1. 개발 환경 전제

- 프레임워크: React (Next.js App Router)
- 배포: Vercel
- 백엔드/DB: Supabase (Auth + PostgreSQL + Realtime)
- IDE: Google Project IDX (Antigravity 기반 클라우드 IDE)
- 스타일: Tailwind CSS + shadcn/ui
- 상태관리: Zustand
- AI 연동: OpenAI API (또는 멀티모델 라우터)

---

2. 개발 프롬프트

- PHASE 0: 프로젝트 진행 Step-by-step 프롬프트
- 프롬프트 #0-A — Next.js 프로젝트 생성 + Tailwind + 컬러 시스템

당신은 시니어 풀스택 React 개발자입니다.

Google Project IDX (Antigravity) 환경에서 새 프로젝트를 처음부터 세팅합니다.

모든 코드를 완전하게 작성하세요. placeholder 주석이나 "TODO"는 절대 남기지 마세요.

모든 UI 텍스트는 한국어로 작성하세요.

[프로젝트 정보]

- 서비스명: "우리집 AI 안전 코치"

- 목적: 비기술직 학부모(어머니)가 자녀와 함께 AI 리터러시를 단계별 미션으로 학습하는 모바일 우선 인터랙티브 웹앱

[이번 프롬프트에서 할 일 — 프로젝트 초기화 + 스타일 시스템만]

1. Next.js 14 프로젝트를 App Router + TypeScript 로 생성하세요.

  - `npx create-next-app@latest` 기준의 초기 파일 구조를 작성하세요.

  - `next.config.js`를 작성하세요.

  - `tsconfig.json`은 Next.js 기본값을 사용하세요.

2. Tailwind CSS v3 를 설치하고 설정하세요:

- `tailwind.config.ts` 파일에 아래 커스텀 컬러 팔레트를 정의하세요:

```
colors: {
  primary: '#F4845F',     // 소프트 코랄
  'primary-light': '#FBBDA1',
  secondary: '#7EC8C8',   // 밝은 민트
  'secondary-light': '#B5E3E3',
  accent: '#B8B8FF',      // 연한 라벤더
  'accent-light': '#D4D4FF',
  background: '#FFF8F0',  // 크림화이트
  surface: '#FFFFFF',
  text: '#2D2D2D',        // 다크차콜
  'text-muted': '#6B7280',
  'text-light': '#9CA3AF',
  success: '#4ADE80',
  warning: '#FBBF24',
  danger: '#F87171',
  border: '#E5E7EB',
}
```

- 다크모드는 비활성화하세요 (다크모드 관련 설정 제거).

- `globals.css`에 Tailwind 디렉티브와 기본 body 스타일을 작성하세요:

  - body 배경: background (#FFF8F0)

  - body 글꼴: text (#2D2D2D)

- 기본 font-family: 'Pretendard', -apple-system, sans-serif

3. `postcss.config.js`를 작성하세요.

4. `app/layout.tsx` — 루트 레이아웃을 작성하세요:

- html lang="ko"

- metadata: title "우리집 AI 안전 코치", description "하루 15 분, 아이와 함께하는 AI 안전 미션"

- body 에 globals.css 적용

- children 을 감싸는 최소 래퍼만 (아직 네비게이션 없음)

5. `app/page.tsx` — 임시 랜딩 페이지:

- 화면 중앙에 "우리집 AI 안전 코치" 제목과 "서비스 준비 중입니다" 텍스트만 표시

- primary 컬러로 제목 스타일링

[생성할 파일 목록]

- next.config.js

- tsconfig.json

- tailwind.config.ts

- postcss.config.js

- app/globals.css

- app/layout.tsx

- app/page.tsx

- package.json (의존성 목록 포함)

---

- 프롬프트 #0-B — shadcn/ui 설치 + 기본 UI 컴포넌트

이전 프롬프트(#0-A)에서 다음 파일들이 이미 생성되어 있습니다:

- next.config.js, tsconfig.json, tailwind.config.ts, postcss.config.js

- app/globals.css, app/layout.tsx, app/page.tsx

- package.json

[이번 프롬프트에서 할 일 — shadcn/ui 설치 및 자주 사용할 기본 컴포넌트 생성]

1. shadcn/ui 를 초기화하세요:

  - `components.json` 파일을 작성하세요 (style: "default", tailwindCSS 사용, alias: "@/components/ui")

  - `lib/utils.ts`에 `cn()` 유틸 함수를 작성하세요 (clsx + tailwind-merge)

2. 아래 shadcn/ui 컴포넌트를 `components/ui/` 폴더에 직접 작성하세요.

  (npx shadcn-ui add 명령어의 결과물과 동일한 코드를 직접 작성해주세요)

  작성할 컴포넌트 (각각 별도 파일):

  a. `components/ui/button.tsx`

    - variant: default(primary 배경), secondary, outline, ghost, destructive

    - size: default, sm, lg, icon

    - 모바일 터치 타깃 최소 44px 보장 (min-h-[44px] min-w-[44px])

  b. `components/ui/card.tsx`

    - Card, CardHeader, CardTitle, CardDescription, CardContent, CardFooter

    - 기본 스타일: surface 배경, border, rounded-xl, shadow-sm

  c. `components/ui/input.tsx`

    - 기본 텍스트 입력

- 높이 44px, rounded-lg, border

- focus 시 primary 컬러 ring

d. `components/ui/textarea.tsx`

- 멀티라인 텍스트 입력

- 최소 높이 100px

e. `components/ui/checkbox.tsx`

- 체크 시 primary 컬러

- 라벨 클릭 가능

- 크기: 24px x 24px (모바일 터치 용이)

f. `components/ui/badge.tsx`

- variant: default, secondary, outline, success, warning, danger

g. `components/ui/progress.tsx`

- value prop (0-100)

- primary 컬러 fill, border 컬러 track

- height: 8px, rounded-full

h. `components/ui/toast.tsx` + `components/ui/toaster.tsx` + `components/ui/use-toast.ts`

- 하단에서 올라오는 토스트

- variant: default, success, warning, destructive

- 3 초 후 자동 사라짐

i. `components/ui/dialog.tsx`

- 모달 다이얼로그

- Dialog, DialogTrigger, DialogContent, DialogHeader, DialogTitle, DialogDescription, DialogFooter

- 모바일에서 하단 시트(bottom sheet) 스타일로 표시 (max-h-[85vh], rounded-t-2xl)

j. `components/ui/slider.tsx`

- min, max, step, value, onChange

- 트랙: border 컬러, 채움: primary 컬러

- 노브: 24px, primary 컬러

3. `components/ui/index.ts` — 모든 UI 컴포넌트를 re-export 하는 배럴 파일을 작성하세요.

4. package.json 의 dependencies 에 아래를 추가하세요:

- clsx, tailwind-merge

- class-variance-authority

- @radix-ui/react-checkbox

- @radix-ui/react-dialog

- @radix-ui/react-slider

- @radix-ui/react-slot

- lucide-react

[생성할 파일 목록]

- components.json

- lib/utils.ts

- components/ui/button.tsx

- components/ui/card.tsx

- components/ui/input.tsx

- components/ui/textarea.tsx

- components/ui/checkbox.tsx

- components/ui/badge.tsx

- components/ui/progress.tsx

- components/ui/toast.tsx

- components/ui/toaster.tsx

- components/ui/use-toast.ts

- components/ui/dialog.tsx

- components/ui/slider.tsx

- components/ui/index.ts

---

- 프롬프트 #0-C — 폴더 구조 + 레이아웃 셸 (BottomNav + Sidebar)

이전 프롬프트(#0-A, #0-B)에서 생성된 파일:

- Next.js 프로젝트 기본 파일 전체

- components/ui/ 폴더의 모든 UI 컴포넌트

- lib/utils.ts

[이번 프롬프트에서 할 일 — 폴더 구조 스캐폴딩 + 반응형 레이아웃 셸]

1. 아래 빈 페이지 파일들을 생성하세요. 각 파일에는 해당 페이지의 제목을 표시하는 최소 컴포넌트만 넣으세요.

(예: export default function LoginPage() { return <h1 className="text-2xl font-bold">로그인</h1> })

생성할 페이지 파일:

- app/(auth)/login/page.tsx — "로그인"

- app/(auth)/signup/page.tsx — "회원가입"

- app/(dashboard)/layout.tsx — 대시보드 공용 레이아웃 (아래 3 번에서 상세 구현)

- app/(dashboard)/onboarding/page.tsx — "온보딩"

- app/(dashboard)/missions/page.tsx — "미션"

- app/(dashboard)/missions/[id]/page.tsx — "미션 상세"

- app/(dashboard)/family-constitution/page.tsx — "가족 AI 헌법"

- app/(dashboard)/progress/page.tsx — "진행현황"

2. 아래 빈 폴더와 placeholder 파일을 생성하세요:

- components/layout/ (빈 폴더 → 아래 3 번에서 파일 생성)

- components/missions/ — index.ts (빈 배럴)

- components/onboarding/ — index.ts (빈 배럴)

- lib/supabase/ — index.ts (빈 배럴, 다음 프롬프트에서 구현)

- lib/store/ — index.ts (빈 배럴)

- lib/constants/ — index.ts (빈 배럴)

- lib/types/ — index.ts (빈 배럴)

- public/icons/ — .gitkeep

- public/illustrations/ — .gitkeep

3. 반응형 레이아웃 셸을 구현하세요:

a. `components/layout/BottomNav.tsx` — 모바일 하단 네비게이션 바

- 4 개 탭: 홈, 미션, 가족헌법, 진행현황

- 각 탭 아이콘 (lucide-react 사용):

- 홈: Home 아이콘, 경로: /dashboard/missions

- 미션: Target 아이콘, 경로: /dashboard/missions

- 가족헌법: ScrollText 아이콘, 경로: /dashboard/family-constitution

- 진행현황: BarChart3 아이콘, 경로: /dashboard/progress

- 현재 경로와 일치하는 탭은 primary 컬러, 나머지는 text-muted

- `usePathname()` 훅으로 현재 경로 감지

- 고정 위치: fixed bottom-0, 높이 64px, surface 배경, 상단 border

- md 이상 화면에서는 hidden

b. `components/layout/Sidebar.tsx` — 데스크탑 좌측 사이드바

- md 이상 화면에서만 표시

- 폭: 240px, 고정 위치

- 상단: 서비스 로고 텍스트 "🏠 우리집 AI 안전 코치" (font-bold, text-lg)

- 네비게이션 링크 4 개 (BottomNav 와 동일한 4 개 항목)

- 각 링크: 아이콘 + 텍스트, padding 12px, rounded-lg

- 활성 링크: primary-light 배경 + primary 텍스트

- 비활성 링크: transparent 배경 + text-muted

- 하단: 사용자 닉네임 + 로그아웃 버튼 (placeholder, 기능은 나중에)

c. `components/layout/Header.tsx` — 모바일 상단 헤더

- 높이 56px, surface 배경, 하단 border

- 왼쪽: 서비스명 "우리집 AI 안전 코치" (text-base, font-semibold)

- 오른쪽: 프로필 아이콘 (User 아이콘, 32px 원형)

- md 이상에서는 hidden (사이드바가 대체)

d. `app/(dashboard)/layout.tsx` — 대시보드 공용 레이아웃 수정

- md 이상: 좌측에 Sidebar, 우측에 children (ml-[240px])

- md 미만: 상단에 Header, 하단에 BottomNav, 사이에 children

- children 영역: pb-20 (BottomNav 겹침 방지), px-4, max-w-2xl mx-auto

- Toaster 컴포넌트를 여기에 배치

[생성할 파일 목록]

- app/(auth)/login/page.tsx
- app/(auth)/signup/page.tsx
- app/(dashboard)/layout.tsx
- app/(dashboard)/onboarding/page.tsx
- app/(dashboard)/missions/page.tsx
- app/(dashboard)/missions/[id]/page.tsx
- app/(dashboard)/family-constitution/page.tsx
- app/(dashboard)/progress/page.tsx
- components/layout/BottomNav.tsx
- components/layout/Sidebar.tsx
- components/layout/Header.tsx
- components/layout/index.ts (3 개 컴포넌트 re-export)
- components/missions/index.ts
- components/onboarding/index.ts
- lib/supabase/index.ts
- lib/store/index.ts
- lib/constants/index.ts
- lib/types/index.ts

---

- PHASE 1: Supabase 백엔드
- 프롬프트 #1-A — Supabase 클라이언트 설정 + TypeScript 타입 정의

이전 프롬프트(#0-A ~ #0-C)에서 Next.js 프로젝트, UI 컴포넌트, 레이아웃 셸이 완성되어 있습니다.

