**무서핑 서비스 PRD (Product Requirements Document)**

**프로젝트 개요**

**무서핑**은 AI를 활용해 유머 또는 공포 테마의 **귀신 캐릭터 콘텐츠**를 생성하고 공유하는 웹 서비스입니다. 현재 React/Next.js 기반 프론트엔드가 Vercel에 배포된 상태로, 사용자들이 간단히 AI가 만들어낸 이미지를 감상할 수 있습니다. 다음 단계에서는 **익명 커뮤니티 게시판** 기능과 **게임적 요소**, 그리고 **Google AdSense 광고 수익화** 기능을 추가하여 사용자 참여를 높이고 수익 모델을 확립하고자 합니다 . 이를 통해 무서핑을 단순한 AI 콘텐츠 감상에서 나아가, **직접 콘텐츠를 생성하고 공유**하며 **재미있는 상호작용**을 즐길 수 있는 커뮤니티로 발전시키는 것이 목표입니다.

**서비스 현황 및 확장 방향**:

* 현재 프론트엔드는 [무서핑 데모 사이트](https://v0-korean-react-example.vercel.app/)에 배포되어 있습니다 (GitHub 저장소: kyho2k/Nosurfing).
* 기존에는 **로그인 없이** AI가 생성한 이미지만 보는 기능이 있었지만, 앞으로는 **익명 게시판**을 도입하여 사용자가 **직접 AI 콘텐츠를 생성**하고 게시할 수 있게 합니다.
* 또한 생성된 귀신 캐릭터를 소재로 한 **간단한 게임적 요소**(예: 뽁뽁이 터뜨리기 미니게임)와 **“강화”** 시스템을 도입하여 재미를 높입니다.
* 수익 창출을 위해 **Google AdSense 광고**를 웹사이트 적절한 위치에 통합하고, 서비스 지속 운영에 필요한 재원을 마련합니다. 광고 게재 시에도 **콘텐츠 품질과 정책 준수**를 철저히 관리하여 AdSense **승인 및 유지**에 문제가 없도록 할 것입니다 .

