FalseCam Project
최종 결과 보고서

AI 기반 추억 생성 서비스

배포	GitHub
https://falsecam.pages.dev	https://github.com/bravery33/FalseCam

팀명	주석열
팀장	서민석
팀명	최태열, 김주현

목차페이지

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 : AI 기반 '경험 재구성' 서비스

선정 배경

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

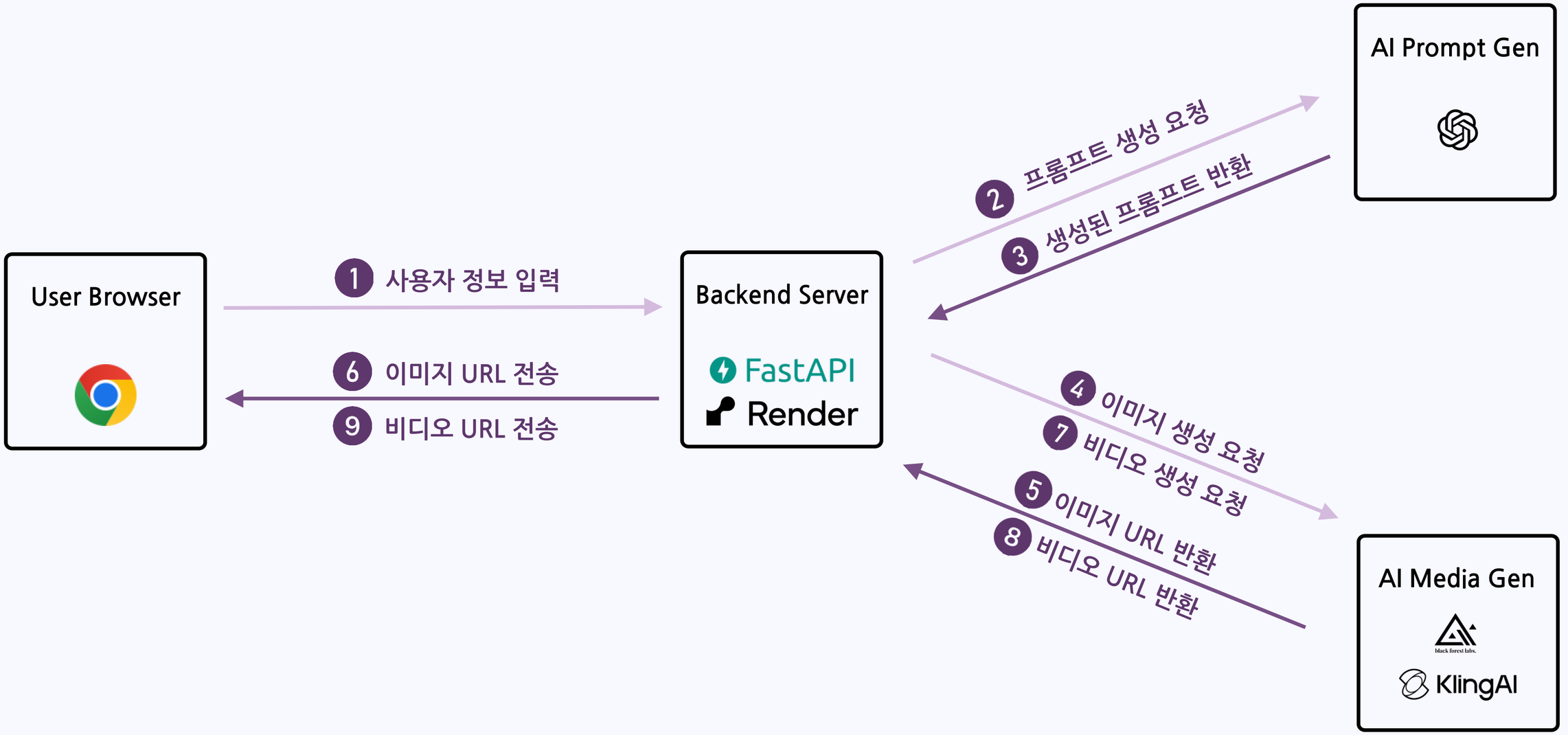
기획 의도

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.

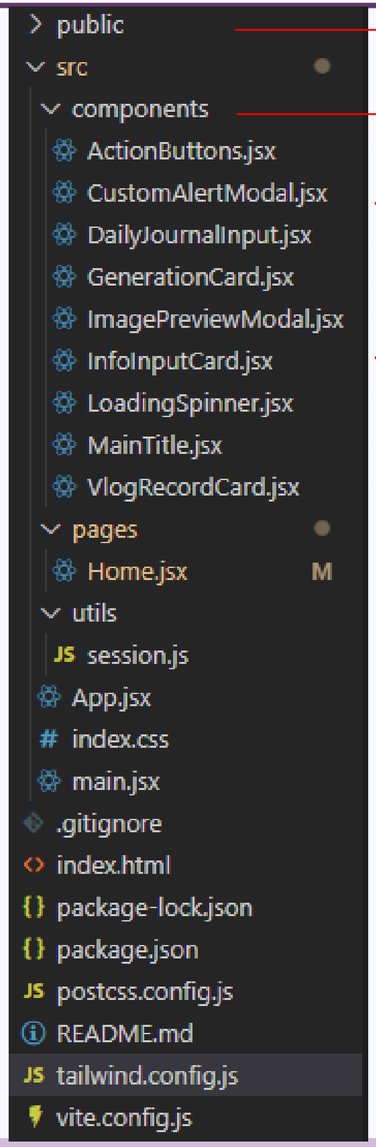
1.2. 활용 기술 및 개발 환경

구분	기술 스택
Frontend	React, Vite, JavaScript, Tailwind CSS, Headless UI
Backend	Python, FastAPI, Uvicorn
AI 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)



이미지, 영상 등 정적 파일 리소스

프론트 핵심 로직 (컴포넌트, 페이지, 라우팅, 유틸 등)

함수 실행 버튼 (다운로드, 트리거용)

경고 또는 안내 팝업 (조건 안 맞을 때 등)

오늘 하루 일기를 입력받는 텍스트 입력 필드

이미지 업로드 + “오늘 하루 소환” 버튼 포함 생성 인터페이스

썸네일 클릭 시 전체 보기 모달 (좌우 넘기기 포함)

스타일, 나이, 성별을 선택하는 옵션카드

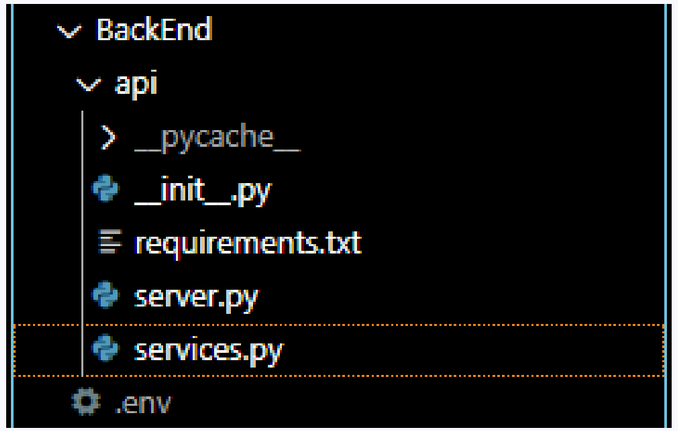
로딩 중 애니메이션 및 진행바 표시

메인 문구 타이틀 영역 출력 (로고 아래 문장 포함)

생성된 이미지/영상을 썸네일 리스트로 보여주는 카드

전체 페이지 구성의 중심
사용자 입력 → 이미지/영상 생성 → 결과 표시까지 전 과정을 연결

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 공유하다 보면, 자연스럽게 나만의 디지털 페르소나가 만들어집니다.

