Interview ideas and shape better work

Session M7-02 | Product and Delivery Teams | 3 hours

GitHub Copilot is required

Before the session:

- Copy the supplied project skill to `.github/skills/decision-interview/SKILL.md`.
- Start a new project session.
- Check `decision-interview` under **Customize** → **Skills**.
- Run `/decision-interview` with the short test request.

Stop the lab if any check fails.

Start with the actual problem

> "Make request status clearer so people stop asking support."

Before this idea can enter a sprint, decide who it serves and what it changes. You also need decisions about data, scope, ownership, and approval.

The **Delivery Decision Studio** path carries the Session M7-01 context brief into this interview. Service Request Portal remains the initiative.

The shaping flow

```
Vague request
  -> project skill installation
  -> GitHub Copilot decision interview
  -> human decisions
  -> Markdown decision brief
  -> GitHub issue proposal
  -> acceptance criteria
  -> definition of done
```

Do not jump from one sentence to an issue template.

The project skill keeps this method consistent across requests. It does not make the decisions.

Interview as a decision tree

Each answer can raise new questions.

Ask the current **frontier**: every decision the owner can make now without guessing about an unsettled decision.

Questions that depend on this round wait for the next round.

Work in rounds

```
Round 1: outcome, affected user, current pain
Round 2: scope, current state, constraints
Round 3: success, failure, edge cases
Round 4: owner, reviewer, approval, rollout
```

Ask GitHub Copilot to work out the next frontier from the answers the owner accepted.

Use **Interactive** mode for each frontier. Switch to **Plan** before proposed file changes. Autopilot is outside this interview.

Check context between rounds

- Run `/context`.
- Use `/compact` only when the same thread must continue.
- Use `/clear` or `/reset` when old assumptions are polluting the interview.
- Resume with the approved context and the owner's settled decisions, not a transcript.

Give a recommendation

For each question:

1. State the decision clearly.
2. Offer concrete choices when useful.
3. Recommend an answer and explain the tradeoff.
4. Wait for the request owner to decide.

GitHub Copilot frames the choice. **A person owns the choice.**

Facts and decisions are different

Use approved sources for facts	Ask people for decisions
Existing workflow	Which users are in scope
Current labels or fields	Which behavior should change
Known technical limits	Which tradeoff to accept
Policy and data rules	Who approves release

Mark unsupported claims as unverified. Do not present them as facts.

Settle six decision areas

- **Scope:** who and what are included?
- **Constraints:** what limits the solution?
- **Success:** what result will show that the change helped?
- **Failure:** what can go wrong, and what should happen?
- **Ownership:** who decides and who maintains it?
- **Approval:** who reviews the brief and accepts the work?

Keep unresolved decisions visible.

The decision brief

A useful brief records:

- problem and target outcome;
- users, scope, and exclusions;
- constraints and dependencies;
- success and failure measures;
- settled decisions and open questions;
- owner, reviewer, and who approves the brief and when.

Keep it short enough to review in one sitting.

Create a GitHub issue proposal

Ask GitHub Copilot to draft:

- user and outcome;
- defined scope and non-goals;
- acceptance criteria;
- dependencies and risks;
- owner and reviewer;
- definition of done.

Compare the draft with the approved brief.
Each field should trace to an approved answer or an open question.

Session M7-03 creates the approved issues through GitHub MCP.

Reopen only the affected branch

New source information changes requester and support visibility.

```
Approved brief
  + new visibility constraint
  → reopen visibility decisions
  → update affected criteria
  → preserve unrelated decisions
```

Do not restart the interview or rewrite settled scope without new source information.

Check in a fresh session

Give a new session only:

- the revised decision brief;
- the revised issue proposal.

Ask it to find contradictions, unresolved owners, and acceptance criteria with no source decision.

The fresh session shows which assumptions you left in the conversation instead of recording in the files.

Then use `/rubber-duck` to challenge unsupported criteria, hidden scope, and missing owners.

Acceptance criteria describe testable behavior

Weak:

> Status is easy to understand.

Testable:

> A requester can see the current status and last updated time for each submitted request.

Include a failure case when the feature can mislead, expose data, or block progress.

Define what the team needs before release

Acceptance criteria describe product behavior.

Definition of done covers the work needed to release and hand over safely:

- criteria reviewed;
- permissions checked;
- supported states documented;
- test or review results attached;
- product and service owners approve.

Azure Boards update paths

The interview always runs with **GitHub Copilot**.

Azure Boards path	Method
MCP	Copilot prepares and reviews the approved update before the MCP tool sends it
Browser	Copilot prepares the exact values, then the learner enters and checks them

There is no static or no-Copilot route.

Lab deliverables

1. The installed `decision-interview` project skill.
2. A revised **decision brief**.
3. An aligned **GitHub issue proposal**.
4. Any fixes made after the fresh-session check.

The request owner decides whether the brief is ready for issue planning.

Stop conditions

Stop when:

- GitHub Copilot access does not pass preflight;
- the input is not approved or sanitized;
- a required decision owner is absent;
- policy or data boundaries are unclear;
- the team cannot state how it will check success or failure.

Record content blockers. Resolve Copilot access before starting the exercise.