# ♥목차페이지

1. 프로젝트 개요

2. 프로젝트 팀 구성 및 역할

3. 프로젝트 수행 절차 및 방법

4. 프로젝트 수행 경과(핵심기능)

5. 피드백 반영

6. 트러블 슈팅

7. 기술 연구

8. 형상관리 및 배표

9. 회의록

10. 자체 평가 의견

# 1. 프로젝트 개요

- 1.1. 프로젝트 주제 및 선정 배경, 기획의도
- 1.2. 활용 기술 및 개발 환경
- 1.3. 프로젝트 구조
- 1.4. 활용방안 및 기대 효과

# ● 1.1. 프로젝트 주제 및 선정 배경, 기획의도

### 프로젝트 주제

Fasle Cam: Al 기반 '경험 재구성' 서비스

### 선정 배경

생성형 AI 기술이 빠르게 발전하면서, 사용자마다 개성과 이야기가 담긴 디지털 콘텐츠를 만들어내는 데 관심이 커 졌습니다.이제는 단순히 필터나 효과만 적용하는 것이 아니라, 각자의 이야기를 담은 콘텐츠가 필요하다는 점을 느끼게 되었습니다.

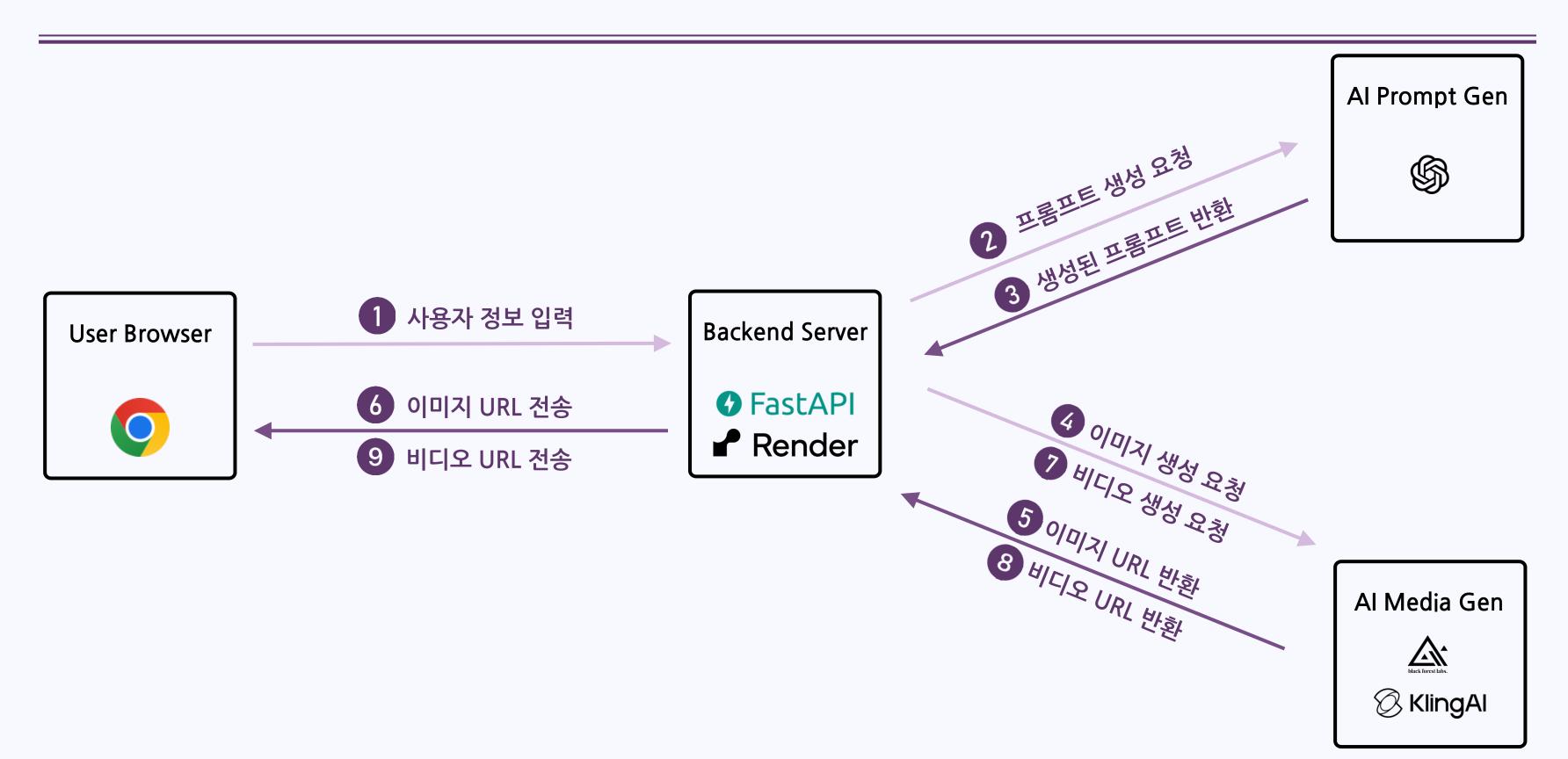
### 기획 의도

누구나 쉽게 자신의 생각이나 얼굴 사진을 넣으면, AI가 전에 없던 새로운 이미지나 영상을 만들어주는 서비스를 기획했습니다. 이를 통해 사용자가 직접 자신만의 디지털 페르소나를 만들어보고 다양한 방식으로 자신을 표현하는 재미를 느낄 수 있으면 좋겠습니다.

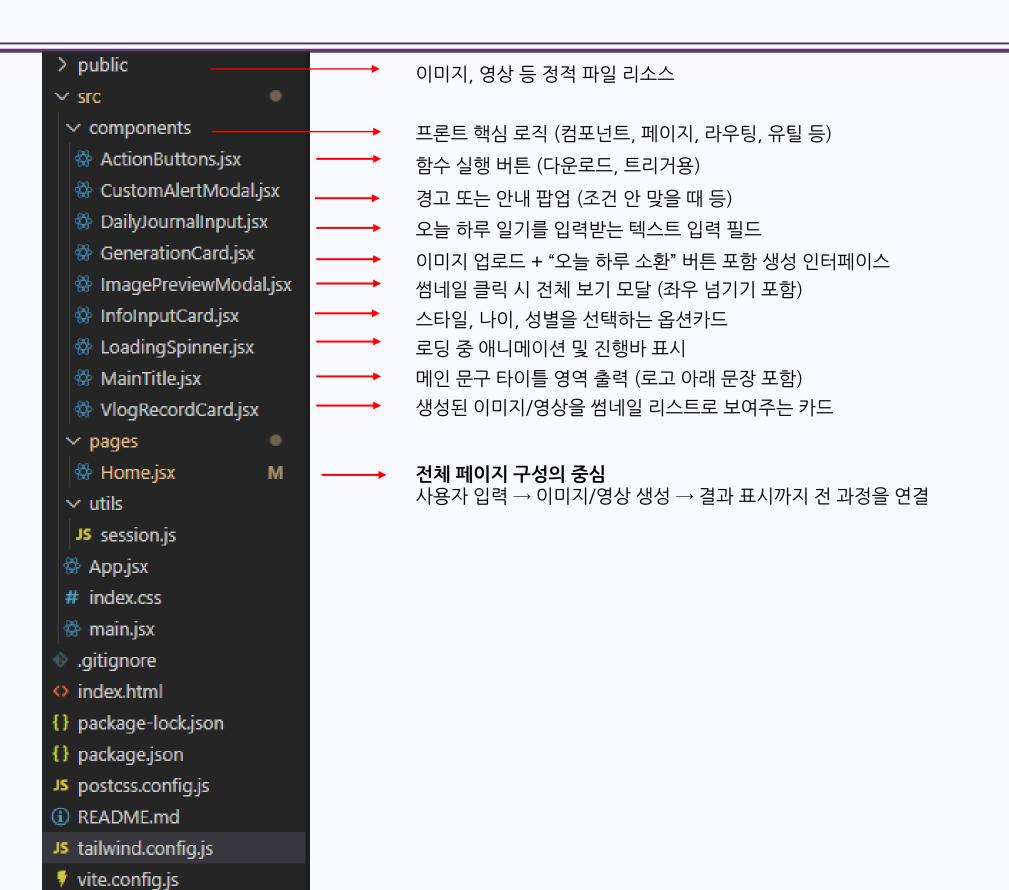
# ◎ 1.2. 활용 기술 및 개발 환경

구분	기술 스택
Frontend	React, Vite, JavaScript, Tailwind CSS, Headless UI
Backend	Python, FastAPI, Uvicorn
Al Models	LLM: GPT-3.5-Pro Text-to-Image: Flux-Kontext-Pro Image-to-Video: Kling-2.1-standard
Deployment	Frontend: Cloudflare PagesBackend: Render
Version Control	GitHub
Collaboration	Discord

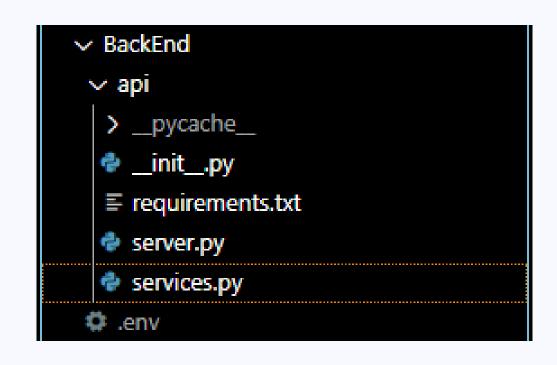
# ● 1.3. 프로젝트 구조 (시스템 아키텍처)



# ● 1.3. 프로젝트 구조 (프론트엔드)



# ● 1.3. 프로젝트 구조 (백엔드)



requirements.txt

: FastAPI, openai 등 의존성 패키지 목록

server.py

: FastAPI 앱 초기화 및 라우터 구성

services.py

: 이미지/프롬프트/비디오 처리 핵심 기능 담당

\_\_init\_\_.py

: API 모듈 초기화 파일

.env

: API 키 등 보안 변수 관리

# ◎ 1.4. 활용 방안 및 기대 효과



### 쉽고 재밌는 표현 놀이터

FalseCam에서는 일상을 새롭게 해석해서, 그냥 기록하는 것만이 아니라 재미있게 남길 수 있습니다.

