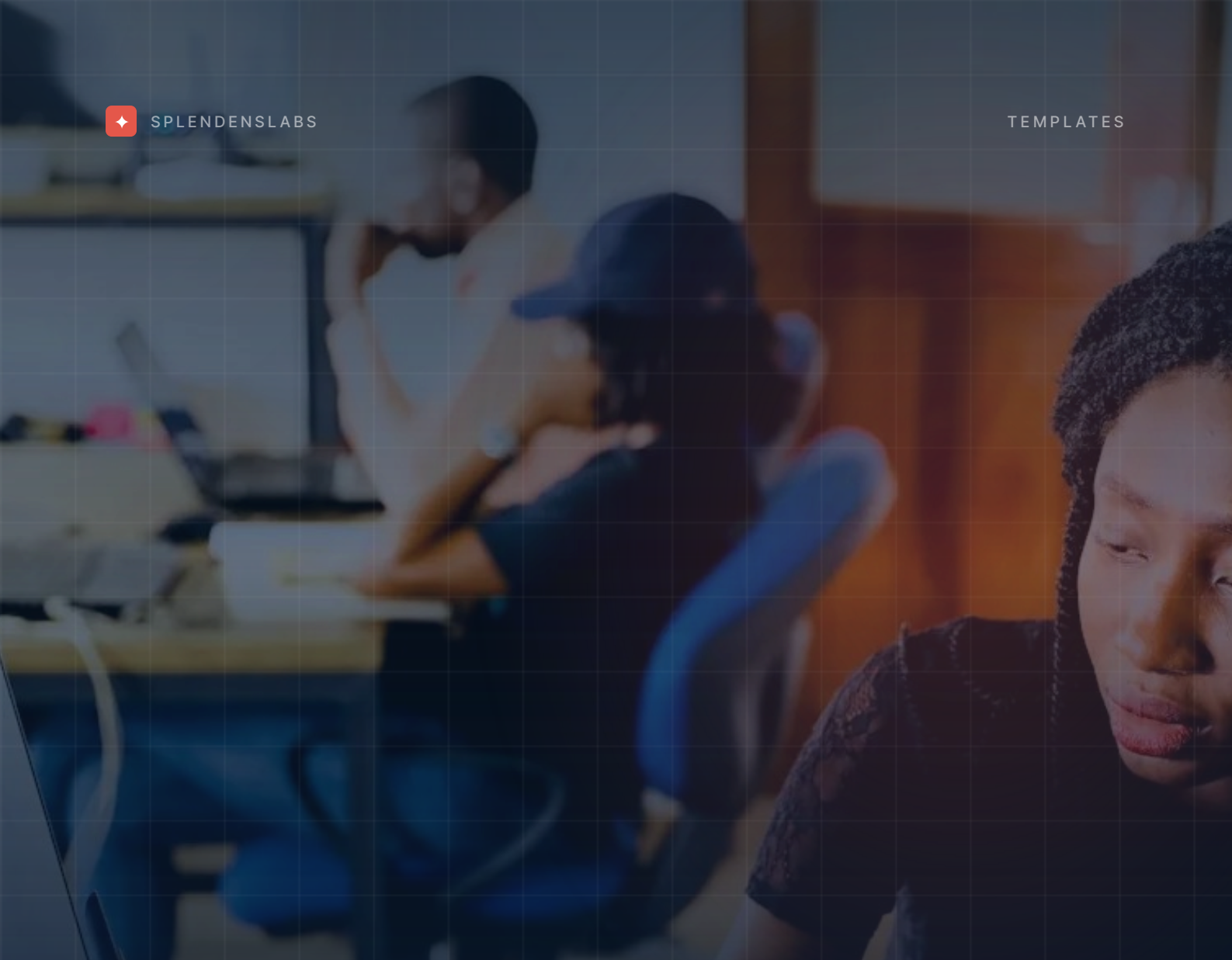


✦ TEMPLATE · TEMPLATES

Agent Design Document

A design document for AI agents that act on real systems: goal, tools, permissions, autonomy limits, failure handling and evaluation — written before the first tool call ships.

FORMAT	LEVEL	READ TIME	UPDATED
Template	Beginner	Fill-in	Oct 2026

OVERVIEW

What this template covers

An agent is software that decides what to do next. That makes its boundaries — which tools it holds, what it may do without asking, and how it stops — the most important part of the design. This template walks an engineering team through each decision and leaves a reviewable record for security, product and operations.

KEY TAKEAWAYS

- Define the agent's goal and its stopping condition before listing its tools.
- Give each tool the narrowest permission that still does the job.
- Classify every action by reversibility; irreversible actions need human approval.
- Design for loops, timeouts and partial failure from day one.
- Evaluate on recorded task traces, not on a handful of demo runs.

WHO IT'S FOR

- AI / ML engineers building agents
- Engineering and platform leads
- Security reviewers
- Product owners of automated workflows

HOW TO USE IT

The engineer leading the build drafts sections 01–05, then walks a security reviewer and an operations owner through it; sign-off in 06 gates production. Every box is filled in with a worked example — a refund agent for a fictional home-goods store — so you can see the level of detail expected. Replace the examples with your own.

