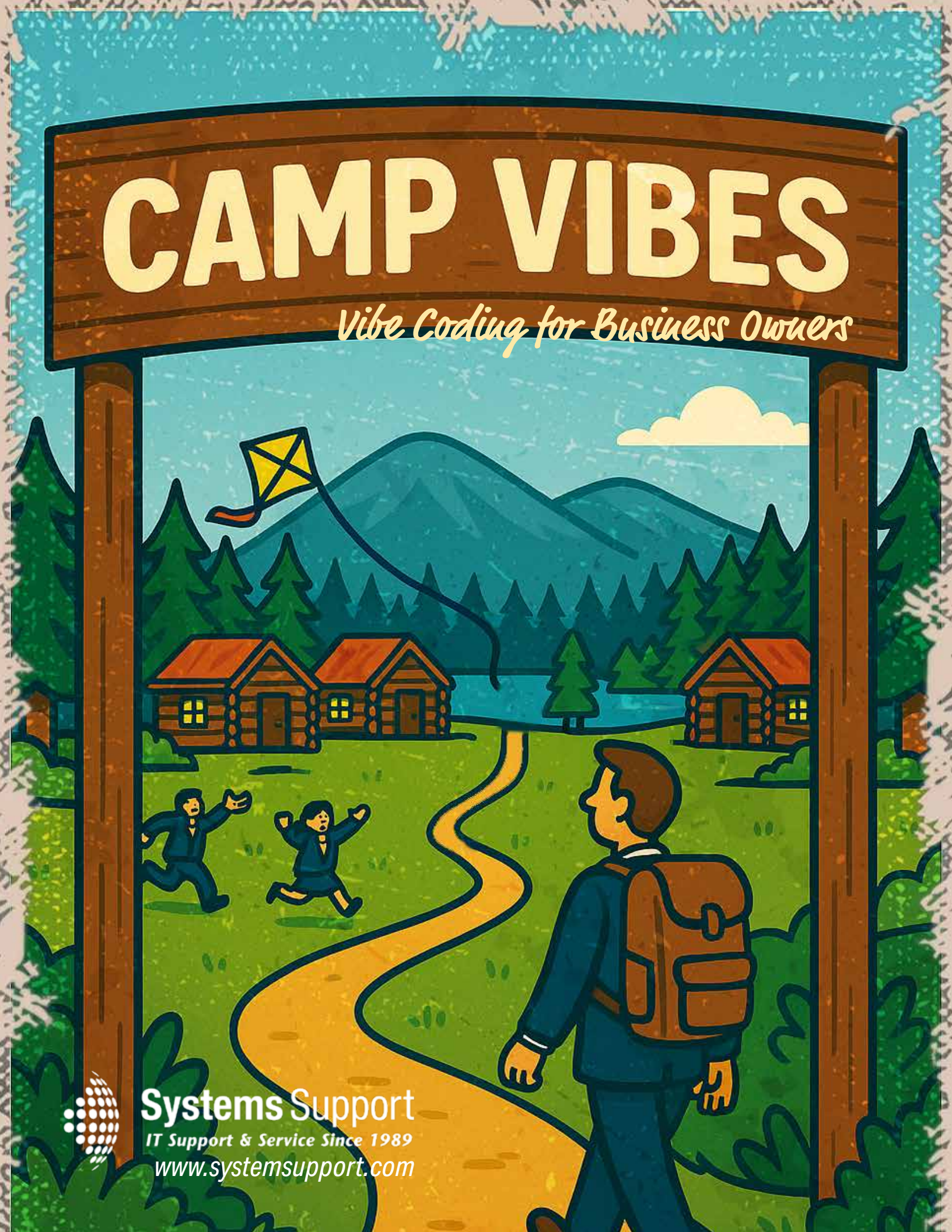

CAMP VIBES
Vibe Coding for Business Owners
Systems Support
IT Support & Service Since 1989
www.systemsupport.com

CAMP VIBES

Welcome to Camp Vibes, your month-long adventure into the world of vibe coding — where you can trade boardrooms for bonfires and still learn the skills to build smarter, faster, and more creatively with AI. Over the next few weeks, you'll collect merit badges, tackle fun challenges, and walk away with real tools and prompts you can use in your business immediately.

462 Plain St, Suite 206
Marshfield, MA 02050
(781) 653-7916
info@systemsupport.com

What is Camp Vibes?