사진과 글을 AI로 변신시켜 친구들과 공 유하다 보면, 자연스럽게 나만의 디지털 페르소나가 만들어집니다.

딱딱한 기록 대신, 쉽고 빠르게 내 개성을 마음껏 뽐낼 수 있는 새로운 디지털 놀이터가 열립니다.

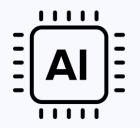


### 개인화 콘텐츠 생성

FalseCam이 만들어주는 이미지와 영상은 내 SNS 프로필, 블로그, 유튜브, 어디든 자유롭게 쓸 수 있습니다.

복잡한 도구나 기술을 몰라도, 원하는 스타일을 쉽게 골라 바로 멋진 콘텐츠로 완성할 수 있습니다

내가 원하는 스타일, 나이, 성별까지 선택하면서 나만의 온라인 공간을 더 특별하게 꾸밀 수 있습니다.



### 생성형 AI기술의 대중화

이 서비스는 누구나 AI의 다양한 기능을 어렵지 않게 경험할 수 있도록 설계되었 습니다.

복잡한 설명이나 기술 지식 없이도, 직접 텍스트나 이미지를 넣고 결과를 바로확인할 수 있습니다.

AI 기술에 대한 이해가 없어도, 직접 만져보며 '생성형 AI'가 무엇인지 자연스럽게 느껴볼 수 있는 기회를 제공합니다.

# ♥ 2. 프로젝트 팀 구성 및 역할

이름	역할	주요 담당 업무
서민석	팀장(PM)	프로젝트 총괄 기획 및 전체 일정 관리 팀원 간의 역할 조율 및 의사소통 관리 생성형 AI 모델 선정 및 성능 테스트 서버리스 환경 배포 및 운영 최종 결과물 품질 검수 및 보고서 작성 총괄
최태열	프론트엔드 개발자	UI/UX 설계 및 React 컴포넌트 개발 와이어프레임 및 프로토타입 제작 React와 Tailwind CSS를 사용한 UI 컴포넌트 구현 백엔드 API 연동 및 상태 관리 FastAPI로 구현된 백엔드 REST API 호출 및 데이터 처리
김주현	백엔드 개발자	FastAPI 기반 REST API 서버 구축 핵심 API 엔드포인트 설계 및 구현 텍스트 → 이미지 → 비디오로 이어지는 순차적 AI 생성 파이프라인 개발 프롬프트 엔지니어링 및 모델 파라미터 튜닝 LLM을 활용한 프롬프트 생성 시스템 설계

# ● 3.1. 프로젝트 수행 절차

### 착수 및 기획

WBS 수립 및 기능 정의

와이어프레임 및 화면 설계

기술 스택 확정 및 아키텍처 설계

### MVP 개발

개발 환경 설정

백엔드 개발 (AI 연동)

프론트엔드 개발 (UI / API 연동)

통합 테스트 및 배포

### 고도화 및 마무리

Image-to-Video 기능 연동

비디오 결과 UI 구현

최종 보고서 작성 및 발표

# ● 3.2. WBS 상세

																7	월								
	<u>Wo</u>	rk Brea	k-down Stru	ucture			Н		2주		$\top$			3주		Ť			<b>1</b> 주		$\neg$		5주	5/1주	
Procedures St	teps - Lattols	ELELTI	Scho	edule	A KEE	1150 dil 76 l' 11 \	화	수		토	일 월	화	수	목글	토	일	화	수	목	금토	일	월 화	수	목 금	토 일
	Tasks(작업)	담당자	시작일	종료일	Goal(목표)	산출물/비고(Deliverable)	8	9	10 11	12	13 14	4 15	16	17 1	3 19	20 2	1 22	23	24 2	25 26	27	28 29	30	목 금 31 1	2 3
1.0.0. 착수 및 기	획		07월 09일	07월 11일	프로젝트 범위 및 실행 계획 확정	최종 기획서, WBS																			
1.1	.0 기획 및 설계																	П					П	$\perp$	
	1.1.1. WBS 수립 및 기능 요구사항 정의	서민석	07월 09일	07월 09일	개발 범위와 규칙의 명확화	WBS 조안, 기능명세서																			
	1.1.2 와이어프레임 및 화면 설계	최태열	07월 10일	07월 11일	사용자 경험(UX) 시각화	와이어프레임																		$\top$	
	1.1.3 기술 스택 확정 및 아키텍처 설계	김주현	07월 10일	07월 11일	안정적인 기술 구조 설계	아키텍처 설계도																			
1.2	2.0 프로젝트 관리																								
	1.2.1. 상세 WBS 및 기획서 작성	서민석	07월 10일	07월 11일	효율적인 힘업 체계 구축	WBS, 기획서																			
	1.2.2. Glt 레포지토리 및 브랜치 전략 수립	주석열	07월 10일	07월 11일	체계적인 코드 관리 환경 구축	Git 레포지토리																			
2.0.0. MVP 구현	및 검증		07월 14일	07월 18일	핵심 기능 프로토타입 완성	동작 가능한 MVP 서비스																			
2.1	.0 개발 환경 설정																								
	2.1.1. Cloudflare 프로젝트 생성 및 연동	김주현	07월 14일	07월 14일	신속한 개발 및 배포 환경구축	클라우드 프로젝트		П										П					$\Box$	$\top$	
	2.1.2. React 프로젝트 초기화 및 초기 환경 설정	최태열	07월 14일	07월 14일	프론트엔드 개발 환경 구축	초기화된FE프로젝트구조																			
	2.1.3. Model Testing(img-to-img)	서민석	07월 14일	07월 14일	프로젝트와 적합한 img-to-img model 찾기	N/A																		$\top$	
	2.1.3. Model Testing(vid-to-vid)	서민석	07월 15일	07월 16일	프로젝트와 적합한 vid-to-vid model 찾기	N/A												П	$\top$				П	$\top$	
2.2	2.0 백엔드 개발																								
	2.2.1. AI 연동 및 image-to-image 생성 API 개발	김주현	07월 15일	07월 17일	서비스의 핵심 AI 기능 구현	text-to-image API																		$\perp$	
	2.2.2. 백엔드 서버 배포(Render)	서민석	07월 17일	07월 17일	API 엔드포인트 공개 및 접근성 확보	공개 URL을 통해 접근 가능한 백엔드 API																			
2.3	3.0 프론트엔드 개발																								
	2.3.1. 공통 UI 컴포넌트(레이아웃, 버튼) 개발	최태열	07월 15일	07월 16일	일관성 있는 ui 및 개발 효율 확보	재사용 가능한 공통 ui 컴포넌트																			
	2.3.2. 메인/결과 페이지 UI 및 API 연동	최태열	07월 17일	07월 17일	사용자가 핵심 기능을 사용할 화면 구현	API가 연동된 메인/결과 페이지																			
	2.3.3. 이미지 생성 후 비디오 생성 API 호출 플로우 구현	최태열	07월 18일	07월 18일	이미지.비디오 전환 사용자 경험(ux) 개선	이미지-비디오 자동 연동 기능 구현																		$\perp$	
2.4	l.0 통합 및 테스트																								
	2.4.1. 프로덕션 환경 구축 (Render + Cloudflare)	주석열	07월 18일	07월 18일	커스텀 도메인을 통한 서비스 접근 환경 마련	최종 배포 URL이 적용된 MVP																			
	2.4.2. MVP 기능 통합 및 E2E 테스트, 버그 수정	주석열	07월 18일	07월 18일	MVP 버전의 안정성 확보	테스트 완료된 MVP																			
3.0.0. 고도화 및	마무리		07월 21일	07월 28일	서비스 완성도 향상 및 최종 보고	최종 배포 URL, 최종 보고서																			
3.1	.0. 백엔드 개발																	П					$\Box$	$\top$	
	3.1.1. Image-to-Video API 연동	김주현	07월 21일	07월 24일	콘텐츠의 매력도 중대	Image-to-Video API																		$\top$	
		서민석	07월 21일	07월 24일	는 단의의 해목도 중대	Image-to-video API																			
3.2	2.0 프론트엔드 개발																								
	3.2.1. session_id(uuid) 관리 및 전송 정책 적용	최태열	07월 21일	07월 22일	세션 단위의 트래킹 및 데이터 관리 신뢰성 향상	session_id 기반 데이터 전송 및 검증 내역																		$\perp$	
	3.2.2. 생성 결과 표시 및 SNS 공유 기능 강화	최태열	07월 23일	07월 24일	콘텐츠의 동적 표현 및 바이럴 유도	생성된 이미지/영상 결과 확인 및 공유 기생	do																		
3.3	1.0 최종 검수 및 배포																								
	3.3.1. 전체 기능 통합 및 최종 QA	주석열	07월 25일	07월 25일	서비스의 전체적인 안정성 확보	최종 테스트 결과서																			
	3.3.2. 프로덕션 배포 및 안정화	주석열	07월 28일	07월 28일	실제 사용자가 접근 가능한 서비스 출시	배포된 서비스 URL																			
	3.3.3. 프로젝트 최종 보고서 작성 및 제출	서민석	07월 28일	07월 28일	프로젝트 과정과 결과를 공식화	프로젝트 최종 보고서																			

