Foundational Work

Purpose

I. Core / Surface-Level Purposes

(these are easy to agree on and non-threatening)

These are usually what people *say* first — and that's okay.

1. **Keep track of work**
    - Know what needs to be done, by whom, and by when
    - Avoid dropped balls
2. **Improve coordination**
    - Reduce back-and-forth messages
    - Make handoffs clearer between people or units
3. **Reduce confusion**
    - One source of truth for tasks
    - Less "Where are we on this?" moments

👉 These are *necessary but not differentiating*.

II. Efficiency & Cognitive Load Reduction

(this is where your earlier intuition fits perfectly)

4. **Free up mental space**
    - People don't have to remember everything
    - Lower anxiety about forgetting obligations
5. **Reduce rework**
    - Clear definitions of "done"

- Fewer revisions caused by missing context

6. **Make work feel lighter**
    - Smaller, well-defined tasks
    - Clear progress signals

Framing this as *reducing cognitive load* is often more human than “efficiency.”

III. Compliance & Reporting (VERY important for you)

These purposes are *quietly powerful* because they justify rigor.

7. **Automatic compliance tracking**
    - Every task leaves a timestamped trail
    - Activities are already “logged” as work happens
8. **Auto-generation of reports**
    - Monthly activity reports
    - Quarterly narrative reports
    - End-of-project reports
9. **Audit readiness**
    - Ability to answer: *What was done? When? By whom? Why?*
    - Without scrambling or reconstructing history

This is a **strong anchor purpose** because:

- It’s concrete
- It saves real time

- It justifies structure without sounding bureaucratic

IV. Quality, Rigor, and Defensibility

(this matters a lot in AIPO-like work)

10. **Improve quality of outputs**

- Tasks require evidence, not just completion
- Reviews are explicit, not implicit

11. **Make decisions defensible**

- Why a decision was made is recorded
- Alternatives and rationale are preserved

12. **Reduce reliance on memory and personalities**

- Less “I remember we decided…”
- More institutional clarity

This is where **ontology really pays off**.

V. Institutional Memory & Continuity

(often underestimated, but huge)

13. **Preserve institutional knowledge**

- Past analyses, decisions, and workflows don’t disappear
- New staff can see how work is *actually* done

14. **Enable smoother onboarding**

- New people learn by seeing real tasks, not just SOPs
- Less shadowing, less guesswork

15. **Protect against staff turnover**

- Work doesn't collapse when key people leave

This purpose resonates strongly with leadership.

VI. Learning & Improvement (meta-level value)

16. **Enable reflection and learning**

- What tasks repeatedly get stuck?
- Where do delays or confusion occur?

17. **Identify systemic bottlenecks**

- Not individual blame, but process insight
- Data-driven improvement

18. **Support better future planning**

- Real timelines instead of optimistic guesses
- Evidence-based workload planning

This is where **thematic analysis of tasks themselves** becomes possible.

VII. Human–AI Collaboration (future-facing, optional)

19. **Prepare for AI assistance**

- AI can summarize, cluster, retrieve — safely

- Humans remain accountable

20. **Prevent AI misuse**

- Clear boundaries on what AI can and cannot do
- Traceability of AI-generated content

21. **Turn tasks into structured prompts**

- Each task becomes a unit AI can help with
- Without hallucinating or overstepping

This purpose doesn't have to be foregrounded — but it *shapes design*.

VIII. Strategic Framing (if you had to say it in one breath)

Here are **example purpose statements**, different tones:

Conservative / Safe

> "To streamline task execution while ensuring accurate, timely reporting and compliance."

Human-centered

> "To reduce cognitive load and make work clearer, lighter, and more predictable."

Governance-focused

> "To ensure tasks, decisions, and outputs are traceable, defensible, and auditable."

Strategic (my favorite for you)

> "To make everyday work automatically produce compliance-ready records, institutional memory, and actionable insight."

