

MINI PACK

Get Found on Google

SEO basics for your vibe-coded app, so search engines can actually see it.

For Next.js, Vite + React, Lovable, Bolt, and v0 · with Claude Code or Cursor

10 prompts · 4 sections

Welcome! How to Use This Pack

You built the app, you published it, and it works. Then you search Google for it and find nothing, or a result with the wrong title and no description. Sharing the link in a group chat shows a blank card. It feels like the internet can't see your app.

Usually there are a few plain reasons. Google can run JavaScript, but its own docs say rendering can be delayed and not all bots can run JavaScript, so a page that only fills in after the scripts load is harder to read. Then there are the basics that are easy to miss: one generic title on every page, no description, no sitemap, or the same site at two addresses. Lovable's docs, for example, say new projects start with generic metadata and don't always get a sitemap. Google publishes clear guidance on all of this, and so do the frameworks these tools use. This pack walks you through it: checking what Google sees, telling it what each page is about, handing it a map of your site, and writing pages that match what people search for.

The prompts work whichever tool you built with. Some have extra notes for Next.js (what v0 builds), for Vite + React single-page apps (what older Lovable projects use), and for Lovable's own SEO tools.

Search engines and these tools change often. Everything here comes from official docs as of October 2026, mainly Google's, Next.js's, Vercel's, and Lovable's. If a menu or setting looks different for you, check the current docs. Google's docs also say there's no guarantee any site will be added to its index, and changes can take anywhere from a few hours to several months to show up. This pack can't promise rankings. It helps you remove the things that stop Google from seeing and understanding your pages.

How each prompt is laid out

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it.

Anything in **[BRACKETS IN CAPS]** is a blank to fill in.

Works best with: Claude Code or Cursor, opened on your app's code, so the AI can read how your pages are built and fix them. If you build in Lovable, Bolt, or v0, paste the prompts into that tool's chat. You'll also use Google Search Console, Google's own tool for site owners. Prompts 4.1 and 4.2 only need the details you type in, so you can also run them in any AI chat.

The words, in one line each. **SEO** (search engine optimization) means making your pages easy for search engines to find, read, and show. **Crawling** is when Google's bot visits a page, and **indexing**

is when Google saves it so it can show up in results. **Rendering** is running a page's JavaScript to see the finished page. A **title** is the page name shown in the browser tab and often in search results, and a **meta description** is a short summary in the page's code that results sometimes show. A **sitemap** is a file that lists your pages, and **robots.txt** is a file that tells crawlers which parts of your site they can visit. A **canonical URL** is the one address you want Google to use when the same page can be reached at more than one.

Quick Start: Find Your Problem

- **Not sure what's wrong and want a full checkup?** Prompt 1.1.
- **Your app is a single-page app, or the page source looks almost empty?** Prompt 1.2.
- **Every page has the same title, or no description?** Prompt 2.1.
- **Shared links show a blank card or the wrong image?** Prompt 2.2.
- **Your site works at both a .vercel.app address and your own domain, or with and without www?** Prompt 2.3.
- **No sitemap, or not sure your robots.txt is right?** Prompt 3.1.
- **Never set up Google Search Console?** Prompt 3.2.
- **Search Console says a page isn't indexed?** Prompt 3.3.
- **Want to know what your users actually search for?** Prompt 4.1.
- **Ready to write pages that answer those searches?** Prompt 4.2.

Golden rule: Check what Google actually sees before you change anything, and write every page for people first.

What's Inside

1. Can Google See My App? (2 prompts)
2. Tell Google What Each Page Is (3 prompts)
3. Sitemap, robots.txt, and Search Console (3 prompts)
4. Pages People Actually Search For (2 prompts)

SECTION 1

Can Google See My App?

Look at your site the way a search engine does, before you fix anything.

1.1 SEO Health Check for My Live Site

Use this when your app is live and you want to know why it isn't showing up on Google. It's a read-only checkup of the basics on your real pages, with the biggest problems listed first.

Fill in: [MY LIVE URL] (e.g. "https://myrecipeapp.com"), [MY MAIN PAGES] (e.g. "home, features, /recipes, one recipe page"), [HOW I BUILT IT] (e.g. "Next.js from v0", "Lovable", "Vite + React in Cursor", or "not sure").

