

HandoffCheck

A TECHNICAL STUDY OF PRODUCT-TO-ENGINEERING HANDOFFS

From a feature idea to a shared agreement.

How structured requirements, source-linked AI assistance, revision-bound approvals, and durable automation connect a product idea to an implementation brief that both teams have reviewed.

Written by Abdul Moiz

Architecture / Implementation / Evaluation

AI assistance / Human approvals / TypeScript / PostgreSQL / Team workflows

Contents

A complete technical description of the product, its architecture, implementation boundaries, and evaluation. Select a chapter below to jump to its page.

01 / Make the agreement explicit.	3
02 / One application, several clear boundaries.	4
03 / A request is a structured contract.	6
04 / Start with a conversation. Confirm the form.	6
05 / Build context for the task at hand.	7
06 / Keep questions beside their source.	9
07 / Two people approve one exact revision.	9
08 / Access follows participation.	11
09 / A brief preserves the approved wording.	12
10 / Make background work observable.	12
11 / Connect the team's existing tools.	14
12 / Test the workflow at its boundaries.	15

PART I / PRODUCT INTENT

01 / Make the agreement explicit.

A feature request can look complete while leaving engineering to guess who can use it, what happens on failure, or which behavior is actually in scope. Questions and decisions spread across chat, documents, and meetings. HandoffCheck brings requirements, clarification, reviewer decisions, and the final implementation brief into one workspace.

An author captures a request, selects a product lead and an engineering lead, and submits a revision for review. Deterministic checks and optional AI assistance identify missing details and ambiguous wording. Both assigned leads must approve the same reviewed revision before a brief can be generated.

TABLE 01 / Participant / Responsibility / Product outcome

Participant	Responsibility	Product outcome
Author	State the problem, behavior, scope, criteria, and reviewer assignments	An inspectable requirements revision
Product lead	Review the intended behavior and resolve product questions	An explicit decision on the current revision
Engineering lead	Review the handoff and its acceptance criteria	An explicit engineering decision and reviewable QA drafts
Administrator	Manage membership and workspace configuration	The right people and integrations are available

WHAT APPROVAL ESTABLISHES

Approval records the team's agreed handoff. It does not establish technical feasibility, a delivery estimate, completed implementation, or a release commitment.

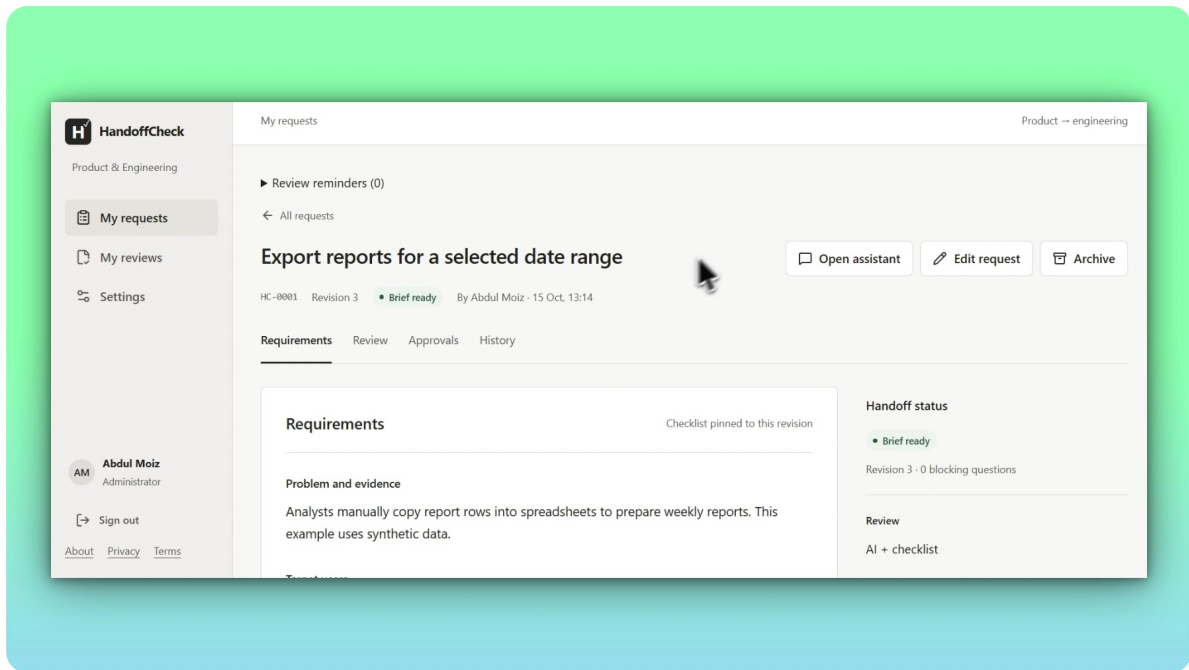


Figure 01. The request workspace connects a feature, its revision, and the current handoff state.

PART II / SYSTEM ARCHITECTURE

02 / One application, several clear boundaries.

The interface uses React, Vite, and TypeScript, with React Router for screens, TanStack Query for remote state, React Hook Form and Zod for forms and validation, and Radix primitives for selected accessible controls. Vercel hosts the web application. Electron provides a Windows shell over the hosted workspace.

Supabase supplies Google authentication, PostgreSQL, private object storage, and Deno Edge Functions. Shared TypeScript contracts express requirements, findings, approvals, jobs, and command payloads. Edge handlers verify the current user and route commands into database transactions or bounded provider operations.

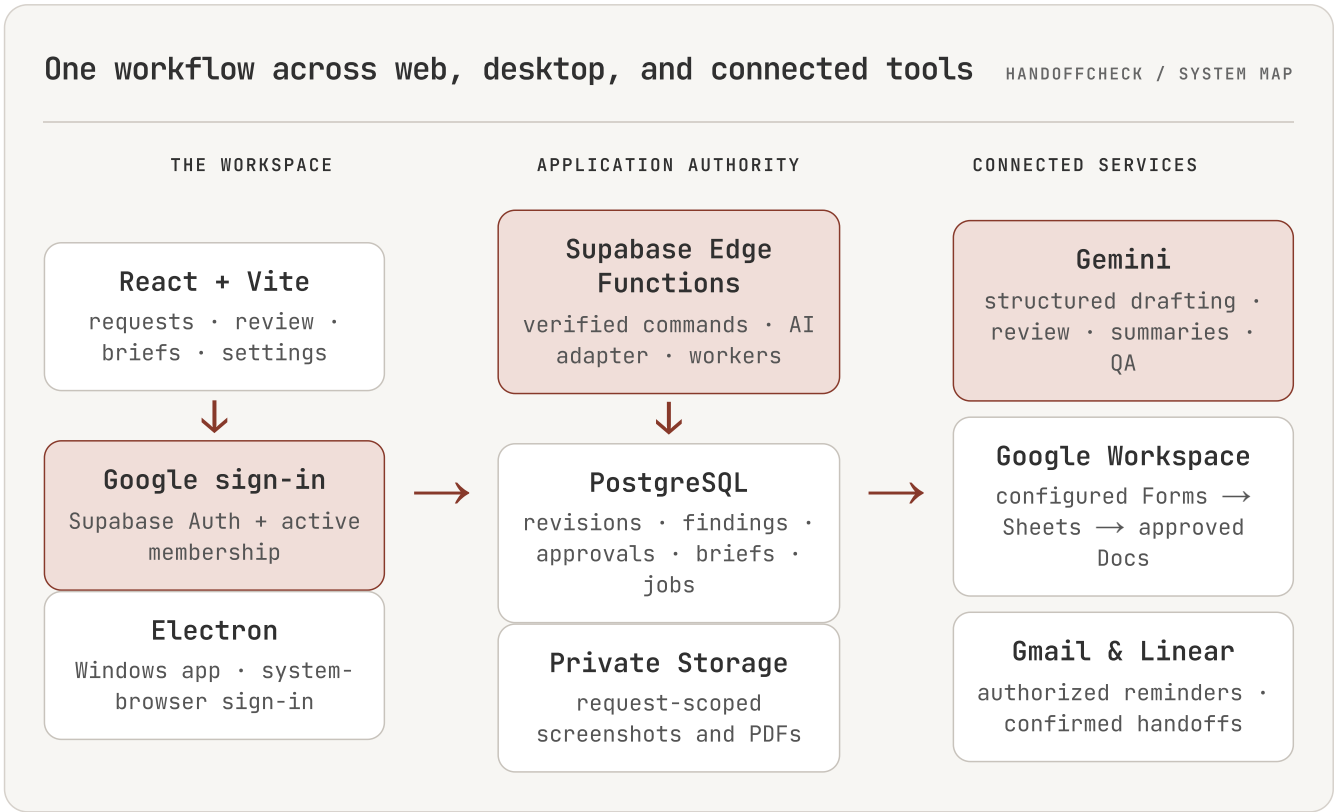


Figure 02. React and the Windows shell share the hosted workspace. Supabase enforces identity, request access, workflow transactions, and private file storage.

