

AMAZING

AMAZING AI · OPERATOR TOOLKIT

FREE DOWNLOAD · PROMPT PACK · JUNE 2026

The Product-to-Page Prompt Pack

Three copy-paste AI prompts that take a product from a high-converting landing page, to premium imagery, to a verified page live on Shopify.

Edition
June 2026

Inside
Strategy · Imagery · Launch

Built for
Amazon · Shopify operators

THREE PROMPTS · ONE WORKFLOW

From a winning idea to a live landing page

You found the opening. This pack is the build. Three copy-paste prompts move a product from strategy, to premium imagery, to a verified page live on your store. Run them in order, or pull the one you need today.

01 · STRATEGIZE

The Landing Page Strategist turns your product plus a few proven competitor pages into an original, conversion-focused page — strategy, wireframe, and a renderable prototype.

02 · VISUALIZE

The Product Photography prompt turns a plain product shot into premium imagery that keeps your product's exact shape, logo, and finish intact.

03 · SHIP

The Shopify install prompt hands the finished page and images to Codex or Claude, which installs it as real theme files and verifies it live on your store.

These prompts are the manual lever. **The Amazing platform is the engine** — live demand signals, listing-quality scores, and pre-vetted suppliers for every category you build in.

amazing.com/membership →

The Landing Page Strategist

Turn a product and a few proven competitor pages into a complete, original, high-converting landing page.

Use with: **ChatGPT, Claude, or any capable model**

INPUTS

Your product, offer, and brand details, plus 2–4 example landing pages pulled from the Meta Ads Library that look tied to active ads.

WHAT YOU GET

A strategy pass, a full section-by-section wireframe, an image-asset list, the data it still needs from you, and a renderable HTML prototype.

OPERATOR TIP

Feed it real Meta Ads Library pages connected to active ads. It studies the underlying framework, not the words, then builds something fully original to your product.

COPY EVERYTHING BELOW INTO YOUR TOOL

You are an expert DTC landing page strategist, direct-response copywriter, conversion-focused UX designer, and front-end wireframe builder.

I am going to give you:

1. My product details
2. My offer details
3. My brand details, if available
4. A few example landing pages I found from Meta Ads Library that appear to be connected to active ads

Your job is to create the strongest possible high-converting landing page wireframe for my product and offer, using the marketing patterns and conversion framework most likely found in the example landing pages.

Important: Do not copy the example landing pages word-for-word. Do not copy their brand, images, exact claims, or design illegally. Instead, study them to identify the underlying marketing framework, page structure, offer strategy, proof strategy, CTA flow, section order, visual style, and persuasion logic. Then create a fully original landing page for my product using those proven patterns.

The goal is not to create a pretty homepage. The goal is to create one focused landing page with one job: sell the product.

Use the attached product, offer, brand, and sample landing page materials as your source of truth. Do not invent product facts, review counts, certifications, guarantees, scarcity, urgency, discounts, or technical claims. If something important is missing, use a clearly marked placeholder and list it in a "Data Needed From Merchant" section.

Design assumptions:

- Traffic source: cold-to-warm paid social, especially Meta ads.
- Device priority: mobile-first, but desktop should also look strong.
- Page goal: direct product sale, unless my inputs clearly require a different goal.
- Default archetype: PDP direct-response landing page, unless the examples and product clearly suggest a better archetype.
- The page should feel like premium ecommerce but persuade like direct response.
- The page should be skimmable, visual, proof-driven, and conversion-focused.
- Use one clear primary CTA throughout the page.
- Include a sticky CTA for mobile and long-scroll behavior.

Before creating the wireframe, do a brief strategy pass:

1. Identify the likely marketing framework used by the strongest example landing pages.
Examples may include problem-agitation-solution, premium positioning, offer-led, proof-led, mechanism-led, comparison-led, scarcity-led, bundle/value-stack, transformation-led, or risk-reversal-led.
2. Create 2-3 possible landing page strategies for my product.
For each, briefly list:
 - * Strategy name
 - * Hero promise
 - * Main emotional hook
 - * Main proof angle
 - * Main objection it overcomes
 - * Why it might convert
3. Choose the strongest strategy and explain why in 3-5 sentences.