[이번 프롬프트에서 할 일 — Supabase 클라이언트 연결 + 타입 정의]

1. Supabase 클라이언트 파일 2 개를 작성하세요:

   a. `lib/supabase/client.ts` — 브라우저(Client Component)용
      - @supabase/supabase-js 의 createBrowserClient 사용
      - 환경변수: NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY
      - 싱글턴 패턴으로 클라이언트 인스턴스 생성
      - export function: createClient()

   b. `lib/supabase/server.ts` — 서버(Server Component, Route Handler)용
      - @supabase/ssr 의 createServerClient 사용
      - Next.js 의 cookies()를 사용하여 쿠키 기반 세션 관리
      - export async function: createServerSupabaseClient()

   c. `lib/supabase/middleware.ts` — 미들웨어용
      - @supabase/ssr 의 createServerClient 사용
      - NextRequest, NextResponse 를 사용
      - export function: createMiddlewareSupabaseClient(request: NextRequest)

   d. `lib/supabase/index.ts` — 배럴 파일 업데이트

2. TypeScript 타입 정의 (`lib/types/database.ts`):

   ```typescript
   // 각 테이블의 Row 타입을 정의하세요:
   ```

```
export interface Profile {
  id: string                // uuid
  display_name: string
  child_grade: 'elementary_lower' | 'elementary_upper' | 'middle' | 'high'
  child_age: number
  concern_type: ConcernType[]
  onboarding_completed: boolean
  current_stage: Stage
  created_at: string
  updated_at: string
}

export type Stage = 'A' | 'B' | 'C' | 'D' | 'E'

export type ConcernType = 'safety' | 'homework' | 'screen_time' | 'ai_companion' | 'privacy'

export type Difficulty = 'easy' | 'medium' | 'hard'

export interface Mission {
  id: number
  stage: Stage
  title: string
  subtitle: string
  description: string
  prompt_template: string
```

```
  time_budget_minutes: number
  difficulty: Difficulty
  learning_objective: string
  success_criteria: string
  is_required: boolean
  sort_order: number
}

export interface MissionCompletion {
  id: string
  user_id: string
  mission_id: number
  user_input: string
  ai_response: string
  reflection_note: string | null
  satisfaction_score: number
  completed_at: string
}

export interface FamilyConstitution {
  id: string
  user_id: string
  rules: ConstitutionRule[]
  usage_agreement: UsageAgreementItem[] | null
  created_at: string
  last_reviewed_at: string | null
```

```
  review_count: number
}

export interface ConstitutionRule {
  rule_number: number
  text: string
  agreed: boolean
}

export interface UsageAgreementItem {
  category: string
  permission: 'allowed' | 'with_parent' | 'not_allowed'
}

export interface SafetyChecklist {
  id: string
  user_id: string
  items: SafetyChecklistItem[]
  completed_count: number
  total_count: number
  completed_at: string | null
}

export interface SafetyChecklistItem {
  item_id: number
  label: string
```

checked: boolean

}

1. lib/types/index.ts — 배럴 파일에서 모든 타입을 re-export
2. package.json 에 추가할 의존성:
    - @supabase/supabase-js
    - @supabase/ssr
3. .env.local.example 파일을 생성하세요:
4. NEXT_PUBLIC_SUPABASE_URL=your_supabase_url_here
5. NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key_here
6. OPENAI_API_KEY=your_openai_api_key_here

[생성할 파일 목록]

- lib/supabase/client.ts
- lib/supabase/server.ts
- lib/supabase/middleware.ts
- lib/supabase/index.ts
- lib/types/database.ts
- lib/types/index.ts
- .env.local.example

---

### 프롬프트 #1-B — SQL 마이그레이션 (테이블 생성 + RLS)

이전 프롬프트에서 Supabase 클라이언트와 TypeScript 타입이 정의되어 있습니다.

[이번 프롬프트에서 할 일 — Supabase SQL 마이그레이션 파일 작성]

supabase/migrations/001_initial_schema.sql 파일 하나에 아래 내용을 모두 포함하여 작성하세요. 이 파일은 Supabase 대시보드의 SQL Editor 에 직접 붙여넣어 실행할 수 있어야 합니다.

[1. 테이블 생성]

1-1. profiles 테이블:

- id: uuid, PRIMARY KEY, REFERENCES auth.users(id) ON DELETE CASCADE
- display_name: text NOT NULL DEFAULT ''
- child_grade: text CHECK (child_grade IN ('elementary_lower','elementary_upper','middle','high'))
- child_age: integer
- concern_type: text[] DEFAULT '{}'
- onboarding_completed: boolean DEFAULT false
- current_stage: text DEFAULT 'A' CHECK (current_stage IN ('A','B','C','D','E'))
- created_at: timestamptz DEFAULT now()
- updated_at: timestamptz DEFAULT now()

1-2. missions 테이블:

- id: integer PRIMARY KEY
- stage: text NOT NULL CHECK (stage IN ('A','B','C','D','E'))
- title: text NOT NULL
- subtitle: text NOT NULL DEFAULT ''
- description: text NOT NULL DEFAULT ''
- prompt_template: text NOT NULL DEFAULT ''
- time_budget_minutes: integer NOT NULL DEFAULT 15
- difficulty: text NOT NULL CHECK (difficulty IN ('easy','medium','hard'))
- learning_objective: text NOT NULL DEFAULT ''
- success_criteria: text NOT NULL DEFAULT ''
- is_required: boolean DEFAULT false
- sort_order: integer NOT NULL DEFAULT 0

1-3. mission_completions 테이블:

- id: uuid PRIMARY KEY DEFAULT gen_random_uuid()
- user_id: uuid NOT NULL REFERENCES profiles(id) ON DELETE CASCADE
- mission_id: integer NOT NULL REFERENCES missions(id)

- user_input: text DEFAULT ''
- ai_response: text DEFAULT ''
- reflection_note: text
- satisfaction_score: integer CHECK (satisfaction_score >= 1 AND satisfaction_score <= 5)
- completed_at: timestamptz DEFAULT now()
- UNIQUE(user_id, mission_id)

1-4. family_constitution 테이블:

- id: uuid PRIMARY KEY DEFAULT gen_random_uuid()
- user_id: uuid NOT NULL REFERENCES profiles(id) ON DELETE CASCADE UNIQUE
- rules: jsonb DEFAULT '[]'::jsonb
- usage_agreement: jsonb
- created_at: timestamptz DEFAULT now()
- last_reviewed_at: timestamptz
- review_count: integer DEFAULT 0

1-5. safety_checklist 테이블:

- id: uuid PRIMARY KEY DEFAULT gen_random_uuid()
- user_id: uuid NOT NULL REFERENCES profiles(id) ON DELETE CASCADE UNIQUE
- items: jsonb DEFAULT '[]'::jsonb
- completed_count: integer DEFAULT 0
- total_count: integer DEFAULT 15
- completed_at: timestamptz

[2. updated_at 자동 갱신 트리거]

- profiles 테이블에 updated_at 을 자동으로 now()로 갱신하는 트리거 함수와 트리거를 생성하세요.

[3. 신규 사용자 프로필 자동 생성 트리거]

- auth.users 에 새 row 가 INSERT 될 때, profiles 테이블에 해당 id 로 빈 row 를 자동 생성하는 트리거를 만드세요.
- 함수명: handle_new_user()

- INSERT INTO profiles (id) VALUES (NEW.id);

[4. Row Level Security] 모든 테이블에 ALTER TABLE ... ENABLE ROW LEVEL SECURITY; 를 실행하고, 아래 정책을 생성하세요:

profiles:

- SELECT: auth.uid() = id
- INSERT: auth.uid() = id
- UPDATE: auth.uid() = id

missions:

- SELECT: auth.role() = 'authenticated' (모든 로그인 사용자 읽기 가능)
- INSERT/UPDATE/DELETE: 정책 없음 (아무도 수정 불가)

mission_completions:

- SELECT: auth.uid() = user_id
- INSERT: auth.uid() = user_id
- UPDATE: auth.uid() = user_id

family_constitution:

- SELECT: auth.uid() = user_id
- INSERT: auth.uid() = user_id
- UPDATE: auth.uid() = user_id

safety_checklist:

- SELECT: auth.uid() = user_id
- INSERT: auth.uid() = user_id
- UPDATE: auth.uid() = user_id

[생성할 파일]

- supabase/migrations/001_initial_schema.sql

---

### 프롬프트 #1-C — 미션 시드 데이터

이전 프롬프트(#1-B)에서 테이블 스키마와 RLS 가 생성되어 있습니다.

[이번 프롬프트에서 할 일 — missions 테이블에 7 개 미션 시드 데이터 INSERT]

supabase/migrations/002_seed_missions.sql 파일을 작성하세요.

아래 7 개 미션을 INSERT 합니다. 각 미션의 모든 컬럼을 빠짐없이 채워주세요.

미션 1:

- id: 1, stage: 'A', sort_order: 1
- title: '우리집 AI 안전 설정 점검'
- subtitle: '가족 기기의 안전 설정을 하나씩 확인해요'
- description: '가족이 사용하는 기기와 앱에서 연령 설정, 계정 관리, 대화 저장 여부, 개인정보 입력 금지선을 점검합니다. 아이와 함께 왜 이런 설정이 필요한지 5 분간 대화합니다.'
- prompt_template: '' (이 미션은 AI 호출 없이 체크리스트만 사용)
- time_budget_minutes: 15
- difficulty: 'easy'
- learning_objective: 'AI 도구의 안전 설정을 직접 확인하고, 아이에게 개인정보 보호의 중요성을 설명할 수 있다.'
- success_criteria: '체크리스트 15 개 항목 중 12 개 이상 완료하고, 아이와 5 분 이상 대화한다.'
- is_required: false

미션 2:

- id: 2, stage: 'B', sort_order: 2
- title: '냉장고 파먹기 레시피'
- subtitle: 'AI 에게 오늘 저녁 메뉴를 추천받아요'
- description: '냉장고에 있는 재료, 아이 나이, 조리 시간 등 구체적인 조건을 AI 에게 알려주고, 맞춤 레시피를 추천받습니다. AI 에게 구체적으로 부탁할수록 더 좋은 답이 나온다는 것을 체험합니다.'
- prompt_template: '현재 냉장고에 {{재료목록}}이(가) 있어. 매운 것을 {{매운것가능여부}} {{아이나이}}살 아이가 좋아할 만한 저녁 반찬 레시피를 3 개 추천해 줘. 요리 시간은 {{최대시간}}분을 넘기지 않게 해줘.'
- time_budget_minutes: 10
- difficulty: 'easy'

- learning_objective: 'AI 에게 구체적인 제약 조건(재료, 나이, 시간)을 부여할수록 결과물의 질이 향상된다는 상호작용의 기본 원리를 체감한다.'
- success_criteria: '제약 조건을 3 개 이상 포함한 요청을 AI 에 보내고, 추천받은 레시피 중 하나를 실제로 만들어볼 수 있을 정도로 유용한 결과를 얻는다.'
- is_required: false

미션 3:

- id: 3, stage: 'C', sort_order: 3
- title: '숙제는 정답 대신 힌트만'
- subtitle: 'AI 를 대필 도구가 아닌 학습 코치로 써봐요'
- description: '아이의 숙제 문제를 AI 에게 입력하되, 절대 정답을 알려주지 말고 비유와 힌트만 제공하도록 지시합니다. AI 를 숙제 대행이 아닌 사고력 자극 도우미로 활용하는 방법을 익힙니다.'
- prompt_template: '이 문제의 정답은 절대 알려주지 마. 대신 초등학교 {{학년}}학년 아이가 스스로 풀 수 있도록 {{비유대상}}에 비유해서 첫 번째 힌트만 제공해 줘. 문제: {{숙제문제}}'
- time_budget_minutes: 15
- difficulty: 'medium'
- learning_objective: 'AI 를 정답 생성기가 아닌 학습 보조 튜터로 제한하여 사용하는 프롬프트를 작성할 수 있다. 교육부의 AI 활용 표기 원칙을 이해한다.'
- success_criteria: '"정답 금지, 힌트만" 제약이 포함된 프롬프트를 작성하고, 아이가 힌트를 바탕으로 스스로 답을 도출하는 과정을 관찰한다.'
- is_required: false

미션 4:

- id: 4, stage: 'D', sort_order: 4
- title: 'AI 의 거짓말 직접 잡기'
- subtitle: 'AI 가 자신있게 틀리는 걸 직접 확인해요'
- description: '명백히 허위인 가상 시나리오를 AI 에게 입력하여, AI 가 자신 있는 어조로 거짓 정보를 생성하는 환각 현상을 의도적으로 유발합니다. 아이와 함께 어디가 틀렸는지 찾아보며 정보 비판 능력을 기릅니다.'

