무서핑 서비스 익명 게시판 PRD (개정판)

1. 소개 및 목표

무서핑은 공포 이야기와 괴담을 좋아하는 사용자들이 익명으로 콘텐츠를 공유하고 소통할 수 있는 커뮤니티 서비스입니다. 특히 MBC 프로그램 **‘심야 괴담회’**를 즐기는 공포물 팬층을 주요 타겟으로 삼아, 사용자들이 자신의 무서운 경험담이나 창작 괴담을 자유롭게 올리고 서로 공감할 수 있는 공간을 제공합니다. 서비스의 핵심 목표는 **익명성 보장**을 통해 자유로운 참여를 이끌어내고, **콘텐츠 품질**과 **커뮤니티 안전**을 유지하면서 **광고 수익화**를 달성하는 것입니다.

- **주요 키워드:** 공포 테마 커뮤니티, **익명 게시판**, AI 이미지 생성, 사용자 생성 콘텐츠(UGC), Google AdSense 수익화, **안전한 커뮤니티**.

2. 주요 발견사항 및 변경 필요사항

최근 기술적 검토 결과, 기존 기획에는 **중대한 수정이 필요한 부분**이 발견되었습니다. 특히 **익명성 유지 기법과 광고 수익화 전략**에서 현실적인 한계와 위험이 확인되었으며, 이에 따라 서비스의 설계 방향을 조정해야 합니다. 핵심 발견사항과 필요 변경점은 다음과 같습니다.

⚠ 치명적 위험: 디바이스 핑거프린팅의 실효성 상실

초기 기획에서 익명 사용자 식별을 위해 제안된 **디바이스 핑거프린팅** 기법은 **사실상 무력화된 상태**입니다. 2024 년 기준 Safari 의 ITP(Intelligent Tracking Prevention)와 Chrome 의 Privacy Sandbox 등의 영향으로, 기존 브라우저 핑거프린트 기법 중 90% 이상이 차단되고 있습니다. 또한 사용자 동의 없이

이러한 정보를 수집하면 **한국 개인정보보호법** 위반 소지가 있습니다. 따라서 **디바이스 핑거프린팅 방식은 폐기**하고 대안을 마련해야 합니다.

대안: 개별 세션에 의존한 임시 익명 ID 부여(예: 세션 스토리지에 24 시간 유지되는 UUID) 등으로 전환하고, 필요 시 사용자가 자발적으로 닉네임을 지정하거나 소셜 로그인을 선택할 수 있게 유도합니다.

⚠ 높은 위험: AdSense 정책 준수 및 수익화 불확실성

익명 UGC 기반 서비스에서 **Google AdSense 승인과 정책 준수**를 달성하기는 매우 어렵습니다. Google 은 퍼블리셔(운영자)에게 사이트 내 모든 콘텐츠에 대한 책임을 요구하며, **사용자가 생성한 익명 콘텐츠도 예외가 아닙니다** . 공포/괴담 특성상 **충격적이거나 유해한 콘텐츠**가 등장할 가능성이 있고, 이는 AdSense 정책 위반으로 이어질 수 있습니다. 현재 추정되는 AdSense 승인 성공 확률은 **60~70% 수준**에 불과합니다. 승인 후에도 익명 댓글 등이 스팸과 정책 위반의 온상이 될 위험이 높아, **지속적인 모니터링과 필터링**이 필수적입니다.

대응: 철저한 콘텐츠 **모더레이션 시스템 구축**(상세 내용은 아래 5.절 참조)과 함께, AdSense 이외의 **대안 수익화 모델**도 동시에 고려합니다. 예를 들어 **네이버 애드포스트**나 **카카오 애드핏**과 같은 국내 광고 플랫폼은 승인 절차가 비교적 수월하여 초기 수익 창출에 도움이 될 수 있습니다. 또한 후원 시스템이나 프리미엄 콘텐츠 등 **직접 수익 모델**도 장기적으로 검토합니다.

⚠ 기타 주요 이슈 요약

- **토큰 기반 익명 인증의 보안:** 익명 게시물 수정/삭제를 위한 토큰 방식은 편리하지만, XSS 를 통한 토큰 탈취, 세션 하이재킹 등의 **보안 취약점**에 노출됩니다. 실제로 2023 년 **Storm-0558 해킹 사건**에서도 공격자가 탈취한 키로 인증 토큰을 위조하여 수많은 계정에 무단 접근한 사례가 있었습니다 . 다층적인 보안 대책이 필요합니다.
- **서버리스 아키텍처 한계:** Vercel 서버리스 환경은 콜드스타트 지연, 함수 실행 시간(최대 10~15 초) 및 동시 실행 제한(함수별 1,000 개) 등의 제약이 있습니다. 갑작스런 트래픽 증가 시 성능 저하와 비용 폭증 가능성을 면밀히 고려해야 합니다.

- **한국 규제 준수:** 공포 콘텐츠라 하더라도 **청소년 보호법** 및 방송통신심의 규정에 저촉될 소지가 있는지 확인해야 합니다. 과도한 폭력 묘사나 선정적 내용은 **"청소년 유해매체물"**로 분류될 수 있어, 성인인증이나 경고 문구 표시 등의 조치가 필요합니다.
- **개발 일정 재평가:** 초기 제시된 개발 기간 **11~16 주**는 주요 리스크 대응을 감안하면 지나치게 낙관적입니다. 보안 강화와 컴플라이언스 작업을 포함하여 **약 18~22 주**로 일정을 재산정해야 합니다 (섹션 9 에서 상세 제시).