# 4. 프로젝트 수행 경과

- 4.1. 텍스트/이미지 기반 AI 이미지 생성
- 4.2. 생성된 이미지 기반 AI 비디오 생성
- 4.3. 결과물 조회 및 인터렉션
- 4.4. 익명 세션 기반 사용자 관리

# ● 4.1. 텍스트/이미지 기반 AI 이미지 생성(프론트엔드)

- 사용자의 모든 입력을 state로 관리합니다.
- '오늘 하루 소환 버튼 클릭 시, 모든 입력 데이터를 FormData 객체로 만들어 백엔드 API(/generate-image) 에 전송합니다.
- 요청 중에는 로딩 상태를 활성화하여 사용자에게 피드백을 제공합니다.

```
const handleGenerate = async () => {
 if (!text.trim()) {
   setAlertMessage("일기 내용을 먼저 작성해주세요!");
   setAlertSubMessage("오늘 있었던 일을 간단하게 들려주세요.");
   setShowAlert(true);
   return;
 setIsSummoning(true);
 setProgress(0);
 const interval = setInterval(() => {
   setProgress(prev => {
     if (prev >= 95) {
       clearInterval(interval);
      return prev;
    return prev + 1;
 }, 1200);
   const formData = new FormData();
   formData.append('text', text);
   formData.append('style', style);
   formData.append('age', age);
   formData.append('gender', gender);
   if (file) {
     formData.append('image', file);
```

## ● 4.1. 텍스트/이미지 기반 AI 이미지 생성(백엔드)

/generate/image 엔드포인트에서는 텍스트, 이미지, 선택 옵션, 그리고 session ID를 함께 전달받습니다. 입력된 텍스트는 get\_translated\_text() 함수를 통해 OpenAI API를 호출하여 영어 프롬프트로 변환됩니다. 변환된 프롬프트와 이미지 데이터를 FAL 모델에 전달하여 최종 이미지를 생성합니다. 서버는 FAL 모델로부터 받은 이미지 URL의 데이터를 직접 가져온 뒤, 이를 Base64 데이터 URI로 인코딩하여 JSON 형식으로 클라이언트에 응답합니다.

```
logging.info(f"* fal.ai 이미지 프롬프트: {composed_prompt}")

try:

img_result = await asyncio.to_thread(
 fal.run, FAL_IMAGE_MODEL_ENDPOINT, arguments=payload
)

final_image_url = img_result["images"][0]["url"]

except Exception as e:

logging.error(f" X fal.ai 이미지 처리 중 예외 발생: {e}")

return JSONResponse(
 status_code=500, content={"success": False, "error": str(e)}
)
```

## ● 4.2. 생성된 이미지 기반 AI 비디오 생성(프론트엔드)

• 이미지 생성 API 호출이 성공하면, 반환 된 이미지 URL과 프롬프트를 사용하여 즉시 비디오 생성 API(/generatevideo)를 호출합니다.

```
const result = await response.json();
if (result.success && result.image) {
  const newImageItem = { src: result.image, type: 'image', date: new Date() };
  setImageList((prev) => [newImageItem, ...prev]);
 setCurrentIndex(0);
  console.log('이미지 생성 성공!', result.image);
  const videoRes = await fetch('https://falsecam.onrender.com/generate/video',
   method: 'POST',
   headers: { 'Content-Type': 'application/json', 'sessionID': sessionID },
    body: JSON.stringify({
     prompt: text,
      image_url: result.image,
      style,
      age,
     gender,
  const videoResult = await videoRes.json();
```

## ● 4.2. 생성된 이미지 기반 AI 비디오 생성(백엔드)

• 엔드포인트에서 이미지 URL과 프롬프 트를 받아 Kling 비디오 모델을 호출합 니다.

```
@router.post("/generate/video")
async def generate_video(
   request: VideoRequest,
   session_id: str = Header(None, alias="sessionID")
 -> JSONResponse:
   if not BFL API KEY:
       return JSONResponse(status_code=500, content={"success": False, "error": "API key missing."})
   logging.info(f" 🚀 비디오 생성 요청 (세션: {session_id}): {request.prompt}")
   logging.info(f"♥ 비디오 생성 진행 (세션: {session_id})")
   payload = {
       "image_url": request.image_url,
        "prompt": request.prompt,
   logging.info(f" 💉 비디오 생성 요청: {request.prompt}")
    try:
       video_result = await asyncio.to_thread(
           fal.run,
           FAL_VIDEO_MODEL_ENDPOINT,
           arguments=payload
       final_video_url = video_result["video"]["url"]
```

# ♥4.3. 결과물 조회 및 인터랙션(프론트엔드)

- 생성된 이미지와 비디오를 모달 창에서 좌우로 넘겨볼 수 있는 갤러리 기능을 구현했습니다
- 현재 보고 있는 콘텐츠의 타입(image 또는 video)에 따라 다른 태그를 렌더링 하여 사용자 경험을 최적화했습니다.

```
return (
 <div
   className="fixed inset-0 z-50 bg-black/80 flex items-center justify-center"
   onClick={() => setIsOpen(false)}
     className="relative w-auto h-full max-h-[90vh]"
     onClick={(e) => e.stopPropagation()}
     {/* 이전 버튼 */}
     {currentIndex !== 0 && (
       <button
        onClick={prevImage}
        className="absolute left-4 top-1/2 -translate-y-1/2 z-20 text-white text-4xl hover:text-pri
        aria-label="이전"
       </button>
     {/* 다음 버튼 */}
     {currentIndex !== imageList.length - 1 && (--
     {/* 메인 이미지/비디오 */}
     {currentItem && currentItem.type === 'video' ? (
       <video
        key={currentItem.video_url}
         src={currentItem.video_url}
         controls
         autoPlay
         className="object-contain w-full h-full rounded-xl shadow-2xl"
```

### ♥ 4.4. 익명 세션 기반 사용자 관리(프론트엔드)

- 회원가입 없이도 사용자를 식별하고 연 속적인 작업 흐름을 지원하기 위해 익명 세션 ID를 사용합니다.
- 사용자가 처음 접속할 때 SSID로 세션 ID를 생성하여 로컬 스토리지에 저장하고, 모든 API 요청 시 이 ID를 HTTP 헤더에 담아 전송합니다.

```
import { v4 as uuidv4 } from 'uuid';

export function getSessionID() {
   let sessionID = localStorage.getItem('sessionID');
   if (!sessionID) {
      sessionID = uuidv4();
      localStorage.setItem('sessionID', sessionID);
   }
   return sessionID;
}
```

```
const sessionID = getSessionID();
const response = await fetch('https://falsecam.onrender.com/generate/image', {
    method: 'POST',
    headers: {
        'sessionID': sessionID
     },
     body: formData,
});
```

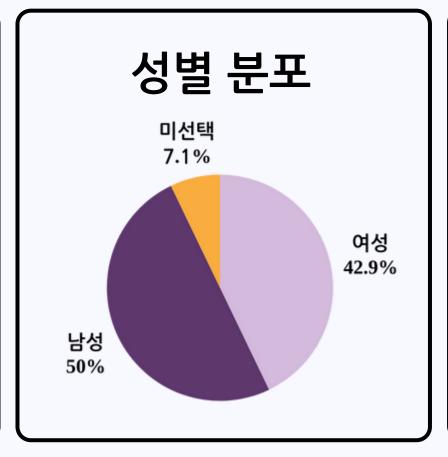
### ● 4.4. 익명 세션 기반 사용자 관리(백엔드)

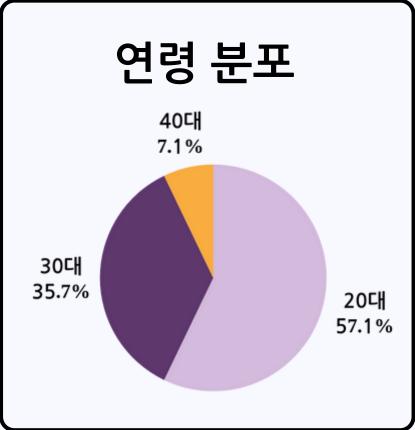
- 클라이언트는 고유한 sessionID를 생성하 여 HTTP 요청의 Header에 포함시켜 전송 합니다.
- 서버는 FastAPI의 Header 의존성을 통해 해당 세션 ID를 수신하고, 요청을 세션 단 위로 구분합니다.
- 생성된 이미지 및 영상 결과물은 sessionID 기준으로 폴더를 분리하여 저장되며,
- 로그 역시 세션 단위로 기록되어 개별 요청 추적이 용이합니다.
- 본 방식은 사용자 인증 없이도 요청 식별이 가능하므로, 비회원 기반 서비스에 적합한 구조입니다.