Don't change any files yet. My app is live, but it doesn't show up on Google. Do a read-only SEO health check and explain it in plain words. I'm a beginner.

My live URL: [MY LIVE URL]

My main pages: [MY MAIN PAGES]

How I built it: [HOW I BUILT IT]

For each main page, check the HTML the server sends first (before any JavaScript runs), using curl or the project code, and tell me:

1. The <title>. Is it there, and is it different from the other pages?
2. The meta description. Is it there, and does it describe this page?
3. Whether the main heading and the main text are in that first HTML, or only appear after JavaScript runs.
4. Any robots meta tag or X-Robots-Tag header, especially "noindex".
5. The rel="canonical" link, and whether it points to this page's own address on my real domain.
6. The HTTP status code, including for a made-up address like /this-page-does-not-exist (it should be 404, not 200).
7. Whether links between pages are normal links, and whether any page addresses use # (like /#/recipes).
8. Whether /robots.txt and /sitemap.xml exist, and whether they return real files or just my home page.

For each check, say "OK", "Problem", or "Not sure", with one line on why. Then list the problems from most to least likely to keep me out of Google, with a short fix for each and which prompt type would fix it (visibility, titles, previews, duplicate addresses, sitemap, or Search Console). If you're not sure how Google handles something today, tell me to check Google Search Central instead of guessing.

 **Tips to customize:** Run this in plan mode in Cursor or Claude Code, so the AI explains before it edits anything. Google's docs say pages behind a login should return a status like 401, and pages

that don't exist should return 404, so Google knows not to index them. If you built in Lovable, its docs describe an SEO & AI search review on the SEO & AI tab of Analytics that checks many of these same things, so run that too and compare. A health check finds technical problems. It can't tell you whether your content matches what people search for, which is what Section 4 is for.

1.2 Is My Content Visible Without JavaScript?

Use this when your app is a single-page app (often Vite + React) and you're not sure Google can read it. It's for comparing the HTML your server sends with the finished page, then picking the right fix.

Fill in: [MY LIVE URL], [MY FRAMEWORK AND HOST] (e.g. "Vite + React on Netlify", "Lovable", or "not sure"), [PAGES THAT MATTER MOST FOR SEARCH] (e.g. "home page, features, the public recipe pages").

Don't change any files yet. I want to know whether search engines can read my app's content, or only see an empty shell until JavaScript runs. I'm a beginner.

My live URL: [MY LIVE URL]

My framework and host: [MY FRAMEWORK AND HOST]

Pages that matter most for search: [PAGES THAT MATTER MOST FOR SEARCH]

What I know from Google's docs:

- Google crawls a page, then renders it in a headless Chromium browser later. The page can wait in the rendering queue for a few seconds or longer.
- Server-side rendering or pre-rendering is still a great idea, because it makes the site faster and not all bots can run JavaScript.
- Google can only reliably follow links that are `<a>` elements with an href. Single-page apps should use the History API for routing, not `#` fragments.
- In single-page apps, returning a real 404 status can be hard, so a missing page may return 200 with a "not found" message. Google calls that a soft 404. The fix is to redirect to a URL that returns a real 404, or add a `noindex` robots meta tag on the error view.
- If the original HTML has `noindex`, Google may skip rendering, so removing `noindex` with JavaScript may not work.
- Google calls dynamic rendering a workaround, not a long-term solution, and recommends server-side rendering, static rendering, or hydration instead.

Please:

1. Tell me how this project renders pages: in the browser only, on the server, at build time, or a mix. Show me the files that prove it.
2. For each page I listed, show me what's in the first HTML response (titles, headings, main text) compared to what appears after JavaScript runs.
3. Check routing for `#` URLs, links that aren't `<a href>`, and missing pages that return 200.
4. Recommend the smallest fix that gets my important pages into the first HTML: for example, pre-rendering the public pages at build time, or moving them to a framework that renders on the server. Explain the tradeoffs in plain words, and don't recommend dynamic rendering.