以上의 사항들을 바탕으로, 아래에서는 서비스 기능과 기술 구현 방안을 **상세하고 현실적으로** 재정립하였습니다.

3. 서비스 기능 및 요구사항

3.1 주요 기능 개요

- **익명 게시판:** 로그인 없이 **익명으로 게시글 및 댓글 작성**이 가능합니다. 사용자는 별명이나 프로필 없이 자유롭게 글을 올릴 수 있으며, 각 게시물에는 조회수, 좋아요(공감) 수, 작성 시간 등이 표시됩니다.
- **게시글 관리:** 글 작성 시 시스템이 **편집/삭제 토큰**을 발급하여 쿠키 또는 로컬스토리지에 저장하고, 작성자는 이 토큰을 통해 일정 시간 내(예: 5 분 이내) 게시물 수정, 영구적으로 게시물 삭제를 할 수 있습니다. (토큰 없이는 작성자도 삭제 불가)
- **댓글 및 좋아요:** 사용자들은 게시글에 **익명 댓글**을 달 수 있고, 게시물 및 댓글에 **좋아요**를 눌러 공감 표시를 할 수 있습니다. 좋아요 수는 실시간으로 업데이트되며, 중복 방지를 위해 클라이언트 단 및 서버 단에서 동일 기기의 중복 클릭을 제한합니다.
- **AI 공포 이미지 생성:** (선택 기능) 사용자가 원하면, 글 작성 시 **AI 이미지를 생성**하여 첨부할 수 있습니다. 예를 들어 "어두운 골목에 서 있는 실루엣" 등의 프롬프트를 입력하면 Stable Diffusion 기반 AI 가 이미지를 생성해 줍니다. 이 기능은 무서운 분위기를 살리는 용도로, 1 회 생성당 약 **$0.01~0.03** 의 비용이 예상되며 , 생성 시간은 5~10 초 정도 소요됩니다. **NSFW 및 잔혹한 이미지 필터**를 적용하여 부적절한 결과물이 나타나지 않도록 합니다.
- **반응형 UI 및 UX:** Tailwind CSS 를 활용한 반응형 디자인으로 모바일/데스크톱 모두 쾌적하게 이용할 수 있습니다. 어두운 테마를 기본으로 하여 공포 분위기를 조성하되, **명도 대비**와 **폰트 가독성**을

충분히 확보하여 사용자의 눈 피로를 줄입니다. 무한 스크롤 또는 페이지네이션으로 피드 탐색이 가능하며, Skeleton UI 등 로딩 피드백을 제공하여 사용자 경험을 개선합니다.

- **검색 및 해시태그:** 사용자가 원하는 키워드(괴담 주제, 장소 등)로 게시물을 검색하거나 인기 해시태그로 콘텐츠를 탐색할 수 있는 기능을 염두에 두고 있습니다. (MVP 단계에서는 기본 검색만 지원하고, 향후 고도화 예정)
- **광고 통합:** 수익 모델로서 페이지 상단 배너, 본문 목록 중간, 사이드바 등에 Google AdSense 광고를 반응형으로 게재할 계획입니다. 광고는 사용자 경험을 저해하지 않도록 **최대 2~3 개 이내**로 제한하고, Lazy Loading 을 적용해 콘텐츠 로딩을 방해하지 않도록 합니다. *만약 AdSense 승인이 어려울 경우 섹션 5.3 의 대안 수익화를 고려합니다.*

3.2 사용자 정책 및 커뮤니티 가이드라인

- **익명성 보장:** 회원가입이나 로그인 절차 없이 이용 가능하며, IP 주소 등 개인정보는 게시되지 않습니다. 단, **운영상의 필요에 따라 내부적으로 IP 해시나 세션 ID 를 저장**하여 악용을 방지할 수 있습니다 (외부에 공개되거나 평상시 식별에 사용되지 않음).
- **커뮤니티 규칙:** 욕설, 혐오, 명예훼손, 노골적인 폭력 묘사 등 **운영원칙에 반하는 콘텐츠는 금지**됩니다. 이러한 콘텐츠가 게시될 경우 운영자가 즉시 삭제하고 반복 위반자는 차단 조치합니다. 이용자는 게시글/댓글 신고 기능을 통해 부적절한 콘텐츠를 알릴 수 있습니다.
- **연령 제한:** 공포 콘텐츠 특성상 **청소년에게 부적절한 내용**이 있을 수 있으므로, 필요한 경우 앱/웹 자체에 **19 금 표시** 또는 성인인증 절차를 도입할 수 있습니다. 초기 런칭 단계에서는 자체 심의를 거쳐 노골적인 유혈 묘사나 선정성 콘텐츠는 제한합니다.

4. 기술 스택 및 시스템 아키텍처

4.1 전반적인 기술 스택

- **프론트엔드:** Next.js 14 (React 기반) + TypeScript. Vercel 에 호스팅하여 SSR(서버사이드 렌더링)과 ISR(Incremental Static Regeneration)을 적절히 활용합니다. UI 는 Tailwind CSS 로 구성하고, 주요 컴포넌트는 Headless UI 등을 참고하여 개발합니다.