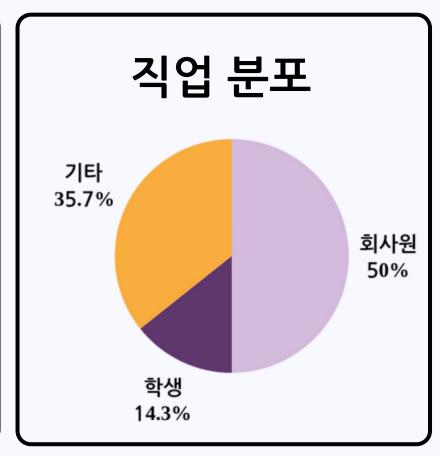
```
@router.post("/generate/video")
async def generate_video(
    request: VideoRequest,
    session_id: str = Header(None, alias="sessionID")
) -> JSONResponse:
    if not BFL_API_KEY:
        return JSONResponse(status_code=500, content={"success": False, "error": "API key missing."})
    logging.info(f"   비디오 생성 요청 (세션: {session_id}): {request.prompt}")
    logging.info(f"   비디오 생성 진행 (세션: {session_id})")
    payload = {
        "image_url": request.image_url,
        "prompt": request.prompt,
    }
```

피드백 참여자 분석









### 14명의 목소리

# 좋았어요 가볍게 재미로 하기 좋은 앱 같아요 깔끔하고 트렌디하다 단순해서 간단하게 이용하기 좋습니다 문구들이 센스가 넘쳐서 재미있었습니다 간단하고 직관적이라 좋았습니다 색이 이쁘고, 사용하기 편리했어요

# 아쉬웠어요 영상 생성속도가 느리다 예시 멘트가 추상적입니다 어디에 기입해야 하는지 한눈에 파악이 어려웠습니다 너무 단조롭고 심플합니다 선택칸의 간격이 좁아서 성별을 선택할 수 없어요 가끔씩 영상의 움직임이 자연스럽지 못해요

# 이런 건 어때요? 화풍이 더 추가되었으면 좋겠다 친구들에게 공유 기능 그림일기 형식처럼 나오면 더 재미있을 듯 하다 하루에 대한 피드백이나 운세 영상 길이를 선택할 수 있으면 좋을 것 같습니다 사진이나 동영상 중 하나만 생성되는 기능 추가

### 사용자 불편사항

이미지와 영상을 생성하시면서 불편하거나 아쉬웠던 점이 있으셨나요?

프롬프트 예시가 있으면 좋을 것 같습니다

이미지와 영상을 생성하시면서 불편하거나 아쉬웠던 점이 있으셨나요?

예시 멘트인 오늘 하루를 한 줄로 남겨볼까요? 라는 질문이 추상적입니다.

### 플레이스 홀더(전)

오늘 하루를 한 줄로 남겨볼까요?(최대100자)

플레이스 홀더(후)

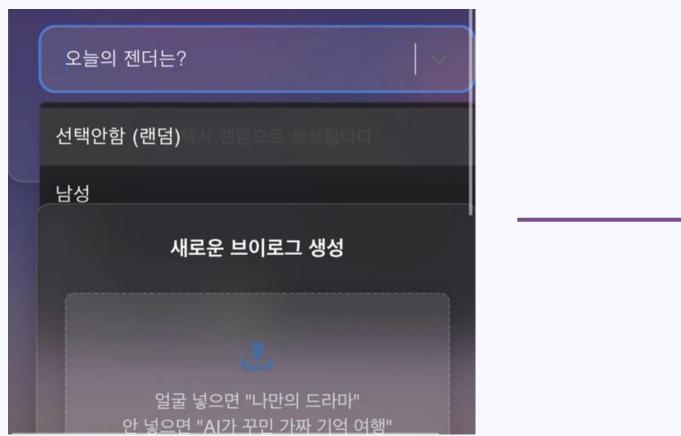
빗소리를 들으며 따뜻한 커피를 한잔했던 여유로운 주말 오후(최대100자)

### 사용자 불편사항

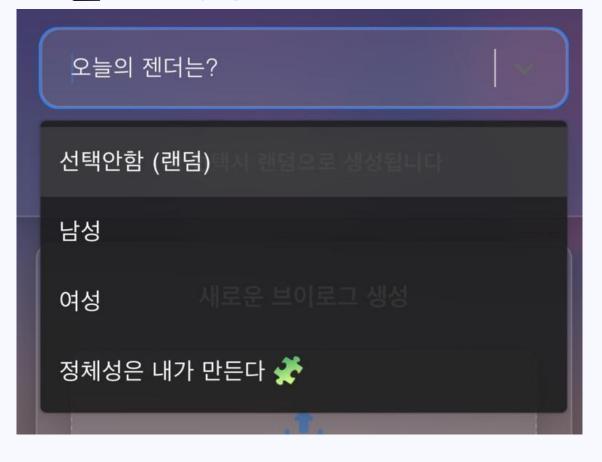
FalseCam의 전체적인 디자인과 UI(화면 구성)는 어땠나요? 느낀 점을 자유롭게 적어주세요.

화면구성은 깔끔하나 전체적으로 너무 어두워서 가볍게 재미로 하는 느낌은 안든다. 선택칸 간격이 너무 좁아서 성별을 선택하는 칸에서는 밑에 브이로그기록칸때문에 여성을 선택할수없었다 ..!

### 드롭다운 오버플로우(전)



### 포탈 렌더링 방식으로 해결(후)



# 6. 트러블 슈팅

- 6.1. 이미지 미리보기 모달(Modal) 렌더링 오류
- 6.2. blob을 이용한 외부 파일 다운로드 문제
- 6.3. base 64 인코딩 문제
- 6.4. 시스템 프롬프트

# ● 6.1. 이미지 미리보기 모달(Modal) 렌더링 오류

### 문제

이미지 썸네일을 클릭하면 미리보기 모달이 렌더링은 되지만, 화면에는 아무것도 표시되 지 않는 문제가 발생했습니다. 코드는 정상 적으로 실행되는 것처럼 보여 디버깅에 오랜 시간이 소요되었습니다.

### 원인

문제의 핵심은 CSS의 z-index와 position 속성이었습니다. 모달 컴포넌트는 렌더링되 었지만, z-index 속성이 누락되어 다른 컴포 넌트들 뒤에 가려진 상태로 출력되고 있었습 니다.

```
// FrontEnd/src/components/ImagePreviewModal.jsx (수정 전)
export default function ImagePreviewModal({ isOpen }) {
  if (!isOpen) return null;

return (

  <div className="flex items-center justify-center">
        <div className="bg-white p-8 rounded-xl shadow-lg">
        이미지 미리보기
        </div>
        </div>
        </div>
        );
}
```

### 해결 과정

모달 컴포넌트의 최상위 〈div〉에 position: fixed와 함께 화면 전체를 덮는 inset-0, 그리고 다른 모든 요소들보다 위에 표시되도록 z-index: 50 속성을 추가했습니다. 이 과정을 통해 모달이 항상 화면 최상단 중앙에 정상적으로 출력되도록 문제를 해결했으며, 다른 팝업 요소에도 일관된 레이어 구조를 설계하는 기반을 마련했습니다.

# ● 6.2. Blob을 이용한 외부 파일 다운로드 문제

### 문제

AI가 생성한 이미지나 비디오는 외부 서버에 저장되는데, 이 결과물을 다운로드하려고 하 면 파일이 저장되지 않고 새 탭에서 열리기 만 하는 문제가 발생했습니다.

