AI-Powered SOP Hub — Product Spec + Hiring Kit

A custom dashboard for capturing, standardizing, approving, and enforcing SOPs with AI assistance, granular permissions, and team incentives.

---

1) Vision & Non-Goals

Vision: A single web app where teams (PPC, SEO, Design, Sales, Ops, etc.) can access only the SOPs they need, run tasks while **following** those SOPs, and continuously improve them via AI-structured suggestions that route to Admin review and version control.

Non-Goals (v1): Enterprise SSO beyond Google/Microsoft; offline desktop app; multilingual authoring; legal/compliance certifications.

---

2) Roles & Permissions (RBAC)

* **Admin:** Manages users/teams, approves/merges SOP updates, sets incentives, configures prompts, views all analytics.
* **Dept Lead:** Owns SOPs in their department; reviews suggestions; publishes new versions.
* **Editor:** Can create/modify drafts within assigned dept; cannot publish.
* **Contributor (Default):** Can submit suggestions (text/image/video) and run SOPs.
* **Viewer:** Read-only access to assigned SOPs.

Access Control: Row-level policy: each SOP has `department` and `allowed_teams`. Users inherit `teams`. Permission check gates: view/run/suggest/edit/publish.

---

3) Core Flows

A) Capture → Standardize → Review → Publish

1. **Capture:** Teammate records a Loom/screen capture, uploads screenshots, or writes steps.
2. **Standardize (AI):** LLM converts raw input into the **Standard SOP Format** (Purpose, Scope, Roles, Prereqs, Tools, Step-by-Step, QA, Templates, Change Log).
3. **Review:** Dept Lead sees AI draft + diffs against current SOP; comments or requests changes.
4. **Publish:** Approved draft becomes new version; read receipts triggered to relevant users.

B) Suggestion While Working (In-Run Suggest)

1. User runs a SOP in **Run Mode** (checklist overlay).
2. Sees a better step → clicks **“Suggest improvement”** → attach note/screenshot/clip.
3. **AI** clusters, dedupes, and writes a structured **Change Proposal**.
4. Admin/Lead reviews and merges into SOP or archives.

C) Enforcement (Follow-SOP)

* “Start Run” requires opening the SOP overlay; checkboxes for each step; optional timers for steps; sub-evidence uploads.
* Completion requires all required steps checked (or justified exceptions logged).

---

4) Standard SOP Format (AI Target)

```
Title: <SOP Name> (SOP-<DEPT>-<slug>-vX.Y)
Purpose:
Scope:
Roles & RACI:
Prerequisites:
Tools & Access:
Definitions:
Procedure:
  1) Step …
  2) Step …
Quality Standard / Acceptance Criteria:
Common Errors & Fixes:
Templates & Links:
Change Log:
Next Review:
```

---

5) Data Model (simplified)

* **users**(id, name, email, role, teams\[], status)
* **teams**(id, name, department)
* **sops**(id, sop_id, title, department, status, version, content_md, content_json, allowed_teams\[], owner_id, approver_id, effective_date, next_review)
* **sop_versions**(id, sop_id, version, content_md, content_json, diff, published_by, published_at)
* **runs**(id, sop_id, user_id, started_at, completed_at, passed, exception_note)
* **run_steps**(id, run_id, step_no, checked_at, evidence_urls\[])
* **suggestions**(id, sop_id, user_id, raw_text, attachments\[], source_run_id, ai_summary, ai_changeset_json, status{queued,review,merged,rejected}, score)
* **reviews**(id, suggestion_id, reviewer_id, decision, notes, created_at)

* **read_receipts**(id, sop_id, user_id, version, acknowledged_at)
* **incentives**(id, user_id, suggestion_id, points, payout_amount, month)
* **audit_log**(id, actor_id, action, entity, entity_id, metadata, at)

---

6) Tech Stack

* **Frontend:** Next.js (App Router) + Tailwind + shadcn/ui; file uploads (tus/resumable); React-Query.
* **Backend:** Node (NestJS) or Python (FastAPI). Choose one; both fine.
* **DB:** PostgreSQL (+ Row Level Security using PostgREST policies or Prisma middle layer).
* **Cache/Queues:** Redis for job queues (BullMQ/RQ).
* **Storage:** S3-compatible (Wasabi/Cloudflare R2) for media.
* **Auth:** Google OAuth (NextAuth) + JWT, optional Auth0.
* **AI:** OpenAI (GPT-4.1/GPT-4o) for standardization, summarization, diff-aware edits; Whisper for transcription.
* **Search:** Postgres full-text or Meilisearch/Typesense for fast SOP lookup.
* **Browser Extension (optional v2):** Chrome/Edge extension to enforce Run Mode in supported tools.
* **Integrations:** Slack (webhooks), ClickUp (task deep links), Loom API, Google Drive.

