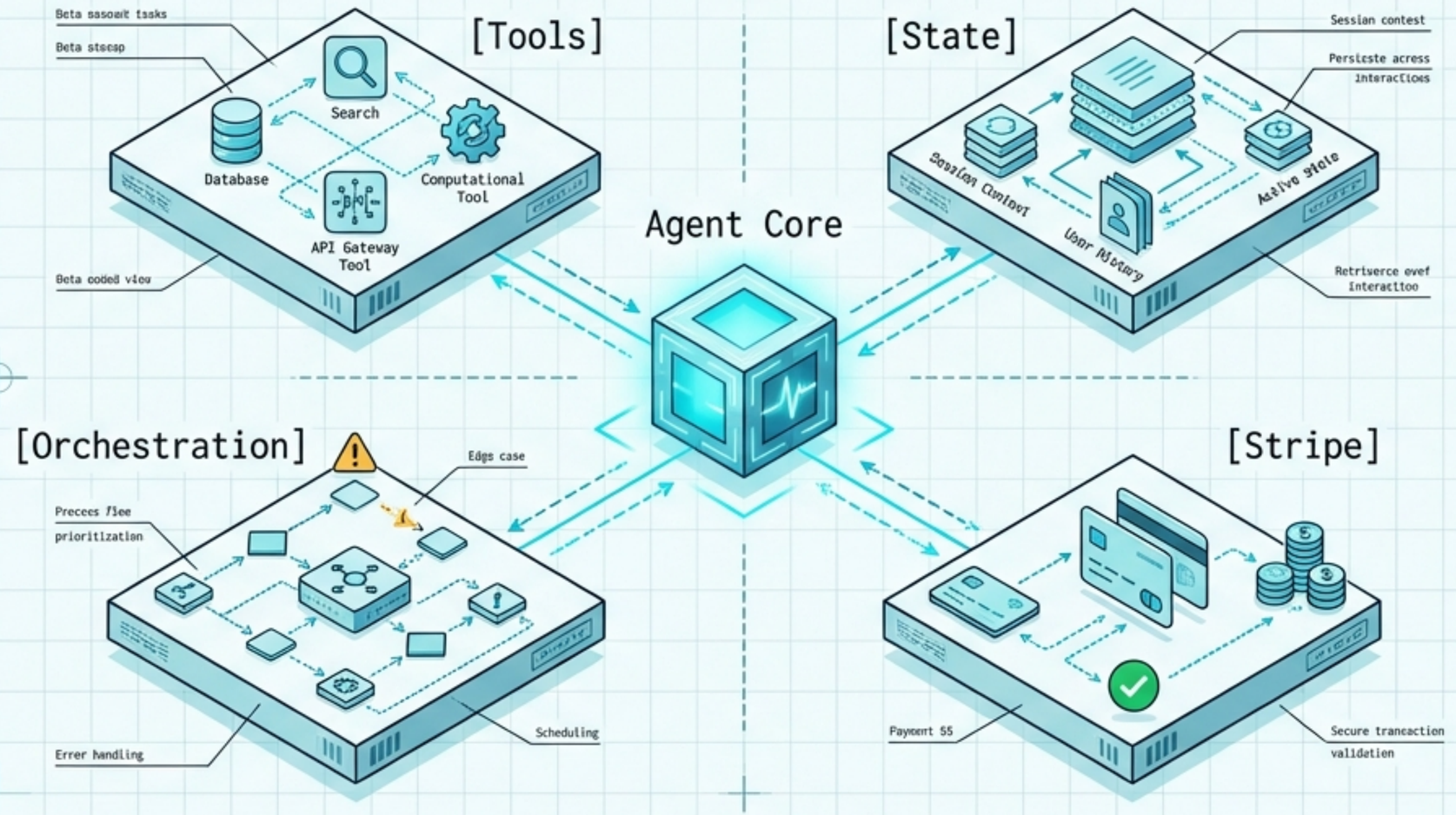
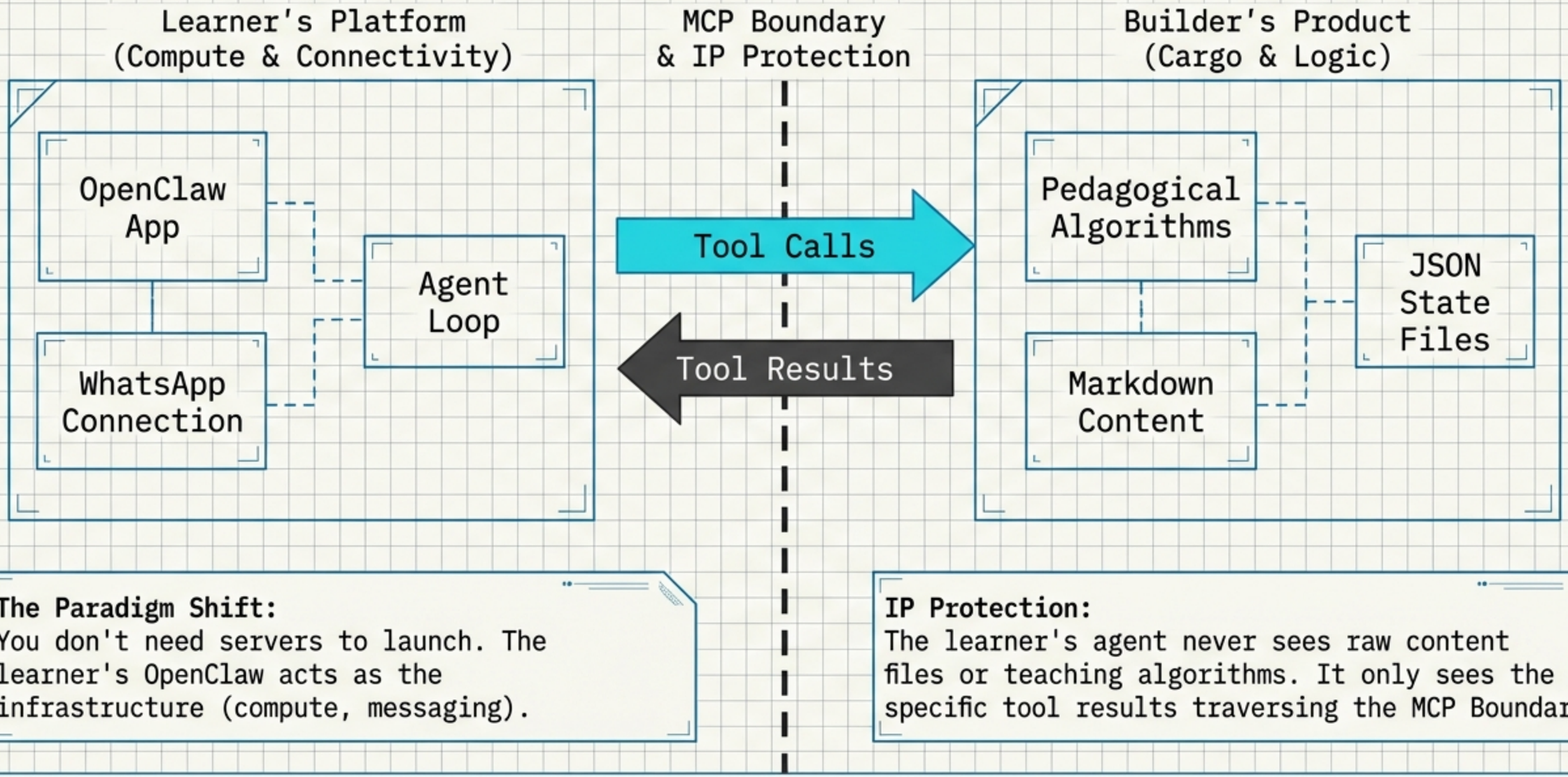
The TutorClaw Blueprint: Building a Monetized AI Agent Product



