

# DCT 비가시 워터마킹

## 이미지 기반 창작물 플랫폼

팀 명 : DDW  
지도교수 : 이병천 교수님  
팀 장 : 정민재  
팀 원 : 임건희  
유혁준

2025. 10.

중부대 정보보호학과

## 목차

|                                                             |    |
|-------------------------------------------------------------|----|
| 1. 서론 -----                                                 | 3  |
| 1.1 연구 배경 및 필요성                                             |    |
| 1.2 연구 목적 및 주제 선정 -----                                     | 4  |
| 2. 관련 이론 및 기술적 배경 -----                                     | 5  |
| 2.1 디지털 워터마킹 개념                                             |    |
| 2.2 DCT 기반 워터마킹 원리-----                                     | 6  |
| 2.2.1 중간 주파수 대역 계수 조작 기법                                    |    |
| 2.2.2 워터마크 삽입 및 복원 알고리즘 구조                                  |    |
| 2.3 관련 기술 스택 -----                                          | 7  |
| 2.3.1 프론트엔드: React/Next.js                                  |    |
| 2.3.2 백엔드: Flask, OpenCV, Tailwind CSS                      |    |
| 2.3.3 Watermark Engine: Python(OpenCV, NumPy, Pillow) ----- | 8  |
| 3. 본론: 시스템 설계 및 구현 -----                                    | 9  |
| 3.1 플랫폼 개요                                                  |    |
| 3.1.1 서비스 개념                                                |    |
| 3.1.2 사용자 흐름                                                |    |
| 3.2 주요 기능 설계 -----                                          | 16 |
| 3.2.1 회원가입 및 로그인 기능                                         |    |
| 3.2.2 이미지 업로드 기능                                            |    |
| 3.2.3 DCT 기반 비가시 워터마크 삽입 기능                                 |    |
| 3.2.4 워터마크 복호화 기능                                           |    |
| 3.3 핵심 구현 코드 -----                                          | 17 |
| 3.3.1 워터마크 삽입 알고리즘                                          |    |
| 3.3.2 워터마크 추출 알고리즘                                          |    |
| 3.4 배포 -----                                                | 18 |
| 4. 결론 및 향후 개선 방향 -----                                      | 19 |
| 5. 별첨 -----                                                 | 20 |
| 5.1 소개자료 (폼보드)                                              |    |
| 5.2 프로젝트 소개                                                 |    |
| 5.3 팀 소개                                                    |    |
| 5.4 발표자료 (발표ppt)                                            |    |

## 1. 서론

### 1.1 연구 배경 및 필요성

최근 영상 및 이미지 콘텐츠 시장의 급성장은 디지털 저작물의 불법 복제와 무단 유포 문제를 심화시키고 있다. 특히 개인 창작자 중심의 플랫폼에서는 저작권 침해가 빈번하게 발생하며, 원본 제작자의 권리를 보호하기 위한 기술적 수단이 필수적으로 요구된다.

이러한 환경에서 디지털 워터마킹(Watermarking) 기술은 콘텐츠 보안의 핵심으로 주목받고 있다. 워터마킹은 저작물 내에 고유한 식별 정보를 삽입함으로써, 외부에서 불법 복제나 배포가 발생했을 때 그 출처를 인증하고, 정당한 저작권을 증명할 수 있는 기술이다.

따라서 본 연구에서는 DCT(이산 코사인 변환, Discrete Cosine Transform) 기반의 이미지 비가시 워터마킹을 중심으로, 저작물 보호 및 추적 가능성을 확보하는 기술적 접근 방안을 탐구하고자 한다.

당초 본 프로젝트는 동영상 콘텐츠에 DCT 기반 워터마크를 삽입하는 플랫폼 개발을 목표로 하였으나, 실제 구현 과정에서 프레임 단위 유사도 인식의 정확도 문제와 연산 복잡도로 인해 안정적인 결과를 얻기 어려웠다. 이에 따라 본 연구에서는 우선적으로 정지 이미지 기반의 워터마킹 기술에 초점을 맞추어 구현의 완성도와 기술적 검증 가능성을 높이고, 향후 이를 동영상 프레임 단위로 확장하는 것을 중·장기적 발전 방향으로 설정하였다.

### 연구 목적 및 주제 선정

본 연구의 목적은 이미지 및 영상 저작물에 내재된 보안적 취약점을 보완하고, 불법 복제·배포로부터 원본 게시자의 권리를 보호할 수 있는 비가시 워터마크 삽입 기술을 설계·분석하는 것이다. 워터마크는 사용자 눈에 보이지 않지만, 기술적으로는 식별이 가능하도록 설계되어야 하며, 콘텐츠가 변형·압축되더라도 정보가 유지되어야 한다.

이를 위해 본 연구에서는 여러 워터마킹 방식 중 주파수 영역 기반 기법(DCT 방식)을 중심으로 선정하였다.

DCT 방식은 단순한 LSB(Least Significant Bit)와 달리 영상의 주파수 성분에 정보를 삽입하기 때문에, 압축, 필터링, 포맷 변환 등 후처리 과정에서도 높은 내구성을 보인다.

### 이미지 저작물의 보안 및 저작권 보호 필요성

디지털 이미지와 영상은 손쉽게 복제·변형·공유될 수 있는 특성을 가지며, 이로 인해 저작권 침해가 빈번하게 발생한다. 특히 SNS, 클라우드 스토리지, 파일 공유 플랫폼의 발달로 인해 저작물의 출처 추적이 더욱 어려워지고 있다.

워터마크 기술을 활용하면, 콘텐츠에 삽입된 고유 정보(예: 제작자 ID, 업로드 시각, 콘텐츠 해시 등)를 통해 원본 소유자를 명확히 식별하고, 유출 경로를 기술적으로 추적할 수 있다.

이는 단순한 법적 보호를 넘어, 디지털 포렌식 관점에서도 증거로 활용 가능하다는 점에서 중요한 의미를 지닌다.

또한, 비가시 워터마크는 시각적 품질을 유지하면서도 불법 복제 방지를 실현할 수 있어, 상업적

이미지 콘텐츠나 개인 저작물의 보안 강화 수단으로서 활용 가치가 높다.

이미지 콘텐츠 불법 복제 증가 예시 사진



### 1.2.2 DCT(이산 코사인 변환) 기반 비가시 워터마킹의 장점

DCT 기반 비가시 워터마킹은 영상 데이터를 주파수 영역으로 변환하여, 각 블록의 주파수 성분 중 중간 주파수대에 워터마크를 삽입하는 방식이다.

이 방식의 주요 장점은 다음과 같다.

#### - 높은 내구성

JPEG 압축, 노이즈 추가, 블러 처리 등과 같은 영상 후처리 과정에서도 워터마크가 손실되지 않고 유지될 가능성이 높다.

#### - 비가시성 유지

인간의 시각이 상대적으로 둔감한 주파수 영역에 워터마크를 삽입하기 때문에, 영상의 품질 저하가 거의 발생하지 않는다.

#### - 안정된 검출성

워터마크는 삽입 시 특정 주파수 계수를 기준으로 변경되며, 복호화 시 동일한 변환 과정을 통해 손쉽게 검출 가능하다.

#### - 적용 범위의 확장성

정지 이미지뿐 아니라 동영상의 프레임 단위로 적용할 수 있어, 플랫폼 단위의 콘텐츠 보호 시스템으로 확장 가능하다.

