FOUNDER'S FIELD GUIDE

AI Tools & Tech Stacks for Startups

The shortlist we use at haynes lab to ship faster, spend less, and skip the wrong tools.

V1.0 · 2026 EDITION

haynes lab

How to Read This Guide

This is not a comprehensive list. It is a curated shortlist of the AI tools and stack components we have actually shipped with at Haynes Lab in the last year. If a tool is missing, it is missing on purpose — either it is no longer the best in its category, or it is not worth the cost or complexity for a startup.

The new rules of building in 2026

- **AI is the default IDE, not an add-on.** Every line of new code starts with an agent suggesting it. Pick coding tools first, framework second.
- **Buy the boring parts.** Auth, payments, email, error tracking, and analytics are commodity. Pay $20–$50 a month and move on.
- **Postgres until proven otherwise.** Don't add Mongo, Redis, Kafka, or Elasticsearch on day one. Postgres covers all four for the first 50,000 users.
- **One framework per layer.** Resist the urge to mix Next.js + Remix + a separate API. Pick one.
- **Deploy where the docs are best.** Vercel, Fly, Railway, Cloudflare — all good. Pick the one with the cleanest deploy story for your stack and don't look back.

Inside

AI Coding Tools — Pick Your Pair Programmer

The coding agent you choose now affects your velocity more than your framework choice. **Claude Code is the required piece of our stack** — everything else on this page is optional. The leading tools all sit on top of the same frontier models (Claude, GPT, Gemini), so the differentiator is the harness — how it reads your repo, runs commands, and plans multi-step work. Claude Code's harness is the best.

Tool	Best For	Pricing (approx)	Verdict
`Claude Code`	Long-running agentic work in the terminal; large refactors; multi-file changes. Rock solid harness — it just works. The non-negotiable piece of our stack.	$20–$200/mo (Pro/Max)	REQUIRED
`Gemini (Code Assist + CLI)`	Currently the highest-ranked frontier model on most coding benchmarks. Slower than Claude Code in agentic loops but better in specific situations (huge context, certain refactors).	$0–$20/mo	WATCH / PICK
`Cursor / Windsurf`	VS Code-style IDEs with strong agent flows. Useful, but the AI features overlap heavily with what Claude Code already does for free if you have a Max plan.	$15–$20/mo	OPTIONAL
`GitHub Copilot`	Inline completion in any IDE; teams already on GitHub Enterprise	$10–$39/mo	ADD-ON
`v0 / Lovable / Bolt`	Generating throwaway prototypes from a text prompt	$20/mo	DEMOS ONLY

OUR SETUP

Claude Code Max ($200/mo) inside a normal IDE — no Cursor subscription. Claude Code does the heavy lifting: planning, multi-file changes, long agentic runs. The IDE is just the editor. For 99% of working developers, one full day of Claude Code use pays for the entire month. Gemini is at the top of the benchmark leaderboard right now and we use it for specific situations where its strengths show up — but Claude Code's harness is still the most productive end-to-end loop.

Adjacent tools worth knowing

- **CodeRabbit / Greptile** — automated PR reviews. Worth it once you have more than two contributors.
- **Linear + Linear Agent** — issue tracking with an AI agent that can pick up tickets and open draft PRs.
- **Warp** — AI-native terminal. Useful if you live in the shell.
- **Granola / Fathom** — meeting notes that auto-extract action items. Cuts product-management overhead in half.

The Four Layers of a Modern Startup Stack

04

Every product we ship sits on four layers. Decide one tool per layer, then resist swapping any of them for the first six months.

LAYER 01	**Frontend & Mobile** What the user actually sees and touches. Web app, mobile app, or both.
LAYER 02	**Backend & Data** Where business logic lives, where rows are stored, and how clients fetch them.
LAYER 03	**AI** LLM calls, embeddings, agents, voice, image — anything that uses a model.
LAYER 04	**Plug-and-Play Services** Auth, payments, email, monitoring, analytics, file storage — bought, never built.