5. Tell me how to check the result myself afterward.
If you're not sure what my framework or host supports, tell me which docs to check instead of guessing.

💡 **Tips to customize:** To see what Google sees, use the URL Inspection tool in Google Search Console (Prompt 3.2 sets it up). Its live test shows a screenshot and the HTML Google rendered. Lovable's docs say its newer apps use server-side rendering, and older React + Vite apps are pre-rendered for verified crawlers such as Google, Bing, and social-preview bots, so check its crawlability docs before you move anything. If you built with Bolt, check Bolt's help center for its current SEO and hosting options. In Next.js's App Router, which v0 uses, pages are Server Components by default and render on the server, so this prompt matters most for Vite + React single-page apps.

SECTION 2

Tell Google What Each Page Is

Give every page its own name, its own preview, and one official address.

2.1 Give Every Page Its Own Title and Description

Use this when every page shows the same title in the browser tab, or your pages have no meta description. It's for writing a clear title and description for each page and putting them where search engines can read them.

Fill in: [MY APP IN ONE SENTENCE] (e.g. "an app that turns your fridge leftovers into recipes"), [MY APP'S NAME], [MY PAGES] (e.g. "home, features, /recipes, each recipe page, about"), [MY FRAMEWORK] (e.g. "Next.js App Router", "Vite + React", or "not sure").

Give every public page in my app its own title and meta description. Show me the plan before you change anything.

My app in one sentence: [MY APP IN ONE SENTENCE]

My app's name: [MY APP'S NAME]

My pages: [MY PAGES]

My framework: [MY FRAMEWORK]

What I know from Google's docs:

- Each page should have a unique, descriptive `<title>`. Avoid the same boilerplate on every page, and avoid repeating keywords.
- It's fine to put the site name at the start or end of each title, after a dash, colon, or pipe.
- The main heading should be the most prominent text on the page, for example the first `<h1>`.
- Google often builds the snippet from the page content, and sometimes uses the meta description if it describes the page better. Each page's description should be unique and describe that page.

What I know from Next.js's docs (if I use the App Router):

- Static pages can export a metadata object, and pages that depend on data (like a single recipe page) can use `generateMetadata`.
- These only work in Server Components.
- A root layout can set `title.template` (like `"%s | [MY APP'S NAME]"`) and `title.default`, and child pages fill in `%s`.

Please:

1. List my public pages and what each one's title and description is today.
2. Write a new title and description for each page, in plain words that match what the page actually offers. Don't stuff in keywords.
3. Put them in the right place for my framework. For a Vite + React app, tell me

what's in `index.html`, how per-page titles will be set, and whether they'll be in the first HTML or only added by JavaScript (and what that means for search, per Prompt 1.2).

4. Check that each page has exactly one clear main heading.
5. Tell me how to check the result in the page source.

 **Tips to customize:** Google's docs say there's no limit on how long a title or description can be, but results get cut off to fit the screen, so put the most useful words first. Pages behind a login don't need search titles, so skip them. For data pages like a single recipe, ask the AI to build the title from the data (for example, the recipe name). React's docs say you can render a `<title>` from any component and React places it in the page's head, but in a single-page app that still happens in the browser.

2.2 Link Previews That Look Right

Use this when you share your link and the preview shows no image, the wrong image, or the wrong text. It's for adding Open Graph tags and a share image so each page previews properly.

Fill in: [MY LIVE URL], [MY FRAMEWORK], [PAGES PEOPLE SHARE MOST] (e.g. "home page and each recipe page"), [MY SHARE IMAGE] (e.g. "I have a 1200×630 PNG with my logo", "make one from each recipe's photo", or "I don't have one yet").

Set up Open Graph tags and share images so my links preview properly in social apps and chats. Show me the plan before you change anything.

My live URL: [MY LIVE URL]
My framework: [MY FRAMEWORK]
Pages people share most: [PAGES PEOPLE SHARE MOST]
My share image: [MY SHARE IMAGE]

What I know:

- The Open Graph protocol lists four required properties for every page: `og:title`, `og:type`, `og:image`, and `og:url` (the canonical address).
- `og:image` should be a full URL (starting with `https://`) that anyone can open without logging in.
- Facebook's docs say images are cached by URL, so to change an image after it's been shared, use a new image URL.
- Not all bots run JavaScript, so tags added only by JavaScript may not appear in previews.