Then create the landing page.

Very important image rule:

Do not generate final images. Do not use generic stock images. Do not describe images vaguely.

Instead, create exact image placeholders directly inside the wireframe where the final images will go.

Each image placeholder must:

- Appear in the exact position where the final image belongs
- Use the exact intended size or aspect ratio
- Include the recommended desktop dimensions
- Include the recommended mobile dimensions

- Include a short visible placeholder label
- Include a detailed image-generation description that I can later use in ChatGPT image generation, Nano Banana Pro, Midjourney, or another AI image tool
- Explain the image's conversion purpose, such as product clarity, aspiration, proof, demonstration, comparison, urgency, or trust

Example placeholder format:

[IMAGE PLACEHOLDER - HERO PRODUCT IMAGE]

Desktop: 1400 × 1000 px

Mobile: 900 × 900 px

Aspect ratio: 1:1 on mobile, 7:5 on desktop

Purpose: Instantly show the product and make it feel premium, useful, and worth the price

Image description: Create a cinematic but realistic image of [product] being used in [specific scene]. The product should match the reference image exactly, including shape, color, logo, proportions, and visible details. The scene should communicate [benefit/emotion]. No text in the image.

Create the full landing page wireframe with these sections, adjusting if the product or examples suggest a better structure:

1. Promo or announcement bar
2. Hero section with product, promise, offer cue, proof cue, CTA, and reassurance microcopy
3. Trust/proof strip
4. Main problem or desire section
5. Product benefits/value prop grid
6. Why this product is different
7. How it works or how to use it
8. Offer stack and pricing section
9. Bundle, upsell, or value-stack section if appropriate
10. Comparison table if useful, especially for premium products
11. Product details/specs/materials/what's included
12. UGC or lifestyle proof section
13. Reviews/testimonials section, using placeholders if real reviews are not provided
14. Guarantee/risk reversal section
15. FAQ section focused on objections
16. Final offer recap
17. Email/SMS capture as a non-buyer exit ramp
18. Footer

For each section, include:

- Section name
- Section goal
- Customer-facing headline

- Customer-facing subheadline
- Customer-facing body copy
- CTA copy, if relevant
- Proof used, if relevant
- Offer element used, if relevant
- Image placeholder, if relevant
- Builder notes that are clearly separated from customer-facing copy

Copywriting rules:

- Make the copy specific, persuasive, and benefit-driven.
- Do not write boring “premium quality” filler.
- Turn features into real-world benefits.
- Use clear headlines that communicate value fast.
- Make the offer feel compelling without making false claims.
- If the product is expensive, justify the price with value, proof, specificity, durability, uniqueness, performance, guarantee, scarcity, or comparison.
- If the product is limited, scarce, or special edition, lean into that only if it is true.
- Use short sections, strong bullets, and clear hierarchy.
- Keep mobile readability in mind.
- Write copy that sounds like a strong human marketer wrote it, not like generic AI copy.
- Do not include internal instructions, bracketed notes, or builder comments in customer-facing copy.

Offer rules:

Build the offer as a stack:

- Primary incentive
- AOV lever, if appropriate
- Friction reducer
- Risk reversal
- Urgency or scarcity only if real

CTA rules:

- Use a CTA that matches the actual action.
- Examples: Add to Cart, Choose Your Bundle, Select Your Option, Claim Limited Edition, Get Yours Today, Build My Kit
- For premium products, consider putting the price or offer cue near the CTA.
- Include reassurance microcopy near the CTA, such as guarantee, secure checkout, free shipping, warranty, easy returns, or limited availability, but only if true.

