

# Style License

당신의 화풍을  
거래하세요

직무 풀스택 개발

기간 2025.10 - 2025.12(2개월)

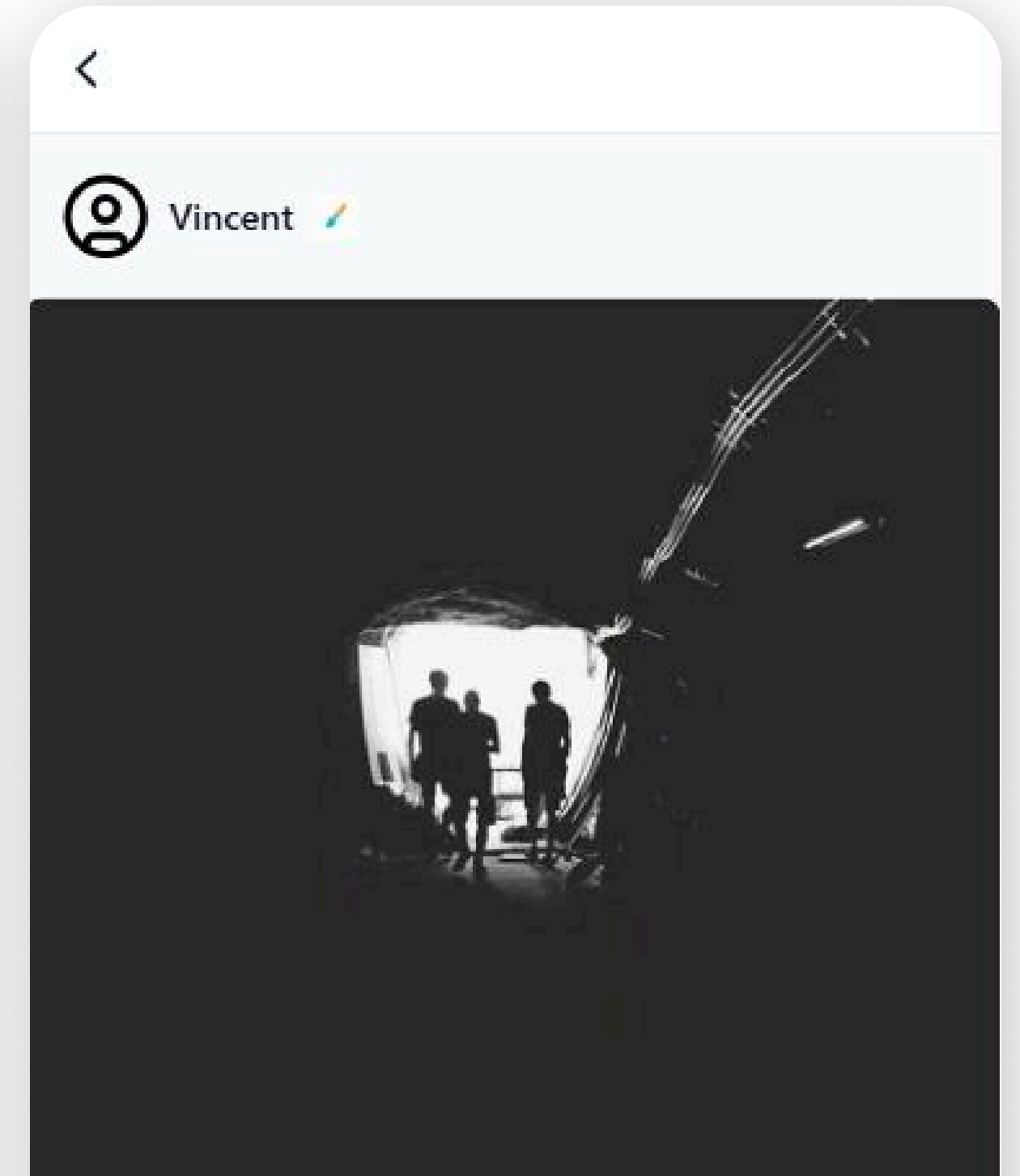
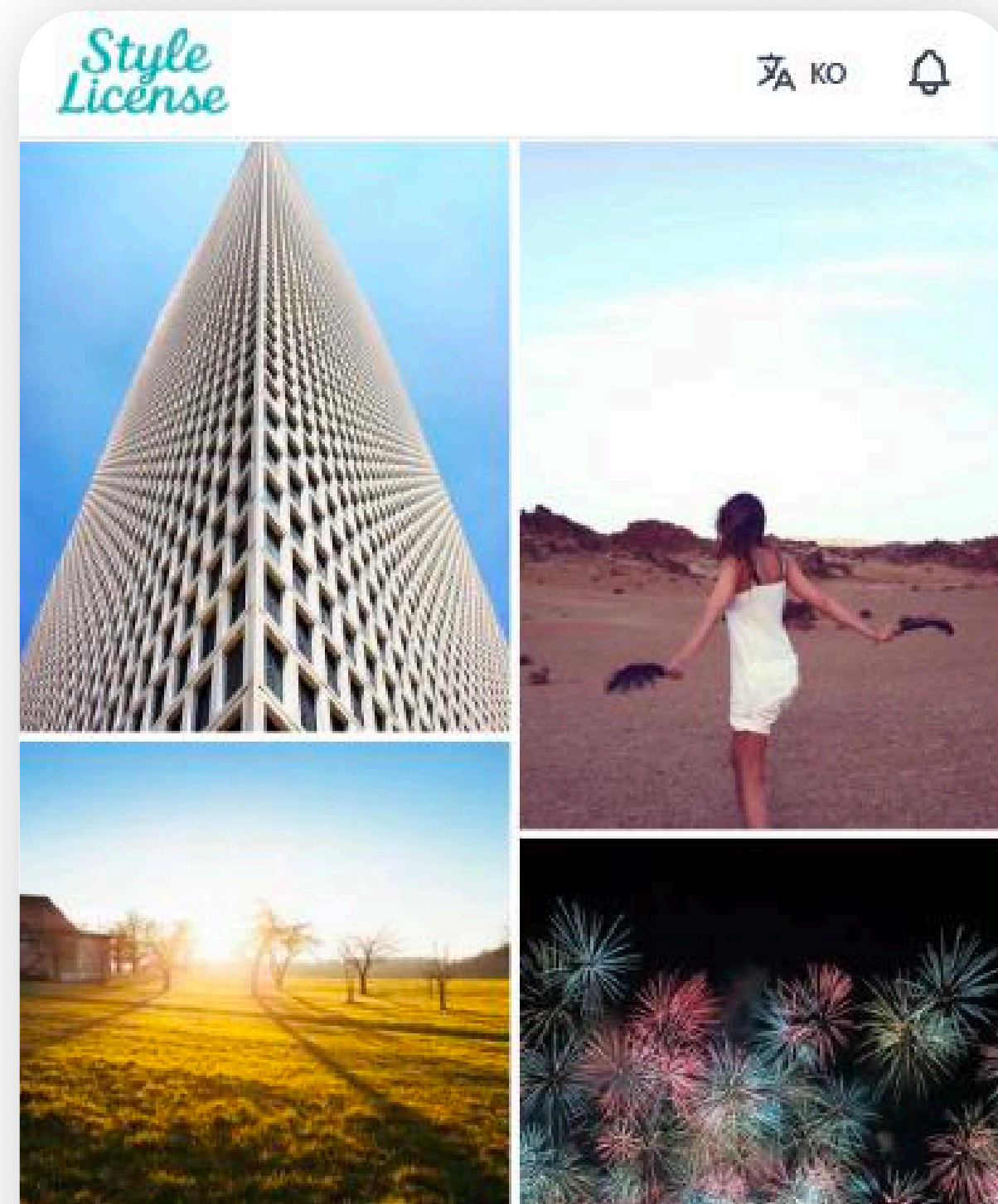
1 LoRA Fine-tuning

