DINHDUY TRAN

AI-assisted Development

No more “vibe coding”

The Problem: “Vide Coding”

“ SpecKit aims to solve the problems of inconsistent, unpredictable, and hard-to maintain code often generated by vague, iterative prompts to an AI assistant. “

What is GitHub SpecKit?

- An **open-source toolkit** (MIT-licensed) and methodology from GitHub that formalizes **Spec Driven Development (SDD)**.
- Brings structure and discipline to **AI-assisted development**.
- Make specifications a **living**, **central**, **executable artifacts**.
- The AI assistant (like Copilot, Cursor, Claude, etc…) uses this spec as its single "**source of truth**" to generate code
- Bundles **templates**, a CLI, **prompts/commands**, and agent-friendly workflows.

Key Goals & Features

Solves “Vide Coding”

Replace chaotic, ad-hoc prompting with a structured, gated workflow for predictable results.

AI Agentic-Agnostic

Designed to work with various AI coding assistants, such as GitHub Copilot, Claude, Cursor, Gemini, and more.

Human-in-the-Loop

Enforce explicit checkpoints for a Human developer to review, refine, and approve all AI-generated artifacts.

The Core Artifacts

constitution.md (The “Rules”)

Define the “non-negotiable” project principles. This Include coding standards, preferred frameworks, security requirements, and testing methodologies. The AI must adhere to this.

spec.md (The “What”)

Details the feature, user stories, goals, and edge cases. It focuses on what to build, not how to build it

plan.md (The “How”)

An AI-generated technical implementation plan. Based on the spec and constitution, it outlines the architecture, data models, and file structures.

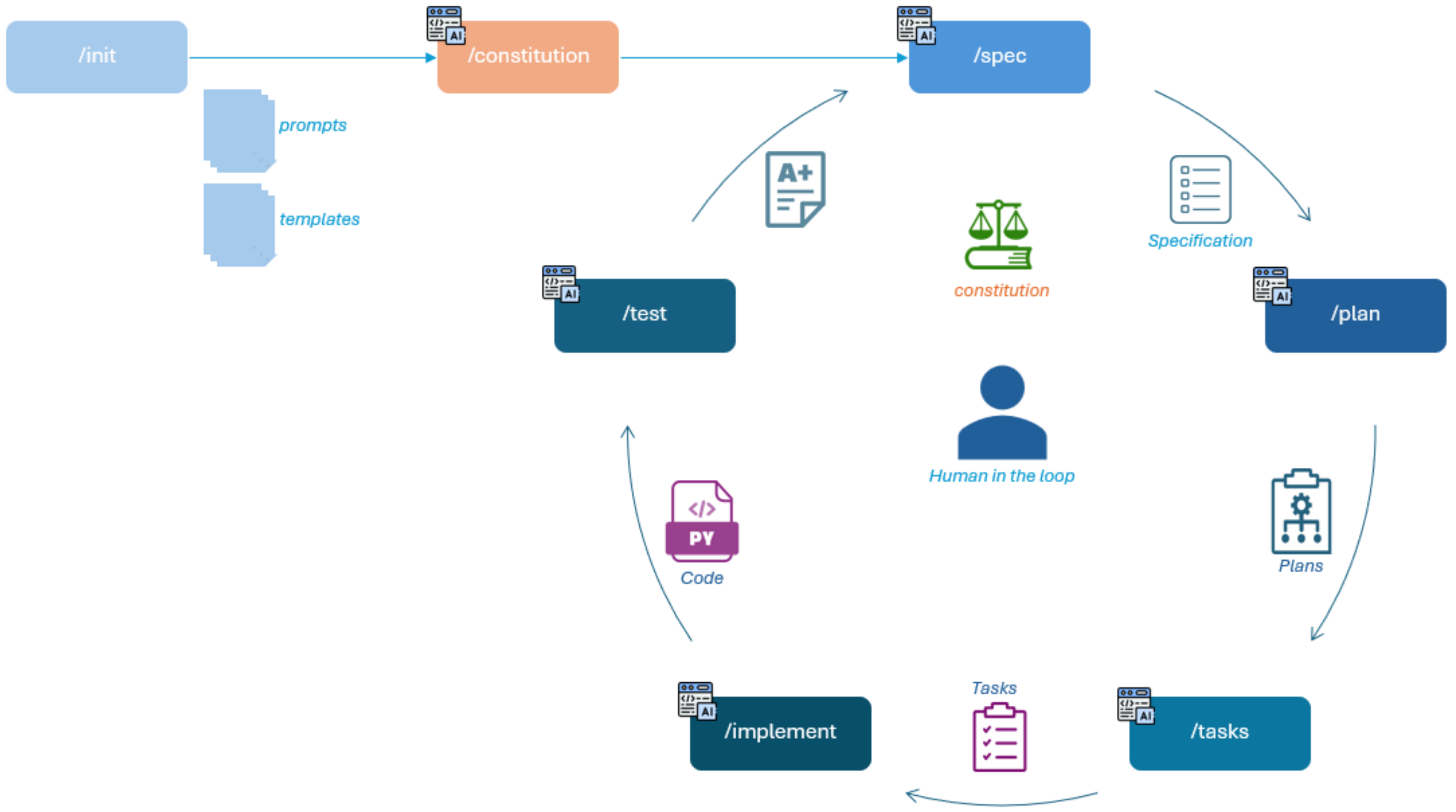
tasks.md (The “Checklist”)