TABLE 02 / Layer / Owns / Boundary

Layer	Owns	Boundary
Web / desktop interface	Drafting, review, source inspection, briefs, and settings	Browser state does not grant approval or database authority
Edge API	Verified commands, provider requests, file checks, and integration coordination	Secrets stay in server configuration
PostgreSQL	Participant access, revisions, approvals, jobs, and immutable brief records	Transactions resolve conflicting updates
External services	Gemini language tasks, configured Google workflows, Gmail, and Linear	Each integration has its own consent, configuration, and failure behavior

PART II / SYSTEM ARCHITECTURE

03 / A request is a structured contract.

An incomplete idea can be saved privately as a draft. Submission requires the baseline engineering checklist, including the problem and evidence, target users, expected behavior, scope and exclusions, testable acceptance criteria, edge cases, dependencies, data and permissions impact, success measure, and priority rationale.

TABLE 03 / Field group / Purpose

Field group	Purpose
Problem, audience, and evidence	Explain who needs the change and why
Behavior, scope, and exclusions	Describe the flow and its agreed boundaries
Acceptance criteria and edge cases	Make expected outcomes observable, including relevant failure cases
Dependencies and data impact	Expose constraints and permission changes
Success, priority, and scheduling	Record the intended outcome, rationale, and an explicit date/timezone or unscheduled state
References, files, and assigned leads	Retain supporting material and identify the two reviewers

The declared contract limits acceptance criteria to twenty, links to ten, and reviewed text to 24 KB. Screenshots and PDFs are private supporting files, with a five-file and 5 MB-per-file limit. Reference URLs are validated but are not fetched for AI review; attachment contents are not sent to the language model.

MISSING INFORMATION STAYS MISSING

Conditional not-applicable answers require a reason where the checklist allows them. AI suggestions cannot fill unknown facts, invent success metrics, or erase deterministic required-field failures.

PART III / ASSISTED REQUIREMENTS

04 / Start with a conversation. Confirm the form.

The drafting assistant accepts the author's description, proposes structured fields with verbatim evidence from author messages, and asks at most one clarification question per turn. Optional choices explain their tradeoffs. A recommended choice remains a proposal until the author selects it or supplies another answer.

The author checks the preview, applies it to an editable form, selects both leads, and explicitly saves or submits. Conversation output cannot create a request, assign reviewers, or approve a revision. If creation succeeds but submission fails, the saved draft opens with the failure instead of encouraging a duplicate request.

TABLE 04 / Intake boundary / Implemented behavior

Intake boundary	Implemented behavior
Input	Up to forty messages, 3,000 characters each, 46 KB total
Source evidence	Quotes must occur in an author message at the stated index
Response recovery	A UUID idempotency key can recover the same response without another provider attempt
Draft memory	Author/workspace-scoped response receipts expire after twenty-four hours
Consent to requirements	The author explicitly confirms proposed wording through the form

The input transcript is not stored in the receipt. The validated response can contain quoted author text, and the conversation remains in component memory while the user is drafting. Reusing a receipt requires matching actor, workspace, input prefix, policy version, and lifetime.

PART III / ASSISTED REQUIREMENTS

05 / Build context for the task at hand.

Drafting, requirement review, edit proposals, reviewer summaries, and QA use one Gemini adapter with separate task contexts. The server supplies the model identifier through GEMINI_MODEL_ID; the code does not require a universal default model. Calls use structured JSON output and independent Zod/domain validation.

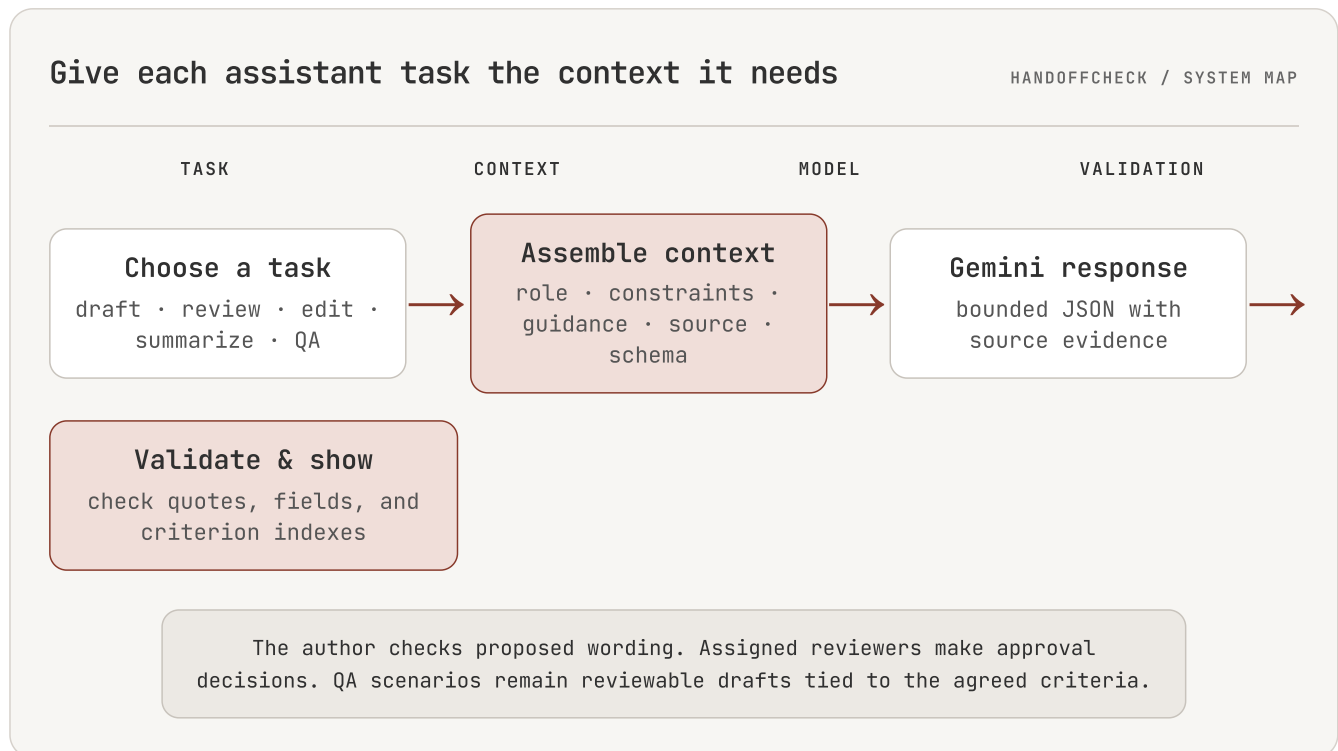


Figure 03. Drafting, review, edits, summaries, and QA use separate contexts through the same Gemini adapter. The model identifier is supplied by server configuration.

TABLE 05 / Order / Context layer / Purpose

Order	Context layer	Purpose
1	Role and goal	Define the selected specialist task
2	Constraints	Require source evidence and keep permissions and approvals outside model output
3	Guidance and authorized source	Select up to three versioned guidance chunks and the current task's source
4	Illustrative example	Demonstrate the required shape without introducing feature facts
5	Active task and schema	Specify the requested result, evidence rules, and JSON fields

Older assistant dialogue is compacted while author statements are preserved. Review and QA retain the full source revision and original criterion indexes. A serialized provider request exceeding 120 KB fails before network access rather than dropping source facts. This is task guidance retrieval, not a vector search over company documents.

EVIDENCE SCOPE

A valid quote establishes correspondence to supplied text. It does not prove that the text is true or that the model's interpretation is correct. The author and reviewers still inspect the proposed result.

PART III / ASSISTED REQUIREMENTS

06 / Keep questions beside their source.

A review first applies required-field rules in code. Optional language review can identify missing details, ambiguous behavior, conflicts, and potential implementation risks. Findings carry the source field, preserved quote, severity, explanation, and clarification question.

The adapter bounds output size and finding counts, verifies field references, and checks source quotes. Missing-detail findings can have empty evidence where the contract permits it. Unresolved blocking findings prevent approval. Leads can dismiss an AI finding with a reason; missing mandatory fields require an updated revision.

Answers become part of the agreement only after the author incorporates them into requirements and submits the updated revision. The assistant's activity feed shows task status, actor, revision, source inspection, prior responses, and sanitized errors. Personal Clear chat affects only that member's browser view; reviews, approvals, and source records remain available.

CHECKLIST-ONLY MODE

A deliberate checklist-only configuration runs the workflow without language review and labels that limitation. A failed provider call does not silently become a successful checklist-only review.

PART IV / DECISIONS & PERSISTENCE

07 / Two people approve one exact revision.

Every successful content save appends an immutable revision containing requirements, attachment metadata, reviewer assignments, checklist version, and canonical input hash. Changing files or assigned leads also changes the revision. Comments and personal interface preferences do not.

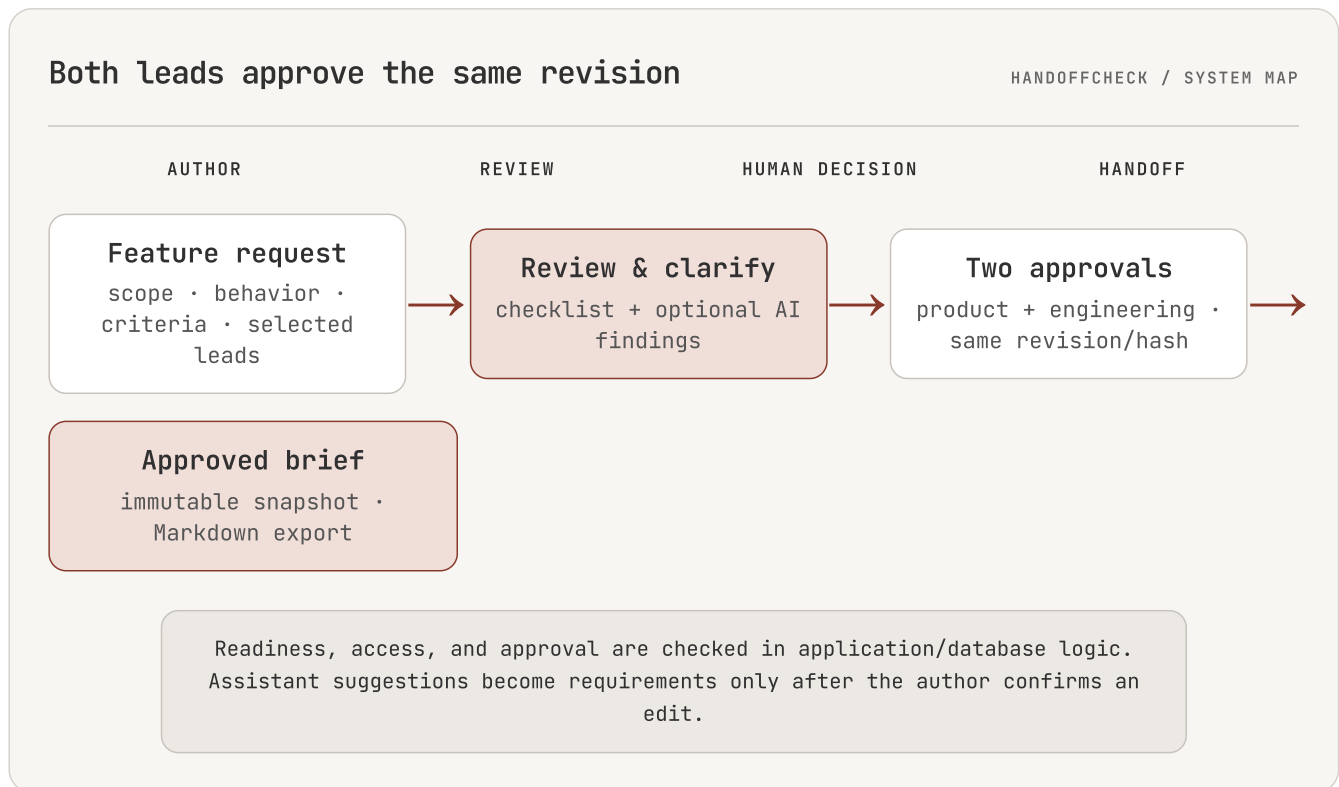


Figure 04. A saved change creates a new revision. Prior review and approval records remain in history; the new revision needs its own review and decisions.

Approval requires two distinct active leads in the appropriate roles, neither of whom is the author. Save, review, approval, and brief generation are serialized through transactions on the same request. The database rechecks current assignment, membership, role, revision, and review before accepting a decision or generating a brief.

TABLE 06 / Event / Effect on the agreement

Event	Effect on the agreement
Save revised content	Create a new author-only draft that needs resubmission and review
Replace the current review	Invalidate current approvals before a brief is generated
Lead requests changes	Require a new revision and successful review before approval resumes
Reviewer reassignment or revocation	Remove prior participant access and recheck decision eligibility
Generate a brief	Preserve one immutable artifact for that reviewed revision

CONCURRENT UPDATES

If a save arrives first, approvals for the earlier source cannot authorize the new content. If brief generation completes first, that artifact retains the earlier agreement and remains historical after a subsequent edit.

PART IV / DECISIONS & PERSISTENCE

08 / Access follows participation.

Google sign-in establishes identity; active workspace membership establishes application access. Drafts are visible to their author. Submitted requests are available to the author and the two currently assigned active leads with the matching roles. Administrators manage settings and members without automatic access to request contents.

The canonical participant predicate is applied to database policies and command paths, including request reads, writes, attachments, comments, findings, approvals, briefs, AI artifacts, notifications, and integration previews. Reassignment, role changes, and membership revocation are checked against current state.

TABLE 07 / Record / Retained responsibility

Record	Retained responsibility
Requests and revisions	Current state plus immutable revision snapshots
Reviews and findings	Evidence, questions, outcomes, and the reviewed input
Approvals and briefs	Human decisions and their deterministic output artifact
Jobs and usage	Durable attempts, leases, state, quotas, and filtered context metadata
Files and connections	Private references and separately authorized integration state

COPIES HAVE THEIR OWN LIFETIME

Exported Markdown, Google documents, downloaded files, and already issued short-lived file URLs have their own sharing and access lifetime. Changing an application role does not retract a previously downloaded copy.

09 / A brief preserves the approved wording.

Brief generation uses a deterministic template over the approved fields rather than asking the model to rewrite the agreement. The result includes requirements, scope and exclusions, criteria, dependencies, data impact, references, revision/hash, and the two approval records. Users can read it in the app, copy Markdown, or download it.

The QA assistant then drafts scenarios from that approved snapshot. Every scenario references a real criterion index and quote, with steps and expected results. Supported positive, negative, boundary, and permission cases are presented for inspection; the assigned engineering lead reviews the draft.

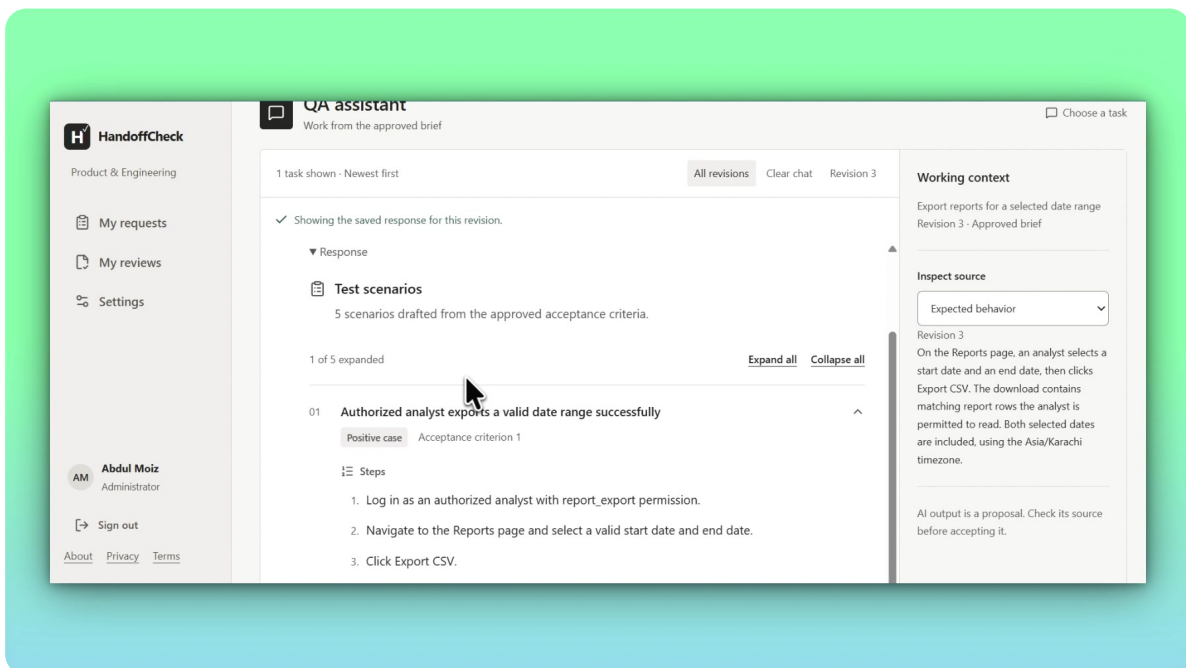


Figure 05. The QA assistant presents reviewable scenarios and their acceptance-criterion sources.

DRAFT TESTS VERSUS EXECUTED TESTS

Generated QA scenarios describe checks the team can run. Their existence does not mean the application behavior has been implemented, exercised, or passed.

10 / Make background work observable.

Provider tasks are stored before dispatch with bounded leases and attempts. Each claim rechecks access, revision, policy, and budget. A stale result remains historical and cannot advance a newer source revision. The interface distinguishes queued, working, retry-pending, and failed states.

TABLE 08 / Operation / Bound or recovery behavior

Operation	Bound or recovery behavior
AI usage	Initial allowances of ten attempts per actor/hour, fifty per workspace/day, and two concurrent calls
Provider call	Twenty-five-second timeout and bounded structured output
Transient review failure	At most three total attempts with scheduled thirty- and sixty-second eligibility delays
Invalid output or access/configuration failure	Visible failure requiring deliberate retry
Approval reminders	Author-set interval from fifteen minutes to seven days; only active pending leads are selected
Ambiguous Gmail send	Delivery unknown; no blind resend in the same cycle

Reminder timers follow the current revision and approval state. Clarification, a new draft, completed approval, archive, or pause suspends or stops the schedule. Optional email uses the author's separately connected Gmail identity. Private credentials and PKCE state remain server-side, and dispatch checks the current connection generation.

ge, an analyst selects a start date and an end date, then clicks Export
d contains matching report rows the analyst is permitted to read. Both
included, using the Asia/Karachi timezone.

V download, permission checks and export error messages.

cheduled exports and changes to existing permissions.

orized analyst, when they export a valid date range, the CSV contains only
s within that inclusive range.

ithout report_export permission, the export control is unavailable and
export attempts with HTTP 403.

ange containing no reports, the CSV contains column headers and no

✓ Engineering Approved

Implementation brief

Open brief →

▼ Approval reminder schedule

Reminders go only to assigned leads who have not approved. The author controls the repeat interval.

☒ Repeat reminders

☒ Email reviewers too

Repeat every

1 Days

Choose 15 minutes to 7 days. The timer starts when this revision is ready for approval.

Save reminder schedule

Figure 06. The author controls the reminder interval and whether email is enabled.

11 / Connect the team's existing tools.

TABLE 09 / Connection / Workflow / Explicit boundary

Connection	Workflow	Explicit boundary
Google Workspace	Configured Forms intake through Sheets into requests, followed by approved Docs export	Owner-operated Apps Script, signed bridge requests, and Google file-sharing controls
Gmail	Optional author-authorized approval reminders	Separate consent, encrypted server credentials, and delivery uncertainty tracking
Linear	Preview and explicitly confirm a handoff of approved work	Configured destination and backend credentials required
Windows app	Electron loads the hosted workspace and signs in through the system browser	Requires internet access; packaged delivery has separate release/signing configuration

App-created requests are excluded from automatic Google Sheet mirroring and Doc export; Form-origin requests use their configured Google pipeline. Bridge signatures, nonce replay checks, owner membership, artifact hashes, and export acknowledgements protect the integration path. External destinations retain their own access policies.

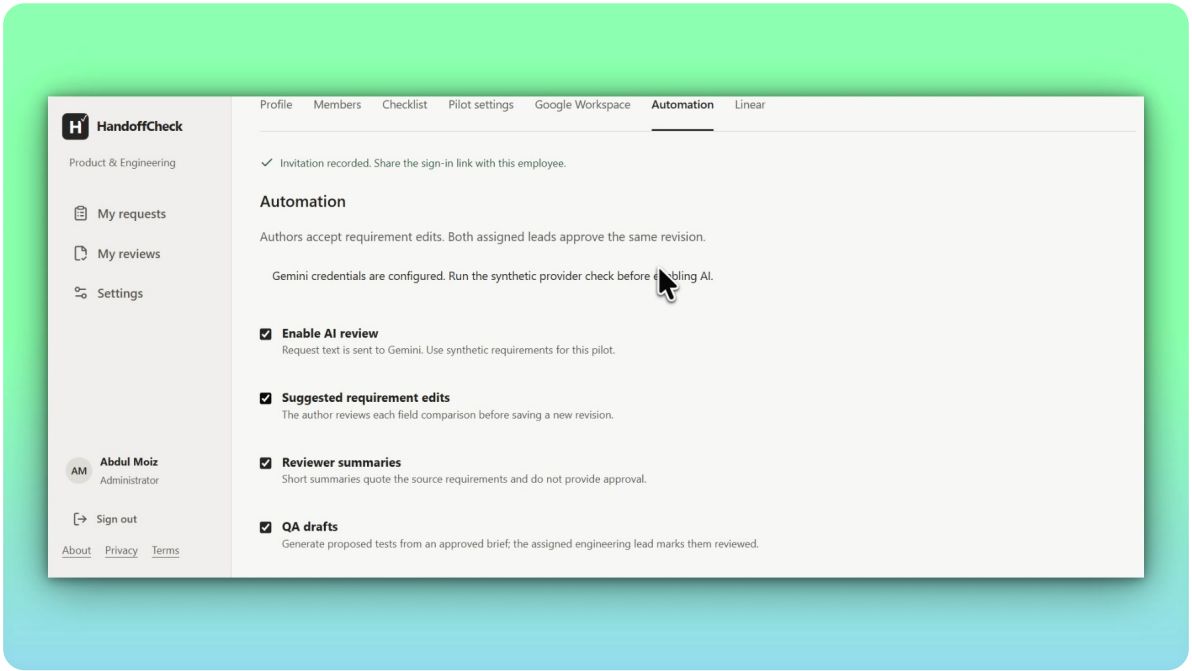


Figure 07. Workspace settings expose member roles, checklist versions, assistant features, and integration configuration.

PART VI / VERIFICATION & TRADEOFFS

12 / Test the workflow at its boundaries.

The project includes application/provider tests, database tests against PGlite with Supabase Auth/Storage substitutes, native authentication tests, and browser workflows. These exercise domain transitions, participant policies, stale revisions, source evidence, job recovery, quota behavior, and narrow-screen interaction.

TABLE 10 / Boundary / Useful verification

Boundary	Useful verification
Requirements	Incomplete drafts remain editable; mandatory gaps block submission or approval
Evidence	Invalid quotes, fields, criterion indexes, and oversized outputs are rejected
Decisions	Two different assigned leads approve the same revision; concurrent saves cannot reuse old approvals
Access	Authors, current leads, unrelated members, and administrators receive the correct view
Recovery	Lost responses, expired leases, provider failures, and new revisions preserve consistent state
Integrations	Consent, signing, replay, disconnect, and ambiguous delivery are tested separately

Synthetic provider tests verify contracts and failure handling; they do not establish live model wording quality. PGlite tests do not replace checks against a running Supabase installation. Real Google consent, external delivery, account configuration, and independent security review require their own evidence.

- Keep product facts, model suggestions, and human decisions distinguishable.
- Preserve the source revision through reviews, approvals, and exported artifacts.
- Use transactions for decisions that must agree under concurrent updates.
- Make provider errors, retries, and ambiguous external outcomes visible.
- Evaluate assistant usefulness with representative requirements and reviewer feedback before claiming productivity gains.