Proof rules:

Use multiple types of proof when available:

- Reviews

- Testimonials
- UGC
- Founder story
- Technical specs
- Demonstrations
- Certifications
- Press
- Warranty
- Guarantee
- Comparison
- Before/after or use-case proof, if allowed and accurate

If proof is missing, use placeholders and list exactly what is needed.

After the written wireframe, create a renderable landing page prototype in a single self-contained HTML file with embedded CSS and simple JavaScript only if needed.

Rendering requirements:

- Mobile-first responsive design
- Clean premium ecommerce look
- Strong hierarchy
- Rounded cards and clear CTA buttons
- One dominant CTA accent color
- Alternating section contrast where useful
- Sticky mobile CTA
- Image placeholders rendered as styled boxes in the exact size and position where final images will go
- Placeholder boxes must be visually obvious but clean
- Placeholder boxes must contain short labels only, not long notes
- Long image descriptions should appear separately below the wireframe or in a structured image asset list, not cluttering the page

Output format:

1. Brief Strategy Analysis
2. Chosen Landing Page Strategy
3. Complete Landing Page Wireframe
4. Image Placeholder Asset List
5. Data Needed From Merchant
6. Compliance Holds / Claims To Verify
7. A/B Test Ideas
8. Full Renderable HTML/CSS Prototype

Quality check before finalizing:

- Make sure no builder notes appear inside customer-facing page copy.
- Make sure no placeholder variables like {{price}} or [insert review] appear in the rendered customer-facing page unless they are intentionally styled as placeholders for missing merchant data.
- Make sure every image placeholder has a size, position, purpose, and generation description.
- Make sure the page has one clear conversion goal.
- Make sure the hero communicates product, promise, proof, offer, and CTA quickly.
- Make sure the landing page uses a marketing framework inspired by the best examples, but is fully original to my product.
- Make sure the final page feels like a page that could actually sell, not just a design exercise.

Now review the attached product information, offer information, brand information, and sample landing pages, then create the best possible landing page wireframe and renderable prototype.

The Product Photography Prompt Writer

Turn a plain product photo into premium, realistic imagery without losing the product's identity.

Use with: **any AI image model** — Nano Banana, Midjourney, ChatGPT images

INPUTS

One product photo (and an in-use shot if you have it), a short product description, and the platform or use case for the final image.

WHAT YOU GET

One polished, copy-paste image prompt built on a fidelity-first lock that preserves your exact product.

OPERATOR TIP

The fidelity lock is the whole point. It forces the model to keep your real shape, logo, materials, and finish instead of inventing a generic look-alike.

COPY EVERYTHING BELOW INTO YOUR TOOL

You are an expert AI product photography prompt writer.

I am going to give you:

1. One product image and potentially an image showing it in use
2. A short product description
3. The platform or use case for the final image, such as a Shopify landing page, Amazon image, social ad, email image, or website hero section

Your job is to create one high-quality AI image prompt that turns my product photo into a premium, realistic product image.

Use a Fidelity-First approach.

The most important priority is that the product must still look like the product in the attached image. Start the prompt by locking in the product identity before describing the scene.

Make sure the prompt clearly tells the image model to preserve:

- The product's exact shape and proportions
- The visible logo, label, text, or branding
- The product's color, materials, texture, finish, and key design details
- Any visible buttons, caps, lids, handles, straps, screens, packaging, lenses, or special features

Do not let the product get redesigned, simplified, reinterpreted, or turned into a generic version of itself.

After locking in the product identity, create a realistic landing-page-quality product photography scene around it.