A detailed, step-by-step checklist of small, testable, and verifiable tasks that the AI will follow to implement the plan.

The Spec-Driven Workflow

1. **Define:** Teams first define a clear **spec**.
2. **Plan:** Derive a technical **plan** from that spec.
3. **Breakdown:** Break the plan into small, testable **tasks**.
4. **Execute:** Use code-generation agents to **implement** each task.
5. **Verify:** Agents **verify** the implementation of each task.

SpecKit Workflow



The Goal & Value

- **Shifts away from:** Ad-hoc “vibe-coding.”
- **Moves toward: Reproducible** and **verifiable** development using AI assistance.
- The project is actively maintained and supported by documentation and a public site.

Core components and concepts

Executable specifications: Specs are written as living artifacts that drive planning, implementation, and verification rather than being discarded after design.

CLI and templates: A command-line toolkit plus project templates to scaffold spec-driven projects and agent interactions.

AI agent integration: Recipes and prompts designed to work with multiple code-generation agents (Copilot, Claude, Gemini CLI, etc.) so tasks can be delegated reliably to AI assistants.

Task decomposition and verifiability: The workflow emphasizes small, testable tasks and automated verification to reduce brittle, non-reproducible outputs from generative models.

Community and docs: Official website and GitHub repo provide quick-start guides, examples, and agent recipes to adopt the pattern

How Spec Kit changes the development loop

1. **Specification first**: capture intent, acceptance criteria, constraints, and nonfunctional requirements in a machine-readable spec.
2. **Plan generation**: produce a concrete technical plan from the spec (architecture, interfaces, dependencies) so generated code aligns with intent.
3. **Tasking**: split the plan into minimal tasks with clear inputs/outputs and tests so AI agents can implement each reliably.
4. **Implementation by agents**: run agents against individual tasks using standardized prompts and the CLI to produce code and tests.
5. **Automated verification**: run the generated tests and integration checks; failed tasks are iterated until they pass, creating a verifiable history from spec to working code

Starting a New (Greenfield) Project

Step 1: Installation and Initialization

Step 2: Define the Constitution

Step 3: Specify the Feature

Step 4: Review and Clarify the Spec

Step 5: Generate the Technical Plan

Step 6: Review the Plan

Step 7: Generate Tasks

Step 8: Implement the Tasks

Step 9: Test and Review

Step 1: Installation and Initialization

First, you must install the SpecKit CLI (often via a tool like uv). You then initialize your project.

1. Install Spec Kit:

```
uvx --from git+https://github.com/github/spec-kit.git specify init <YOUR_PROJECT_NAME>
```

2. Initialize Project:
   - .github/ for agent prompts
   - .specify/ for specs) and the core artifact files

Step 2: Define the Constitution

Before you build anything, you set the rules.

1. Navigate to the generated */memory/constitution.md* file.
2. Edit this file to define your project's standards. This is your one-time setup to guide all future AI work.
   - *Example* "All new API endpoints must include unit tests." or "Use React with TypeScript for all front-end components."
3. You can also use an AI command like */speckit.constitution* to help you draft these principles.

Step 3: Specify the Feature

Now, you define your first feature, focusing on *what* it is, not *how* it's built.