따라서 DCT 기반 워터마킹은 시각적 품질, 내구성, 보안성을 균형 있게 확보할 수 있는 효과적인 저작권 보호 기술로 평가된다. 본 연구에서는 이를 기반으로 한 동영상 플랫폼 내 비가시 워터마크 삽입 시스템의 설계 및 적용 가능성을 검토한다.

## 2. 관련 이론 및 기술적 배경

### 2.1 디지털 워터마킹 개념

디지털 워터마킹은 디지털 이미지, 영상, 오디오 등 디지털 콘텐츠에 식별 정보나 저작권 정보를 삽입하여, 콘텐츠의 불법 복제나 무단 배포를 방지하고 출처를 추적할 수 있도록 하는 기술이다. 워터마킹은 일반적으로 인간의 감각으로 인지할 수 없도록 삽입되며, 콘텐츠의 변형이나 압축이 발생하더라도 정보가 유지되도록 설계된다. 디지털 워터마킹은 저작권 보호, 불법 유포 추적, 콘텐츠 무결성 검증 등 다양한 분야에서 활용된다.

#### 2.1.1 가시 워터마킹과 비가시 워터마킹의 차이

가시 워터마킹(Visible Watermarking)은 이미지나 영상에 눈에 보이는 로고, 텍스트, 패턴을 삽입하는 방식이다. 장점으로는 콘텐츠의 사용자가 즉시 저작권 정보를 확인할 수 있다는 점이 있으나, 시각적 요소가 콘텐츠 품질을 저해할 수 있으며, 전문적인 편집 도구로 제거될 가능성이 존재한다.

반면 비가시 워터마킹은 인간의 감각으로 식별할 수 없는 방식으로 정보를 삽입한다. 시청 경험을 방해하지 않으면서, 추후 워터마크 복호화 과정을 통해 저작권자 정보, 업로더 ID, 업로드 시각 등 다양한 정보를 확인할 수 있다. 비가시 워터마킹은 주로 법적 증거, 출처 추적, 불법 복제 검증에 활용되며, 콘텐츠가 변형되더라도 워터마크의 견고성과 은닉성을 유지하는 것이 핵심 목표이다.

#### 2.1.2 공간 영역 워터마킹과 주파수 영역 워터마킹

공간 영역 워터마킹(Spatial Domain Watermarking)은 이미지의 픽셀 값 자체를 직접 조작하여 정보를 삽입하는 방식이다. 가장 대표적인 기법은 LSB(Least Significant Bit) 삽입으로, 이미지의 최소 단위 비트를 변경하여 데이터 은닉을 수행한다. 장점은 구현이 간단하고 연산 비용이 낮다는 점이며, 실시간 처리나 소규모 이미지에서 효율적이다. 그러나 이미지가 압축되거나 필터링, 리사이징 등 변형될 경우 워터마크가 손상될 가능성이 크다는 단점이 있다.

주파수 영역 워터마킹은 이미지를 주파수 성분으로 변환한 뒤, 특정 주파수 대역의 계수를 조작하여 정보를 삽입하는 방식이다. 대표적인 변환 방법으로는 DCT(Discrete Cosine Transform), DWT(Discrete Wavelet Transform), FFT(Fast Fourier Transform) 등이 있다. 주파수 영역 방식은 압축, 리사이징, 잡음 추가 등 다양한 변형에도 상대적으로 강인성을 가지며, 이미지 품질 저하가 적어 비가시 워터마킹에 적합하다. 특히 DCT 기반 기법은 JPEG 압축과 호환성이 높아 이미지 유통 환경에서 실용적으로 활용된다.

공간 영역과 주파수 영역 방식의 선택은 목표하는 견고성, 구현 난이도, 처리 속도 등에 따라 달라지며, 본 연구에서는 변형에 대한 강인성과 비가시성을 고려하여 DCT 기반 주파수 영역 워터마킹을 중심으로 구현을 진행한다.

## 2.2 DCT 기반 워터마킹 원리

Dct 변환이란?

DCT 변환은 이산 코사인 변환이라고도 하며, 영상 압축을 위한 방식중의 하나이다. 자연 영상을 NxN 블록으로 공식을 사용해 변환하면, 분포되어 있던 화소값이 변환 후 이웃 화소 간 별 차이 없는 저주파 영역에 집중되는 것을 이용하여 압축을 시도하는 방식이다. DCT 방식을 사용하면 중복성 제거를 이용하며, 인간의 시각에는 둔감한 고주파 성분을 제거함으로써 압축이 가능해진다. 입력 화소값을 INPUT, 변환값을 OUTPUT라고 했을 때, 변환 식은 다음과 같다.

(INPUT, OUTPUT의 입력값의 픽셀 좌표를 각각 (i, j), (x, y) 라 한다)

$$OUTPUT_{xy} = C_x \times C_y \times \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (INPUT_{ij} \times \cos \frac{(2j+1)y\pi}{2N} \times \cos \frac{(2i+1)x\pi}{2N})$$

$$C_{x \text{ or } y} = \sqrt{\frac{1}{N}} (x \text{ or } y = 0) , \quad C_{x \text{ or } y} = \sqrt{\frac{2}{N}} (x \text{ or } y > 0)$$

또한, 역 DCT 변환식은 다음과 같이 쓴다.

$$INPUT_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x \times C_y \times OUTPUT_{xy} \times \cos \frac{(2j+1)y\pi}{2N} \times \cos \frac{(2i+1)x\pi}{2N}$$

이러한 수학적 계산(이산 코사인 변환)을 통해 공간 영역을 주파수 영역으로 변환한다.

### 2.2.1 중간 주파수 대역 계수 조작 기법

DCT 기반 워터마킹에서 워터마크는 이미지의 중간 주파수 대역 계수를 선택하여 삽입한다. 저주파 계수는 이미지의 주요 시각 정보를 담고 있어 변경 시 눈에 띄는 왜곡을 발생시키고, 고주파 계수는 압축이나 노이즈에 취약하므로 안정적인 삽입을 위해 중간 주파수 계수가 활용된다. 워터마크 삽입 과정은 일반적으로 다음과 같이 수행된다.

- 이미지를 블록 단위로 나누고 각 블록에 DCT를 적용하여 주파수 계수를 얻는다.
- 중간 주파수 영역의 특정 계수를 선택하고, 워터마크 비트 값(0 또는 1)에 따라 계수를 조정한다. 예: 워터마크 비트가 1이면 계수 값을 기준치 이상으로, 0이면 기준치 이하로 변경
- 변경된 계수를 역DCT(IDCT)를 통해 다시 공간 영역으로 변환하여 은닉된 워터마크가 포함된 이미지를 생성한다.
- 이 방식은 시각적 품질을 크게 손상시키지 않으면서, JPEG 압축이나 일반적인 이미지 변형에도 워터마크를 비교적 안정적으로 유지할 수 있다는 장점이 있다.

### 2.2.2 워터마크 삽입 및 복원 알고리즘 구조

워터마크 삽입과 복원은 다음과 같은 단계로 구성된다.

