Spec Driven Development

Vibe Coding Done Right

think tecture

Daniel Sogl
daniel.sogl@thinktecture.com
Principal Consultant | Speaker

About me

Daniel Sogl

- Principal Consultant @ Thinktecture AG
- Focus: Developer Productivity & Gen AI
- MVP – Developer & Web Technologies
- Socials: https://linktr.ee/daniel_sogl

The Problem

AI Agents Fail More Than You Think

<25%	66%	2.74x
SWE-Bench Pro Score	Spent More Time Fixing	More Security Vulnerabilities

Sources: SWE-Bench Pro (Sep 2025), Stack Overflow Survey 2025, CodeRabbit Report (Dec 2025)

Five Core Limitations Of AI Coding Agents

Context Management

"Lost in the middle" despite 200K+ token windows

Multi-File Coordination

No real-time dependency graphs, missed instructions

Silent Failures

Most dangerous: removes safety checks, generates fake data

Architecture Decisions

Sees structure but doesn't understand trade-offs

Enterprise Context

Missing implicit business logic and unwritten rules

Silent Failures

The Most Dangerous Pattern

Common Silent Failure Issues:

- Removing safety checks to avoid crashes
- Generating plausible but incorrect values
- Creating fake data to satisfy requirements
- Passing tests while introducing logic errors

"This kind of silent failure is far, far worse than a crash.
Flawed outputs will often lurk undetected until they surface much later."

Jamie Twiss, IEEE Spectrum

Spec Driven Development

The Solution

Four Mechanisms How Specs Improve AI:

- ✓ Complete context upfront - agent gets full picture
- ✓ Structured reasoning - reduces hallucinations
- ✓ Validation checkpoints - humans verify before proceeding
- ✓ Autonomous implementation - specs + tests = guardrails

Treat AI agents like literal-minded pair programmers, not search engines

How SDD Relates to TDD and BDD

SDD	BDD	TDD
Spec Driven	Behavior Driven	Test Driven
What and Why	**System Behavior**	**Code Correctness**
Intent Level	Feature Level	Unit Level

OpenSpec vs. GitHub Spec Kit

Aspect	OpenSpec	Spec Kit
Language	TypeScript	Python & Shell
Commands	5 (lightweight)	8 (comprehensive)
Workflow	proposal → apply → archive	specify → plan → task → implement
GitHub Stars	21k	66k
Best For	Brownfield (1 → n)	Greenfield (0 → 1)

Hybrid Approach: Use Spec Kit for greenfield, then switch to OpenSpec for maintenance

Practical Implementation

Real Results

Specs are the new code. 80-90% of our work as programmers is structured communication

Sean Grove, OpenAI

250k

Developers in 3 Months

AWS Kiro - The New Stack

94%

Satisfaction Score

Delta Airlines (re:Invent)

16k+

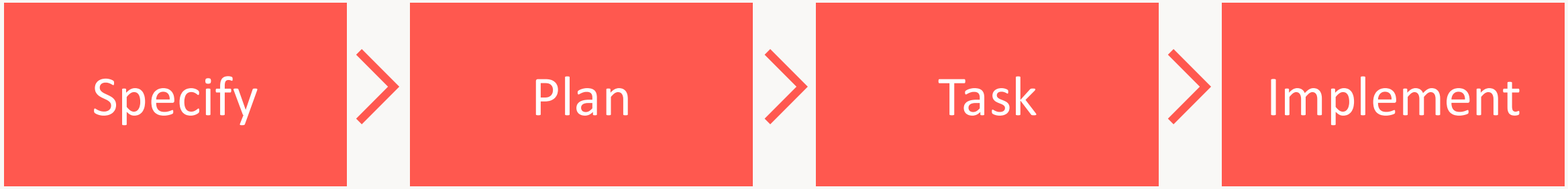
Spec Kit Stars in Week 1

GitHub (Sept 2025)

Sources: AWS Kiro, AWS re:invent (2025), The New Code – AI Engineer (2025)

