

# From scattered requests to accountable delivery.

A complete improvement plan for a fictional 25-person B2B SaaS company.

Illustrative composite. Company details, request records, estimates and scenarios are synthetic. This is a proposed operating design, not an implemented customer result.

## Inside this package

Company brief / Diagnosis / Proposed workflow / Seven operating steps / Technology / Responsibilities / Rollout / Measurement / Sources

Baseline: limited engineering capacity. Existing tools remain in place. Automation follows a tested manual process.

12 October 2026 · withweft.app

# The company and its constraints

A self-service B2B reporting product has 25 employees, including six engineers, one product manager and three support/customer-success staff. The product serves individual users and business accounts.

Requests arrive in Intercom and Slack. Notion holds specifications and feedback notes; Jira tracks engineering work; GitHub holds code. An internal panel controls release availability. PostHog holds some usage events.

## Planning assumptions

- 100 feedback items per month
- 4 engineering hours per week for process improvements
- US\$200/month ceiling for added tools
- Existing tools remain in place
- Product manager owns the process

Budget is a scenario constraint, not a vendor quote. Account identifiers, API access, existing subscription entitlements and analytics coverage still need discovery.

## Objective

Make every accepted customer need traceable from source evidence to a delivery decision, confirmed release and assigned follow-up. Do this first with a controlled manual process, then automate only repeated and verified steps.

## Boundary

This blueprint covers one product and one request-to-follow-up workflow. It does not deliver code changes, operate customer support, replace product strategy, or promise a retention uplift.

# Where the process breaks

Intercom + Slack Messages arrive Repeated accounts look like new demand

→

Notion Someone summarizes Source evidence disappears

→

Jira + GitHub Work gets completed Done has several meanings

→

Internal release Availability changes Support has no dependable signal

These are designed failure modes for the sample. A real engagement would confirm them by tracing records and interviewing the people responsible.

## Sample evidence ledger

| Records                    | Customer evidence                              | Correct interpretation                                                           |
|----------------------------|------------------------------------------------|----------------------------------------------------------------------------------|
| R01–R05 · account A        | Five messages requesting scheduled CSV exports | One account requesting a recurring reporting feature                             |
| R06–R08 · accounts B, C, D | Each reports missing rows in exports           | Three accounts affected by a potential correctness bug; investigate shared cause |
| R09 · account E            | Requests branded PDF reports                   | A separate presentation need                                                     |
| R10 · account B            | Asks when the missing-row fix will arrive      | A status request linked to existing evidence, not a new vote                     |
| R11 · identity unknown     | “Export doesn’t work”                          | Request clarification; do not infer account or merge automatically               |

Download the synthetic request records ↓

# Every handoff has an owner.

Read the numbered lanes from top to bottom. The branches describe what happens when a request cannot move forward.

01 · Support Capture original message → attach source reference → identify account → assign triage owner

Enough context? No → ask the customer and hold in “needs clarification.” Yes → review themes.

02 · Product + AI assistance Suggest themes and duplicates → review evidence → separate bugs from features → count distinct known accounts

Existing need? Yes → attach evidence to it. No → create a candidate need. Uncertain matches stay separate for review.

03 · Product Assess impact, product fit, urgency and effort → record approve / investigate / defer / decline decision

Approved? No → communicate the decision and set a review date if deferred. Yes → agree scope and acceptance criteria.

04 · Engineering Link Jira work and specification → implement → test → review PR → record completion

Review fails or scope changes? Return to implementation or product approval. Preserve the original request links.

05 · Release owner Confirm deployment → verify account eligibility and feature flags → record availability evidence

Available to the customer? No → hold the release message. Yes → prepare a customer update.

06 · Support Review update draft → send through original channel → record sender, timestamp and response

07 · Product Check customer confirmation and available usage signals → record solved / partial / unresolved / unknown → feed learning into next review

# Capture and clarify

## Trigger

A customer reports a need in Intercom or Slack.

## Owner

Support lead

## Inputs

Original message, source reference, account identifier or unknown.

## Action

Record the problem in the customer's words. Link the source and assign an owner. Ask for reproduction steps or desired outcome when unclear.

## Output

A traceable request, ready for review or marked needs clarification.

## Exception

Unknown identity stays unknown. Restrict private source access; do not copy sensitive content to a broader audience.

## Acceptance check

Every request has a source, owner, received time and identity status.

# Review themes and duplicates

## Trigger

A request has enough context for triage.

## Owner

Product manager

## Inputs

Request records and existing need register.

## Action

AI may suggest matches. Review each suggested merge, retain every source and count distinct known accounts. Separate correctness bugs from enhancement requests.

## Output

Reviewed need with linked evidence and distinct-account count.