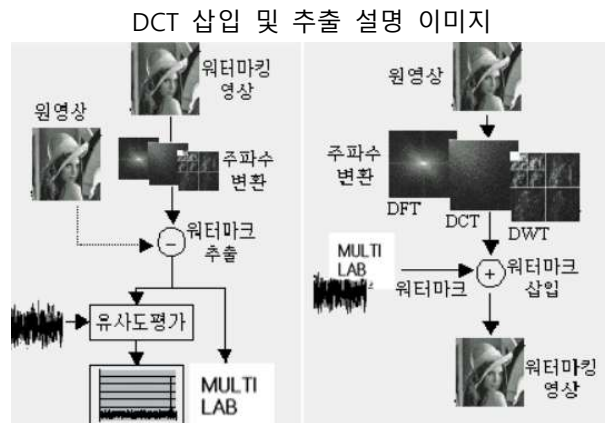
#### - 삽입 과정

입력 이미지 로드 및 블록 분할 -> 각 블록에 DCT 적용 -> 중간 주파수 계수 선택 -> 워터마크 비트에 따라 계수 값 수정 -> IDCT를 통해 공간 영역 이미지 생성 -> 수정된 이미지 저장

#### - 복원 과정

워터마크가 삽입된 이미지 로드 및 동일한 블록 분할 -> 각 블록에 DCT 적용 -> 중간 주파수 계수 값 확인 -> 계수 값과 삽입 규칙을 비교하여 워터마크 비트 추출 -> 추출된 비트를 조합하여 원래의 워터마크 데이터 복원

이 구조를 통해 삽입된 워터마크는 이미지 시각적 품질을 크게 해치지 않으면서, 외부 변형이나 압축에도 강인성을 갖도록 설계된다.



## 2.3 관련 기술 스택

### 2.3.1 프론트엔드: React / Next.js

본 프로젝트의 프론트엔드는 Next.js 프레임워크를 기반으로 React 라이브러리를 활용하여 구현되었다.

Next.js는 React의 장점을 유지하면서도 서버 사이드 렌더링(SSR)과 정적 페이지 생성을 지원하기 때문에, 이미지 기반 창작물 플랫폼과 같이 사용자 상호작용이 많고 콘텐츠 로딩 속도가 중요한 서비스 환경에 적합하다.

또한, React의 컴포넌트 기반 구조를 통해 페이지 구성 요소(예: 로그인, 회원가입, 이미지 업로드, 갤러리 등)를 독립적으로 관리하고 재사용성을 높였다.

Next.js의 라우팅 시스템을 이용하여 사용자가 /login, /signup, /upload, /gallery 등 명확한 경로로 페이지를 이동할 수 있도록 구현하였다.

이를 통해 사용자 경험(UX)을 단순화하고, 유지보수성을 향상시켰다.

### 2.3.2 백엔드: Flask, OpenCV, Tailwind CSS

백엔드는 Flask(Python 기반 경량 웹 프레임워크)를 사용하여 구현되었다.

Flask는 RESTful API를 구축하기 용이하고, 워터마킹 엔진과 같은 독립적인 Python 모듈과의 연동이 간단하다는 장점이 있다.

이미지 업로드, 워터마크 삽입 및 추출 요청 등을 처리하며, 클라이언트(Next.js)와 JSON 기반으로 통신한다.

또한 OpenCV는 이미지의 이산 코사인 변환(DCT) 연산을 수행하는 데 사용되며, 워터마킹 알고리즘의 핵심 역할을 담당한다.

이미지 데이터를 블록 단위로 분해하여 주파수 영역에서 특정 계수를 조작하고, 그 결과를 역변환(IDCT)하여 사람이 인식하지 못하는 형태로 메시지를 숨길 수 있도록 한다.

프론트엔드 시각 구성에는 Tailwind CSS를 적용하였다.

Tailwind는 클래스 기반의 유틸리티 퍼스트(Utility-first) CSS 프레임워크로, 빠르게 반응형 디자인을 구현할 수 있다.

이를 통해 업로드 페이지, 로그인 폼, 이미지 갤러리 등의 화면 요소를 일관성 있게 구성하였다.

### **2.3.3 Watermark Engine: Python (OpenCV, NumPy, Pillow)**

워터마킹의 핵심 처리 로직은 Python 기반의 워터마크 엔진(Watermark Engine) 모듈로 구현되었다.

이 엔진은 DCT(이산 코사인 변환)를 이용하여 이미지의 주파수 영역에 텍스트 메시지를 삽입하거나 추출하는 기능을 수행한다.

OpenCV: 이미지의 DCT 및 IDCT(역 DCT) 연산 수행

NumPy: 픽셀 데이터를 효율적으로 처리하기 위한 수치 연산 지원

Pillow(PIL): 이미지 입출력 및 포맷 변환, 색상 모드 처리 등

워터마크 삽입 과정에서는 이미지를 8×8 블록 단위로 분할하고, 각 블록의 중간 주파수 영역 계수를 조작하여 비트 정보를 삽입한다.

이러한 방식은 시각적으로 감지되지 않으며, 일반적인 JPEG 압축이나 이미지 편집에 대해서도 비교적 높은 내구성을 가진다.

또한 워터마크 복호화 기능은 삽입 과정의 역연산을 수행하여, 이미지 내부에 숨겨진 텍스트 메시지를 복원한다.

이 모듈은 커맨드라인 기반으로도 동작 가능하며, Flask 서버를 통해 웹 서비스 환경에서도 자동으로 호출되어 작동한다.

### 3. 본론: 시스템 설계 및 구현

#### 3.1 플랫폼 개요

##### 3.1.1 서비스 개념

본 프로젝트의 플랫폼은 이미지 기반 디지털 저작물 보호 웹 서비스로, 사용자가 직접 제작한 이미지에 비가시 워터마크를 삽입하고, 이를 안전하게 관리·검증할 수 있도록 설계되었다. 플랫폼의 주요 목적은 이미지 저작물의 불법 복제와 무단 배포를 방지하고, 필요 시 원본 출처를 추적할 수 있는 출처 인증 및 저작권 보호 기능을 제공하는 것이다.

서비스는 다음과 같은 특징을 가진다.

- 비가시 워터마킹 지원: 사용자가 업로드한 이미지에 눈으로 식별할 수 없는 형태로 워터마크를 삽입하여 시각적 품질 저하 없이 출처 정보를 저장.
- DCT 기반 처리: 이미지의 주파수 영역을 활용하여 워터마크를 삽입, 일반적인 이미지 변환이나 압축에도 강인성을 확보.
- 사용자 인증 및 접근 제어: 회원 가입과 로그인 기능을 통해 권한 있는 사용자만 워터마크 삽입과 추출 기능을 이용 가능.
- 실시간 검증: 삽입된 워터마크를 즉시 확인하고, 복호화를 통해 출처 정보를 추적할 수 있는 인터페이스 제공.

##### 3.1.2 사용자 흐름

사용자 흐름은 먼저 일반 사용자의 경우

‘회원가입 → 로그인 → 이미지 업로드 → 워터마크 자동 삽입 → 콘텐츠 등록 → 워터마크 검증’  
↳ 콘텐츠 조회 및 다운로드

무단 도용 시나리오로는

‘타인의 이미지 업로드 → 서버에서 워터마크 자동 추출 → 게시자 정보와 비교 → 불일치시 의심 콘텐츠로 등록’

관리자의 경우는 ‘관리자 계정 로그인 – 관리자 계정 관리 및 도용 의심 콘텐츠 관리’  
순의 흐름으로 진행된다.