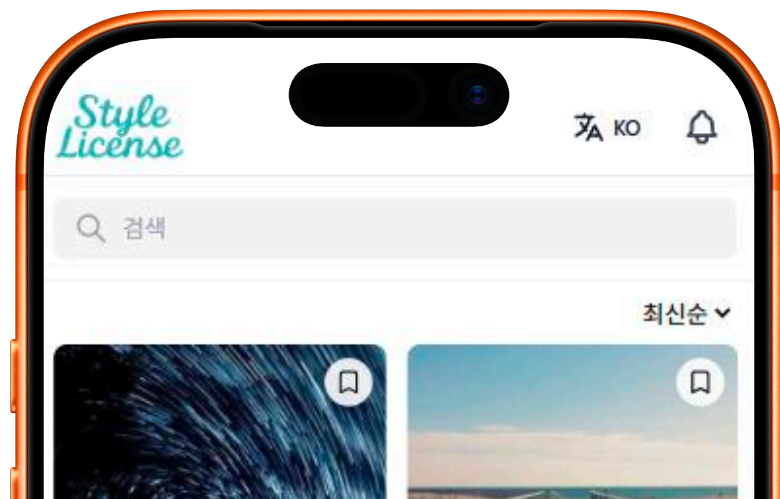
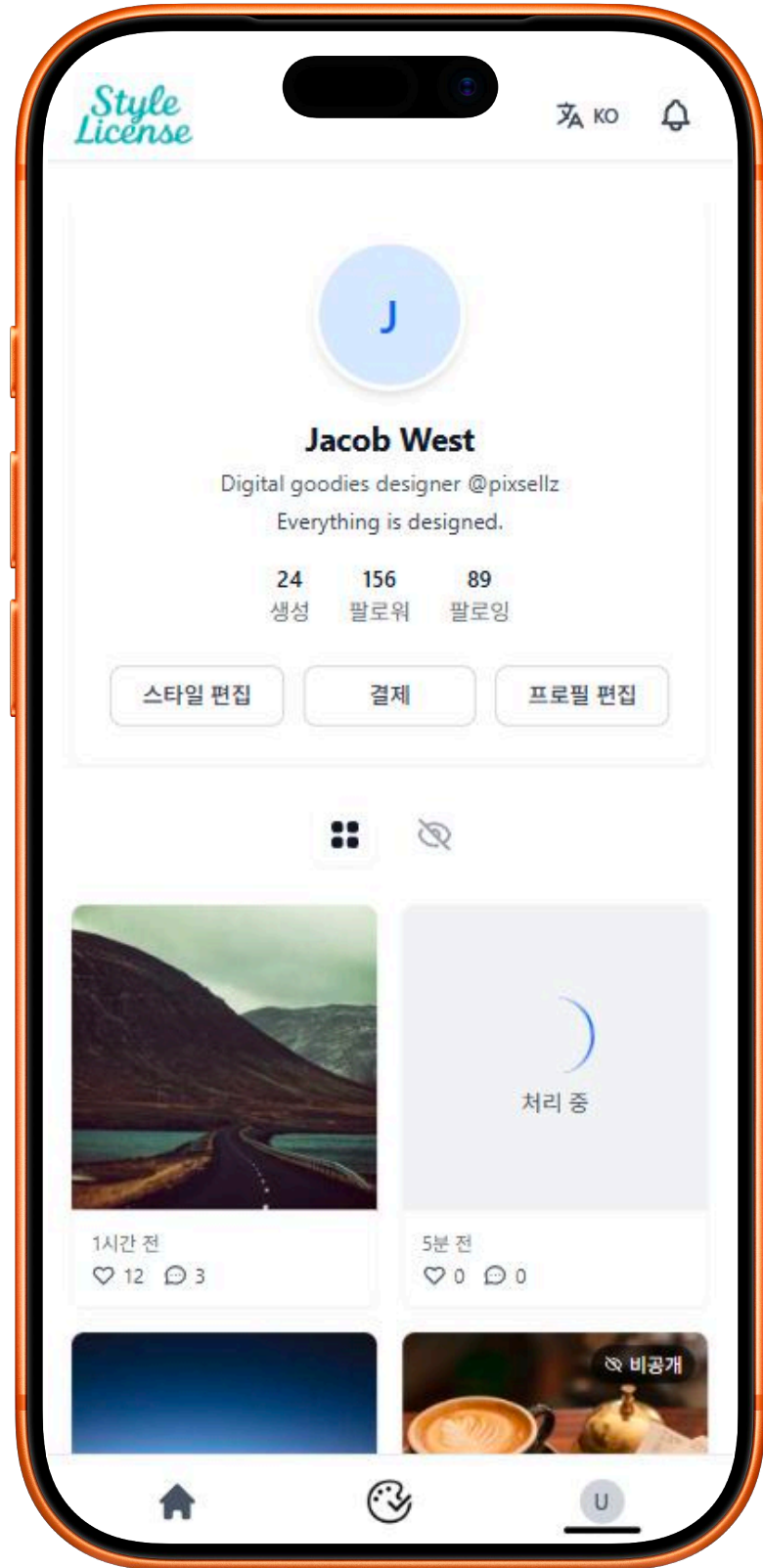
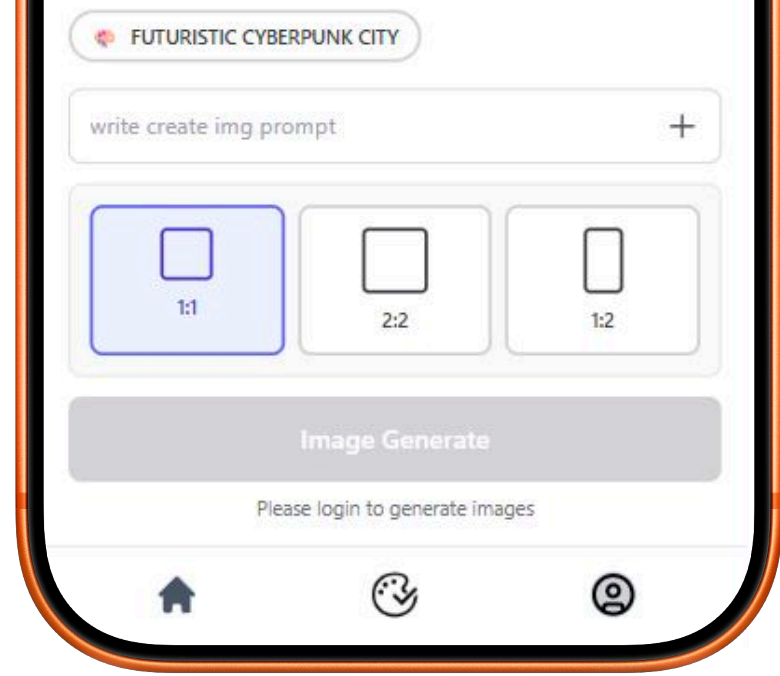
2 SDD

3 AI - Assisted Development

Style License는 AI 시대, 자신의 화풍으로 이미지를 생성할 수 있는 권리를 판매하는 플랫폼입니다.

AI 창작 생태계, 작가의 화풍에 권리를 부여하기 제작된 프로젝트입니다.