**유저 플로우**

1. **메인 페이지 방문**: 사용자는 무서핑 메인 페이지에 접속합니다. 상단에는 **“최신 글”** 섹션과 **“월간 베스트”** 섹션이 배치되어, 새로운 콘텐츠와 한달 간 인기 있었던 콘텐츠를 한 눈에 볼 수 있습니다. 예를 들어, 최신 글 영역에는 실시간으로 갱신되는 새 게시물이 카드 형태로 표시되고, 월간 베스트 영역에는 좋아요(👻)를 가장 많이 받은 상위 게시물이 노출됩니다.
2. **콘텐츠 둘러보기**: 사용자는 스크롤하여 다양한 AI 생성 귀신/캐릭터 이미지를 감상합니다. 게시물에는 생성된 **이미지**와 간략한 **설명 텍스트**(AI가 만들어낸 유머 혹은 오싹한 한 줄 소개)가 포함됩니다. 각 게시물 카드에는 좋아요 수(🩸 또는 🦴 아이콘으로 표시)와 생성 시각 등이 표시되어 있어, 인기와 신선도를 쉽게 파악할 수 있습니다.
3. **존재 만들기 (AI 콘텐츠 생성)**: 사용자는 하단의 **“존재 만들기”** 버튼을 눌러 자신만의 AI 귀신 캐릭터를 생성합니다. 별도의 회원가입 없이 바로 생성 페이지로 이동하며, 간단한 프롬프트 입력이나 테마 선택으로 AI 이미지/텍스트 생성을 요청할 수 있습니다. 예를 들어 “웃는 얼굴의 유령” 같은 키워드를 입력하면, OpenAI의 DALL·E API를 통해 해당 키워드에 맞는 이미지를 생성하고, GPT-3.5를 이용해 짧은 소개 글을 곁들입니다. *(초기 버전에서는 OpenAI API를 사용하고, 이후 Google의 Imagen/Gemini API로 대체하기 쉽도록 구조화할 계획입니다.)*
4. **AI 생성 결과 확인**: 수 초 내에 AI가 생성한 귀신 캐릭터 이미지와 소개 텍스트가 화면에 나타납니다. 사용자는 **업로드 전에 미리보기**를 통해 결과물을 확인하고 만족도를 평가합니다. 프롬프트에 오탈자가 있거나 원치 않는 결과가 나왔을 경우 다시 수정하거나 재생성할 수 있습니다. (OpenAI DALL·E API를 서버 사이드에서 호출하여 이미지를 생성한 뒤, 결과 이미지를 클라우드 스토리지에 저장하고 URL을 불러와 표시합니다. 이는 API 키 노출을 막고 생성된 이미지를 게시판에 **영구적으로 호스팅**하기 위함입니다.)
5. **강화 시스템 (재생성)**: 생성된 귀신 캐릭터가 마음에 들긴 하지만 다른 버전도 보고 싶다면, \*\*“강화하시겠습니까?”\*\*라는 안내와 함께 **강화 버튼**을 누를 수 있습니다. 이 기능은 일종의 **재생성**으로, 현재 결과물을 기반으로 AI에게 추가 이미지를 만들도록 요청합니다. 예를 들어 포켓몬 게임의 진화처럼, 현재 귀신 캐릭터를 한 단계 “강화”한다는 컨셉으로 UI를 구성하고, 내부적으로는 OpenAI API를 다시 호출하여 **새로운 변형 이미지를 생성**합니다. 사용자는 강화 결과도 확인하여 원본과 비교해볼 수 있습니다. 강화는 1~2회 정도 가능하도록 제한하여, **AI 결과에 대한 무분별한 재시도 남용을 방지**합니다.
6. **게시 및 피드 보기**: 생성 결과에 만족하면, **“게시하기”** 버튼을 눌러 해당 콘텐츠를 공개 게시판에 업로드합니다. 이때 사용자는 여전히 익명 상태이며, 별도의 작성자 이름 없이 게시물이 등록됩니다. 게시가 완료되면 **“피드 보기”** 페이지로 이동하여 방금 올린 게시물을 포함한 전체 피드를 확인합니다. 피드 페이지에서는 양 옆 사이드 영역에 **작은 게임 요소**가 제공됩니다. 화면 양 측면에 포장용 뽁뽁이 비닐 이미지가 배치되고, 사용자가 거품을 클릭할 때마다 “펑” 소리와 함께 터지며 **랜덤한 점수**가 표시됩니다 🎮. 이 **뽁뽁이 터뜨리기 미니게임**은 서비스 컨셉에 맞춘 **소소한 재미 요소**로, 콘텐츠 감상 사이에 유저들의 지루함을 덜고자 합니다. (점수는 세션에 임시 저장되고 리셋되며, 순전히 재미를 위한 기능입니다.)
7. **콘텐츠 상호작용**: 피드에서 다른 사용자의 게시물을 볼 수도 있습니다. 마음에 드는 게시물이 있으면 **좋아요** 버튼(예: 빨간 피방울 모양의 아이콘)을 눌러 반응할 수 있습니다. 무서핑에서는 좋아요 수를 통해 **인기순 정렬**이나 **월간 베스트 선정**이 이뤄지므로, 사용자의 호응이 콘텐츠 평가에 중요하게 반영됩니다. 좋아요는 로그인 없이도 누를 수 있으나, 동일한 사용자(브라우저)에게는 중복 누르기가 제한됩니다. (쿠키나 로컬스토리지를 사용하여 이미 좋아요를 누른 게시물에는 다시 누를 수 없도록 처리합니다.) 만약 사용자가 브라우저 시크릿 모드로 반복 시도하거나 IP를 바꿔서 좋아요 조작을 시도할 경우를 대비해, **간단한 레이트 리미팅**도 적용할 예정입니다 .
8. **월간 베스트 열람**: 사용자는 메뉴에서 \*\*“월간 베스트”\*\*를 눌러 해당 월에 가장 인기있었던 상위 게시물 목록을 볼 수 있습니다. 이는 좋아요 수를 집계하여 정렬한 랭킹 페이지로, 월별로 초기화됩니다. 인기 콘텐츠 모아보기를 통해 새로운 방문자도 서비스의 흥미로운 콘텐츠를 접할 수 있고, 기존 사용자에게는 **콘텐츠 생산 동기**를 부여합니다 (예: “베스트 도전!”과 같은 유도 문구 표시).
9. **지속 참여 및 재방문**: 익명 게시판 특성상 **별도의 프로필 관리나 친구 추가 기능은 없지만**, 사용자들은 자유롭게 콘텐츠를 생산하고 소비하는 과정을 통해 커뮤니티에 참여하게 됩니다. 운영 측면에서는 **건강한 커뮤니티 유도**를 위해, 도배나 욕설이 발견되면 자동 필터링 및 관리자 조치를 취하고 (자세한 내용은 **콘텐츠 모더레이션** 섹션 참조), 신고 기능을 추후 도입할 계획입니다.
10. **광고 노출 및 수익화**: 페이지 이용 과정 중 곳곳에 **Google AdSense 광고**가 반응형 배너 형태로 표시됩니다. 처음 방문 시 헤더 배너로 광고를 노출하고, 피드 목록 중간중간에 **네이티브 광고** 스타일의 배너를 끼워 넣습니다. 또한 사이드바에는 스크롤을 따라오는 고정형 광고 슬롯이 있어 **지속 노출**됩니다. 긴 게시물 본문 중간에도 자동으로 광고가 들어가 수익을 극대화할 예정입니다 . 광고는 콘텐츠와 명확히 구분되도록 디자인하여 **실수로 클릭**되는 일이 없게 합니다 (예: “스폰서 광고” 라벨 표시 및 콘텐츠와 150px 이상 이격) . 사용자는 광고로 인해 방해받지 않으면서도 자연스럽게 서비스를 이용하도록 UX를 최적화합니다.