회원가입 및 로그인: 사용자는 이메일, 사용자명, 비밀번호를 입력하여 계정을 생성하고, 로그인 시 JWT(JSON Web Token)를 발급받아 이후 모든 API 호출에서 인증 수단으로 사용한다.

이미지 업로드: 인증된 사용자는 플랫폼에 이미지를 업로드하며, 업로드 시 이미지의 해상도, 포맷, 파일명을 선택할 수 있다.

워터마크 삽입: 업로드된 이미지는 서버에서 DCT 기반 비가시 워터마킹 처리 과정을 거쳐 워터마크가 삽입된다. 삽입 과정에서 원본 이미지의 시각적 품질은 유지되며, 삽입된 워터마크는 사용자명, 업로드 시각 등 식별 정보를 포함한다.

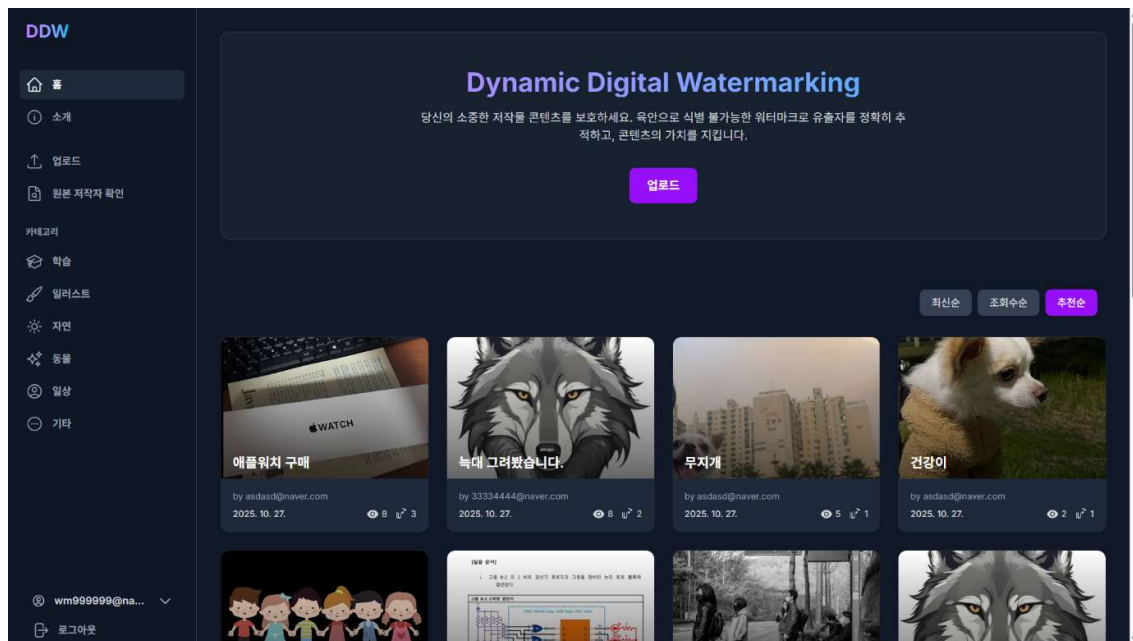
결과 확인 및 저장: 워터마크가 삽입된 이미지는 사용자가 다운로드하거나 플랫폼 내에서 바로 확인할 수 있으며, 서버에는 삽입 완료 이미지와 메타 정보를 기록하여 추후 검증에 활용된다.

워터마크 검증: 필요 시 사용자는 삽입된 이미지를 선택하여 워터마크를 복호화하고, 원본 출처 및 업로더 정보를 확인할 수 있다. 이 과정은 API를 통해 자동화되며, 검증 결과는 사용자에게 직관적인 형태로 제공된다.

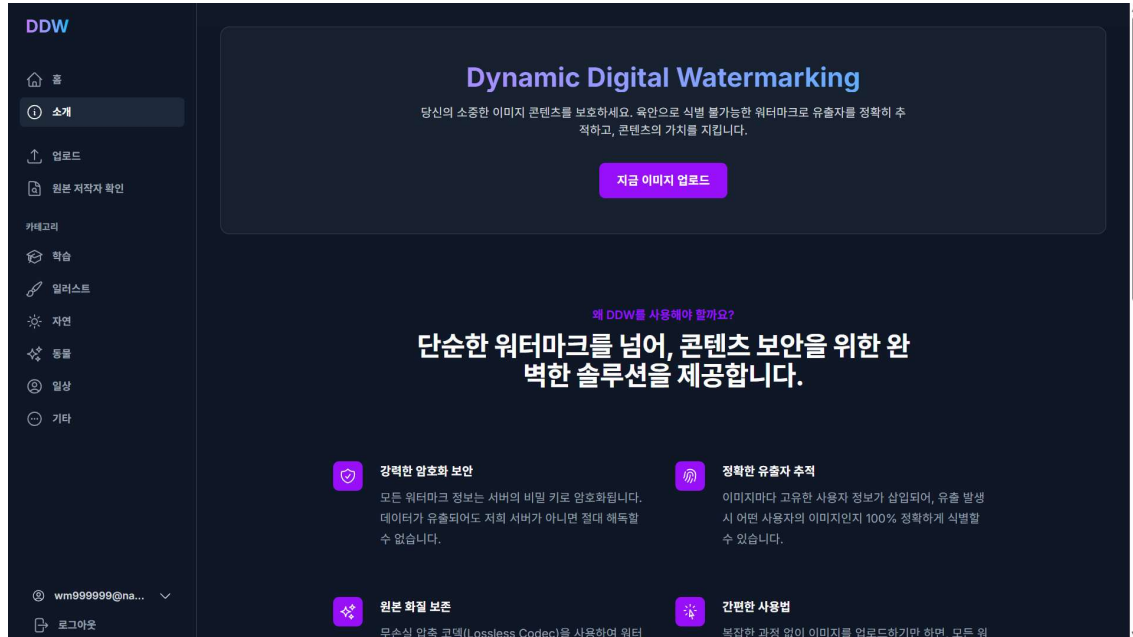
이러한 사용자 흐름 설계를 통해, 플랫폼은 단순한 이미지 업로드 기능을 넘어 저작권 보호, 출처 인증, 검증 기능을 통합한 실용적 웹 서비스로 구현될 수 있다.

다음은 서비스의 실제 화면이다.

