

CLIENT REPORT EXAMPLE / SEPTEMBER 10, 2026

Product quality & release readiness

Atlas Workspace

A fictional team workspace for founders and small teams managing projects, tasks and client delivery.

RELEASE RECOMMENDATION

**Fix signup and invitations.
Keep billing on hold.**

The core workflow has a useful foundation. The next release should make account creation reliable, team invitations predictable and recovery clear before opening the product to a wider audience.

12

PRIORITIZED RECORDS

03

HIGH PRIORITY

06

MEDIUM PRIORITY

03

LOW PRIORITY

ENGAGEMENT	DETAIL
Report ID	ODS-HQA-SAMPLE-002
Example release	0.9.0-rc2 · web application
Reader	Founder, product owner, design and engineering leads

About this sample. Atlas Workspace, its build and all reported outcomes are fictional. The report demonstrates the depth and format of an Odesa deliverable. No client engagement, human testing, user research, screenshots or recordings are represented as real.

01 / EXECUTIVE READOUT

The release decision

A practical view of what to fix, what to preserve and what still needs evidence.

RELEASE PATH	STATUS	WHY
Controlled free beta	Conditional	Resolve QA-01/02, retest the affected journeys and document acceptance of remaining issues.
Broad acquisition	Hold	Close the core signup and invitation defects, verify keyboard/recovery fixes and run a target-user round.
Paid launch	Hold	QA-03 requires a defined service and verified purchase-to-cancellation lifecycle.

THE THREE DECISIONS THAT MATTER

First, fix the moments where a new owner commits: creating an account and inviting a teammate. Second, make interrupted work recoverable. Third, keep the paid offer informational until access, cancellation and downgrade have acceptance evidence.

STRENGTHS TO PRESERVE

In the scenario, people can create a project, save a task and reopen it. Draft text remains available when saving fails. Core export content is retained. A free starting path and clear billing-off message make the offer understandable. These are scenario assumptions, not live verification.

KEEP THE DIAGNOSIS PRECISE

The offline example is a recovery-copy issue, not confirmed data loss. The export example concerns field meaning, not a failed download. The paid-lifecycle record is an untested release gate. The readability and return-experience records need further measurement or research.

NEXT DELIVERY CHECKPOINT

Engineering addresses the two High defects; QA reruns their exact reproductions and adjacent paths. Product confirms the paid-service contract and the plan for closing QA-03. The final go/no-go decision remains with the release owner.

Reading guide: scope 3 · journey coverage 4 · backlog 5 · detailed findings 6–11 · evidence examples 12–13 · product experience 14 · delivery plan 15 · retest 16 · validation and handoff 17.

02 / SCOPE AND METHOD

What this review covers

Example review contract. Environments below describe the intended test matrix; none were executed for this fictional report.

DIMENSION	REVIEW CONTRACT
Product boundary	Web onboarding, project/task work, invitations, error recovery, keyboard use, exports and paid-offer readiness.
Example environment matrix	Desktop Chromium at 1440×900; mobile emulation at 390×844 and 320×700; tablet at 768×1024. Actual browser/OS/build versions belong in the delivered evidence register.
Test data and accounts	Synthetic workspace owner and teammate accounts. No real customer content. Account credentials and access details belong in a separate secure channel.
Functional approach	Risk-based exploration and state-transition checks. Follow setup → first task → teammate → saved work → return. Test pending, empty, invalid and interrupted states.
Usability approach	Expert walkthrough of clarity, feedback, recovery, language and task priority. Distinguish observations from predictions about target users.
Accessibility approach	Keyboard/focus and responsive checks, plus measured readability against the agreed target. No conformance or assistive-technology certification is implied.
Commercial boundary	Review offer comprehension and lifecycle requirements. A pricing screen does not establish paid access, reliable billing or recurring value.

EVIDENCE AND CONFIDENCE

A real record identifies the exact build, environment, preconditions, steps, settled result and source artifact. An observation can be reproduced; its business impact may still be a hypothesis. Untested capabilities remain explicitly open.

Outside this sample: security/penetration tests, load tests, native apps, physical-device execution, external email delivery, live payment, representative user research and actual multi-day retention.