Use this structure:

1. Start with the image type and use case Example: "A photorealistic premium lifestyle product image for a Shopify landing page..."
2. Describe the product clearly and naturally Use a simple product name, such as "the flashlight," "the lantern," "the bottle," "the supplement jar," "the coffee bag," or "the kitchen tool."
3. Add the Fidelity Lock Use wording similar to: "Use the attached product photo as the identity foundation for the product. Preserve the visible shape, proportions, materials, colors, logos, labels, text, and key product details. Do not redesign, simplify, or reinterpret the product."
4. Place the product in a believable scene The product should feel like it naturally belongs there, not like it was pasted onto a background. Include a simple reason why it is in the scene.
5. Add realistic lighting Use clear photography language such as soft window light, softbox studio lighting, warm golden-hour light, subtle rim light, realistic reflections, or soft contact shadows.
6. Add physical realism Make sure the prompt includes realistic contact shadows, believable scale, natural reflections, and interaction with the surface or environment.
7. Add composition and camera direction Use practical photography language, such as 50mm lens, 85mm lens, f/2.8, f/8, centered composition, rule-of-thirds composition, or generous negative space for landing page copy.
8. Add restrictions Tell the image model not to add fake text, fake logos, fake badges, random claims, extra product parts, warped shapes, distorted hands, cluttered props, or anything that changes the product.

Keep the final image prompt polished but not overly complicated. It should be easy for a beginner to copy and paste into an AI image tool.

Here is the product description:

[PASTE PRODUCT DESCRIPTION HERE]

The image should be used for:

[PASTE PLATFORM OR USE CASE HERE]

The desired mood or style is:

[PASTE MOOD HERE — example: premium, rugged, warm, clean, modern, family-friendly, luxurious, adventurous, tactical, refreshing]

Now write the final AI image prompt only.

Install the Page on Shopify with Codex or Claude

Hand the finished page and images to a coding agent and let it install, wire, and verify a real Shopify landing page.

Use with: **Codex or Claude Code, with Chrome access recommended**

INPUTS

Your AI-designed page files, the generated images, and access to your Shopify Admin.

WHAT YOU GET

A live, verified Shopify landing page installed as real theme files — not a giant Custom Liquid paste-in — with images on Shopify Files and CTAs wired to checkout.

OPERATOR TIP

Two access paths, never mixed: Theme Access plus the CLI handles code; a Dev Dashboard app plus the Admin API handles store data. Never paste secrets, tokens, or .env contents into chat.

COPY EVERYTHING BELOW INTO YOUR TOOL

This handout is for a store owner who already has:

- An AI-designed landing page.
- Generated images for that page.
- Access to their Shopify Admin.
- Codex or a similar coding agent on their computer.

The goal is to let Codex guide the setup and, when possible, do the technical work for you: set up Theme Access, set up a Shopify development app for Admin API access, upload images, install the landing page as real Shopify theme files, create the Shopify page, and verify the result.

The Big Idea

Shopify automation uses two different access paths:

```
Theme Access + Shopify CLI
Used for theme files:
- Liquid sections
- CSS
- JavaScript
- JSON templates
- layout/theme.liquid changes
```

```
Shopify Dev Dashboard app + Admin API
Used for store data:
- Shopify Pages
- Shopify Files
- products and variants
- collections
- discounts
- metadata
- navigation
- metafields and metaobjects
```

Do not try to use one credential for everything. Theme Access handles code. The development app handles store data.

Best Setup: Enable Codex Chrome Access

If your Codex setup supports the Chrome skill or Chrome plugin, enable it before starting.

With Chrome access enabled, Codex can use your real Chrome browser, including your logged-in Shopify session. That means Codex can usually:

- Open Shopify Admin and the Shopify Dev Dashboard.
- Show you exactly where to click.
- Install or open the Theme Access app.
- Walk you through creating a Theme Access password.
- Create the Shopify development app.
- Add the correct Admin API scopes.
- Help install the app on your store.
- Verify the app scopes.
- Open the live landing page for desktop and mobile checks.

You may still need to personally approve installs, log in, complete two-factor authentication, or copy one-time secrets. Do not paste passwords, client secrets, access tokens, or .env contents into public chat, video comments, screenshots, or docs.

What You Need Ready

Before asking Codex to begin, gather:

- Your Shopify *.myshopify.com domain.
- Your public store domain.

