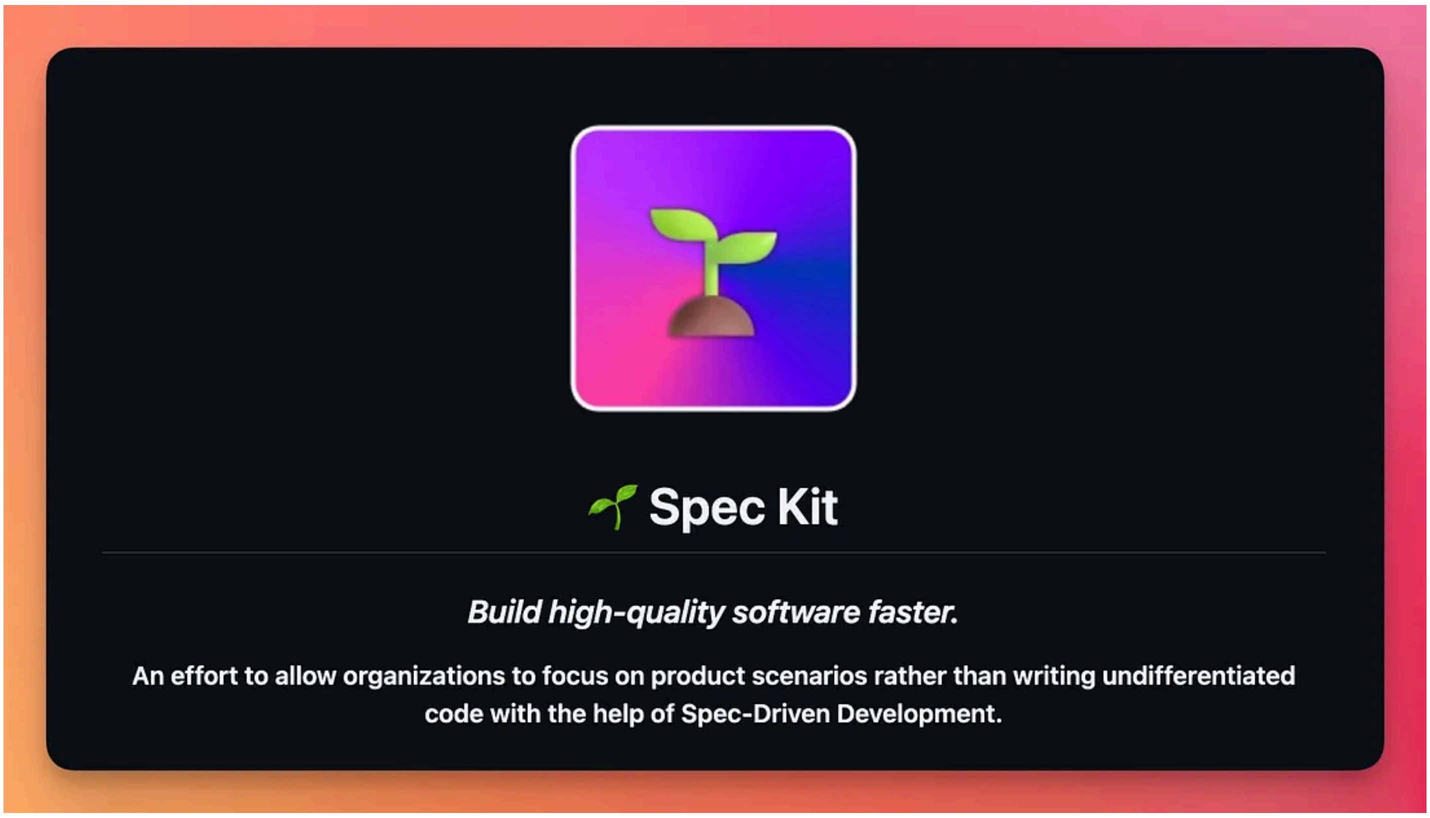
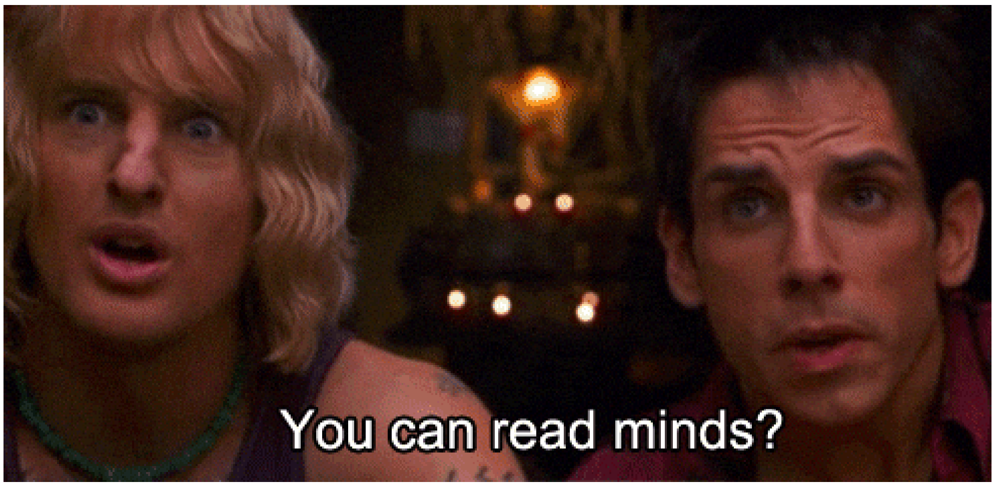
Stop Vibe-Coding: Ship What You Meant with GitHub's Spec Kit