### Period

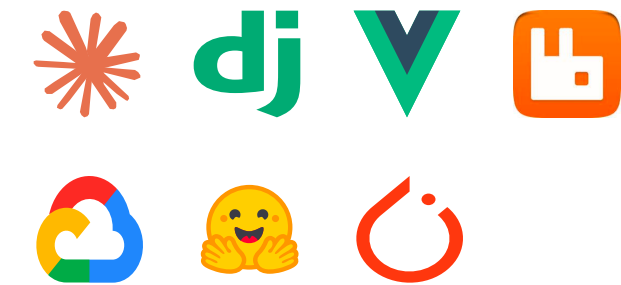
2025.10 ~ 2025. 12

### Link

<https://github.com/Strangekim/StyleLicense>

<https://noted-sled-478700-r3.web.app/profile>

### Skills



### Member

김연호, 풀스택 개발자

방승재, 풀스택 개발자

### 핵심 개발

- ▶ SDD 원천 TechSpec 작성
- ▶ 전체 아키텍처 설계
- ▶ LoRA 학습 구조 검토 및 실험
- ▶ 화면 UI 시안 제작
- ▶ GPU 서버 테스트

# Background

Style License 프로젝트는 AI 이미지 생성 기술의 급속한 발전과 이로 인해 발생하는 창작 생태계의 저작권 문제를 해결하기 위해 시작하였습니다.

## Problem

### 1 원작자의 저작권 침해 문제

AI 학습 과정에서 원작자의 이미지가 무단으로 사용되어 저작권 침해 논란이 끊이지 않고 발생하고 있습니다.

### 2 화풍 사용에 대한 법적 권리

특정 작가의 고유한 화풍 (Style)에 대한 법적 보호 기준이 불명확하여, AI가 이를 모방하거나 학습에 사용하는 것에 대한 명확한 권리관계가 부재하였습니다.

### 3 2차 창작물의 상업적 이용 제약

AI로 생성된 이미지나 이를 활용한 2차 창작물의 상업적 이용 범위와 조건이 모호하여, 창작 활동과 수익 창출에 제약이 있었습니다.

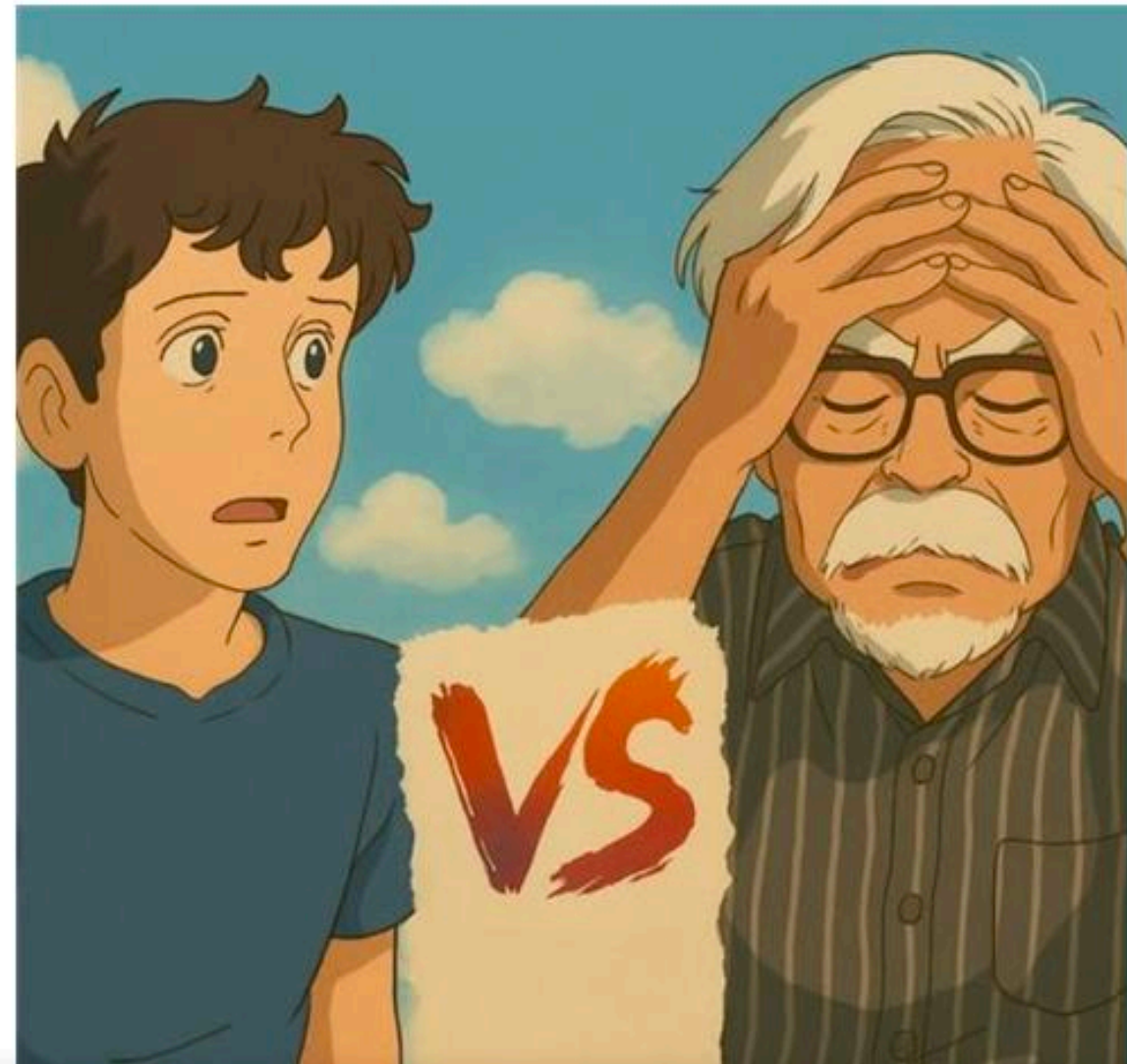
### 4 작가의 수익 창출 기회 상실

작가들의 화풍이 무단으로 AI 모델에 학습되어 대규모로 복제되면서, 작가 고유의 창작물 판매 및 라이선스 수익과 같은 수익 창출 기회가 상실되는 위기에 처했습니다.

## [샷!] "지브리 프사, 예술에 대한 모욕처럼 느껴져"

송고 2025-04-17 05:50

챗GPT '지브리 프사' 열풍, 창작자들에 '실존적 고민' 안겨  
"오랜 시간과 노력 기울였는데 AI가 몇초만에 따라 해버려"  
'나만의 콘텐츠' 고민..."불안해하기보다 AI 잘 활용해야"



▶ 최근 화두가 된 지브리 풍 프로필 사진 기사

## [AI 시대의 저작권] ① 버튼 하나로 만든 그림... 저작권 등록해도 될까

송고 2025-11-01 08:00

'인간의 창작적 기여' 정도가 관건...단순 AI 산출물 저작권 등록은 법적 불이익

[\* 편집자 주 = 기술 발전이 만들어낸 인공지능(AI) 생성물이 창작과 산업생태계 전반을 혁신적으로 변화시키고 있습니다. AI의 개입은 저작권자, 실연자, 창작자의 권리 보호와 산업 성장의 공정이용 범위라는 해묵은 숙제를 다시 던지고 있습니다. 연합뉴스는 AI 기술과 저작권의 충돌 지점을 집중적으로 분석하고, 나아가 해외 입법 동향과 AI 시대 K-콘텐츠의 지속 가능한 미래 비전도 모색하는 기획 기사를 매주 1건씩 4건을 송고합니다.]

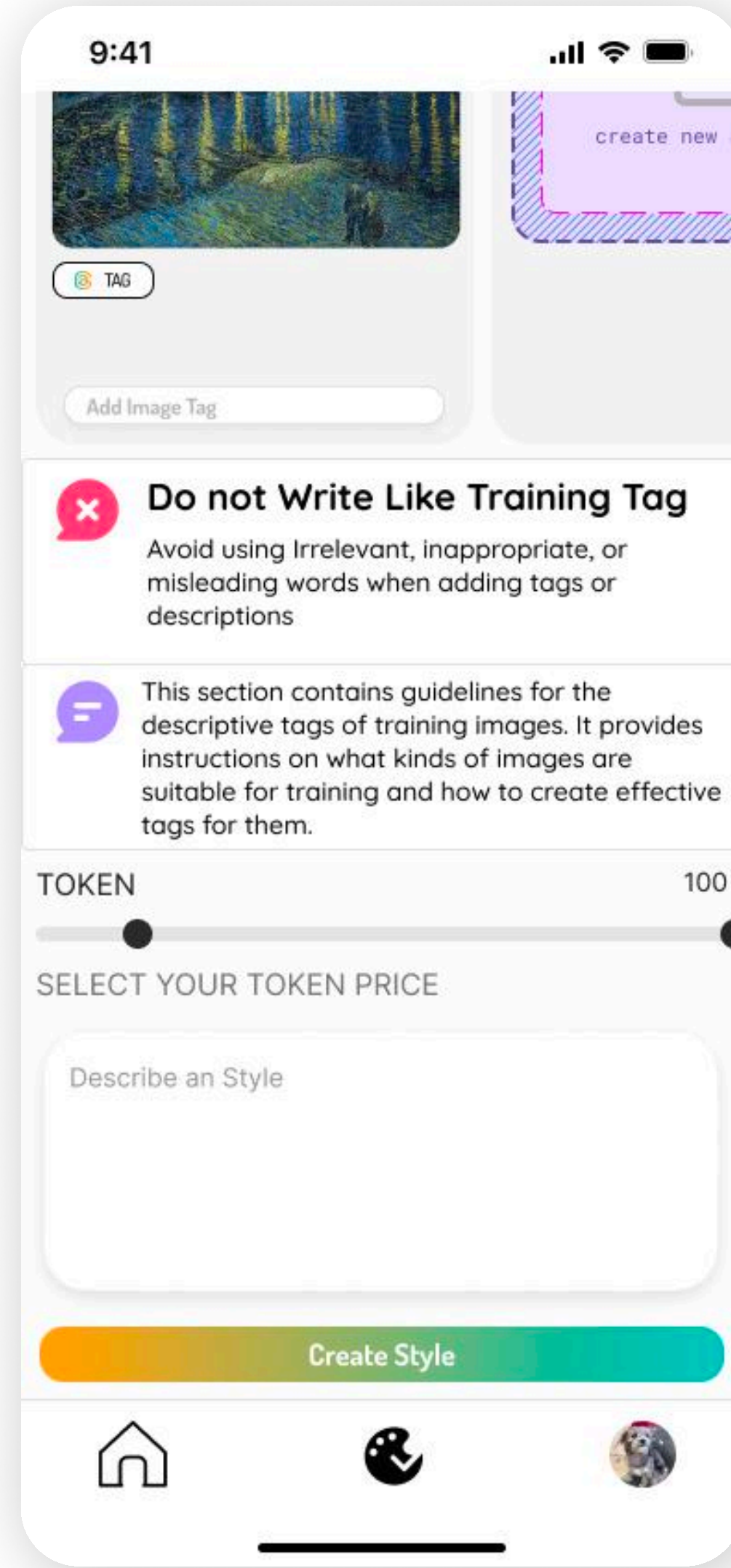


▶ AI 저작권 문제에 대한 기사

# Solution

## ▶ 작가가 본인의 화풍을 학습한 AI 모델을 직접 제작

작가가 직접 자신의 화풍을 AI 모델에 학습 데이터로 등록하고, 그 모델이 생성한 이미지를 판매할 수 있습니다. 합법적 2차 창작물을 상업적으로 이용할 수 있는 기반을 마련하며, 작가의 화풍에 대한 수익을 작가가 직접 책임함으로써 라이선스 기반 수익에 법적 근거를 제시합니다.



▶ 작가 화풍 학습 페이지

```
import os
import tempfile
from training.trainer import LoRATrainer

logger.info(f"Starting real LoRA training for style_id={style_id}")
logger.info(f" Images: {len(images)}")
logger.info(f" Epochs: {num_epochs}")
logger.info(f" Learning rate: {learning_rate}")

try:
    # Step 1: Download images from GCS
    logger.info("Downloading images from GCS...")
    with tempfile.TemporaryDirectory() as temp_dir:
        image_dir = os.path.join(temp_dir, "images")
        os.makedirs(image_dir, exist_ok=True)

        local_image_paths = self.gcs_service.download_images(images, image_dir)

        if not local_image_paths:
            logger.error("Failed to download any images")
            return (None, 0.0)

        logger.info(f"Downloaded {len(local_image_paths)} images")

    # Step 2: Setup training output directory
    output_dir = os.path.join(temp_dir, "output")
    os.makedirs(output_dir, exist_ok=True)

    # Step 3: Initialize trainer
    trainer = LoRATrainer(
        style_id=style_id,
        output_dir=output_dir,
        num_epochs=num_epochs,
        learning_rate=learning_rate,
    )

    # Step 4: Define progress callback
    def progress_callback(current_epoch, total_epochs, loss):
        progress_percent = int((current_epoch / total_epochs) * 100)
        remaining_epochs = total_epochs - current_epoch
        estimated_seconds = remaining_epochs * 36 # Assume 36s/epoch


```

▶ GPU 학습 서버 코드

```
def check_cuda():
    """Check CUDA availability and log GPU information"""
    logger.info("Checking CUDA availability...")

    if torch.cuda.is_available():
        gpu_count = torch.cuda.device_count()
        logger.info(f"CUDA is available! Found {gpu_count} GPU(s)")

        for i in range(gpu_count):
            gpu_name = torch.cuda.get_device_name(i)
            gpu_memory = torch.cuda.get_device_properties(i).total_memory / (1024**3)
            logger.info(f"GPU {i}: {gpu_name} ({gpu_memory:.2f} GB)")

        # Test simple tensor operation on GPU
        try:
            x = torch.tensor([1.0, 2.0, 3.0]).cuda()
            y = x * 2
            logger.info(f"GPU tensor test successful: {y.cpu().tolist()}")
        except Exception as e:
            logger.error(f"GPU tensor test failed: {e}")
    else:
        logger.warning("CUDA is not available. Training will run on CPU (not recommended for production)")

    return torch.cuda.is_available()


```

▶ CUDA 가용성 확인

# Solution

## ▶ SDD

이번 프로젝트는, 개발 과정에서 AI를 적극 활용하여 개발하기 위하여 **SDD (Spec-Driven-Development)** 접근 방식을 활용하였습니다.

SDD는 코드 작성 전 명세서를 먼저 작성하는 “**문서 우선**” 접근 방식으로, 명세서가 개발자와 AI 모두에게 **단일 진실 공급원 역할**을 수행합니다.

여기서 **명세서 (spec)**란, 자연어로 작성된 구조화되고 행동 지향적인 산출물입니다. 이는 **AI 코딩 에이전트에게 가이드** 역할을 수행합니다.

## ▶ TECHSPEC.md

**TECHSPEC.md** 파일은 모든 기술적 세부사항을 총망라한 **개발 총집합 문서**입니다. 2천 자의 자연어로 작성되어 있으며 18개의 섹션으로 구분되어 있습니다.

TECHSPEC.md는 **모든 기술적 의사결정의 근거**가 되고, 새로운 기능을 구현하기 전 이 문서를 참조하여 아키텍처, API 설계, 데이터베이스 스키마를 이해하고 **일관된 방식으로 코드를 작성**할 수 있습니다.

특히 각 세션 끝에는 상세 문서 링크가 있어, 더 깊은 구현 세부 사항을 확인할 수 있습니다.