**핵심 기능**

**1. 익명 게시판 및 게시물 관리**

* **익명 게시글 작성**: 회원가입이나 로그인 **없이** 누구나 게시글을 작성할 수 있습니다. 게시물에는 **제목**, **AI가 생성한 이미지**, **AI 생성 설명 텍스트**로 구성되며, 작성자의 신원이나 별명은 표시되지 않습니다. 모든 사용자가 평등한 익명으로 참여하여 부담 없이 콘텐츠를 올릴 수 있는 환경을 조성합니다 .
* **게시글 수정/삭제**: 기본적으로 게시물이 등록되면 수정은 불가하며, 삭제는 관리자만 처리합니다. 다만 \*\*게시 직후 일정 시간(예: 3분 이내)\*\*에는 작성자 기기에서 수정 또는 삭제 요청을 할 수 있는 기능을 고려합니다. 이는 오탈자 수정이나 잘못 올린 이미지 교체 등을 위한 최소한의 편의 장치입니다. 구현 방법으로는 게시 성공 시 클라이언트에 **일회성 편집 토큰**을 발급하고, 해당 토큰을 가진 경우에만 일정 시간 내 수정/삭제를 허용하는 방식을 사용할 수 있습니다. (전통적인 익명 게시판의 **게시물 비밀번호** 입력 방식도 대안으로 검토되나, 사용자 부담을 줄이기 위해 자동 토큰 발급 방식을 우선 고려합니다.) 이러한 토큰 기반 접근은 쿠키 삭제 등으로 권한을 잃을 수 있는 문제를 줄이고, **익명성을 유지하면서도 본인 게시물 관리 권한**을 확보하기 위한 방안입니다 .
* **최신순 / 인기순 정렬**: 사용자는 게시판을 **최신순**으로 보거나 **인기순**(좋아요 많은 순)으로 볼 수 있습니다. 기본은 최신 게시물이 위에 나오지만, 토글을 통해 인기순으로 전환하면 많은 호응을 받은 글이 상단에 나타납니다. 이를 위해 데이터베이스에서 게시글 조회 시 created\_at 필드에 인덱스를 두고 DESC 정렬하거나, like\_count 필드에 DESC 인덱스를 적용하여 효율적으로 쿼리합니다 . 게시글 수가 증가하더라도 인덱싱을 통해 **응답 속도**를 높이고, 큰 비용 없이 정렬 기능을 제공할 수 있습니다.
* **월간 베스트 게시물**: 매월 1일 00시를 기준으로 지난달의 좋아요 수 상위 N개 게시물을 **월간 베스트**로 지정합니다. 월간 베스트는 별도 페이지에서 순위 리스트로 제공되며, 각 게시물의 썸네일, 제목, 좋아요 수, 게시일 등이 표시됩니다. 이 기능은 커뮤니티 참여를 장려하고, 우수 콘텐츠를 한눈에 보여줌으로써 신규 사용자가 서비스의 **대표 콘텐츠**를 접하는 통로가 됩니다.
* **좋아요 (반응) 기능**: 게시물에 대한 **좋아요(공감)** 기능을 제공합니다. 사용자는 게시물당 한 번만 좋아요를 누를 수 있으며, 이미 누른 경우 버튼이 활성화된 상태로 표시되어 중복 클릭이 방지됩니다. 좋아요 수는 실시간으로 증가하며, 이는 인기순 정렬과 베스트 선정의 기초 데이터가 됩니다. 무서핑의 테마에 맞춰 **하트 모양 대신 피 방울**(🩸)이나 **해골/뼈다귀**(💀) 아이콘으로 커스텀 디자인할 예정이며, 이 아이콘은 추후 테마에 따라 쉽게 교체 가능하도록 **구조화**합니다 (ex: 설정 파일이나 DB에서 아이콘 코드만 변경하면 UI 전체에 적용). 좋아요 데이터 처리는 **경량화**를 위해 아래 기술 스택 섹션에서 설명하는 서버리스 DB 및 캐시를 활용합니다.
* **댓글 미제공**: 커뮤니티의 복잡도와 모니터링 부담을 고려하여 **댓글 기능은 제공하지 않습니다**. 이는 익명 환경에서 발생할 수 있는 스팸, 욕설, 분쟁을 줄이고 서비스 초기에 **콘텐츠 생산과 소비에 집중**하기 위한 결정입니다. 사용자 피드백이나 토론이 필요할 경우를 위해 **별도의 소통 채널**(예: Discord 커뮤니티나 설문조사 링크)을 운영할 수 있습니다. 댓글이 없는 대신, 추후 필요 시 **대체 기능**으로 게시물 신고 또는 리액션(emoticon 반응) 정도를 검토할 수 있습니다.

