Instruction file for Claude Code (BUILD)

Instructions for Claude Code – Builder Gel / BIAB Site

You are an expert full-stack developer using Next.js 14, TypeScript, Tailwind, and Supabase.

Your task is to build a production-ready SEO site that dominates the **builder gel + BIAB** niche, following the attached spec.

Use this updated V4 spec and JSON. Treat both ‘builder gel’ and ‘builder gel nails’ as core head terms. Make sure homepage metadata and internal linking reinforce both.

---

1. Files you must read first

Before you write any code:

1663. Carefully read the attached **V4 spec document** (Blueprint + Specs + Tone).
1664. If provided, also read `builder-gel-site-spec-v4.json`.
1665. If provided, also read `做网站的要求.pdf` and obey any general website requirements there.

Treat these as **source of truth**. Do not simplify the architecture or change slugs/routes.

---

2. Tech stack & project setup

- Framework: **Next.js 14 (App Router)** with **TypeScript**
- Styling: **Tailwind CSS**
- Data: **Supabase (Postgres)**
- Deployment target: **Vercel** (assume typical setup)

Tasks:

2221. Initialize a new Next.js 14 project with TypeScript and Tailwind.
2222. Set up an environment-variable structure that **never exposes secrets on the client**.
2223. Create a basic `app` structure with `layout.tsx`, `page.tsx`, and folder routes for the content sections defined in V4.

---

3. Database schema and seeding (Supabase)

Implement the schema exactly as defined in **Specs V4 (fixed version)**:

- `authors` table (primary key `id` as `text`, e.g. `'sara-kim'`)
- `posts` table (with `author_id` referencing `authors.id` as `text`)
- `products` table
- `post_products` join table

Steps:

2836. Create SQL migration(s) or `schema.sql` for these tables.
2837. Seed `authors` with **Sara Kim** as described in the spec.
2838. Seed `products` from `builder-gel-site-spec-v4.json` if available (Beetles, Modelones, Makartt, Mia Secret, The GelBottle BIAB, Kokoist).
2839. Seed **empty post records** for all slugs in `builder-gel-site-spec-v4.json` (title, slug, category, primary_keyword, secondary_keywords). The `content` field can be temporary placeholder text until we insert real article content.

All content pages must load their data from the database, not hardcoded in code.

---

4. Routing & navigation

Implement routes and navigation exactly as per V4:

- Main sections:
  - `/` – homepage
  - `/guides`
  - `/how-to`
  - `/problems`
  - `/products`
  - `/blog`
  - `/about`
- Content pages for the initial 11 URLs:
  - `/guides/builder-gel-complete-guide`
  - `/how-to/how-to-use-builder-gel`
  - `/how-to/how-to-remove-builder-gel`
  - `/comparisons/builder-gel-vs-acrylic-vs-biab`
  - `/comparisons/builder-gel-vs-poly-gel`

    - `/products/best-builder-gel-kits`
    - `/problems/builder-gel-sticky-or-lifting-fixes`
    - `/problems/builder-gel-cracking-fixes`
    - `/guides/builder-gel-natural-nails`
    - `/guides/builder-gel-nails-near-me`
    - `/about`

Use **SSR** for all content pages. Implement breadcrumbs and navigation menus exactly as described.

---

5. Core layout & components

Create reusable components:

- `Navbar`
- `Footer`
- `Breadcrumbs`
- `ArticleLayout` (wraps each article page)
- `AffiliateDisclosure`
- `InternalLinkCard`
- `StepGif`
- `VideoBlock`
- Product display components (`ProductCard` , optional `ProductList`)

AffiliateDisclosure:

- Rendered **at the top of every monetized article**, inside `ArticleLayout` , immediately after author/date, before the first paragraph.
- Text:

> - "We independently review everything we recommend. When you buy through our links, we may earn a commission."

InternalLinkCard:

- Props: `slug: string` .
- Fetches title + excerpt from `posts` table by slug.
- Renders a small internal-link card with title, short text, and a "Read more" CTA.

StepGif:

- Props: `step: string`.
- Internal mapping from `step` → human-readable title and description.
- Implements a **fallback state**:
  - If no real gif is configured, render a styled "Tip box" with a "💡" icon and text description of what the visual would show.
  - Never leave empty space or broken media.

VideoBlock:

- Props: `id: string`.
- If a video embed URL is defined for that `id`, render the player.
- Otherwise, show a "Video coming soon" card with short description.

---

6. Homepage implementation

Build the homepage according to the V4 layout:

- Hero area (H1 + subheading + CTA buttons)
- Short "What are Builder Gel & BIAB" block
- Question grid linking to the 10 core articles
- "Start Here" 3-card section
- "Problems & Fixes" section
- Product preview (pull first 3–4 products from DB)
- Latest articles (3 most recent posts)
- About & trust block introducing **Sara Kim** with a link to `/about`

Keep it **mobile-first**, fast, and visually clean.

---

7. SEO, robots, sitemap, and schema

Implement:

- `robots.txt` – allow indexing (unless explicitly instructed to stage with `noindex`).
- `sitemap.xml` – include homepage, static pages, and all posts from the DB.
- `generateMetadata` for each page/route in Next.js:
  - Title, description, canonical URL.
  - OpenGraph and Twitter meta tags.

Schema:

- Article schema for posts.
- FAQPage schema where FAQ sections exist.
- HowTo schema for step-by-step guides.
- VideoObject schema when `VideoBlock` is used for a main tutorial video.

---

8. Staging vs production (optional but preferred)

If instructed:

- On staging, apply `<meta name="robots" content="noindex, nofollow">` in the root layout to avoid premature indexing.
- Only remove this when:
  - Homepage, all 10 SEO pages, and About page are filled with real content, and
  - Basic QA on routing, components, and disclosures is complete.

---

9. Content integration

Do **not** generate article content yourself.

The Codex/content agent will produce MDX/Markdown for each slug following the tone and structure rules in the V4 file (Section F: Content Tone & Author Style).

Your job:

- Provide a simple way (e.g. scripts or admin utility) to load those articles into the `posts.content` field and publish them.
- Ensure all `<StepGif />`, `<VideoBlock />`, and `<InternalLinkCard />` components used in MDX render correctly.

---

10. Constraints & quality

- Follow V4 exactly — no changes to slugs, categories, or URL structure.
- Keep components as server components where possible; use client components only where necessary.
- Avoid exposing Supabase service keys on the client.
- Optimize for Core Web Vitals (LCP, CLS, FID).