- **백엔드:** Supabase (PostgreSQL 데이터베이스 + Auth)와 Vercel Serverless Functions 를 조합합니다. Supabase 의 인증은 **익명 모드**로 사용하고, PostgreSQL 에 **Row Level Security (RLS)** 정책을 적용하여 데이터 접근을 제어합니다. 서버리스 함수는 Next.js API Routes 또는 Edge Functions 로 구현하며, 이미지 생성과 같은 무거운 작업은 Node.js Serverless Function 으로 처리합니다.
- **스토리지:** 사용자 업로드 이미지나 AI 생성 이미지는 **Vercel Blob Storage** 또는 Supabase Storage 에 저장하고, CDN 을 통해 배포합니다. (대용량 미디어보다는 텍스트 위주의 서비스이므로 기본 제공 용량으로 시작)
- **캐싱 및 실시간:** 데이터 캐싱 레이어로 **Upstash Redis**(Serverless Redis)를 도입하여 인기 게시물, 실시간 카운터 등에 활용합니다. 또한 Next.js 내장 ISR 기능으로 게시판 리스트 페이지를 정적 캐싱해 **뷰 생성 부하를 최소화**합니다.
- **모니터링 및 배포:** Vercel 을 통한 CI/CD 파이프라인을 설정하여 main 브랜치에 푸시 시 자동 배포합니다. 모니터링 도구로 Sentry 를 적용해 클라이언트/서버 에러를 추적하고, Google Analytics 또는 Amplitude 로 사용자 행동을 분석합니다.

4.2 시스템 아키텍처 다이어그램

아래는 제안하는 **전체 시스템 구조**를 요약한 아키텍처 개요입니다:

```
Frontend: Next.js 14 (Vercel)
 ├─── UI/UX: Tailwind CSS 다크 테마, 반응형 디자인
 ├─── 데이터 통신: React Query + SWR (CSR fetch 병행)
 ├─── 캐싱: ISR 페이지 생성 + Upstash Redis 활용
 ├─── 보안: CSRF 토큰 적용, Rate Limiting (IP 기준 요청 제한)
 └─── 익명 식별: 브라우저 세션 스토리지 기반 임시 ID + 행동 분석

Backend: Supabase (PostgreSQL + Auth)
 ├─── 인증: 익명 사용자 세션 (JWT 토큰) 발급
 ├─── DB: 게시글/댓글/좋아요 테이블 (RLS 로 사용자별 접근 통제)
 ├─── 파일 저장: Vercel Blob Storage 연동 (이미지 등)
 └─── 서버리스 함수: Next.js API Routes (좋아요 증가, AI 이미지 생성 등
비동기 처리)

Moderation & Security:
 ├─── 1 차 필터: `badwords-ko` 라이브러리로 금칙어 필터링
```

├─── 2 차 필터: Google Perspective API 통한 독성/혐오 점수 분석 [oai_citation:4‡developers.google.com](https://developers.google.com/codelabs/setup-perspective-api#:~:text=1)
├─── 3 차 대응: 커뮤니티 신고 시스템 + 운영자 수동 검토 (24 시간)
├─── Bot 차단: Cloudflare Bot Management 로 스팸 트래픽 차단 [oai_citation:5‡cloudflare.com](https://www.cloudflare.com/application-services/products/bot-management/#:~:text=)
└─── 기타 보안: XSS/CSRF 방지, HTTPS 강제, HttpOnly 쿠키, 콘텐츠 Security Policy

※ **참고:** 위 구조에서 **RLS** 는 Postgres 의 행 수준 보안(Row-Level Security) 기능으로, 익명 사용자의 DB 접근을 제어합니다. RLS 도입 시 쿼리마다 정책 검증이 추가되므로 **성능 오버헤드**가 발생할 수 있음을 인지하고 있습니다 . 따라서 RLS 정책은 최대한 단순하게 유지하고, 필요한 경우 **DB 인덱싱**과 **실행계획 튜닝**을 병행합니다. 또한 Supabase 에서 제공하는 **Edge Functions** 나 Vercel 서버리스 함수에서 일부 비즈니스 로직을 처리하여 RLS 부하를 줄이는 방안을 고려합니다.

5. 익명성 및 보안: 구현 방안과 이슈

익명 게시판의 **핵심 가치**인 익명성을 지키면서도, 악의적인 사용자를 제어하기 위한 다양한 기법을 적용합니다.

5.1 익명 권한 관리와 토큰 보안

- **게시물 편집/삭제 토큰:** 사용자가 게시글을 올리면 서버가 **난수 토큰(JWT 또는 UUID)**을 생성하여 응답 쿠키(HttpOnly, Secure 속성)로 반환하거나 클라이언트 측 localStorage 에 저장합니다. 사용자는 이 토큰을 이용해 일정 기간 내 자신의 글을 수정하거나 언제든 삭제할 수 있습니다. 토큰에는 게시물 ID 및 비밀 키 정보가 포함되어 있어 다른 글에는 사용할 수 없도록 합니다.
- **토큰 유효기간 및 재사용: 편집 토큰은 15 분~30 분 내 유효**하도록 하여, 탈취되더라도 피해 범위를 줄입니다. 삭제 토큰은 장기 보관이 필요하므로 **1 회성 일회용**으로 설계하고, 삭제 수행 시 바로 만료시킵니다.
- **보안 위협 및 대응:** 토큰 방식은 XSS 등으로 탈취 위험이 있으므로, **다층 보안**으로 방어합니다:
    - 모든 쿠키에 HttpOnly 와 SameSite=Strict 옵션 적용 (클라이언트 스크립트로 탈취 불가).