1. In your AI assistant, use the specify command: /speckit.specify
2. Provide a high-level prompt describing your feature.
   - Example: /speckit.specify Build a user authentication system that allows users to sign up with an email and password and log in.
3. The AI will generate a detailed spec.md file, including user stories, acceptance criteria, and a "Needs Clarification" section.

Step 4: Review and Clarify the Spec

This is a critical human-in-the-loop step.

1. Open the newly created **spec.md** file.
2. Manually review the AI's interpretation of your feature. Pay close attention to the "Needs Clarification" or "Edge Cases" sections.
3. Use the **/speckit.clarify** command or simply chat with your AI to answer these questions and refine the spec until it's accurate and complete.

Step 5: Generate the Technical Plan

Once the "what" is locked, you generate the "how.“

In your AI assistant, use the plan command: /speckit.plan

The AI will read the constitution.md and the approved spec.md.

It will generate a plan.md file, which includes the proposed architecture, data models, API endpoints, and a list of files to be created or modified.

Step 6: Review the Plan

You must review the AI's technical proposal.

1. Open **plan.md** and any related files (e.g., **data-model.md**).
2. Verify that the technical approach aligns with your constitution and best practices.
3. Request changes from the AI until you approve the plan.

Step 7: Generate Tasks

Break the plan into small, actionable steps.

1. In your AI assistant, use the tasks command: /speckit.tasks
2. The AI will read the plan.md and generate a tasks.md file, which is a detailed checklist.
   - Example:
     - "1. Create file src/models/user.js..."
     - "2. Add email and passwordHash fields to User schema..."

Step 8: Implement the Tasks

This is where the AI writes the code.

In your AI assistant, use the implement command: /speckit.implement

The AI will now execute every task from tasks.md one by one, writing and saving the code to your file system.

Your role is to monitor the process, answer any questions the AI has, and intervene if it gets stuck.

Step 9: Test and Review

Once implementation is complete, you review the final output, run tests, and merge the code.

Adding a New Feature (Brownfield) Project

Step 1: Installation and Initialization

Step 2: Start a New Feature Branch

Step 3: Specify the New Feature

Step 4: Review and Clarify the Spec

Step 5: Generate the Plan

Step 6: Review the Plan

Step 7: Generate Tasks

Step 8: Implement the Tasks

Step 9: Test and Merge

Best practices and recommendations

- Keep **tasks small** and strictly **testable** to minimize agent hallucination and increase **verifiability**.
- Provide rich, precise **context** to agents: relevant files, small code excerpts, and test harness snippets reduce churn.
- Treat generated code as reviewable artifacts: combine automated tests with human review for design and security decisions.
- Maintain traceability: link tasks, commits, and PRs to spec IDs so the spec is the source of truth for why changes exist

Known limitations and considerations

- AI agents can still produce **incorrect** or **insecure** code; the approach **reduces** but does not **eliminate** the need for expert **review** and security scanning.
- Adoption requires cultural shift: teams must invest in authoring **high-quality specs** and in **CI** that enforces **verification** for each generated task.
- Tooling maturity and integrations **vary by agent** and language; check the Spec Kit repo for current supported workflows and community-contributed adapters

Where to learn more

Spec Kit - AI-Powered Specification-Driven Development Toolkit

https://speckit.org/

GitHub - github/spec-kit: Toolkit to help you get started with Spec ...

https://github.com/github/spec-kit

Spec-driven development with AI: Get started with a new open source ...

https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/

GitHub Open Sources Kit for Spec-Driven AI Development

https://visualstudiomagazine.com/articles/2025/09/03/github-open-sources-kit-for-spec-driven-ai-development.aspx

Diving Into Spec-Driven Development With GitHub Spec Kit

https://developer.microsoft.com/blog/spec-driven-develop

GitHub Spec Kit: A Guide to Spec-Driven AI Development

https://intuitionlabs.ai/articles/spec-driven-development-spec-kit

GitHub's New SpecKit Guide : The Future of AI-Assisted Software Development

https://www.geeky-gadgets.com/github-speckit-ai-coding-tool/

Inside Spec-Driven Development: What GitHub's Spec Kit Makes Possible ...

https://www.epam.com/insights/ai/blogs/inside-spec-driven-development-what-githubspec-kit-makes-possible-for-ai-engineering

Spec-Driven Development with GitHub's SpecKit: The Future of AI-Powered ...

https://innohub.powerweave.com/?p=385