### 원인

브라우저의 보안 정책(CORS) 때문이었습니다. 〈a〉 태그의 download 속성은 같은 서버(Same-Origin)에 있는 파일에만 동작하는데, 우리 파일은 외부 서버(Cross-Origin)에 있었기 때문에 다운로드 기능이 작동하지않았습니다.

link.click();

### 해결 과정

API를 사용해 파일 데이터를 먼저 가져온 뒤, 이 데이터를 Blob 객체로 변환했습니다. 그다음 URL.createObjectURL을 이용해 이 Blob에 대한 임시 로컬 URL을 생성하고, 보이지 않는 〈a〉 태그에 연결하여 클릭시키는 방식으로 브라우저가 파일을 다운로드하도록 구현하여 문제를 해결했습니다.

```
// FrontEnd/src/components/ActionButtons.jsx (수정 후)
                                                                    // 4. 메모리 해제를 위해 임시 URL 페기
                                                                    URL.revokeObjectURL(objectUrl);
const ActionButtons = ({ currentItem }) => {
                                                                } catch (error) {
                                                                    console.error("다문로드 실패:", error);
   const handleDownload = async () => {
                                                            ξ;
           // 1. fetch로 파일 데이터를 가져와 Blob으로 변환
           const response = await fetch(currentItem.src);
                                                            return (
           const blob = await response.blob();
                                                                <button onClick={handleDownload}>
                                                                    다윤로드
          // 2. Blob으로 임시 URL 생성
                                                                </button>
           const objectUrl = URL.createObjectURL(blob);
                                                            );
                                                        };
           // 3. 보이지 않는 <a> 태그로 다운로드 실행
           const link = document.createElement('a');
           link.href = objectUrl;
           link.download = `falsecam-download.jpg`;
```

## ● 6.3. Base 64 인코딩 문제

### 문제

사용자가 업로드한 이미지 파일을 외부 AI API로 전달했지만, API가 이미지를 인식하지 못하고 오류를 반환했습니다.

### 원인

외부 AI API는 이미지의 원본 파일 데이터 (bytes)가 아닌, 웹에서 접근 가능한 URL 혹은 base64로 인코딩된 데이터 URL 형식을 요구했습니다. 원본 데이터를 그대로 전달하려고 했기 때문에 형식이 맞지 않아 오류가 발생했습니다.

### 해결 과정

base64 라이브러리를 사용하여 이미지 파일의 원본 데이터를 인코딩하고, "data:[MIME타입];base64,[인코딩된 데이터]" 형식의 data URL로 변환해주었습니다. 이 방식을 통해 파일을 서버에 임시 저장하지 않고도, 외부 API가 이미지를 즉시 인식하고 처리할 수 있도록 문제를 해결했습니다

```
@router.post("/generate-image")
async def generate_image(
    image: UploadFile | None = File(None),
    ...
):
    image_data_url = None
    if image:
        image_data = await image.read()
        # 문제: 원본 파일 데이터(bytes)를 그대로 전달하려고 함
        image_data_url = image_data

# ...

# Fal.ai API는 image_data_url에 URL 문자열을 기대하지만,
# 실제로는 bytes가 전달되어 오류 발생
img_result = await asyncio.to_thread(
        fal.run,
        ...,
        arguments={"image_url": image_data_url, ...}
)
```

# ●6.4. 시스템 프롬프트

### 문제

사용자의 입력 텍스트를 기반으로 이미지를 생성하는 과정에서 system\_prompt가 제대로 반영되지 않아 원하는 프롬프트 형식이 출력되지 않았습니다.

### 원인

초기에는 OpenAI API 호출 시 messages 목록에 단순한 system 역할의 텍스트만 포함되어 있었으며, 프롬프트 구성 기준(시점, 배경, 복장 등)이 명확히 전달되지 않아 이미지 생성 품질이 저하되는 문제가 있었습니다.

### 해결 과정

사용자가 입력한 텍스트만으로도 자연스러운 시점, 배경, 의상 등이 포함된 프롬프트가 생성될 수 있도록 LLM이 스스로 추론하여 구성할 수 있도록 설계하였습니다. 이를 위해 프롬프트 생성 단계에서 system\_prompt를 정교하게 구성하여, LLM이 장면에 맞는 시점, 배경, 복장 등을 창의적으로 생성하도록 유도하였습니다. 그 결과, 간단한 입력만으로도 시각적 정보가 풍부한 프롬 프트를 안정적으로 생성할 수 있게 되었습니다.

```
system_prompt = (
    "You are an AI prompt creator. Based on the user's input, create a detailed and "
    "atmospheric English prompt for an image generation AI. Key requirements: "
    "1. Describe a scene with a clear background, action, and a positive, cheerful mood. "
    "2. This is the most important rule: The main character MUST be facing the camera or "
    "be in a three-quarter view. Absolutely no back views. "
    "3. Do not describe the character's face in any way. Focus only on the scene, "
    "setting, and action. "
    "4. Based on the scene and action, briefly suggest an appropriate and stylish "
    "outfit for the character. Example: for 'a walk in the park', suggest "
    "'wearing a casual hoodie and jeans'. For 'a beach vacation', suggest "
    "'wearing a light summer dress'."
)
messages = [
    {"role": "system", "content": system_prompt},
    {"role": "user", "content": text},
]
```

# 7. 기술연구

- 7.1. 연구 목표
- 7.2. 테스트 과정
- 7.3. 연구 결과 및 최종 모델 선정

### 7.1. 연구목표

프로젝트의 핵심은 사용자의 입력을 바탕으로 고품질의 이미지와 영상을 생성하는 것이므로, 단순히 기능을 구현하는 것을 넘어 서비스의 품질과 운영 비용을 최적화하기 위한 기술 연구를 진행했습니다. 다양한 외부 AI 모델을 사전에 체계적으로 테스트하고 비교 분석하여, 아래 세 가지 핵심 지표를 기준으로 종합적으로 평가하여 균형 잡힌 선택을 하고자 했습니다.



### 품질

사용자의 입력(텍스트, 이미지)을 가장 잘 해석하고, 창의적이며 미적으로 우수 한 결과물을 생성하는가?



### 속도

사용자가 지루함을 느끼지 않을 합리적 인 시간 내에 결과물을 생성하는가?



### 비용

서비스의 지속 가능한 운영을 고려했을 때, 생성 비용이 합리적인가?

Image to Image 모델 테스트는 python 코드로 실행했습니다.

```
MAGE MODEL NAME = "flux-pro/kontext"
sare_image_modet_name = image_modet_name.reptace( / , _ )
OUTPUT_DIR = "output"
os.makedirs(OUTPUT_DIR, exist_ok=True)
local_path = "samples/sample_image.png"
                                                                                 ▶ 2.입력이미지 선택
 mage url = fal_client.upload_file(local_path)
lef get_timestamp():
   return datetime.now().strftime("%Y%m%d%H%M%S")
                                                                                 ▶ 3.입력 프롬프트
   generate_image_prompt_from_daily_life(daily_text: str) -> str:
      "fal-ai/any-llm",
          "model": "google/gemini-flash-1.5",
               y신은 창의로운 이미지 프롬프트 제너레이터입니다. "
                     적은 평범한 일상 텍스트와 제공된 이미지를 받아, "
                                                                                ▶ 4.시스템 프롬프트
              이미지의 인물이 주인공인 픽사 애니메이션 스타일로 묘사할 수 있는 "
             'AI 이미지 생성용 프롬프트 한 개를 번호 매김 없이 영어로 출력하세요.
              배경에도 환상적이고 디테일하게 신경 써서 설명하세요."
          "prompt": f"일상 설명: "{daily_text}"",
          "reasoning": False
   result = handler.get()
   return result["output"]
                                                                                 ▶ 5.이미지 생성 요청
   generate_image_from_prompt(image_url: str, prompt: str) -> str:
   resp = fal_client.run(
      f"fal-ai/{IMAGE_MODEL_NAME}",
          "image_url": image_url,
          "prompt": prompt
   return resp["images"][0]["url"]
```

입력한 이미지와 프롬프트는 아래와 같습니다.

입력 이미지	입력 텍스트	시스템 프롬프트
	"눈부신 태양 아래서 돌고래와 같이 서핑을 했어"	"당신은 창의로운 이미지 프롬 프트 제너레이터입니다. 사용자가 적은 평범한 일상 텍 스트와 제공된 이미지를 받아, 평범했던 일상을 제3자가 찍은 파파리치샷으로 변형시키고, 얼 굴이 보일 수 있는 구도로 , 의 상도 상황에 맞게 변형시켜주고, 성별은 구분하지 말고, 이미지 의 인물이 주인공인 픽사 애니 메이션 스타일로 묘사할 수 있 는 AI 이미지 생성용 프롬프트 한 개를 번호 매김 없이 영어로 출력하세요. 배경에도 일상에서 보일수 있는 점을 디테일하게 신경 써서 설 명하세요."