- The AI-generated landing page files.
- The generated images for the landing page.
- The product handle or product URL, if the page sells a product.
- Any offer details, discounts, bundles, or checkout behavior.
- Your preferred page URL handle, such as summer-offer or new-product-launch.

If you do not know some of these, Codex can help find them from Shopify.

Step 1: Create One Local Folder For The Store

Ask Codex to create a dedicated repo or folder for the store. A typical folder looks like this:

```
my-shopify-store/  
  .env  
  .env.example  
  .gitignore  
  AGENTS.md  
  deploy.md  
  docs/  
  scripts/  
  shopify-landing-page-upload/  
    sections/  
    assets/  
    templates/
```

The .env file stores local secrets and should not be committed.

The .env.example file should look like this:

```
SHOPIFY_SHOP_DOMAIN=your-store.myshopify.com  
SHOPIFY_API_VERSION=2026-04  
SHOPIFY_CLIENT_ID=  
SHOPIFY_CLIENT_SECRET=  
SHOPIFY_CLI_THEME_TOKEN=
```

Step 2: Install Theme Access

Theme Access lets Shopify CLI safely push theme files.

Human version:

1. Open Shopify Admin.
2. Install the Shopify Theme Access app.
3. Open Apps > Theme Access.
4. Click Create theme password or Create password.
5. Send the password link to yourself.
6. Open the link and copy the password once.
7. Save it in local .env as:

Codex-assisted version:

Ask Codex to use Chrome to open Shopify Admin, find Theme Access, and guide you through each click. Codex can verify the token later without printing it.

Important: installing Theme Access is not enough. You must create a theme password.

Step 3: Verify Theme Access

Once the Theme Access password is saved locally, Codex should run:

```
npx -y @shopify/cli@latest theme list `
  --store your-store.myshopify.com `
  --password "$token"
```

If this lists your Shopify themes, the theme-file rail works.

Step 4: Create The Shopify Development App

The development app gives Codex Admin API access for pages, files, products, metadata, discounts, and related store data.

Human version:

1. Open the Shopify Dev Dashboard.
2. Create a new app for this store.
3. Create an app version.
4. If the app has no embedded UI, use:

```
https://shopify.dev/apps/default-app-home
```

5. Add the Admin API scopes below.
6. Release the version.
7. Install the app on your Shopify store.
8. Copy the Client ID and Secret into .env.

Codex-assisted version:

Ask Codex to use Chrome to open the Dev Dashboard, create the app, add the scopes, release the version, and guide you through the install. You may need to personally approve the install and handle credentials.

Step 5: Add The Recommended Admin API Scopes

For landing-page installation, Shopify Files, pages, product wiring, metadata, and common offer setup, use this baseline:

```
read_content
write_content
read_discounts
write_discounts
read_files
write_files
read_inventory
write_inventory
read_locations
read_metaobject_definitions
write_metaobject_definitions
read_metaobjects
write_metaobjects
read_online_store_navigation
write_online_store_navigation
read_online_store_pages
write_online_store_pages
read_products
write_products
read_publications
write_publications
read_themes
write_themes
```

Only add customer, order, draft-order, checkout, market, subscription, or protected customer data scopes if your specific workflow needs them.

If you add scopes later, release a new app version and approve the updated access in Shopify Admin. Releasing a version alone may not update the installed app's granted scopes.

Step 6: Verify Admin API Access

Codex should run a small verification script that:

- Reads .env.
- Requests a short-lived Admin API token with the client credentials grant.
- Queries the current app installation.
- Prints the shop name/domain and granted scopes.
- Does not print secrets or access tokens.

The key GraphQL check is:

```
query CurrentAppScopes {
  currentAppInstallation {
    accessScopes {
      handle
    }
  }
}
```

If a required scope is missing, Codex should stop and help you approve the updated app access.

Step 7: Convert The AI Page Into Real Shopify Files

Do not paste a large AI-generated page into one Custom Liquid block.