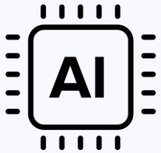
딱딱한 기록 대신, 쉽고 빠르게 내 개성을 마음껏 뽐낼 수 있는 새로운 디지털 놀이터가 열립니다.

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 설계되었습니다.

복잡한 설명이나 기술 지식 없이도, 직접 텍스트나 이미지를 넣고 결과를 바로 확인할 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

Work Break-down Structure								7월																														
								2주							3주							4주							5주/1주										
Procedures (단계)	Steps (절차)	Tasks(작업)	담당자	Schedule		Goal(목표)	산출물/비고(Deliverable)	화	수	목	금	토	일	월	화	수	목	금	토	일	월	화	수	목	금	토	일	월	화	수	목	금	토	일				
				시작일	종료일			8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3				
1.0.0. 착수 및 기획				07월 09일	07월 11일	프로젝트 범위 및 실행 계획 확정	최종 기획서, WBS																															
	1.1.0 기획 및 설계																																					
		1.1.1. WBS 수립 및 기능 요구사항 정의	서민석	07월 09일	07월 09일	개발 범위와 규칙의 명확화	WBS 초안, 기능명세서																															
		1.1.2 와이어프레임 및 화면 설계	최태열	07월 10일	07월 11일	사용자 경험(UX) 시각화	와이어프레임																															
		1.1.3 기술 스택 확정 및 아키텍처 설계	김주현	07월 10일	07월 11일	안정적인 기술 구조 설계	아키텍처 설계도																															
	1.2.0 프로젝트 관리																																					
		1.2.1. 상세 WBS 및 기획서 작성	서민석	07월 10일	07월 11일	효율적인 협업 체계 구축	WBS, 기획서																															
		1.2.2. Git 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																															
		2.1.2. React 프로젝트 초기화 및 초기 환경 설정	최태열	07월 14일	07월 14일	프론트엔드 개발 환경 구축	초기화된 FE 프로젝트 구조																															
		2.1.3. Model Testing(img-to-img)	서민석	07월 14일	07월 14일	프로젝트와 적합한 img-to-img model 찾기	N/A																															
		2.1.3. Model Testing(vid-to-vid)	서민석	07월 15일	07월 16일	프로젝트와 적합한 vid-to-vid model 찾기	N/A																															
	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																															
		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																															
	2.3.0 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																															
	2.4.0 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																																					
		3.1.1. Image-to-Video API 연동	김주현	07월 21일	07월 24일	콘텐츠의 매력도 증대	Image-to-Video API																															
			서민석	07월 21일	07월 24일																																	
	3.2.0 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																															
		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 기능																															
	3.3.0 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);

  try {
    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.

```python
@router.post("/generate/image")
async def generate_image(
    text: str = Form(...),
    style: str = Form(""),
    age: str = Form(""),
    gender: str = Form(""),
    image: Optional[UploadFile] = File(None),
    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."}
        )
```

```python
        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"❌ 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, 반환된 이미지 URL과 프롬프트를 사용하여 즉시 비디오 생성 API(/generate-video)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 프롬프트를 받아 Kling 비디오 모델을 호출합니다.

```python
@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)}
  >
    <div
      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
          loop
          className="object-contain w-full h-full rounded-xl shadow-2xl"
```

4.4. 익명 세션 기반 사용자 관리(프론트엔드)

- 회원가입 없이도 사용자를 식별하고 연속적인 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 생성하여 HTTP 요청의 Header에 포함시켜 전송합니다.
- 서버는 FastAPI의 Header 의존성을 통해 해당 세션 ID를 수신하고, 요청을 세션 단위로 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.