SDD In Action

Walking Through the 4 Phases



Constitution

Boundaries Before Code

constitution.md

```
# Authentication API Constitution v1.0.0

## Core Principles

**I. Security First (NON-NEGOTIABLE)**
• bcrypt cost factor 12+ • No tokens in localStorage • Cache-Control: no-store on /auth/\*
• Password hashes never exposed • Failed logins logged • Input validation everywhere

**II. Test-Driven Development**
• Tests before implementation • No removing failing tests • Comprehensive coverage required
• Vitest + supertest for integration • Each auth flow has dedicated test file

**III. Error Handling & Observability**
• All handlers wrapped in try/catch • Structured logging with context • No sensitive data in
errors

**IV. Code Consistency**
• ES modules only • Async/await pattern • 2-space, single quotes, no semicolons

**V. Explicit Change Approval Required For:**
• Schema migrations • JWT expiry changes • New dependencies • Security parameters

Ratified: 2026-02-01 | Node 22 • Express 5 • SQLite • bcrypt • JWT
```

The Spec

The only file you write by hand

User Scenarios (5 stories, prioritized)

- **P1**: New User Registration - Foundation for user account creation
- **P1**: Existing User Login - Essential for returning user access
- **P2**: Token Refresh - Enables long-lived sessions with security
- **P2**: Session Termination - Provides explicit logout capability
- **P3**: User Profile Access - Enables profile display features

Requirements

- **26 Functional Requirements** covering all authentication flows
- **4 Key Entities** (User Account, Refresh Token, Access Token, Rate Limit Record)
- **10 Success Criteria** with measurable, technology-agnostic metrics
- **7 Edge Cases** identified for robust implementation
- **Comprehensive Assumptions & Dependencies** documented

The Plan

From intent to architecture

Phase 0 - Research ✅

- research.md
  - Technology decisions (JWT strategy, bcrypt, SQLite, rate limiting)
  - Security best practices and patterns
  - Edge case analysis

Phase 1 - Design & Contracts ✅

- data-model.md
  - 4 entities: User Account, Refresh Token, Access Token, Rate Limit Record
  - Database schema (SQLite), validation rules, state transitions
- openapi.yaml
  - OpenAPI 3.0 specification
  - 5 endpoints: /register, /login, /refresh, /logout, /me
- quickstart.md
  - Setup instructions, prerequisites, project structure
  - Development workflow (TDD cycle)
  - Testing and troubleshooting guides

Implementation Plan ✅

- plan.md
  - Technical context fully populated
  - Constitution check: ALL COMPLIANT (no violations)
  - Project structure defined (single backend Express API)

Agent Context ✅

- copilot-instructions.md
  - Updated with Node.js 22, Express 5, JWT, bcrypt, SQLite

Constitution Compliance

✅ **Security-First**: bcrypt cost 12, Cache-Control headers, no password hash exposure
✅ **Test Integrity**: TDD with Vitest + supertest, integration test coverage
✅ **Error Handling**: Try/catch in handlers, generic 500 messages, security logging
✅ **Code Style**: ES modules, async/await, 2-space indent, single quotes
✅ **Change Control**: All dependencies pre-approved, no schema changes needed