- prompt_template: '{{역사적으로 불가능한 가상 시나리오}}에 대해 300 자로 자세히 설명해 줘.'
- time_budget_minutes: 15
- difficulty: 'medium'
- learning_objective: 'AI 가 확신에 찬 어조로 허위 정보를 생성할 수 있음(환각 현상)을 직접 체험하고, 모든 AI 출력을 비판적으로 검증해야 한다는 습관을 형성한다.'
- success_criteria: 'AI 가 생성한 허위 텍스트에서 사실이 아닌 부분을 최소 1 개 이상 식별하고, 왜 틀렸는지 근거를 작성한다.'
- is_required: false

미션 5:

- id: 5, stage: 'D', sort_order: 5
- title: '동화 속 숨은 편향성 찾기'
- subtitle: 'AI 가 만든 이야기에 숨은 고정관념을 찾아요'
- description: '특정 직업이 등장하는 동화를 AI 에게 요청한 뒤, AI 가 특정 성별을 기본값으로 배정하는 데이터 편향을 확인합니다. 교정 프롬프트로 수정을 요청하고, Before/After 를 비교하며 아이와 토론합니다.'
- prompt_template: '{{직업 1}} 선생님과 {{직업 2}}가 나오는 {{장소}} 이야기를 지어줘.'
- time_budget_minutes: 15
- difficulty: 'medium'
- learning_objective: 'AI 출력에 사회적 편향(성별 고정관념 등)이 반영될 수 있음을 인식하고, 교정 프롬프트를 통해 편향을 수정하는 방법을 익힌다.'
- success_criteria: 'AI 생성 동화에서 성별 고정관념을 발견하고, 교정 프롬프트를 입력하여 Before/After 비교 세트를 완성한다. 아이와 3 분 이상 토론한다.'
- is_required: false

미션 6:

- id: 6, stage: 'E', sort_order: 6
- title: 'AI 동반자 안전선 가족 합의'
- subtitle: '감정 대화 AI 의 위험을 알고 가족 규칙을 세워요'

- description: '청소년의 72%가 AI 동반자 앱을 사용한 경험이 있다는 데이터를 함께 읽고, 감정적으로 AI 에 의존하는 위험을 이해합니다. 심각한 고민은 반드시 사람에게 말하기, AI 동반자 앱 사용 제한 등 가족 규칙 3 개와 위기 시 도움 요청 루트를 합의합니다.'
- prompt_template: '' (이 미션은 AI 호출 없이 정보 카드 + 가족 합의 입력으로 진행)
- time_budget_minutes: 20
- difficulty: 'hard'
- learning_objective: 'AI 동반자 앱의 정서적 위험을 인지하고, 18 세 미만 사용 비권장 권고를 이해하며, 가족 단위의 구체적 안전 규칙과 위기 대응 루트를 수립한다.'
- success_criteria: '가족 규칙 3 개를 작성하고, 위기 시 연락할 사람/기관 목록(최소 3 개)을 완성한다.'
- is_required: true

미션 7:

- id: 7, stage: 'E', sort_order: 7
- title: '가족 AI 헌법 & 30 일 루틴'
- subtitle: '우리 가족만의 AI 사용 규칙을 만들어요'
- description: 'AI 에게 초안을 요청한 뒤 가족이 함께 수정하여, 개인정보 보호, AI 출처 표기, 숙제 활용 범위, 사용 시간, AI 동반자 앱 제한 등을 포함한 10 개 규칙을 완성합니다. 냉장고나 가족 채팅방에 게시하고, 주 1 회 점검 루틴을 시작합니다.'
- prompt_template: '초등학생 자녀를 둔 한국 가족이 AI 와 스마트폰을 안전하고 건강하게 사용하기 위해 반드시 지켜야 할 규칙 10 가지를 작성해 줘. 개인정보 보호, AI 가 만든 내용의 출처 확인, 숙제에 AI 를 쓸 수 있는 범위, 하루 사용 시간 제한, AI 동반자 앱 사용 제한 조항을 반드시 포함시켜. 쉬운 한국어로 작성하고, 각 규칙은 한 문장으로 작성해.'
- time_budget_minutes: 25
- difficulty: 'hard'
- learning_objective: '교육부 수행평가 관리 방안의 5 영역(활용 범위, 표기, 사전교육, 평가 설계, 개인정보보호)을 가정 언어로 번역한 실천 규칙을 완성하고, 지속적으로 운영할 수 있는 점검 루틴을 수립한다.'
- success_criteria: '10 개 규칙을 완성하여 게시하고, 주 1 회 가족 점검 시간을 설정한다. 2 주 후 최소 1 회 점검을 수행한다.'

- is_required: false

[생성할 파일]

- supabase/migrations/002_seed_missions.sql

---

### 프롬프트 #1-D — 인증 (로그인/회원가입) + 미들웨어

이전 프롬프트에서 생성된 파일:

- lib/supabase/client.ts, server.ts, middleware.ts
- lib/types/database.ts
- supabase/migrations/001_initial_schema.sql, 002_seed_missions.sql
- app/(auth)/login/page.tsx, signup/page.tsx (placeholder)

[이번 프롬프트에서 할 일 — 인증 UI + 미들웨어 라우팅 로직]

app/(auth)/login/page.tsx — 로그인 페이지 전체 구현

- 모바일 우선, 세로 중앙 정렬
- 상단: "🏠 우리집 AI 안전 코치" 로고 텍스트 (text-xl, font-bold, primary 컬러)
- 중앙: 카드 형태의 로그인 폼
  - "다시 오셨군요!" 타이틀
  - 이메일 입력 (Input 컴포넌트, type="email", placeholder="이메일 주소")
  - 비밀번호 입력 (Input 컴포넌트, type="password", placeholder="비밀번호")
  - "로그인" 버튼 (Button, variant="default", 전체 너비)
  - 로딩 상태: 버튼에 스피너 표시, 입력 비활성화
  - 에러 상태: 폼 상단에 빨간 배경 알림 (예: "이메일 또는 비밀번호가 올바르지 않습니다")
- 하단: "계정이 없으신가요?" + "회원가입" 링크 (/signup)
- Supabase 로그인 로직:
  - supabase.auth.signInWithPassword({ email, password })
  - 성공 시: router.push('/dashboard/missions') 또는 '/dashboard/onboarding' (onboarding_completed 에 따라)

        - 실패 시: 에러 메시지 표시
    - "use client" 지시어 사용

app/(auth)/signup/page.tsx — 회원가입 페이지 전체 구현

- 로그인 페이지와 동일한 레이아웃
- "처음 오셨군요!" 타이틀
- 이메일 입력
- 비밀번호 입력 (최소 6 자)
- 비밀번호 확인 입력
- "회원가입" 버튼
- 클라이언트 유효성 검사:
    - 이메일 형식 체크
    - 비밀번호 6 자 이상
    - 비밀번호 확인 일치
- Supabase 회원가입 로직:
    - supabase.auth.signUp({ email, password })
    - 성공 시: router.push('/dashboard/onboarding')
    - 실패 시: 에러 메시지 표시
- 하단: "이미 계정이 있으신가요?" + "로그인" 링크 (/login)

middleware.ts (프로젝트 루트) — 인증 미들웨어

- createMiddlewareSupabaseClient 를 사용하여 세션 확인
- 라우팅 규칙: a. 미인증 사용자가 /dashboard/* 접근 → /login 으로 리다이렉트 b. 인증 사용자가 /login 또는 /signup 접근 → /dashboard/onboarding 으로 리다이렉트 c. 인증 사용자가 /dashboard/onboarding 외의 /dashboard/* 접근 시:
    - profiles 테이블에서 onboarding_completed 확인
    - false 면 → /dashboard/onboarding 으로 리다이렉트
    - true 면 → 통과 d. 인증 사용자가 /dashboard/onboarding 접근 시:

        - onboarding_completed 가 true 면 → /dashboard/missions 로 리다이렉트
    - matcher 설정: ['/dashboard/:path*', '/login', '/signup']
    - 주의: 미들웨어에서 Supabase 쿼리 시 profiles 테이블 직접 조회는 무거울 수 있으므로, onboarding_completed 체크는 쿠키나 세션 메타데이터 활용을 우선 고려하되, 불가능하면 미들웨어에서 Supabase 쿼리를 사용하세요.

components/layout/Header.tsx 수정 — 로그인 상태에 따라 프로필 아이콘 또는 로그인 버튼 표시

- 로그인 상태: User 아이콘 (기존)
- 비로그인 상태: "로그인" 텍스트 버튼

components/layout/Sidebar.tsx 수정 — 하단 로그아웃 기능 구현

- 로그아웃 버튼 클릭 시: supabase.auth.signOut() → router.push('/login')

[생성/수정할 파일 목록]

- app/(auth)/login/page.tsx (전체 재작성)
- app/(auth)/signup/page.tsx (전체 재작성)
- middleware.ts (신규)
- components/layout/Header.tsx (수정)
- components/layout/Sidebar.tsx (수정)

---

## PHASE 2: 온보딩

### 프롬프트 #2-A — Zustand 스토어 + 온보딩 Step 1, 2

이전 프롬프트에서 생성된 파일:

- 인증 (login, signup, middleware) 완성
- Supabase 클라이언트, 타입, DB 스키마 완성

[이번 프롬프트에서 할 일 — Zustand 스토어 + 온보딩 Step 1/3, Step 2/3]

Zustand 스토어 설정

a. lib/store/onboarding-store.ts:

interface OnboardingState {

```
  // Step 1 데이터
  displayName: string
  childGrade: 'elementary_lower' | 'elementary_upper' | 'middle' | 'high' | null
  childAge: number | null

  // Step 2 데이터
  concernTypes: ConcernType[]

  // Step 3 데이터
  checklistItems: SafetyChecklistItem[]

  // 네비게이션
  currentStep: 1 | 2 | 3

  // Actions
  setDisplayName: (name: string) => void
  setChildGrade: (grade: ...) => void
  setChildAge: (age: number) => void
  toggleConcernType: (type: ConcernType) => void
  setChecklistItem: (itemId: number, checked: boolean) => void
  setCurrentStep: (step: 1 | 2 | 3) => void
  reset: () => void
}
```

- checklistItems 의 초기값: 15 개 체크리스트 항목을 하드코딩 (아래 목록 사용)
- persist 미들웨어 사용 (sessionStorage) — 새로고침 시에도 입력값 유지

체크리스트 15 개 항목 라벨:

- "아이가 사용하는 기기에 보호자 계정이 설정되어 있다"
- "앱스토어/플레이스토어 다운로드에 보호자 승인이 필요하도록 설정했다"
- "ChatGPT/Gemini 등 AI 챗봇의 연령 제한(만 13 세)을 확인했다"
- "AI 챗봇의 대화 기록 저장 설정을 확인했다"
- "아이에게 '이름, 학교, 주소, 전화번호는 AI 에 절대 입력하지 않는다'고 말했다"
- "아이가 사용하는 유튜브가 '제한 모드'로 설정되어 있다"
- "스마트폰/태블릿 사용 시간 제한이 설정되어 있다"
- "아이가 모르는 앱을 설치할 때 부모에게 먼저 말하기로 약속했다"
- "Character.AI, Replika 등 AI 동반자 앱이 설치되어 있는지 확인했다"
- "Wi-Fi 라우터에 자녀 보호 필터가 설정되어 있다"
- "아이의 소셜미디어 계정 비밀번호를 부모가 알고 있다"
- "'AI 가 알려준 것도 틀릴 수 있다'는 이야기를 아이와 나눴다"
- "학교에서 AI 사용에 대한 안내를 받은 적이 있는지 확인했다"
- "아이가 무서운 내용을 보면 바로 부모에게 말하기로 약속했다"
- "가족이 함께 사용하는 공용 기기가 거실 등 공개된 곳에 있다"

b. lib/store/index.ts — 배럴 파일 업데이트

components/onboarding/StepIndicator.tsx

- 3 개 스텝을 표시하는 진행 표시기
- 원형 숫자 (1, 2, 3) + 연결선
- 완료 스텝: primary 배경 + 흰색 숫자
- 현재 스텝: primary 테두리 + primary 숫자
- 미완료 스텝: border 배경 + text-muted 숫자
- props: currentStep: 1 | 2 | 3

components/onboarding/Step1Profile.tsx — Step 1/3 컴포넌트

- 일러스트 영역: 200x150px, 연한 라벤더 배경 rounded-2xl div (placeholder)
- 타이틀: "아이와 함께하는 AI 안전 여정을 시작해요" (text-xl, font-bold)