What I know from Next.js's docs (if I use the App Router):

- I can put an `opengraph-image` file (`.jpg`, `.jpeg`, `.png`, or `.gif`) in a route folder, or generate one with code, and Next.js adds the tags for me. An `opengraph-image.alt.txt` file sets the alt text.
- The `opengraph-image` file must be under 8MB and `twitter-image` under 5MB, or the build fails.
- `metadataBase` in the root layout turns relative image paths into full URLs.

Please:

1. Show me the Open Graph and Twitter/X tags each page has today, from the first HTML the server sends.
2. Add og:title, og:description, og:type, og:url, and og:image (with width, height, and alt text) to the pages I listed, using my framework's way of doing it.
3. If I don't have an image, suggest a simple design and how to make it, or set up a generated one if my framework supports it.
4. Make sure every image URL is absolute and points at my real domain.
5. Tell me how to test the preview with each platform's own link-checking tool.

💡 **Tips to customize:** Facebook's Sharing Debugger shows which tags its crawler reads, and fetches your page again when you run it, which helps after you fix something. Other platforms have their own tools, so search their help pages. Next.js's docs say that for pages rendered on each request, it streams metadata in after the page starts loading, but not for bots that expect tags in the head, like Twitterbot, Slackbot, and Bingbot. If you use Lovable, its docs list Open Graph metadata as one of the basics to set up.

2.3 One Address Per Page: Canonicals and Duplicate Domains

Use this when your site opens at a .vercel.app address and your own domain, or with and without www. It's for choosing one official address per page so search engines don't split your pages between copies.

Fill in: [MY REAL DOMAIN] (e.g. "https://www.myrecipeapp.com"), [OTHER ADDRESSES THAT SHOW MY SITE] (e.g. "myrecipeapp.vercel.app and myrecipeapp.com without www", or "not sure"), [MY HOST] (e.g. "Vercel", "Netlify", "Lovable", or "Bolt").

Make sure each page on my site has one official address, and that search engines know which one it is. Don't change any files until I approve the plan.

My real domain: [MY REAL DOMAIN]

Other addresses that show my site: [OTHER ADDRESSES THAT SHOW MY SITE]

My host: [MY HOST]

What I know from Google's docs:

- A redirect is a strong signal for which URL is canonical, rel="canonical" is a strong signal, and listing a URL in the sitemap is a weak one. Using more than one together helps.
- Each canonical page should have a rel="canonical" link pointing to itself, as a full URL.
- Don't give the same page different canonical URLs in different places (for example, one in the sitemap and another in the canonical tag).
- Don't use noindex to pick a canonical within a site, because it removes the page from search completely.

What I know from Vercel's docs (if I use Vercel):

- Preview deployments get an X-Robots-Tag: noindex header by default, except when a custom domain is assigned to a non-production branch.
- If the production .vercel.app address and my custom domain both show the same pages, Vercel suggests a permanent redirect from the .vercel.app hostname to my custom domain that keeps each page's path, or an absolute canonical for each page. Don't point every page's canonical at the home page.

Please:

1. Check which of my addresses load the site, and whether they redirect to my real domain or show a copy.
2. Plan permanent redirects from the other addresses to my real domain, keeping the path, in the right place for my host.
3. Add a self-pointing canonical URL to every public page, built from my real domain.
4. Check that my sitemap and internal links use the same addresses.
5. Give me a list of addresses to test afterward, and how to check each redirect's status code.

If you're not sure how my host handles domains or redirects today, tell me to check its docs instead of guessing.

 **Tips to customize:** In Next.js, `metadataBase` plus `alternates: { canonical: '/' }` (or each page's own path) creates the canonical tag for you. To check whether a Vercel preview is hidden from search, run `curl -I` on its address and look for `x-robots-tag: noindex`. Our Ship It Live pack covers connecting a custom domain (Prompt 4.1) and trailing slashes, redirects, and a real 404 page (Prompt 3.2), so use those for the setup itself.

