Collab Admin Project Map (Meticulous v4)

Control plane repo + rubrics + dashboards + Workflows ecosystem wiring

Prepared for: Tim
Date: December 31, 2025
Scope: Collab Admin repo design plus explicit GitHub automation details from the Workflows ecosystem.

Source inputs used for v4: prior v3 project map + current state of repositories in the stranske GitHub ecosystem (Workflows, Workflows-Integration-Tests, Template, and Travel-Plan-Permission).

Table of contents

1. What Collab Admin is

Collab Admin is a private control plane repository. It makes the collaboration measurable without turning it into a points game.

- Defines deliverables for the four workstreams.
- Stores policies and protocols that constrain how work is submitted and reviewed.
- Defines descriptor-first rubrics and writes review records (ratings + highlighted descriptors + feedback).
- Logs time/expenses/friction and generates weekly/month-end reports.
- Dogfoods the same automation ecosystem used by your other repos by starting from the Template repo.

Design anchor: Collab Admin should be created by cloning **stranske/Template** and then layering Collab Admin-specific docs/rubrics/templates on top. Template already contains the consumer-side wiring for the shared automation library (**stranske/Workflows**).

2. Repo blueprint

The recommended structure is the v3 file tree, with one important practical update: treat Workflows integration as a first-class dependency (not an afterthought).

High-level tree:

```
collab-admin/
README.md
config/ (project.yml, dashboard configs)
docs/ (policies, protocols, and this roadmap)
rubrics/ (descriptor-first YAML rubrics; no numeric storage)
templates/ (submission packets, memo templates, plan templates)
reviews/ (Tim's review YAML records)
logs/ (time, expenses, friction, weekly/month-end reports)
dashboards/ (public static markdown; no numeric scoring)
streamlit_app/ (Tim-private dashboard, may compute numeric trends locally)
scripts/ (validators, summarizers, dashboard builders)
.github/ (issue forms, PR template, minimal CI for this repo's scripts/logs)
```

Workflows ecosystem files: Because you plan to clone Template, Collab Admin will inherit consumer workflows like `.github/workflows/pr-00-gate.yml`, `ci.yml`, keepalive/agent workflows, and autofix. Those should be kept thin and kept in sync via the Workflows sync pipeline.

3. Rubrics and evidence standards

Rubrics are descriptor-first with four levels: Poor, Mediocre, High quality, Excellent. Numeric mapping exists only in Tim's local dashboard config and is never written back to the repo.

3.1 Trend evidence standard (no AI)

- Trend understanding write-ups must be created by direct reading/tracing (no AI assistance).
- Every memo includes a References section with file paths and line ranges (path/to/file.py#Lx-Ly) and a short rationale.
- Claims that something is used must include at least one call-site reference.
- Minimum evidence mix per subsystem brief: entrypoints, core path, error/edge handling, data boundaries/config, and change hotspots.

3.2 Writing quality rubric (applies to every memo)

- Rigor of analysis: makes falsifiable claims, states assumptions, considers edge cases, ties claims to evidence.
- Precision and correctness: avoids hand-waving and conflating hypothesis with known behavior.
- Structure and conciseness: strong logical flow, minimal redundancy; bloat is penalized.

4. Dashboards and review records

Two dashboards prevent point-gaming while still giving you analytics.

- Tim-private dashboard (Streamlit, local): can compute numeric trends from rubric levels; never writes numeric scores into repo files.
- Shared dashboard (static markdown committed in-repo): shows qualitative ratings and distributions only.

Inputs the dashboards read:

- GitHub metadata: Issues/PRs/labels/CI status.
- Logs: time, expenses, friction.
- Review YAML records (rubric selections + highlighted descriptors + feedback).
- Deliverable index files and submission packets.

5. Automations inside Collab Admin

Automation in Collab Admin should be conservative: validate hygiene and generate drafts, but do not merge or push directly to main.

Automation	Where	Purpose	Notes
Time log validator	scripts/validate_time_log.py + CI	Enforce <= 40 hours/week, schema correctness	Runs on PR; flags violations
Submission packet validator	scripts/validate_submission_packet.py (planned)	Ensure required links/fields exist	Keep it fast; no deep parsing
Reference-format validator	scripts/validate_references.py (planned)	Check that Trend memos contain path#Lx-Ly patterns	Format only; review enforces content
Dashboard builder	.github/workflows/build_dashboard.yml	Regenerate dashboards/public/dashboard.md	Opens PR; no direct pushes

6. Workstreams 1-4 deliverables

This section is unchanged in structure from v3, but now references the real automation surface area in the GitHub repos.

6.1 Workstream 1 - Trend_Model_Project (core)

- Deliverables: system map; 8-14 subsystem briefs; risk register; 2+ characterization tests; 1-2 safe PRs improving clarity/safety without changing intent.
- Unit of work: Subsystem epic -> module notes -> risk entries -> (optional) tests -> safe PR.
- No AI assistance for understanding deliverables; enforced via policy + evidence standard + walkthrough.