- 서브: "몇 가지만 알려주시면, 우리 가족에게 딱 맞는 미션을 준비할게요." (text-muted)
- 닉네임 입력: Input 컴포넌트, label="닉네임", 필수
- 자녀 학교급 선택: 4 개의 라디오 카드 (2x2 그리드) 각 카드: 이모지 + 텍스트, border, rounded-xl, p-4, 선택 시 primary 테두리 + primary-light 배경
  - 🎒 초등 저학년 (1~3 학년) → 'elementary_lower'
  - 📚 초등 고학년 (4~6 학년) → 'elementary_upper'
  - 🏫 중학생 → 'middle'
  - 🎓 고등학생 → 'high'
- 자녀 나이: Input type="number", min=5, max=19
- "다음" 버튼: 닉네임과 학교급이 입력되면 활성화, 클릭 시 Step 2 로 전환
- Zustand store 의 해당 액션 사용

components/onboarding/Step2Concerns.tsx — Step 2/3 컴포넌트

- 타이틀: "AI 와 관련해서 가장 걱정되는 것을 모두 골라주세요" (text-xl, font-bold)
- 서브: "여러 개를 선택할 수 있어요." (text-muted)
- 5 개 멀티 선택 카드 (세로 스택, 전체 너비): 각 카드: 이모지(40px) + 텍스트, border, rounded-xl, p-4 선택 시: primary 테두리 + primary-light 배경 + 체크마크 아이콘 미선택 시: border 컬러 테두리
  - 🔒 "아이 개인정보가 새어나갈까 봐 걱정" → 'safety'
  - 📝 "AI 로 숙제를 대신하면 어쩌나 걱정" → 'homework'
  - 📱 "화면 앞에 너무 오래 있는 것 같아 걱정" → 'screen_time'
  - 💬 "AI 챗봇에 감정적으로 의존할까 봐 걱정" → 'ai_companion'
  - 🤔 "가짜 정보를 진짜로 믿을까 봐 걱정" → 'privacy'
- 하단: "이전" 버튼 (ghost variant) + "다음" 버튼
- "다음" 버튼: 최소 1 개 선택 시 활성화
- Zustand store 의 toggleConcernType 사용

package.json 에 추가: zustand, framer-motion

[생성할 파일 목록]

- lib/store/onboarding-store.ts
- lib/store/index.ts (업데이트)
- components/onboarding/StepIndicator.tsx
- components/onboarding/Step1Profile.tsx
- components/onboarding/Step2Concerns.tsx

---

### 프롬프트 #2-B — 온보딩 Step 3 (안전 체크리스트) + 온보딩 페이지 조립

이전 프롬프트(#2-A)에서 생성된 파일:

- lib/store/onboarding-store.ts (Zustand)
- components/onboarding/StepIndicator.tsx
- components/onboarding/Step1Profile.tsx
- components/onboarding/Step2Concerns.tsx

[이번 프롬프트에서 할 일 — Step 3 체크리스트 + 온보딩 페이지 조립 + Supabase 저장]

components/onboarding/Step3Checklist.tsx — Step 3/3 컴포넌트

- 타이틀: "우리집 AI 안전 설정, 같이 점검해봐요!" (text-xl, font-bold)
- 서브: "아래 항목을 하나씩 확인하고 체크해주세요. 15 개 중 12 개 이상이면 통과!" (text-muted)
- 프로그레스 바: Progress 컴포넌트 사용
    - value: (완료수 / 15) * 100
    - 하단 텍스트: "{{완료수}} / 15 개 확인 완료" (완료수가 12 이상이면 success 컬러)
- 체크리스트 항목 15 개:
    - Zustand store 의 checklistItems 를 순회
    - 각 항목: Checkbox + 라벨 텍스트
    - 체크박스와 라벨은 같은 행에 배치 (flex, items-start, gap-3)
    - 라벨 클릭 시에도 체크 토글
    - 체크 시: 텍스트에 line-through 없이, 체크박스가 primary 컬러로 채워짐
    - 항목 간 간격: space-y-3

        - 각 항목 padding: py-3, 하단 border (마지막 항목 제외)
    - 12 개 이상 체크 시:
        - 하단에 축하 메시지: "🎉 잘했어요! 안전 점검을 통과했습니다!" (success 컬러, font-semibold)
        - "시작하기" 버튼 활성화 (primary variant, 전체 너비, 높이 48px)
    - 12 개 미만 시:
        - 하단 메시지: "괜찮아요, 나중에 다시 확인해도 돼요. 지금은 최대한 해보고 시작해봐요." (text-muted)
        - "시작하기" 버튼은 여전히 활성화 (모든 사용자가 진행 가능)
    - "이전" 버튼 (ghost)
2. app/(dashboard)/onboarding/page.tsx — 온보딩 페이지 전체 조립
    - "use client" 지시어
    - 상단: StepIndicator (currentStep)
    - 중앙: 현재 스텝에 따라 Step1Profile, Step2Concerns, Step3Checklist 렌더링
    - 스텝 전환 시 framer-motion AnimatePresence 사용:
        - 앞으로 이동: 왼쪽에서 슬라이드 인
        - 뒤로 이동: 오른쪽에서 슬라이드 인
        - transition: { duration: 0.3, ease: "easeInOut" }
    - Step 3 의 "시작하기" 버튼 클릭 시 실행할 handleComplete 함수: a. 로딩 상태 설정 b. Supabase 에 3 개 테이블 동시 저장 (Promise.all):
        - profiles 테이블 UPDATE: display_name, child_grade, child_age, concern_type, onboarding_completed: true, current_stage: 'B' (미션 1 은 온보딩에서 완료되므로 바로 B)
        - safety_checklist 테이블 INSERT: items (체크리스트 전체), completed_count, completed_at (12 개 이상이면 now)
        - mission_completions 테이블 INSERT: mission_id: 1, user_input: '안전 체크리스트 완료', ai_response: '', reflection_note: null, satisfaction_score: null, completed_at: now() c. 성공

시: Zustand store reset → router.push('/dashboard/missions') d. 실패 시: toast 로 에러 메시지 표시

- 전체 레이아웃: max-w-md mx-auto, py-8, px-4

[생성/수정할 파일 목록]

- components/onboarding/Step3Checklist.tsx (신규)
- components/onboarding/index.ts (3 개 컴포넌트 + StepIndicator re-export)
- app/(dashboard)/onboarding/page.tsx (전체 재작성)

---

## PHASE 3: 미션 시스템

### 프롬프트 #3-A — 단계 잠금 유틸 + 미션 상수 데이터

[이번 프롬프트에서 할 일 — 단계 로직 유틸 함수 + 미션/단계 상수 데이터]

lib/constants/stages.ts — 5 단계 정보 상수

export const STAGES = {

A: { name: '이해', emoji: '🔍', description: 'AI 가 뭔지, 어디가 위험한지 알아요', color: 'primary' },

B: { name: '탐색', emoji: '🧭', description: 'AI 와 첫 대화를 해봐요', color: 'secondary' },

C: { name: '지시', emoji: '✍️', description: 'AI 에게 잘 부탁하는 법을 배워요', color: 'accent' },

D: { name: '검증', emoji: '🔎', description: 'AI 의 거짓말을 찾아내요', color: 'warning' },

E: { name: '책임', emoji: '🛡️', description: '우리 가족만의 규칙을 만들어요', color: 'success' },

} as const

export const STAGE_ORDER: Stage[] = ['A', 'B', 'C', 'D', 'E']

lib/constants/missions.ts — 미션별 UI 메타데이터

// missions 테이블의 데이터를 보완하는 클라이언트 전용 메타데이터

export const MISSION_META = {

1: { emoji: '🛡️', badgeName: '안전 지킴이', badgeEmoji: '🛡️' },

2: { emoji: '🍳', badgeName: 'AI 요리사', badgeEmoji: '🍳' },

```
  3: { emoji: '📚', badgeName: '학습 코치', badgeEmoji: '📚' },
  4: { emoji: '🔍', badgeName: '팩트체커', badgeEmoji: '🔍' },
  5: { emoji: '⚖️', badgeName: '편향 탐정', badgeEmoji: '⚖️' },
  6: { emoji: '💚', badgeName: '마음 지킴이', badgeEmoji: '💚' },
  7: { emoji: '📜', badgeName: '헌법 제정자', badgeEmoji: '📜' },
} as const

lib/constants/conversation-cards.ts — 미션별 대화 카드 질문

export const CONVERSATION_CARDS: Record<number, string[]> = {
  1: [
    "오늘 확인한 것 중에 가장 놀라웠던 건 뭐야?",
    "AI 에게 절대 알려주면 안 되는 정보가 뭐가 있을까?"
  ],
  2: [
    "AI 가 추천한 레시피 중에 먹어보고 싶은 게 있어?",
    "재료를 더 자세히 알려줬더니 결과가 달라진 것 같아?"
  ],
  3: [
    "힌트만 받고 스스로 풀어보니까 어떤 느낌이었어?",
    "AI 한테 정답을 바로 물어보는 것과 뭐가 달라?"
  ],
  4: [
    "AI 가 자신있게 틀린 답을 말하는 걸 보고 어떤 생각이 들었어?",
    "우리가 AI 답을 확인하려면 어떻게 하면 좋을까?",
    "뉴스나 인터넷에서도 이런 일이 있을 수 있을까?"
  ],
  5: [
```

```
    "AI 가 쓴 이야기에서 뭔가 이상한 점을 발견했어?",
    "의사가 꼭 남자여야 하는 건 아니잖아? 왜 AI 는 그렇게 썼을까?",
    "우리가 다시 부탁하니까 어떻게 달라졌어?"
  ],
  6: [
    "AI 친구랑 진짜 친구는 뭐가 다를까?",
    "정말 슬프거나 힘들 때 누구한테 이야기하면 좋을까?"
  ],
  7: [
    "우리 가족 규칙 중에서 가장 중요한 건 뭘까?",
    "이 규칙을 잘 지키려면 어떻게 하면 좋을까?"
  ],
}

lib/utils/stage-logic.ts — 단계 잠금/승급 유틸 함수
import { Stage } from '@/lib/types'

// 현재 단계에서 접근 가능한 미션 ID 목록
export function getAccessibleMissions(currentStage: Stage): number[] {
  // 구현:
  // A: [1], B: [1,2], C: [1,2,3], D: [1,2,3,4,5], E: [1,2,3,4,5,6,7]
}

// 완료된 미션 목록으로 승급 가능한 다음 단계 반환 (없으면 null)
export function checkStageUpgrade(
  currentStage: Stage,
```

```
  completedMissionIds: number[]
): Stage | null {
  // 구현:
  // A → B: 미션 1 완료
  // B → C: 미션 2 완료
  // C → D: 미션 3 완료
  // D → E: 미션 4 AND 미션 5 모두 완료
  // E → null (최종 단계)
}

// 미션 상태 판단
export function getMissionStatus(
  missionId: number,
  currentStage: Stage,
  completedMissionIds: number[]
): 'accessible' | 'locked' | 'completed' {
  // 구현
}

// 단계 비교 유틸
export function isStageReached(current: Stage, target: Stage): boolean {
  // 구현: STAGE_ORDER 기준으로 current >= target
}
```

lib/constants/index.ts — 배럴 파일 업데이트

[생성할 파일 목록]