A 21-Step Schematic for Platform Inversion, Context Engineering, and AI Monetization.
Moving from isolated tool calls to a resilient, revenue-generating product architecture.

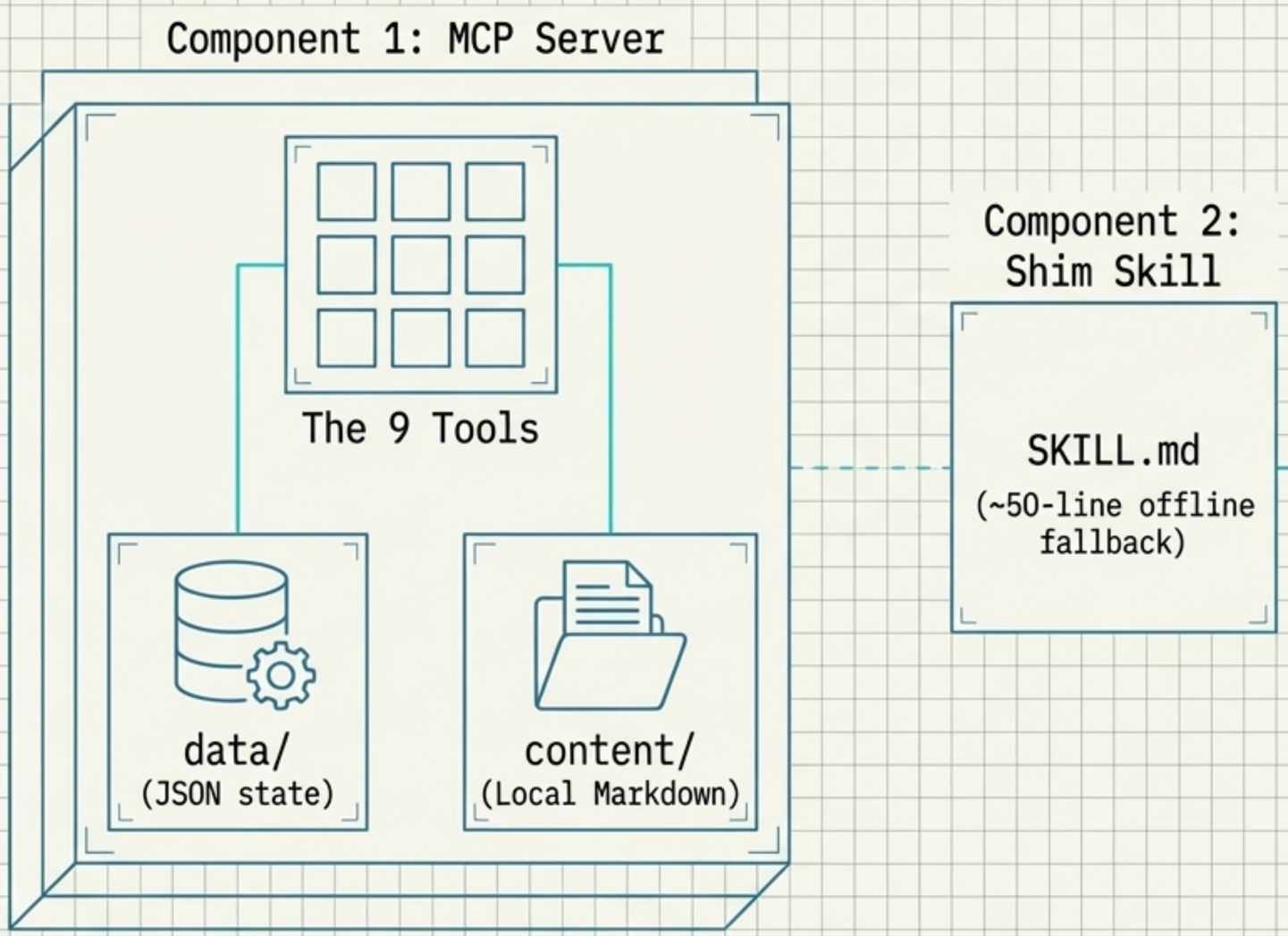
Platform Inversion and the IP Boundary



The Paradigm Shift:
You don't need servers to launch. The learner's OpenClaw acts as the infrastructure (compute, messaging).

IP Protection:
The learner's agent never sees raw content files or teaching algorithms. It only sees the specific tool results traversing the MCP Boundary.

The 2-Component Architecture



- **Zero Infrastructure Base:** No databases. No cloud storage. No deployment scripts.
- **Component 1: MCP Server:** Houses the 9 tools, local JSON state files, and local markdown curriculum.
- **Component 2: Shim Skill:** A local instruction file providing a functional floor when the server is unreachable.

The 9-Tool Surface

State (Manages Identity/Progress)
- register_learner
- get_learner_state
- update_progress

Content (Delivers Material)
- get_chapter_content
- get_exercises

Pedagogy (Teaches & Assesses)
- generate_guidance
- assess_response

Action & Business
- submit_code (Mock Sandbox)
- get_upgrade_url (Stripe)

Three groups handle the learning loop. One handles practice. One handles the business.

This is the entire product surface.

Spec-First Engineering: The Tool Contract

Job Description

Name: get_chapter_content	
Description:	Fetch the content for a specific chapter. Call this when the learner asks to study a topic. (Critical for agent selection)
Inputs:	learner_id, chapter_number
Outputs:	title, content, exercises_available
Tier Access:	Gated (Free = Ch 1-5; Paid = All)
Dependencies:	Local Content Directory, JSON State File

- Every tool gets a strict specification before a single line of code is written.
- Dependencies remain strictly local during development to prevent infrastructure blockers.