Use real theme files:

```
sections/custom-landing-page.liquid
assets/custom-landing-page.css
assets/custom-landing-page.js
templates/page.custom-landing-page.json
```

The Liquid section should load CSS and JavaScript like this:

```
{{ 'custom-landing-page.css' | asset_url | stylesheet_tag }}
<script src="{{ 'custom-landing-page.js' | asset_url }}" defer></script>
```

Use one unique wrapper class around the whole page so the CSS does not leak into the rest of the theme.

Step 8: Upload Images To Shopify Files

Generated product, hero, lifestyle, proof, offer, gallery, and social preview images should go to Shopify Files, not theme assets.

Use:

```
Shopify Admin > Content > Files
```

Use simple filenames:

```
hero-product.jpg
lifestyle-scene.jpg
offer-stack.jpg
proof-detail.jpg
social-preview.jpg
```

Then reference them in Liquid with `file_url`:

```

```

Use `asset_url` for CSS, JavaScript, fonts, small icons, and code-owned theme assets. Use `file_url` for merchant-facing landing-page images.

Step 9: Push Only The Intended Files

Before pushing to Shopify, Codex must:

1. Read `deploy.md`.

2. Run git status --short.
3. Tell you exactly what is uncommitted.
4. Ask whether uncommitted work should be included before any deploy.
5. Never print secrets.

Codex should use targeted Shopify CLI pushes with --nodelete and --only.

Example:

```
npx -y @shopify/cli@latest theme push `
  --store your-store.myshopify.com `
  --theme 123456789 `
  --path .\shopify-landing-page-upload `
  --nodelete `
  --allow-live `
  --password "$token" `
  --only "sections/custom-landing-page.liquid" `
  --only "assets/custom-landing-page.css" `
  --only "assets/custom-landing-page.js" `
  --only "templates/page.custom-landing-page.json"
```

Do not push the whole live theme unless you intentionally approve that.

Step 10: Pull Back And Verify

After pushing, Codex should pull back the same files into a verification folder and compare them.

This confirms Shopify received the files you intended to install.

Step 11: Create Or Update The Shopify Page

Codex can use Admin GraphQL to create a Shopify Page and assign the custom template.

The template suffix must match the JSON template filename:

```
templates/page.custom-landing-page.json
```

The suffix is:

```
custom-landing-page
```

The page create input should include:

```
{
  "title": "Landing Page Title",
  "handle": "landing-page-handle",
  "body": "<p>Short summary of the page or offer.</p>",
  "isPublished": true,
  "templateSuffix": "custom-landing-page"
}
```

Codex should inspect GraphQL userErrors before reporting success.

Step 12: Set SEO And Social Metadata

Codex should set:

- Page title.
- SEO title.
- Meta description.
- Open Graph title.
- Open Graph description.
- Open Graph image.
- Twitter/social image where supported by the theme.

For Shopify Pages, search listing metadata commonly uses:

```
global.title_tag  
global.description_tag
```

Step 13: Wire Buttons To The Real Next Step

Top-of-page buttons can scroll to the offer section:

```
<a href="#order">Shop Now</a>
```

Final purchase buttons should use the real Shopify variant ID, not the product ID:

```
<form action="/cart/add" method="post">  
  <input type="hidden" name="id" value="{ { landing_variant.id } }">  
  <input type="hidden" name="quantity" value="1">  
  <input type="hidden" name="return_to" value="/checkout">  
  <button type="submit">Checkout Now</button>  
</form>
```

Do not place a real test order during verification.

Step 14: Verify In A Real Browser

With the Chrome skill enabled, Codex should use your real Chrome browser to verify:

- The landing page loads on the correct public URL.
- Desktop layout looks right.
- Mobile layout looks right.
- All images load from Shopify Files.
- CTAs point to the correct section, product, cart, or checkout.
- Add-to-cart uses a variant ID.

- The page redirects to checkout when intended.
- No real order is placed.
- Metadata appears in the live HTML.
- Normal Shopify pages still work.
- Any header/footer hiding applies only to the landing page template.

