The Niche Content Site Playbook

How to build a niche site that turns a visitor into a lead: an assessment or tool that gives real value, then points them to your offer.

The pattern here is simple: give someone a personalized result, then make the ask. A quiz, assessment, or tool that produces something specific to them earns trust and an email address in one move. A live site doing real revenue uses exactly this stack.

01 **Niche + value hook** — pick the topic, then build a quiz/assessment that pays off

↓

02 **Next.js on Vercel** — interactive app + push-to-deploy

↓

03 **Clerk sign-in** — save results behind a login

↓

04 **Postgres via Prisma** — store sessions, responses, reports

↓

05 **Multi-step assessment** — collect personality/niche signals across phases

↓

06 **Claude generates the result** — strong model does the work; OpenAI fallback for resilience

↓

07 **PDF + dashboard** — something to keep, something to share

↓

08 **Resend delivery email** — you own the contact now

↓

09 **CTA to paid offer** — Stripe checkout or a link-in-bio storefront

---

The Build Sequence

1. The idea — niche down, then give something away

Pick a narrow topic where you can make a confident claim: “I can tell you your best content niche in 10 minutes.” The hook is a quiz or assessment that produces a personalized result — not generic advice, but something that feels tailored to that person. That specificity is what makes people opt in and share it.

The build only works if the result is actually useful. Spend time here before you write a line of code.

2. Framework + hosting — Next.js on Vercel

Next.js handles the interactive UI (multi-step forms, dashboards, download flows), the API routes, and server-side rendering for pages that need it. Vercel takes a `git push` and deploys it — preview URLs on every pull request, serverless functions automatically. You don’t manage a server.

Why this pair: everything lives in one repo and one dashboard; you can ship a fix in minutes.

3. Auth — Clerk for sign-in

Clerk is a drop-in authentication service. Sign-up, sign-in, session management — it’s handled. You add a provider, wrap your layout, protect your routes. A live implementation of this stack uses Clerk so results can be saved to a dashboard after the assessment completes. Sign-up happens naturally at the point where someone wants to see their result.

Why Clerk over rolling your own: it handles email verification, session tokens, social logins, and webhooks (to sync users to your DB) out of the box. You’d spend a week building what Clerk gives you in an afternoon.

4. Database — Postgres via Prisma

Postgres is the database. Prisma sits in front of it: you define your schema in a `.prisma` file, run a migration, and get fully typed database queries. No raw SQL for the common paths.

What gets stored: user profiles (synced from Clerk via webhook), assessment sessions (which phase they’re on, their answers), generated report records, and Stripe payment status. Prisma’s migration system means you can evolve the schema without fear.

5. The value engine — multi-phase assessment + Claude

This is the whole product. The assessment runs across multiple phases — in the live build, that’s a short personality-and-values assessment followed by niche-specific questions. Each phase hits an API route that stores progress so the user can come back.

At the end, the collected answers go to Claude. The model reads the full profile and writes a personalized niche recommendation with context, rationale, and next steps. The live build uses Claude (Anthropic SDK) as the primary model with an OpenAI fallback for resilience. The AI isn’t doing generic work here — it’s synthesizing 20+ data points into something the user couldn’t have gotten by Googling.

Why AI for this: the personalization at scale is the moat. You can’t write a bespoke recommendation for every user by hand.

6. Deliver the result — dashboard + downloadable PDF

The generated result renders on a dashboard page the user can return to. It also exports as a downloadable PDF via `@react-pdf/renderer` — a library that lets you build a PDF layout in React components and render it server-side.

Why a PDF: something to download and keep has more perceived value than a webpage. People share PDFs with their communities. It's also a natural "save this" moment that reinforces the product's value.

7. Capture the contact — Resend delivery email

Once the report is ready, Resend fires a transactional email with the download link. That email is the start of the relationship. You now have a verified address, permission to follow up, and a reason to stay in touch.

Why Resend: one API for both transactional (receipts, links) and broadcast (newsletters, follow-up sequences). You don't need two email tools.

8. Monetize — CTA from the result to your paid offer

The free result earns the right to make the ask. The live build has a clear call to action from the report page — a Stripe Checkout session for an upgrade, or a link to a link-in-bio storefront for lower-friction purchases. The CTA isn't a pop-up; it's a natural next step that grows from what the user just read.

Why tie the offer to the result: conversion is highest at the exact moment someone just received value. Don't wait to follow up by email — surface the offer while they're engaged.

9. Ship it safe — security headers + Vercel

A site that handles personal data (personality results, email addresses, payment info) needs to be locked down. The live build uses a strict Content Security Policy (CSP) and standard security headers applied via Next.js middleware. Vercel enforces HTTPS automatically.

Why bother: sloppy security headers are the difference between a professional product and one that browsers warn users about.

What it costs to run

Near-free to start: Vercel Hobby is free, Clerk's free tier covers thousands of monthly active users, a managed Postgres host (Neon, Railway, or similar) has a free tier that holds plenty to start, Stripe charges per transaction (no monthly fee), Resend's free tier covers the first several thousand emails a month. Claude costs a few cents per report at typical input/output sizes. A first month of real usage might cost $5–15 in AI calls.

The first upgrades you'd actually pay for:

- **Clerk paid (~$25/mo)** once monthly active users pass the free cap, or if you need advanced features like organization support.

- **Database upgrade** — most managed Postgres hosts (Neon, Railway) have a paid tier ($10–25/mo) when you hit free storage or row limits, or when you want automated daily backups.
- **AI spend** grows directly with volume. Set a monthly budget cap in your Anthropic account. Watch it — it's the only line item that scales automatically with traffic.

If you want to see how each piece connects and actually get something out the door, the Workshop walks you through it.

KEEP GOING

Pick one step. Do it today. Ship the thing — then build the next one.

MORE PLAYBOOKS — FREE

The Tools I Use	paulguardado.com/playbooks/the-tools-i-use
The Creator Platform Playbook	paulguardado.com/playbooks/creator-platform-playbook
The AI Tool (Micro-SaaS) Playbook	paulguardado.com/playbooks/ai-tool-playbook

---

— Paul