    - **콘텐츠 Security Policy(CSP)** 헤더 설정으로 외부 스크립트 삽입 차단.
    - **CSRF 토큰**을 모든 쓰기 요청에 요구하여, 토큰 탈취 시에도 악용한 크로스 사이트 요청을 막습니다.
    - **Rate Limiting:** 동일 IP 나 세션에서 과도한 글쓰기/삭제 요청 시 차단하여 브루트포스 공격 방지.
    - **모니터링:** 이상 로그인 시도나 토큰 오남용 패턴을 실시간 탐지하여 관리자에게 경고.
- **참고 사례:** 2023 년 발생한 **Storm-0558 공격**에서는, 공격자가 Microsoft 의 토큰 서명 키를 탈취하여 임의의 인증 토큰을 만들어 수십 개 조직의 이메일 계정에 무단 접근한 일이 있습니다 . 이 사례는 **토큰 자체의 탈취/위조 위험**을 잘 보여주며, 우리 서비스도 만일의 사태에 대비해 **비정상 토큰 사용 탐지**(예: 동일 토큰의 다중 IP 사용)와 **토큰 서명키 보호**(서명키를 HSM 이나 Key Management Service 에 저장) 등 최선의 보안 실천을 따라야 합니다.

5.2 익명성 vs 악용 방지: 접근 제어 전략

익명성을 악용한 **도배, 스팸, 어뷰징**을 막기 위해, **“다층 신뢰 등급”**과 **행동 기반 식별**을 도입합니다.

- **세션 단위 식별:** 디바이스 핑거프린팅은 배제하지만, 브라우저당 하나의 **세션 ID**(쿠키 또는 localStorage 에 저장)를 발급하여 사용자의 활동을 임시 식별합니다. 이 세션 ID 는 24~48 시간 유지되며, 해당 기간 동안 과도한 부정행위(예: 짧은 시간 내 게시글 10 개 연속 작성 등)가 발견되면 일시적으로 **행동 제한**을 가합니다.
- **IP 및 UA 활용:** IP 주소와 User-Agent (브라우저 정보)는 직접적인 식별자으론 사용하지 않지만, 서버 측에서 **대략적인 접근 빈도 모니터링**에 활용합니다. 예를 들어 **동일 IP 대역에서 짧은 간격으로 수백 개 요청**이 발생하면 해당 IP 대역을 일정 시간 차단하거나 reCAPTCHA 인증을 요구합니다.
- **신뢰 등급 부여:** 커뮤니티 기여도가 높은 사용자를 내부적으로 식별하여(예: 여러 건의 글이 호응을 얻고 신고 기록이 없는 세션), 이들에게는 일정 수준의 **신뢰 점수**를 부여합니다. 신뢰 점수가 높으면 게시글이 자동 차단 필터에 덜 걸리도록 하고, 낮은 새 세션의 사용자는 초기에 **엄격한 필터** 적용 및 게시 빈도 제한을 둡니다.
- **VPN/프록시 탐지:** Tor 등 **익명 프록시를 이용한 접속**은 악용 사례가 많으므로, Cloudflare 등의 **VPN 탐지 API** 를 활용해 의심 트래픽을 식별합니다. 필요 시 해당 요청에는 추가 CAPTCHA 를 요구하거나 글쓰기 기능을 제한합니다.

- **법적 컴플라이언스:** 한국의 **인터넷 실명제는 위헌 결정**이 났지만, 악성 게시글에 대한 관리자 책임은 여전히 존재합니다. 운영자는 수사 협조 요청 시 익명 사용자라도 관련 정보를 제공해야 할 수 있으므로, **백엔드에 최소한의 로그(IP 해시, 타임스탬프 등)**는 보관합니다. 단, 개인정보보호 지침에 따라 이러한 정보는 **암호화**하여 저장하고 90 일~1 년 등 일정 기간 후 파기합니다.

5.3 콘텐츠 모더레이션 및 AdSense 정책 준수

"익명"과 **"UGC (사용자생성콘텐츠)"** 조합은 콘텐츠 관리 측면에서 가장 어려운 분야입니다. 이를 해결하기 위해 **3 단계 모더레이션 시스템**을 구현합니다:

1. **1 차 자동 필터 – 금칙어 리스트:** 우선 사용자 입력 시 한국어 욕설 및 비속어, 혐오 표현에 대한 **키워드 필터링**을 적용합니다. 예를 들어 오픈소스 목록인 badwords-ko 등을 활용하여 한국어 욕설/차별어 사전을 구축하고, 금칙어가 포함된 게시물은 즉시 차단하거나 별도 검토 큐에 넣습니다. 또한 "자살", "자해" 등의 단어가 포함되면 이용자에게 관련 도움 리소스를 제공하고 게시를 재확인합니다.
2. **2 차 AI 검열 – 내용 분석:** 1 차 필터를 통과한 콘텐츠에 대해 Google 의 Perspective API 등의 머신러닝 서비스를 활용해 **유해성 점수**를 계산합니다. Perspective API 는 **댓글이나 글의 독성이 얼마나 높은지** 등 지표를 0~1 사이로 제공하며 , 예컨대 욕설, 괴롭힘, 증오 표현 등에 반응합니다. 설정한 임계치 이상으로 **"TOXICITY"** 점수가 높게 나온 글은 곧바로 운영자에게 플래그를 띄우거나, 커뮤니티에 노출되기 전에 승인이 필요하도록 합니다. *(참고: AI 판정은 오탐지 가능성이 있으므로 완전 자동 삭제보다는 **플래그 및 숨김** 처리로 구현합니다.)*
3. **3 차 사람 감시 – 신고 및 수동 조치:** 사용자들이 **신고 버튼**을 통해 문제 게시물을 제보할 수 있게 하고, **커뮤니티 자율 규제**도 유도합니다 (예: 일정 수 이상의 사용자가 downvote 또는 신고 시 자동 숨김). 동시에 관리자가 **24 시간 모니터링 체계**를 갖추어, 신고 접수되거나 자동 플래그된 콘텐츠를 신속히 검토/삭제합니다. 초기 운영 인력이 부족할 경우 밤시간대는 주요 키워드 기반 알림 위주로 대응하고, 점차 모니터 인력을 확보합니다.