Camp Vibes is a learning experience designed for small and medium-sized businesses curious about AI and automation. We combine the fun and nostalgia of summer camp (merit badges, campfire challenges) with the practical power of AI tools (ChatGPT, Claude, GitHub Copilot, Replit). By the end, you'll know how to:

- Write effective prompts using the GCES framework (Goal, Context, Source, Expectations)
- Build a real, working tool using AI-generated code
- Identify safe practices for experimenting with AI at work
- Understand when to DIY and when to call in the pro

What's Inside This Guide

1. A quick overview of vibe coding and why it matters for your business
2. The GCES framework for writing prompts that actually work
3. A step-by-step mini project to help you build your first AI-powered tool
4. Fun badge challenges to level up your skills
5. Resources and next steps to keep the learning going

What is Vibe Coding?

Vibe coding is a new way of creating software and automations using AI — where you describe what you want in plain language, and the AI generates the code for you. Instead of memorizing syntax or hiring a developer for every small project, you simply set the vibe (goal, style, and expectations) and let AI handle the heavy lifting.

- You talk, AI codes: "Build me a contact form with name, email, and message fields."
- You refine, AI improves: "Now add a confirmation message and style it with camp-themed colors."
- You test and deploy: A working tool in hours, not weeks.

The term was popularized in 2025 by Andrej Karpathy, who described it as "see stuff, say stuff, run stuff" — a creative, iterative approach that empowers non-developers to build real solutions.

Why It Matters for Small Businesses

Small businesses face constant pressure to do more with less — limited budgets, lean teams, and the need for fast solutions. Vibe coding:

- Cuts development time: Build internal tools, forms, or dashboards in days instead of months.
- Reduces costs: No need for full-time developers for simple tasks.
- Empowers teams: Marketing, operations, and admin staff can create their own solutions.
- Fuels innovation: Encourages experimentation — try new ideas without high stakes.

Real-World Success Stories

- Otto's Grotto (Shopify/Etsy Seller): Used ChatGPT to customize storefronts and automate workflows, doubling revenue in under a year — no coding background required.
- Solo Tech Entrepreneurs: Increasingly, startups launch using vibe coding as their primary build method — saving money and validating ideas faster than traditional dev cycles.
- Internal SMB Tools: From ticket dashboards to onboarding forms, vibe coding enables tailored solutions for unique small-business needs.

Important Cautions

While vibe coding is powerful, it's not magic — and it's not risk-free.

- **AI can hallucinate code:** Bugs, inefficiencies, or missing logic may occur.
- **Security gaps:** AI doesn't automatically enforce compliance or secure data handling.
- **Maintenance needed:** Quick prototypes can become hard to manage if used long-term.
- **Know your limits:** For sensitive systems (e.g., financial data, healthcare), involve IT professionals.

Takeaways:

Vibe coding is best for prototyping, internal tools, and quick wins — especially when paired with good prompting and safe practices. For SMBs, it's a gateway to innovation: fast, affordable, and surprisingly fun.

The Gear You Need and the Map to Use It

Before we dive into building, let's talk about the essentials you'll need to set up your "camp."

Core Tools

- **AI Assistants:** ChatGPT, Claude, or GitHub Copilot — for generating and refining code.
- **Code Sandboxes:** Replit, CodeSandbox, or Glitch — to run and test your code safely.
- **No-Code Automators:** Zapier, Make.com — for connecting your new tools to real workflows.
- **Security Checkers:** Snyk, SonarLint — to scan AI-generated code for vulnerabilities.

Tip: Think of these tools like your backpack — lightweight, portable, and ready for exploration.

Safety First:

Sandboxing Your Experiments

AI-generated code can be surprisingly powerful — but that means it can also be surprisingly risky if deployed without testing.

- Always experiment in a sandbox (isolated environment) before moving to live systems.
- Avoid using real customer data in your first builds.
- Review every line of code AI produces — you don't have to understand it all, but check for anything that looks suspicious.
- Document your prompts and results for future reference (and repeatability).

The GCES Prompt Framework

The GCES framework helps you write better prompts and avoid the "AI guesswork" trap. It ensures you provide enough detail for AI to give you exactly what you need.