6.2 Workstream 2 - Agent integration (Workflows ecosystem)

- Integrate Claude Code as next agent; then add a third agent.
- Define and implement a stable agent output contract so keepalive/verifier do not require per-agent hacks.
- Use Workflows-Integration-Tests as the compatibility harness when possible; changes that affect consumers must land in Workflows first and then be propagated.

6.3 Workstream 3 - Consumer usability validation

- Build a medium-sized project that integrates Workflows via thin callers.
- Maintain a friction log with minutes lost and PRs that reduce friction.
- Fix cross-repo problems at the source (Workflows) rather than locally in the consumer, when appropriate.

6.4 Workstream 4 - Marketplace implementation plan

- Pick two workflows and draft a step-by-step plan for packaging them for broader use.
- Include a rubric for time, reliability, security controls, maintainability, and cost.
- Execution optional unless approved.

7. Meta-workstream roadmap (Project Instrumentation)

Phase	Outcome	Key deliverables
P0 (Week 1)	Repo boots cleanly	Docs skeleton + templates + issue forms + labels/project views
P1 (Week 1-2)	Rubrics v1 usable	Rubric YAML packs + review YAML format + submission packet
P2 (Week 2-3)	Validation gates	Time cap validator + reference validator + rubric schema validator
P3 (Week 3-4)	Streamlit MVP	Overview + Review Console + Time/Expenses page
P4 (Week 4-5)	Static dashboard via PR	Scheduled dashboard build opens PR; no direct pushes
P5 (Week 6+)	Tighten + extend	Auto-open revision issues; richer workstream reporting

Appendix A. Workflows ecosystem wiring (explicit repo detail)

This appendix is the concrete mapping: what exists in the repos today and how the pieces interact.

A.1 Core repos and roles

Repo	Role in ecosystem
stranske/Workflows	Source of truth for reusable CI workflows, keepalive/agent automation, sync pipelines, and guard tests.
stranske/Workflows-Integration-Tests	External consumer harness that runs multiple configurations of reusable CI and notifies Workflows on config change.
stranske/Template	Consumer template repo; starting point for new repos. Intended to stay in sync with Workflows templates.
stranske/Travel-Plan-Permission	Primary consumer reference implementation; used to validate cross-repo behavior and keepalive ergonomics.

A.2 Consumer sync (Workflows -> consumers)

- Workflows runs a manifest-driven sync: maint-68-sync-consumer-repos.yml reads .github/sync-manifest.yml and creates PRs in registered consumer repos.
- The manifest lists thin-caller workflows, prompts, scripts, codex config, and selected docs. Consumer-owned files can be marked create_only so local tuning survives sync.
- Registered consumer repos are listed inside maint-68 under REGISTERED_CONSUMER_REPOS.

A.3 Integration repo sync + validation

- Workflows templates/integration-repo provides a minimal external consumer project plus a multi-job CI workflow that exercises reusable-10-ci-python.yml under different configurations.
- Workflows maint-69-sync-integration-repo.yml pushes template updates into Workflows-Integration-Tests and replaces the __WORKFLOW_REF__ placeholder with the selected workflow reference.
- Workflows health-67-integration-sync-check.yml compares integration repo files to templates and creates issues when drift is detected.
- Workflows-Integration-Tests notify-workflows.yml sends repository_dispatch to Workflows when its config changes, closing the loop.

A.4 Keepalive chain (consumer + Workflows)

- Consumer pr-00-gate.yml must publish the commit status context Gate / gate.
- Consumer agents-pr-meta.yml listens for issue_comment and workflow_run of Gate; it base64-encodes the comment body and calls Workflows reusable-20-pr-meta.yml.
- Workflows reusable-20-pr-meta.yml uses a dual checkout: consumer repo for context, Workflows repo for scripts.

- Workflows agents-70-orchestrator.yml (and consumer orchestrator wrappers) coordinate keepalive sweeps and belt processing.

A.5 Required secrets and variables (consumer repos)

Name	Type	Purpose (high level)
SERVICE_BOT_PAT	Secret	Bot pushes branches and posts comments (classic PAT; bot account)
ACTIONS_BOT_PAT	Secret	Workflow dispatch + cross-repo triggers
OWNER_PR_PAT	Secret	Elevated PR/branch operations when needed
CODEX_AUTH_JSON	Secret	Codex CLI auth for workflows that run Codex
WORKFLOWS_APP_ID / WORKFLOWS_APP_PRIVATE_KEY	Secret (optional)	Preferred auth: mint a GitHub App token; fall back to PATs when absent
ALLOWED_KEEPALIVE_LOGINS	Repo variable	Comma-separated allowlist for who can trigger keepalive

A.6 Known sharp edges (already documented in repos)

- Artifact conflicts: multi-job CI calling reusable-10-ci-python.yml requires unique artifact-prefix per job.
- GitHub startup_failure has been observed when mixing reusable-workflow jobs (uses:) and normal jobs (runs-on:) inside one workflow file. Keep them in separate workflow files.
- Some consumer thin-caller workflows warn against adding top-level permissions blocks because they can contribute to startup_failure. Follow the template comments unless you have a verified reason to deviate.

End of document.