03 / JOURNEY COVERAGE

Follow the work, including recovery

“Clear” means no issue is included in that modeled path. All statuses on this page are fictional scenario outcomes, not completed tests.

JOURNEY / STATE	SCENARIO OUTCOME	RECORD / NEXT CHECK
Account creation · valid input	Clear	Confirm one account and the correct destination in a live run.
Account creation · invalid email	Issue	QA-01: valid form values clear on validation failure.
Workspace setup · Back/Forward	Issue	QA-04: browser Back exits the step sequence.
First project · no tasks	Friction	QA-08: weak empty-state guidance.
Task creation · save/reload	Clear	Compare exact task content after settled save and reload.
Task editor · keyboard/modal	Issue	QA-06: focus remains behind the dialog.
Task update · offline/retry	Issue + recovery	QA-05: draft retained; recovery wording is unclear.
Invite teammate · slow/repeat	Issue	QA-02: duplicate pending invitations. Email delivery excluded.
Mobile labels and layout	Needs measurement	QA-07: check readability and record actual metrics.
Export project · CSV content	Clarity issue	QA-11: internal headers; preserve content exactness.
Return visit · next action	Needs research	QA-09: interface-priority hypothesis.
Pro · paid account lifecycle	Untested	QA-03: purchase, access, cancel and downgrade gate.

Coverage gaps to close before broad release: physical iOS/Android paths, actual authentication/recovery, invitation delivery and permissions, accessibility measurements, and the authorized billing lifecycle. Scope these explicitly rather than treating absence from this table as a pass.

04 / CONSOLIDATED FINDINGS

One prioritized backlog

12 records: 3 High, 6 Medium and 3 Low. No Critical finding is included. Priority combines consequence, recovery and release timing.

ID / PRIORITY	FINDING	CLASSIFICATION	P.
QA-01 High	An invalid email clears the entire signup form	Functional defect	6
QA-02 High	Repeated invitation clicks create duplicate pending invites	Functional defect	6
QA-03 High	The paid account lifecycle has no acceptance evidence	Release gate: untested capability	7
QA-04 Medium	Browser Back exits workspace setup	Navigation defect	7
QA-05 Medium	A failed save exposes an error with no recovery guidance	Recovery and interface copy	8
QA-06 Medium	The task dialog loses the keyboard's place	Keyboard accessibility	8
QA-07 Medium	Secondary labels are hard to distinguish at narrow widths	Readability concern: measurement pending	9
QA-08 Medium	An empty project gives the owner no next action	First-use experience	9
QA-09 Medium	Returning owners get activity, but no clear priority	Return-experience opportunity	10
QA-10 Low	The saved toast briefly covers project navigation	Transient interaction defect	10
QA-11 Low	CSV headers obscure otherwise usable task data	Export clarity	11
QA-12 Low	Task and item labels describe the same object	Interface consistency	11

High: resolve before the affected launch path proceeds. **Medium:** meaningful friction, access or recovery concern. **Low:** limited interruption or clarity issue. A High untested gate carries a different evidence status from a reproduced defect.

05 / DETAILED FINDINGS

QA-01 + QA-02

QA-01 / HIGH / FUNCTIONAL DEFECT

An invalid email clears the entire signup form

Owner: Frontend engineering · **State:** Open in scenario

Where: /signup · desktop and mobile

Reproduce. Enter a name, company and password. Enter an email without a domain. Select Create account. Correct the email and attempt to continue.

Observed in scenario. The form returns to an empty state with a general error. Valid name and company values are lost along with the invalid email.

Expected. Keep appropriate non-sensitive values, identify the email error beside its field and let the person correct it. Apply the product's password-handling policy.

Why it matters. Account creation becomes repeat work at the first commitment point. Abandonment is a plausible consequence; this sample assigns no conversion-loss estimate.

Recommended change. Preserve form state through validation failure. Attach the error to the field, move focus appropriately and keep the submit action available after correction.

Retest acceptance. Invalid email produces a field-specific error. Name and company remain unchanged. Correcting the email creates one account; keyboard and narrow-screen paths also succeed.

