# Chapter 7: What's Next

> *"You've learned to build. Now learn to ship consistently."*

## What You've Accomplished

Let's take a moment to appreciate how far you've come.

**You went from** "I can't code" **to** "I built and deployed a web application."

**You learned:**

- How apps are structured (front-end, back-end, database)
- How to choose your tech stack
- How to set up authentication
- How to work with databases
- How to deploy to production
- How to debug when things break

**You built:**

- A real application with user accounts
- Database integration
- Multiple features working together
- Something live on the internet

**Most importantly:** You proved you don't need to learn traditional programming to build real products.

This isn't just a tutorial you completed. This is evidence that you can build. That's a different identity. You're now someone who ships.

---

## The Path Forward

You have three options from here:

### Option 1: Build Your Next Project

**If you have a product idea**, build it. You know enough now. Use the same process:

1. Describe what you want to AI
2. Build incrementally
3. Test as you go
4. Deploy early
5. Iterate based on feedback

**Start simple:**

- One core feature that solves a real problem
- Basic authentication if needed
- Simple UI (polish comes later)
- Deploy as soon as it does anything useful

**Common second projects:**

- Internal tool for your current job
- Side project to solve your own problem
- Freelance client project
- Productized service

**Timeline:** 2-4 weeks to first version

### Option 2: Deepen Your Skills

**If you want to get better before starting your own project**, build variations:

**Different app types:**

- Build a blog with comments (content-focused)
- Build a task manager (CRUD operations)
- Build a link-in-bio tool (simple but useful)
- Build a form builder (meta: build a tool to build tools)

**Add new features:**

- Add payments to your bookmark app (monetize it)
- Add collaboration (invite others to your workspace)
- Add API (let others integrate with your app)
- Add mobile responsiveness (works great on phones)

**Learn new stacks:**

- Try Flask instead of Next.js (Python lovers)
- Try mobile with React Native
- Try Chrome extension with the same skills

**Timeline:** 1-2 weeks per project

### Option 3: Ship Something Today

**If you want to prove you can ship fast**, build micro-products:

**One-page tools:**

- Calculator for your industry
- Unit converter
- Form generator
- Simple game

**Landing pages:**

- For your newsletter
- For your services
- For a product idea (before building it)
- Template you can sell

**Timeline:** 4-8 hours

**Why this matters:** Shipping builds muscle memory. The more you ship, the easier shipping becomes.

---

## Join the Build to Launch Community

**You're not learning in isolation.**

**The Build to Launch newsletter** is where non-technical builders share what they're shipping, what they're stuck on, and what they're learning.

**What you'll get:**

**Weekly:**

- New AI building techniques
- Real builder case studies
- Common patterns and solutions
- Tool recommendations
- Framework comparisons

**Monthly:**

- Deep-dive tutorials
- Expert interviews
- Product teardowns
- Template libraries
- Community showcases

**Plus access to:**

- Monthly office hours with me
- Templates and boilerplates
- Exclusive deals on tools
- First access to new guides

**Why this matters:** Building alone is hard. Building with a community of people who understand your journey is energizing.

**Subscribe at:** buildtolaunch.substack.com

**Free tier:** Weekly tips and case studies **Premium tier:** $90/year - Everything plus templates, community, office hours

---

## Ideas for Your Next Build

Stuck on what to build? Here are proven ideas that solve real problems:

### For Your Current Job

**Internal tools** are the best first projects because:

- You know the problem deeply
- You have immediate users
- Feedback is instant
- Can prove value to your employer

**Examples:**

- Report generator (save 2 hours/week)
- Data dashboard (visualize KPIs)
- Form automation (reduce manual work)
- Booking system (replace spreadsheets)
- Knowledge base (organize team docs)

**Start Monday:** Identify one task you do manually that takes 30+ minutes per week.

### For A Specific Audience

**Vertical SaaS** - tools for specific industries:

- CRM for real estate agents
- Booking system for yoga studios
- Menu planner for dietitians
- Case tracker for lawyers
- Gig tracker for musicians