```python
@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,
    }
```

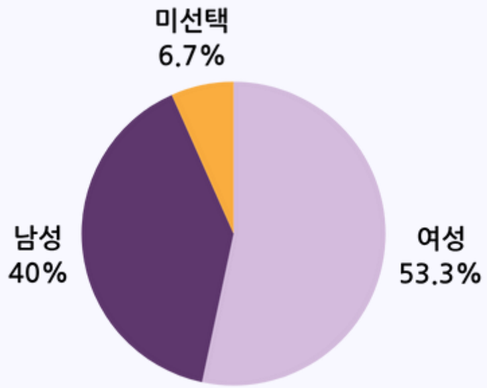
5. 사용자 피드백

피드백 참여자 분석

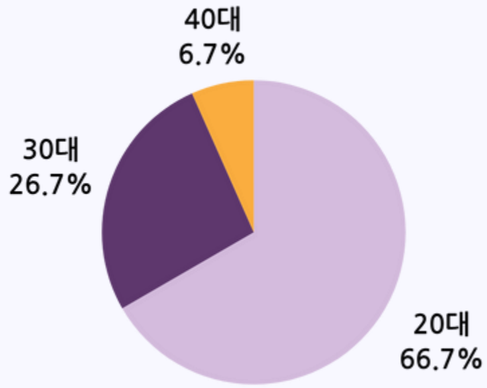
총 응답자

15명

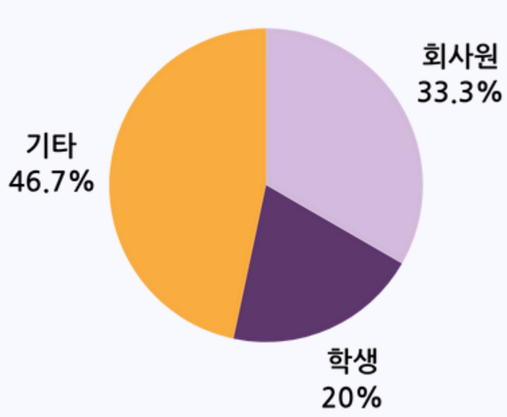
성별 분포



연령 분포



직업 분포



5. 사용자 피드백

15명의 목소리

좋았어요	아쉬웠어요	이런 건 어때요?
가볍게 재미로 하기 좋은 앱 같아요	영상 생성속도가 느리다	화풍이 더 추가되었으면 좋겠다
깔끔하고 트렌디하다	예시 멘트가 추상적입니다	친구들에게 공유 기능
단순해서 간단하게 이용하기 좋습니다	어디에 기입해야 하는지 한눈에 파악이 어려웠습니다	그림일기 형식처럼 나오면 더 재미있을 듯 하다
문구들이 센스가 넘쳐서 재미있었습니다	너무 단조롭고 심플합니다	하루에 대한 피드백이나 운세
간단하고 직관적이라 좋았습니다	선택칸의 간격이 좁아서 성별을 선택할 수 없어요	영상 길이를 선택할 수 있으면 좋을 것 같습니다
색이 이쁘고, 사용하기 편리했어요	가끔씩 영상의 움직임이 자연스럽지 못해요	사진이나 동영상 중 하나만 생성되는 기능 추가

5. 사용자 피드백

사용자 불편사항

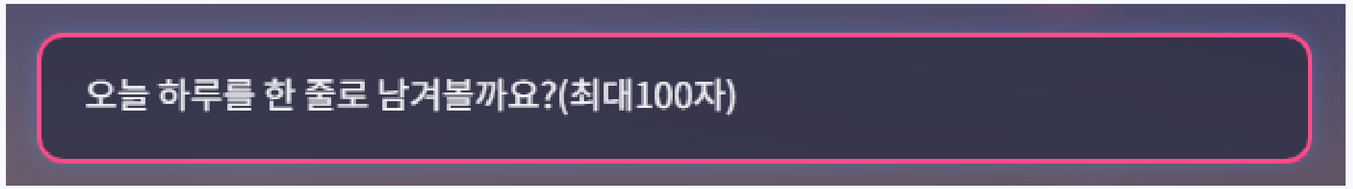
이미지와 영상을 생성하시면서 불편하거나 아쉬웠던 점이 있으셨나요?

프롬프트 예시가 있으면 좋을 것 같습니다

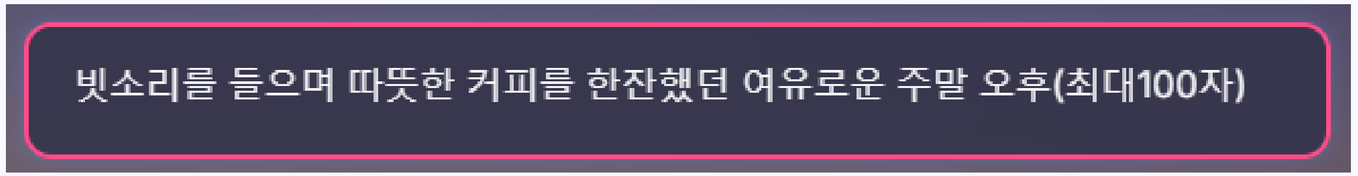
이미지와 영상을 생성하시면서 불편하거나 아쉬웠던 점이 있으셨나요?

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

플레이스 홀더(전)



플레이스 홀더(후)



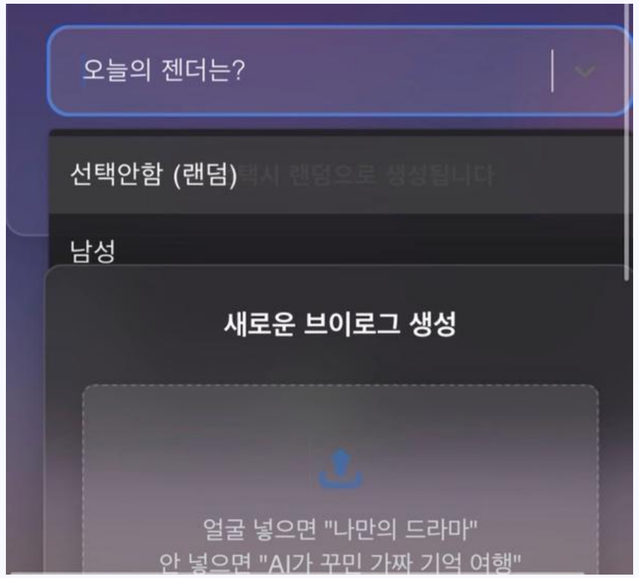
5. 사용자 피드백

사용자 불편사항

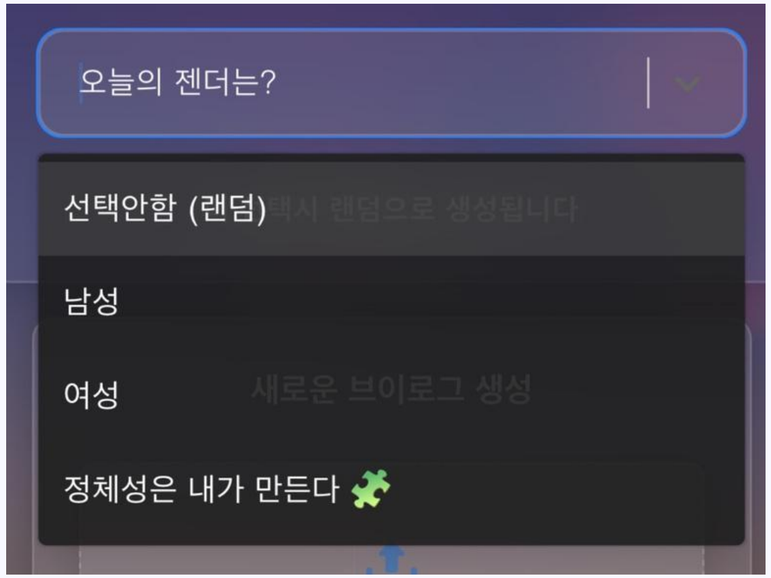
FalseCam의 전체적인 디자인과 UI(화면 구성)는 어땠나요? 느낀 점을 자유롭게 적어주세요.

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 표시되지 않는 문제가 발생했습니다. 코드는 정상적으로 실행되는 것처럼 보여 디버깅에 오랜 시간이 소요되었습니다.

원인

문제의 핵심은 CSS의 z-index와 position 속성이었습니다. 모달 컴포넌트는 렌더링되었지만, z-index 속성이 누락되어 다른 컴포넌트들 뒤에 가려진 상태로 출력되고 있었습니다.

```jsx
// 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">
        <p>이미지 미리보기</p>
      </div>
    </div>
  );
}
```

해결 과정

모달 컴포넌트의 최상위 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.

```jsx
// FrontEnd/src/components/ImagePreviewModal.jsx (수정 후)
return (
  <div
    // z-index와 position 속성을 추가하여 모달이 최상단에 표시되도록 수정
    className="fixed inset-0 z-50 bg-black/80 flex items-center justify-center"
    onClick={() => setIsOpen(false)}
  >
    <div
      className="relative w-auto h-full max-h-[90vh]"
      onClick={(e) => e.stopPropagation()}
    >
      {/* ... Modal Content ... */}
    </div>
  </div>
);
```

6.2. Blob을 이용한 외부 파일 다운로드 문제

문제

AI가 생성한 이미지나 비디오는 외부 서버에 저장되는데, 이 결과물을 다운로드하려고 하면 파일이 저장되지 않고 새 탭에서 열리기만 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 작동하지 않았습니다.

```jsx
// currentItem.src가 외부 AI 서버의 URL이라고 가정
// 문제: 이 링크를 클릭하면 다운로드 대신 이미지/비디오가 열림
const DownloadButton = ({ currentItem }) => (
    <a
        href={currentItem.src}
        download={`falsecam-image-${new Date().getTime()}.jpg`}
        className="..."
    >
        다운로드
    </a>
);
```

해결 과정

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

```jsx
// FrontEnd/src/components/ActionButtons.jsx (수정 후)

const ActionButtons = ({ currentItem }) => {

    const handleDownload = async () => {
        try {
            // 1. fetch로 파일 데이터를 가져와 Blob으로 변환
            const response = await fetch(currentItem.src);
            const blob = await response.blob();

            // 2. Blob으로 임시 URL 생성
            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`;
            link.click();

            // 4. 메모리 해제를 위해 임시 URL 폐기
            URL.revokeObjectURL(objectUrl);
        } catch (error) {
            console.error("다운로드 실패:", error);
        }
    };

    return (
        <button onClick={handleDownload}>
            다운로드
        </button>
    );
};
```

6.3. Base64 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.

```python
@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, ...}
    )
```

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.