Evidence: E-01 · schematic on page 12

QA-02 / HIGH / FUNCTIONAL DEFECT

Repeated invitation clicks create duplicate pending invites

Owner: Frontend + backend engineering · **State:** Open in scenario

Where: /workspace/setup · Invite teammates

Reproduce. Enter one teammate email. On a slow connection, select Send invite twice before the first response. Open the pending-invitations list.

Observed in scenario. No pending feedback appears. Two identical pending-invitation rows are created for the same workspace and email. Email delivery is outside this scenario.

Expected. A pending state appears immediately. One deliberate action creates one invitation. Repeated requests cannot create duplicate active invitations.

Why it matters. An uncertain action becomes an untidy team record and creates avoidable cleanup. Potential duplicate emails require a separate delivery check.

Recommended change. Add visible pending/success/failure states and protect the write against retries. Provide a deliberate resend action for an existing invitation.

Retest acceptance. Double-click, keyboard repeat and retry after a delayed response each leave one pending invitation. A failed request can be retried. Resend does not create a second member record.

Evidence: E-02 · schematic on page 12

05 / DETAILED FINDINGS

QA-03 + QA-04

QA-03 / HIGH / RELEASE GATE: UNTESTED CAPABILITY

The paid account lifecycle has no acceptance evidence

Owner: Product + billing engineering · **State:** Untested in scenario

Where: /settings/billing · Pro preview

Reproduce. Open the Pro offer from a free workspace. Inspect upgrade, account management and the stated access rules. Review the proposed purchase-to-cancellation test plan.

Observed in scenario. The scenario includes an informational offer with billing off. Purchase, entitlement activation, renewal, recovery, cancellation and downgrade have no execution record.

Expected. Before billing opens, the service, checkout terms and account lifecycle work together and have a traceable acceptance record.

Why it matters. Paid readiness cannot be established. The informational preview may remain available; the gap is a launch requirement, not evidence of an incorrect charge.

Recommended change. Define plan limits and cancellation behavior. Complete authorized sandbox lifecycle tests, including provider events and access readback, before a limited paid pilot.

Retest acceptance. Purchase grants the correct access once; retries do not duplicate it. Cancel stops renewal. The access-end date is clear, and downgrade preserves the agreed read/export path.

Evidence: No transaction or provider receipt supplied

QA-04 / MEDIUM / NAVIGATION DEFECT

Browser Back exits workspace setup

Owner: Frontend engineering · **State:** Open in scenario

Where: /workspace/setup · step 3 of 4

Reproduce. Advance from workspace details to teammate invitations and then the first project. Wait for the saved indicator. Press browser Back, then Forward.

Observed in scenario. Back returns to the dashboard rather than the previous setup step. Forward restores the third step. The scenario does not establish loss of settled saved data.

Expected. Browser navigation follows the visible step sequence and retains completed values. The in-app Back control and browser history have consistent meaning.

Why it matters. A person trying to correct an earlier answer leaves setup unexpectedly. The product asks them to reconstruct where they were.

Recommended change. Represent setup steps in navigation state and restore both the step and its saved values when history changes.

Retest acceptance. Steps 1 → 2 → 3 reverse to 2 → 1 and move forward again with values retained. Repeat after reload. Dashboard history still behaves correctly.

Evidence: Required: before/after route and retained-state record

05 / DETAILED FINDINGS

QA-05 + QA-06

QA-05 / MEDIUM / RECOVERY AND INTERFACE COPY

A failed save exposes an error with no recovery guidance

Owner: Frontend engineering + UX writing · **State:** Open in scenario

Where: /projects/launch · task editor

Reproduce. Open a task and change its description. Make the browser offline. Select Save changes. Restore the connection, retry, wait for success and reload.

Observed in scenario. The editor shows “Failed to fetch.” The draft remains in the open form. A settled retry retains the change in the modeled recovery path.

Expected. Explain that saving failed, where the draft remains and how to retry. Show saved confirmation only after the write completes.

Why it matters. The person cannot tell whether it is safe to close the editor. The defect is unclear recovery; the retained draft should be preserved.