**Why these work:** Generic tools are too broad. Specific tools win specific audiences.

### Micro-SaaS

**Small, focused tools** that do one thing well:

- Email signature generator
- Screenshot beautifier
- Social media scheduler
- Link tracker
- Invoice generator

**Why these work:** Solve a specific pain point. Can charge $5-20/month. Grow organically.

### Content Tools

**Tools for creators:**

- Newsletter archive (like you built)
- Podcast transcript search
- YouTube video summarizer
- Twitter thread compiler
- Blog post generator

**Why these work:** Creators pay for tools that save time or improve content.

### Personal Productivity

**Tools you'd use:**

- Habit tracker
- Reading list manager
- Recipe organizer
- Expense tracker
- Learning journal

**Why these work:** If you'd pay for it, others will too.

---

## The Builder's Framework

Here's the process I use for every project:

### Week 1: Validate

**Before building anything:**

1. Talk to 5 people in target audience
2. Describe the problem (not the solution)
3. Ask: "How do you handle this now?"
4. Ask: "Would you pay $X to solve this?"
5. Document their exact words

**Red flag:** People say "cool idea" but won't pay **Green light:** People say "where do I sign up?"

### Week 2: Build MVP

**Absolute minimum features:**

- One core workflow
- Basic UI (ugly is fine)
- Manual onboarding (you email them login)
- No billing (pay later)

**Ship it** to those 5 people from week 1.

### Week 3: Learn

**Watch them use it:**

- Where do they get confused?
- What features do they ask for?
- What do they use most?
- What do they ignore?

**Iterate based on behavior**, not opinions.

### Week 4: Polish

**Now make it good:**

- Fix the confusing parts
- Add the requested features
- Polish the UI
- Set up billing
- Write real landing page

**Launch to more people.**

### Ongoing: Grow

**Focus on:**

- One acquisition channel at a time
- Talk to every user who cancels
- Build what people pay for
- Ignore feature requests from non-payers

**This framework works** because you don't waste time building things nobody wants.

---

## Shipping Consistently

The difference between people who build once and people who build careers is consistency.

### How to Ship Every Week

**Sunday:** Brainstorm ideas (30 min) **Monday:** Pick one, validate with 3 people (1 hour) **Tuesday-Thursday:** Build core feature (3-4 hours each) **Friday:** Deploy and share (2 hours) **Saturday:** Rest or respond to feedback

**That's 12-15 hours per week** to ship something new.

**Can't do weekly?** Ship monthly. Consistency matters more than frequency.

### How to Get Faster

**First project:** 40 hours (this guide) **Second project:** 25 hours (same patterns) **Third project:** 15 hours (reuse code) **Fourth project:** 10 hours (templates ready) **Fifth project:** 6 hours (mostly assembly)

**Speed comes from:** Recognizing patterns, reusing solutions, having templates, making fewer mistakes.

**By project 10**, you'll ship substantial apps in a weekend.

---

## Common Fears (And Why They're Wrong)

### "My idea isn't original"

**Good.** Unoriginal ideas have proven demand. You're not trying to invent Facebook. You're solving a real problem for a specific group of people.

**Every successful product** has 10+ competitors. They still succeed because execution and positioning matter more than originality.

### "I don't know if people will pay"

**Find out fast.** Build a landing page, write the copy, add a "Join Waitlist" button. Share it. If nobody signs up, don't build it. If people do, you have validation.

**Money talks.** Excitement is free. Paying customers are real.

### "What if it doesn't work?"

**Then you learned something.** Every failed project teaches you:

- What people actually want
- How to build faster next time
- What tech stack works for you

- How to ship despite fear

**Successful builders** have 10x more failed projects than successful ones. The winners are just the ones who kept building.

### "Real developers will judge my code"

**Real developers are too busy building.** The ones who judge aren't building anything worth judging.

**Also:** AI writes better code than most developers anyway. Your code quality is fine.

### "I should learn 'proper' programming first"

**No.** That's just fear disguised as professionalism. You'll learn programming concepts through building. That's how everyone actually learns - the tutorials are theater.