CONTENTS

01 Goal and boundaries
02 Tools and permissions
03 Autonomy and human approval
04 Control loop and failure handling
05 Evaluation and observability
06 Design review and sign-off

01 / GOAL

Goal and boundaries

Write the goal as a task a capable new hire could be given. If it cannot be stated that clearly, the agent will not do it reliably either.

Agent name and goal

01

e.g. “Refund agent — resolve refund requests under [limit] by checking order status and policy, then issuing or escalating.”

WORKED EXAMPLE

Refund Assistant — resolve refund requests up to ₹5,000 for a fictional home-goods store by checking order status, delivery proof and the returns policy, then issuing the refund or escalating to a support lead.

Trigger and inputs

02

What starts a run (ticket, schedule, user message, event) and what it receives.

WORKED EXAMPLE

- New support ticket tagged “refund” in the helpdesk
- Inputs: ticket text, customer ID, order ID (if given), attached photos
- Re-runs when the customer replies on the same ticket

Done means...

03

The observable end state that ends a run, e.g. “Refund issued and customer notified, or case escalated with a summary.”

WORKED EXAMPLE

The refund is issued (or proposed for approval) and the customer is sent the confirmation template — **or** the ticket is reassigned to a support lead with a three-line summary and the evidence checked.

Explicitly not this agent's job

04

Adjacent tasks it must hand off rather than attempt.

WORKED EXAMPLE

- Exchanges, replacements and warranty claims → route to the Returns queue
- Payment failures and chargebacks → route to Finance
- Any change to the customer's account details

Do you need an agent at all?

A FIXED WORKFLOW IS ENOUGH WHEN	AN AGENT EARNS ITS PLACE WHEN
Steps are known and always the same order	The next step depends on what was found
Inputs are structured	Inputs are messy or unstructured
Errors can be handled by rules	Many tool combinations are possible
Auditors expect deterministic behavior	Judgment calls are bounded and reviewable

✦ 02 / TOOLS

Tools and permissions

List every tool the agent can call. Each one is an attack surface and a way to make a mistake at machine speed.

Tool inventory

TOOL	WHAT IT DOES	READ / WRITE	REVERSIBLE?	SCOPE LIMIT
get_order	Reads order, items, amounts and status	Read	—	Orders for the ticket's customer only
get_delivery_proof	Reads courier events and photos	Read	—	Same order only
search_policy	Searches the returns policy	Read	—	Current published policy

TOOL	WHAT IT DOES	READ / WRITE	REVERSIBLE?	SCOPE LIMIT
draft_reply	Writes a reply draft on the ticket	Write	Yes	Draft only, not sent
propose_refund	Creates a refund proposal for approval	Write	Yes, until approved	≤ ₹5,000, one per order
escalate_ticket	Reassigns with a summary	Write	Yes	Support-lead queue only

Identity the agent acts as

01

A dedicated service account, or on behalf of the signed-in user? Which credentials, and how are they scoped and rotated?

WORKED EXAMPLE

Dedicated service account `svc-refund-agent` with its own API keys per system, stored in the secrets manager and rotated every 90 days. It never borrows a staff member's session.

Data the agent can see

02

Systems, records and fields. Note personal or sensitive data and how it is minimized in prompts and logs.

WORKED EXAMPLE

- Orders: status, items, amounts, delivery events — no full card data
- Customer: name, email, order history; phone and address are masked
- Policy: the current returns policy document (read-only)
- PII is redacted from traces; raw ticket text is kept 30 days

✦ 03 / AUTONOMY

Autonomy and human approval

Decide action by action what the agent may do alone, what it may propose, and what it must never do.

Autonomy levels

LEVEL	MEANING	TYPICAL ACTIONS
Act	Agent performs it and logs it	Read records, draft text, classify, look up policy
Propose	Agent prepares it; a person approves	Send customer email, issue refund, update a record
Escalate	Agent hands off with a summary	Ambiguous policy, angry customer, low confidence
Never	Blocked in code, not just in the prompt	Delete data, change permissions, move money above a limit

Actions at each level

01

Map every write tool from section 02 to Act / Propose / Escalate / Never.

WORKED EXAMPLE

- **Act:** read order, read delivery proof, look up policy, draft reply
- **Propose:** refund ≤ ₹5,000, send customer email
- **Escalate:** refund > ₹5,000, damaged-item disputes, abusive messages
- **Never:** edit order totals, change payment method, close tickets without a reply

Approval experience

02

Where approvers see proposals, what context they get, and the time limit before a proposal expires.