```python
@router.post("/generate-image")
async def generate_image(
    image: UploadFile | None = File(None),
    ...
):
    image_data_url = None
    if image:
        image_data = await image.read()
        image_mime_type = image.content_type

        # 해결: 이미지 데이터를 base64로 인코딩하여 data URL 형식으로 변환
        encoded_image = base64.b64encode(image_data).decode("utf-8")
        image_data_url = f"data:{image_mime_type};base64,{encoded_image}"

    # ...

    # 정상적으로 변환된 data URL을 전달
    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.

```
    messages = [
        {"role": "system",
         "content": "Translate the following Korean text to natural English for an
         image generation prompt."},
        {"role": "user", "content": text}
    ]
```

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 프롬프트를 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 우수한 결과물을 생성하는가?

속도

사용자가 지루함을 느끼지 않을 합리적인 시간 내에 결과물을 생성하는가?

비용

서비스의 지속 가능한 운영을 고려했을 때, 생성 비용이 합리적인가?

7.2. 테스트 과정 (Image to Image)

Image to Image 모델 테스트는 python 코드로 실행했습니다.

```python
IMAGE_MODEL_NAME = "flux-pro/kontext"
safe_image_model_name = IMAGE_MODEL_NAME.replace("/", "_")

OUTPUT_DIR = "output"
os.makedirs(OUTPUT_DIR, exist_ok=True)

local_path = "samples/sample_image.png"
image_url = fal_client.upload_file(local_path)

def get_timestamp():
    return datetime.now().strftime("%Y%m%d%H%M%S")

def generate_image_prompt_from_daily_life(daily_text: str) -> str:
    handler = fal_client.submit(
        "fal-ai/any-llm",
        arguments={
            "model": "google/gemini-flash-1.5",
            "system_prompt": (
                "당신은 창의로운 이미지 프롬프트 제너레이터입니다. "
                "사용자가 적은 평범한 일상 텍스트와 제공된 이미지를 받아, "
                "평범했던 일상을 환상적으로 변형시키고, "
                "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"]

def generate_image_from_prompt(image_url: str, prompt: str) -> str:
    print(f"🚀 Image-to-Image 요청 payload: {{'image_url': image_url, 'prompt': prompt}}")
    resp = fal_client.run(
        f"fal-ai/{IMAGE_MODEL_NAME}",
        arguments={
            "image_url": image_url,
            "prompt": prompt
        }
    )
    return resp["images"][0]["url"]
```

1.모델명 지정

2.입력이미지 선택

3.입력 프롬프트

4.시스템 프롬프트

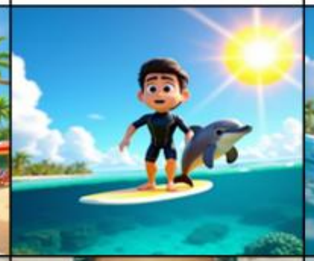
5.이미지 생성 요청

7.2. 테스트 과정 (Image to Image)

입력한 이미지와 프롬프트는 아래와 같습니다.

입력 이미지	입력 텍스트	시스템 프롬프트
	"눈부신 태양 아래서 돌고래와 같이 서핑을 했어"	“당신은 창의로운 이미지 프롬프트 제너레이터입니다. 사용자가 적은 평범한 일상 텍스트와 제공된 이미지를 받아, 평범했던 일상을 제3자가 찍은 파파리치샷으로 변형시키고, 얼굴이 보일 수 있는 구도로 , 의상도 상황에 맞게 변형시켜 주고, 성별은 구분하지 말고, 이미지의 인물이 주인공인 픽사 애니메이션 스타일로 묘사할 수 있는 AI 이미지 생성용 프롬프트 한 개를 번호 매김 없이 영어로 출력하세요. 배경에도 일상에서 보일수 있는 점을 디테일하게 신경 써서 설명하세요.”

7.2. 테스트 과정 (Image to Image)

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_v2a_remix	0.04	8.99					
3	juggernaut-flux/base/image-to-image	0		생성 실패 (3분 경과)	생성 실패 (3분 경과) 테스트 중지			
4	flux-kontext_dev	0.025	8.1					

7.2. 테스트 과정 (Image to Image)

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

7.2. 테스트 과정 (Image to Image)

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/fast	0.025	6.22					
11	imagen4/preview/fast	0.02	5.05					

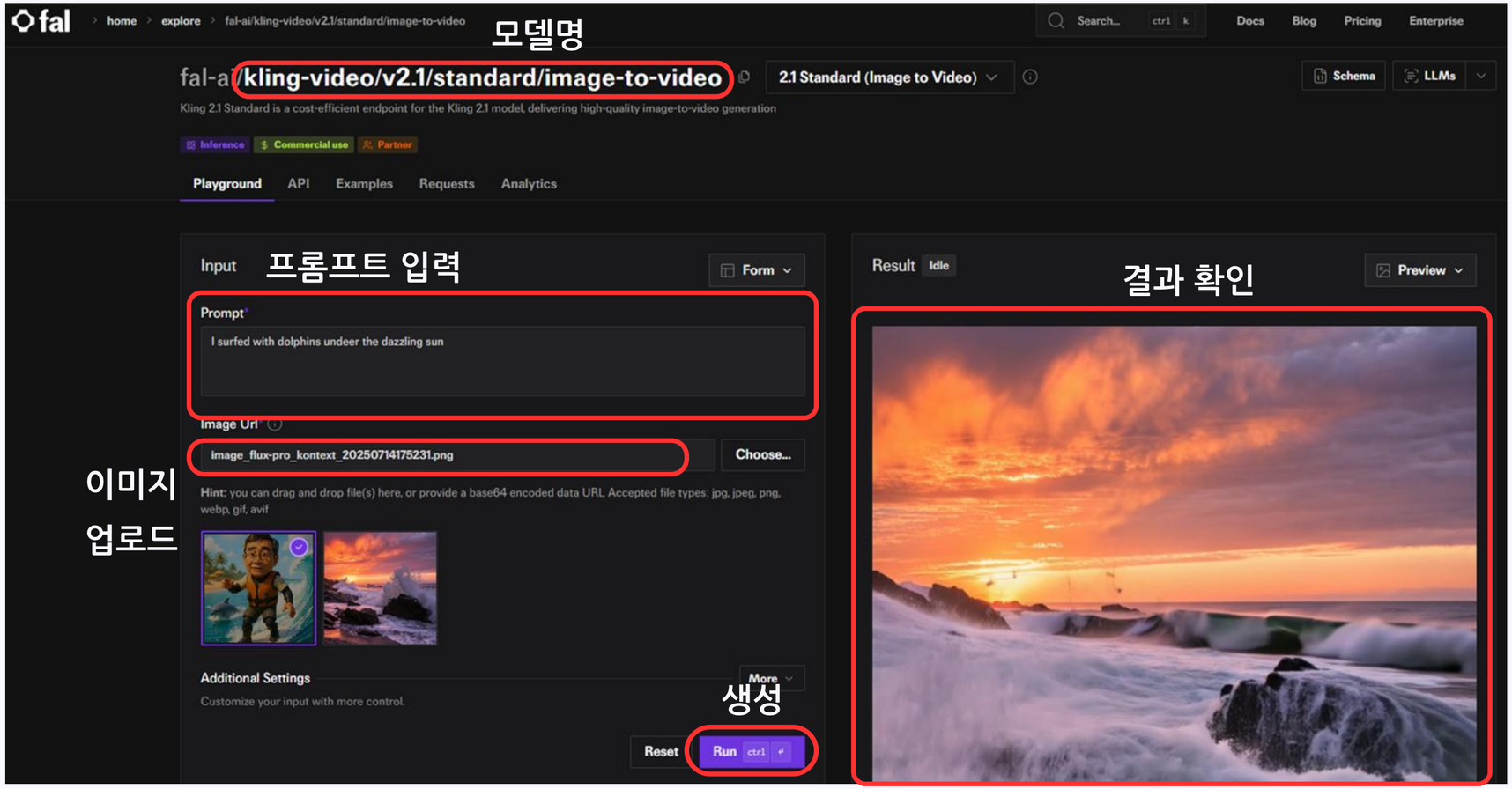
7.2. 테스트 과정 (Image to Image)

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)

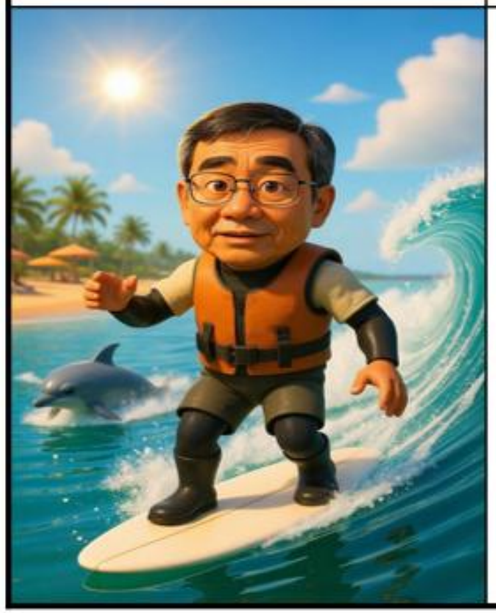
Image to Video 의 모델 테스트 과정은 다음과 같이 진행하였습니다.

fal.ai Playground 접속 → 모델 선택 → 프롬프트 입력 → 이미지 업로드 → 생성 → 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	
생성시간(초)	232.22	48.99	60.21	58.23	69.48	
						

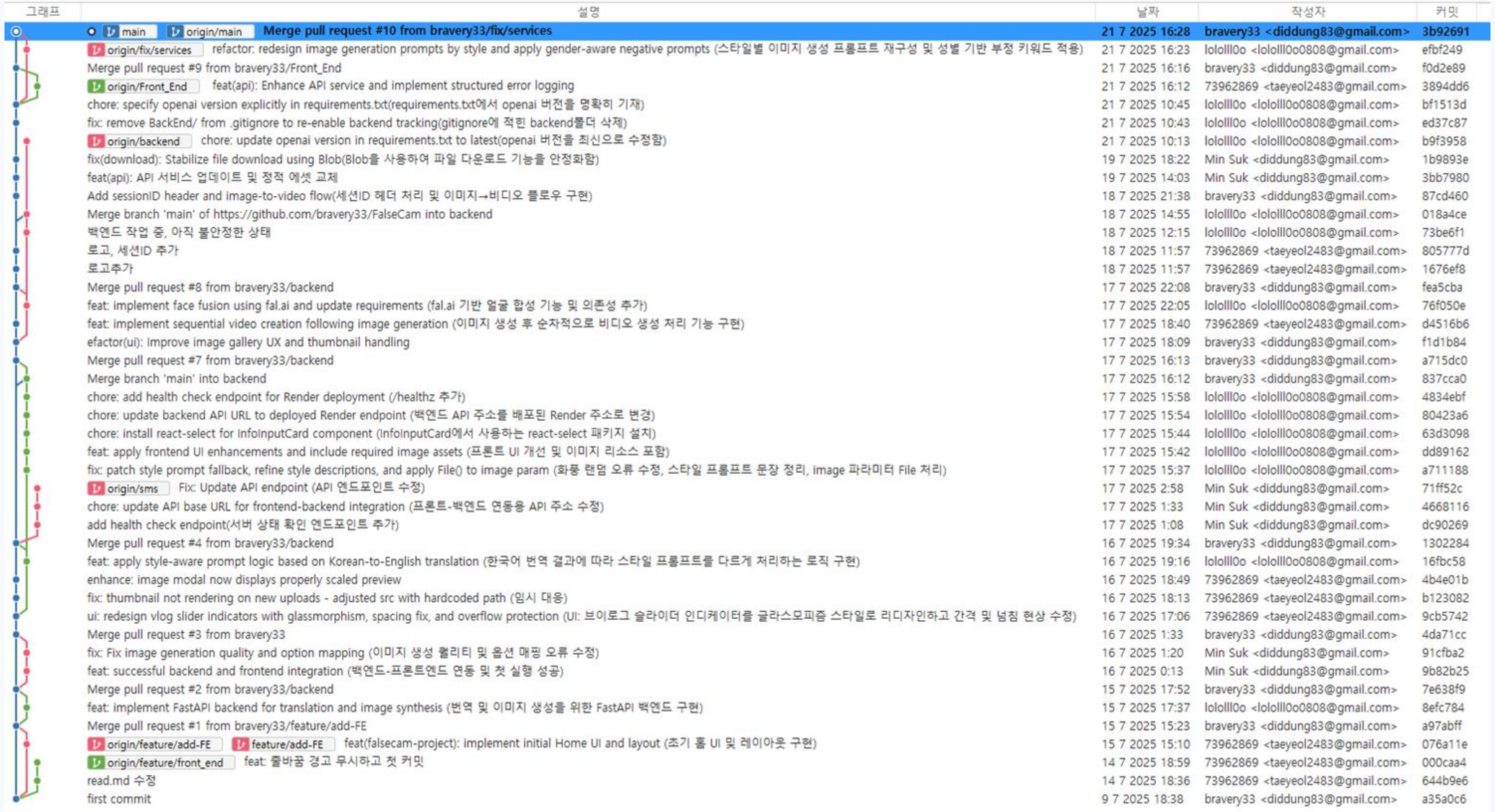
7.2. 테스트 과정 (Image to Video) Test-03

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

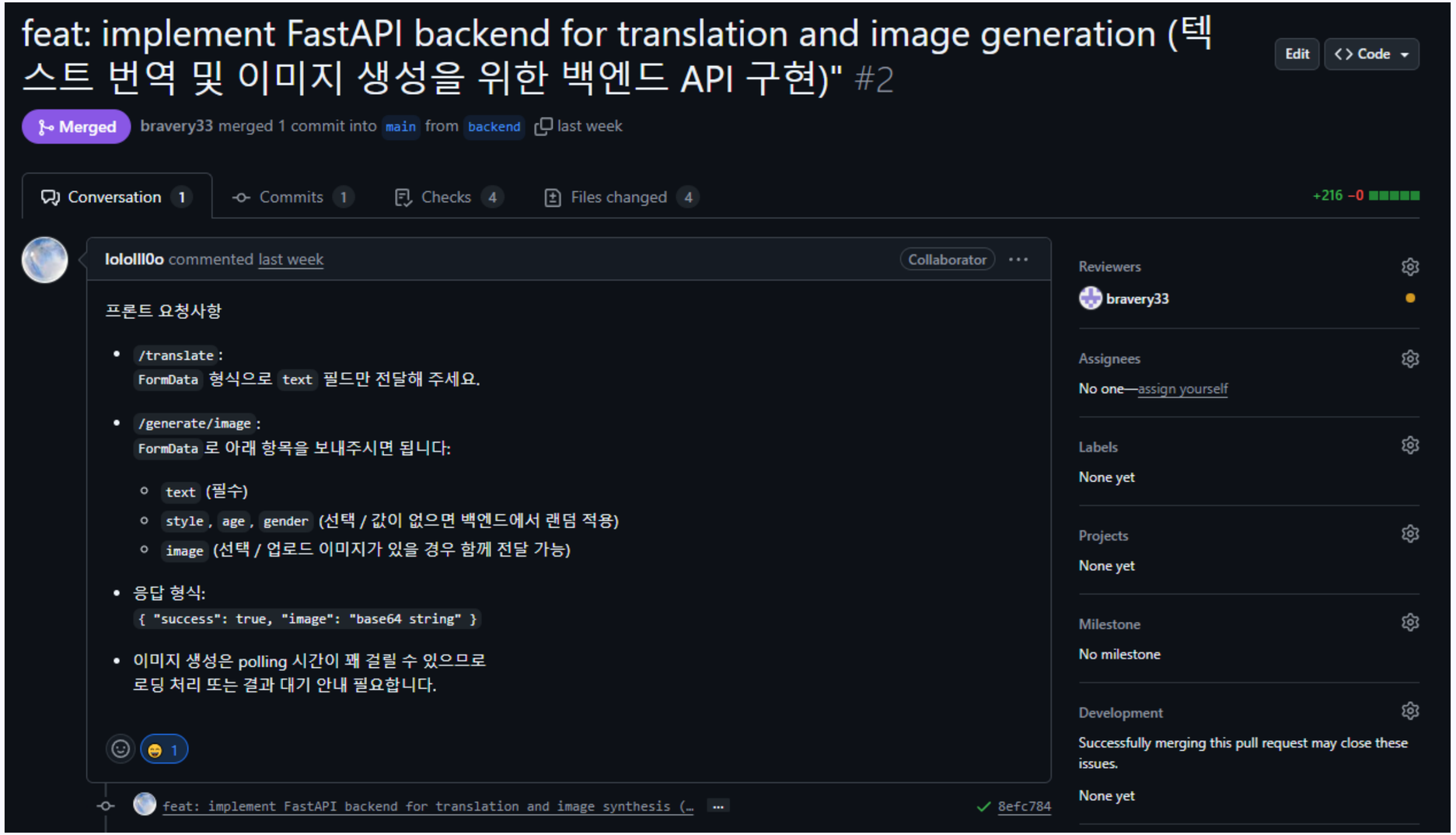
구분	선정 모델	선정 사유
Image-to-Image	FLUX Pro Kontext	품질: 사용자의 이미지를 기반으로 결과물을 생성할 때, 원본의 특징을 일관성 있게 유지하는 능력이 장점으로 고려 되었습니다. 속도/비용: 다른 고품질 모델 대비 합리적인 생성 속도와 비용을 보여주어, 서비스 운영에 가장 적합하다고 판단했습니다. (속도: 7.25초, 비용: 0.04USD)
Image - to-Video	Kling Video 2.1 Standard	품질: 입력 이미지의 특징과 스타일을 일관성 있게 유지하면서도, 가장 자연스러운 움직임을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

8. 형상 관리 및 배포 (Sourcetree)



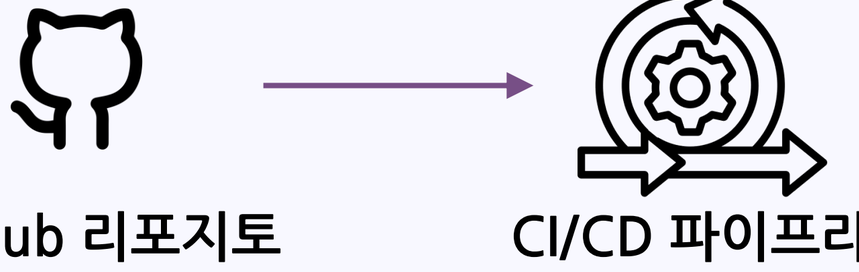
그래프	설명	날짜	작성자	커밋
	main origin/main **Merge pull request #10 from bravery33/fix/services**	**21 7 2025 16:28**	**bravery33 <diddung83@gmail.com>**	**3b92691**
	origin/fix/services refactor: redesign image generation prompts by style and apply gender-aware negative prompts (스타일별 이미지 생성 프롬프트 재구성 및 성별 기반 부정 키워드 적용)	21 7 2025 16:23	lolollll0o <lolollll0o0808@gmail.com>	efbf249
	Merge pull request #9 from bravery33/Front_End	21 7 2025 16:16	bravery33 <diddung83@gmail.com>	f0d2e89
	origin/Front_End feat(api): Enhance API service and implement structured error logging	21 7 2025 16:12	73962869 <taeyeol2483@gmail.com>	3894dd6
	chore: specify openai version explicitly in requirements.txt(requirements.txt에서 openai 버전을 명확히 기재)	21 7 2025 10:45	lolollll0o <lolollll0o0808@gmail.com>	bf1513d
	fix: remove BackEnd/ from .gitignore to re-enable backend tracking(gitignore에 적힌 backend폴더 삭제)	21 7 2025 10:43	lolollll0o <lolollll0o0808@gmail.com>	ed37c87
	origin/backend chore: update openai version in requirements.txt to latest(openai 버전을 최신으로 수정함)	21 7 2025 10:13	lolollll0o <lolollll0o0808@gmail.com>	b9f3958
	fix(download): Stabilize file download using Blob(Blob을 사용하여 파일 다운로드 기능을 안정화함)	19 7 2025 18:22	Min Suk <diddung83@gmail.com>	1b9893e
	feat(api): API 서비스 업데이트 및 정적 에셋 교체	19 7 2025 14:03	Min Suk <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>	87cd460
	Merge branch 'main' of https://github.com/bravery33/FalseCam into backend	18 7 2025 14:55	lolollll0o <lolollll0o0808@gmail.com>	018a4ce
	백엔드 작업 중, 아직 불안정한 상태	18 7 2025 12:15	lolollll0o <lolollll0o0808@gmail.com>	73be6f1
	로고, 세션ID 추가	18 7 2025 11:57	73962869 <taeyeol2483@gmail.com>	805777d
	로고추가	18 7 2025 11:57	73962869 <taeyeol2483@gmail.com>	1676ef8
	Merge pull request #8 from bravery33/backend	17 7 2025 22:08	bravery33 <diddung83@gmail.com>	fea5cba
	feat: implement face fusion using fal.ai and update requirements (fal.ai 기반 얼굴 합성 기능 및 의존성 추가)	17 7 2025 22:05	lolollll0o <lolollll0o0808@gmail.com>	76f050e
	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>	d4516b6
	efactor(ui): Improve image gallery UX and thumbnail handling	17 7 2025 18:09	bravery33 <diddung83@gmail.com>	f1d1b84
	Merge pull request #7 from bravery33/backend	17 7 2025 16:13	bravery33 <diddung83@gmail.com>	a715dc0
	Merge branch 'main' into backend	17 7 2025 16:12	bravery33 <diddung83@gmail.com>	837cca0
	chore: add health check endpoint for Render deployment (/healthz 추가)	17 7 2025 15:58	lolollll0o <lolollll0o0808@gmail.com>	4834ebf
	chore: update backend API URL to deployed Render endpoint (백엔드 API 주소를 배포된 Render 주소로 변경)	17 7 2025 15:54	lolollll0o <lolollll0o0808@gmail.com>	80423a6
	chore: install react-select for InfoInputCard component (InfoInputCard에서 사용하는 react-select 패키지 설치)	17 7 2025 15:44	lolollll0o <lolollll0o0808@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	lolollll0o <lolollll0o0808@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	lolollll0o <lolollll0o0808@gmail.com>	a711188
	origin/sms Fix: Update API endpoint (API 엔드포인트 수정)	17 7 2025 2:58	Min Suk <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>	4668116
	add health check endpoint(서버 상태 확인 엔드포인트 추가)	17 7 2025 1:08	Min Suk <diddung83@gmail.com>	dc90269
	Merge pull request #4 from bravery33/backend	16 7 2025 19:34	bravery33 <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	lolollll0o <lolollll0o0808@gmail.com>	16fbc58
	enhance: image modal now displays properly scaled preview	16 7 2025 18:49	73962869 <taeyeol2483@gmail.com>	4b4e01b
	fix: thumbnail not rendering on new uploads - adjusted src with hardcoded path (임시 대응)	16 7 2025 18:13	73962869 <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>	9cb5742
	Merge pull request #3 from bravery33	16 7 2025 1:33	bravery33 <diddung83@gmail.com>	4da71cc
	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>	91cfba2
	feat: successful backend and frontend integration (백엔드-프론트엔드 연동 및 첫 실행 성공)	16 7 2025 0:13	Min Suk <diddung83@gmail.com>	9b82b25
	Merge pull request #2 from bravery33/backend	15 7 2025 17:52	bravery33 <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	lolollll0o <lolollll0o0808@gmail.com>	8efc784
	Merge pull request #1 from bravery33/feature/add-FE	15 7 2025 15:23	bravery33 <diddung83@gmail.com>	a97abff
	origin/feature/add-FE 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>	076a11e
	origin/feature/front_end feat: 줄바꿈 경고 무시하고 첫 커밋	14 7 2025 18:59	73962869 <taeyeol2483@gmail.com>	000caa4
	read.md 수정	14 7 2025 18:36	73962869 <taeyeol2483@gmail.com>	644b9e6
	first commit	9 7 2025 18:38	bravery33 <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 브랜치에 푸시될 때마다 자동으로 빌드 및 배포가 트리거됩니다.
URL: https://falsecam.pages.dev/