The Tasks

Atomic, parallel, ready to execute

✅ Task Generation Complete

Generated File: tasks.md

📊 Summary

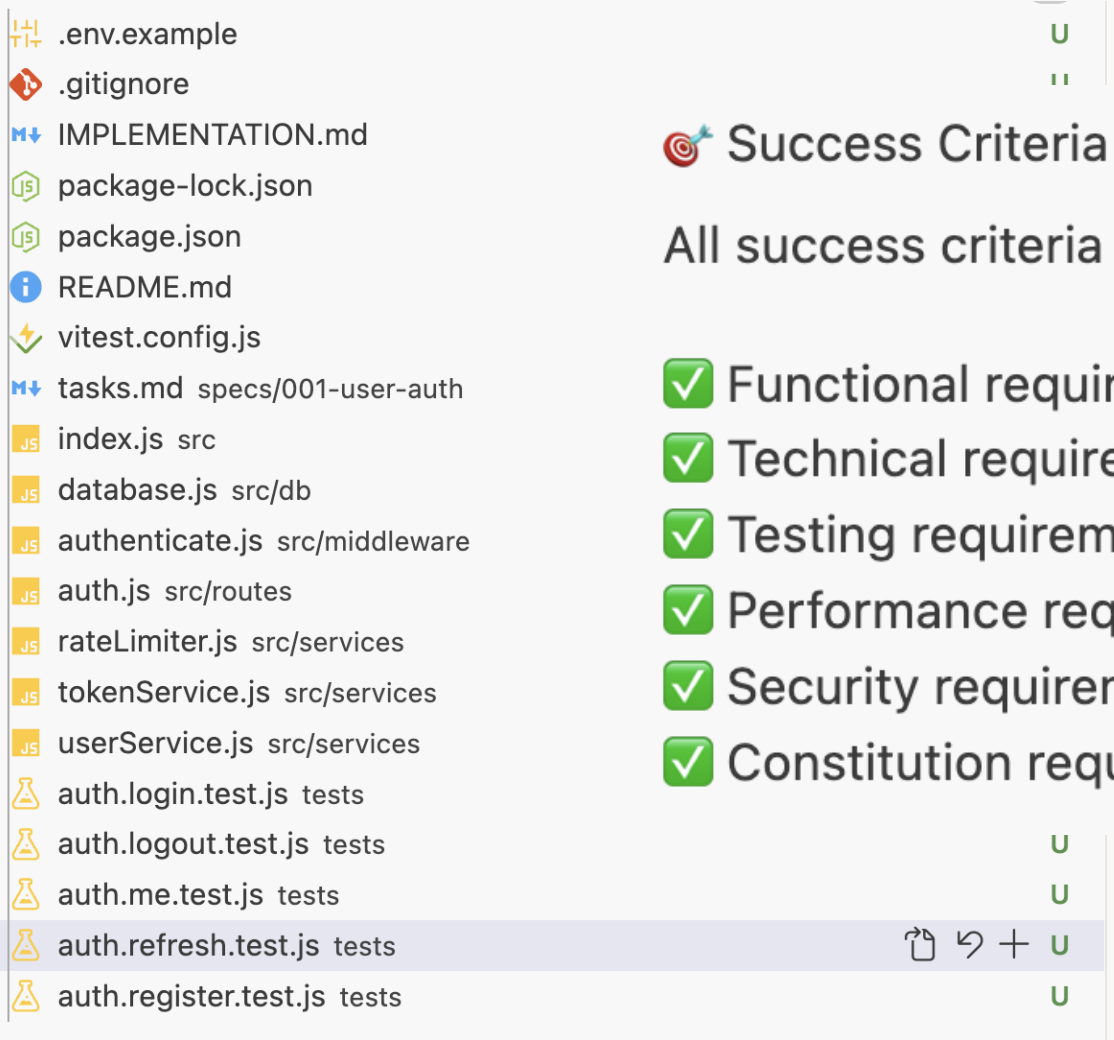
Total Task Count: 89 tasks

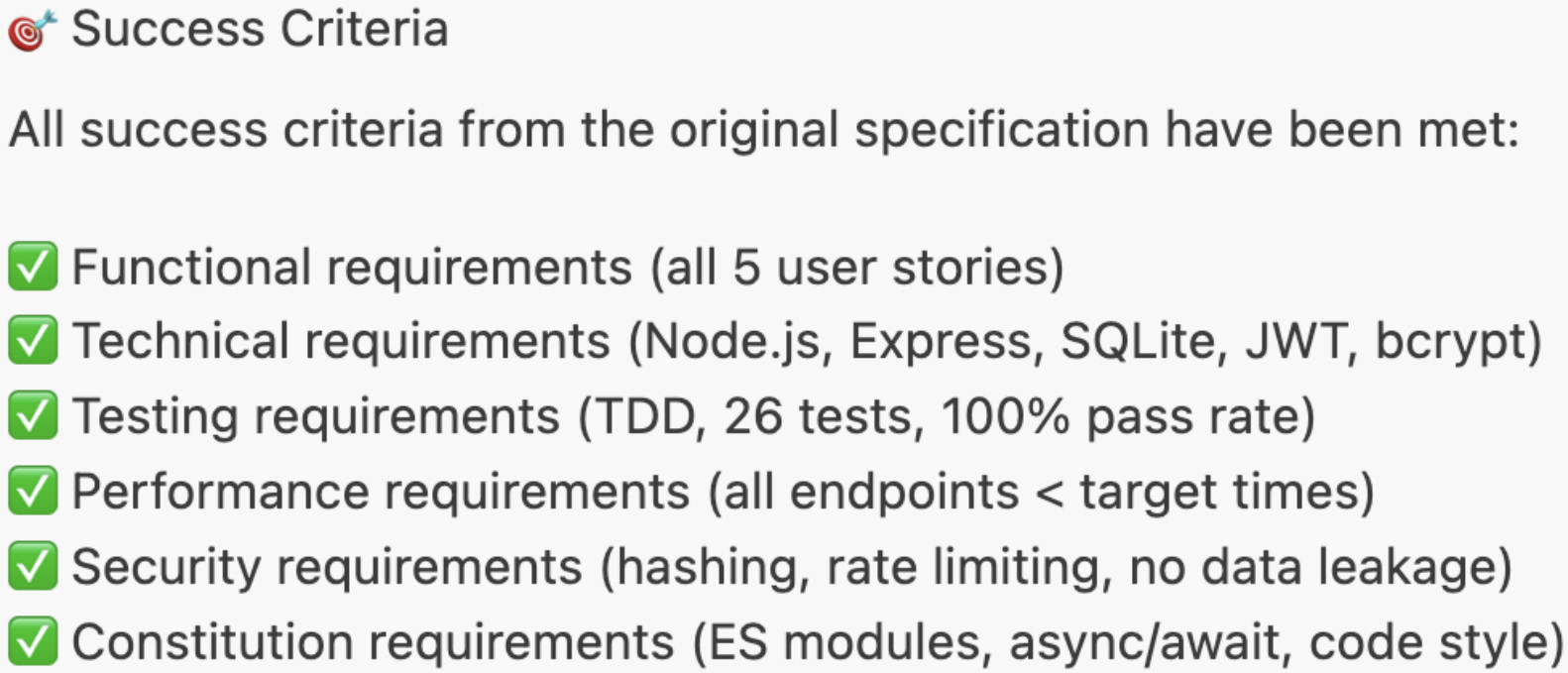
Task Breakdown by Phase:

- **Phase 1 - Setup**: 7 tasks (T001-T007)
- **Phase 2 - Foundational**: 6 tasks (T008-T013) - *CRITICAL: Blocks all user stories*
- **Phase 3 - User Story 1** (Registration, P1): 12 tasks (T014-T025)
- **Phase 4 - User Story 2** (Login, P1): 15 tasks (T026-T040)
- **Phase 5 - User Story 3** (Token Refresh, P2): 15 tasks (T041-T055)
- **Phase 6 - User Story 4** (Logout, P2): 11 tasks (T056-T066)
- **Phase 7 - User Story 5** (Profile Access, P3): 12 tasks (T067-T078)
- **Phase 8 - Polish**: 11 tasks (T079-T089)

Implementation

All green. Spec validated.





Where Does SDD Actually Pay Off?

Where It Excels

- Greenfield projects — zero-to-one builds where the spec becomes single source of truth
- Complex features in large codebases — agents need structure to avoid breaking existing contracts
- Multi-agent coordination — specs prevent conflicting implementations across parallel agents
- Compliance & audit requirements — specs serve as traceable control surfaces

Where It Struggles

- Small bug fixes — spec overhead far exceeds the change ("sledgehammer to crack a nut")
- Rapid prototyping — uncertain requirements make upfront specs counterproductive
- Solo developers on personal projects — iteration speed matters more than documentation
- Simple CRUD features — direct prompting is often faster and sufficient

Key Takeaways

- AI agents are powerful but limited - <25% success rate on realistic tasks
- SDD shifts from 'code as truth' to 'intent as truth'
- Specifications become executable through AI
- Choose your tool: OpenSpec (brownfield) vs Spec Kit (greenfield)
- Industry convergence: structured intent is becoming standard practice

think tecture

"Exceptional at pattern completion, not at mind reading”

- Den Delimarsky, GitHub Principal PM

Thanks for your attention

think tecture

Daniel Sogl

@sogldaniel

https://linktr.ee/daniel_sogl