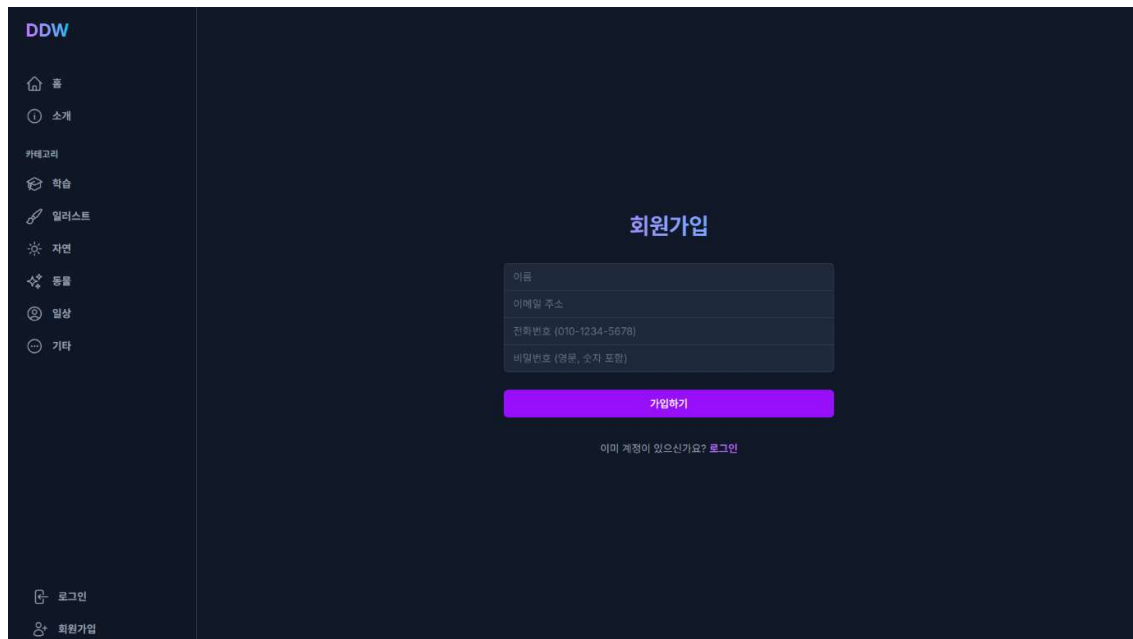
## 메인 홈 페이지



## 소개 페이지



## 회원가입 페이지



## 로그인 페이지

DDW

홈

소개

카테고리

학습

일러스트

자연

동물

일상

기타

로그인

회원가입

로그인

wm999999@naver.com

\*\*\*\*\*

로그인

계정이 없으신가요? 회원가입

## 콘텐츠 업로드 페이지

DDW

홈

소개

업로드

원본 저작자 확인

카테고리

학습

일러스트

자연

동물

일상

기타

wm999999@na... ▼

로그아웃

창작물 업로드

원본 이미지 업로드

파일 선택 선택된 파일 없음

제목

게시물 제목을 입력하세요

카테고리

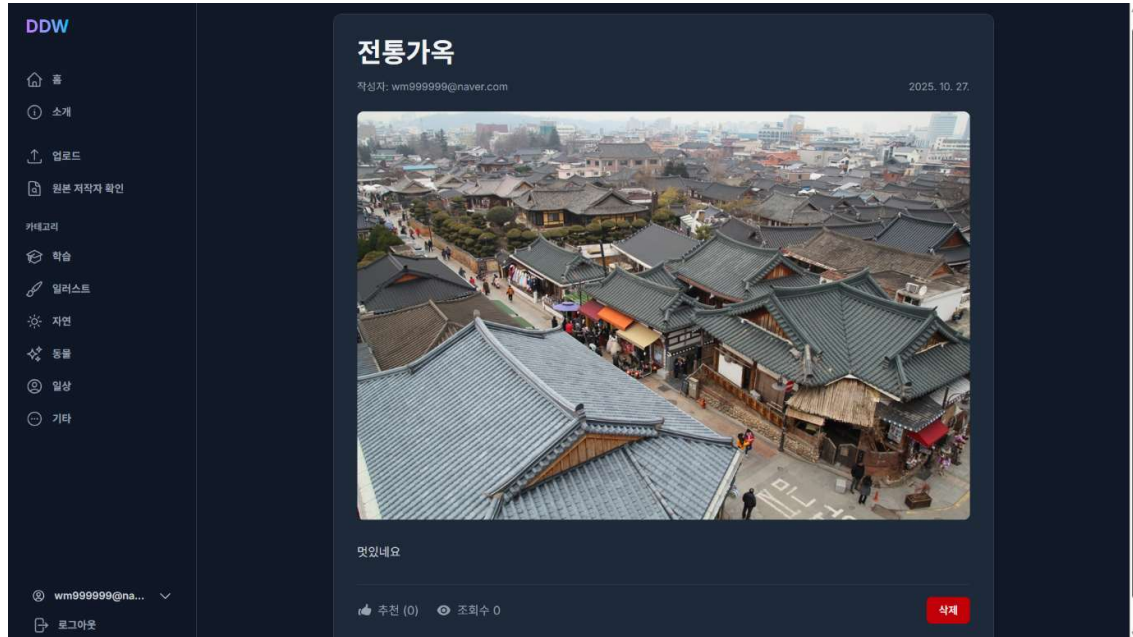
카테고리를 선택하세요 ▼

내용

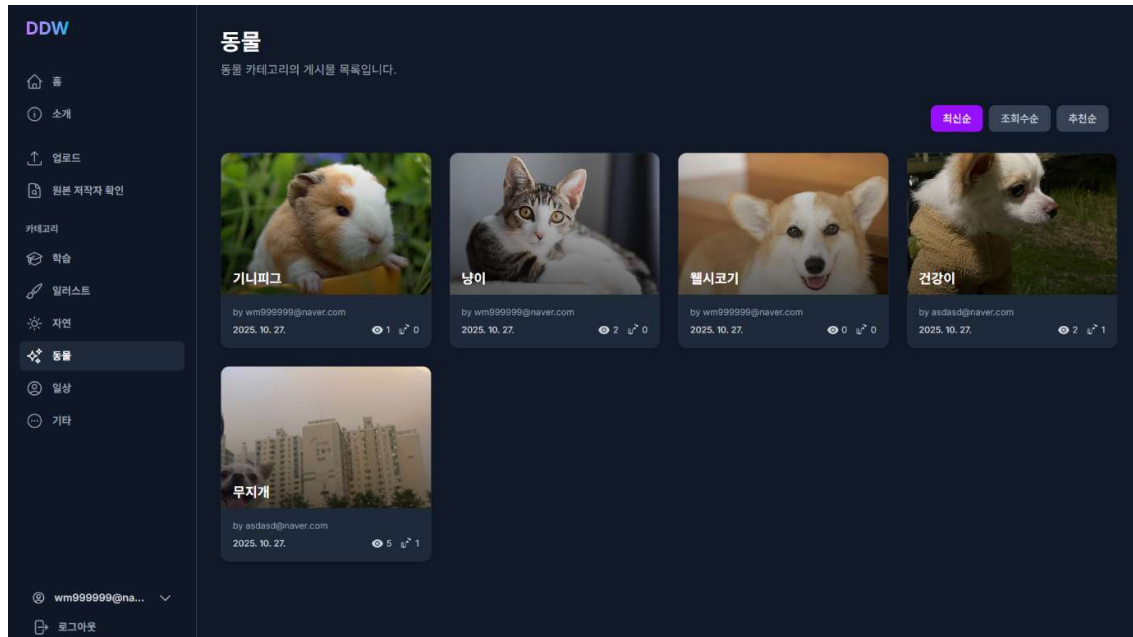
게시물 내용을 입력하세요

게시물 생성

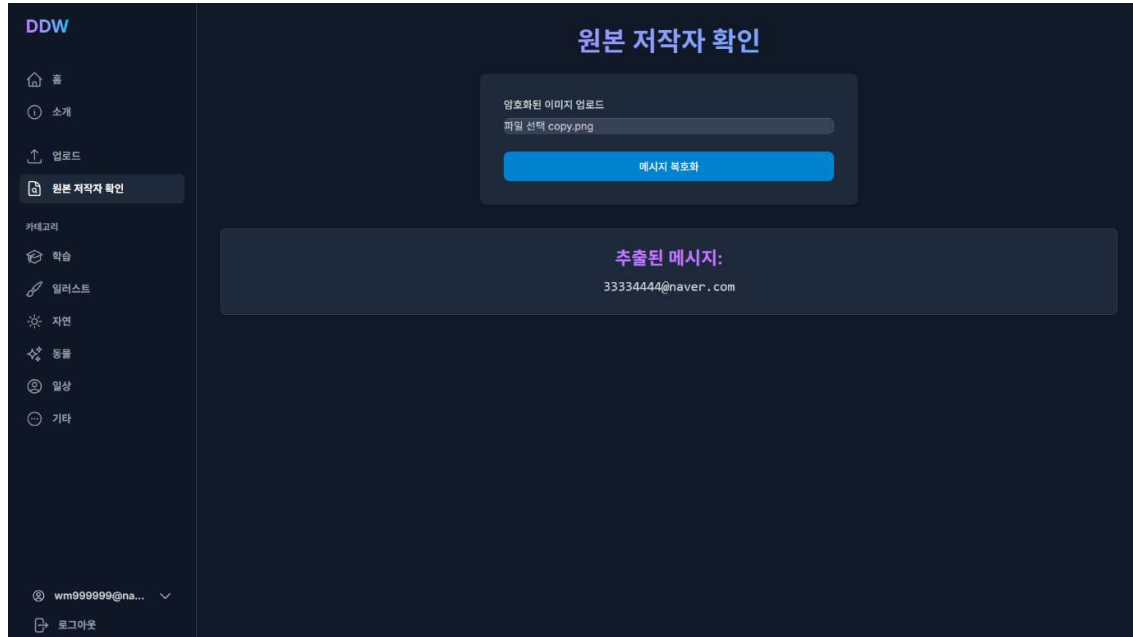
## 콘텐츠 페이지



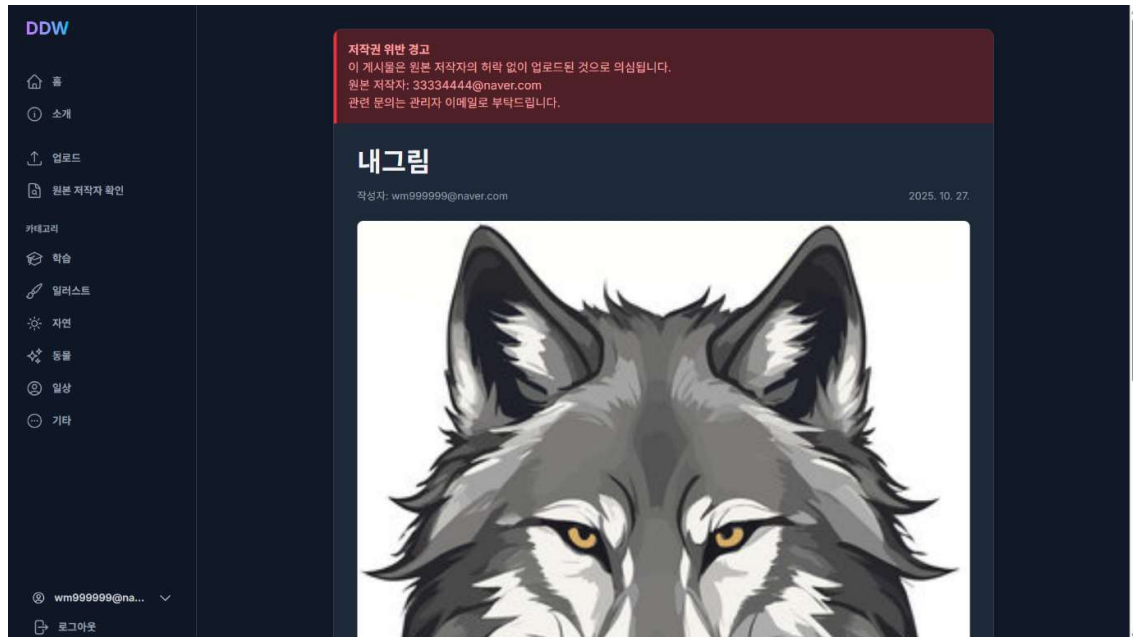
## 카테고리 별 페이지



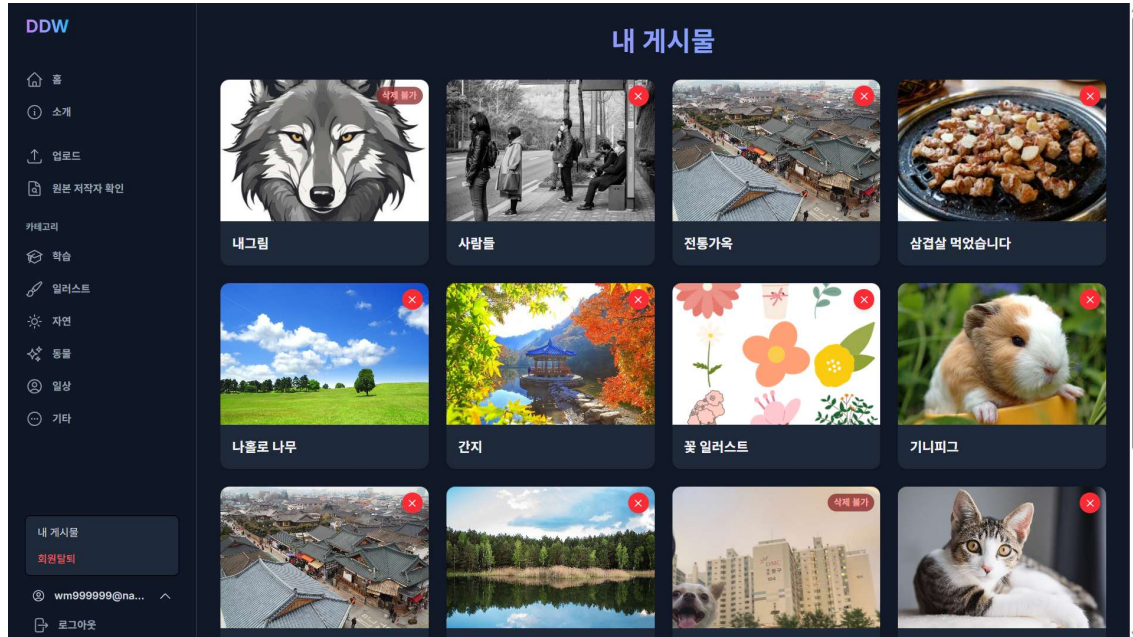
## 워터마크 추출 페이지



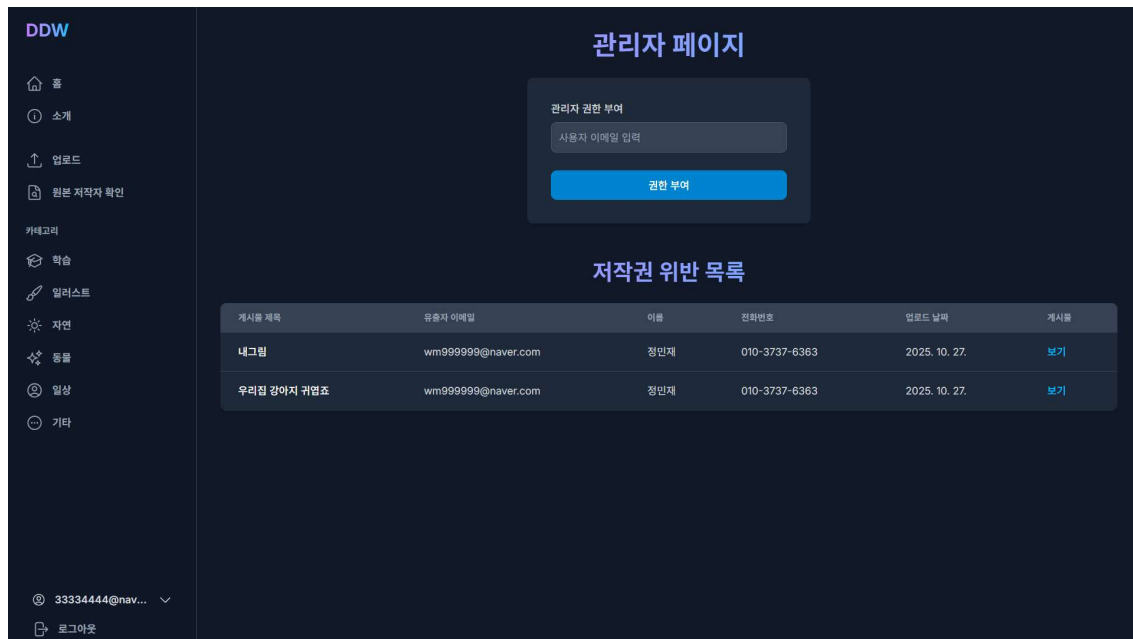
## 도용 의심 페이지



## 나의 게시물 페이지



## 관리자 페이지



## 3.2 주요 기능 설계

### 3.2.1 회원가입 및 로그인 기능

플랫폼의 핵심 기능 중 하나는 사용자 인증으로, 워터마크 삽입과 추출 기능이 인증된 사용자에게만 제공되도록 설계되었다. 회원가입 과정에서는 이메일, 사용자명, 비밀번호를 입력하여 계정을 생성하며, 비밀번호는 해시 암호화(Bcrypt)를 통해 안전하게 저장된다. 로그인 시에는 입력한 사용자 정보가 데이터베이스와 일치하는지 검증하고, 성공 시 JWT(JSON Web Token)를 발급하여 이후 API 호출 시 인증 토큰으로 활용한다. 이로 인해 워터마크 삽입 및 복호화 기능은 권한 있는 사용자만 접근 가능하며, 관리자 계정은 추가적인 관리 기능(사용자 관리, 이미지 관리)을 수행할 수 있다.

### 3.2.2 이미지 업로드 기능

사용자는 플랫폼에 이미지를 업로드할 수 있으며, 업로드 시 이미지의 해상도, 포맷, 파일명 정보를 선택할 수 있다. 서버는 업로드된 파일의 SHA-256 해시를 계산하여 중복 업로드를 방지하며, 이미지 파일이 워터마크 처리 가능 여부를 확인한다. 업로드된 이미지는 임시 저장 후 DCT 기반 워터마킹 처리를 거치며, 원본 이미지 손실을 최소화하면서 삽입 과정을 준비한다. 또한, 프론트엔드에서는 업로드 진행 상태 표시, 썸네일 생성, 오류 안내 등의 UI/UX 기능을 제공하여 사용자가 업로드 과정을 직관적으로 확인할 수 있도록 설계되었다.