순번	모델명	가격 (USD)	생성시간 (초)	생성이미지 1	생성이미지 2	생성이미지 3	생성이미지 4	생성이미지 5
1	omnigen-v2	0.15	16.59					
2	ideogram_v 2a_remix	0.04	8.99					
3	juggernaut- flux/base/im age-to- image	0		생성 실패 (3분 경과)	생성 실패 (3분 경과) 테스트 중지			
4	flux- kontext dev	0.025	8.1					

순번	모델명	가격 (USD)	생성시간 (초)	생성이미지 1	생성이미지 2	생성이미지 3	생성이미지 4	생성이미지 5
5-1	flux- pro/ <u>kontext</u>	0.04	7.25					LLM 에러
5-2	flux- pro/context (추가 진행)	0.04	7.25					
6	flux- pro/ <u>kontext</u> /max	0.08	9.37					
7	minimax/im age-01/ subject- reference	0.01	28.4					

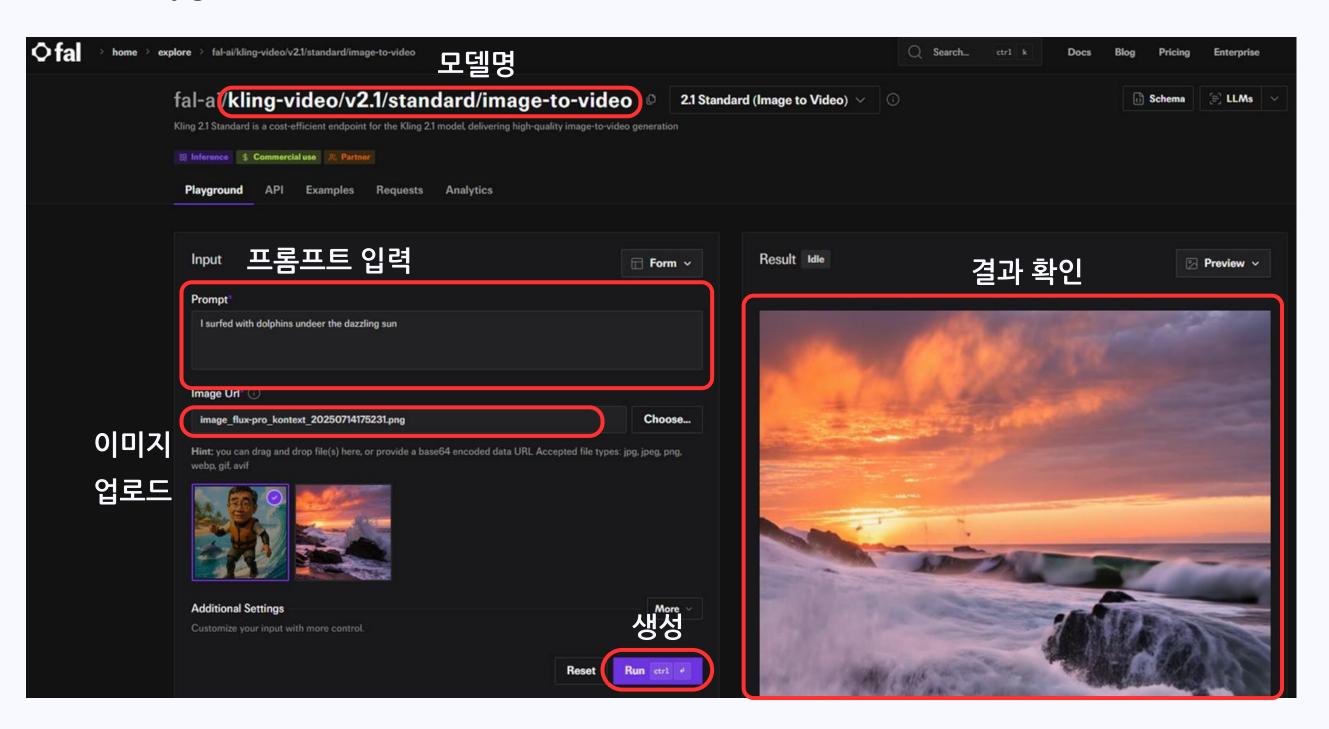
순번	모델명	가격 (USD)	생성시간 (초)	생성이미지 1	생성이미지 2	생성이미지 3	생성이미지 4	생성이미지 5
8	hidream-i1- full/	0.05	12.22					
9	hidream-e1- full	0.06	136.62					
10	imagen3/fas t	0.025	6.22					
11	imagen4/pr eview/fast	0.02	5.05					

순번	모델명	가격 (USD)	생성시간 (초)	생성이미지 1	생성이미지 2	생성이미지 3	생성이미지 4	생성이미지 5
12	stable- diffusion- v3-medium	0.05	6.34					

## ● 7.2. 테스트 과정 (Image to Video)

Image to Video 의 모델 테스트 과정은 다음과 같이 진행하였습니다.

fal.ai Playground 접속  $\rightarrow$  모델 선택  $\rightarrow$  프롬프트 입력  $\rightarrow$  이미지 업로드  $\rightarrow$  생성  $\rightarrow$  결과 확인



## ● 7.2. 테스트 과정 (Image to Video)

Image to Video 모델 테스트에는 3 가지 유형의 이미지를 사용했습니다. 각 이미지와 함께 입력한 프롬프트는 아래와 같습니다.

Test-01		Test-02		Test-03		
입력 이미지	입력 텍스트	입력 이미지	입력 텍스트	입력 이미지	입력 텍스트	
	"I surfed with dolphins under the dazzling sun."		"A women walks forward along a busy street."		"Woman in a black suit gently out to place her hand on the lowered head of an ancient dragon."	

# ● 7.2. 테스트 과정 (Image to Video) Test-01

모델명	wan-i2v	kling-video-1.6-pro	kling-video-v2.1- standard	kling-video-v2.1-pro	minimax-video-01	minimax-hailuo-02- standard
금액(USD)	0.4	0.475	0.25	0.45	0.5	0.27
생성시간(초)	44.04	189.67	61.65	72.52	164.87	209.16
모델명	minimax-hailuo-02- pro	pixverse-v4.5	pixverse-v4.5-fast	bytedance-seedance- v1-lite	bytedance-seedance- v1-pro	
금액(USD)	0.48	0.2	0.4	0.3707	0.6135	
생성시간(초)	346.40	33.68	46.07	117.64	70.52	

# ● 7.2. 테스트 과정 (Image to Video) Test-02

모델명	wan-i2v	kling-video-1.6-pro	kling-video-v2.1- standard	kling-video-v2.1-pro	minimax-video-01	minimax-hailuo-02- standard
금액(USD)	0.4	0.475	0.25	0.45	0.5	0.27
생성시간(초)	39.19	210.63	57.61	74.28	137.83	145.66
모델명	minimax-hailuo-02- pro	pixverse-v4.5	pixverse-v4.5-fast	bytedance-seedance- v1-lite	bytedance-seedance- v1-pro	
금액(USD)	0.48	0.2	0.4	0.3707	0.6135	
금액(USD) 생성시간(초)	-	0.2 48.99	0.4 60.21	0.3707 58.23	0.6135 69.48	

## ● 7.2. 테스트 과정 (Image to Video) Test-03

모델명	wan-i2v	kling-video-1.6-pro	kling-video-v2.1- standard	kling-video-v2.1-pro	minimax-video-01	minimax-hailuo-02 standard
금액(USD)	0.4	0.475	0.25	0.45	0.5	0.27
생성시간(초)	38.75	189.22	62.58	72.86	168.69	196.99
모델명	minimax-hailuo-02- pro	pixverse-v4.5	pixverse-v4.5-fast	bytedance-seedance- v1-lite	bytedance-seedance- v1-pro	
금액(USD)	0.48	0.2	0.4	0.3707	0.6135	
생성시간(초)	386.04	29.43	37.42	111.78	231.72	