The Pedagogy Engine: PRIMM-Lite

`generate_guidance` shapes the prompt. Agent asks "What will this do?" before showing code.

Agent reveals actual output.

`assess_response` evaluates the learner's answer.

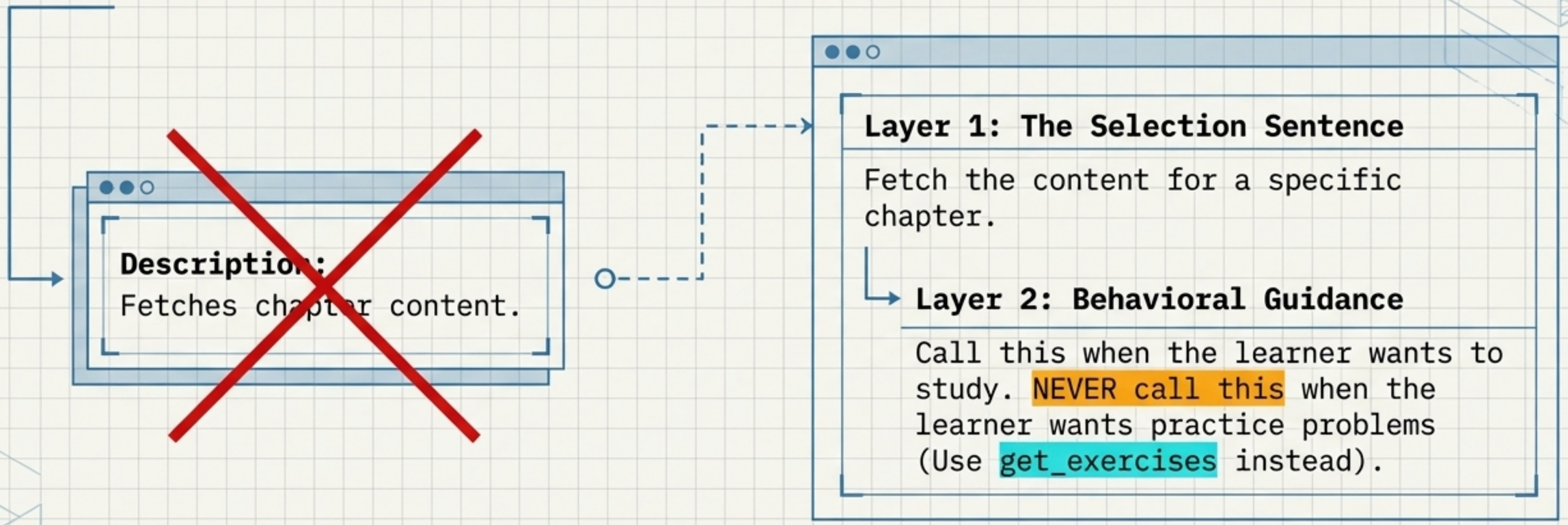
Confidence Delta

Dumping content is not teaching.

`generate_guidance` shapes how the learner engages, forcing a prediction before revealing the answer.

`assess_response` returns a positive or negative `Confidence Delta`, continuously feeding back into the learner's JSON state.

Context Engineering: The 2-Layer Description



Layer 1 provides a precise job title for the agent's initial scan.
Layer 2 provides behavioral boundaries.
NEVER statements aggressively prune the agent's candidate list, eliminating tool selection errors without writing routing code.

Orchestration via AGENTS.md

1. Session Start Protocol

Always fetch state first (get_learner_state).

2. Tutoring Flow

Content → Guidance → Response → Assessment → Progress

3. Tool Selection Rules

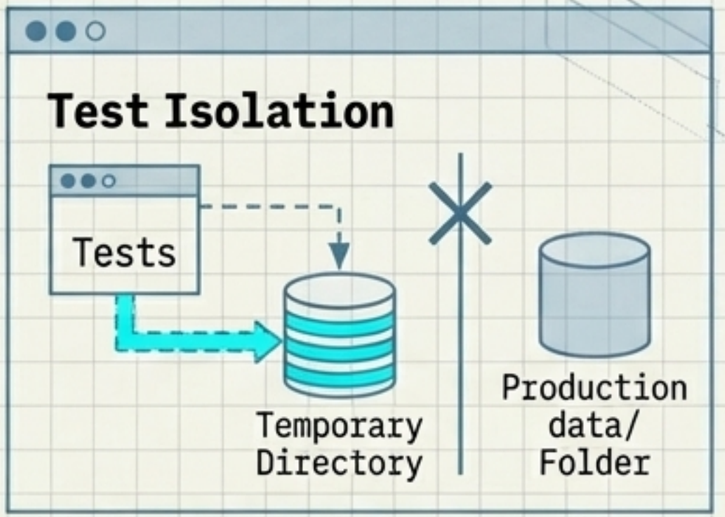
Explicit disambiguation (e.g., 'If user says practice, use get_exercises').

AGENTS.md is natural language context, not code.

It dictates the order of operations, ensuring the agent never attempts to teach before establishing the learner's current state and tier.

The Testing Safety Net

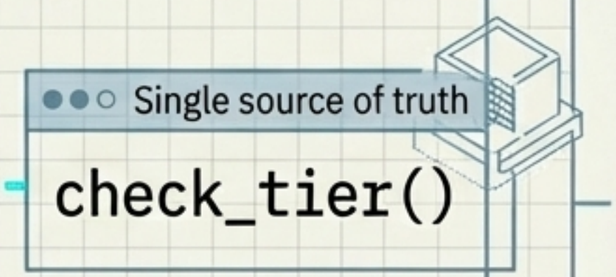
	Valid Input	Invalid Input	Tier Gating	State Persistence
register_learner	✓	✓	✓	✓
get_learner_state	✓		✓	✓
update_progress	✓	✓	✓	✓
get_chapter_content	✓		✓	✓
get_exercises	✓		✓	✓
generate_guidance	✓		✓	
assess_response	✓	✓	✓	✓
submit_code	✓	✓	✓	✓
get_upgrade_url	✓		✓	



- A green test suite proves the tool contracts are fulfilled.
- By testing against JSON mocks with temporary directories, production data is shielded from corruption during test cycles.

The Access Comparison Matrix

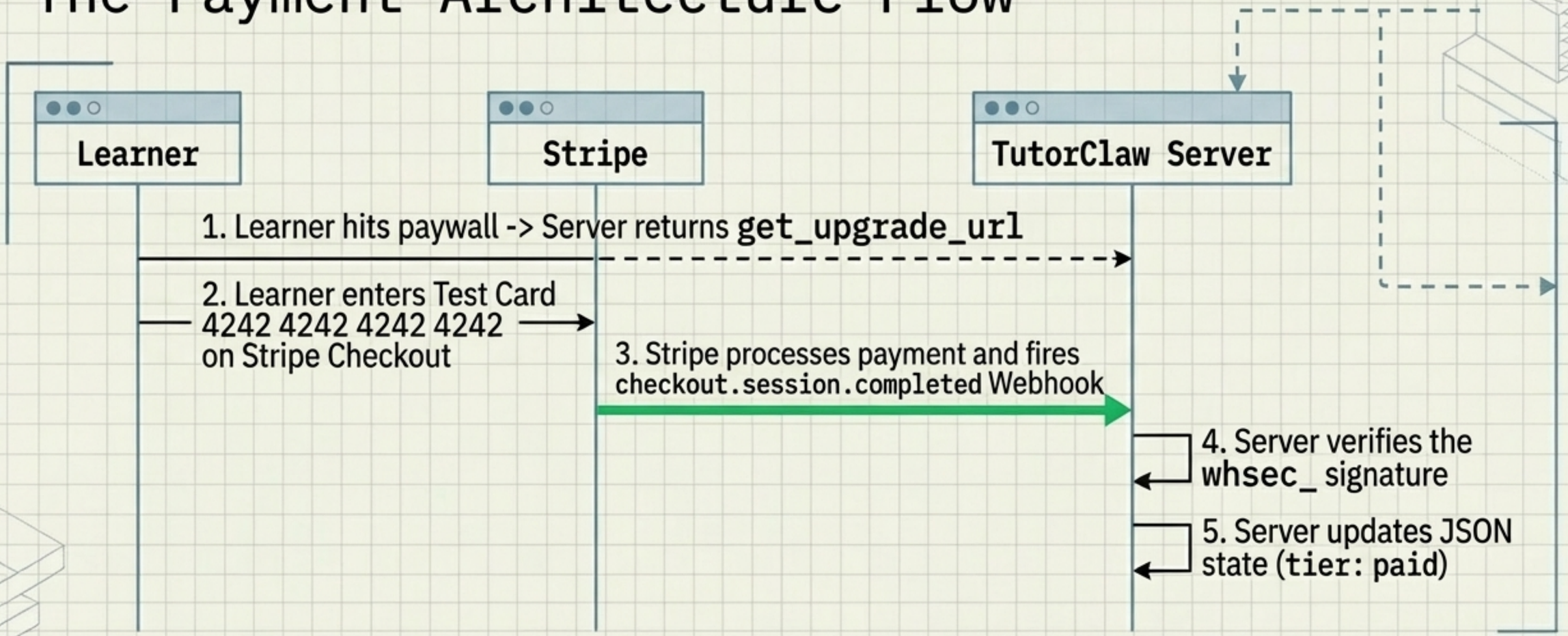
Feature	Free	Paid
Daily Exchanges	50 daily exchanges	Unlimited exchanges ✓
Content Access	Chapters 1-5 only	All Chapters ✓
Code Execution	10 code runs/day	Unlimited code runs ✓
Upgrade Prompt	Receives Stripe URL when blocked	Never sees upgrade URLs ✓



Gating is a product decision wrapped in code.

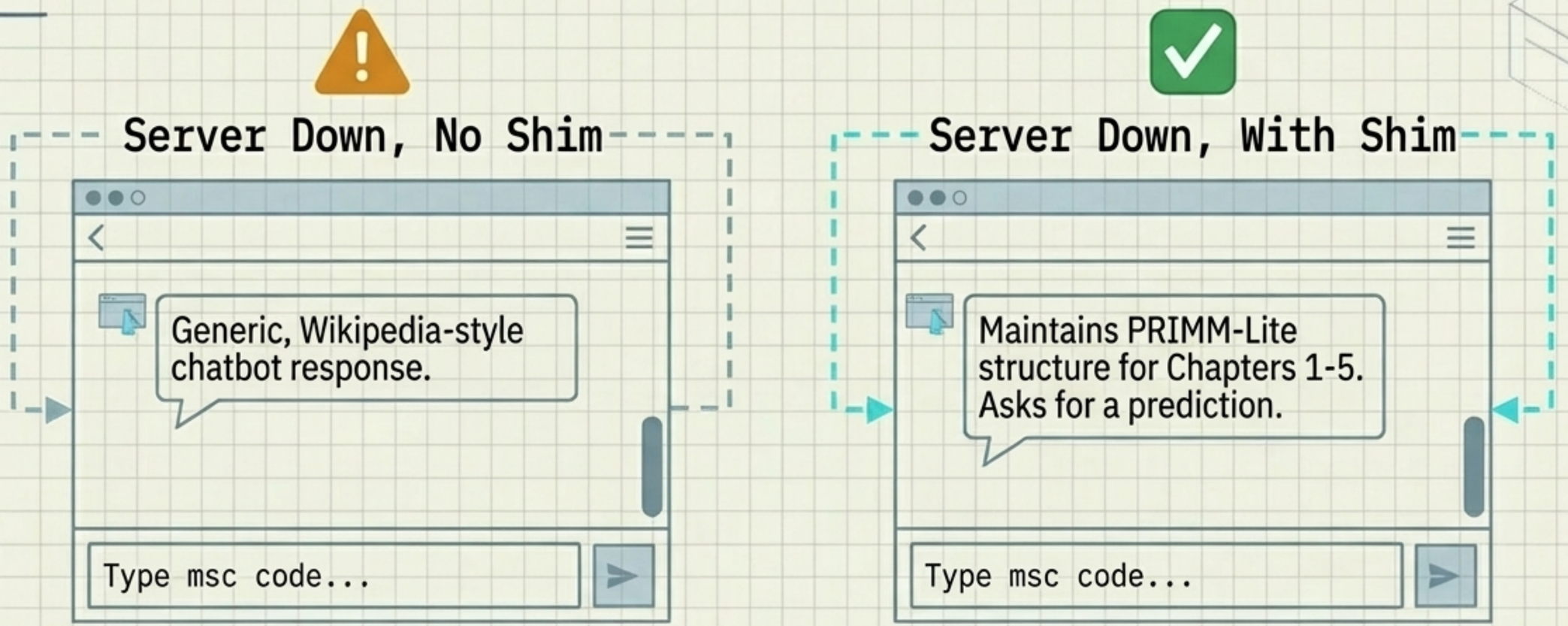
A shared `check_tier()` function enforces consistent exchange counting and error messages across the entire tool surface.

The Payment Architecture Flow



The checkout URL alone cannot confirm payment. The webhook is the only reliable signal. The server verifies the signature, extracts the `learner_id`, and flips the access tier in real-time.

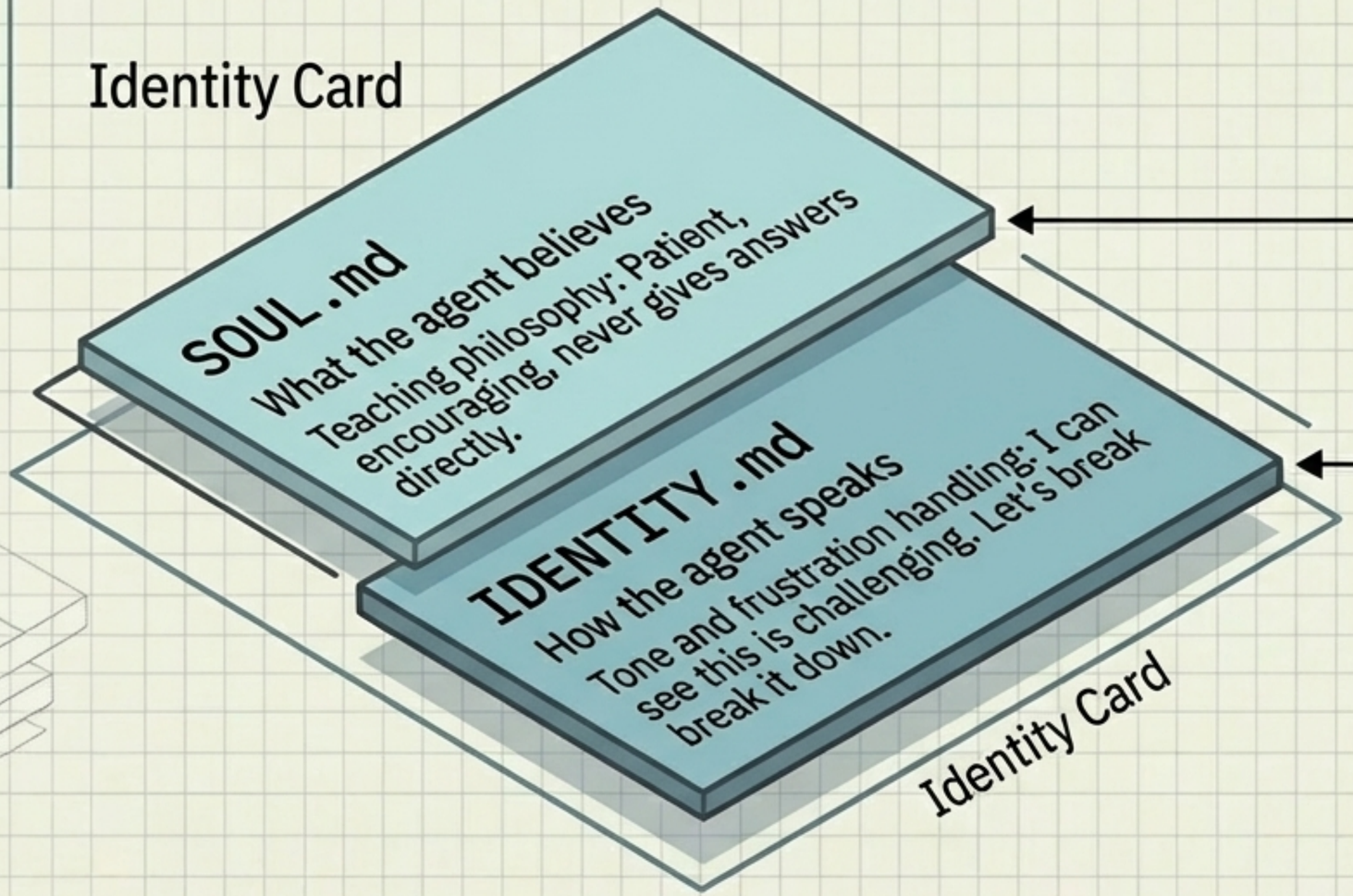
Resilience: The Shim Skill



When the MCP server goes offline, tools vanish, but instructions remain.

A 50-line SKILL.md provides a functional floor—sacrificing personalized state and premium content, but preserving the core teaching methodology.

Product Identity vs. Mechanics

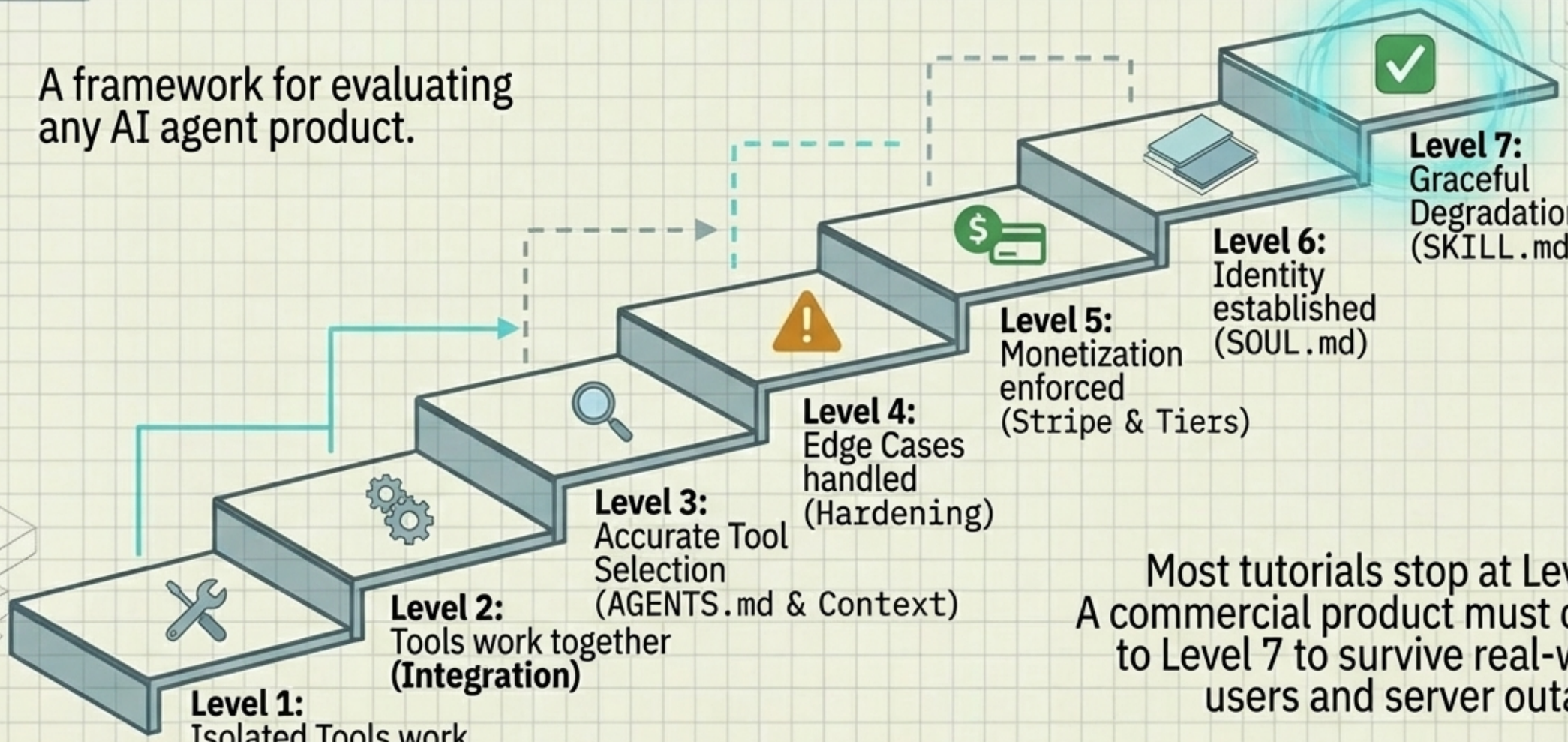


Mechanics make the product function; **identity** makes it a product users return to.

Separating logic (AGENTS.md) from personality (SOUL.md / IDENTITY.md) allows you to tune the tutor's voice without breaking the orchestration.

Synthesis Arc: The Verification Ladder

A framework for evaluating any AI agent product.



Most tutorials stop at Level 2.
A commercial product must climb to Level 7 to survive real-world users and server outages.

The Scaling Roadmap: Moving to Production

What Changes (Infrastructure)		What Stays (Product)
JSON Files → PostgreSQL		Tool Contracts
Local Markdown → Cloudflare R2 CDN	**The Test Suite**	The Test Suite
Mock Sandbox → Docker Containers		Orchestration Rules
		Agent Identity

The gap between a local prototype and production deployment is infrastructure, not product design.

Because the test suite verifies the contract (inputs/outputs), not the storage mechanism, swapping JSON for a database is entirely safe. The tool interface remains intact.