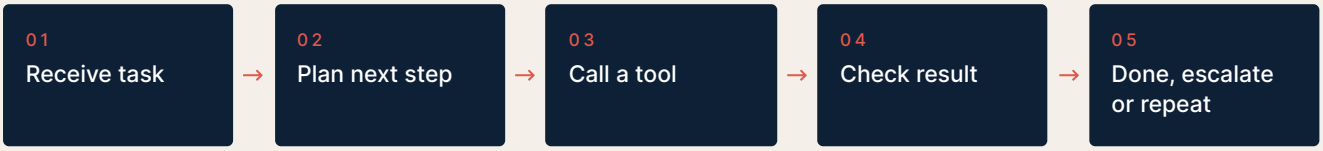
WORKED EXAMPLE

Proposals appear in the helpdesk side panel with the order, the policy clause relied on and the drafted reply. One-click approve or edit-then-approve. Proposals expire after 4 working hours and fall back to the normal queue.

✦ 04 / CONTROL

Control loop and failure handling

Agents fail in new ways: looping, wandering off task, or half-completing a multi-step change. Plan for each.



Limits

01

Maximum steps per run, maximum tool calls per tool, wall-clock timeout, token and cost budget per run.

WORKED EXAMPLE

- Max 12 steps and 3 calls per tool per run
- 90-second wall-clock timeout
- Token budget per run capped; alert if the daily spend exceeds the agreed budget

Partial failure and rollback

02

If step 3 of 5 fails, what state is the world in? Compensating actions, idempotency keys, retries.

WORKED EXAMPLE

Refund calls carry an idempotency key (ticket ID + order ID), so a retry can never pay twice. If the refund succeeds but the email fails, the agent retries the email twice, then flags the ticket “refund issued — notify customer” for a human.

Prompt-injection defenses

03

How untrusted content (emails, web pages, documents) is kept from issuing instructions; which tools are disabled when it is in context.

WORKED EXAMPLE

- Ticket text and attachments are passed as quoted data, never as instructions
- Write tools are disabled in any step that reads an attachment
- Refund amount is taken from the order system, never from the customer’s message

Model and prompt choices

04

Model(s) used per step, system prompt location and versioning, structured output formats.

WORKED EXAMPLE

A small, fast model classifies and extracts the order ID; a larger model plans and drafts replies. Prompts live in the repo under `prompts/refund/`, versioned with the code, and every tool call uses a JSON schema.

✦ 05 / QUALITY

Evaluation and observability

Judge the agent on whole tasks, including how it got there.

Task test set

01

Realistic scenarios with expected outcomes, including edge cases, ambiguous requests and injection attempts.

WORKED EXAMPLE

- 120 anonymised historical tickets with the decision a lead actually made
- 20 edge cases: partial deliveries, duplicate tickets, missing order IDs
- 15 injection attempts hidden in ticket text and image captions

Success and safety metrics

02

Task completion, correct escalation, policy violations, steps per task, cost per task, human override rate.

WORKED EXAMPLE

- Correct decision vs the lead’s decision on the test set
- Escalation precision — escalations a lead agrees were needed
- Policy violations: target zero; any one blocks release
- Steps and cost per resolved ticket; human override rate in production

Tracing and audit

03

Every run logs input, plan, each tool call with arguments and result, approvals and final outcome.

WORKED EXAMPLE

Every run writes one trace: ticket ID, plan, each tool call with arguments and result, the approver and the final outcome. Traces are searchable by ticket and order ID and retained for 12 months for audit.

Kill switch and rollout

04

Who can disable the agent, how fast, and the staged rollout (shadow mode → propose-only → limited autonomy).

WORKED EXAMPLE

- A feature flag owned by the support lead disables the agent within a minute
- Week 1–2: shadow mode, decisions logged but not shown
- Week 3–4: propose-only for all refunds
- After review: autonomous refunds up to a low limit, raised only with data

✦ 06 / SIGN-OFF

Design review and sign-off

Tick each item before the agent is connected to production systems.

SCOPE

- [ ] Goal, trigger and done condition are written down
- [ ] Out-of-scope tasks hand off cleanly

PERMISSIONS

- [ ] Each tool has the narrowest scope that works
- [ ] Irreversible actions require approval or are blocked in code
- [ ] Credentials are dedicated, scoped and rotated

SAFETY

- [ ] Step, time and cost limits are enforced

- [ ] Untrusted content cannot trigger write tools
- [ ] Kill switch tested

QUALITY

- [ ] Task test set passes agreed thresholds
- [ ] Full traces are stored and searchable

Decision and conditions 01

Approved for shadow mode / propose-only / limited autonomy — and any conditions.

WORKED EXAMPLE

Approved for **propose-only**. Conditions: human approval on every refund for the first four weeks; re-review before any autonomous limit is set.

Engineering lead — name, signature, date 02

WORKED EXAMPLE

A. Engineer · signed · 14 Oct 2026

Security reviewer — name, signature, date 03

WORKED EXAMPLE

S. Reviewer · signed with condition on attachment handling · 14 Oct 2026

Product / operations owner — name, signature, date 04

WORKED EXAMPLE

P. Owner, Support Operations · signed · 15 Oct 2026

GET IN TOUCH

Want us to run this with your team?

We design and build agents that act on real systems with tight permissions, human approval where it matters, and traces you can audit.

- Agent design reviews
- Tool and permission hardening
- Agent evaluation and rollout

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem