Before a partner bank, auditor or examiner asks: 15 questions about an AI step in your workflow

For: engineering and risk leads at lenders and fintechs who run, or plan to run, an AI step in a credit, servicing or operations workflow.

This is a technical and process checklist. It is not legal or regulatory advice, and it does not say what any rule requires.

How to use it

1. Pick one AI step in one workflow. For example, a model that reads income documents and proposes a decision for an underwriter.
2. Answer each question for that step. Score 0 if the answer is no, 1 if partly, 2 if yes. Write "?" if nobody on the team knows.
3. A "?" is a finding. If the team cannot say, a reviewer outside the team will not be able to either.
4. Total the score (maximum 30). Write your three lowest-scoring questions in the box at the end.

Time: about 20 minutes for one step.

A. Ownership and limits

1. Is this AI step on a written list of AI systems, with a named owner, a stated purpose and a risk tier?

- Why it matters: a reviewer cannot ask about a step that nobody has listed, and an unowned step has no one to answer.
- Good looks like: one list or register entry per step, with owner, purpose and tier filled in, and a date for the last review.

2. Is it written down what this step must not do?

- Why it matters: limits that exist only in someone's head are not checked when the prompt, the model or the inputs change.
- Good looks like: a short list of known limits and out-of-scope inputs, stored next to the entry in question 1.

B. Review gate

3. Which outputs must a named person approve before anything happens?

- Why it matters: "a human is in the loop" is not an answer until the step, the person and the action are named.
- Good looks like: a list of the actions that wait for approval (for example, sending a customer notice or writing to the loan system) and the role allowed to approve each.

4. Who approved this output?

- Why it matters: this is the first question a reviewer asks about any single decision.
- Good looks like: for any past output you can retrieve the approver's identity, the time, and the reason they gave, in under a few minutes.

5. Did the reviewer see what the model saw, and what it proposed?

- Why it matters: an approval of a summary the reviewer could not check against the source is a weak control.
- Good looks like: the review screen shows the model's inputs (or links to them) next to the proposed output.

6. Is the approval tied to the exact action approved?

- Why it matters: an approval that can be reused, or applied after the proposed action has changed, does not show what the person agreed to.
- Good looks like: the approval record holds the parameters of the action, and a changed action needs a new approval.

7. What happens if the reviewer does not respond or is unavailable?

- Why it matters: a step that proceeds when nobody answers has no gate.
- Good looks like: the step stops and the item waits or is rejected after a set time. It does not proceed by default.

C. Evidence

8. What inputs did the model see for a given decision?

- Why it matters: this is the question the worked example below traces. Without it you cannot tell whether a wrong output came from the data, the prompt or the model.
- Good looks like: for any run, you can show the full text the model received after templating, including retrieved documents and any free-text fields, as it was at that time.

9. Which model and which version produced the output?

- Why it matters: providers change and retire models, and behaviour differs between versions.
- Good looks like: each record names the provider and model identifier used, including when a fallback model answered instead of the first choice.

10. Where is the record kept, for how long, and who can retrieve it?

- Why it matters: a record that lives in a developer's logs, or expires before a review, cannot be produced when asked.
- Good looks like: one named store, a stated retention period, named people with access, and a tested way to pull the records for one decision.

11. Can you replay or reconstruct a past run?

- Why it matters: reconstruction is how you explain an output after the fact, and how you test a fix against the case that failed.
- Good looks like: from the record alone you can re-run the same input through the same prompt version, or you can state which parts cannot be reproduced (for example, model nondeterminism) and how you cover that.

D. Errors and change

12. What happens on a model error, a timeout or an invalid output?

- Why it matters: error paths are where steps skip the gate or send partial results downstream.
- Good looks like: a defined behaviour for each failure (retry, fall back, stop, route to a person), and errors are recorded the same way as successes.

13. Who can change the prompt or the workflow, and is each change recorded?

- Why it matters: a prompt edit changes the decision logic. If it is not recorded, an old decision cannot be tied to the logic that made it.
- Good looks like: changes need a named author and a reviewer, each version is kept, and each run record points to the version it used.