AdSense 정책 준수: 위 3 단계에도 불구하고 **정책 위반 콘텐츠가 100% 차단된다고 보장하기 어렵기 때문에**, AdSense 를 도입할 경우 **운영자가 모든 UGC 에 책임을 진다는 각오로 임해야 합니다** . 구글은 "사용자 댓글이 달린

페이지도 게시자가 정책을 준수하도록 관리해야 하며, 위반 시 계정 정지될 수 있다"고 명시하고 있습니다. 따라서:

- **광고 게재 위치 통제:** 댓글이 달리는 실시간 피드나 사용자 프로필 근처에는 광고를 피하고, 주로 **본문 상단/하단**처럼 운영자가 컨텐츠를 통제하기 쉬운 영역에 게재합니다.
- **연령 제한 설정:** AdSense 콘솔에서 콘텐츠 카테고리를 조정하여, 민감한 카테고리(공포, 폭력 등)에 대한 광고 노출을 제한하거나 우회합니다.
- **모니터링:** Google AdSense 에서 제공하는 **정책 위반 알림** 및 **제한 보고**를 수시로 확인하고 즉각 조치합니다. 예를 들어 "무효 트래픽 증가" 경고가 오면 Cloudflare 로그 등으로 원인을 파악하고 필요한 경우 광고 슬롯을 일시 중지합니다.
- **클라우드플레어 봇 관리:** 광고 수익을 노린 **클릭 작업**이나 트래픽 조작을 막기 위해 **Cloudflare Bot Management** 를 도입하여, 알려진 나쁜 봇이나 의심스러운 트래픽을 차단합니다. Cloudflare Bot Management 는 **머신러닝과 행동 분석을 통해** 실시간으로 봇 여부를 판정하고, CAPTCHA 없이도 악성 봇을 걸러낼 수 있습니다 . 이를 통해 AdSense 정책의 주요 위반 원인인 **무효 트래픽**을 선제적으로 줄입니다.

대안 수익화: 만약 AdSense 승인에 실패하거나 정책 유지가 어렵다고 판단되면, **대안으로 국내 광고 플랫폼**을 활용합니다. **네이버 애드포스트**는 블로그 형태의 UGC 에 적용하기 쉽고 초기 승인 문턱이 낮습니다. **카카오 애드핏**도 모바일 앱/웹에 적합한 광고 네트워크로, 한국 사용자 대상의 fill-rate 가 높을 것으로 기대됩니다. 이들 플랫폼의 광고를 우선 적용하고, 추후 커뮤니티 규모가 성장하면 AdSense 를 재신청하거나 프리미엄 구독, 굿즈 판매 등의 다른 수익 모델을 모색합니다.

6. 성능 및 확장성 고려

6.1 서버리스 환경 최적화

본 서비스는 서버리스 기반으로 구현되어 초기 비용 면에서 이점이 있으나, **높은 동시성 처리**와 **낮은 지연 시간** 유지를 위해 몇 가지 패턴을 적용합니다.

- **콜드 스타트 대응:** Vercel 서버리스 함수는 간헐적 호출 시 콜드 스타트로 수백 ms 지연이 발생할 수 있습니다. 유저 경험에 민감한 API(예: 글 목록

불러오기)는 **Edge Function(Region: ICN)**으로 배포하여 콜드 스타트를 최소화하고, 백그라운드 작업(API Routes)은 사전 워밍업하거나 주기적 호출로 캐시를 유지합니다.

- **좋아요 동시 증가 처리:** 여러 사용자가 동시에 같은 게시물에 좋아요를 누를 때 생길 수 있는 **경합 상황(race condition)**을 다루기 위해 **낙관적 UI 업데이트 + 백엔드 보정** 전략을 채택합니다. 사용자가 좋아요를 클릭하면 클라이언트는 즉시 UI 에 반영하되, 서버에는 **이벤트 큐**로 처리 요청을 보냅니다. 서버에서는 Upstash Redis 의 원자적 **INCR 명령**을 활용하여 좋아요 수를 누적하고, 최종값을 DB 에 반영합니다. 이러한 **Eventually Consistent** 패턴을 통해 사용자 경험은 실시간성을 느끼면서, 실제 데이터 일관성은 약간 지연되어 보장됩니다. (참고로 Redis 연산은 단일 스레드여서 Race condition 없이 누적 가능)
- **SSR/정적 생성 활용:** 게시판 목록, 인기글 등 변동이 비교적 적은 페이지는 Next.js 의 **ISR** 로 정적 생성하여 캐싱합니다. 새로운 글이 등록될 때만 해당 페이지를 **Revalidate** 하여, 대부분의 트래픽에 대해서는 정적 파일 제공으로 응답 시간을 50ms 이하로 유지합니다. 댓글처럼 실시간성이 필요한 부분은 클라이언트에서 SWR/React Query 로 주기적 폴링하거나 WebSocket (추후)으로 업데이트합니다.
- **데이터베이스 부하 분산:** 읽기 쿼리가 몰리는 경우를 대비해 Supabase Postgres 의 **리드 레플리카**를 활성화할 수 있습니다. 또한 인기글 조회수 증가 등 간단한 카운트는 DB 대신 Redis 캐시로 누적하고 일정 주기마다 flush 하는 **Write-behind** 전략을 사용하여 DB 트랜잭션 횟수를 줄입니다.
- **이미지 생성 작업 최적화:** Stable Diffusion 을 이용한 AI 이미지 생성은 5-10 초 가량 시간이 걸리므로, 이 작업은 **백엔드에서 비동기**로 처리합니다. 사용자 요청 시 바로 응답하지 않고, **작업 수락 응답**을 준 뒤(예: "이미지 생성 중" 메시지), 작업 완료 시점에 WebSocket 이벤트나 폴링 응답으로 이미지를 전달합니다. 서버리스 함수가 장시간 실행되는 것을 피하고, 외부 AI API 호출로 인한 **타임아웃**을 방지하기 위한 설계입니다.