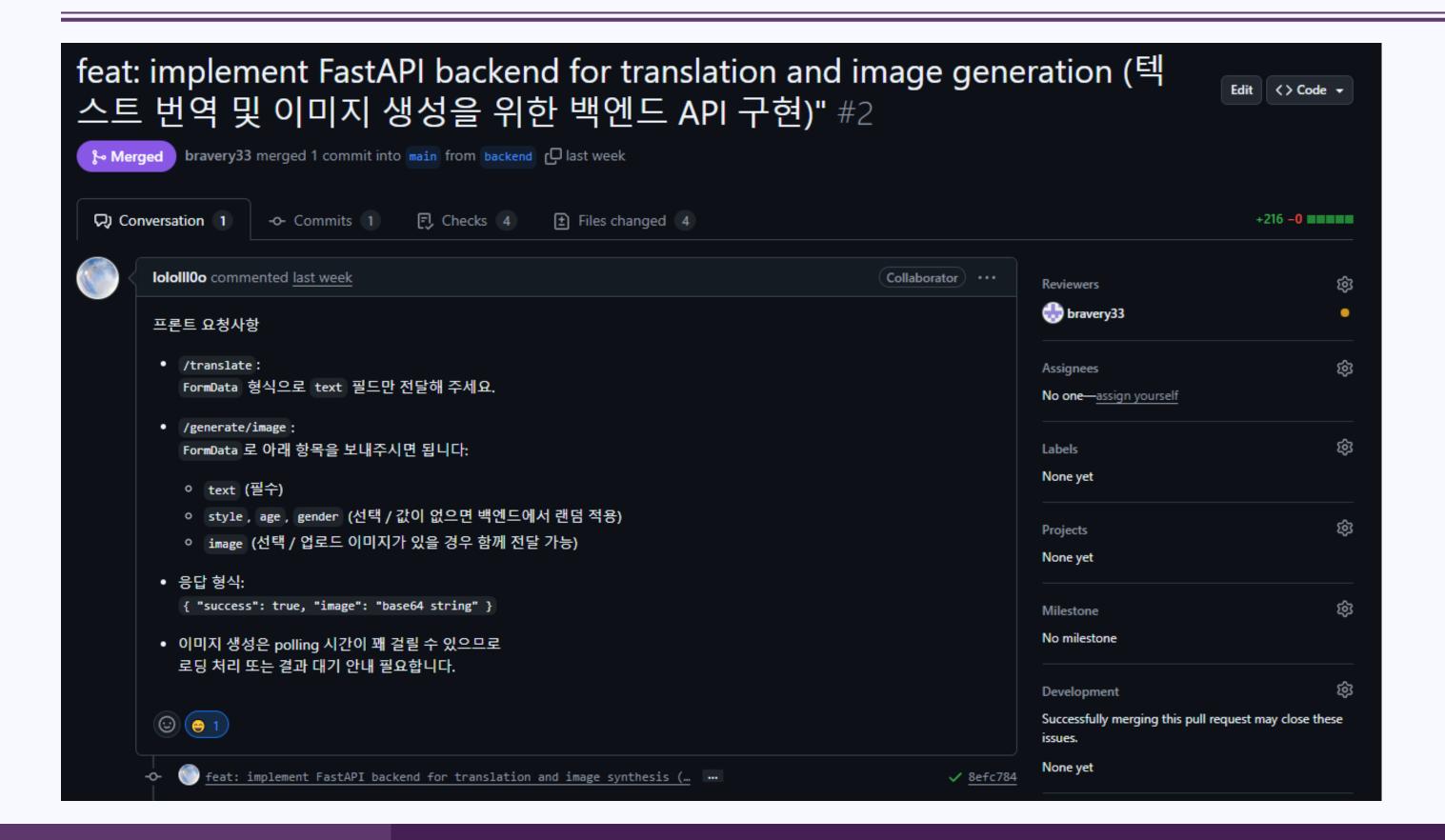
# ● 7.3. 연구 결과 및 최종 모델 선정

구분	선정 모델	선정 사유
Image- to- Image	FLUX Pro Kontext	품질: 사용자의 이미지를 기반으로 결과물을 생성할 때, 원본의 특징을 일관성 있게 유지하는 능력이 장점으로 고려 되었습니다. 속도/비용: 다른 고품질 모델 대비 합리적인 생성 속도와 비용을 보여주어, 서비스 운영에 가장 적합 하다고 판단했습니다. (속도: 7.25초, 비용: 0.04USD)
Image - to-	Kling Video 2.1	품질: 입력 이미지의 특징과 스타일을 일관성 있게 유지하면서도, 가장 자연스러운 움직임을 보여줬습니다.
Video	Standard	속도/비용: 경쟁 모델 대비 빠른 생성 속도와 합리적인 비용을 제공하였습니다. (속도: 60.6초, 비용: 0.25USD)

# ● 8. 형상관리 및 배포 (Sourcetree)

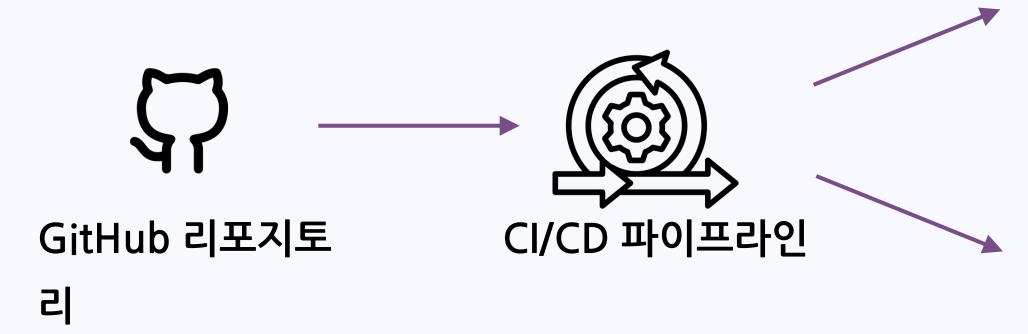
그래프	설명	날짜	작성자	커밋
0	O D main Porigin/main Merge pull request #10 from bravery33/fix/services	21 7 2025 16:28	bravery33 < diddung83@gmail.com>	3b92691
•	🚺 origin/fix/services   refactor: redesign image generation prompts by style and apply gender-aware negative prompts (스타일별 이미지 생성 프롬프트 재구성 및 성별 기반 부정 키워드 적용)	21 7 2025 16:23	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	efbf249
•	Merge pull request #9 from bravery33/Front_End	21 7 2025 16:16	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	f0d2e89
1 +	origin/Front_End feat(api): Enhance API service and implement structured error logging	21 7 2025 16:12	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	3894dd6
	chore: specify openai version explicitly in requirements.txt(requirements.txt에서 openai 버전을 명확히 기재)	21 7 2025 10:45	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	bf1513d
•	fix: remove BackEnd/ from .gitignore to re-enable backend tracking(gitignore에 적힌 backend몰더 삭제)	21 7 2025 10:43	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	ed37c87
•	15 origin/backend chore: update openal version in requirements.txt to latest(openal 버전을 최신으로 수정함)	21 7 2025 10:13	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	b9f3958
÷ l	fix(download): Stabilize file download using Blob(Blob을 사용하여 파일 다운로드 기능을 안정화함)	19 7 2025 18:22	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	1b9893e
+	feat(api): API 서비스 업데이트 및 정적 에셋 교체	19 7 2025 14:03	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	3bb7980
÷	Add sessionID header and image-to-video flow(세션ID 헤더 처리 및 이미지→비디오 플로우 구현)	18 7 2025 21:38	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	87cd460
· ·	Merge branch 'main' of https://github.com/bravery33/FalseCam into backend	18 7 2025 14:55	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	018a4ce
	백엔드 작업 중, 아직 불안정한 상태	18 7 2025 12:15	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	73be6f1
ė l	로고, 세션ID 추가	18 7 2025 11:57	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	805777d
÷ l	로고추가	18 7 2025 11:57	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	1676ef8
•	Merge pull request #8 from bravery33/backend	17 7 2025 22:08	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	fea5cba
<del> </del>	feat: implement face fusion using fal.ai and update requirements (fal.ai 기반 얼굴 합성 기능 및 의존성 추가)	17 7 2025 22:05	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	76f050e
÷ I	feat: implement sequential video creation following image generation (이미지 생성 후 순차적으로 비디오 생성 처리 기능 구현)	17 7 2025 18:40	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	d4516b6
-	efactor(ui): Improve image gallery UX and thumbnail handling	17 7 2025 18:09	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	f1d1b84
	Merge pull request #7 from bravery33/backend	17 7 2025 16:13	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	a715dc0
L.	Merge branch 'main' into backend	17 7 2025 16:12	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	837cca0
•	chore: add health check endpoint for Render deployment (/healthz 추가)	17 7 2025 15:58	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	4834ebf
•	chore: update backend API URL to deployed Render endpoint (백엔드 API 주소를 배포된 Render 주소로 변경)	17 7 2025 15:54	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	80423a6
	chore: install react-select for InfoInputCard component (InfoInputCard에서 사용하는 react-select 패키지 설치)	17 7 2025 15:44	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	63d3098
•	feat: apply frontend UI enhancements and include required image assets (프론트 UI 개선 및 이미지 리소스 포함)	17 7 2025 15:42	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	dd89162
	fix: patch style prompt fallback, refine style descriptions, and apply File() to image param (화풍 랜덤 오류 수정, 스타일 프롬프트 문장 정리, image 파라미터 File 처리)	17 7 2025 15:37	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	a711188
	1 origin/sms Fix: Update API endpoint (API 엔드포인트 수정)	17 7 2025 2:58	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	71ff52c
	chore: update API base URL for frontend-backend integration (프론트-백엔드 연동용 API 주소 수정)	17 7 2025 1:33	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	4668116
•	add health check endpoint(서버 상태 확인 엔드포인트 추가)	17 7 2025 1:08	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	dc90269
•	Merge pull request #4 from bravery33/backend	16 7 2025 19:34	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	1302284
•	feat: apply style-aware prompt logic based on Korean-to-English translation (한국어 번역 결과에 따라 스타일 프롬프트를 다르게 처리하는 로직 구현)	16 7 2025 19:16	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	16fbc58
•	enhance: image modal now displays properly scaled preview	16 7 2025 18:49	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	4b4e01b
÷ l	fix: thumbnail not rendering on new uploads - adjusted src with hardcoded path (임시 대응)	16 7 2025 18:13	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	b123082
	ui: redesign vlog slider indicators with glassmorphism, spacing fix, and overflow protection (UI: 브이로그 슬라이더 인디케이터를 글라스모피즘 스타일로 리디자인하고 간격 및 넘침 현상 수정)	16 7 2025 17:06	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	9cb5742
	Merge pull request #3 from bravery33	16 7 2025 1:33	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	4da71cc
i i	fix: Fix image generation quality and option mapping (이미지 생성 퀄리티 및 옵션 매핑 오류 수정)	16 7 2025 1:20	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	91cfba2
i i	feat: successful backend and frontend integration (백엔드-프론트엔드 연동 및 첫 실행 성공)	16 7 2025 0:13	Min Suk <diddung83@gmail.com></diddung83@gmail.com>	9b82b25
+<	Merge pull request #2 from bravery33/backend	15 7 2025 17:52	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	7e638f9
•	feat: implement FastAPI backend for translation and image synthesis (번역 및 이미지 생성을 위한 FastAPI 백엔드 구현)	15 7 2025 17:37	lololll0o <lololll0o0808@gmail.com></lololll0o0808@gmail.com>	8efc784
•<	Merge pull request #1 from bravery33/feature/add-FE	15 7 2025 15:23	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	a97abff
•	☑ origin/feature/add-FE      feat(falsecam-project): implement initial Home UI and layout (초기 홈 UI 및 레이아웃 구현)	15 7 2025 15:10	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	076a11e
ę	Dorigin/feature/front_end feat: 줄바꿈 경고 무시하고 첫 커밋	14 7 2025 18:59	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	000caa4
	read.md 수정	14 7 2025 18:36	73962869 <taeyeol2483@gmail.com></taeyeol2483@gmail.com>	644b9e6
	first commit	9 7 2025 18:38	bravery33 <diddung83@gmail.com></diddung83@gmail.com>	a35a0c6

## ● 8. 형상 관리 및 배포 (Github Pull Request)



### ● 8. 형상 관리 및 배포

프로젝트는 GitHub을 중심으로 프론트엔드와 백엔드가 각각 Cloudflare Pages와 Render를 통해 자동 배포됩니다. CI/CD 파이프라인을 통해 코드 변경 사항이 즉시 서비스에 반영됩니다.