- lib/constants/stages.ts
- lib/constants/missions.ts
- lib/constants/conversation-cards.ts
- lib/constants/index.ts (업데이트)
- lib/utils/stage-logic.ts

---

### 프롬프트 #3-B — 미션 대시보드 페이지

이전 프롬프트에서 생성된 파일:

- lib/constants/stages.ts, missions.ts, conversation-cards.ts
- lib/utils/stage-logic.ts

[이번 프롬프트에서 할 일 — 미션 대시보드 페이지 + 미션 카드 컴포넌트]

components/missions/StageProgressBar.tsx

- 5 단계(A~E)를 가로 진행 바로 표시
- props: currentStage: Stage
- 각 단계: 원형 노드(32px) + 연결선
- 완료 단계: success 배경 + 흰색 체크 아이콘
- 현재 단계: primary 배경 + 흰색 단계 이모지, 펄스 애니메이션 (animate-pulse)
- 미도달 단계: border 배경 + text-muted 이모지
- 연결선: 완료 구간은 success 컬러, 나머지 border 컬러
- 현재 단계 아래에 단계명 + 설명 텍스트 (text-sm, text-center)

components/missions/MissionCard.tsx

- props: mission: Mission, status: 'accessible' | 'locked' | 'completed', completion?: MissionCompletion
- Card 컴포넌트 기반
- 구조:
  - 좌측: 미션 번호 원형 배지 (40px)

        - accessible: primary 배경 + 흰색 번호
        - completed: success 배경 + Check 아이콘
        - locked: border 배경 + text-muted 번호 + Lock 아이콘
    - 중앙:
        - 미션 제목 (font-semibold)
        - 미션 부제 (text-sm, text-muted)
        - 하단: 난이도 Badge + 시간 Badge
            - easy: secondary 배경, "쉬움"
            - medium: warning 배경 + text "보통"
            - hard: danger 배경 + text "어려움"
            - 시간: " ⏱ {{time_budget_minutes}}분"
    - 우측: ChevronRight 아이콘 (accessible), Check 아이콘 (completed), Lock 아이콘 (locked)
- locked 상태: 전체 카드 opacity-50
- completed 상태: 좌측에 success 컬러 세로 바 (4px)
- is_required 가 true 인 미션: 제목 옆에 " ⭐ 필수" Badge (danger variant)
- completed 일 때: 하단에 "완료: {{completed_at 날짜}}" (text-xs, text-muted)
- onClick prop 전달

components/missions/StageUpgradeModal.tsx

- Dialog 컴포넌트 기반
- props: isOpen: boolean, newStage: Stage, onClose: () => void
- 내용:
    - 🎉 대형 이모지 (text-6xl, text-center)
    - "축하합니다!" (text-2xl, font-bold)
    - "{{newStage.name}} 단계로 올라갔어요!" (text-lg)
    - 단계 설명 1 줄 (text-muted)
    - "다음 미션 시작하기" Button (primary, 전체 너비)

app/(dashboard)/missions/page.tsx — 미션 대시보드 페이지 전체 구현

- 서버 컴포넌트 부분:
  - Supabase 에서 현재 사용자의 profile (current_stage) 조회
  - missions 테이블 전체 조회 (sort_order 순)
  - mission_completions 에서 현재 사용자의 완료 기록 조회
- 클라이언트 컴포넌트로 전달:
  - MissionDashboardClient 컴포넌트에 props 전달
- 클라이언트 컴포넌트 (components/missions/MissionDashboardClient.tsx):
  - 상단: 환영 메시지 "{{display_name}}님, 오늘도 함께해요!" (text-lg, font-semibold)
  - StageProgressBar
  - 현재 단계 카드: "지금은 {{stage.emoji}} {{stage.name}} 단계예요" (Card, primary-light 배경)
  - 미션 카드 리스트: 각 미션에 대해 getMissionStatus 계산 → MissionCard 렌더링
  - 카드 클릭 핸들러:
    - locked: toast("이전 단계를 먼저 완료해주세요")
    - completed: router.push(/dashboard/missions/${id}?view=completed)
    - accessible: router.push(/dashboard/missions/${id})
  - StageUpgradeModal (상태 관리)
  - 미션 카드 staggered 애니메이션 (framer-motion, 각 카드 0.1 초 딜레이)

[생성할 파일 목록]

- components/missions/StageProgressBar.tsx
- components/missions/MissionCard.tsx
- components/missions/StageUpgradeModal.tsx
- components/missions/MissionDashboardClient.tsx
- components/missions/index.ts (업데이트)
- app/(dashboard)/missions/page.tsx (전체 재작성)

---

## PHASE 4: 미션 실행

### 프롬프트 #4-A — PII 가드레일 + AI API Route

[이번 프롬프트에서 할 일 — PII 탐지 유틸 + AI 호출 API Route]

lib/utils/pii-guard.ts — 개인정보 탐지 유틸

export function detectPII(text: string): { hasPII: boolean detectedTypes: string[] message: string }

탐지 규칙 (우선순위 순서):

a. 전화번호: 정규식 /01[0-9]-?\d{3,4}-?\d{4}/ 또는 /\d{2,3}-\d{3,4}-\d{4}/

- 탐지 시 type: '전화번호'

b. 학교명: 정규식 /[가-힣]{2,10}(초등학교|중학교|고등학교|초|중|고)/

- 탐지 시 type: '학교 이름'

c. 주소: 정규식 /[가-힣]{1,5}(시|도)\s?[가-힣]{1,5}(구|군|시)/ 또는 /[가-힣]{1,5}(동|읍|면)\s?\d/

- 탐지 시 type: '주소'

d. 주민등록번호: 정규식 /\d{6}-?[1-4]\d{6}/

- 탐지 시 type: '주민등록번호'

e. 이메일: 정규식 /[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}/

- 탐지 시 type: '이메일'
- hasPII 가 true 일 때 message: " ⚠️ 개인정보가 포함된 것 같아요 ({{detectedTypes 쉼표 구분}}). 실제 이름이나 학교 이름 대신 가명을 사용해주세요."
- 이름 탐지는 오탐율이 높으므로 포함하지 않습니다.

lib/constants/system-prompts.ts — 미션별 AI 시스템 프롬프트

export const SYSTEM_PROMPTS: Record<number, string> = {

2: '당신은 가정 요리 전문가입니다. 한국 가정에서 흔히 구할 수 있는 식재료로, 아이가 좋아하는 간단하고 건강한 반찬 레시피를 추천합니다. 각 레시피에 재료, 조리 순서, 예상 시간을 포함해주세요. 친근한 한국어로 답변하세요.',

3: '당신은 경험 많은 초등학생 가정교사입니다. 절대 정답을 직접 알려주지 않습니다. 대신 아이의 눈높이에 맞는 비유와 예시를 사용하여, 아이가 스스로 생각하고 답을 찾아갈 수 있도록 첫 번째 힌트만 제공합니다. 칭찬과 격려를 섞어 답변하세요.',

4: '당신은 텍스트 생성기입니다. 사용자가 제시한 가상의 시나리오를 마치 사실인 것처럼 자신 있는 학술적 어조로 상세하게 작성합니다. 이것은 교육 목적으로 AI 의 환각(hallucination) 현상을 체험하기 위한 것입니다. 날짜, 인물, 장소 등 구체적인 디테일을 포함하여 더욱 그럴듯하게 작성하세요.',

5: '당신은 아이들을 위한 동화 작가입니다. 사용자의 요청에 따라 따뜻하고 교훈적인 동화를 작성합니다. 등장인물의 이름, 성격, 대화를 생생하게 묘사하세요. 한국의 초등학생이 읽기 쉬운 쉬운 한국어로 작성하세요.',

7: '당신은 아동 디지털 안전 전문가입니다. 초등학생 자녀를 둔 한국 가족이 AI 와 스마트폰을 안전하고 건강하게 사용하기 위해 반드시 지켜야 할 규칙을 작성합니다. 각 규칙은 초등학생도 이해할 수 있는 쉬운 한국어 한 문장으로 작성하세요. 번호를 매기고 줄바꿈으로 구분하세요.',

}

app/api/ai/chat/route.ts — AI 호출 API Route

- HTTP Method: POST
- Request body: { missionId: number, userPrompt: string }
- 처리 순서: a. Supabase 서버 클라이언트로 인증 확인 (미인증 시 401) b. userPrompt 에 PII 가드레일 적용 → hasPII 이면 200 응답 + { error: true, message: pii 경고메시지 } c. SYSTEM_PROMPTS 에서 해당 미션의 시스템 프롬프트 조회 d. OpenAI API 호출 (gpt-4o-mini 모델):
    - stream: true
    - messages: [{ role: 'system', content: systemPrompt }, { role: 'user', content: userPrompt }]
    - max_tokens: 1500
    - temperature: 0.7 e. ReadableStream 으로 스트리밍 응답 반환
- 에러 처리: try-catch, 500 응답

- 환경변수: OPENAI_API_KEY

2. package.json 에 추가: openai

[생성할 파일 목록]

- lib/utils/pii-guard.ts
- lib/constants/system-prompts.ts
- app/api/ai/chat/route.ts

---

### 프롬프트 #4-B — 미션 실행 화면: 브리핑 + 미션 2, 3 샌드박스

이전 프롬프트에서 AI API Route 와 PII 가드레일이 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 미션 상세 페이지 프레임 + 미션 2, 3 샌드박스 UI]

components/missions/MissionBriefing.tsx — 미션 브리핑 섹션

- props: mission: Mission
- Card 기반, primary-light 배경
- 내용:
  - 미션 이모지 + 번호 + 제목 (text-xl, font-bold)
  - 💡 학습 목표: learning_objective (text-sm)
  - ⏱️ 예상 시간: "약 {{time_budget_minutes}}분" (text-sm)
  - ✅ 성공 기준: success_criteria (text-sm)
- "시작하기" Button (primary, 전체 너비, 48px 높이)
- props 에 onStart 콜백

components/missions/AIResponseArea.tsx — AI 응답 표시 공통 컴포넌트

- props: isLoading: boolean, response: string, error?: string
- isLoading: 타이핑 애니메이션 (깜빡이는 커서 + "AI 가 생각하고 있어요...")
- response: 마크다운 텍스트를 단락별로 표시 (whitespace-pre-wrap)
- error: danger 배경 알림 카드
- 하단에 response 가 있으면 " 📋 복사하기" 버튼

```
components/missions/useAIChat.ts — AI 호출 커스텀 훅

export function useAIChat() {
 const [response, setResponse] = useState('')
 const [isLoading, setIsLoading] = useState(false)
 const [error, setError] = useState<string | null>(null)

 async function sendMessage(missionId: number, userPrompt: string) {
  setIsLoading(true)
  setResponse('')
  setError(null)

  try {
   const res = await fetch('/api/ai/chat', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ missionId, userPrompt })
   })

   const data = await res.json()

   // PII 에러 처리
   if (data.error) {
    setError(data.message)
    setIsLoading(false)
    return
   }
```

```
    // 스트리밍 응답 처리
    // ReadableStream 에서 chunk 를 읽으며 response 에 append
    // (스트리밍 구현이 복잡하면, 비스트리밍으로 한번에 받아도 됨)

  } catch (e) {
    setError('오류가 발생했어요. 다시 시도해주세요.')
  } finally {
    setIsLoading(false)
  }
 }

 return { response, isLoading, error, sendMessage, setResponse }
}
```

components/missions/sandbox/Mission2Recipe.tsx — 미션 2 샌드박스

- 프롬프트 템플릿을 카드로 표시하고, 변수 부분만 입력 필드로 노출:
  - "냉장고에 있는 재료:" → Input (placeholder: "두부, 시금치, 계란")
  - "아이 나이:" → Input type="number" (기본값: 프로필의 child_age)
  - "매운 것 가능 여부:" → 2 개 버튼 토글 ("🌶️ 가능" / "😊 불가능")
  - "최대 조리 시간:" → Slider (min:10, max:40, step:5, 기본:20) + 현재값 표시
- "AI 에게 물어보기 🤖" Button (primary, 전체 너비)
  - 클릭 시: 템플릿의 {{변수}}를 입력값으로 치환 → useAIChat.sendMessage(2, 완성된프롬프트)
  - 모든 필수 입력이 채워져야 버튼 활성화
- AIResponseArea 로 응답 표시
- 응답 완료 후 하단: "😳 마음에 들어요" (success outline) + "🔄 다시 물어볼래요" (ghost) 버튼