SECTION 3

Sitemap, robots.txt, and Search Console

Hand Google a map of your site, then use Google's own tool to see what it found.

3.1 Make a Sitemap and robots.txt That Actually Work

Use this when your site has no sitemap or robots.txt, or you're not sure they're right. It's for creating both, listing only the pages you want in search, and checking that they really load.

Fill in: [MY REAL DOMAIN], [MY FRAMEWORK] (e.g. "Next.js App Router", "Vite + React"), [MY PUBLIC PAGES] (e.g. "home, features, about, and every recipe page from the database"), [PAGES TO KEEP OUT OF SEARCH] (e.g. "dashboard, settings, admin, anything after login").

Create a sitemap and robots.txt for my site, and make sure they work on the live site. Show me the plan before you change anything.

My real domain: [MY REAL DOMAIN]

My framework: [MY FRAMEWORK]

My public pages: [MY PUBLIC PAGES]

Pages to keep out of search: [PAGES TO KEEP OUT OF SEARCH]

What I know from Google's docs:

- A sitemap should list full, absolute URLs (like `https://www.example.com/features`), and only the canonical pages I want in search results.
- Google ignores `<priority>` and `<changefreq>`. It uses `<lastmod>` only if it's consistently accurate.
- robots.txt must be at the root of the site, like `https://www.example.com/robots.txt`. It can include a "Sitemap:" line with the full sitemap URL.
- robots.txt is not a way to hide pages from Google. A blocked page can still show up in results without a description. To keep a page out, use `noindex` or put it behind a login, and don't block it in robots.txt, or Google can't see the `noindex`.
- Don't block scripts or styles the page needs, or Google may have trouble understanding the page.

What I know from my framework's docs:

- Next.js: an `app/sitemap.ts` file can return the list of URLs (including ones from my database), and `app/robots.ts` can return the rules. Both are served at the root.
- Vite: files in the public folder are copied to the root of the build as-is, so `public/robots.txt` is served at `/robots.txt`.

Please:

1. Build the sitemap from my real public pages, including pages that come from my database, using my real domain. Leave out private, duplicate, and test pages.

2. Write a robots.txt that allows my public pages and scripts, and points to the sitemap. Only add Disallow rules if they're truly needed, and explain each one.
3. Make sure private pages are kept out with noindex or a login, not with robots.txt alone.
4. After I deploy, tell me how to check that /robots.txt and /sitemap.xml return the real files, not my app's home page.

 **Tips to customize:** In a single-page app, a missing file can quietly load your home page instead of a 404, so open both addresses in your browser after deploying and make sure you see plain text and XML. If your site has fewer than a few dozen pages, Google's docs say you can write the sitemap by hand. Lovable's docs say it creates a robots.txt with every new project but not always a sitemap, and its SEO & AI search review flags both. A sitemap is only a hint, so Prompt 3.2 shows you how to tell Google about it and see whether it worked.

3.2 Set Up Google Search Console and Submit My Sitemap

Use this when you've never used Google Search Console, or you set it up but never submitted a sitemap. It's for proving you own the site, telling Google where your sitemap is, and asking it to look at your most important pages.

Fill in: [MY REAL DOMAIN], [WHERE I MANAGE MY DOMAIN'S DNS] (e.g. "Namecheap", "Cloudflare", "Vercel", or "not sure"), [MY SITEMAP URL] (e.g. "https://www.myrecipeapp.com/sitemap.xml"), [MY MOST IMPORTANT PAGES] (e.g. "home page and features page").

Walk me through setting up Google Search Console for my site, one step at a time. I've never used it. Wait for me to finish each step before giving me the next.

My real domain: [MY REAL DOMAIN]

Where I manage my domain's DNS: [WHERE I MANAGE MY DOMAIN'S DNS]

My sitemap URL: [MY SITEMAP URL]

My most important pages: [MY MOST IMPORTANT PAGES]

What I know from Google's docs:

- A Domain property covers every subdomain and both http and https, and must be verified with a DNS record. A URL-prefix property covers one exact address prefix and has more ways to verify.
- I need owner permissions to submit a sitemap in the Sitemaps report. I paste the sitemap URL into "Add a new sitemap" and click Submit. You can't upload a sitemap file to Google, only tell it where to find it.
- Submitting a sitemap is a hint. It doesn't guarantee Google will crawl every URL in it.
- The URL Inspection tool shows whether a page is indexed. "Test live URL" checks the page as it is now, and "Request indexing" asks Google to crawl it. There's a daily limit, and asking again for the same URL won't make it faster.
- Crawling can take anywhere from a few days to a few weeks.