## Exception

Ambiguous matches remain separate. Repeated messages add context, not new accounts.

## Acceptance check

R01–R05 count as one account; R06–R08 retain three accounts; R09 remains separate.

# Make and record the decision

## Trigger

A need is ready for prioritization.

## Owner

Product manager with engineering input

## Inputs

Customer impact, product fit, urgency, evidence strength and rough effort.

## Action

Choose approve, investigate, defer or decline. Record the reason, owner and review date where applicable. Scope approved work and define observable acceptance criteria.

## Output

Decision and specification linked to evidence.

## Exception

Urgent bugs follow an agreed escalation route. Large-customer requests still require product-fit review.

## Acceptance check

A reviewer can explain the decision from the record without reconstructing a Slack conversation.

# Implement with context

## Trigger

Product approves a scoped need.

## Owner

Engineering lead and assigned engineer

## Inputs

Specification, source links, acceptance criteria and linked Jira issues.

## Action

Estimate and assign work. Link all required issues and PRs to the need. Implement, test and obtain review. Escalate scope changes to product.

## Output

Reviewed implementation with technical completion recorded.

## Exception

A need with several issues stays incomplete until all required work is ready. Rejected reviews return to implementation.

## Acceptance check

Original evidence remains accessible and required tests are recorded.

# Confirm customer availability

## Trigger

Engineering reports completion.

## Owner

Release owner

## Inputs

Deployment reference, rollout settings and affected accounts.

## Action

Check the internal release panel. Record when each affected account can access the change and any restrictions.

## Output

Account-specific release confirmation.

## Exception

Merged code or a done task alone is insufficient. Hold messages if availability is uncertain or deployment rolls back.

## Acceptance check

Partial rollout permits notifications only for eligible accounts.

# Follow up with the customer

## Trigger

Availability is confirmed for an affected account.

## Owner

Support lead

## Inputs

Original request, approved release details and eligible recipient.

## Action

Prepare a draft describing what changed and how to access it. Review and send through the appropriate existing channel. Record timestamp, sender and reply.

## Output

Sent update or assigned exception.

## Exception

Prevent duplicate messages. Never promise unsupported functionality or send to an unrelated account.

## Acceptance check

Each eligible requester has one tracked update or an owned reason for delay.

# Review the outcome

## Trigger

The agreed review date arrives after release.

## Owner

Product manager

## Inputs

Customer replies, valid usage events and the original acceptance criteria.

## Action

Check whether the original need was met. Separate adoption from resolution. Record solved, partial, unresolved or unknown and the next action.

## Output

Evidence-backed outcome and follow-on decision.

## Exception

If identity mapping or instrumentation is missing, use customer follow-up and record the measurement gap.

## Acceptance check

No claim of success is based only on shipping the feature.

# Start with access and evidence.

Available implementation capacity

These are authored example variants, not an automated assessment of your company.

| System              | Proposed role                                                                            | Verify before implementation                                                                                     |
|---------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Intercom / Slack    | Keep original conversations; record source links and account identity                    | Export permissions, channel coverage, retained access and private-channel restrictions                           |
| Notion              | Structured evidence and decision register for the initial pilot; preserve specifications | Owner, field schema, link visibility and update discipline; evaluate whether Jira can absorb this register later |
| Jira                | Engineering execution; link one need to all required issues                              | Project permissions, custom fields, existing integrations and agreed status definitions                          |
| GitHub              | Code, PR review and technical checks                                                     | Repository permissions and current Jira linking; merging is not evidence of customer availability                |
| Internal panel      | Release owner records actual availability per affected account                           | Export/API availability, deployment references and staged rollout rules                                          |
| PostHog             | Optional evidence for the agreed product outcome                                         | Account mapping, event definitions, consent requirements and missing instrumentation                             |
| Optional automation | Draft classification and updates; synchronize approved records                           | Existing licenses first; quote usage and maintenance before selecting a vendor                                   |

## Minimum data contract

Request: request ID, source reference, original wording, received time, account ID or unknown, owner, need ID. Need: problem, evidence links, distinct account count, decision and reason. Delivery: linked issues, acceptance criteria, release reference and account availability. Follow-up: owner, channel, approved message, sent time, response and outcome.

## Integration safeguards

Use source IDs to prevent duplicate imports. Log failed transfers in a review queue. Keep customer-facing sends behind approval. Define which system owns each field; do not allow competing status updates to loop. Restrict source-record access to authorized roles. A named maintainer reviews errors weekly and after permission or schema changes.

## Cost and effort assumptions

Baseline uses existing licenses; incremental vendor cost is unknown until current entitlements are checked. Illustrative setup effort: 8–12 engineering hours and 8–12 hours across product, support and release ownership over three weeks. This excludes product feature development. The US\$200/month ceiling must include any automation/AI usage. If a quote exceeds it, reduce automation scope and keep the manual fallback.