**2. AI 콘텐츠 생성 기능**

* **이미지 생성 (OpenAI DALL·E API)**: 사용자가 입력한 키워드나 간단한 문장을 바탕으로 **AI 이미지 생성**을 수행합니다. OpenAI의 DALL·E API를 사용하여 귀신 캐릭터나 공포스러운 장면 이미지를 생성하고, 결과물을 사용자에게 보여줍니다. 이미지 생성은 **백엔드 서버리스 함수**에서 진행하며, 완성된 이미지는 일시적으로 **클라우드 스토리지**에 저장됩니다. 그런 다음 게시글로 등록되거나 사용자에게 미리보기로 제시됩니다. *초기에는 DALL·E를 이용하지만, 구조를 유연하게 만들어 추후 Google의 Imagen API로 쉽게 전환할 수 있도록 할 것입니다.* 예를 들어, 이미지 생성 모듈에 인터페이스를 정의하고, 현재는 OpenAI 구현체를 사용하되, 차후 Google API 키와 요청 포맷으로 구현체만 교체하면 되도록 추상화합니다.
* **텍스트 생성 (OpenAI GPT-3.5)**: 이미지에 어울리는 **짧은 설명 문구**나 **스토리 한 줄**을 AI가 생성하여 제공합니다. 예를 들어 “이 귀신은 밤마다 웃는다”와 같은 유머러스하거나 오싹한 한마디를 OpenAI GPT를 통해 생성합니다. 이를 통해 이미지에 맥락과 재미를 부여합니다. 이 또한 서버 측에서 API를 호출하여 처리하며, 필요 시 사용자 프롬프트(예: 원하는 문체)를 입력받을 수도 있습니다. 장기적으로 OpenAI 대신 **Google Gemini** 등 대체 모델로 교체하기 쉽도록 API 호출부를 모듈화합니다.
* **AI 생성 결과 필터링**: AI가 생성한 콘텐츠가 서비스 정책이나 광고 정책에 위배되지 않도록 **필터링 단계**를 거칩니다. 예를 들어 DALL·E API 자체로도 콘텐츠 필터링이 있지만, 추가로 **OpenAI 콘텐츠 검열 API**나 자체 설정을 통해 **NSFW나 과도한 폭력** 이미지를 걸러냅니다. 텍스트의 경우 욕설이나 혐오 표현이 섞이지 않도록 **금칙어 필터**를 적용합니다. 이중 방어로 안전한 결과만 사용자에게 전달하여, 부적절한 AI 출력으로 인한 문제를 예방합니다.
* **이미지 미리보기 및 편집**: 사용자가 새 게시글을 작성할 때, 생성된 이미지를 **미리보기**로 확인시켜 줍니다. (React의 상태 관리로 이미지 URL을 임시로 보여주고, 필요하면 다시 생성하도록 유도). 또한 원치 않는 결과일 경우 **다시 생성**(재시도)하거나 앞서 설명한 **“강화”** 기능을 통해 변형된 결과를 얻을 수 있습니다. 이러한 UI/UX는 사용자가 만족스러운 콘텐츠를 얻을 때까지 **피드백 루프**를 제공하고, 동시에 AI 생성의 놀이적 요소를 경험하게 합니다.

**3. 게임적 요소 (미니게임 & 강화 시스템)**

* **뽁뽁이 터뜨리기 미니게임**: 사용자 참여도를 높이기 위한 **작은 재미 요소**로서, 피드 화면 양옆에 뽁뽁이 비닐 이미지를 배치합니다. 각 뽁뽁이는 마우스로 클릭하거나 터치하면 “뽁!” 하는 소리와 함께 터지며, 화면에 +n점 숫자가 뜹니다. 점수는 랜덤 (예: 1~5점)으로 부여되어 사용자에게 **가벼운 중독성**을 유발합니다. 이 점수에는 실질적 보상이나 저장은 없지만, 나중에 **“내 점수”** 형태로 세션 동안 획득한 총점을 표시하여 성취감을 줄 수 있습니다. 기술적으로 React 컴포넌트로 구현하여, 클릭 시 state를 변경하고 애니메이션을 보여주는 방식으로 작동합니다. 이러한 게임 요소는 커뮤니티의 \*\*밈(meme)\*\*이 될 수도 있고, 공포 컨셉의 긴장감을 해소하는 **코믹 relief** 역할도 기대됩니다.
* **강화 (레벨업) 시스템**: 앞서 **유저 플로우 5번**에서 설명한 “강화하시겠습니까?” 기능은, AI가 생성한 결과물을 **한 단계 업그레이드**한다는 게임적 상상력을 불어넣은 것입니다. 예를 들어, 1단계 귀신 캐릭터를 2단계로 강화하면 더 강력하고 무서운 모습으로 변신하는 느낌을 줍니다. 구현은 실제로는 새로운 이미지를 재생성하는 것이지만, UI 상에서는 **번개 효과**나 **변이 애니메이션** 등을 넣어 재미를 줍니다. 사용자는 강화 전후 이미지를 비교하면서 마치 게임에서 캐릭터를 육성하듯 즐길 수 있습니다. 이 기능은 콘텐츠 소비에 **능동적 참여**를 유발하고, 결과적으로 더 많은 AI 요청과 게시글 생성을 촉진할 것으로 기대됩니다.
* **포인트 및 배지 (추후 고려)**: 미니게임 점수나 좋아요 활동에 따라 포인트를 부여하고, 특정 포인트 이상인 사용자에게 **배지**나 **칭호**를 제공하는 방안도 고려하고 있습니다. 다만 익명 서비스의 특성상 현재는 개인별 포인트를 누적하여 저장하지는 않으며, 추후 **소셜 로그인 도입 시**에 게이미피케이션 요소로 확장할 여지를 남겨둡니다.

**4. 광고 수익화 및 정책 준수**

* **광고 배치 전략**: Google AdSense의 **반응형 디스플레이 광고**를 활용하여, 화면 크기별 최적화된 배너를 보여줍니다. 주요 배치 위치는 **헤더 상단**(가로 배너), **사이드바**(고정 스카이스크래퍼 배너), **피드 목록** (몇 개 게시물마다 네이티브 스타일 광고), 그리고 **게시물 상세 화면 중간**입니다. 이렇게 다양한 위치에 광고를 분산 배치함으로써 **시각적 피로**를 줄이고, 사용자 스크롤 동선에 자연스럽게 녹아들게 합니다 . 특히 **게시물 본문 중간** 광고는 긴 공포 이야기 텍스트가 있을 경우 효과적이며, **사이드바 고정 광고**는 사용자가 스크롤해도 항상 노출되어 **높은 클릭률**을 기대할 수 있습니다. 단, 광고와 콘텐츠는 명확히 구분하여, **실수 클릭** 유도나 오인을 방지합니다 (예: 광고 배경색이나 테두리로 구분, “광고” 레이블 표기 등).
* **AdSense 정책 준수**: 광고 게재에 앞서 Google AdSense **프로그램 정책**을 면밀히 준수합니다. 무서핑은 사용자 생성 콘텐츠(UGC)를 다루므로, **모든 페이지의 콘텐츠가 정책 위반이 없어야** 합니다 . 이를 위해 **충격적 콘텐츠** (지나친 폭력/혐오/외설 등)를 생성하지 않도록 AI 프롬프트를 제한하고, 부적절한 게시물이 업로드되면 즉시 삭제 및 차단 조치를 취합니다 . 예를 들어 **과도한 잔인함**이나 **18금 성적 요소**가 포함된 이미지는 생성 단계에서 거르고, 사용자 입력 프롬프트에도 금칙어 필터를 적용합니다. 또한 **광고 코드 삽입** 시 AdSense가 제공한 스크립트를 **변경 없이** 그대로 사용하고, 허가 없이 코드를 수정하거나 특수한 환경(예: 팝업, 앱 웹뷰 등)에 넣지 않습니다 .
* **AdSense 승인 준비**: 광고를 게재하기 위해 우선 AdSense 승인을 받아야 하므로, **서비스 콘텐츠의 품질**을 높이는 데 집중합니다. 출시 시점에 최소 5~10개의 **고유한 게시물 콘텐츠**를 생성하여 사이트에 채워둘 예정이며, 이는 AdSense 심사 담당자가 사이트를 방문했을 때 “콘텐츠가 풍부하고 유용한지”를 판단하는 근거가 됩니다 . 또한 **필수 페이지** (개인정보처리방침, 문의처 등)를 푸터에 게시하여 신뢰도를 높입니다. 운영자 정보 및 연락 수단도 투명하게 공개합니다. 이러한 준비를 통해 **최초 승인**을 획득하고, 이후에는 정책 위반이 발생하지 않도록 **지속 관리**합니다.
* **수익 극대화와 사용자 경험 균형**: 무서핑 사용자들은 **공포/유머 콘텐츠 팬층**이 많을 것으로 예상되며, 이들은 과도한 광고에 민감할 수 있습니다. **팝업 광고나 소리나는 광고는 지양**하고, **콘텐츠 흐름을 해치지 않는 수준**에서만 광고를 유지합니다 . 만약 사용자가 광고 차단기(AdBlock)을 사용할 경우를 대비해, 중요한 공지나 컨텐츠는 광고 없이도 볼 수 있게 하고, 차단 시 노출되는 빈 영역을 최소화합니다. 필요하면 “광고가 이 사이트 유지에 도움이 됩니다”와 같은 안내를 통해 사용자의 이해를 구할 수 있습니다. *추후 트래픽이 증가하고 수익이 안정되면, 광고 노출 빈도를 조절하거나 프리미엄 구독(광고 제거 옵션) 같은 모델도 검토할 계획입니다.*

**5. 콘텐츠 모더레이션 및 안전성**

* **욕설/비속어 필터링**: **익명** 환경에서는 사용자가 공격적인 언어를 쓸 가능성이 높으므로, **입력 단계에서 욕설 필터**를 적용합니다. 게시물 제목이나 (향후 댓글이 생긴다면 댓글 내용)에 금칙어가 포함되면 경고를 표시하고 제출을 막습니다. 이를 위해 오픈소스 라이브러리 **badwords-ko** (한국어)와 **bad-words** (영어)를 사용하여 1차 필터링을 합니다 . 예를 들어 *“시@발”*, *“개새##”* 등의 변형 표현도 정규식 패턴으로 탐지합니다. 다만, 사전 기반 필터는 한계가 있어서, 위장된 욕설(자음/모음 분리, 특수문자 삽입 등)은 놓칠 수 있습니다 . 이를 보완하기 위해 장기적으로 **딥러닝 기반 욕설 탐지 모델** 도입을 고려합니다. 국내 게임업계 사례에서 **CNN/LSTM 기반 욕설 탐지**를 적용한 결과 정확도가 90%까지 향상된 바 있으며 , 이러한 기술을 활용하면 변형 욕설도 높은 확률로 잡아낼 수 있습니다. 예를 들어 넥슨이 도입한 욕설 탐지기는 금칙어 방식 56% 정확도를 딥러닝으로 88~90%까지 끌어올렸습니다 . 우리 서비스도 데이터 축적 시 **자모 단위 CNN** 등 모델을 학습시켜 고도화할 수 있을 것입니다.
* **자동 차단 및 로그**: 필터에 걸린 부적절 콘텐츠는 **즉시 차단**되며, 해당 내용과 시각이 **로그로 기록**됩니다. 관리자는 로그를 주기적으로 확인하여 반복적으로 시도하는 IP 등을 파악하고 추가 조치를 취합니다. 초기 단계에서는 별도 관리자 화면 없이, 서버 콘솔 로그나 Slack 연동 알림으로 모니터링할 계획입니다. 또한 OpenAI의 **Moderation API**를 활용하여 AI 생성 텍스트의 유해성 점수를 확인하고, 임계치 이상일 경우 해당 결과를 폐기 처리합니다.
* **관리자 수동 검토**: 서비스가 성장하여 UGC가 폭증하면 모든 문제를 자동으로 잡아내기 어려울 수 있습니다. 따라서 **신고 기능**을 마련하여, 사용자들이 부적절한 게시물을 신고할 수 있게 합니다 . 신고가 접수된 게시물은 **임시 숨김 처리**되고 관리자가 검토 후 삭제 또는 복원합니다. 관리 도구로는 Supabase의 Dashboard를 활용하거나, 별도의 **관리자 전용 웹페이지**(숨겨진 경로에 로그인 보호)에서 게시물/유저 IP 목록을 조회하고 삭제하는 기능을 구축합니다. 정책 위반이 심각한 경우 (예: 불법정보 게시 등) 해당 IP를 **차단**하고, 필요한 경우 수사기관 협조를 위해 로그를 보관합니다. 이는 AdSense 정책 준수 측면에서도 매우 중요한데, Google은 광고 게재 사이트의 UGC에 대해서도 **게시자가 책임**질 것을 요구하기 때문입니다 . 즉, 유해 콘텐츠가 잠깐이라도 노출되지 않도록 **사전/사후 모니터링 체계**를 갖추는 것이 광고 수익 안정화와 커뮤니티 건강성 모두에 필수적입니다 .
* **개인정보 보호 및 법적 준수**: 무서핑은 익명성을 존중하여 **개인정보를 거의 수집하지 않는 것**이 원칙입니다. 아이디, 이메일 등의 회원정보는 받지 않고, 게시글에 포함된 이미지나 텍스트에도 개인 신상이 드러나는 내용은 AI가 생성하는 한 거의 없습니다. 다만 **IP 주소** 정도는 비정상 행위 대응을 위해 서버 로그에 남길 수 있는데, 이는 해시 처리 등 **비식별화**하여 저장하고 일정 기간 후 폐기합니다. 또한 쿠키에 익명 식별자나 세션 토큰을 저장하므로 **쿠키 정책 배너**를 통해 사용자 동의를 받습니다. 서비스 운영 시 **청소년 보호법**도 준수하여, 19세 미만이 이용하기 부적절한 콘텐츠는 애초에 생성하지 않도록 차단합니다. (예: 선정성 높은 콘텐츠 금지) 만약 연령 제한이 필요한 경우, 성인 인증 절차(아이핀, 휴대폰 인증 등)는 서비스 특성상 도입하지 않을 계획이므로, **해당 콘텐츠 자체를 서비스에 허용하지 않는 방향**으로 합니다.

**기술 스택**

**프론트엔드 (React / Next.js 기반)**

* **Next.js 14 (React)**: SEO와 초기 로딩 최적화를 위해 **SSR/SSG**가 가능한 Next.js를 사용합니다. Vercel에 최적화되어 배포 및 스케일이 용이하며, 이미지 최적화나 API Routes 등 유용한 기능을 활용할 수 있습니다. 페이지 간 이동은 SPA처럼 부드럽게 처리하고, 게시판 목록은 정적 생성(ISR)하여 빈번한 조회에 대응합니다 .
* **UI 라이브러리**: **Tailwind CSS**를 채택하여 신속하게 스타일링하고 반응형 디자인을 구현합니다. Utility-first 클래스 체계로 유지보수가 편하며, 다크모드 대응에도 용이합니다. 일부 컴포넌트에는 **Chakra UI**도 병행 사용하여 접근성 높은 모달, 토스트 등을 구현할 예정입니다.
* **State 관리**: 필요한 경우 **React Query(SWR)** 등을 사용하여 서버 데이터 캐싱과 요청 상태 관리(로딩, 에러)를 쉽게 처리합니다. 예를 들어 좋아요 버튼 클릭 시 Optimistic Update로 즉시 UI에 반영하고, 백엔드 동기화는 비동기로 처리하는 등의 패턴을 적용해 **반응 속도**를 높입니다.
* **이미지 처리**: Next.js 내장 **Image 컴포넌트**를 활용하고, **이미지 CDN** 기능으로 Lazy Loading, 사이즈 최적화를 자동화합니다. 사용자가 업로드(생성)한 이미지는 .webp 등의 포맷으로 변환되어 전송되며, Cloudflare와 같은 CDN을 통해 전세계 지역에서도 빠르게 로드됩니다【44†】. 이는 이미지 파일 크기를 크게 줄이고 사용자 데이터 사용량을 절약해줍니다.
* **기타**: 에러 모니터링을 위해 **Sentry**를 프론트에 설치하여 UI에서 발생하는 예외를 추적합니다. 사용자 행동 분석은 \*\*구글 애널리틱스 (GA4)\*\*를 최소한으로 삽입하여 (익명 IP 설정) 페이지뷰, 체류시간 등을 확인합니다. 다만, 개인정보 이슈로 **과도한 트래킹은 지양**합니다.

