Skills Guidebook — SBA

Reusable agent skills for planning, building, reviewing, and teaching. Provided for: All SBA teams · **Revised:** July 2026

Skills are reusable capability packs. Installed ones load automatically when a matching task comes up; recommended ones can be added with `npx skills add`. Each entry explains what the skill does and when to reach for it.

1. Then vs Now — from ad-hoc prompts to reusable skills

Then	Now
Virtually no reusable agent skills.	Matt Pocock's engineering skills are installed and ready.
Every task started from a blank prompt.	Each skill carries a trigger, a workflow, and a stop condition.
Tooling knowledge lived in scattered notes and memory.	Work starts with the right skill instead of a blank prompt.
Even the **REPI Tools Guidebook** had to be built by hand each time.	The guidebook itself is kept in sync by the same skills.

2. Skills — then virtually none, now many

The environment now ships with a library of skills for planning, building, reviewing, teaching, and grilling designs. Reach for them by name or by intent.

Installed tags: Prototype · Wayfinder · Teach-me · Grill-me · TDD · Code Review · Diagnosing Bugs · QA · Research · Handoff · Plan · and many others

3. Matt Pocock's Skills — main engineering skills installed in this environment

- **setup-matt-pocock-skills** — Scaffolds the repo conventions the other engineering skills expect: issue tracker, triage label vocabulary, and domain-doc layout. Run once per repo.
- **plan** — Writes an implementation plan as a markdown file under .hermes/plans/. No execution — just a clear map before work starts.
- **wayfinder** — Plans work too large for one agent session. Creates a shared map of decision tickets on the issue tracker and resolves them one at a time until the path is clear.
- **prototype** — Builds a throwaway prototype to answer a design question or explore what a UI should look like before committing to real code.

- **tdd** — Enforces red-green-refactor: tests before code. Use for features or bug fixes where test-first discipline matters.
- **code-review** — Reviews changes since a commit or branch against repo standards and the originating spec, running both reviews in parallel.
- **implement** — Implements a spec or set of tickets end-to-end. The workhorse skill for turning a plan into working code.
- **diagnosing-bugs** — Structured diagnosis loop for hard bugs and performance regressions. Use when the cause is not obvious.
- **qa** — Interactive QA session: report bugs conversationally and the agent explores the codebase, then files well-formed issues.
- **research** — Investigates a question against high-trust primary sources and captures the findings as a Markdown file in the repo.
- **grilling / grill-me** — Relentlessly interviews you to stress-test a plan, decision, or design before you build on it.
- **teach** — Teaches a new skill or concept within this workspace, with examples tied to your actual code.
- **handoff** — Compacts the current conversation into a handoff document another agent can pick up and continue.
- **to-spec, to-tickets, to-questionnaire** — Turn conversation into a spec, a set of tracer-bullet tickets, or a questionnaire for someone else to fill in.
- **domain-modeling & ubiquitous-language** — Builds a domain model and canonical glossary, flagging ambiguities and proposing terms the whole team can use.
- **codebase-design & design-an-interface** — Shared vocabulary for deep modules and parallel exploration of interface shapes before choosing one.
- **request-refactor-plan** — Creates a detailed, tiny-commit refactor plan via interview, then files it as an issue. Use before large restructuring.

Source: github.com/mattpocock/skills · Matt Pocock's YouTube channel

---

4. /teach-me — how to implement and use a skill

1. **Load it.** Say the skill name in chat, e.g. `/teach TDD`, or call `skill_view(name='tdd')` to read its full instructions.
2. **Match the trigger.** Each skill opens with "Use when..." — wait for that situation, then invoke the skill by name.
3. **Follow the workflow.** Skills are deterministic recipes. Let the skill ask its setup questions, confirm choices, then execute.
4. **Save learnings.** If a skill taught you a stable workflow, ask the agent to save it as a new skill or patch the existing one.

Start small: pick one skill per week, use it for real work, and add the next only when the first feels automatic.

Skills Guidebook · July 2026 · Online version: http://10.0.0.123:4116/kimi-code-guidebook