## Baseline recommendation

Validate the register using authorized exports or manual source references. After two review cycles, automate one repeated handoff within the four-hour weekly engineering allocation. Keep release confirmation manual.

# Responsibility survives automation.

| Role                   | Responsible for                                                     | Decision authority                                                                         |
|------------------------|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Product manager        | Workflow, theme review, priorities and outcome reviews              | Approves scope and deferrals; resolves conflicting priorities                              |
| Support lead           | Source capture, clarification, follow-up queue                      | Approves customer-facing messages                                                          |
| Engineering lead       | Estimates, acceptance checks and integration maintenance assignment | Approves technical approach and readiness                                                  |
| Assigned engineer      | Implementation and tests                                            | Reports completion and blockers; cannot declare customer availability by task status alone |
| Release owner          | Deployment and account eligibility evidence                         | Confirms availability; may be an existing engineer on rotation                             |
| Founder / budget owner | Tool spend and capacity allocation                                  | Approves purchases and expanded scope                                                      |

AI suggests classifications and draft text. It does not own prioritization, release confirmation, or customer commitments. Assign a named backup for triage and release checks during absences.

# Prove the process before automating it.

## WEEK 1

### Define and baseline

Trace sample records, confirm access, agree statuses and owners, create the evidence register.

Gate: each sample record has a source, owner and identity status.

## WEEK 2

### Run a manual pilot

Review a small request batch, link accepted needs to Jira and practice release confirmation.

Gate: ambiguous requests, deferrals and partial rollouts follow explicit branches.

## WEEK 3

### Automate selectively

Automate only the most repetitive verified handoff. Check failures, duplicate events and permissions.

Gate: maintainer can recover a failed transfer and revert to the manual process.

Timeline is illustrative and depends on access and the four-hour weekly engineering allocation.

Instrumentation gaps or custom APIs can extend it. Changes to the SaaS product itself are outside this setup estimate.

### Handover checklist

Named owners and backups • Approved schema and status definitions • Recorded manual walkthrough • Integration access inventory • Error queue and recovery instructions • Acceptance results • First review date

# Measure the handoffs we changed.

| Metric                   | Definition                                                                            | Proposed pilot check                                             |
|--------------------------|---------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Evidence completeness    | Accepted needs with original source and known/unknown account status + accepted needs | All sampled accepted needs pass                                  |
| Decision delay           | Time from receipt to first recorded product decision                                  | Measure baseline first; compare similar requests                 |
| Follow-up coverage       | Eligible requesters contacted + eligible requesters for confirmed releases            | Every eligible requester has a sent record or assigned exception |
| Manual coordination time | Observed time spent copying, chasing and reconciling records                          | Compare equivalent batches; report actual observations           |
| Outcome evidence         | Released needs with customer confirmation or valid usage evidence                     | Unknown remains an explicit result                               |

## Acceptance scenarios to execute

- Repeated messages from one account count once.
- Different export needs stay distinct until reviewed.
- Missing account identity triggers clarification.
- One completed issue does not close a need with unfinished linked work.
- Merged code without release confirmation does not authorize a release message.
- A partial rollout contacts only eligible accounts.
- Retrying an import does not create a second record.
- A failed integration creates a visible exception with an owner.

These are proposed implementation acceptance checks, not reported results. The sample page has no live integrations. Retention and revenue effects cannot be inferred from these checks.

# What this example is based on

- Linear, November 6, 2025: a first-person description of intake, triage ownership, engineering and customer follow-up. Used as an operating-process reference.
- Linear, April 10, 2026: theme analysis and AI-assisted handoffs. Some capabilities were internal experiments when described.
- Public discussion, September 20, 2026: a commenter describes separate feedback-pattern notes and the distorting effect of repeated requests. Anonymous and unverified.
- Productboard documentation, updated June 23, 2026: existing Jira linking and synchronization capabilities are a reason to evaluate available tools before custom work.

The company, stack combination, records, budget and rollout are authored assumptions. Sources do not establish demand for this service, endorse it, or support savings claims. Actual tool entitlements and APIs must be checked for each client.

<https://linear.app/now/cx-in-linear>

<https://linear.app/now/how-we-use-linear-agent-at-linear>

[https://www.reddit.com/r/micro\\_saas/comments/1wl3ck1/how\\_do\\_you\\_handle\\_user\\_feedback\\_once\\_it\\_starts/](https://www.reddit.com/r/micro_saas/comments/1wl3ck1/how_do_you_handle_user_feedback_once_it_starts/)

<https://support.productboard.com/hc/en-us/articles/11535151728275-Getting-started-with-Productboard-s-Jira-Integration>