Please:

1. Help me choose between a Domain property and a URL-prefix property for my setup, and explain why.
2. Walk me through verifying it, including exactly where to add the DNS record at my DNS provider. Tell me what to do if it doesn't verify right away.
3. Walk me through submitting my sitemap in the Sitemaps report, and what each status means.
4. Walk me through running a live test on my most important pages, and requesting indexing for each one once.
5. Tell me what to check over the next few weeks, and which reports to look at. If a screen or menu name may have changed, tell me to check Google's Search Console Help instead of guessing.

 **Tips to customize:** If you can't edit your DNS, a URL-prefix property is the easier start. Add a "Sitemap:" line to your robots.txt too, since Google's docs list that as another way to tell it about your sitemap. If you use Lovable, its docs say you can connect Search Console from the Google searches card on the SEO & AI tab, and Lovable can verify your site and submit your sitemap for you. Publish your latest changes first, because Google reads the sitemap from your live site.

3.3 Why Isn't This Page on Google?

Use this when Search Console says a page isn't indexed and you don't know what the status means. It's for reading the reason in plain words and fixing only what actually needs fixing.

Fill in: [THE PAGE URL], [WHAT SEARCH CONSOLE SAYS] (paste the status from the Page Indexing report or URL Inspection, e.g. "Crawled - currently not indexed"), [WHAT THE LIVE TEST SHOWS] (paste what "Test live URL" says, or "haven't run it").

Don't change any files yet. Google Search Console says one of my pages isn't indexed. Help me understand why and what, if anything, to fix. I'm a beginner.

The page URL: [THE PAGE URL]

What Search Console says: [WHAT SEARCH CONSOLE SAYS]

What the live test shows: [WHAT THE LIVE TEST SHOWS]

What I know from Google's docs:

- "Crawled - currently not indexed" means Google crawled the page but didn't index it. It may be indexed later, and there's no need to resubmit it.
- "Discovered - currently not indexed" means Google found the page but hasn't crawled it yet.
- "Duplicate without user-selected canonical" means Google picked another page as the canonical one. URL Inspection shows which.
- "Soft 404" means the page shows a "not found" style message but doesn't return a 404 status code.
- A page with noindex is dropped from results. If robots.txt blocks the page, Google can't see the noindex at all.

Please:

1. Explain what this status means for my page, in plain words.
2. Tell me whether it's a real problem or something to wait out.
3. If it's a problem, check my code and live page for the likely cause: noindex, a robots.txt block, a wrong or missing canonical, a soft 404, content that only appears after JavaScript, or a thin page with little useful text.
4. Suggest the smallest fix, and how to confirm it with "Test live URL" afterward. If you don't recognize the status, say so and tell me to look it up in Google's Page Indexing report help instead of guessing.

 **Tips to customize:** Not every page needs to be indexed. Login pages, settings, and near-empty pages are fine to leave out. Google's docs say a live test doesn't check everything, such as quality guidelines or manual actions, so a passing test doesn't guarantee indexing. If the cause is duplicate addresses, use Prompt 2.3. If the page is thin, Prompt 4.2 can help you make it more useful.

SECTION 4

Pages People Actually Search For

Technical fixes let Google read your site. Helpful pages give people a reason to find it.

4.1 Find What My Users Type Into Google

Use this when you don't know which words your future users search for. It's for listing the problems your app solves in your users' own words, so you know what pages to write.

Fill in: [WHAT MY APP DOES] (e.g. "turns your fridge leftovers into recipes"), [WHO IT'S FOR] (e.g. "busy college students who cook for one"), [WHAT THEY USE TODAY INSTEAD] (e.g. "googling recipes, or ordering takeout"), [SEARCH CONSOLE QUERIES] (paste the top rows from the Performance report's Queries tab, or "no data yet").

Help me figure out what my future users type into Google when they have the problem my app solves. Don't invent search numbers. If you don't have real data, say so.

What my app does: [WHAT MY APP DOES]

Who it's for: [WHO IT'S FOR]

What they use today instead: [WHAT THEY USE TODAY INSTEAD]

Search Console queries: [SEARCH CONSOLE QUERIES]

Google's SEO Starter Guide says to think about the words people might search for, and that beginners may use different words than experts.

Please:

1. List 15 to 20 searches my users might type, grouped by the problem behind them. Use plain, everyday words, the way a beginner would type them, not marketing words.
2. For each group, say what the person really wants to find (for example, a quick answer, a how-to, or a tool to try).
3. Mark which ones my app genuinely helps with, and which ones it doesn't, so I don't write pages that promise something my app can't do.
4. If I pasted Search Console queries, point out any that I already appear for but get few clicks on.
5. Suggest the 5 best groups to write a page for first, and why.



Tips to customize: Search Console's Performance report shows the searches your site appeared for, but only once Google has some data for your site. Its docs note you may not see your site when you search the same words yourself, because results depend on time, place, device, and history. Ask a few real users how they'd search for your app, and add their exact words. If you use Lovable, its docs say you can ask it in chat for keyword and competitor research.

4.2 Write Problem-Based Landing Pages That Help

Use this when you know what people search for and want a page that answers it. It's for writing a small number of genuinely useful pages, each built around one problem, instead of lots of thin copies.

Fill in: [THE SEARCH GROUP] (one group from Prompt 4.1, e.g. "what to cook with leftover rice"), [HOW MY APP HELPS WITH IT] (e.g. "you type in your leftovers and get three recipes"), [WHAT I KNOW FIRST-HAND] (e.g. "I tested 40 leftover-rice recipes while building this", or "nothing special yet"), [MY FRAMEWORK].

Help me write one landing page that genuinely helps people who search for this problem, and add it to my site. Show me the outline before you write the page.

The search group: [THE SEARCH GROUP]

How my app helps with it: [HOW MY APP HELPS WITH IT]

What I know first-hand: [WHAT I KNOW FIRST-HAND]

My framework: [MY FRAMEWORK]

What I know from Google's docs:

- Google's ranking systems are designed to prioritize helpful, reliable content made for people, not content made mainly to rank.
- "Scaled content abuse" means making many pages mainly to manipulate rankings, including using AI to generate many pages without adding value.
- "Doorway" pages are made to rank for similar searches and lead people to in-between pages that are less useful than the final destination.
- Keyword stuffing means filling a page with keywords to try to rank, and it can make a page look spammy.

Please:

1. Outline a page that answers the question first, in plain words, then shows how my app helps, with one clear next step.
2. Use what I know first-hand. Don't make up stats, reviews, or claims. Mark anything I need to fill in or check.
3. Give it its own title, description, main heading, and a readable URL with real words in it.
4. Link to it from at least one relevant page on my site with a normal link, and add it to my sitemap.
5. Before writing a second page, check that it would answer a different question. If it would only swap a few words, tell me to make one better page instead.



Tips to customize: One good page beats ten near-copies. Write one, publish it, and use URL Inspection (Prompt 3.2) to ask Google to look at it. Then give it a few weeks before judging it, since Google's docs suggest waiting a few weeks to see whether a change helped. Our Make It Pretty pack (Section 4) covers landing page copy and structure, so use it to polish the words once the page answers the right question.

Bonus: Magic Phrases for SEO

- "Don't change any files yet. Show me what Google sees first."
- "Check the first HTML the server sends, before any JavaScript runs."
- "Give every page its own title, description, and one official address."
- "Don't use robots.txt to hide a page. Use noindex or a login."
- "Write it for people first. Don't stuff keywords or make thin copies."
- "If you're not sure how Google or my framework works now, tell me to check the docs."

Get Found on Google

That's the whole pack: **10 prompts across 4 sections**. Check what Google can see, tell it what each page is, hand it a map of your site, and write pages people are actually looking for.

Happy building!