Recommended change. Use: “We couldn’t save your changes. They’re still in this open window. Reconnect, then try again.” Offer Retry and keep the editor open on failure.

Retest acceptance. Offline save has no success state. Draft text remains exact. Reconnect and Retry complete one update; reload shows the same text. Closing an unsaved draft asks for confirmation.

Evidence: E-03 · schematic on page 13

QA-06 / MEDIUM / KEYBOARD ACCESSIBILITY

The task dialog loses the keyboard’s place

Owner: Frontend engineering + accessibility QA · **State:** Open in scenario

Where: /projects/launch · Edit task dialog

Reproduce. Use Tab to focus Edit task and press Enter. Continue tabbing through the dialog. Press Escape and attempt to continue from the task card.

Observed in scenario. Focus remains on the page behind the open dialog. Tabbing reaches background controls. Closing the dialog does not return focus to Edit task.

Expected. Opening the dialog moves focus into it. Keyboard movement stays within the active modal. Closing returns focus to its initiating control.

Why it matters. The visible task and keyboard location disagree. A keyboard user may interact with the wrong controls or lose the path back to their work.

Recommended change. Use a named dialog with deliberate initial focus, contained tab order and focus restoration. Keep unsaved-change confirmation in the same interaction sequence.

Retest acceptance. Open, edit, save, cancel and Escape work with keyboard alone. Focus stays visible, never reaches the background while modal, and returns to the correct task control.

Evidence: E-04 · schematic on page 13

05 / DETAILED FINDINGS

QA-07 + QA-08

QA-07 / MEDIUM / READABILITY CONCERN: MEASUREMENT PENDING

Secondary labels are hard to distinguish at narrow widths

Owner: Product design + accessibility QA · **State:** Needs measurement in scenario

Where: /projects · mobile navigation and metadata

Reproduce. Open Projects at a narrow viewport. Inspect inactive navigation, task due dates and help links against their backgrounds. Repeat with larger text and reduced brightness.

Observed in scenario. The scenario uses small, muted secondary labels that lose visual prominence. No measured contrast ratio or automated accessibility result is supplied.

Expected. Navigation and task context remain readable, including secondary states. Active and inactive controls remain distinct without relying on faint text.

Why it matters. People may miss a due date or navigation option. Confirm the concern with measurements and assistive-technology testing before assigning a standards failure.

Recommended change. Measure the actual combinations, adjust text tokens and review type size. Verify across light/dark states if both are offered.

Retest acceptance. Record before/after measurements against the agreed accessibility target. At larger text, labels remain readable and controls do not overlap. Keep measured results with the retest.

Evidence: Required: exact text/background pairs and font sizes

QA-08 / MEDIUM / FIRST-USE EXPERIENCE

An empty project gives the owner no next action

Owner: Product design + frontend engineering · **State:** Open in scenario

Where: /projects/new · zero-task state

Reproduce. Create a workspace and its first project. Skip teammate invitations. Open the project before adding any tasks.

Observed in scenario. A blank board shows “No data.” The primary Add task action is available only in a compact toolbar, with no explanation of how to start.

Expected. The empty state explains what belongs here and offers a clear action that produces the first useful result.

Why it matters. A new owner has finished setup but still needs to work out how to use the product. The screen leaves the first useful task to discovery.

Recommended change. Add “Start with one task” and a visible Create task action. Offer an optional example without filling the workspace with unexplained sample data.

Retest acceptance. An owner can create and reopen the first task from the empty state. The message changes after creation; search-empty and permission-empty states use separate explanations.

Evidence: Required: first-project empty-state capture

05 / DETAILED FINDINGS

QA-09 + QA-10

QA-09 / MEDIUM / RETURN-EXPERIENCE OPPORTUNITY

Returning owners get activity, but no clear priority

Owner: Product design + research · **State:** Hypothesis in scenario

Where: /dashboard · return after first project

Reproduce. Create a project, assign tasks and leave one invitation pending. Return to the dashboard the next day or use seeded equivalent state for interface review.

Observed in scenario. Activity, workspace settings and project cards have equal weight. Nothing summarizes overdue work, pending invitations or the next useful action.