TECHSPEC.md 파일은, claude code가 작성하고, codex cli, gemini cli로 교차 검증 하였습니다. 최종 파일은 개발자들이 직접 점검하였습니다.

```

PROJECT ROOT
├── README.md           → 프로젝트 전체 소개 (외부 공개용)
├── TECHSPEC.md        → 전체 기술 명세 / 시스템 개요 (최상위)
├── PLAN.md            → 프로젝트 일정/개발 관리
├── claude.md          → AI 작업 지침/프롬프트 가이드
├── docs/              → 전문 문서화 영역
│   ├── database/     → 데이터베이스 문서
│   │   ├── README.md → DB 개요 + 전체 ERD + 설계 철학
│   │   └── guides/
│   │       ├── queries.md → 주요 쿼리 패턴/예제
│   │       └── operations.md → 운영/마이그레이션 규칙
│   │           └── TABLES.md → 테이블 정의 총정리
│   ├── API.md        → API 상세 스펙
│   └── PATTERNS.md    → 서비스 내 공통 코드/설계 패턴 명세
├── apps/              → 실제 서비스 실행 구조 (MSA 지향)
│   ├── backend/
│   │   ├── README.md → 백엔드 아키텍처 설명
│   │   ├── PLAN.md   → 백엔드 진행 계획
│   │   └── CODE_GUIDE.md → 코드 컨벤션 및 구조 가이드
│   ├── frontend/
│   │   ├── README.md
│   │   ├── PLAN.md
│   │   └── CODE_GUIDE.md
│   ├── training-server/ → AI 모델 학습 서비스
│   │   ├── README.md
│   │   ├── PLAN.md
│   │   └── CODE_GUIDE.md
│   └── inference-server/ → 모델 추론 실행 서비스
│       ├── README.md
│       ├── PLAN.md
│       └── CODE_GUIDE.md

```

▶ SDD 진행을 위한 문서 구조도

```

# Style License 프로젝트 기술 명세서 (TECHSPEC.md)

**Repository:** 작성 예정
**Version:** 1.0

---

## 목차
1. [프로젝트 개요](#1-프로젝트-개요)
2. [목표 및 비목표](#2-목표-및-비목표)
3. [배경 및 문제 정의](#3-배경-및-문제-정의)
4. [시스템 아키텍처](#4-시스템-아키텍처)
5. [데이터 모델 및 스키마](#5-데이터-모델-및-스키마)
6. [API 설계](#6-api-설계)
7. [프론트엔드 아키텍처](#7-프론트엔드-아키텍처)
8. [비즈니스 로직 및 핵심 기능](#8-비즈니스-로직-및-핵심-기능)
9. [폴더 구조, 공통 패턴](#9-폴더-구조-공통-패턴)
10. [보안 설계](#10-보안-설계)
11. [에러 처리 및 로깅](#11-에러-처리-및-로깅)
12. [성능 및 확장성](#12-성능-및-확장성)
13. [테스트 전략](#13-테스트-전략)
14. [배포 및 CI/CD](#14-배포-및-cicd)
15. [문서화 요구사항](#15-문서화-요구사항)
16. [마이그레이션 계획](#16-마이그레이션-계획)
17. [위험 요소 및 완화 전략](#17-위험-요소-및-완화-전략)
18. [참고 자료](#18-참고-자료)

## 1. 프로젝트 개요

### 1.1 프로젝트명
**Style License** - AI 기반 화풍 라이선싱 및 2차 창작 플랫폼

### 1.2 목적
디자인 작가가 자신의 스타일과 화풍을 거래할 수 있는 웹 플랫폼을 제공합니다. 화풍 거래를 라이선스 형식으로 인정하고, 사용자는 저작권이 보장된 상태로 작가의 화풍을 모방한 2차 창작물을 소유, 공유, 상업적으로 이용할

### 1.3 핵심 가치 제안
- 작가의 화풍에 대한 명확한 저작권 보호
- 합법적이고 투명한 AI 화풍 라이선싱
- 라이선스 기반의 2차 창작물 생성 및 상업적 이용 가능

### 1.4 주요 이해관계자
- **디자인 작가**: 화풍 라이선스 제공자, 수익 창출
- **일반 사용자**: 라이선스 구매자, 2차 창작물 생성자
- **플랫폼 운영자**: 라이선스 중개, 저작권 관리

```

▶ TECHSPEC.md 문서 중 일부

# Solution

## ▶ PLAN.md

AI 에이전트가 개발을 순차적, 계획적으로 진행하기 위한 **전체 개발 일정표 및 진행 상황 추적 문서**입니다. 프로젝트의 마스터 스케줄러 역할을 하며, **모든 작업의 우선순위, 의존성, 완료 여부를 한눈에 파악**할 수 있습니다.

미리 기술해둔 AI 에이전트의 명령어를 입력하면, 이 PLAN.md 파일에서 다음 작업을 확인하고, 진행합니다.

```

##### PT-M2-StyleAPI: Style Model API
- **Status**: DONE
- **Type**: PARALLEL
- **Can Run With**: [PT-M2-TokenAPI, PT-M2-TagAPI]
- **Dependencies**: [CP-M2-1, CP-M2-2]
- **Owner**: Backend
- **Reference**: [apps/backend/PLAN.md#m2-style-model-api](apps/backend/PLAN.md#m2-style-model-api)
- **Summary**:
  - [x] POST /api/models/ (image upload, tag assignment, RabbitMQ integration) (Commit: 69951bd)
  - [x] GET /api/models/ (pagination, filter by tags/artist/status, sort by popular/recent) (Commit: 69951bd)
  - [x] GET /api/models/:id (detail with artist info, artworks, tags) (Commit: 69951bd)
  - [x] DELETE /api/models/:id (owner-only permission, soft delete) (Commit: 69951bd)
  - [x] Validation: 10-100 images, JPG/PNG only, max 10MB each (Commit: 69951bd)
  - [x] 13 tests passing (Commit: 69951bd)

##### PT-M2-TokenAPI: Token System API
- **Status**: DONE
- **Type**: PARALLEL
- **Can Run With**: [PT-M2-StyleAPI, PT-M2-TagAPI]
- **Dependencies**: [CP-M2-1, CP-M2-3]
- **Owner**: Backend
- **Reference**: [apps/backend/PLAN.md#m2-token-api](apps/backend/PLAN.md#m2-token-api)
- **Summary**:
  - [x] GET /api/tokens/balance/ (Commit: 09341ab)
  - [x] POST /api/tokens/purchase/ (mock payment, validation) (Commit: 09341ab)
  - [x] GET /api/tokens/transactions/ (filter by type, pagination, direction) (Commit: 09341ab)
  - [x] 12 tests passing (Commit: 09341ab)

```

### ▶ 전체 PLAN.MD 문서 일부

```

- [x] Connection pooling (Commit: 6c4bfec)
- [x] Connection reuse with context manager
- [x] Implement connection retry logic (max 3 attempts)
- [x] Close connections gracefully on application shutdown

- [x] Testing (Commit: 6c4bfec)
- [x] Test message delivery to RabbitMQ (mocked, 8/8 passing)
- [x] Verify message format
- [x] Test connection retry logic
- [x] Test no connection leaks
- [x] Test singleton pattern
- [x] Test webhook URL generation

**Implementation Reference**: [CODE_GUIDE.md#rabbitmq-integration](CODE_GUIDE.md#rabbitmq-integration)

**Exit Criteria**:
- [x] Messages appear in RabbitMQ queue after API call (implementation complete)
- [x] Message format matches schema
- [x] No connection leaks (context manager + singleton)

```

### ▶ 각 프로젝트별 PLAN.MD 문서 일부

## ▶ CLAUDE.md

이 파일은 AI 에이전트가 프로젝트를 개발할 때 따라야 할 **워크플로우 메뉴얼**입니다.