**Start building today.** Learn what you need when you need it.

---

## Resources You Might Need

### Learning Resources

**This guide** - Foundation **Next.js Docs** - When you need specific Next.js info (nextjs.org/docs) **Supabase Docs** - When you need database help (supabase.com/docs) **Tailwind Docs** - When you need specific styling (tailwindcss.com/docs)

### Communities

**r/SideProject** - Share what you're building **Indie Hackers** - Founders building in public **Next.js Discord** - Technical Next.js help **Supabase Discord** - Database and auth help

### Tools

**Claude Pro** - $20/month, best AI for building **Cursor** - $20/month, AI-powered code editor **Supabase** - Free tier, upgrade at $25/month **Vercel** - Free tier, rarely need to upgrade **Stripe** - Free, pay 2.9% + $0.30 per transaction

**Total monthly cost** to build: $40 **Total monthly cost** to run: $0 (until you have users)

### Inspiration

**Follow builders shipping:**

- @levelsio - Location independence, building in public
- @stephsmithio - From writing to building
- @dannypostmaa - Designer who builds
- @buildtollaunch - Me! Domain experts → builders

**The best way to learn** is watching others build and ship.

---

## My Challenge to You

**Build and deploy something in the next 7 days.**

It doesn't have to be big. It doesn't have to be original. It doesn't have to be perfect.

It just has to be **shipped**.

**Ideas:**

- A calculator for your industry
- A simple form that emails you results
- A landing page for something you might build
- A tool you'd use yourself
- A one-page game

**7 days from now**, you should be able to share a link that shows something you built.

**Why?** Because you'll prove to yourself that you can ship. And once you know you can ship, everything changes.

---

## Final Thoughts

You picked up this guide because you had an idea but thought "I can't code."

**Now you know:** You don't need to code. You need to describe what you want and let AI translate that into code.

**This changes everything:**

- Teachers can build classroom tools
- Consultants can build internal tools for clients
- Marketers can build landing pages without developers
- Designers can build their designs without handoff
- Anyone with domain expertise can build for their industry

**The barrier isn't technical anymore.** The barrier is deciding to start.

You've proven you can build. You built something real, deployed it, and learned the fundamentals.

**What you do next is up to you.**

You can:

- Build your idea
- Build for your job
- Build to learn
- Build to earn
- Build because it's fun

**All of these are valid.** The only invalid choice is not building because you "can't code."

**You can build.**

Go ship something.

---

## One Last Thing

Building can feel lonely. You'll have days where nothing works, where you question if this is worth it, where you want to quit.

**Those days are normal.** Every builder has them.

**What separates people who ship from people who don't** isn't talent or intelligence or even persistence. It's community.

**Join us:** buildtolaunch.substack.com

Share what you're building. Ask for help when you're stuck. Celebrate your wins. Learn from others.

**Building is more fun together.**

I'm excited to see what you create.

— Jenny

---

## Quick Links

📬 **Newsletter:** buildtolaunch.substack.com 🐦 **Twitter:** @buildtolaunch 📧 **Email:** jenny@buildtolaunch.com

---

## Keep Building

Remember:

**Vibe coding isn't about writing perfect code.** It's about shipping real products.

**You're not a "non-technical person trying to code."** You're a builder using modern tools.

**The goal isn't to become a developer.** The goal is to build products that matter.

**Start today. Ship this week. Repeat next week.**

Welcome to the community of builders.

Now go build something.

---

💌 **Subscribe at buildtolaunch.substack.com for weekly building tips, real builder stories, and exclusive templates**

---

## Acknowledgments

This guide wouldn't exist without:

**The vibe coding community** - For showing that domain expertise + AI > traditional programming

**The builders who ship** - For proving this works and sharing your journey

**AI tools that enable us** - Claude, ChatGPT, Cursor, and everyone pushing the boundaries

**You** - For taking the leap and proving you can build

Thank you for reading. Now go ship.

---

## Connect & Share

💌 **Newsletter**: Build to Launch - Weekly AI building tips, templates, and real builder stories

🦋 **Bluesky**: @jenny-ouyang - Daily insights

💼 **LinkedIn**: Jenny Ouyang - Professional network