Expected. The dashboard helps the owner understand what changed and choose a relevant next step, while preserving access to the full project list.

Why it matters. The return visit may feel like starting from scratch. This is a design hypothesis; seeded or simulated days cannot establish actual retention.

Recommended change. Prioritize a small, factual “Needs your attention” group. Use real task and invitation states; let users dismiss items that are irrelevant.

Retest acceptance. In formative research, returning owners can explain what needs attention and complete a chosen next action. Observe real repeat use before attributing a retention benefit.

Evidence: Required: target-user return-session observation

QA-10 / LOW / TRANSIENT INTERACTION DEFECT

The saved toast briefly covers project navigation

Owner: Frontend engineering · **State:** Open in scenario

Where: /projects/launch · after task save

Reproduce. Save a task, then immediately select the neighboring project-navigation link while the success toast is visible.

Observed in scenario. The toast sits over the navigation target and receives the click. Once it disappears, navigation works normally.

Expected. Confirmation remains visible without covering the next likely action or preventing pointer and keyboard navigation.

Why it matters. A successful action is followed by a brief dead end. The scenario shows interruption rather than lost work.

Recommended change. Move confirmation into a reserved message area or a location that does not overlap controls. Avoid a focus change for passive success feedback.

Retest acceptance. After save, the same navigation target works immediately with pointer and keyboard. The message remains available long enough to read and does not cover mobile controls.

Evidence: Required: overlap capture and pointer-target check

05 / DETAILED FINDINGS

QA-11 + QA-12

QA-11 / LOW / EXPORT CLARITY

CSV headers obscure otherwise usable task data

Owner: Backend engineering + product · **State:** Open in scenario

Where: /projects/launch/export · CSV

Reproduce. Create a task with an owner and due date, then export the project. Open the downloaded CSV and compare it with the visible task.

Observed in scenario. The modeled export retains task content but labels fields with internal keys such as assignee_id and due_at, without a human-readable owner or timezone context.

Expected. Exported records remain useful outside the app. Names, stable identifiers and date/time meaning are clear.

Why it matters. Someone reviewing or handing off the file needs additional interpretation. This is an export-semantic issue, not a broken download.

Recommended change. Add readable column names and owner display values alongside identifiers. Define date formatting and timezone behavior in the export contract.

Retest acceptance. Compare Unicode text, commas, line breaks, unassigned tasks and due dates against source records. Content remains exact; spreadsheet opening does not change intended values.

Evidence: Required: actual downloaded CSV and source record comparison

QA-12 / LOW / INTERFACE CONSISTENCY

Task and item labels describe the same object

Owner: UX writing + frontend engineering · **State:** Open in scenario

Where: Project board, creation dialog and confirmation

Reproduce. Open the board, create a task, save it and inspect its confirmation and empty-state messages.

Observed in scenario. The board says Add task, the dialog says New item, and confirmation says Record created. Each refers to the same project task.

Expected. The same object keeps the same name throughout the journey. Confirmation describes the result in the user's vocabulary.

Why it matters. A small terminology mismatch adds avoidable interpretation during onboarding and makes help documentation harder to follow.

Recommended change. Use task consistently: Create task, Task details and Task created. Apply the same vocabulary to validation, search and export help.

Retest acceptance. Review the complete create/edit/delete journey and associated help. Task is used consistently, while project and workspace retain their distinct meanings.

Evidence: Required: label inventory across the task journey

06 / EVIDENCE EXAMPLES

Make the finding easy to reproduce

Reconstructed interface schematics for this fictional product. These are explanatory visuals, not screenshots, recordings or test results.

E-01 / SIGNUP VALIDATION / QA-01

ATLAS WORKSPACE / BEFORE	ATLAS WORKSPACE / AFTER
NAME / COMPANY Alex / Launch team	NAME / COMPANY Both fields empty
EMAIL alex@	EMAIL ERROR Please check your details

1. The email is invalid. 2. The scenario clears valid values too. Acceptance: preserve the appropriate fields and explain the correction beside Email.