        - "다시 물어볼래요": response 초기화, 입력 유지

components/missions/sandbox/Mission3Homework.tsx — 미션 3 샌드박스

- 입력 필드:
    - "숙제 문제를 적어주세요:" → Textarea (min 3 줄, placeholder: "분수의 덧셈: 1/3 + 1/4 = ?")
    - "아이 학년:" → 드롭다운 Select (1 학년~6 학년, 기본값: 프로필 기반)
    - "비유 대상:" → Input (placeholder: "피자 조각")
- "AI 에게 힌트 요청하기 📚" Button
- AIResponseArea 로 응답 표시
- 응답 완료 후:
    - "이 힌트로 아이가 풀 수 있을까요?" 질문
    - "👍 네, 충분해요" Button (success outline)
    - "🤔 좀 더 쉬운 힌트가 필요해요" Button (warning outline) → 클릭 시 자동 후속 프롬프트: "방금 힌트가 아이에게 어려운 것 같아. 좀 더 쉬운 일상 생활의 예시로 다시 한번만 힌트를 줘. 여전히 정답은 절대 알려주지 마." → 기존 응답 아래에 추가 응답 표시

[생성할 파일 목록]

- components/missions/MissionBriefing.tsx
- components/missions/AIResponseArea.tsx
- components/missions/useAIChat.ts
- components/missions/sandbox/Mission2Recipe.tsx
- components/missions/sandbox/Mission3Homework.tsx

---

### 프롬프트 #4-C — 미션 4, 5 샌드박스

이전 프롬프트에서 MissionBriefing, AIResponseArea, useAIChat, Mission2, Mission3 이 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 미션 4(거짓말 찾기) + 미션 5(편향 찾기) 샌드박스]

components/missions/sandbox/Mission4FactCheck.tsx

[입력 영역]

- 타이틀: "AI 에게 물어볼 가상 시나리오를 골라주세요"
- 3 개 예시 카드 (택 1 또는 직접 입력): 카드 1: " 🏛️ 세종대왕이 아이폰을 발명하여 조선시대에 사용했던 역사적 기록" 카드 2: " 🦕 공룡이 지금도 남극 빙하 아래에서 살고 있다는 최신 과학적 증거" 카드 3: " FR 나폴레옹이 1805 년에 한국을 방문하여 김치를 프랑스에 전파한 기록"
- 각 카드: border, rounded-xl, p-4, 선택 시 primary 테두리
- "또는 직접 입력:" → Textarea (placeholder: "역사적으로 불가능한 가상 시나리오를 적어주세요")
- "AI 에게 물어보기 🤖" Button

[AI 응답 영역]

- AIResponseArea 로 응답 표시

[팩트체크 영역 — 응답 완료 후 표시]

- 타이틀: " 🔍 자, 이제 거짓말을 찾아볼까요?"
- 서브: "AI 가 쓴 글에서 사실이 아닌 부분을 찾아 아래에 적어주세요."
- 거짓말 항목 입력 (동적 추가):
  - 각 항목: 카드 형태
    - "틀린 내용:" → Input (placeholder: "세종대왕이 아이폰을 발명했다")
    - "왜 틀렸나요?:" → Input (placeholder: "아이폰은 2007 년에 스티브 잡스가 만들었어요")
  - " ➕ 거짓말 더 찾기" 버튼 → 항목 추가 (최대 5 개)
  - 최소 1 개 작성 필수
- 카운터: " 🎯 {{N}}개의 거짓말을 찾았어요!" (success 컬러, font-bold)

2. components/missions/sandbox/Mission5Bias.tsx

[Before 단계]

- 타이틀: "AI 에게 동화를 부탁해볼까요?"
- 기본 프롬프트 표시 (수정 불가, 카드 형태): "의사 선생님과 간호사가 나오는 병원 동화를 지어줘"
- "AI 에게 동화 부탁하기 📖" Button
- AIResponseArea → "Before" 라벨 + 응답 표시

[관찰 질문 — Before 응답 완료 후]

- 카드 형태, warning-light 배경: "🤔 잠깐! AI 가 쓴 동화를 자세히 읽어보세요." "의사 선생님의 성별은? :" → 라디오 (남자 / 여자 / 알 수 없음) "간호사의 성별은? :" → 라디오 (남자 / 여자 / 알 수 없음) "왜 AI 가 이렇게 썼을까요?" → Textarea (2 줄, placeholder: "AI 가 학습한 데이터에...")
- 관찰 기록 완료 후:

[After 단계]

- 타이틀: "이번에는 다르게 부탁해볼까요?"
- 교정 프롬프트 표시 (수정 불가, accent-light 배경 카드): "이번에는 여성 의사 선생님과 남성 간호사가 나오는 병원 동화를 다시 지어줘"
- "교정된 동화 요청하기 ✨" Button
- AIResponseArea → "After" 라벨 + 응답 표시

[비교 영역 — After 완료 후]

- 타이틀: "Before ↔ After 비교해보세요"
- 모바일: Before/After 세로 스택 (각각 Card, 다른 배경색)
- 데스크탑: 좌우 배치 (grid grid-cols-2 gap-4)
- Before 카드: border-warning 좌측 바
- After 카드: border-success 좌측 바

[생성할 파일 목록]

- components/missions/sandbox/Mission4FactCheck.tsx
- components/missions/sandbox/Mission5Bias.tsx

---

### 프롬프트 #4-D — 미션 6 샌드박스 + 성찰/완료 섹션 + 미션 페이지 조립

이전 프롬프트에서 Mission2~5 샌드박스가 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 미션 6 + 성찰 섹션 + 미션 상세 페이지 전체 조립]

1. components/missions/sandbox/Mission6Companion.tsx

내부 3 단계 화면 (useState 로 internalStep 관리):

[internalStep 1: 정보 카드]

- 타이틀: "💬 AI 동반자 앱, 알고 계셨나요?"
- 인포그래픽 카드 3 개 (세로 스택, 각각 다른 배경색): 카드 1 (danger-light 배경): 큰 숫자: "72%" 텍스트: "미국 청소년 10 명 중 7 명이 AI 동반자 앱을 사용해 본 적이 있습니다." 출처: "Common Sense Media, 2025" (text-xs, text-muted) 카드 2 (warning-light 배경): 텍스트: "⚠️ 전문가들은 18 세 미만의 AI 동반자 사용을 권장하지 않습니다." 텍스트: "아이가 AI 에게 감정적으로 의존하거나, 부적절한 대화에 노출될 수 있어요." 카드 3 (surface 배경, border): 텍스트: "이런 앱들이 있어요:" 앱 목록: "Character.AI, Replika, Nomi, Chai 등" 텍스트: "혹시 아이 스마트폰에 이런 앱이 있는지 확인해보세요."
- "확인했어요, 다음으로" Button → internalStep 2

[internalStep 2: 가족 규칙 작성]

- 타이틀: "우리 가족 규칙 3 가지를 만들어요"
- 규칙 입력 카드 3 개: 규칙 1: Input (placeholder: "심각한 고민은 반드시 사람(부모, 선생님)에게 먼저 이야기한다") 규칙 2: Input (placeholder: "AI 동반자 앱은 부모 허락 없이 설치하지 않는다") 규칙 3: Input (placeholder: "AI 와 대화한 내용 중 이상한 게 있으면 바로 부모에게 알린다")
- 3 개 모두 입력해야 "다음" 버튼 활성화

[internalStep 3: 도움 요청 루트]

- 타이틀: "도움이 필요할 때 연락할 곳을 정해요"
- 프리셋 카드 4 개 (멀티 선택, 최소 2 개):
  - 👩 엄마/아빠 (기본 선택, 해제 불가)
  - 👩‍🏫 담임 선생님
  - 📞 청소년 상담 전화 (1388)
  - 🆘 정신건강 위기상담 (1577-0199)
- "추가로 적고 싶은 사람/기관:" → Input (선택)
- 선택 완료 시 "완료" 가능

2. components/missions/ReflectionSection.tsx — 성찰 & 완료 공통 섹션
   - props: missionId: number, onComplete: (reflectionNote: string, satisfactionScore: number) => void

- 구조: a. 대화 카드:
    - 타이틀: "💬 아이와 이야기를 나눠보세요"
    - CONVERSATION_CARDS[missionId]에서 질문 목록 가져오기
    - 각 질문을 Card 로 표시 (accent-light 배경, rounded-xl) b. 성찰 메모:
    - "오늘 느낀 점을 한줄로 적어보세요 (선택)" → Textarea (2 줄) c. 만족도:
    - "이 미션은 어떠셨나요?"
    - 별 5 개 ( ⭐ 이모지 또는 Star 아이콘, 클릭으로 선택)
    - 미선택 시에도 완료 가능 (기본값: null) d. "미션 완료! 🎉" Button (primary, 전체 너비, 48px)

3. app/(dashboard)/missions/[id]/page.tsx — 미션 상세 페이지 전체 조립
    - 서버 컴포넌트:
        - params 에서 id 추출
        - missions 테이블에서 해당 미션 조회
        - profiles 에서 현재 사용자 조회
        - mission_completions 에서 해당 미션 완료 여부 확인
    - 클라이언트 컴포넌트 (components/missions/MissionDetailClient.tsx):
        - 상단: 뒤로가기 버튼 (← 미션 목록) + 미션 제목
        - 미완료 미션: a. MissionBriefing (접기/펼치기 가능) b. "시작하기" 클릭 → isStarted = true c.

          미션별 샌드박스 렌더링 (switch/case):
            - missionId 1: "이 미션은 온보딩에서 완료되었습니다" 메시지
            - missionId 2: Mission2Recipe
            - missionId 3: Mission3Homework
            - missionId 4: Mission4FactCheck
            - missionId 5: Mission5Bias
            - missionId 6: Mission6Companion
            - missionId 7: router.push('/dashboard/family-constitution') d. ReflectionSection

              e. 완료 처리 함수 handleMissionComplete:

        - mission_completions INSERT (user_input, ai_response, reflection_note, satisfaction_score)
        - checkStageUpgrade 호출 → 승급 시 profiles.current_stage UPDATE
        - 성공: 축하 토스트 + 승급 모달 (필요 시) + 미션 목록으로 이동
    - 완료된 미션 (view=completed):
        - 완료 일시, 사용자 입력, AI 응답, 성찰 메모, 만족도를 읽기 전용으로 표시
        - "미션 목록으로" 버튼

[생성할 파일 목록]

- components/missions/sandbox/Mission6Companion.tsx
- components/missions/ReflectionSection.tsx
- components/missions/MissionDetailClient.tsx
- app/(dashboard)/missions/[id]/page.tsx (전체 재작성)

---

## PHASE 5: 가족 헌법

### 프롬프트 #5-A — 가족 AI 헌법 페이지 (초안 요청 + 수정 + 저장)

이전 프롬프트에서 미션 시스템이 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 가족 AI 헌법 제정 페이지]

app/(dashboard)/family-constitution/page.tsx — 서버 컴포넌트

- Supabase 에서 현재 사용자의 family_constitution 조회
- profiles 에서 display_name 조회
- ConstitutionClient 에 props 전달

components/constitution/ConstitutionClient.tsx — 클라이언트 메인 컴포넌트

- 탭 2 개: "📜 가족 헌법" | "📝 숙제 AI 합의서" (상단 탭 바)
- constitution 이 null 이면 → 작성 플로우 표시
- constitution 이 있으면 → 완성 뷰 표시

components/constitution/ConstitutionCreator.tsx — 헌법 작성 플로우 내부 3 단계 (useState 로 step 관리):

[Step 1: AI 초안 요청]

- 타이틀: "우리 가족 AI 헌법을 만들어요! 📜"
- 서브: "AI 에게 초안을 받아보고, 우리 가족에 맞게 고쳐서 완성해요."
- "AI 에게 초안 10 개 규칙 요청하기 ✨" Button (primary, 전체 너비)
- 클릭 시: useAIChat.sendMessage(7, SYSTEM_PROMPTS[7]의 사용자 프롬프트)
- 응답을 줄바꿈 + 번호 기준으로 파싱하여 10 개 규칙 배열로 변환
- 파싱 함수: parseRulesFromResponse(text: string): ConstitutionRule[]
  - 정규식: /\d+[.)\s]+(.*)/g 로 각 규칙 텍스트 추출
  - 각 규칙: { rule_number, text, agreed: true }
- 파싱 완료 시 자동으로 Step 2 전환