### 프론트엔드 배포



React 프로젝트는 GitHub 리포지토리의 main 브랜치와 연동되어 Cloudflare Pages에 배포됩니다. main 브랜치 에 푸시될 때마다 자동으로 빌드 및 배포가 트리거됩니다. URL: <a href="https://falsecam.pages.dev/">https://falsecam.pages.dev/</a>

### 백엔드 배포

**₽** Render

FastAPI 서버는 GitHub 리포지토리와 연동되어 Render에 배포됩니다. 마찬가지로 자동 배포가 설정되어 있으며, 민감한 API 키는 Render의 환경 변수 기능을 통해 소스코드와 분리하여 안전하게 관리됩니다.

URL: <a href="https://falsecam.onrender.com/">https://falsecam.onrender.com/</a>

## ● 9. 회의록 (요약)

2차 회의 (25.07.10)

WBS를 구체화하고 팀원별 태스크를 분배했습니다. 와이어프레임 설계 및 기술 스택을 확정했습니다.

8차 회의 (25.07.18)

익명 세션 관리 방식(sessionID) 적용을 확정하고, 이미지 생성 후 비디오 자동 생성 플로우 구현에 대해 논의했습니다.

1차 회의 (25.07.09)

팀명('주석열'), 프로젝트 주제('FalseCam') 및 전체 일정을 확정했습니다. 초기 환경 설정 (GitHub) 및 WBS 초안 작성을 완료했습니다.

5차 회의 (25.07.15)

백엔드 API 명세를 구체화하고, 프론트엔드 공통 컴포넌트(모달, 버튼) 설계에 대해 논의했습니다.

10차 회의 (25.07.23)

논의했습니다

MVP 기능 통합 테스트 및 발견된 버그(UI, API 연동)를 리뷰했습니다. 피드백 기반 개선 사항을

## ♥ 9. 회의록 (원본)

#### 25.07.14 4차 회의

(프론트엔드/백엔드 개발 착수 및 핵심 모델 선정)

#### [주요 결정 사항]

• Al 모델 선정: Image-to-Image 기능 구현을 위해 flux-pro/kontext 모델을 최종 선정

#### [금일 완료된 업무]

- 프론트엔드
  - o React 프로젝트 초기화 및 초기 환경 설정을 완료
  - 。 공통 UI 컴포넌트(버튼, 입력창 등) 개발을 시작
- 백엔드 & 인프라
  - o Cloudflare 프로젝트를 생성하고 연동을 완료
  - o 선정된 모델을 기반으로 Image-to-Image 생성 API 개발을 시작
- AI모델
  - 。 주요 Image-to-Image 모델에 대한 테스팅을 완료하고 결과물 비교를 완성

#### [다음 진행 예정]

- 공통 UI 컴포넌트 개발을 지속하고, 이를 활용하여 핵심 기능 UI 개발을 본격화
- Image-to-Image API 개발을 계속하고, 선정된 'flux-pro/kontext' 모델과의 연동을 준비
- 프론트엔드와 백엔드 간의 데이터 통신을 위한 초기 API 연동을 테스트

#### [기타 특이사항]

- Image-to-Image 모델 테스팅 과정과 결과물을 유튜브 영상으로 기록 및 공유 https://youtu.be/MZYFLnbvYDo
- 깃허브 URL: <a href="https://github.com/bravery33/FalseCam">https://github.com/bravery33/FalseCam</a>

#### 25.07.17 7차 회의

(이미지-비디오 연동 및 세션 관리 설계)

#### [금일 완료된 업무]

#### • 배포

백엔드(Rander)와 프론트엔드(Cloudflare) 각각 연결 및 배포 완료 양쪽 연동을 통해 서비스 전체가 성공적으로 배포됨

#### • 프론트엔드:

UI/UX 개선 상세 작업

초기 썸네일 없이 imageList를 빈 배열로 시작하도록 처리

imageList가 비었을 때 undefined 에러 방지를 위한 안전 처리 추가

imageList 구조를 {src, type} 객체 형식으로 통일

이미지 접근 방식을 항상 imageList[currentIndex].src로 일원화

이미지가 없을 때 "no vlog yet" 안내문구 출력

썸네일/미리보기 이미지를 object-cover에서 object-contain으로 변경해 전체 이미지가 잘리지 않게 처리

썸네일 배경을 투명하게 변경하여 더 깔끔하게 개선

업로드 미리보기 이미지에도 object-contain 적용

미리보기 모달에서 로딩 오버레이 제거

새 이미지 생성 시 currentindex를 0으로 자동 설정(새 이미지가 항상 선택되게)

썸네일 인디케이터 점(도트) 스크롤을 currentIndex에 맞춰 자동 이동 처리

모달 좌우 버튼을 카드와 동일한 삼각형 버튼(채운 형태)으로 변경

#### 백엔드:

프롬프트 개선 작업 및 이미지 생성 → 비디오 생성 자동 연동 흐름 설계 시작

## 10. 자체 평가 의견

- 10.1. 자체 평가
- 10.2. 잘한 점과 아쉬운 점
- 10.3. 추후 개선 사항
- 10.4. 소감

# ● 10.1. 자체평가

순번	항목	서민석	김주현	최태열
1	소프트웨어 요구사항은 소프트웨어 구조에 반영되어 있는가?	9	9	8
2	모듈은 기능적으로 독립되어 있는가?	8	9	8
3	자료구조는 소프트웨어 요구사항과 일치하는가?	8	8	9
4	지역 자료구조조는 적절하게 정의 되었는가?	9	8	9
5	알고리즘은 원하는 기능을 수행하고 있는가?	9	8	9
6	알고리즘은 논리적으로 정확한가?	9	9	8
7	유지보수성은 고려되었는가?	7	8	7

### ● 10.2. 잘한 점과 아쉬운점

### 잘한 점

- 적극적인 신기술 도입 및 문제 해결: 다양한 외부 AI 모델(GPT, Fal-ai)의 API를 성공적으로 연동했으며, 이 과정에서 발생한 CORS, 비동기 처리, base64 인코딩, Blob을 이용한 다운로드 등 실제 웹 개발에서 마주할 수 있는 여러 기술적 난제를 성공적으로 해결했습니다.
- 효율적인 협업 및 버전 관리: GitHub를 중심으로 명확한 커밋 컨벤션을 적용하고, WBS에 따른 체계적인 일 정 관리를 통해 원활하게 협업했습니다. CI/CD 파이프라인을 구축하여 개발자가 코드에만 집중할 수 있는 효율적인 개발 문화를 만들었습니다.

### 아쉬운 점

- 최신 기술의 적극적인 도입 및 학습: FastAPI, React 등 최신 웹 기술 스택을 성공적으로 도입했으며, 특히 여러 시부 생성형 AI 모델의 API를 연동하고 비교 분석하는 과정에서 프롬프트 엔지니어링과 비동기 처리 등 최신 기술에 대한 깊은 이해와 경험을 쌓을 수 있었습니다.
- 사용자 경험(UX) 중심의 기능 개선: MVP 배포 후, 실제 **사용자 피드백을 적극적으로 수용**하여 UI/UX를 개선 했습니다. 로딩 애니메이션 추가, 입력 옵션 구체화 등 사용자의 목소리를 반영한 빠른 개선 작업은 서비스의 완성도를 높이는 데 크게 기여했습니다.

## ● 10.3. 추후 개선 사항

### 회원 시스템 도입

정식 회원가입 및 로그인 기능을 추가하여 사용자별 개인화된 서비스와 안정적인 세션 관리를 제공합니다.

### 마이페이지 저장 기능

사용자가 생성한 이미지와 비디오 결과물을 마이페이지에 저장하고 관리할 수 있도록 하여, 언제든지 다시 확인하고 활용할 수 있게 합니다.

### 고급 커스터마이징 옵션

사용자가 AI 모델의 스타일, 세부 프롬프트 등 생성 파라미터를 더욱 정교하게 제어할 수 있는 고급 설정 기능을 추가합니다.