E-02 / INVITATION FEEDBACK / QA-02

ATLAS WORKSPACE / BEFORE	ATLAS WORKSPACE / AFTER
TEAMMATE alex@example.test	PENDING INVITATION 1 alex@example.test
ACTION Send invite clicked twice	PENDING INVITATION 2 alex@example.test

1. The first request is still pending. 2. A second click produces a duplicate row in the scenario. Acceptance: one active invitation, with clear pending and success feedback.

In the delivered evidence: preserve the actual before/after state, network timing, relevant record identifiers and any redacted recording. Duplicate rows alone do not establish duplicate email delivery.

06 / EVIDENCE EXAMPLES CONTINUED

Show the failure and the recovery

The examples below model interface behavior. A real report ties each image or clip to its finding ID, build and reproduction.

E-03 / INTERRUPTED SAVE / QA-05

ATLAS WORKSPACE / BEFORE	ATLAS WORKSPACE / AFTER
TASK DESCRIPTION Prepare the launch checklist	ERROR Failed to fetch
ACTION Save while offline	DRAFT Text remains in open editor

The retained text is a strength. The missing element is a clear recovery instruction. Acceptance: retry after reconnect, wait for success, then reload and compare exact text.

E-04 / KEYBOARD FOCUS / QA-06

BACKGROUND PAGE	VISIBLE TASK DIALOG
Edit task Project navigation Focus stays behind modal	Title field Description field Save changes / Cancel

The highlighted area represents the wrong keyboard location. Acceptance: opening places focus inside the dialog; closing restores focus to the initiating task control.

EVIDENCE REGISTER

E-01 to E-04 are schematics only. A delivered report links actual captures, recordings, logs and inspected outputs to each finding, with build and access details.

07 / PRODUCT EXPERIENCE AND OFFER CLARITY

The first useful task is the starting point

Functional QA tells the team whether a path works. The product review asks whether the next action and its value are clear.

MOMENT	EXPERIENCE TO DELIVER	VALIDATION
First visit	Create a workspace and one useful task. Explain what the empty board is for. Keep optional team setup skippable.	Observe a new owner without coaching. Can they create and reopen a task? QA-01/08/12.
First teammate	Make invite status visible, prevent duplicates and explain the difference between pending and joined.	Follow owner → invite → teammate acceptance → shared task with synthetic accounts. QA-02.
Next-day return	Show useful changes and a short factual priority list. Avoid presenting every dashboard card as equally urgent.	Observe real returning users. Seeded state can review layout, not retention. QA-09.
End of first week	Help the team review completed and outstanding work. Keep their content available and exportable.	Ask what became easier and inspect actual repeated task use. QA-11.
Pro consideration	Explain the specific ongoing benefit, limits and free continuation. Keep billing off until the lifecycle is accepted.	Separate offer comprehension, purchase intent, actual purchase and continued value. QA-03.

PRODUCT RECOMMENDATION

Lead with dependable team coordination: a clear first task, a reliable invitation and useful returning context. Additional AI features should support an identifiable job in that workflow. The presence of AI does not establish usefulness or justify a recurring charge.

Research questions: What brought the owner here? What result would make this useful today? What would a teammate need to trust it? What changes enough each week to support continued use? Record the answers before choosing premium positioning.

08 / DELIVERY SEQUENCE

A fix plan your team can pick up

Role assignments below are suggested ownership. The client confirms named owners and dates after reviewing dependencies.

ORDER / OWNER	WORK PACKAGE	EXIT EVIDENCE
1 · Engineering	QA-01/02: preserve signup state; make invite creation predictable under slow and repeated requests.	Exact failed path and adjacent valid path pass on the release candidate; record and delivery checks scoped separately.
2 · Engineering + QA	QA-04/05/06/10: setup history, interrupted-save recovery, modal focus and confirmation placement.	Before/after captures plus state readback; keyboard completion and retry after reconnection.
3 · Design + writing	QA-07/08/12: measure readability, improve empty-state guidance and normalize task vocabulary.	Measurements and responsive review; uncoached first-task check; label inventory updated.
4 · Product + engineering	QA-11: specify readable export fields without changing underlying content meaning.	Inspect the actual file against source records, including special characters and date conventions.
5 · Product + research	QA-09: test a clearer return dashboard with target owners and teammates.	Observed next-action comprehension and real repeat use. Keep uplift unclaimed until measured.
Before billing · Product + billing engineering	QA-03: define ongoing service, access limits and the complete paid account lifecycle.	Authorized lifecycle tests, provider readback and an approved limited-pilot decision.