6.2 예상 사용자 규모와 비용 산정

초기 예상 사용자 수는 **월 1 만 명** (일 활성 사용자 500 명 내외) 수준이며, 이에 따른 월간 인프라 비용은 아래와 같습니다:

- **Vercel 프로(plan):** 약 $51.5/월 - 프로젝트 팀원 계정 및 일정 수준의 서버리스 사용량 포함【추정】.
- **Supabase 프로(plan):** 약 $100.4/월 - 데이터베이스 호스팅 및 50k MAU, 8GB 저장소, 100GB 대역폭 기본 포함 .(추가 사용량 발생 시 별도 과금)
- **Upstash Redis:** 약 $20/월 - 기본 플랜 (저용량/저트래픽 시).

- **도메인 및 인증서:** Cloudflare 이용 시 연간 도메인 비용 약 $10, SSL 인증서는 무료.
- **기타 서비스:** AI 이미지 생성 API 비용은 **사용량 기반**으로, 1,000 장 생성 시 약 $10 (0.01 * 1000) 정도. 이를 많이 사용할 경우 별도 예산 편성이 필요합니다.

합계: 월 **$172** (한화 약 23 만 원) 수준으로 추산됩니다. 사용자 1 인당 월 비용 환산시 약 $0.017 로, 수익화만 원활하다면 충분히 감당 가능한 수준입니다.

또한 추후 **캐싱 최적화**와 **리소스 업그레이드 지연 전략**(예: 최대 수용량까지는 프로플랜 유지)으로 월 $30-50 까지도 절감 여지가 있습니다. 만약 사용자수가 크게 증가하면 Supabase **엔터프라이즈 요금제**나 자체 인프라 이전도 검토해야 하지만, 이는 100 만 MAU 이상에서의 시나리오입니다.

7. 기술적 난제 및 해결 방안

서비스 구현 과정에서 예상되는 **핵심 기술 과제**와 이에 대한 대응 전략은 다음과 같습니다.

7.1 익명성 vs 악용의 트레이드오프

문제: 완전한 익명성을 보장하면서도 악성 유저를 제어하는 것은 근본적으로 모순적인 도전입니다. 익명 환경에서는 사용자 차단 및 제재가 어렵고, 이를 악용한 도배/혐오 발언이 발생할 수 있습니다.

해결: 다층 방어 체계를 구축합니다. 우선 Rate Limiting 으로 초당/분당 게시글 작성 횟수를 제한하고, **콘텐츠 필터**(욕설/스팸 차단)를 1 차 적용합니다. 그 다음 **행동 패턴 분석**으로 의심 사용자(예: 새벽 시간대 1 시간 동안 50 개 글 작성 등)를 색출해 자동 조치합니다. 마지막으로 **커뮤니티 자율 규제**를 도입하여, 사용자들의 신고, 투표로 문제 인물을 솎아낼 수 있게 합니다. 이러한 계층적 접근으로 익명성을 유지하되 악용 여지는 최소화합니다.

7.2 세션 및 사용자 식별의 복잡성

문제: 로그인 없는 서비스에서 사용자의 연속성을 어느 정도 유지하려면 세션 관리가 필수인데, **쿠키 거부**나 **브라우저 시크릿 모드** 사용 시 동일 사용자를 구분하기 어렵습니다. 또한 토큰 기반 세션은 앞서 언급한 탈취 위험이 있습니다.

해결: 세션 토큰 + 클라이언트 스토리지의 혼합 방식을 씁니다. 기본적으로 서버는 익명 JWT 세션을 발급하고, 클라이언트는 이를 HttpOnly 쿠키로 유지해 인증에 사용합니다. 동시에 브라우저 localStorage 또는 IndexedDB 에 식별용 임시 ID 를 저장하여, 사용자가 쿠키를 지워도 동일 브라우저에서 크게 벗어나지 않는 한 행동 패턴을 이어받도록 합니다. (예: localStorage 의 ID 를 읽어 동일인으로 간주) 이때 localStorage ID 는 무기한 사용하지 않고 주기적으로 rotation 하여 개인정보 침해 우려를 낮춥니다.

또한 세션에 중요한 권한정보를 담지 않고, **서버 측 RLS 정책**과 대조하여 매 요청을 검증함으로써 세션 하이재킹 시 피해를 경감합니다.

7.3 서버리스 환경의 제약 극복

문제: Vercel 서버리스 함수는 CPU/메모리 제약과 짧은 실행 시간 제한이 있어, 이미지 생성 같은 heavy 작업이나 대용량 트래픽 처리에 한계가 있습니다. 또한 지역 CDN 으로 인한 **네트워크 지연** 이슈도 있습니다.