14. What tests run before a prompt, workflow or model change ships, and who decides a model can be swapped?

- Why it matters: without a fixed test set you learn about regressions from customers or reviewers.
- Good looks like: a fixed set of cases with expected results, a pass/fail result stored for each change, and a named person who approves model swaps.

E. Data

15. What customer data reaches the model provider, under whose contract, and how is it disposed of?

- Why it matters: sensitive fields sent to a provider are a vendor-risk item, and disposal that is not logged cannot be shown later.
- Good looks like: a list of data fields sent, a step that removes or masks fields that should not leave, the contract the calls run under, and a log entry when records are disposed of.

Score and gaps

Total: ____ / 30

Three lowest-scoring questions:

1. ____
2. ____

3. ____

The scoring is ours and is not a standard. Use it to rank your own gaps, not to compare with other teams.

Worked example: question 8, "What did the model see?"

Everything below is fictional and synthetic. The company, the people and the numbers are invented for this example. No real company, customer or employer data is used. The record is written by hand to show the fields a good record holds. It is not output from a real run.

The workflow. A fictional lender, Larkfield Credit, runs an AI step called `income-doc-check` on small-business loan applications. The step reads the applicant's uploaded pay-stub text and the application fields, and proposes either `accept` or `request_more_documents`, with a one-sentence reason. An underwriter approves or rejects the proposal. Only an approved proposal changes the application status.

The question from the reviewer. A partner bank sample includes application `APP-1042`. The reviewer asks: what did the model see when it proposed `request_more_documents`?

An answer without a record. The team says the prompt asks the model to compare stated income with the pay stub. They cannot say which pay-stub text was sent, whether the free-text notes field was included, or which model answered. The score for question 8 is 0.

An answer from a record. The team pulls the run record for `APP-1042`:

Field	Value
Run	`run-7f3a` (synthetic)
Workflow version	`income-doc-check` v14
Step	`propose_decision`
Provider and model	`provider-a` / `model-x-2026-09` (placeholder names)
Fallback attempts	none
Input, as sent (after templating)	Application fields: `stated_monthly_income: 6,000`, `business_type: landscaping`, `notes: "Seasonal income, see stub"`. Pay-stub text: `Gross pay period: 1,820. Period: 2 weeks. YTD: 7,140.`
Output	`{"decision": "request_more_documents", "reason": "Stated monthly income 6,000 is not supported by one pay stub showing 1,820 for two weeks."}`
Approval	`underwriter-3` (synthetic ID), reject/approve: approve, reason: "One stub is not enough for seasonal income", time: 2026-10-14 15:02

What the record lets the team say:

- The model saw one pay stub and the free-text `notes` field. It did not see bank statements, because the workflow did not send them.
- The proposal followed from the inputs. The follow-up question is whether the workflow should send more documents, That is a workflow design question.
- The underwriter saw the same inputs on the review screen and gave a reason.
- The version and model are named, so the team can re-run `APP-1042` against v14 and against a later version and compare.

What the example also shows about the other questions:

- Question 4 is answered by the approval row.
- Question 9 is answered by the provider and model fields, and by the fallback attempts field.
- Question 13 is answered by the workflow version.
- Question 11 depends on whether the stored input is complete. If the pay-stub text had been truncated before storage, the team could not replay the run, and the score for question 11 would drop to 1.

What to do with the result

Take the three lowest-scoring questions. For most teams, questions 4, 8 and 11 are the first to check, because they come up in every sampled decision.

Grove

Grove, the workflow engine Magic Ingredient builds on, stores for each step its input, output, model, provider and any failover attempts, and stores each approval decision with the approver, the reason and the time (`kb/compliance/nist-ai-rmf.md`, `kb/decisions/0026-durable-waits-and-approval-gate.md` in grove-core). Grove has no SOC 2 or other third-party attestation, and it does not answer questions about your own policies, retention periods or contracts.

Next step

A 30-minute review: we look at one of your AI workflows and your three lowest-scoring questions, and tell you what we would fix first. To book it, [TBD: booking link or reply address, not decided]. Magic Ingredient is the consulting practice that does this work.

DRAFT 1, 2026-11-12. Magic Ingredient. Not published. Technical and process checklist; not legal or regulatory advice.