**백엔드 및 인프라**

* **서버리스 함수 (Vercel)**: Next.js의 API Routes 혹은 Vercel Serverless Functions를 이용해 백엔드 기능을 구현합니다. 사용자 요청(게시물 작성, 좋아요 클릭 등)에 대응하여 **데이터베이스 CRUD**를 수행하거나 **외부 API 호출**(OpenAI 등)을 처리합니다. 서버리스 환경은 무상태 호출이므로, 데이터 저장은 DB에 의존하고 함수 내에서는 **비즈니스 로직**만 수행합니다. 예를 들어 POST /api/createPost 함수는 AI 생성 결과(이미지 URL, 텍스트)를 받아 DB에 신규 게시물을 기록하고, POST /api/like 함수는 특정 게시물의 좋아요 이벤트를 처리합니다.
  + 서버리스 함수의 **콜드 스타트**에 대비하여, 성능이 중요한 부분(예: 메인 피드 조회)은 ISR(사전 생성)로 커버하고, 나머지 동적 호출은 지역(서울 리전) Edge로 배포합니다. Vercel은 전세계 서버에 코드를 배포하므로 한국 사용자도 충분히 빠른 응답을 얻을 수 있습니다.
  + Node.js 환경에서 실행되며, 필요한 경우 **Node-fetch**나 Axios로 외부 API 통신을 합니다.
* **데이터베이스 (Supabase Postgres)**: **Supabase**를 메인 DB로 사용하여 게시물, 좋아요 등 **관계형 데이터**를 관리합니다. Supabase는 PostgreSQL 기반이므로 **SQL 쿼리** 최적화 및 **인덱스** 설계를 적극 활용합니다. 주요 테이블:
  + Posts 테이블: 필드 – id, image\_url, text\_content, created\_at, like\_count 등. 인덱스 – created\_at(DESC), like\_count(DESC) .
  + Likes 테이블: 필드 – post\_id, user\_token, created\_at. 좋아요 이벤트 기록용 테이블로, Posts.like\_count를 정규화하지 않고 별도 저장. (궁극적 일관성 모델: 일정 주기마다 Likes 집계하여 Posts.like\_count 업데이트) .
  + (댓글 기능은 없으므로 댓글 테이블은 없음)
  + Supabase의 \*\*Row Level Security (RLS)\*\*는 익명 공개 서비스이므로 일단 비활성화하거나, 최소한의 검증(예: user\_token 매칭 시에만 삭제 허용 등)에만 사용합니다. 대부분의 데이터는 공개 조회 가능하므로 RLS보다 **API 단**에서 권한을 제어합니다.
  + **확장성**: Supabase 무료 플랜에서 시작하며, 500MB~1GB까지 데이터를 수용할 수 있습니다 . 추후 사용자 증가로 데이터가 많아지면 **유료 플랜 업그레이드** 또는 **별도 PostgreSQL 호스팅**으로 이관도 고려합니다. 또한 읽기 트래픽 분산을 위해 Supabase의 **리드 레플리카**나 **캐싱 계층**을 도입할 수 있습니다.
* **캐시 / 실시간 업데이트**: 실시간 피드 업데이트를 위해, Supabase의 **Realtime 기능**을 사용하여 게시글 생성이나 좋아요 이벤트가 발생하면 클라이언트에 **변화**를 알려줄 수 있습니다. 예를 들어 새로운 게시물이 DB에 추가되면 웹소켓을 통해 프론트에 push하고, 프론트는 피드 상단에 “새로운 글 X개 – 새로고침” 배너를 표시합니다. 좋아요 숫자 역시 비슷하게 업데이트 가능합니다. 다만 초기 버전에서는 복잡도를 줄이기 위해 실시간보다는 사용자가 수동 새로고침하거나, Next.js SWR의 revalidate 간격을 두는 정도로 구현합니다.
  + **Upstash Redis (Vercel KV)**: 좋아요 카운트나 뽁뽁이 게임 스코어 같이 초고속 처리가 필요한 데이터는 **인메모리 캐시**를 사용합니다. Vercel KV(실제 Upstash Redis)는 서버리스 환경에서도 빠른 key-value 조회/설정을 지원하며, 글로벌 분산 캐시로서 지연이 낮습니다【1†】. 예를 들어 좋아요 버튼 클릭시 Redis에 post:{id}:likes 카운터를 즉시 증가시키고, 나중에 주기적으로 이 값을 DB와 동기화합니다. 이를 통해 동시 다발 요청이 와도 DB 부하를 줄이고 성능을 확보합니다. Upstash 무료 용량으로 시작하고, 필요 시 확장하거나 Redis 클러스터를 운영합니다.
* **파일 스토리지**: AI가 생성한 **이미지 파일**은 영구 보관이 필요합니다. 두 가지 옵션을 병행 검토합니다:
  1. **Vercel Blob Storage**: Vercel이 제공하는 키-값 형태 Blob 저장소로, 이미지 파일을 업로드하고 URL로 접근할 수 있습니다【42†】. Vercel의 글로벌 CDN을 통해 제공되므로 속도가 빠르고, 별도 세팅이 간편합니다. 클라이언트가 업로드를 요청하면 서버리스 함수가 OpenAI로부터 받은 이미지를 Blob Storage에 PUT하고, 반환된 URL을 DB에 저장합니다. Blob Storage 무료 제공량(예: 10GB 전송량) 내에서 운영하다가, 초과 시 비용을 지불합니다.
  2. **Supabase Storage**: Supabase에 내장된 객체 스토리지로, 게시물 ID 경로 등에 이미지를 저장할 수 있습니다【35†】【43†】. Supabase 클라이언트 SDK로 직접 브라우저에서 업로드도 가능하지만, API 시크릿 보호를 위해 서버를 경유합니다. CDN 연동은 기본 지원하므로, 전세계 캐싱도 문제없습니다.
  + 두 옵션 모두 **이미지 CDN** 역할을 하므로, 어느 쪽이든 사용자에게는 최적화된 이미지가 전달됩니다【44†】. 현재 Vercel에 이미 프론트가 있으므로 **Blob Storage**를 우선 사용하고, 트래픽 추이에 따라 Supabase Storage로 옮기거나 둘을 혼합 사용할 수 있습니다. 예산을 고려해 무료 용량 범위 안에서 효율적으로 활용할 계획입니다.