---

7) Key Screens (UX Blueprint)

1. **Home / My SOPs:** Filtered by team; quick actions: Run, View, Suggest.
2. **SOP Viewer:** Tabs → Overview | Steps | QA Criteria | Templates | Change Log.
3. **Run Mode Overlay:** Step checkboxes, timers, evidence upload, “Suggest improvement” inline.
4. **Submit Suggestion:** Attachments, notes; preview of **AI-rewritten** proposal.
5. **Admin Review Queue:** Diff view (current vs proposal), accept/modify/reject; merge preview.
6. **Create SOP (AI Draft):** Paste raw notes or upload media → AI draft → editor.
7. **Analytics:** Read compliance, run completion, suggestion impact, most-edited SOPs.
8. **User & Team Admin:** Assign SOP access; bulk assign read receipts.

---

8) AI Pipelines

A) Standardizer

* Inputs: text, transcript (Whisper), screenshots (OCR), metadata (dept/role).
* Output: strict **Standard SOP Format**; step granularity; imperative verbs.
* Safety: remove secrets/PII; flag unknown tools for human fill.

B) Suggestion → Change Proposal

* Cluster similar suggestions; dedupe; generate **changeset JSON**:
`{"action":"append_step","step":{"index":8,"text":"…"}}` etc.
* Produce human-readable rationale + risk note.

C) Diff-Aware Merge

* Compute semantic diff vs current SOP; show conflicts; allow partial merge into version draft.

D) Readability & Consistency Guard

* Enforce style guide (voice, sentence length, list depth); reject if below threshold; ask contributor for missing pieces.

---

9) Prompt Templates (starter)

Standardizer Prompt (system)

```
You convert messy process notes (text/transcripts/screenshots OCR) into a rigorous,
step-by-step SOP using the format. Use imperative verbs, numbered steps, and include QA
criteria. Keep it unambiguous. If context is missing, insert TODO placeholders.
```

User content → Provide raw capture + metadata.
Assistant output → Fill the Standard SOP Format block.

Suggestion → Changeset Prompt

```
Given CURRENT_SOP and USER_SUGGESTION(S), propose precise changes as JSON
operations (append_step, modify_step, insert_after, deprecate_step, add_checklist_item).
Add a short rationale and risk note.
```

Diff-Aware Merge Prompt

```
Given CURRENT_SOP and CHANGESET_JSON, synthesize a clean draft SOP. Preserve
existing numbering; integrate new items; update QA criteria if impacted. Write a changelog
entry.
```

---

10) Enforcement Options

* **Soft Enforcement (v1):** Runs created from the Hub; completion required before ClickUp task can be closed (via webhook); else flag to manager.
* **Hard Enforcement (v2):** Chrome extension detects task context (e.g., Amazon Console, Helium 10) and ensures Run Mode active before allowing task completion.

---

11) Incentives — “SOP Innovator Program”

* **Points:** Accepted suggestion = 5–50 pts based on **Impact (1–5)** × **Adoption (1–5)** × **Quality (1–2)**.
* **Monthly Pool:** Admin sets budget (e.g., X PKR). Payout proportionally to points earned.
* **Badges:** “Fix-Finder,” “Speed Booster,” “Error Slayer,” “Client Win.”
* **Leaderboard:** Dept + company-wide; reset monthly but keep lifetime totals.
* **Auto-Attribution:** The system credits the run/suggestion submitter(s) whose change was merged.

Scoring Rubric (example):

* **Impact:** Saves time/cost, reduces errors, unlocks revenue. (1 minor → 5 major)
* **Adoption:** % of runs using the new step within 30 days. (1 <10% → 5 >75%)
* **Quality:** Completeness, clarity, evidence. (1 acceptable → 2 excellent)

---

12) Metrics & Dashboards

* SOP coverage by dept; average time since last review.
* Read compliance by role; overdue acknowledgements.
* Run completion rate; exception frequency.
* Suggestion volume, acceptance rate, time-to-merge.
* Time saved (self-reported + benchmarked); error rate trend.

---

13) Security & Compliance

* OAuth login; MFA optional; RBAC on every query.
* **Row-Level Access:** `allowed_teams` gate; soft delete + audit trail for all changes.
* **Secrets Hygiene:** Redact API keys/credentials from uploads via pattern filters.
* **PII:** Avoid storing unless necessary; encrypted at rest; signed URLs for media.

---

14) Implementation Phases (scope only)

* **MVP:** Auth/RBAC, SOP catalog, Create (AI draft), View, Run Mode, Suggestion submit + AI rewrite, Admin review/merge, Versioning, Read receipts, Slack notifications.
* **v1.1:** Analytics, Leaderboard & incentives, Review queue filters, Bulk access assignment.
* **v2:** Chrome extension enforcement, Whisper transcription, clustering/dedup, Meilisearch, ClickUp task gating.

---

15) Acceptance Criteria (MVP)

* A non-Admin user sees only their dept SOPs and can **Run** them.
* Submitting raw notes + screenshots returns a clean SOP draft in \<N steps (configurable).*
* Admin merges an accepted **Change Proposal** into a new version with a generated changelog.
* Publishing triggers read receipts to assigned users and logs acknowledgements.
* ClickUp webhook blocks task closure if run wasn't completed (soft enforcement).