AI 에이전트를 위한 “작업 지침서”로, 인간 개발자의 **온 보딩 가이드와 유사한 역할**을 합니다.

프로젝트가 복잡해질수록 AI가 맥락을 잃고 명세를 벗어나거나 일관성 없는 코드를 작성할 위험이 있습니다. 이 문서는 AI 에이전트가 **매번 정확히 같은 방식으로 작업하도록 표준화된 절차**를 제공합니다.

```

## Initial Setup

**첫 작업 시작 전 필수 읽기:**

1. **Read [TECHSPEC_SUMMARY.md](TECHSPEC_SUMMARY.md)** - 전체 시스템 핵심 요약 (필수, 45분 내 파악 가능)
  - Section 1-2: 프로젝트 정체성, 아키텍처, 기술 스택
  - **Section 4**: **프론트엔드 페이지 9개 목록** (페이지 누락 방지!)
  - Section 3.7: API 그룹 7개 개요 (API 누락 방지!)
  - Section 3.8: 에러 코드 체계 (일관된 에러 처리)
  - Section 3.9-3.11: 보안, 성능, 테스트 전략
2. **Read [PLAN.md](PLAN.md)** - 현재 진행 중인 Milestone 확인
3. **필요시 Read [TECHSPEC.md](TECHSPEC.md)** - 상세 명세 (선택적, SUMMARY의 앵커 링크 활용)
  - Section 6: API 전체 스펙 (특정 엔드포인트 구현 시)
  - Section 7: 프론트엔드 아키텍처 전체 (프론트엔드 구현 시)
  - Section 10-13: 보안/에러/성능/테스트 상세 (필요 시)

**항상 참조:**
- [docs/database/README.md](docs/database/README.md) - DB 스키마 확인 시
- [docs/API.md](docs/API.md) - API 엔드포인트 구현 시
- [docs/PATTERNS.md](docs/PATTERNS.md) - 공통 코드 패턴 확인 시

**토큰 절약 전략:**
- **TECHSPEC_SUMMARY.md (300줄)을 먼저 읽어 전체 맵 파악**
- TECHSPEC.md (1853줄)은 필요한 섹션만 선택적으로 읽기
- 예: Frontend 페이지 구현 → SUMMARY Section 4 확인 → 필요시 TECHSPEC Section 7 전체 읽기

```

### ▶ CLAUDE.MD 파일 일부

```

Step 5: Finalize and Commit

**중요**: 아래 순서를 **반드시 따라야** 합니다. 순서를 지키지 않으면 PLAN.md가 동기화되지 않습니다

# 5.1. 코드 변경사항 커밋

**개발 브랜치로 전환 및 동기화**:
* `git checkout dev` 명령어로 공용 개발 브랜치인 `dev`로 전환합니다.
* `git pull origin dev` 명령어로 최신 코드를 반영합니다.

**변경사항 스테이징 및 커밋 (코드만)**:
* `git add .` 명령어로 변경된 모든 파일을 스테이징합니다.
* `git commit -m "feat(app): description"` 형식으로 커밋 메시지를 작성하여 커밋합니다.
* ⚠️ **주의**: PLAN.md 파일은 **아직 커밋하지 않습니다**.

**원격 저장소에 푸시**:
* `git push origin dev` 명령어로 `dev` 브랜치에 변경사항을 푸시합니다.

**커밋 해시 저장**:
* `git rev-parse HEAD` 명령어로 방금 생성한 커밋의 고유 해시값을 가져옵니다.
* 예: `a1b2c3d` (처음 7자리만 사용)

```

### ▶ CLAUDE.MD 파일 일부, git commit 및 push 자동화

# Solution

## ▶ GCP, gcloud CLI로 배포 자동화.

현재 프로젝트는 GCP의 Cloud Run, Cloud SQL, VM 인스턴스 등을 활용하여 실제 프로덕션 환경에 배포되어 있습니다.

모든 배포는 AI 에이전트가 gcloud CLI 명령어를 활용하여 TECHSPEC.md의 아키텍처 명세를 바탕으로 자동 진행하였습니다.

```

Style Loading UI
[✓] 완료된 작업:
1. 프론트엔드 (StyleCreate.vue) 수정 완료
  - 1인 1스타일 제약 구현
  - pending/training 상태 시 블러 + 로딩 화면
  - completed 상태 시 수정 모드
2. 백엔드 (style.py) 수정 완료
  - 1인 1스타일 API 제약 추가
3. 프론트엔드 Firebase 배포 완료 [✓]

[🔄] 진행 중:
- 백엔드 Docker 이미지 빌드 중 (Step 4/16 완료, 패키지 설치 중)
- 예상 완료 시간: 약 2-3분

빌드가 완료되면 자동으로 Cloud Run에 배포하겠습니다.

BashOutput(Reading shell output)
L REMOTE BUILD OUTPUT -----
L starting build "c81e123e-4a38-4165-aabc-6c56dff6a0c"
  ... +478 Lines (ctrl+o to expand)
L Creating temporary archive of 17392 file(s) totalling 150.6 MiB before compression.
  Uploading tarball of [...] to [gs://noted-sled-478700-r3_cloudbuild/source/1764314684.027887-9a5a4355305f4fb9b2de593a7d27a859.tgz]
  Created [https://cloudbuild.googleapis.com/v1/projects/noted-sled-478700-r3/locations/global/builds/c81e123e-4a38-4165-aabc-6c56dff6a0c].
  ... +2 Lines (ctrl+o to expand)

빌드가 순조롭게 진행되고 있습니다! 현재 Step 6/16까지 완료되었고 Python 패키지들을 설치하고 있습니다. 조금만 더 기다리면 빌드가 완료될 것입니다.

진행 상황:
- [✓] Step 1-4: Base image 및 시스템 패키지 설치 완료
- [✓] Step 5: Requirements 파일 복사 완료
- [🔄] Step 6: Python 패키지 설치 중 (Django, DRF, psycopg2 등)
- [🔄] Step 7-16: 코드 복사 및 빌드 완료 대기 중

Bash(sleep 90 && echo "Build status check")
L Running in the background (down arrow to manage)

Thought for 1s (ctrl+o to show thinking)
* Sprouting... (esc to interrupt · 4m 27s · ↓ 11.0k tokens)
  