If the page uses lazy loading or reveal animations, Codex should scroll from top to bottom and back before taking final screenshots.

Step 15: Commit The Verified Source

After the live page is verified, commit the source files to the repo.

The repo becomes the source of truth for:

- Theme files.
- Image filenames.
- Page template suffix.
- Product/variant wiring.
- Metadata.
- Deployment notes.
- Verification notes.

Kickoff Prompt For Codex

Paste this into Codex after putting your landing-page files and images in the store folder.

I want you to help me install an AI-generated landing page on my Shopify store.

Use Chrome if it is available so you can walk me through Shopify Admin and the Shopify Dev Dashboard in my real browser. You may guide me through clicks and setup steps, and you may perform browser actions when safe, but do not print secrets, tokens, Theme Access passwords, client secrets, or .env contents.

Store details:

- Shopify myshopify domain:
- Public domain:
- Landing page name:
- Desired page URL handle:
- Product URL or product handle, if this page sells a product:
- Offer or discount details:
- Folder containing the AI-generated landing page:
- Folder containing the generated images:

Please do the full setup and install workflow:

1. Inspect this folder and create or update the store repo structure.
2. Create or update .env.example, .gitignore, AGENTS.md, and deploy.md.
3. Help me install Theme Access and create a Theme Access password if it is not already configured.
4. Save or guide me to save SHOPIFY_CLI_THEME_TOKEN locally without exposing it.
5. Verify Shopify CLI theme access with theme list.
6. Help me create a Shopify Dev Dashboard app for this store if one is not already configured.
7. Add the Admin API scopes needed for landing pages, Shopify Files, pages, products, variants, discounts, metadata, publications, navigation, metaobjects, metafields, and themes.
8. Help me release the app version, install the app on the store, and save SHOPIFY_CLIENT_ID and SHOPIFY_CLIENT_SECRET locally without exposing them.
9. Verify Admin API access and granted scopes using currentAppInstallation.accessScopes.
10. Convert the AI-generated landing page into real Shopify theme files: section, CSS asset, optional JS asset, and page JSON template.
11. Upload generated merchant-facing images to Shopify Files and reference them with file_url.
12. Use asset_url only for CSS, JavaScript, fonts, icons, and theme-owned assets.
13. Before any Shopify push or Admin API mutation, read deploy.md, run git status --short, list uncommitted work, and ask me whether it should be included.
14. Push only the intended files with Shopify CLI using --nodelete and --only.
15. Pull back the pushed files and verify the remote code.
16. Create or update the Shopify Page with the correct templateSuffix.
17. Set page title, body summary, SEO title, meta description, and social preview metadata where supported.
18. Wire CTAs to the correct page section, product, cart, or checkout path using variant IDs where needed.
19. Verify the live page in Chrome on desktop and mobile.
20. Do not place a real test order.
21. Restore or clear test cart state after checkout testing.
22. Give me a final summary of what changed, what was verified, and what I still need to approve or review.

FROM PROMPT TO BRAND

Take Amazing AI inside the platform

These prompts help you build. Amazing AI helps you choose what to build — continuously. Inside the platform you get live demand-velocity, listing-quality scores, and pre-vetted supplier matches for every category you are working in.

DEMAND**Live category velocity**

Track search, sales, and review-volume trends across Amazon, Shopify, and Walmart in real time.

LISTINGS**Quality & staleness scores**

See which incumbents are vulnerable to a better entrant by age, rating decline, and review sentiment.

SUPPLY**Pre-vetted supplier matches**

Manufacturer shortlists with capability, certification, and MOQ already validated by the Amazing team.

Start Your Free Trial →

amazing.com/membership · No card charged today · Cancel anytime

Prompts and opportunities require execution. Tools are necessary but not sufficient — sourcing, brand, and operational discipline still decide who wins each category. Use this pack as a starting point, not a finish line.