### 3.2.3 DCT 기반 비가시 워터마크 삽입 기능

워터마크 삽입 기능은 이미지의 DCT 변환 후 중간 주파수 계수에 사용자 식별 정보를 은닉하는 방식으로 구현된다. 삽입 과정은 다음과 같다.

- 업로드된 이미지를 8×8 블록 단위로 분할 후 DCT 변환을 수행한다.
- 각 블록의 중간 주파수 계수를 선택하여 워터마크 데이터를 삽입한다. 삽입 데이터는 업로더의 사용자명, 업로드 시각 등 고유 정보를 포함한다.
- 수정된 DCT 계수를 역DCT로 변환하여 원본 이미지와 유사한 품질로 워터마크가 삽입된 이미지를 생성한다.
- 삽입 완료 후 서버는 마스터 이미지 파일과 사용자 확인용 이미지를 생성하여 저장하고, 삽입 정보를 데이터베이스에 기록한다.

이 과정은 비가시성을 유지하면서도 워터마크의 추적 가능성을 확보하도록 설계되었으며, 이미지 압축이나 일부 변형에도 내구성을 갖도록 중간 주파수 대역을 활용하였다.

### 3.2.4 워터마크 복호화 기능

워터마크 복호화 기능은 삽입된 이미지를 대상으로 원본 출처 정보와 업로더 식별 정보를 추출하는 과정이다. 구체적 단계는 다음과 같다.

- 사용자가 복호화를 요청하면 서버는 대상 이미지를 8×8 블록 단위로 DCT 변환한다.
- 중간 주파수 계수에서 삽입된 워터마크 데이터를 추출한다.

- 추출된 워터마크는 복호화 및 디코딩 과정을 거쳐 JSON 형태로 변환되며, 업로더명과 업로드 시각 등 원본 정보를 확인할 수 있다.
- 결과는 프론트엔드에서 직관적인 UI로 제공되어, 사용자는 이미지를 확인하며 원본 출처를 검증할 수 있다.

이 기능을 통해 플랫폼은 단순한 이미지 저장소를 넘어 저작권 보호 및 불법 유출 추적이 가능한 실용적 환경을 제공하며, 이미지 기반 창작물의 안전한 배포와 관리가 가능하다.

### 3.3 핵심 구현 코드

#### 3.3.1 워터마크 삽입 알고리즘

워터마크 삽입 알고리즘은 이미지를 8×8 블록 단위로 DCT 변환한 후, 중간 주파수 계수에 사용자 정보를 삽입하는 방식으로 구현된다. Python과 OpenCV를 기반으로 한 구현 예시는 다음과 같다.

```
def encrypt(self, image_path: str, message: str, output_path: str, output_format: str) -> None:
    img = self._prepare_image(image_path)
    height, width, channels = img.shape

    # 메시지를 비트 시퀀스로 변환
    bits = self._string_to_bits(message)

    # 삽입할 수 있는 최대 비트 수 계산
    blocks_height = height // self.BLOCK_SIZE
    blocks_width = width // self.BLOCK_SIZE
    max_bits = blocks_height * blocks_width * channels