1. Goal
What do you want AI to do?
Be specific: *"Create a contact form with name, email, and message fields."*

2. Context
What does AI need to know about your situation?
Example: *"We're a small IT firm serving compliance-heavy SMBs; the form is for onboarding new clients."*

3. Source
What references or examples can AI use?
Example: *"Follow the tone and layout of our existing support form."*

4. Expectations
What format and quality are you expecting?
Example: *"Provide clean HTML and CSS in one file, with comments explaining each section."*

Bad vs. Good vs. Better Prompt Example

Bad Prompt:
"Write code for a contact form."
Good Prompt:
"Write code for a contact form. It should include name, email, and message fields."
Better Prompt (GCES):

- **Goal:** Build a contact form with name, email, and message fields.
- **Context:** It's for a local IT firm's customer support page, must be mobile-friendly.
- **Source:** Follow our brand colors (green/blue) and layout similar to this example [link].
- **Expectations:** Clean HTML/CSS, include validation for required fields, under 150 lines.

Why GCES Works

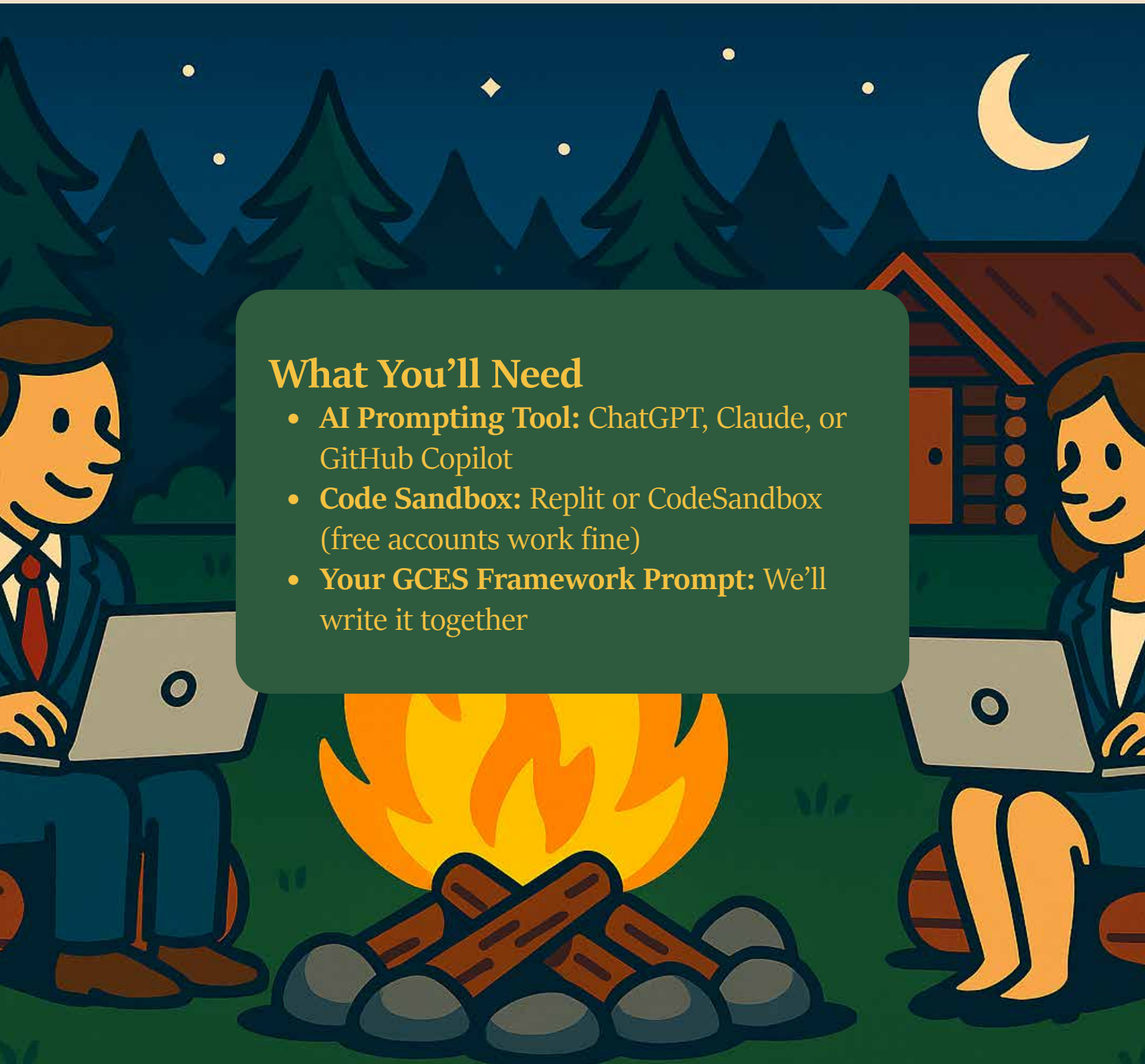
AI isn't a mind reader — but it's an excellent pattern matcher. GCES gives it enough detail to match your needs, while leaving room for creativity. More clarity = fewer revisions = faster builds.