```

## ▶ AI 에이전트가 진행중인 GCP 배포

VM 인스턴스 인스턴스 만들기 VM 가져오기 새로고침 알아보기

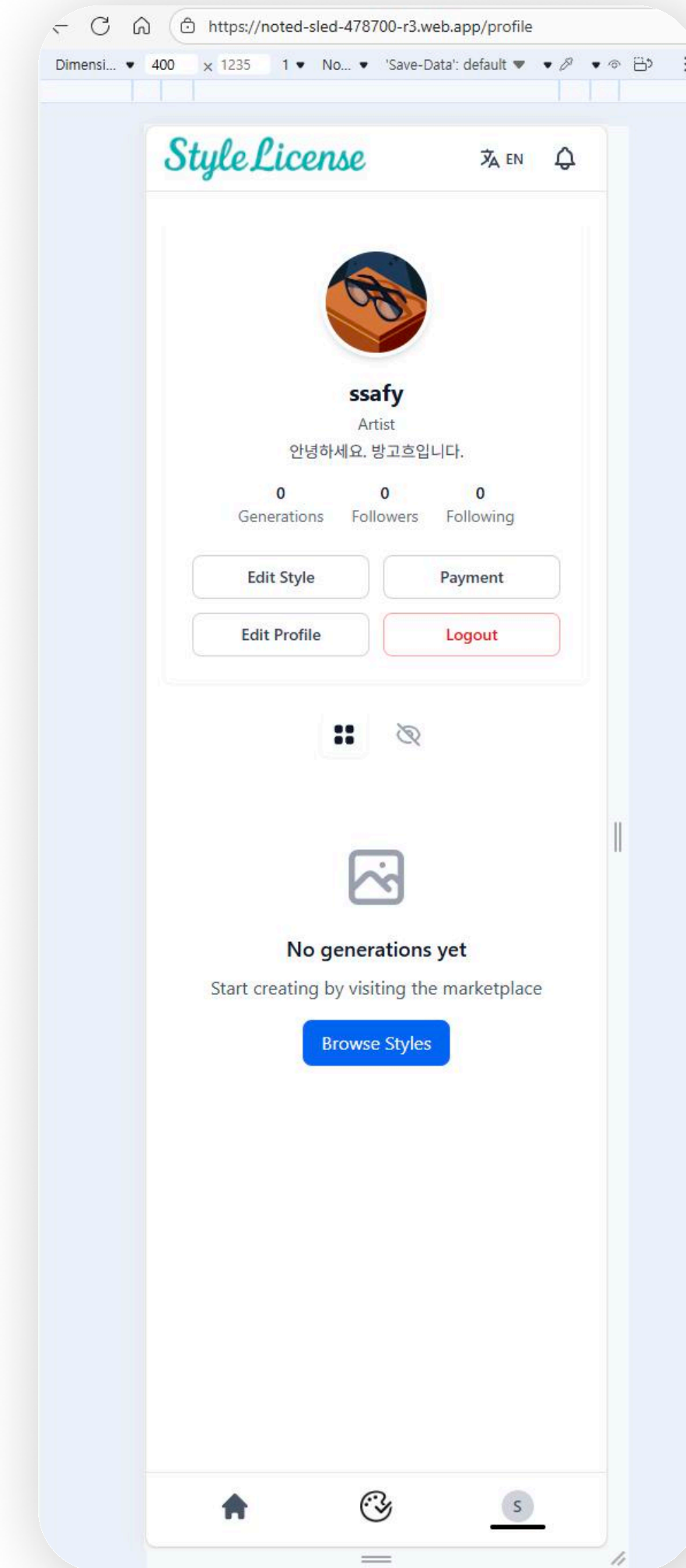
인스턴스 관측 가능성 인스턴스 일정

VM 인스턴스

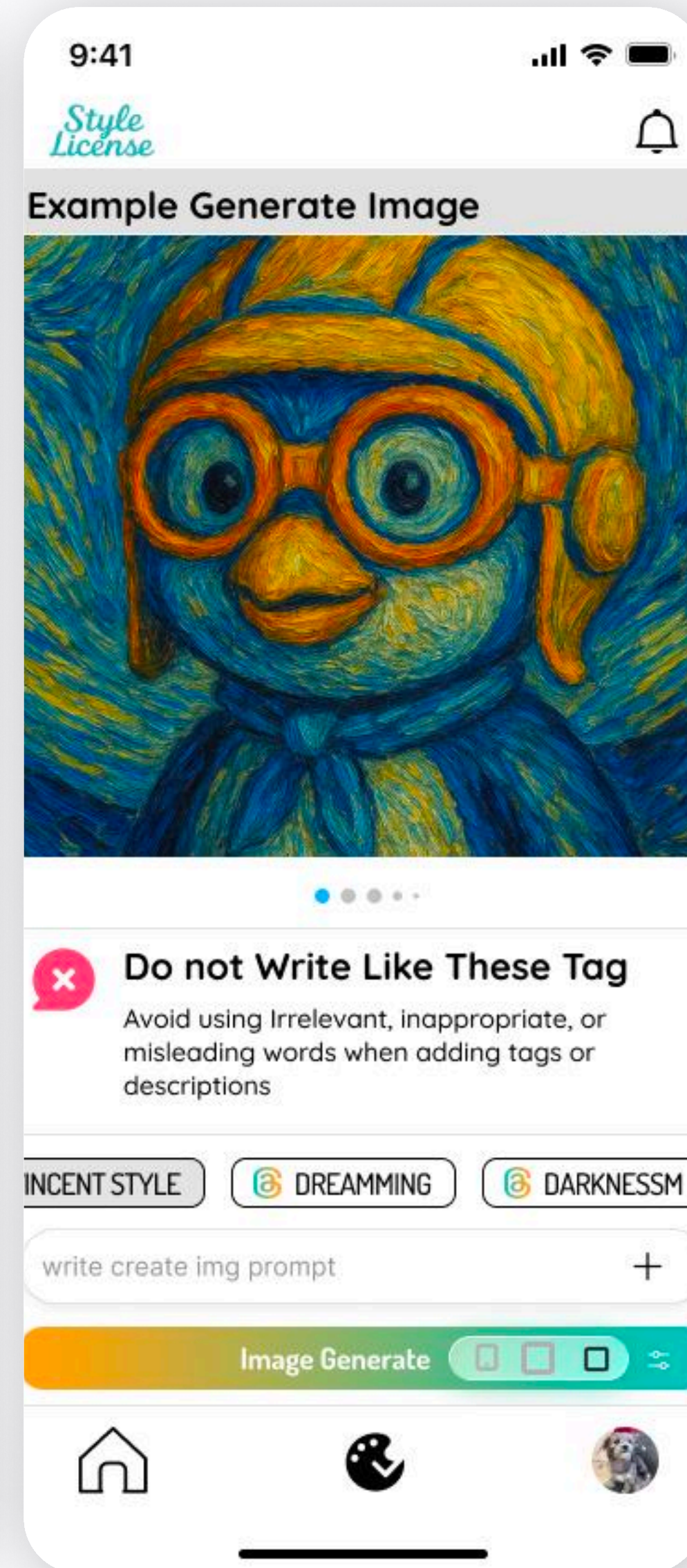
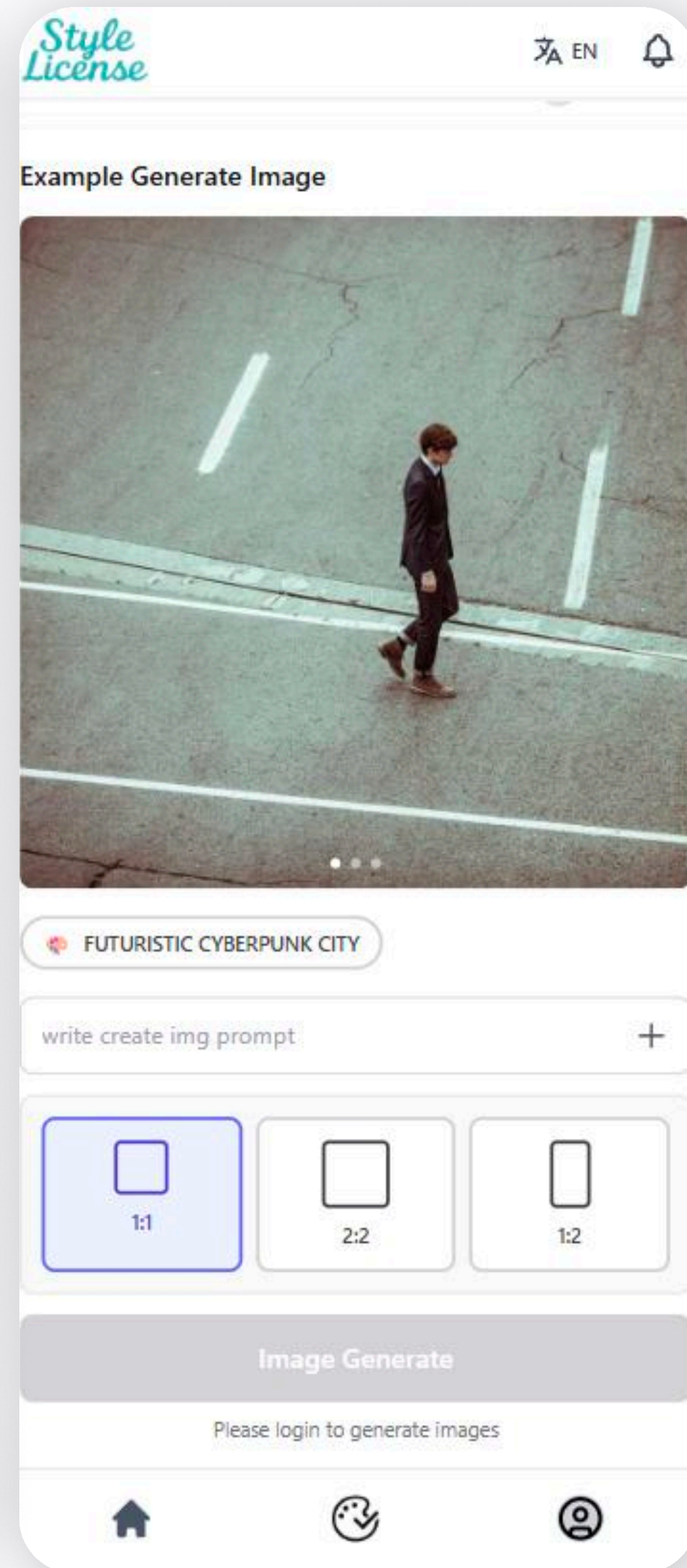
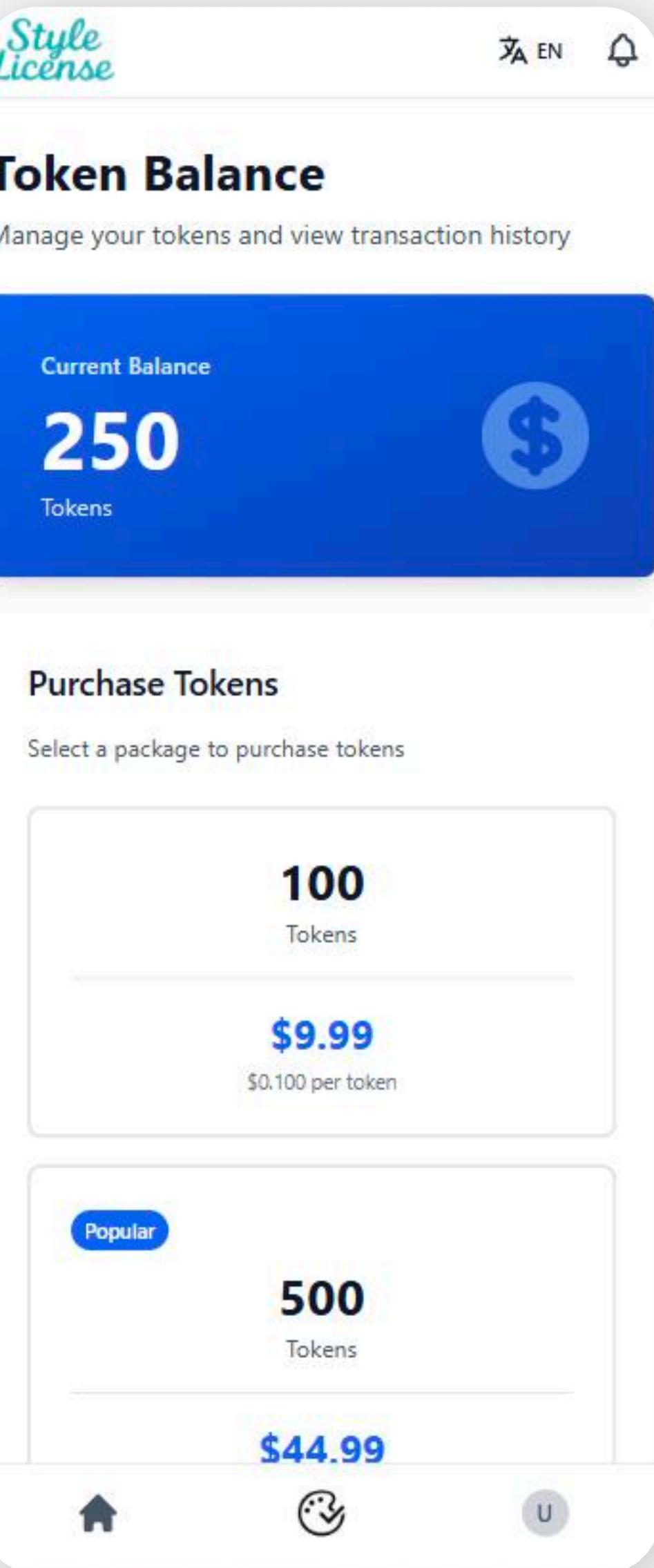
필터 속성 이름 또는 값 입력