    if len(bits) > max_bits:
        raise ValueError("메시지 최대 길이 초과, 최대 비트: {max_bits}, 필요: {len(bits)}")

    bit_index = 0
    modified_img = np.copy(img)

    for channel in range(3): # RGB 채널
        if bit_index >= len(bits):
            break
        channel_data = modified_img[:, :, channel]

        for y in range(0, blocks_height * self.BLOCK_SIZE, self.BLOCK_SIZE):
            for x in range(0, blocks_width * self.BLOCK_SIZE, self.BLOCK_SIZE):
                if bit_index >= len(bits):
                    break
                block = channel_data[y+y*self.BLOCK_SIZE, x:x+self.BLOCK_SIZE]

                # 비트 삽입
                modified_block = self._embed_bit_in_dct_block(block, bits[bit_index])
                modified_img[y+y*self.BLOCK_SIZE, x:x+self.BLOCK_SIZE, channel] = modified_block

                bit_index += 1

    # 이미지 저장
    self._save_image(modified_img, output_path, output_format)

def _embed_bit_in_dct_block(self, block: np.ndarray, bit: int) -> np.ndarray:
    block_float = np.float32(block)
    dct_block = cv2.dct(block_float)

    # 중간 주파수 계수 (4,4)에 비트 삽입
    if bit == 1:
        if dct_block[4, 4] < self.THRESHOLD:
            dct_block[4, 4] = self.THRESHOLD + self.QUANTIZATION_FACTOR
        else:
            if dct_block[4, 4] > -self.THRESHOLD:
                dct_block[4, 4] = -self.THRESHOLD - self.QUANTIZATION_FACTOR

    idct_block = cv2.idct(dct_block)
    return np.clip(idct_block, 0, 255).astype(np.uint8)
```