해결: 작업 유형별 분리를 명확히 합니다. 즉각 응답이 필요한 API 는 Edge Function 으로 배치하고, **무거운 연산은 아예 프론트엔드에서 외부 API 직접 호출**도 고려합니다 (예: Stable Diffusion API 를 클라이언트에서 호출하고, 결과만 서버에 제출). 또는 Vercel 의 Background Functions (새로 지원될 경우)를 이용해 제한 시간 이상 걸리는 작업도 백그라운드로 완료시킵니다.

동시성 제한에 대비해서는, 초당 요청이 1000 건을 넘길 경우 Cloudflare 등 **프록시 레이어에서 QoS** 를 적용해 순차 처리하거나 (응답에 약간의 딜레이

허용), 인기 API 는 별도 서버로 분리(예: Cloud Run 서비스)하여 확장성 확보를 고려합니다.

7.4 데이터베이스 성능 및 RLS 정책

문제: Supabase(Postgres) 상에서 **RLS** 를 사용하면 쿼리마다 정책 조건 확인이 추가되어 **성능 저하**가 우려됩니다 . 또한 익명 사용자의 경우 auth.uid()가 null 인 상태로 처리되어 정책 정의가 까다롭습니다. 동시에, 게시판 특성상 **읽기 트래픽**이 매우 높아질 수 있습니다.

해결: 정확한 인덱싱과 단순한 정책으로 RLS 로 인한 오버헤드를 줄입니다. 예를 들어 게시물 테이블에는 id, created_at 외에 delete_token 등에 인덱스를 추가해 삭제 시 검증을 O(1)로 만듭니다. RLS 정책은 가능한 한 auth.uid() 사용을 피하고 세션 토큰의 클레임 (예: request.jwt.claims.delete_token)과 행을 비교하는 간단한 조건만 사용합니다. 또한 Supabase 의 **Policy Debug 툴**로 쿼리 플랜을 모니터링하며 튜닝합니다.

만약 RLS 로도 힘든 복잡한 로직(예: 다중 조건에 따른 노출 제어 등)은, 차라리 **서버리스 함수로 래핑**하여 인증/검증을 수행한 뒤 안전한 쿼리를 날리는 구조로 변경할 수 있습니다 (즉, RLS 를 일부 포기하고 서버 로직으로 대체).

7.5 AI 이미지 생성 비용 관리

문제: AI 로 이미지를 생성해주는 기능은 사용자 호응을 얻을 수 있으나, **API 비용**과 **실행 시간** 측면에서 부담이 됩니다. 활성 사용자가 많아질 경우 예상치 못한 비용 증가가 발생할 수 있습니다.

해결: 사용량 제한과 최적화가 필요합니다. 예를 들어 *일일 3 회 무료 생성, 추가 생성은 쿨타임 1 시간* 등의 제한을 두어 남용을 막습니다. 그리고 가능한 한 **오픈소스 경량 모델**을 자체 호스팅하는 것도 검토합니다 (예: Stable Diffusion 1.5 를 ONNX 로 변환해 Cloud Run 에 올려두고 필요 시 호출). 이 경우 초기

비용이 들지만 호출량이 많아지면 장기적으로 API 호출 대비 저렴할 수 있습니다.

또한 **멀티 스케일 캐싱**: 동일하거나 유사한 프롬프트에 대한 생성 결과를 캐싱하여 중복 비용을 방지합니다 (완전히 동일 프롬프트 여러 번 생성 시 최근 결과 제시 등). 마지막으로, 일정량 이상의 AI 기능 이용은 **프리미엄 서비스**로 전환하여 수익 보전도 고려합니다.

8. 개발 일정 및 마일스톤

초기 기획 대비 보안/정책 대응 작업이 늘어남에 따라 개발 일정을 재조정했습니다. 새로운 **개발 기간은 총 18~22 주(약 5 개월)**로 예상하며, 주요 단계별 계획은 다음과 같습니다:

1. **요구사항 확정 & 설계 (3 주)** – 법률 자문을 받아 개인정보 및 청소년보호 이슈를 최종 점검하고, 수정된 PRD 에 따라 상세 기술 설계를 완료합니다. DB 스키마 확정, API 명세 작성, UI/UX 프로토타입 제작도 이 단계에서 병행합니다.
2. **핵심 기능 개발 (8 주)** – 익명 게시판 CRUD 기능, 댓글 및 좋아요 구현, 익명 인증 및 보안 토큰, 기본 모더레이션(필터링) 적용을 완료합니다. 프론트엔드 React 컴포넌트 구현과 백엔드 API 개발을 동시에 진행하며, 주 1 회 전체 통합 테스트를 수행합니다.
3. **보안 강화 및 정책 준수 점검 (4 주)** – 2 단계에서 구축한 기능 위에 보안 레이어를 추가합니다. XSS/CSRF 방지 테스트, 부하테스트(시나리오: 한 사용자가 분당 100 개 글 작성) 등을 거쳐 방어체계를 튜닝합니다. 또한 AdSense 정책 가이드에 맞춰 **모의 콘텐츠 검열**을 실시하고, 문제 소지가 있는 부분은 수정합니다. 이 기간에 Google AdSense 승인을 위한 사이트 준비를 마칩니다 (콘텐츠 5~10 개 이상 게시 등).
4. **클로즈드 베타 테스트 (3 주)** – 소수의 초대된 사용자를 대상으로 베타 테스트를 실시합니다. 피드백을 받아 UI 개선, 버그 수정, 모니터링 설정을 조율합니다. 이 단계에서 **서버 비용 및 응답 속도** 데이터를 수집하여, 필요한 최적화(쿼리 개선, 캐시 확대 등)를 적용합니다.
5. **출시 및 버퍼 기간 (추가 4 주)** – 2024 년 1 분기 말 정식 출시를 목표로 합니다. 출시 직후 발생하는 예기치 못한 이슈를 대응하기 위해 한 달간을 버퍼로 두고, 개발팀이 긴밀히 모니터링합니다. 2 분기부터는 정식 운영 전환과 함께 다음 기능(AI 추천 시스템 등) 기획을 시작합니다.