The Camp Checklist App

Project Overview:

In this project, you'll use AI to generate a simple web form that collects user input — perfect for things like camp packing lists, event RSVPs, or onboarding checklists. You'll prompt AI to write the code, test it in a sandbox, and customize it to your needs.

This activity earns you your first Camp Vibes merit badge: Vibe Coder!



What You'll Need

- **AI Prompting Tool:** ChatGPT, Claude, or GitHub Copilot
- **Code Sandbox:** Replit or CodeSandbox (free accounts work fine)
- **Your GCES Framework Prompt:** We'll write it together

Step 1: Define the Goal (G in GCES)

Decide what your form will do. For this exercise:

"Create a Camp Checklist App that allows users to enter their name, email, and check off items to pack for camp. Include a submit button that shows a 'You're ready for camp!' message."

Step 2: Provide Context (C in GCES)

Tell AI what this is for and who will use it:

"This is for a small business learning AI coding. It should be beginner-friendly and have a clean, camp-themed design."

Step 3: Add Source (S in GCES)

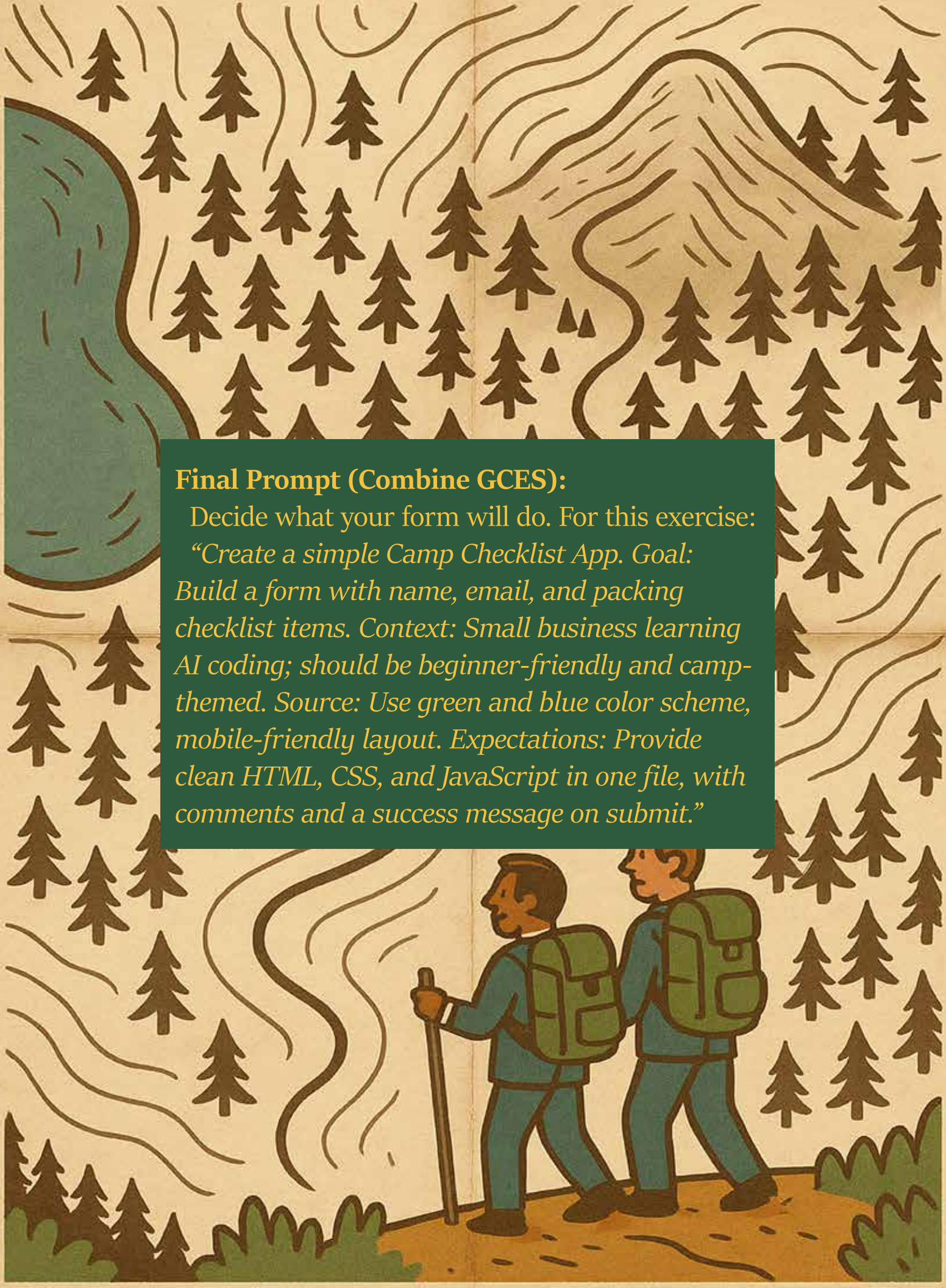
Point to any references or style ideas:

"Use a green and blue color scheme. Layout should be mobile-friendly and simple."

Step 4: Set Expectations (E in GCES)

Clarify what the final output should look like:

"Provide clean HTML, CSS, and JavaScript in one file, with comments explaining key sections. Keep it under 200 lines."



Final Prompt (Combine GCES):

Decide what your form will do. For this exercise:

"Create a simple Camp Checklist App. Goal: Build a form with name, email, and packing checklist items. Context: Small business learning AI coding; should be beginner-friendly and camp-themed. Source: Use green and blue color scheme, mobile-friendly layout. Expectations: Provide clean HTML, CSS, and JavaScript in one file, with comments and a success message on submit."

Step 5: Generate and Test the Code

1. Copy your prompt into ChatGPT or Claude.
2. Paste the generated code into Replit (or CodeSandbox).
3. Run it — check if the form works.
4. Adjust your prompt to refine colors, lay-out, or features.

Step 6: Customize and Add a Feature

- Change checklist items (e.g., “Laptop charger” Client contract”)
- Add more fields (e.g., “Arrival date”)
- Include validation (e.g., make email required)
- Style with a camp vibe: wood-texture backgrounds, rope dividers

Step 7: Show Off Your Badge

Congrats — you built your first vibe-coded tool! Award yourself the Vibe Coder Badge and share it with your team (or on LinkedIn with #CampVibesAI).

Optional Challenge: Level It Up:

- Connect the form to Zapier or Make.com to send email confirmations
- Add a dark mode toggle for “campfire night mode”
- Include a progress bar to track completed checklist items

Camp (And Coding) Safety

Vibe coding is fast and fun, but moving too quickly can introduce risks — from security vulnerabilities to compliance issues. These best practices keep your experiments safe, so you can innovate without worry.

1. Use a Sandbox Environment

- Always build and test in a sandbox like Replit or CodeSandbox — never your live site or production system.
- Treat vibe-coded prototypes as drafts, not production-ready solutions.
- Isolate test data to prevent accidental leaks or deletions.

2. Avoid Sensitive Data

- Don't feed personal information, passwords, or financial details into AI tools.
- Use fake or anonymized data for experiments (e.g., "Jane Doe," "Test Company").
- Check your AI tool's data retention policies before entering anything sensitive.

3. Review and Test AI Code Thoroughly

- AI can produce hallucinations (nonexistent libraries, broken logic).
- Manually review code before running it; highlight anything unfamiliar for extra scrutiny.
- Run small tests often to catch bugs early — don't wait until the end.

4. Security Scanning Tools

- Use tools like Snyk or SonarLint to identify vulnerabilities in AI-generated code.
- Watch for common risks: SQL injection, weak validation, outdated libraries.
- Document fixes or escalate concerns to IT professionals.

5. Compliance Awareness

- If you work in a regulated industry (healthcare, finance, legal), confirm that your prototype meets compliance requirements.
- When in doubt: keep experiments internal or consult your IT/security team before sharing externally.

6. Document Your Work

- Record prompts used, code iterations, and changes made.
- This helps troubleshoot issues and creates a knowledge base for future builds.
- Bonus: Documentation makes it easier to hand off prototypes to developers for production.

7. Know When to Call for Backup

- DIY is great for prototypes — but for scaling, security audits, or customer-facing deployments, involve your IT provider.
- Think of vibe coding as camp crafts, not the finished lodge — it's a starting point.