ESTIMATE AFTER TRIAGE

The backlog is ready for engineering estimation once the team reviews the implementation. Effort, cost and delivery dates are intentionally unquoted; they depend on the code, backend behavior and agreed regression coverage.

Implementation support: your engineers can work from these records directly. If you want Odesa to make the changes, we scope that work separately against the same acceptance criteria. The report remains useful whichever team implements it.

09 / RETEST AND RELEASE GATES

Close findings with evidence

Every record starts unretested in this sample. A fix is complete only after the agreed acceptance check passes on the candidate build.

CHECKPOINT	RETEST CONTRACT
Identity and build	Record finding ID, candidate version, environment, reviewer, date and linked result. Keep the original reproduction.
Exact failure + nearby success	QA-01: invalid and valid signup. QA-02: delay, repeat, failure/retry and deliberate resend. Confirm one intended result.
Navigation + persistence	QA-04/05: browser history, settled save, offline error, reconnect/retry and reload. Separate interrupted requests from saved-state loss.
Keyboard + presentation	QA-06/07/10: open/close and unsaved modal states, focus restoration, readability measurement and controls beneath confirmation.
Clarity + output	QA-08/11/12: first-task flow, distinct empty states, exact CSV content and consistent labels.
Research-dependent record	QA-09 remains a hypothesis until the specified target-user observation. Implementing a dashboard card alone does not close it.
Paid lifecycle	QA-03: authorized sandbox purchase, entitlement, duplicate event, payment failure/recovery, cancellation, access end and downgrade. Keep provider evidence.

DISPOSITION VOCABULARY

Pass: the specified check succeeds. Fail: the issue remains or a regression appears. Blocked: a dependency prevents execution. Not retested: no rerun performed. Accepted risk: the release owner records rationale, scope and review date; this is separate from a pass.

Release readout: publish the current candidate, closed/open findings, blocked coverage and any accepted risks. The product owner makes the launch decision with those limits visible. A sample score, screenshot or “fixed” comment cannot substitute for this record.

10 / VALIDATION AND HANDOFF

What happens after the report

A focused release check finds actionable problems. Target-user sessions and continued use answer the questions that expert review leaves open.

RUN A FORMATIVE USER ROUND

Recruit prospective workspace owners and teammates who match the intended customer. Include people unfamiliar with the product and a range of device/access needs. Give them realistic tasks: create a project, invite a colleague, recover an interrupted edit and find their next priority. Capture consented recordings, task outcomes and the moments they need help.

MEASURE OUTCOMES AT THE RIGHT POINT

Distinguish account creation success from clicking Create account; invitation acceptance from clicking Send invite; a persisted task from clicking Save; paid access from opening Pro. Exclude QA sessions. Collect only the event data needed for the decision, without task text, credentials or invitation addresses in analytics.

KEEP OBSERVED FACTS AND HYPOTHESES SEPARATE

A small research round can reveal friction and misunderstood language. It cannot establish population-wide conversion, month-two retention or willingness to pay. Observe real elapsed use and actual voluntary purchases only after the relevant product and billing gates are met.

DELIVERY ITEM	WHAT THE TEAM RECEIVES
Decision report	Scope, strengths, prioritized findings, recommended changes and release conditions.
Issue backlog	One record per issue with owner role, reproduction, impact and retest acceptance.
Evidence bundle	Actual screenshots, consented recordings, logs and inspected outputs, linked by finding ID when included in scope.
Readout + retest record	Agreed priorities, implementation questions, candidate build, outcomes and remaining risks.

Sample limits. This expanded example spans functional QA, expert usability, accessibility concerns and paid-readiness planning. Actual depth and deliverables depend on the agreed engagement. No live product, tester activity or commercial outcome is claimed. Native apps, security review, load testing and specialized research require their own scope.

Discuss your Human QA scope with Odesa →