마일스톤 요약:

- **Week 1-3:** 기획 확정, 설계 완료
- **Week 4-11:** 기능 개발 및 1 차 테스트 완료 (MVP 구현)
- **Week 12-15:** 보안/모더레이션 강화, AdSense 준비
- **Week 16-18:** 클로즈드 베타 & 피드백 반영
- **Week 19:** v1.0 런칭 (베타 딱지 제거)
- **Week 20-22:** 초기 운영 안정화 및 최종 조율

9. 최종 권고사항 및 결론

종합적인 검토 결과, **무서핑 서비스는 충분히 구현 가능하지만 초기 계획의 일부 수정이 불가피**합니다. 특히 **디바이스 핑거프린팅을 통한 익명성 유지 전략은 폐기**하고, **AdSense 에 대한 과도한 기대를 경계**해야 합니다. 아래에 단계별 권고사항을 정리합니다.

9.1 즉시 실행해야 할 조치 (Critical)

- **디바이스 핑거프린팅 사용 중지:** 개인정보 법규와 기술 트렌드를 고려해 해당 방안을 완전히 제외하고, 앞서 제시한 세션 기반 익명 식별로 대체하십시오.
- **다층 보안 프레임워크 구축:** 서비스 출시 전에 HTTPS 강제, HttpOnly 쿠키, CSRF 토큰, Rate Limiting, 콘텐츠 보안 정책 등 **보안 요소를 전부 활성화**해야 합니다. 초기에는 과할 정도로 보안설정을 하고, 추후 필요시 완화하는 편이 안전합니다.
- **대안 수익 모델 적용:** 런칭 시 바로 AdSense 에 의존하지 말고, **네이버 애드포스트 우선 적용** 등으로 수익원을 다변화하십시오. AdSense 는 내부 콘텐츠가 쌓이고 커뮤니티 품질이 검증된 후에 도전하는 것도 좋은 전략입니다.

9.2 1 개월 이내 추진 (High Priority)

- **개인정보보호 및 컴플라이언스 정비:** 이용약관/개인정보처리방침을 마련하고, 쿠키 동의 배너(필요 시) 등을 준비합니다. 청소년 유해매체물 관련 고지 의무도 검토하십시오 .
- **콘텐츠 모더레이션 시스템 완성:** AI 필터 튜닝, 금칙어 사전 업데이트 등을 통해 자동 필터 정확도를 높이고, 운영자용 **모더레이션 대시보드**(신고 관리, 사용자 차단 UI 등)를 구현합니다.

- **성능 모니터링 도구 도입:** New Relic 이나 Supabase Insights 등을 연결해 쿼리 성능, API 응답 시간을 모니터링하세요. 특히 게시글 리스트 API 와 같은 핵심 경로의 응답 시간이 SLA 내에 있는지 추적해야 합니다.

9.3 3 개월 이내 추진 (Medium Priority)

- **AI 기반 스팸/봇 탐지 고도화:** Cloudflare Bot Management 에 더해, 자체적으로 사용자 행동 데이터를 학습하여 **스팸 계정 식별 모델**을 개발합니다. 예를 들어 Python + scikit-learn 으로 간단한 분류기를 만들어 지속 개선합니다.
- **커뮤니티 자율 규제 기능:** 사용자 레벨/포인트 제도, 베스트 댓글 투표, 일정 신고 누적 시 자동 블라인드 처리 등 **커뮤니티 내에서 콘텐츠 정화를 유도하는 기능**을 추가합니다.
- **확장성 검증:** 향후 10 만 MAU 까지 트래픽 증가를 가정한 **부하 테스트**를 실시하고, 병목을 프로파일링합니다. 필요하다면 데이터베이스를 수평 분할하거나, 검색 기능에 Elasticsearch 도입 등 **아키텍처 개선안**을 마련합니다.

결론적으로, 무서핑 서비스는 **기술적으로 구현 가능하고 시장성 있는 아이디어**입니다. 다만 익명성이 주는 자유와 그로 인한 위험을 모두 관리할 수 있어야 지속 가능한 운영이 가능합니다. 이번 개정된 PRD 에서는 기존 계획 대비 **보안**, **모더레이션**, **컴플라이언스** 측면을 대폭 강화하였습니다.

향후 이 방향대로 개발을 진행한다면, 공포 이야기를 좋아하는 많은 유저들이 안심하고 참여할 수 있는 익명 커뮤니티를 구축할 수 있을 것입니다. 초기에는 보수적으로 운영하더라도, 점진적으로 유저 신뢰를 쌓고 기능을 확장해 나간다면 **“한국의 Reddit 공포 게시판”**에 버금가는 활발하고 건전한 커뮤니티로 성장할 것으로 기대합니다.

참고자료: 관련 기술 구현 사례로 Next.js 와 Supabase 를 활용한 익명 게시판 예제, Tailwind CSS 다크 테마 디자인 가이드, Google AdSense UGC 정책 (공식 블로그) , 익명 게시판의 보안 취약점 연구, PostgreSQL RLS 성능 팁 문서 , 콘텐츠 모더레이션을 위한 Perspective API 사용법 등을 참고하였습니다. (세부 자료 출처는 본 문서 각주에 표시)