /search-youtube`: 튜토리얼 영상 검색을 요청합니다.
  - ✓ `POST /save-log`, `/save-session`: 운동 기록 저장을 요청합니다.

✓ 백엔드 ↔ 외부 서비스:

- ✓ **OpenAI API**: 백엔드 서버가 `httpx` 라이브러리를 사용하여 OpenAI의 Vision API를 호출하고, 이미지 분석 결과를 받아옵니다.
- ✓ **YouTube Data API**: 백엔드 서버가 Google API 클라이언트 라이브러리를 사용하여 YouTube 영상 검색을 수행하고, 결과를 파싱하여 프론트엔드에 전달합니다.

7. 개발 환경 및 실행 방법

본 프로젝트는 Conda 가상환경에서 실행하는 것을 권장합니다. 자세한 절차는 프로젝트 루트의 `HOWTORUN.md` 파일을 참고하십시오.

7.1. 개발 환경 설정

```
 # 1. Conda 가상환경 생성 및 활성화
conda create -n exercise-coach python=3.10
conda activate exercise-coach

# 2. Python 의존성 패키지 설치
pip install -r requirements.txt

# 3. API 키 설정 (.env 파일 생성)
# 프로젝트 루트 또는 frontend/ 폴더에 .env 파일을 생성하고 아래 내용을 입력
# 이 파일은 Git에 의해 추적되지 않습니다.
OPENAI_API_KEY="your_openai_api_key"
YOUTUBE_API_KEY="your_youtube_api_key"
```

7.2. 애플리케이션 실행

현재 아키텍처는 백엔드 FastAPI 서버가 프론트엔드 정적 파일까지 함께 서빙하는 방식입니다.

```
 # 1. 백엔드 서버 실행 (프로젝트 루트 폴더 기준)
# API 키를 환경변수로 export 합니다. (Linux/macOS)
export OPENAI_API_KEY=$(grep OPENAI_API_KEY .env | cut -d '=' -f2)
export YOUTUBE_API_KEY=$(grep YOUTUBE_API_KEY .env | cut -d '=' -f2

# backend 폴더로 이동하여 Uvicorn 서버 실행
```

```
cd backend
uvicorn exercise_server:app --host 0.0.0.0 --port 8003

# 2. 브라우저 접속
# 웹 브라우저를 열고 아래 주소로 접속합니다.
# http://localhost:8003/
```

※ **참고**: 프론트엔드를 별도의 정적 서버로 실행해야 할 경우, `frontend/generate_env_js.py` 스크립트를 실행하여 `env.js` 파일을 생성한 후 진행해야 합니다. 자세한 내용은 `HOWTORUN.md`를 참조하십시오.

참고 자료

[1] MediaPipe 기반 운동자세 교정 시스템의 기능 개선 연구
[2] YOLOv8-MediaPipe를 이용한 운동자세 교정 자동 피드백 시스템
[3] 실시간 동작 인식 및 자세 교정 스마트 미러 피트니스 시스템