# AIDD-30-DAY-CHALLENGE-TASK-7

## *SPECKit Plus*

Spec-Kit Plus is a structured framework based on Specification-Driven Development with Reusable Intelligence (SDD-RI), designed to ensure that every software feature you build produces not only functional code but also long-lasting "intelligence" such as reasoning patterns, architectural decisions, and reusable prompts. It provides clear workflows, templates, and quality-gated phases—from specification and planning to implementation and intelligence capture—so that teams work in a disciplined, repeatable way. By separating temporary project code from permanent knowledge (like ADRs and PHRs), Spec-Kit Plus helps developers accumulate a growing library of insights and tools that make future development faster, more consistent, and more strategic.

## *Core Concept*

Spec-Kit Plus is organized around five slash-commands. Each represents a stage in the SDD-RI workflow and a different type of intelligence you are capturing

### 1. /Constitution:

In Spec-Kit Plus, the /Constitution is the foundational document that establishes the universal rules, standards, and quality expectations for an entire software project. It defines non-negotiable guidelines—such as coding conventions, testing requirements, architectural principles, documentation formats, and security practices—that every feature must follow throughout the development lifecycle. Created once at the beginning of a project, the Constitution acts as a governing contract that ensures consistency, clarity, and discipline across all specifications, plans, and implementations. By setting these project-wide standards upfront, the /Constitution helps teams maintain coherence and quality while enabling AI agents and developers to work in a structured, predictable environment.

### 2. /Specify:

In Spec-Kit Plus, /specify is the command that initiates the specification phase for a new feature, creating a structured document that clearly defines what the feature should do before any planning or coding begins. It captures the feature's purpose, user stories, functional and non-functional requirements, acceptance criteria, constraints, and success conditions. When /specify is triggered, the system also generates a dedicated feature folder and Git branch, ensuring all work related to that feature remains organized and traceable. This phase focuses entirely on clarity of intent—describing what is needed and why—without discussing technical implementation, thereby providing a solid foundation for accurate planning and development.

### 3. /Plan:

In Spec-Kit Plus, /plan is the command that transforms a completed specification into a clear, actionable development plan by breaking the feature into well-defined tasks and outlining how the work will be executed. During this phase, the system analyzes the requirements from the specification, identifies the steps needed to implement them, determines dependencies, and organizes the work into a structured task list that can be assigned, tracked, and executed. The /plan step ensures that the team understands how the feature will be built before implementation begins, creating a bridge between high-level intent and concrete development work.

### 4. /Tasks:

In Spec-Kit Plus, /tasks is the command that takes the planned breakdown of work and converts it into a structured, execution-ready task list that developers or AI agents can follow during implementation. It organizes each task with clear descriptions, expected outcomes, dependencies, and any required inputs, ensuring that all work is actionable and aligned with the specification and plan. The /tasks phase serves as the final preparation step before coding begins, turning the high-level plan into precise units of work that enable a smooth, efficient, and traceable implementation process.

## 5. /implement:

In Spec-Kit Plus, /implement is the command that begins the actual coding phase, using the prepared tasks and plan as the blueprint for generating working software. During this stage, the system or developer follows each task to produce code, tests, documentation, and other deliverables while ensuring full compliance with the specification and Constitution. The /implement phase focuses on turning structured intent into functional, high-quality output, completing the feature in a controlled and traceable manner that aligns with all earlier design decisions.