Spec Kit
Build high-quality software faster.
An effort to allow organizations to focus on product scenarios rather than writing undifferentiated code with the help of Spec-Driven Development.

Spec Kit is GitHub's open-source toolkit for **spec-driven development**. It turns intent into executable artifacts for AI coding agents — replacing guessy prompts with a **structured, reviewable workflow**. By centering a clear specification, a technical plan, and small, testable tasks, Spec Kit reduces rework, bakes in constraints early, and delivers code that actually matches product intent across tools like Copilot, Claude Code, and Gemini.

The Problem: "Looks Right" Isn't Right

Modern AI agents are brilliant at pattern completion — but they're not mind-readers. Vague prompts often produce plausible code that quietly fails against real requirements and edge cases. Spec Kit swaps ad-hoc prompting for an explicit, gated flow — **Specify → Plan → Tasks → Implement** — so each artifact is validated *before* any code is generated.



What Is Spec Kit?

Spec Kit is a CLI-driven toolkit that operationalizes spec-driven development. It makes the specification the **central, executable source of truth** for AI-assisted coding. Teams focus on product scenarios and business logic, while AI handles undifferentiated implementation work **within clearly defined constraints**.

Why Now

Too many teams ship "right-looking" code that drifts from product intent because requirements, policies, and constraints are underspecified — or scattered across wikis and chats. Spec Kit pulls security, compliance, integration rules, and design system constraints **into the spec and plan** so those rules are enforced from day one.

How It Works: Four Gated Phases

Specify — Capture the *what* and *why*: user journeys, outcomes, constraints, and acceptance criteria.

Plan — Translate the spec into architecture, interfaces, and organization standards (security, design tokens, SLAs).

Tasks — Break the plan into small, testable units with explicit acceptance criteria.

Implement — Execute tasks with AI agents guided by the spec and plan, verifying outputs at each checkpoint.

Quickstart: From Zero to Specified

Spec Kit ships a CLI that initializes a project and scaffolds a spec-first workflow for your AI coding agent.

```
# Install and initialize
uvx --from git+https://github.com/github/spec-kit.git specify init <PROJECT_NA
```

After initialization, steer the agent with structured commands — /specify for intent, /plan for architecture, and /tasks for work breakdown — so code generation stays aligned with constraints.

Tips

- In **/specify**, describe outcomes and user journeys without prescribing implementation.
- In **/plan**, set architecture, interfaces, policies, and performance targets.
- In **/tasks**, generate an actionable, reviewable list that **gates** implementation.

Hands-On Demo: "Export to CSV" (End-to-End)

Scenario: Add **Export to CSV** to an analytics dashboard — cover data selection, formatting, pagination, and edge cases (empty results, large datasets).

Goal: Demonstrate **Specify → Plan → Tasks → Implement** while enforcing performance and UX constraints upfront to cut rework.

Specify

Write the feature's user journey, constraints (e.g., maximum rows per export), and acceptance criteria (correct headers, UTF-8 encoding, MIME type, streaming behavior past N rows) as a concise spec artifact.

Plan

Choose generation strategy (in-memory vs streaming), define API boundaries, rate limits, audit logging, and security posture. Align with design tokens/accessibility and document error states.

Tasks

Create a small, testable list: API route, service function, CSV serializer, pagination/streaming handler, content-disposition headers, integration tests, and a UX confirmation flow.

Implement

Let the AI agent generate code per task. Review against the spec and plan, run tests, iterate, and merge when acceptance criteria pass.

What It Creates in Your Workspace

The initialized workspace includes scripts and templates for spec, plan, and task artifacts. This **repeatable scaffolding** speeds onboarding and keeps the workflow discoverable *inside the repo*, not buried in external docs.

Works Across AI Toolchains

Spec Kit standardizes the artifacts that agents consume and produce. It can steer multiple coding agents — GitHub Copilot, Claude Code, Gemini CLIs, and more — so teams improve reliability **without locking into a single provider**.

When It Shines

- **Greenfield features** where intent and constraints should be nailed early.
- **Large codebases** where drift and rework are expensive.
- **Legacy modernization** where you must recapture business logic without importing old tech debt.

Separating the stable **what** from the flexible **how** lets teams evolve architecture safely while preserving intent over time.

Best Practices

- Keep specs **terse, testable, and tied to user journeys**. Acceptance criteria should map directly to tasks and tests.
- Treat the spec as a **living artifact**. When intent changes, update the spec, regenerate the plan, and resync tasks.
- Centralize **policies and constraints** (security, compliance, design system, SLAs) in the spec/plan so the AI enforces them at generation time.
- Use short, unambiguous language. Avoid pixel-level UI details unless they affect behavior or accessibility.

Common Pitfalls

- **Under-specifying behavior/data contracts** while over-specifying UI minutiae leads to rework and brittle code.

- **Skipping the Plan phase** creates local optima that clash with global constraints (you'll discover them late).
- **Task bloat** kills reviewability. Keep tasks small, testable, and tied to explicit acceptance criteria.

Team Adoption: Weaving It into Delivery

Map **Specify → Plan → Tasks** to your backlog refinement, sprint planning, and Definition of Done.

Attach artifacts to issues and PRs so reviews validate against the **spec and plan**, not just code style or local tests. You'll improve governance, auditability, and onboarding.

FAQ

Does Spec Kit lock us into one AI provider?

No. Artifacts are provider-agnostic, so you can steer Copilot, Claude Code, Gemini, and more.

Can this help with legacy modernization?

Yes. Capture the real business logic in a modern spec, design a clean plan, and let the agent rebuild with guardrails — without inheriting old tech debt.

How is this different from traditional specs?

Traditional specs are guidance docs. Spec Kit's **spec and plan are executable steering artifacts** consumed by agents, with tasks that gate implementation and verification.

Call to Action

Initialize a small repo, specify a single feature end-to-end this week, and measure rework and review time versus ad-hoc prompting. If the results are cleaner and faster (they will be), standardize the practice: add a spec template to your repos and make the workflow part of your Definition of Done.

Appendix: Example Prompts

Specify

/specify Add “Export to CSV” to the analytics dashboard with UTF-8 headers, pagination handling up to 50k rows via streaming, and explicit error states for empty results and timeouts. Include acceptance criteria and UX flows.

Plan

/plan Implement server-side streaming with a dedicated CSV serializer service, set rate limits, log exports for audits, and document API boundaries and security posture. Align with design tokens and accessibility standards.

Tasks

/tasks Break down API route, serializer, streaming handler, pagination, header formatting, MIME and disposition handling, integration tests, and UI confirmation — each with acceptance criteria and test checkpoints.

Mantra: intent first, then architecture, then tasks. That’s how you make AI agents ship the thing that was meant to be built — not just the thing that *looked* right in a prompt window.