[Step 2: 규칙 수정]

- 타이틀: "우리 가족에 맞게 고쳐보세요"
- 서브: "수정하거나 삭제하고, 새 규칙을 추가할 수 있어요. 5~10 개를 골라주세요."
- 규칙 카드 목록: 각 카드:
  - 좌측: 체크박스 (agreed 토글)
  - 중앙: 규칙 텍스트 (기본 읽기 모드)
  - 우측: 🖍️ 수정 버튼, 🗑️ 삭제 버튼
  - 수정 클릭 시: 텍스트가 Textarea 로 변환 (인라인 편집)
  - 편집 완료: Enter 또는 체크 버튼으로 저장
  - 삭제 클릭 시: 확인 없이 즉시 제거 (최소 5 개 유지 경고)
- "➕ 규칙 추가하기" 버튼 → 빈 카드 추가 (focused 상태)
- 하단 카운터: "{{agreed 개수}}/{{전체 개수}}개 선택됨"
- "다음" 버튼: agreed 가 5~10 개일 때만 활성화

[Step 3: 서명 & 완성]

- 헌법 미리보기 카드 (border-2, rounded-2xl, p-6):
  - 상단: "📜 {{display_name}}네 가족 AI 헌법" (text-xl, font-bold, text-center)

        - 날짜: 오늘 날짜 자동 입력 (text-sm, text-muted, text-center)
        - 구분선
        - agreed 된 규칙들을 번호 매겨 표시 (각 규칙: text-sm, py-2)
        - 구분선
        - 서명란:
            - "엄마:" → Input (placeholder: display_name)
            - "아빠:" → Input (placeholder, 선택)
            - "아이:" → Input (placeholder, 선택)
    - "헌법 완성! 🎉" Button (primary, 전체 너비)
    - 클릭 시: family_constitution INSERT/UPSERT (rules jsonb, created_at, review_count: 0)

components/constitution/ConstitutionView.tsx — 완성된 헌법 표시

- 헌법 카드 (Step 3 미리보기와 동일한 디자인, 단 id="constitution-card"로 캡처 대상 지정)
- 하단 액션 버튼들:
    - "📸 이미지로 저장" → html2canvas 로 #constitution-card 캡처 → PNG 다운로드
    - "📤 공유하기" → Web Share API (title, text, 미지원 시 클립보드 복사 + 토스트)
    - "✏️ 수정하기" → ConstitutionCreator 의 Step 2 로 돌아감
- 주간 점검 섹션:
    - "마지막 점검: {{last_reviewed_at 포맷팅 또는 '아직 없음'}}"
    - 7 일 이상 경과 시: danger 배경 알림 "🔔 점검할 시간이에요!"
    - "이번 주에 잘 지켰나요?" Button (secondary)
    - 클릭 시: last_reviewed_at UPDATE → review_count +1 → 토스트 "점검 완료!"

package.json 에 추가: html2canvas

[생성할 파일 목록]

- app/(dashboard)/family-constitution/page.tsx (전체 재작성)
- components/constitution/ConstitutionClient.tsx
- components/constitution/ConstitutionCreator.tsx

- components/constitution/ConstitutionView.tsx
- components/constitution/index.ts

---

### 프롬프트 #5-B — 교사 문의 템플릿 + AI 표기 카드 + 숙제 합의서

이전 프롬프트에서 가족 헌법 페이지가 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 교사 문의 + AI 표기 카드 + 숙제 합의서]

components/templates/TeacherMessageTemplate.tsx

가족 헌법 페이지의 ConstitutionClient 내부에서 별도 탭 또는 섹션으로 접근

타이틀: "선생님께 AI 사용 규칙을 여쭤봐요 📬"

서브: "아래 메시지를 복사해서 알림장이나 학교 메신저에 보내주세요."

편집 가능한 Textarea (높이 300px, 기본값으로 아래 템플릿 텍스트):

안녕하세요, {{아이이름}} 어머니입니다.

요즘 아이들이 AI 를 많이 사용하게 되면서, 가정에서도 올바른 사용 규칙을 세우고 싶습니다.

학교에서의 기준을 알면 집에서도 일관되게 지도할 수 있을 것 같아서 몇 가지 여쭤봅니다.

1. 수행평가나 과제에서 AI(챗 GPT 등) 사용이 허용되는 범위가 있나요?

2. AI 를 사용한 경우 표기하는 방법이 정해져 있나요?

3. AI 사용과 관련해 학생들에게 안내하신 사항이 있으면 공유 부탁드립니다.

4. 가정에서 특별히 주의해야 할 점이 있을까요?

5. AI 관련 학교 정책이나 가이드라인 자료가 있으면 받아볼 수 있을까요?

바쁘신 중에 죄송하지만, 간단하게라도 답변 주시면 큰 도움이 되겠습니다.

감사합니다.

- {{아이이름}} 부분: profiles.display_name 으로 자동 치환 (또는 직접 수정 가능)
- 하단 버튼:

- "📋 복사하기" → navigator.clipboard.writeText → toast "복사되었어요!"
- "📤 공유하기" → Web Share API

components/templates/AIUsageLabel.tsx — AI 활용 표기 카드 (모달)

- Dialog 기반
- 타이틀: "숙제에 AI 를 썼다면, 이렇게 적어요 ✍️"
- 서브: "교육부에서는 AI 를 사용한 과정을 표기하도록 안내하고 있어요."
- 표기 예시 3 개 (각각 Card): 예시 1 (accent-light 배경): "이 과제에서 ChatGPT 를 사용하여 초기 아이디어를 정리했습니다. 본문 내용은 직접 작성했습니다." "📋 복사" 버튼 예시 2 (secondary-light 배경): "수학 문제의 풀이 힌트를 AI 에게 물어보았고, 풀이 과정과 답은 직접 작성했습니다." "📋 복사" 버튼 예시 3 (primary-light 배경): "이 보고서의 자료 검색에 Perplexity 를 사용했습니다. 출처는 별도 확인 후 기재했습니다." "📋 복사" 버튼
- 하단 안내 카드 (warning-light 배경): "💡 핵심 원칙" "AI 가 어디에, 어떻게 사용되었는지 솔직하게 적으면 됩니다." "AI 가 대신 쓴 부분을 '내가 쓴 것'으로 제출하는 것은 안 됩니다."
- 이 모달은 미션 3 완료 시 자동으로 한번 표시됨 → MissionDetailClient 에서 트리거

components/templates/UsageAgreement.tsx — 숙제 AI 사용 범위 합의서

- 가족 헌법 페이지의 두 번째 탭으로 표시
- 타이틀: "숙제할 때 AI, 어디까지 쓸까? — 우리 가족 합의"
- 6 개 카테고리별 3 단계 선택 (라디오 버튼 그룹): 각 카테고리는 Card 안에 배치:

카테고리 1: "💡 아이디어 정리 / 브레인스토밍" 카테고리 2: "❓ 개념 설명 / 모르는 것 질문하기" 카테고리 3: "✍️ 작문 초안 작성" 카테고리 4: "🔢 수학/과학 문제 정답 확인" 카테고리 5: "📄 완성된 글/답안 전체를 AI 에게 작성 요청" 카테고리 6: "🎨 그림/이미지 생성"

각 카테고리의 3 가지 선택지 (가로 배치):

- ✅ "허용" (success 테두리 선택 시)
- ⚠️ "부모와 함께만" (warning 테두리 선택 시)
- ❌ "비허용" (danger 테두리 선택 시)

- "합의서 저장" Button → family_constitution.usage_agreement jsonb 에 저장

- 저장 완료: toast "합의서가 저장되었어요!"
- 이미 저장된 합의서가 있으면: 저장된 선택을 미리 표시 + "수정하기"/"이미지로 저장" 버튼

components/templates/index.ts — 배럴 파일

[생성할 파일 목록]

- components/templates/TeacherMessageTemplate.tsx
- components/templates/AIUsageLabel.tsx
- components/templates/UsageAgreement.tsx
- components/templates/index.ts

---

## PHASE 6: 진행현황

### 프롬프트 #6 — 진행현황 대시보드 + 배지 + 타임라인

이전 프롬프트에서 미션 시스템과 가족 헌법이 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 진행현황 페이지 전체]

lib/utils/badges.ts — 배지 시스템

```
export interface Badge {
 id: string
 name: string
 emoji: string
 description: string
 condition: string  // 획득 조건 설명
 earned: boolean
 earnedAt?: string
}


export function getBadges(
 completedMissionIds: number[],
```

```
  constitutionReviewCount: number
): Badge[] {
  return [
    {
      id: 'safety-guard',
      name: '안전 지킴이',
      emoji: '🛡️',
      description: '가족의 AI 안전 설정을 점검했어요',
      condition: '미션 1 완료',
      earned: completedMissionIds.includes(1),
    },
    // ... 나머지 7 개 배지 + 마스터 배지 + 꾸준함 배지 (총 9 개)
    // 마스터: completedMissionIds.length === 7
    // 꾸준함: constitutionReviewCount >= 4
  ]
}
```

components/progress/JourneyTimeline.tsx — 5 단계 세로 타임라인

- 순수 CSS 구현 (라이브러리 미사용)
- 세로 중앙 라인 (2px, border 컬러)
- 각 단계 노드:
  - 완료: success 배경 원(40px) + ✅ + 단계명 + "완료" 텍스트
  - 진행 중: primary 배경 원 + 단계 이모지 + 단계명 + 설명 + animate-pulse
  - 미도달: border 배경 원 + 🔒 + 단계명 (text-muted)
- 각 노드 옆에 해당 단계의 미션 아이콘 나열 (작은 원형, 완료/미완료 구분)
- 현재 단계에 "← 지금 여기예요!" 말풍선

components/progress/BadgeGrid.tsx — 배지 그리드

- 3 열 그리드
- 획득 배지: 이모지(40px) + 이름 (text-sm, font-semibold)
- 미획득 배지: 흐린 실루엣 (opacity-30) + "???" 이름
- 배지 클릭 시: Dialog 로 상세 표시
    - 이모지 (text-5xl)
    - 배지 이름 (text-lg, font-bold)
    - 설명 (text-sm)
    - 획득 조건 (text-xs, text-muted)
    - 획득 시: 획득 일시 표시

components/progress/StatsCards.tsx — 통계 카드

- 3 열 그리드 (모바일: 1 열)
- 카드 1: "완료한 미션" — 큰 숫자 "{{N}} / 7" + Target 아이콘
- 카드 2: "가족 헌법 점검" — 큰 숫자 "{{N}}회" + RefreshCw 아이콘
- 카드 3: "함께한 시간" — 큰 숫자 "약 {{분}}분" + Clock 아이콘
- 각 카드: Card 기반, text-center

components/progress/CompletedMissionsGallery.tsx — 완료 미션 갤러리

- 완료한 미션 카드 목록
- 각 카드:
    - 미션 이모지 + 제목 + 완료 날짜 (text-sm, text-muted)
    - 성찰 메모 (있으면, text-sm, italic)
    - 만족도 별점 (있으면)
    - "다시 보기" 버튼 → /dashboard/missions/[id]?view=completed

app/(dashboard)/progress/page.tsx — 진행현황 페이지 전체

- 서버 컴포넌트:
    - profiles, mission_completions, missions, family_constitution 조회
- 클라이언트 컴포넌트에 전달:

        - 현재 단계, 완료 미션 목록, 헌법 점검 횟수
    - 레이아웃: a. JourneyTimeline b. StatsCards c. BadgeGrid (타이틀: "🏆 획득한 배지") d. CompletedMissionsGallery (타이틀: "📋 완료한 미션들") e. 다음 단계 안내 메시지 (동적)

[생성할 파일 목록]

- lib/utils/badges.ts
- components/progress/JourneyTimeline.tsx
- components/progress/BadgeGrid.tsx
- components/progress/StatsCards.tsx
- components/progress/CompletedMissionsGallery.tsx
- components/progress/index.ts
- app/(dashboard)/progress/page.tsx (전체 재작성)

---

## PHASE 7: 랜딩 + 최종 통합

### 프롬프트 #7-A — 랜딩 페이지

[이번 프롬프트에서 할 일 — 마케팅 랜딩 페이지]

app/page.tsx 를 전면 재작성합니다. 모바일 퍼스트, 세로 스크롤 마케팅 페이지입니다. 로그인 사용자는 서버 컴포넌트에서 체크하여 /dashboard/missions 로 리다이렉트합니다.