상태	이름 ↑	영역	권장사항	다음에서 사용 중:	내부 IP	외부 IP	연결
<input checked="" type="checkbox"/>	<a href="#">rabbitmq-server</a>	asia-northeast3-a	매월 \$15 절감		10.178.15.222 <a href="#">(nic0)</a>	34.50.18.3 <a href="#">(nic0)</a>	SSH
<input type="checkbox"/>	<a href="#">style-license-gpu-server</a>	asia-east1-c			10.140.0.4 <a href="#">(nic0)</a>		SSH
<input type="checkbox"/>	<a href="#">style-license-inference-server</a>	asia-east1-c			10.140.0.5 <a href="#">(nic0)</a>		SSH

## ▶ 실제 VM 인스턴스 생성 목록



## ▶ Firebase로 AI 에이전트가 배포한 페이지



## Lesson Learn

### ▶ 성과 : 빠른 MVP 구축

멀티 서버 아키텍처, 클라우드 인프라, AI 파이프라인, 복잡한 비즈니스 로직 등 상당히 복잡한 시스템 구조를 가지고 있었지만, 구체적인 명세 덕분에 AI 에이전트가 일관되게 작업하며 빠르게 MVP 레벨까지 구축할 수 있었습니다.

### ▶ 예상과 달랐던 점: 코드 리뷰의 한계

AI가 코드를 작성하면 팀원들이 매번 리뷰하여 코드 품질을 보장하자는 계획은, AI가 생성하는 코드량이 너무 방대해 중간부터는 코드 리뷰가 사실상 불가능했습니다. 매 커밋마다 수백 줄에 달하는 코드를 하루에 10~20회 가량 업로드하였으며 이를 꼼꼼하게 리뷰하기 위해서는 한 개의 커밋 당 최소 1~2시간 가량이 소요되었습니다. 결과적으로 코드 리뷰보다는 동작 검증에 초점을 맞추며 프로젝트를 빌드업 해나갔습니다.

### ▶ 한계점 : 명세 위반과 보안사고

프로젝트 진행 중 AI가 TECHSPEC.md를 무시하고, Django의 User 모델의 일반 관습을 따라 임의로 DB 스키마를 변경한 사례가 존재하였습니다. CLAUDE.md에 명세 확인 절차를 명시하였지만, 추후에도 프로젝트가 길어질 때 비슷한 상황이 발생할지 모른다는 우려가 있습니다. 또, 핵심 API key를 GitHub에 그냥 커밋 하는 사고도 존재하였습니다. 앞으로 AI 에이전트를 프로젝트 제작에 활용하며 염두에 두어야 할 보안 사고였습니다.

# Thank you

새로운 것을 개발하고,  
코드 작성하는걸 사랑하는 개발자, 김연호

---

roger345@naver.com

010 • 3324 • 8711