백엔드 배포

Render

FastAPI 서버는 GitHub 리포지토리와 연동되어 Render에 배포됩니다. 마찬가지로 자동 배포가 설정되어 있으며, 민감한 API 키는 Render의 환경 변수 기능을 통해 소스 코드와 분리하여 안전하게 관리됩니다.
URL: https://falsecam.onrender.com/

9. 회의록 (요약)

1차 회의 (25.07.09)
팀명('주석열'), 프로젝트 주제('FalseCam') 및 전체 일정을 확정했습니다. 초기 환경 설정 (GitHub) 및 WBS 초안 작성을 완료했습니다.

2차 회의 (25.07.10)
WBS를 구체화하고 팀원별 태스크를 분배했습니다. 와이어프레임 설계 및 기술 스택을 확정했습니다.

5차 회의 (25.07.15)
백엔드 API 명세를 구체화하고, 프론트엔드 공통 컴포넌트(모달, 버튼) 설계에 대해 논의했습니다.

8차 회의 (25.07.18)
익명 세션 관리 방식(sessionID) 적용을 확정하고, 이미지 생성 후 비디오 자동 생성 플로우 구현에 대해 논의했습니다.

10차 회의 (25.07.23)
MVP 기능 통합 테스트 및 발견된 버그(UI, API 연동)를 리뷰했습니다. 피드백 기반 개선 사항을 논의했습니다.

9. 회의록 (원본)

25.07.14 4차 회의

(프론트엔드/백엔드 개발 착수 및 핵심 모델 선정)

[주요 결정 사항]

- **AI 모델 선정**: Image-to-Image 기능 구현을 위해 **flux-pro/kontext** 모델을 최종 선정

[금일 완료된 업무]

- 프론트엔드
  - React 프로젝트 초기화 및 초기 환경 설정을 완료
  - 공통 UI 컴포넌트(버튼, 입력창 등) 개발을 시작
- **백엔드 & 인프라**
  - Cloudflare 프로젝트를 생성하고 연동을 완료
  - 선정된 모델을 기반으로 Image-to-Image 생성 API 개발을 시작
- **AI 모델**
  - 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: https://github.com/bravery33/FalseCam

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 연동**했으며, 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 일정 관리**를 통해 원활하게 협업했습니다. **CI/CD 파이프라인**을 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 **외부 생성형 AI 모델의 API를 연동**하고 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 개선했습니다. 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.

10.4. 소감

서민석

각기 다른 목표와 배경을 가진 팀원들이 하나의 목표를 위해 힘을 합치는 일은 결코 쉽지 않았습니다. 프로젝트 초반에는 팀의 방향성을 설정하고 기술적인 과제들을 해결해야 한다는 부담이 크게 다가왔습니다. 역할을 분담하고 일정 계획을 세우는 과정에서 다양한 의견이 오갔으며, 각자에게 익숙하지 않은 기술을 익히는 데에도 시간이 많이 필요했습니다. 하지만 팀원들과 지속적으로 소통하고, 서로의 의견을 조율해 나가면서 점차 팀워크가 단단해지는 과정을 겪을 수 있었습니다.

구현 단계에서는 예상치 못한 기술적 난관도 자주 마주쳤습니다. 특히 프론트엔드와 백엔드를 연동하는 과정에서 API 통신이 정상적으로 이루어지지 않거나 CORS 정책 때문에 파일 다운로드가 원활하지 않은 문제들이 반복적으로 발생했습니다. 초기에는 단순히 <a> 태그의 download 속성을 이용해 문제를 해결하려 했으나, 결국 외부 서버로부터 데이터를 받아와 Blob 형태로 가공하고 임시 URL을 생성해 다운로드하는 방식으로 문제를 구조적으로 해결했습니다. 이런 과정을 통해 문제의 근본 원인을 파악하고, 구조적으로 해결책을 마련하는 태도의 중요성을 다시 한 번 실감할 수 있었습니다. 또한, **실제 서비스 환경과 유사한 조건에서 테스트와 배포를 반복**하며, 사용자 관점의 편의성과 품질 관리의 중요성도 크게 배웠습니다. 모델 테스트 결과가 단순히 “잘 된다”에서 끝나지 않고, 실제 서비스 화면에서의 사용자 경험이나 **다양한 환경에서의 일관된 품질까지 꼼꼼히 검증**하는 습관을 기를 수 있었습니다.

이미지 투 이미지, 이미지 투 비디오 등 다양한 **AI 생성 모델을 직접 실험**하면서 각 모델의 특징과 한계를 실제로 체험할 수 있었습니다. 이미지 투 비디오 모델의 경우, 동일한 이미지를 사용하더라도 모델 종류에 따라 결과 영상의 완성도와 자연스러움이 크게 달라졌고, 여러 차례 반복 실험을 통해 실제 서비스 목적에 가장 적합한 모델을 선정할 수 있었습니다. 이미지 투 이미지 모델 역시 프론트엔드에서 사용자가 이미지를 업로드하고, AI가 새로운 이미지를 생성해 반환하는 전체 흐름을 직접 설계·구현했습니다. 아울러, 테스트 과정의 주요 장면을 녹화해 유튜브에 업로드함으로써, **실험 결과와 구현 과정을 팀원들과 투명하게 공유**할 수 있었습니다. 이러한 과정을 통해 테스트 결과에 대한 신뢰도를 높이고, 프로젝트의 전반적인 투명성과 협업 효율도 함께 향상시킬 수 있었습니다.