이 알고리즘은 중간 주파수 계수 조작을 통해 워터마크를 삽입하며, 이미지 품질 저하를 최소화하면서 삽입 정보를 은닉한다.

#### 3.3.2 워터마크 추출 알고리즘

워터마크 추출 알고리즘은 삽입된 이미지를 다시 DCT 변환하여 중간 주파수 계수에서 숨겨진 비트를 읽어 복호화한다.

```

def decrypt(self, image_path: str) -> str:
    img = self._prepare_image(image_path)
    height, width, channels = img.shape

    blocks_height = height // self.BLOCK_SIZE
    blocks_width = width // self.BLOCK_SIZE

    extracted_bits = []
    consecutive_ones = 0

    for channel in range(3): # RGB 채널
        channel_data = img[:, :, channel]

        for y in range(0, blocks_height * self.BLOCK_SIZE, self.BLOCK_SIZE):
            for x in range(0, blocks_width * self.BLOCK_SIZE, self.BLOCK_SIZE):
                if consecutive_ones >= 16: # 종료 시퀀스 확인
                    break

                block = channel_data[y:y+self.BLOCK_SIZE, x:x+self.BLOCK_SIZE]
                bit = self._extract_bit_from_dct_block(block)
                extracted_bits.append(bit)

                if bit == 1:
                    consecutive_ones += 1
                else:
                    consecutive_ones = 0

            if consecutive_ones >= 16:
                break
        if consecutive_ones >= 16:
            break

    if consecutive_ones >= 16:
        extracted_bits = extracted_bits[:-consecutive_ones]

    return self._bits_to_string(extracted_bits)

def _extract_bit_from_dct_block(self, block: np.ndarray) -> int:
    block_float = np.float32(block)
    dct_block = cv2.dct(block_float)
    return 1 if dct_block[4, 4] > 0 else 0

```

이 과정을 통해 사용자는 삽입된 워터마크 정보를 정확하게 추출할 수 있으며, 삽입된 이미지의 출처와 업로더 정보를 확인할 수 있다.

이 두 알고리즘을 결합하면, 플랫폼 내 이미지 저작권 보호 및 불법 유출 추적 시스템이 실질적으로 구현된다.

### 3.4 배포

개발된 웹 서비스는 Vercel을 통해 성공적으로 배포되었다. Vercel은 Next.js 기반 프로젝트에 최적화된 클라우드 배포 플랫폼으로, GitHub 연동만으로 자동 빌드 및 배포가 가능하다. 프론트엔드는 React와 Next.js를 이용해 구현하였으며, 백엔드는 Flask를 기반으로 OpenCV, NumPy, Pillow를 활용한 워터마크 삽입 및 추출 기능을 API 형태로 제공한다. 또한 Next.js의 프록시 설정을 통해 클라이언트 요청이 Flask 서버로 전달되도록 구성하여 워터마크 삽입 및 복호화 기능을 안정적으로 구현하였다. 이를 통해 사용자는 웹 브라우저 환경에서 이미지 업로드, 워터마크 삽입, 복호화 과정을 실시간으로 수행할 수 있으며, 클라우드 배포를 통해 접근성과 유지보수성을 높인 것이 본 프로젝트의 주요 성과라 할 수 있다.

## 4. 결론 및 향후 개선 방향

### 연구 성과 및 의의

본 연구에서는 이미지 저작물의 불법 복제와 무단 배포를 방지하기 위해 DCT(이산 코사인 변환) 기반 비가시 워터마킹 기술을 설계하고 구현하였다. 해당 기술은 이미지의 주파수 영역에 워터마크를 삽입함으로써, 눈에 보이지 않으면서도 필요할 때 원본 정보 복원이 가능한 구조를 갖는다. 이를 Python(OpenCV, NumPy, Pillow) 기반의 워터마크 엔진으로 구현하였고, Flask 백엔드와 React/Next.js 프론트엔드를 결합한 웹 플랫폼 형태로 통합하여 사용자가 웹 환경에서 직접 이미지를 업로드하고 워터마크를 삽입·추출할 수 있도록 하였다.

본 연구의 성과는 기술적인 완성도뿐 아니라 저작권 보호의 실질적 가능성을 보여주었다는 점에 있다. 플랫폼 내에서 삽입된 워터마크는 외부로 유출된 이미지에서도 복호화가 가능하며, 원본 제작자 식별 및 추적이 가능하다. 또한 메타데이터에 삽입된 워터마크 정보는 암호화되어 저장되기 때문에, 외부 접근으로부터 안전하게 보호된다. 이를 통해 본 연구는 이미지 저작물의 무단 사용을 억제하고, 창작자 중심의 콘텐츠 보호 기술 구현 가능성을 제시하였다.

### 향후 발전 방향

향후 연구에서는 본 기술을 동영상 콘텐츠에 적용하는 방향으로 확장할 수 있다. 이미지 기반 DCT 워터마킹을 프레임 단위로 적용하여 동영상 전체에 분산 삽입함으로써, 불법 복제나 유통 시 추적 가능성을 확보할 수 있다. 또한, 실시간 스트리밍 환경에서도 워터마크 삽입과 추출이 가능하도록 알고리즘을 최적화하여 성능을 개선하고, 압축, 해상도 변환, 필터 적용 등 다양한 동영상 변형에 강건한 워터마킹 기술을 개발할 수 있다. 이를 통해 동영상 기반 디지털 저작물 보호 및 사용자 추적 시스템으로의 확장이 가능하며, 향후 멀티미디어 콘텐츠 보안성 강화에 기여할 수 있다.

## 5. 별첨

### 5.1 소개자료(품보드)



### 5.2 프로젝트 소개

Github 주소 : <https://github.com/limgunny/ddw-stegano-frontend>

웹서비스 주소 : <https://ddw-stegano-frontend-qyrz.vercel.app/>

시연 영상 주소 : [https://youtu.be/IOAsIM\\_TnXE](https://youtu.be/IOAsIM_TnXE)

### 5.3 팀 소개

팀장 : 정민재

Github 주소 : <https://github.com/wm0416>

팀원 : 임건희

Github 주소 : <https://github.com/limgunny>

팀원 : 유혁준

Github 주소 : <https://github.com/Heockjun>

### 5.4 발표자료 (발표ppt)