**AI & API 통합**

* **OpenAI API 통합**: OpenAI의 이미지 생성 API (DALL·E)와 텍스트 생성 API (GPT-3.5 Turbo)를 사용합니다. 비동기 호출을 위해 **Node.js용 OpenAI SDK**를 이용하거나, 간단한 fetch로 REST API 호출도 가능합니다. API 키는 Vercel 환경변수에 안전하게 저장하며, 서버리스 함수 내에서만 사용합니다 (클라이언트에는 노출 금지). 이미지 생성 요청 시 평균 5~8초 내외의 응답이 예상되므로, **생성 중 로딩 UI**(점멸하는 점들이나 progress bar)를 제공하여 사용자 이탈을 방지합니다. 응답으로 받은 이미지는 Base64 또는 URL로 오는데, 편의를 위해 **URL 형식**으로 받아 바로 저장/표시합니다. 텍스트 생성은 수백자를 넘지 않는 단문 위주이므로 신속히 처리됩니다.
* **모듈화 설계**: AI 모듈은 훗날 OpenAI 외에 **다른 AI 모델로 교체**할 수 있어야 합니다. Google에서 제공 예정인 **Gemini API**(차세대 멀티모달 모델)나 이미지 생성 모델 **Imagen**으로 변경하는 상황에 대비해, **Adapter 패턴**으로 구현합니다. 즉 ImageGenerator 인터페이스에 generateImage(prompt) 메서드를 정의하고, OpenAIImageGenerator와 GoogleImageGenerator 클래스 두 개를 만들어 둡니다. 현재는 OpenAI 클래스를 사용하지만, 환경변수 스위치나 최소 코드 변경으로 Google 클래스로 전환할 수 있게 합니다. 이렇게 하면 벤더 종속성을 줄이고, 새로운 AI 기술을 빠르게 도입할 수 있습니다.
* **API 사용량 및 비용**: OpenAI 이미지 생성은 1024x1024 기준 1회당 $0.02 비용이 발생하며, 텍스트 생성은 1,000토큰당 $0.002 정도입니다. 초기에는 이용자 규모가 작아 월 몇 달러 이내로 예상되지만, 향후 비용 모니터링을 통해 **생성 빈도 제한**(ex: 하루 5회 생성 제한)을 두거나 **프리미엄 과금 모델**도 고려합니다. Google 모델로 전환 시 비용 정책을 검토해 가장 경제적인 방안을 선택합니다.

**DevOps 및 CI/CD**

* **GitHub 연동 자동 배포**: GitHub 저장소에 코드를 푸시하면 Vercel이 자동으로 빌드하고 배포하는 **CI/CD 파이프라인**이 구축되어 있습니다. main 브랜치에 머지된 커밋은 곧바로 프로덕션 사이트에 반영되며, Pull Request마다 **Preview URL**이 생성되어 변경 사항을 미리 검증할 수 있습니다. 이를 통해 빠른 기능 출시와 실험이 가능하고, 버그 발견 시 롤백도 신속히 할 수 있습니다.
* **환경 변수 관리**: API 키 (OpenAI Key, Google AdSense client ID 등)와 DB 연결 정보는 Vercel 프로젝트의 Environment Variables에 저장되어 빌드시 주입됩니다. 따라서 코드에는 노출되지 않으며, 오픈소스 레포지토리에도 민감정보가 포함되지 않도록 주의합니다.
* **로깅 및 모니터링**: Vercel 제공 로그 콘솔 외에, 필요한 경우 **Supabase Logs**나 CloudWatch(만약 AWS Lambda로 전환 시)를 통해 서버리스 함수 호출 현황, 오류 스택 등을 모니터링합니다. 또한 클라이언트에서 **Sentry**로 오류 리포팅을 받고, 중요한 서버 에러는 Slack 웹훅 등으로 알림을 받을 예정입니다.
* **테스트**: 핵심 기능에 대해 **단위 테스트**와 **통합 테스트**를 점진적으로 추가합니다. 예를 들어 게시물 생성 로직, 좋아요 증가 로직에 대한 백엔드 테스트를 Jest로 구현하고, UI 동작은 Cypress로 간단한 시나리오 테스트를 작성합니다. 초기에는 필수 테스트 위주로 최소한을 두고, 서비스 안정화 단계에서 테스트 커버리지를 높여갑니다.

以上, **무서핑 서비스**의 기획 의도와 기능 사양, 그리고 기술 구현에 대한 상세 내용이었습니다. 이 PRD를 바탕으로 개발을 진행하며, 추후 생길 수 있는 이슈들(확장, 보안, 규제 등)은 **선제적으로 계획**하여 대응할 것입니다. 궁극적으로 무서핑이 **재미와 공포가 공존하는 독특한 커뮤니티**로 성장하고, 사용자들에게 새로운 경험을 제공함과 동시에 안정적인 수익 모델까지 갖출 수 있도록 지속 개선해 나갈 것입니다.