*Performance targets configurable in settings; measure and log latency.

---

16) API Endpoints (sketch)

* `POST /auth/login`
* `GET /me`
* `GET /sops?team=...`
* `GET /sops/:id`
* `POST /sops` (create draft via AI)
* `POST /sops/:id/publish`
* `POST /runs` → start; `PATCH /runs/:id` → check step; `POST /runs/:id/complete`
* `POST /suggestions` (with files)
* `GET /suggestions?status=review`
* `POST /suggestions/:id/ai-proposal`
* `POST /suggestions/:id/merge`
* `POST /read-receipts`
* `GET /analytics/…`

---

17) Sample JSON Schemas

SOP (content_json excerpt)

```json
{
  "title": "Keyword Research — PPC",
```

```
  "purpose": "Identify converting search terms",
  "procedure": [
    {"n":1, "text":"Open Helium10 → Cerebro", "required":true},
    {"n":2, "text":"Export top 200 terms by CPR", "required":true}
  ],
  "qa": ["Min 30 relevant keywords", "No duplicates", "CSV saved to client folder"],
  "templates": [
    {"label":"Keyword Sheet", "url":"https://…"}
  ]
}
```

Change Proposal (ai_changeset_json)

```json
{
  "ops": [
    {"action":"insert_after","index":2,"text":"Use Amazon Brand Analytics to pull Search Query Performance."},
    {"action":"append_checklist","section":"qa","item":"Add ABA SQR terms with share > 5%"}
  ],
  "rationale":"ABA SQR now available; improves accuracy",
  "risk":"Low"
}
```

---

18) Hiring — Job Description (copy-ready)

Title: AI Engineer (SOP Automation & Workflow)

You will build: SOP Hub — AI-assisted SOP capture, standardization, approvals, and enforcement with RBAC.

Responsibilities

* Design backend (Postgres/Redis) and APIs; implement RBAC.
* Build AI pipelines for SOP standardization, suggestion clustering, and diff-aware merges.
* Integrate uploads (video/audio/screenshots) and Whisper/OCR.
* Ship MVP frontend (Next.js) with Run Mode and Admin Review.
* Add analytics, notifications (Slack), and ClickUp webhook gating.

Must-Have Skills

* Production experience with OpenAI or similar LLMs; prompt engineering; function-calling.
* Node/NestJS or Python/FastAPI; Postgres; Redis queues; S3.
* React/Next.js + file uploads; auth (OAuth/OIDC).

* Designing diff/merge workflows and versioned content systems.

Nice-to-Have

* Chrome extension dev; Meilisearch/Typesense; Loom/Drive APIs; ClickUp APIs.

How we work

* Ship fast, instrument everything, write crisp PRDs, automated tests, weekly demos.

---

19) Interview Loop & Take-Home

Screen (30–45m): Systems design for a versioned SOP editor with RBAC.
Tech Deep-Dive: LLM pipelines, retries, cost control, evals.

Take-Home (cap 6–8h):

* Build a **mini SOP Standardizer**:

  * Input: markdown/text + 1 image.
  * Output: our Standard SOP Format; show JSON changeset from a sample suggestion.
  * Simple RBAC (admin/user); Postgres or SQLite ok.
  * Bonus: add a diff viewer and a single webhook (Slack) on “publish.”

Rubric: Clean architecture, tests, prompt quality, error handling, security (upload validation), clarity of README, live demo.

---

20) Incentive Policy (ready-to-announce)

* **Monthly pool:** Set by leadership; transparent.
* **Points formula:** `Impact (1–5) × Adoption (1–5) × Quality (1–2)`.
* **Eligibility:** Only merged suggestions with evidence from runs.
* **Payout:** Pro-rata share of pool; top 3 get 10% bonus points.
* **Transparency:** Leaderboard and monthly “SOP Innovator” shout-out.

---

21) Rollout Plan (scope only)

* Stand up MVP with PPC + Ops departments.
* Migrate 10 core SOPs; enable Run Mode.
* Pilot incentives; gather feedback; iterate prompts.
* Expand to Design/SEO/Sales; enable ClickUp gating.

---

22) Risks & Mitigations

* **Prompt drift / hallucination:** Use strict JSON schemas, guardrails, human review.
* **Over-enforcement friction:** Start soft; allow exception notes; measure.
* **Low participation:** Incentives + visible leaderboard + manager adoption.
* **Security:** RLS, signed URLs, virus scan on uploads, PII redaction.

---

23) Ready-Made Assets

* Standardizer Prompt, Changeset Prompt, Merge Prompt (above).
* Sample JSON schemas for SOP and Change Proposal.
* JD + Interview kit ready to post.

Outcome: A living SOP system the team actually enjoys using — because it makes work faster, cleaner, and rewards improvements automatically.