A simple framing for choosing

- **Layer 1 — Frontend:** ask "do I need SEO and a marketing site?" If yes, Next.js. If pure app-behind-login, you can pick Vite + React or Next.js — both fine.
- **Layer 2 — Backend & Data:** ask "do I want a managed Postgres + auth + storage in one box?" If yes, Supabase — that is our default. If your data shape is changing weekly during early MVP, MongoDB until it stabilizes.
- **Layer 3 — AI:** default to Anthropic Claude for reasoning + tool use, OpenAI for speech, FLUX or Replicate for images. Use OpenRouter to A/B test cheaply.
- **Layer 4 — Services:** Supabase Auth (or Clerk/Auth0 if you want more polish), Stripe for payments, Resend for email, Sentry for errors, PostHog for analytics. Total: about $0–$80/month at MVP scale.

TRAP TO AVOID

Don't pick all four layers in one sitting. Lock in Layer 1 and 2 first, ship something a user can touch, then add the AI and services layers as the product demands them.

Frontend & Mobile Layer

05

For most consumer-facing or B2B SaaS startups, the answer is one of three combinations. Pick by where your users are, not by what looks fun to learn.

Stack	When to use it	Verdict
`Next.js 15 + React 19`	SaaS, marketing site, dashboards, anything SEO-relevant. The default.	DEFAULT
`Flutter`	Mobile-first product where you want one codebase across iOS, Android, web, and desktop with pixel-level UI control. Our most-used mobile stack at Haynes Lab.	FOR MOBILE
`Expo + React Native`	Mobile-first product when the team is already React-heavy. Ship iOS + Android from one codebase.	FOR MOBILE
`Vite + React + TanStack`	Pure SPA behind login. No SEO, no SSR needs.	OPTIONAL
`SvelteKit / SolidStart`	You personally love them. Smaller hiring pool.	NICHE

Styling & UI components

- **Tailwind CSS 4** — utility-first styling. Pair with the design system you actually want, don't fight it.
- **shadcn/ui** — copy-paste components built on Radix. You own the source. Best balance of speed and customization.
- **Radix Themes / Mantine** — full component library if you don't want to assemble shadcn pieces.
- **Framer Motion** — animation library that does not feel like fighting CSS keyframes.
- **Lucide / Heroicons** — SVG icon sets. Don't use emoji as functional icons.

Backend & Data Layer

06

99% of startups should start with Postgres and a managed platform on top. The interesting decision is how thick a layer you want above the database.

Database — almost always Postgres

Option	Why pick it	Verdict
`Supabase`	Postgres + auth + storage + realtime + edge functions in one dashboard. Generous free tier.	DEFAULT
`MongoDB`	Useful when you genuinely cannot predict your data shape — early MVPs where the schema is changing every week. Migrate to Postgres once the model stabilizes.	MVP ONLY
`Turso / D1`	SQLite at the edge. Cheap, fast for read-heavy workloads.	NICHE

Backend framework / API style

- **Next.js Route Handlers** — if your frontend is Next.js, just use these. Zero extra infra.
- **Node.js + Express** — when you split frontend and backend into separate services. Boring, battle-tested, every developer knows it. Our default for a standalone API behind a Flutter or React Native app.
- **Hono on Bun or Cloudflare Workers** — modern, fast, tiny. Pick when you want edge deploy and a smaller runtime.
- **FastAPI (Python)** — when your team is Python-native or you need heavy ML in-process.
- **tRPC** — end-to-end type safety between Next.js and clients. Worth it for dashboards.
- **GraphQL (Apollo, Hasura)** — only if you have multiple consumers with very different shapes. Otherwise REST or tRPC wins.

ORM / data access

- **Drizzle** — TypeScript-first, lightweight, SQL-like. Our default.
- **Prisma** — mature, generates a typed client. Heavier runtime; great if you live in migrations.
- **Kysely** — typed query builder if you want to write closer to SQL.

Background jobs & queues

- **Trigger.dev** — durable background tasks with TypeScript. Best DX for Node-first teams.
- **Inngest** — event-driven workflows, retries, fan-out. Strong for AI pipelines.
- **QStash (Upstash)** — cheap HTTP-based queue. Good for simple delayed jobs.

AI Layer — Models, Vectors, Voice, Images

Pick a primary LLM provider, then add specialized providers as your product grows. Wrap calls behind a thin abstraction in your own code so you can swap models without rewriting features.

LLM providers

Provider	Strength	Use it for
`Anthropic Claude`	Reasoning, agentic tool use, long-context tasks, coding	Default for most product features
`OpenAI`	Speech (Whisper, Realtime), broad ecosystem	Voice, multimodal, fallback model
`Google Gemini`	Massive context windows, cheap fast models	Bulk document processing, summarization
`OpenRouter`	Single API key across 100+ models	A/B testing models without juggling keys
`Groq / Cerebras`	Ultra-fast inference of open-weight models	Latency-sensitive features (autocomplete, voice)

SDKs & orchestration

- **Anthropic SDK / OpenAI SDK** — go direct first. They are simple and well-documented.
- **Vercel AI SDK** — clean React hooks for streaming UIs. Provider-agnostic.
- **Mastra / LangGraph** — only when you need multi-step agent graphs. Skip on day one.
- **LangChain / LlamaIndex** — heavy abstractions. Useful for teaching, less so for shipping.

Vector storage & retrieval

- **pgvector on Supabase or Neon** — the answer for 90% of startups. Same Postgres, no new service.
- **Turbopuffer** — serverless vector DB, very cheap at scale.
- **Pinecone / Weaviate / Qdrant** — managed vector DBs. Worth it past ~10M vectors.

Voice

- **Deepgram** — best speech-to-text latency and accuracy on the market.
- **ElevenLabs** — best voice synthesis. Realistic, multilingual.
- **Cartesia** — sub-100ms TTS, ideal for live voice agents.
- **OpenAI Realtime API** — full-duplex voice chat in one socket. Easiest path to a voice MVP.

Image & video

- **FLUX (BFL) / Ideogram** — best-in-class image generation right now.
- **Replicate / fal.ai** — host and run open-weight models without managing GPUs.
- **Runway / Luma / Pika** — video generation. Maturing fast.

Plug-and-Play Services

These are the boring parts. Don't build any of them yourself. Pick one per category, integrate in an afternoon, move on.

Auth

- **Supabase Auth** — our default. If you are already using Supabase for the database, you get auth for free with social logins, magic links, and row-level security. Covers 95% of startups.
- **Clerk** — genuinely great. Polished UI components, multi-tenant org support, excellent DX. More setup than Supabase Auth and a separate vendor, but worth it if auth UX is a priority.
- **Auth0** — also very solid, especially for enterprise-flavored requirements. Heavier setup than Supabase Auth and pricier than Clerk.
- **Better Auth / Auth.js** — open-source, self-hosted. Pick when you want to own auth in your codebase.
- **WorkOS** — when you need SSO/SAML for enterprise customers. Add later, not at MVP.

Payments

- **Stripe** — the only payment processor we use. Checkout for one-off, Billing for subscriptions, Connect for marketplaces. Don't shop around — Stripe wins on docs, SDK quality, and dashboard.

Email

- **Resend** — what we reach for almost every time. Clean SDK, React Email components for templates, generous free tier, deliverability has been a non-issue.

Monitoring & analytics

- **Sentry** — error tracking, performance monitoring, session replay. Wire it up before launch, not after.
- **PostHog** — product analytics, session replay, feature flags, A/B tests in one tool. Self-hostable if you want to.
- **Better Stack** — uptime monitoring, log aggregation, status pages, on-call rotations in one dashboard. Great DX, replaces three separate tools.

File storage & CDN

- **Supabase Storage** — our default. Free if you are already on Supabase, S3-compatible API, integrates with row-level security.
- **Railway Volumes** — when your app is deployed on Railway, the built-in persistent volumes cover most file storage needs without a separate service.
- **Cloudflare CDN** — put it in front of your assets for cache + global edge delivery. Pair with Supabase or Railway storage. We don't reach for R2 unless we need it.
- **AWS S3** — works fine but more setup, more IAM config, and a steeper learning curve than the above. Pick when you already live in AWS.
- **Cloudflare Images / Cloudinary** — for on-the-fly image transforms.

Customer support

- **Plain** — modern, developer-friendly support tool. Great if your users are technical.
- **Pylon** — B2B-focused, integrates with Slack channels.
- **Intercom / Front** — heavier; pick once you have a real support team.

Deployment & Testing

Two parts of the stack that founders consistently underweight at MVP and overweight after. Get just enough in place to ship safely — and no more.

Deployment

- **Railway — our default for MVPs.** "Connect repo, deploy" is one click. Databases, persistent volumes, cron jobs, preview environments, and per-branch deploys all live in the same dashboard. We ship 90% of MVPs here and only move when scale, multi-region, or compliance forces our hand.
- **Vercel** — Next.js's home turf. Zero-config preview deploys, great for marketing sites and Next-first apps. Pricier at scale.
- **Cloudflare Pages / Workers** — cheapest at scale, edge-first. Great for static + edge functions.
- **Fly.io / Render** — full container hosting when you outgrow Railway and want more control over the runtime.
- **Kubernetes (graduating step).** When traffic, multi-region, regulated workloads, or a hybrid-cloud requirement makes Railway no longer enough, the next stop is a form of Kubernetes — managed (EKS, GKE, DigitalOcean Kubernetes) or lightweight (k3s, Talos). Don't start here. Migrate here when there is a real reason, ideally with someone on the team who has run k8s in production before.
- **GitHub Actions** — our default CI/CD. Free for public repos and small teams. Run tests, build images, deploy to Railway / Fly / k8s from a single workflow file.

Unit & integration testing

Test the things that, if broken, lose money or trust — auth, payments, core data writes. Skip snapshot tests (they rot faster than the code they protect) and skip coverage targets (they reward writing tests that prove nothing). Add a regression test the day a bug ships, not before.

- **Vitest** — our default for JavaScript and TypeScript projects. Fast, ESM-native, Jest-compatible API. Pair with React Testing Library for component tests.
- **Playwright** — for end-to-end browser tests. The only E2E tool we recommend; cross-browser, fast, debuggable.
- **flutter test** — built into Flutter. Covers unit, widget, and integration tests in one runner.
- **pytest** — for Python backends. Pair with `pytest-asyncio` when you have async code.
- **Bun's built-in test runner** — when the project already runs on Bun. No extra dep.

Three Reference Stacks

Three real combinations we have shipped. Copy any of them whole — they are designed to fit together with no glue code beyond what your product needs. Mix freely: a SaaS with an AI feature is just Stack 1 + the AI pieces from Stack 2; a mobile product with a marketing site is Stack 3 + a tiny Next.js site on Vercel.

The Default SaaS

WEB · B2B · SUBSCRIPTION

For: dashboards, internal tools, B2B SaaS, anything behind a login with a marketing site in front.

`Next.js 15` `React 19` `Tailwind 4` `shadcn/ui` `Supabase (DB + Auth + Storage)` `Drizzle` `Stripe` `Resend` `Sentry` `PostHog` `Vercel`

Why it works: single repo, single deploy command, zero ops. Supabase covers Postgres, auth, and file storage in one bill. New engineer is productive on day one. Scales past where you will sell the company.

Monthly cost at MVP: $0–$60

The AI Wrapper Done Right

AI · STREAMING · RAG

For: AI-native products. Chat UIs, agents, RAG over a customer corpus.

`Next.js 15` `Vercel AI SDK` `Anthropic Claude` `OpenRouter` `Supabase + pgvector` `Supabase Auth` `Inngest` `Stripe (metered)` `Sentry` `PostHog` `Vercel`

Why it works: streaming UI is one hook (`useChat`). Embeddings live in the same Postgres as your app data — no second database. Inngest handles long agent runs without timing out.

Monthly cost at MVP: $20–$200 + model usage

The Mobile-First Product

IOS + ANDROID · ONE CODEBASE

For: consumer or prosumer apps where mobile is the product and web is secondary.

`Flutter` `Riverpod` `go_router` `Supabase (DB + Auth + Storage)` `RevenueCat` `Sentry` `PostHog` `Codemagic / Fastlane`

Why it works: Flutter ships pixel-identical UI to iOS, Android, web, and desktop from one codebase. Supabase covers data, auth, and storage. RevenueCat normalizes Apple and Google subscription pain. Swap Flutter for Expo + React Native if your team is already React-heavy — same backend, same outcome.

`Monthly cost at MVP: $0–$130 (RevenueCat + Supabase free tiers)`

What to Skip — Common Pitfalls

The fastest way to ship is to not build the wrong things. These are the choices we see kill startup velocity, in rough order of damage.

Microservices on day one

A monolith with clear modules is faster to ship and easier to refactor. Split when a team owns each service, not before.

Kubernetes

Use Vercel, Fly, or Railway until you have a real reason. K8s is a full-time job that adds zero customer value at <100k users.

Custom auth

Rolling your own auth is how startups end up in security incident reports. Pay Clerk or use Supabase Auth.

A separate marketing-site CMS

MDX in your Next.js repo handles 90% of marketing-site content needs at zero cost.

GraphQL too early

REST or tRPC ships faster, debugs easier, and is what your team probably knows.

A second database

Postgres can do JSON, full-text search, vectors, and queues. Add Redis or Mongo only when Postgres provably cannot.

Premature monorepo tooling

Turborepo and Nx are great when you have 4+ packages. Until then they slow you down.

Heavy AI agent frameworks

LangChain abstractions hide the model. Call the SDK directly until you actually need a graph.

Self-hosted analytics

PostHog Cloud, Plausible Cloud, and Mixpanel cost less per month than the time to operate them.

Building your own admin panel

Use Retool, Tooljet, or a Supabase dashboard query. Your time goes into product, not internal CRUD.

THE BIGGEST TRAP

Choosing a stack to learn from instead of a stack to ship with. Pick the boring, well-documented combination. Save the experimental tooling for a side project. The market does not reward your stack — it rewards the product the stack helped you ship.

Where Haynes Lab Fits In

If you read all this and thought "I want someone to just build it for me — properly, in two weeks, on the right stack," that is what we do.

Haynes Lab

MVP development & fractional CTO for early-stage startups

We ship MVPs in 2–4 weeks on the stacks in this guide. We have built AI-native SaaS, mobile apps, internal tools, and operator platforms — for solo founders and Series-A teams. We charge in scoped fixed bids, not hourly mystery, and we hand over clean code you (or your future engineers) can keep building on.

Common engagements: a 2-week MVP build, a fractional CTO retainer, a 1-week tech-stack audit before you hire your first engineer, or a focused AI-feature add-on for an existing product.

Want to talk? Reply to the email this PDF arrived in, or book a 30-minute call.

Book a Call → | See Our Work

Versioning

This guide is updated quarterly. The current version is **v1.0 · 2026 Edition**. Future updates ship to anyone on the lead-magnet list — no need to re-download.

A note on bias

Every recommendation here reflects what we have actually used. We have no affiliate links, no paid placements, and no sponsorships. Our bias is toward shipping fast and not regretting the decision in six months.