Agent Skills

Test selection, non-selection, and visible failure

Session M3-05 | 3 hours

Agenda

Segment	Time
Skill directory and loading model	8 min
Required frontmatter and description	8 min
Selection and visible failure	10 min
Validation versus live invocation	10 min
Pinned interview comparison and plan handoff	18 min
Access check and lab handoff	6 min

A skill has two jobs

1. **Select the right task.**
2. **Guide the work after selection.**

A broad description can select the wrong task even when the procedure is good. The procedure also needs failure rules so it does not claim success after a required check fails.

Repository skill structure

```
.github/
└── skills/
    └── api-design/
        └── SKILL.md
```

The directory uses lowercase words and hyphens. The instruction file is named exactly `SKILL.md`.

Required frontmatter

```
---
name: api-design
description: Use for creating or reviewing REST routes under src/api/ when acceptance criteria, a reviewer, and a verified focused test command are available. Do not use for documentation, UI, deployment, or generated clients.
---
```

Match `name` to the directory. Use `description` to tell Copilot when to select the skill.

Test the positive trigger

```
Review GET /api/tasks/:id under src/api/.
Check identifier handling and the existing not-found response.
```

Required context:

- acceptance criteria;
- named paths;
- reviewer;
- focused test command that runs in this project.

Expected status: `applied`.

Test a non-trigger

```
Rewrite the installation section in README.md.
```

Expected:

```
{
  "status": "not_applicable",
  "skill": "api-design",
  "reason": "Task is outside REST route work under src/api/."
}
```

A skill should not run on tasks outside its scope.

Matching work can still be blocked

```
Add POST /api/tasks under src/api/.
Acceptance criteria exist.
No reviewer is assigned.
```

The trigger matches. The precondition does not.

Make the failure visible

```
{
  "status": "blocked",
  "skill": "api-design",
  "reason": "Missing precondition: human reviewer."
}
```

Do not invent a reviewer, skip the check, or return success.

A reviewable skill body

```
Preconditions
Procedure
Validation
Failure behavior
Maintenance
```

Each section answers a different question. Keep the procedure short enough that a human can apply it as a checklist.

Validate the files

```
npm run validate:skill
```

The validator checks:

- `SKILL.md` exists;
- frontmatter has `name` and `description`;
- `name` matches the directory;
- required sections are present.

Test all three outcomes

```
npm test
```

The suite checks that:

- a route task applies the skill;
- a documentation task does not;
- a missing reviewer blocks;
- valid and invalid directories are distinguishable.

Run the two demo scenarios

```
npm run demo
```

The output shows:

1. correct application to named API files;
2. a visible missing-reviewer failure.

The local runner is a teaching harness, not a replacement skill platform.

What local tests and live invocation show

Check	What it shows
Local directory validator	Your file has the required structure
Harness unit tests	The fixed scenario contract behaves as expected
`/skills info` and actual invocation	Copilot discovered and used the skill

Live CLI access is required. Stop if a required skill cannot load.

Inspect the pinned interview skill

Matt Pocock's `grill-me` is a community entry point that invokes `grilling`.

- Check the recorded revision and SHA-256 hashes.
- Read both files and the retained MIT license.
- Load only the reviewed project-local copy.
- Keep permissions and the agreed usage limit in place.

No marketplace installer or live download is needed for this lab.

Compare the same request twice

Use separate fresh Interactive sessions:

Run	Skill
Local	`/decision-interview` from Session M7-02
Community	`/grill-me` with its `grilling` dependency

Use the same request, owner decisions, and late visibility constraint.
Record actual loading and changes to the decisions.

Interview first, plan second

Keep Plan mode off while decisions are unsettled.

1. Finish the interview and review the decisions.
2. Save the accepted brief with open questions.
3. Start a fresh `/plan` session from that brief.
4. Name two decisions that change plan steps or acceptance checks.

Do not implement or create live work items during this comparison.

Review candidates before adaptation

Check:

- source and license;
- files and scripts;
- dependencies and network use;
- data flow;
- repository fit;
- owner and retirement condition.

Do not run an unfamiliar script just to inspect it.

Review recurring session corrections

```
/chronicle improve
```

This subcommand scopes history to the current repository or working directory. Check a recommendation against an actual correction.

Choose instructions, a skill, or neither. Cancel the apply step; suggestions may start selected. Do not reindex or inspect unrelated history for this lab.

Too little history is a valid result. This uses the last five minutes of the 45-minute comparison block.

Lab handoff

1. Create `api-design/SKILL.md`.
2. Validate the directory and frontmatter.
3. Run the trigger scenario.
4. Run the non-trigger scenario.
5. Run the missing-reviewer scenario.
6. Finish with the two-scenario demo.
7. Check live API-skill invocation and the missing-reviewer stop.
8. Compare both live interviews and trace decisions into `/plan`.
9. Check scoped Chronicle suggestions without auto-applying them.

Submit actual outputs. The harness or prepared transcripts cannot replace live loading, either interview, or the plan handoff.