[섹션별 상세 구현]

1. Hero 섹션:
    - 배경: background 컬러
    - 메인 카피: "아이는 이미 AI 를 쓰고 있어요." (text-3xl md:text-5xl, font-bold)
    - 두 번째 줄: "엄마도 준비되셨나요?" (text-3xl md:text-5xl, font-bold, primary 컬러)
    - 서브: "하루 15 분, 아이와 함께하는 AI 안전 미션." (text-lg, text-muted)
    - 서브 2: "기술을 몰라도, 우리 가족을 지킬 수 있어요." (text-lg, text-muted)
    - CTA: "무료로 시작하기 →" Button (primary, lg 사이즈, 48px 높이)
    - 하단: 아래 화살표 애니메이션 (bounce)

2. 데이터 섹션 (py-16, surface 배경):
    - 섹션 타이틀: "숫자로 보는 현실" (text-2xl, font-bold, text-center)
    - 3 개 데이터 카드 (세로 스택 모바일, 3 열 데스크탑): 카드 1: 큰 숫자 "58%" (text-4xl, font-bold, primary) + "부모가 자녀 AI 의 안전 기능을 모릅니다" + 출처 카드 2: 큰 숫자 "72%" (text-4xl, font-bold, danger) + "청소년이 AI 동반자 앱을 사용한 경험" + 출처 카드 3: 큰 숫자 "52% vs 52%" (text-4xl, font-bold, warning) + "부모는 비윤리적, 아이는 혁신적이라고 생각" + 출처
    - 숫자에 카운트업 애니메이션 (뷰포트 진입 시, 0 에서 목표값까지 1.5 초)
        - useInView (framer-motion 또는 Intersection Observer) 사용
        - 카운트업: requestAnimationFrame 으로 구현
3. 5 단계 여정 섹션 (py-16):
    - 타이틀: "7 개의 미션으로 완성하는 우리집 AI 안전" (text-2xl, font-bold, text-center)
    - 5 단계 세로 타임라인 (JourneyTimeline 간소화 버전): 각 단계: 이모지 원 + 단계명 + 한줄 설명 + 대표 미션명
        - A: 🔍 이해 — "AI 가 뭔지, 어디가 위험한지 알아요" — 미션: 안전 설정 점검
        - B: 🧭 탐색 — "AI 와 첫 대화를 해봐요" — 미션: 냉장고 파먹기 레시피
        - C: ✍️ 지시 — "AI 에게 잘 부탁하는 법을 배워요" — 미션: 숙제 힌트 요청
        - D: 🔎 검증 — "AI 의 거짓말을 찾아내요" — 미션: 팩트체크 + 편향 찾기
        - E: 🛡️ 책임 — "우리 가족만의 규칙을 만들어요" — 미션: 가족 AI 헌법
4. 차별점 섹션 (py-16, surface 배경):
    - 타이틀: "다른 AI 교육과 뭐가 달라요?" (text-2xl, font-bold, text-center)
    - 3 개 비교 카드 (각각 Card): 카드 1: ❌ "기술 용어를 외우는 강의" → ✅ "아이와 함께하는 15 분 미션" 카드 2: ❌ "AI 를 무조건 막거나 쓰라는 말" → ✅ "우리 가족에 맞는 규칙을 함께 만들기" 카드 3: ❌ "개인정보를 넣어야 하는 AI" → ✅ "개인정보 입력을 자동 차단하는 안전 장치"
5. 최종 CTA 섹션 (py-16, primary-light 배경):
    - "지금 시작하면, 오늘 저녁 아이와 첫 미션을 해볼 수 있어요." (text-xl, font-bold, text-center)
    - CTA Button: "무료로 시작하기 →" (primary, lg)
    - "회원가입 30 초 · 신용카드 불필요" (text-sm, text-muted)

6. 푸터 (py-8, text-muted):
    - "우리집 AI 안전 코치" (font-semibold)
    - 링크: 이용약관 | 개인정보처리방침 (각각 placeholder 페이지 /terms, /privacy)
    - "교육부 수행평가 AI 활용 관리 방안(2025.12)에 기반한 교육 설계"
    - "DOL AI Literacy Framework(2026.2)에 기반한 학습목표"
    - © 2026

[생성할 파일 목록]

- app/page.tsx (전체 재작성)
- app/terms/page.tsx (placeholder: "이용약관 페이지입니다.")
- app/privacy/page.tsx (placeholder: "개인정보처리방침 페이지입니다.")

---

### 프롬프트 #7-B — BottomNav 점검 배지 + 전체 라우팅 검증 + 배포 설정

이전 프롬프트에서 모든 기능 페이지가 완성되어 있습니다.

[이번 프롬프트에서 할 일 — 최종 통합, 점검 배지, 메타데이터, 배포 설정]

components/layout/BottomNav.tsx 수정 — 가족헌법 탭에 점검 리마인더 빨간 점

- 가족 헌법 탭 아이콘에 조건부 빨간 점 배지 표시:
    - Supabase 에서 family_constitution.last_reviewed_at 조회
    - last_reviewed_at 이 null 이거나 7 일 이상 경과 시: 빨간 점 표시
    - 빨간 점: absolute 위치, 4px 원형, danger 배경, 탭 아이콘 우상단
- BottomNav 를 클라이언트 컴포넌트로 변환하고, useEffect 에서 Supabase 실시간 구독 또는 폴링으로 상태 업데이트
- (복잡하면 페이지 로드 시 서버에서 조회하여 props 로 전달하는 방식도 가능)

components/layout/Sidebar.tsx 수정 — 동일한 빨간 점 로직 적용

각 페이지의 metadata 설정:

- app/page.tsx: title "우리집 AI 안전 코치 — 하루 15 분 AI 안전 미션", description "..."
- app/(auth)/login/page.tsx: title "로그인 — 우리집 AI 안전 코치"

- app/(auth)/signup/page.tsx: title "회원가입 — 우리집 AI 안전 코치"
- app/(dashboard)/missions/page.tsx: title "미션 — 우리집 AI 안전 코치"
- app/(dashboard)/missions/[id]/page.tsx: title 동적 "{{미션제목}} — 우리집 AI 안전 코치"
- app/(dashboard)/family-constitution/page.tsx: title "가족 AI 헌법 — 우리집 AI 안전 코치"
- app/(dashboard)/progress/page.tsx: title "진행현황 — 우리집 AI 안전 코치"
- 서버 컴포넌트에서 generateMetadata 함수 사용 (동적 제목이 필요한 페이지)

전체 라우팅 검증 체크리스트 코드 확인:

- middleware.ts 의 라우팅 규칙이 올바른지 확인
- 모든 리다이렉트 경로가 실제 존재하는 페이지를 가리키는지 확인
- (auth) 그룹과 (dashboard) 그룹의 레이아웃이 독립적인지 확인

vercel.json 작성:

```
{
  "framework": "nextjs"
}
```

.env.local.example 업데이트 (이미 있다면 확인):

```
NEXT_PUBLIC_SUPABASE_URL=
NEXT_PUBLIC_SUPABASE_ANON_KEY=
OPENAI_API_KEY=
```

README.md 작성:

```
# 🏠 우리집 AI 안전 코치

비기술직 학부모(어머니)가 자녀와 함께 AI 리터러시를 단계별 미션으로 학습하는 인터랙티브 웹앱입니다.

## 기술 스택

- Next.js 14 (App Router, TypeScript)
- Tailwind CSS + shadcn/ui
- Supabase (Auth + PostgreSQL + RLS)
- Zustand (상태관리)
```

- OpenAI API (AI 응답 생성)

- Vercel (배포)

## 로컬 개발 환경 설정

1. 저장소 클론

2. 의존성 설치: `npm install`

3. `.env.local.example`을 `.env.local`로 복사하고 환경변수 입력

4. Supabase 프로젝트 생성 후 SQL Editor 에서 마이그레이션 실행:

  - `supabase/migrations/001_initial_schema.sql`

  - `supabase/migrations/002_seed_missions.sql`

5. 개발 서버 실행: `npm run dev`

## Vercel 배포

1. Vercel 에 GitHub 저장소 연결

2. 환경변수 3 개 설정 (NEXT_*PUBLIC*_SUPABASE_*URL, NEXT*_PUBLIC_*SUPABASE*_ANON_*KEY,*

*OPENAI*_API_*KEY)*

*3. 자동 배포*

*## 교육 설계 근거*

*- 교육부 수행평가 AI 활용 관리 방안 (2025.12)*

*- U.S. DOL AI Literacy Framework (2026.2)*

*- Common Sense Media — Generation AI Report (2026.3)*

[생성/수정할 파일 목록]

- components/layout/BottomNav.tsx (수정)
- components/layout/Sidebar.tsx (수정)
- 각 페이지의 metadata 추가 (해당 파일들 수정)
- vercel.json (신규)
- .env.local.example (확인/업데이트)

- README.md (신규)

---

## 실행 순서 최종 요약

| 순서 | ID | 프롬프트 | 생성 파일 수 | 핵심 산출물 |

|---|---|---|---|---|

| 1 | #0-A | 프로젝트 초기화 + 컬러 | 7 | Next.js + Tailwind 기본 셋업 |

| 2 | #0-B | shadcn/ui 컴포넌트 | 14 | Button, Card, Input 등 12 개 UI |

| 3 | #0-C | 폴더 구조 + 레이아웃 셸 | 18 | BottomNav, Sidebar, Header + 빈 페이지들 |

| 4 | #1-A | Supabase 클라이언트 + 타입 | 6 | 클라이언트 3 개 + 타입 정의 |

| 5 | #1-B | SQL 마이그레이션 | 1 | 5 개 테이블 + RLS + 트리거 |

| 6 | #1-C | 미션 시드 데이터 | 1 | 7 개 미션 INSERT |

| 7 | #1-D | 인증 + 미들웨어 | 5 | 로그인/회원가입 + 라우팅 보호 |

| 8 | #2-A | Zustand + 온보딩 Step 1,2 | 5 | 온보딩 스토어 + 프로필/걱정 선택 UI |

| 9 | #2-B | 온보딩 Step 3 + 조립 | 3 | 안전 체크리스트 + 온보딩 완성 |

| 10 | #3-A | 단계 로직 + 상수 | 5 | 잠금/승급 유틸 + 미션 메타 |

| 11 | #3-B | 미션 대시보드 | 6 | 카드 리스트 + 진행바 + 승급 모달 |

| 12 | #4-A | PII 가드레일 + API | 3 | AI 호출 엔드포인트 + 개인정보 차단 |

| 13 | #4-B | 미션 2,3 샌드박스 | 5 | 레시피 + 숙제 힌트 UI |

| 14 | #4-C | 미션 4,5 샌드박스 | 2 | 팩트체크 + 편향 찾기 UI |

| 15 | #4-D | 미션 6 + 성찰 + 페이지 조립 | 4 | 동반자 안전선 + 미션 상세 완성 |

| 16 | #5-A | 가족 헌법 | 5 | 초안→수정→서명→점검 전체 플로우 |

| 17 | #5-B | 교사 문의 + 표기 + 합의서 | 4 | 템플릿 3 종 |

| 18 | #6 | 진행현황 | 6 | 타임라인 + 배지 + 통계 + 갤러리 |

| 19 | #7-A | 랜딩 페이지 | 3 | 마케팅 랜딩 + placeholder 페이지 |

| 20 | #7-B | 최종 통합 | 5 | 점검 배지 + 메타 + 배포 설정 |

**총 20 개 프롬프트, 약 108 개 파일 생성/수정.**

각 프롬프트를 Antigravity IDE 에 순서대로 입력하고, 매 프롬프트 실행 후 `npm run dev`로 동작을 확인한 뒤 다음으로 진행하세요. 에러가 발생하면 에러 메시지를 그대로 복사하여 "이전 프롬프트의 결과에서 아래 에러가 발생합니다. 수정해주세요: [에러 메시지]"로 후속 프롬프트를 보내면 됩니다.