이처럼 다양한 도전과 협업의 과정을 거치며 단순한 기술적 성장뿐 아니라 소통, 역할 분담, 문제 해결 과정에서의 리더십과 협업 능력도 함께 키울 수 있었습니다. 프로젝트 후반에는 각자 맡은 역할에 대한 책임감이 더욱 커졌고, 팀원 각자의 역량이 시너지를 내는 경험도 할 수 있었습니다. 이번 프로젝트는 단순히 결과물을 만드는 데 그치지 않고, **협업의 가치와 함께 성장하는 즐거움을 깊이 체감**할 수 있었던 소중한 시간이었습니다. 이 경험을 바탕으로 앞으로도 새로운 도전에 두려움 없이 나아갈 수 있을 것이라 생각합니다.

10.4. 소감

최태열

이번 프로젝트를 시작할 때 저는 사실 코딩을 잘 모르는 상태였습니다. 화면에 버튼 하나 만들어도 어디서부터 건드려야 할지 막막했고, 파일 이름이나 폴더 구조조차 익숙하지 않았습니다. 하지만 '내 손으로 무언가를 만든다'는 점에서, 그동안 영상매체를 만들고 편집하면서 느꼈던 **직접 창작하는 즐거움과 닮아 있다**고 생각했고, 그 기대와 마인드를 갖고 이 프로젝트에 참여하게 됐습니다.

가장 힘들었던 건 문제가 생겼을 때 그 원인을 몰라서 아무 것도 못 하는 순간들이었습니다. 예를 들어, 이미지를 클릭하면 큰 화면으로 보여주는 기능을 만들었는데, 분명 코드는 다 맞게 쓴 것 같고 클릭도 되는데 정작 화면에는 아무것도 뜨지 않았습니다. 몇 시간을 붙잡고 씨름하다가 결국 코드가 보이긴 하는데 뒤에 가려져 있었다 는 걸 알게 됐고, 그때 처음으로 코드는 맞아도 사용자 눈에는 안 보일 수 있다는 걸 체감했습니다. 사소한 설정 하나 때문에 전체 기능이 안 되는 걸 보면서, 개발이라는 게 참 **작은 걸로도 전부가 무너질 수 있다**는 걸 처음 느꼈습니다. 또 데이터를 서버로 보내는 부분에서는 이걸 어떻게 묶어서 보내야 하는지도 몰라서 계속 오류가 났고, 결국 하나하나 찾아보고 다시 넣어보면서 겨우 해결할 수 있었습니다. 지금 와서 보면 별거 아닐 수 있지만, 그 당시에는 그 한 줄을 이해 못 해서 2시간을 멈춰 있었던 날도 많았습니다.

UI 디자인을 꾸미는 것도 마찬가지였습니다. 폰트 크기나 여백, 버튼 모양을 바꾸려다 오히려 화면이 다 망가지기도 했고, 어느 순간부터는 디자인보단 **사용자가 쓰기 편한 화면이 뭘까**를 고민하게 됐습니다. 예쁘게 보이는 것보다, **사용자가 흐름을 끊기지 않고 쓸 수 있도록** 정리해나가는 과정이 오히려 더 중요하다는 걸 깨달았습니다.

무엇보다 이번 프로젝트에서 제게 가장 컸던 건 **‘모른다고 겁먹지 않고 계속 시도하는 자세’** 였습니다. 저는 이 마음가짐을 사실 군 복무 당시부터 익혀왔습니다. 모르는 일도 먼저 손 들고 나서고, **실수해도 도망가지 않는 습관**이 그때부터 생겼는데, 이번 프로젝트 내내 그 태도가 큰 힘이 됐습니다. 몰라도 붙잡고, 이해 안 되면 다시 물어보고, 계속 시도하면서 어느 순간 내가 만든 화면에 결과가 나오는 걸 보게 됐고, 그게 이 프로젝트에서 가장 뿌듯한 순간이었습니다.

앞으로 개발자가 될지는 모르겠습니다. 하지만 이번 경험을 통해, 어떤 일이든 **‘모른다고 포기하지 않고 부딪히는 자세’**만은 꼭 지키고 싶다는 생각이 들었습니다. 누군가에겐 쉬운 코드 한 줄이 저에겐 큰 벽이었고, 그 벽을 넘은 제가 지금 여기까지 왔다는 것만으로도 이 프로젝트는 제게 가장 소중한 배움의 시간이었습니다.

10.4. 소감

김주현

FalseCam 프로젝트는 "아직 사람들이 경험해 보지 못한 새로운 서비스가 무엇일까?"라는 질문에서 시작되었습니다. 저는 AI 이미지 생성 기술 자체보다, "AI가 만든 나의 모습을 제3자의 시선으로 바라본다"는 아이디어의 참신함에 주목했습니다. 이 독특한 콘셉트를 실제 서비스로 구현하기 위해, 프로젝트의 전반적인 기획부터 FastAPI 기반의 백엔드 개발까지 주도하는 역할을 담당했습니다.

주로 프론트엔드 개발 경험이 많았던 저에게, 서비스의 전체 아키텍처를 구상하고 여러 외부 AI와 연동되는 데이터 흐름을 설계하는 것은 의미 있는 도전이었습니다. 이 과정을 통해 사용자 요청 처리, 데이터 가공 및 반환에 이르는 백엔드의 핵심적인 역할을 깊이 이해할 수 있었습니다.

프로젝트의 가장 중요한 과제는 "제3자의 시선"이라는 독특한 콘셉트를 기술로 설득력 있게 구현하는 것이었습니다. 하지만 개발 초반, 이 목표를 위협하는 큰 문제에 직면했습니다. 사용자 얼굴을 인식시킨 결과물이 대부분 증명사진처럼 정면을 응시하는, 단조롭고 인위적인 구도로만 생성되는 현상이었습니다. 이는 저희 서비스의 핵심 차별점을 무력화시키는 치명적인 문제였습니다.

이 문제를 해결하고자, 저는 단순히 코드를 더하는 물리적인 방식이 아닌, 발상의 전환을 시도했습니다. 바로 프롬프트 설계를 정교하게 조율하는 것이었습니다. **"Paparazzi shot"**이나 "Candid shot", "shot on 100mm lens" 같은 전문적인 촬영 용어를 핵심 키워드로 추가하여 AI를 원하는 방향으로 유도했습니다. 그 결과, 복잡한 로직 변경 없이도 다채롭고 안정적인 구도를 만들어내는 데 성공했습니다. 창의적인 아이디어를 통해 기술적 한계를 돌파했을 때 큰 보람을 느꼈고, 문제 해결에 대한 시야를 넓히며 개발자로서 한 단계 성장할 수 있었습니다.

물론 이러한 성과는 저 혼자만의 힘으로 이룬 것이 아닙니다. 팀원들과 끊임없이 소통하며 사용자의 경험에 대해 논의했던 시간 덕분에 프로젝트의 방향을 더욱 선명하게 다듬을 수 있었습니다. 프론트엔드에서의 고민과 백엔드에서의 기술적 구현이 시너지를 이룰 때 비로소 하나의 완성도 있는 서비스가 만들어진다는 사실을 깨달으며, 협업의 가치를 다시 한번 깊이 느낄 수 있었습니다.

FalseCam은 제게 기술적 성장과 더불어, 아이디어를 현실화하고 기술적 문제를 사용자의 관점에서 분석하여 창의적으로 해결하는 경험을 안겨준 뜻깊은 프로젝트입니다.

시연

경청해주셔서
감사합니다.