The Product Context Repo

Customer evidence in. Grounded artifacts out. Slop starved.

Prakash Poudel Sharma · 2026

The demo that starts every AI conversation

Ask any AI assistant: "Write a user story for my product."

You get back something correctly formatted, INVEST-shaped — and hollow.

It doesn't know your permissions model. It doesn't know what "export" means in your contracts. It doesn't know you killed that exact feature in 2024.

The model isn't missing intelligence. It's missing context.

Where your context actually lives

- Support tickets → the help desk (Zendesk, Intercom, Freshdesk, …)
- Call recordings → a recorder (Gong, Chorus, Fireflies, Grain, …)
- Deal notes, lost reasons → the CRM (Salesforce, HubSpot, Pipedrive, …)
- Decisions → docs, chat threads, and whoever was in the room

Six silos. None cross-readable. None versioned. None AI-pointable.

Engineers hit this exact wall with coding agents — and fixed it with **context files, skills, and hooks in a repo**.

The thesis

A **product context repo**: a plain git repository of Markdown files holding what your product knows about itself.

- **Readable by everything** — every AI tool ingests Markdown, today
- **Versioned** — decisions have dates, language has history, diffs get read
- **Owned** — pull requests keep it deliberate; and the PR gate is what lets agents write to it safely

A prompt is context you retype. A repo is context you write once and version.

Arc one: the layout (Part 2)

```
product-context/
├── AGENTS.md          # how AI tools read this repo
├── product/areas/     # one file per area, with sharp edges
├── glossary.md        # what words mean HERE
├── decisions/         # dated records — the rot-proof part
├── signals/           # customer evidence, structured
├── templates/         # story, PRD, test-register formats
└── skills/            # the workflows, as versioned files
```

Design rule: **structure for retrieval, not writing.** Small files, one question each.

Every area file: **Sharp edges** + **What we will NOT do** — the facts that make naive drafts wrong.

The signal pipeline (Part 3)

Signals are the only part of the repo **customers write**. One card = one observation:

```
id: SIG-2026-0142          # citable
source: support-ticket     # ticket | call | sales-call | crm-note …
segment: mid-market
type: pain
severity: 3
```

Verbatim (sacred) + **interpretation** (separate, revisable). Anonymized at creation.

A scheduled agent extracts tickets, transcripts, and CRM lost-reasons weekly → opens a **pull request**. Your cost: a twenty-minute review.

The digest section that pays rent: **"Contradicts what we believe."**

Connecting the tools (Part 4)

Don't sync systems. **The assistant is the integration.**

System	Owns
Context repo	What the product is, means, decided
Tracker (Jira, Linear, ...)	Work items and state
Docs (Notion, Confluence, ...)	Drafts and collaboration
Chat (Slack, Teams, ...)	Real-time discussion

MCP makes all of them readable in one session. **Read wide, write narrow:** tracker writes → `ai-drafted` label + triage state; repo writes → PRs; chat → never autonomous.

Before connecting help desk / CRM / recorder: the privacy conversation. **Know the answer is yes before the connection exists.**

Skills, hooks, always-on (Part 5)

Skills — workflows as files (open standard, portable across assistants):
story-writing, prd-drafting, signal-extraction. The third time you type a prompt, it becomes a skill.

Hooks — rules that don't rely on being remembered:
citation check, frontmatter validation, anonymization scan, write-destination guard. A rule in AGENTS.md is advisory; **a hook is a gate.**

The always-on agent — scheduled runs, or a self-hosted daemon (Hermes, OpenClaw, or similar) that runs the same skills on cron and messages you the digest for approval.

Slop (Part 6)

Not "AI-written text." **Fluency without grounding.** The tells:

- Acceptance criteria that restate the title
- Confident invented numbers ("reduces tickets by 30%")
- The vibes-based user ("as a user, I want a dashboard")
- Hedged non-decisions that can't be wrong

Worse than an empty backlog — **slop passes triage** and fails mid-sprint.

Defenses: no claim without a card (slop can't cite) · falsifiable sentences · required negative space · one accountable name · delete without ceremony.

Arc two: a senior product owner, using it

The same system, artifact by artifact:

7. **A PRD** from signal to sign-off
8. **Estimation** when agents do the typing
9. **A quarterly roadmap** assembled from signals
10. **Tickets & acceptance criteria** that survive the sprint
11. **A test register** tracing criteria to real tests
12. **Release verification** with Playwright MCP

Every walkthrough has the same shape: agents assemble, humans rule, artifacts remember.

The PRD walkthrough (Part 7)

- **Stage 1:** can the evidence state a problem without a solution? If not — stop. A PRD dying in ten minutes is the system working.
- **Stage 2:** the skill interrogates you — who exactly? why now? what's out?
- **Stage 3:** alternatives mandatory — models elaborate the first idea; force three
- **Stage 4:** metrics from the tree or `TBD-needs-baseline`. **Never an invented percentage.**
- **Stage 5:** verifier pass, then your name on it.

One morning. Of decisions, not typing.

Estimation (Part 8)

Story points proxied **implementation effort**. Agents broke the proxy:

- Mechanically large + conceptually settled → **collapsed** in cost
- Mechanically small + conceptually risky → **barely moved**

Size on two axes instead: **uncertainty class** (settled / bounded / open) × **verification weight** (light / standard / heavy).

The sprint is bounded by its **verification budget**, not build capacity.

Reference-class estimation is finally free: query four quarters of tracker actuals in refinement.

Roadmap & tickets (Parts 9–10)

Roadmap: an agent sweeps the quarter's signal base → themes ranked by evidence weight and trajectory. Strategy weighting stays yours. Confidence becomes **a count, not an adjective**. The verifier's best trick: "strongest theme appearing nowhere on this roadmap."

Tickets: ask for **three alternative slicings**; choose with the evidence. Criteria the repo writes for you:

> Given report data is still reconciling, the section carries a "reconciling" marker — not final totals.

That line exists because the area file's sharp edge got loaded. It settles a day-eight argument on day one.

Test register & Playwright MCP (Parts 11–12)

The register: per-feature table — every criterion → the test that proves it (auto or manual), plus **deliberate gaps, named**. A scheduled drift check flips it to `drifted` when tests move. "Done" becomes a table you read, not a claim you're handed.

Playwright MCP: the agent drives a real browser (accessibility-tree based) against staging:

- UAT scripts → supervised sessions with screenshots per criterion
- Bugs → filed with minimal reproduction steps attached
- Red e2e → triaged: app bug or stale test? Facts assembled, you rule

Citations in, citations out: **no claim of "done" without a named verification.**

The loop, closed

A Monday-morning complaint → signal card → theme → PRD (cites cards) → sliced tickets (criteria settle arguments) → register rows (name their proof) → a browser walks the release and attaches the screenshots.

Agents were never once asked to be trusted. Models where they're strong: volume, assembly, recall. Humans where they're irreplaceable: rulings, slices, accountability. Git diffs and staging gates at every seam.

Start this week

1. **An afternoon:** `AGENTS.md`, 5 glossary terms, 2 area files, 3 decision records from memory
2. **Next week:** first signal digest — however thin
3. **Then:** tracker read → draft-write with staging → the first skill
4. **Then:** hooks, the schedule, and arc two

Each stage pays for itself before the next one starts.

Thanks

The twelve-part series, with every template and skill file:

sharmaprakash.com.np


QR