A gentle but important suggestion

For your project, I'd recommend:

- **Primary purpose (anchor):**

  👉 *Auto-generation of compliance and narrative reports*

- **Secondary purposes:**

  👉 Reduce cognitive load

  👉 Improve quality and defensibility of work

- **Implicit purpose (don't advertise yet):**

  👉 Build institutional memory + AI readiness

This keeps everyone comfortable **while still enabling the deeper architecture you're designing**.

Core Module and Ontology

MODULE 1 — Task Module (the atomic unit)

Why this is non-negotiable

Reports are summaries of **work done**.

Tasks are the **smallest reportable unit**.

If this module is designed well, reports become filters + narratives, not writing marathons.

Minimum ontology (report-relevant only)

Identity & Time

- Task ID
- Task Title
- Task Type (Analysis / Execution / Review / Coordination / Decision)
- Start Date
- End Date / Completion Date
- Reporting Period (Month, Quarter) *(can be derived)*

Responsibility

- Assignee
- Accountable Owner

Work Description

- Objective (1–2 sentences)
- Activity Description (what was done)

Outputs

- Deliverable / Output
- Evidence / Link

Status

- Planned / In Progress / Completed / Deferred

📌 **Key design insight:**

Narrative reports are **not written later** — they are *assembled* from these fields.

MODULE 2 — Activity Log / Event Module (lightweight but powerful)

Why this matters for compliance

Compliance reports care about:

- **what happened**
- **when it happened**
- **by whom**

This module captures **timestamped events**, not just task states.

Minimum ontology

- Event ID
- Related Task (link)
- Event Type (Started / Updated / Reviewed / Completed / Submitted)
- Timestamp
- Actor

- Short Note (optional)

📌 This enables:

- automatic timelines
- defensible audit trails
- accurate monthly aggregation

💡 You can implement this implicitly at first (auto-generated events).

MODULE 3 — Reporting Module (your anchor module)

Why this must exist early

If reporting is the *primary purpose*, it deserves its own module — not an afterthought.

Minimum ontology

Report Identity

- Report ID
- Report Type (Monthly Activity / Quarterly Narrative / Terminal)
- Reporting Period (Date range)
- Project / Program

Inputs

- Linked Tasks (filtered by period)
- Linked Events (derived)

Outputs

- Auto-generated Activity Table
- Auto-generated Narrative Draft
- Manual Narrative Notes (human layer)

Status

- Draft / Reviewed / Submitted

📌 **Key insight:**

Reports are *views over tasks*, plus a small human interpretation layer.

MODULE 4 — Project / Program Module (scope container)

Why you need this

Compliance is almost always **project- or grant-based**.

This module defines **what tasks roll up into which reports**.

Minimum ontology

- Project ID
- Project Name
- Funding Source
- Reporting Requirements
- Reporting Frequency
- Start / End Dates

📌 Tasks belong to a project → projects generate reports.

MODULE 5 — Evidence / Artifact Module (optional but high-leverage)

Why this helps reporting quality

Reports often require:

- attachments
- links
- proofs

Separating evidence from tasks keeps things clean and reusable.

Minimum ontology

- Artifact ID
- Type (Document, Presentation, Filing, Dataset)
- Linked Task(s)
- Date Created
- Storage Link

📌 This avoids hunting for files at report time.

What to EXPLICITLY defer (important)

To avoid overwhelm, **do not build these yet**:

❌ Full OKR module

❌ Knowledge graph

❌ Advanced dependency modeling

❌ AI agents

❌ Performance dashboards

These are *downstream beneficiaries* of clean task + reporting data.

Designing for Flexibility

How to Ensure Flexibility Without Future Pain

The Core Idea (hold onto this)

> **Freeze meaning early.**
> **Delay structure as long as possible.**

Meaning = ontology, purpose, intent

Structure = schema, database constraints, services

---

1. Freeze the

Ontology

, Not the

Schema

What to freeze now

- What a **task means**
- What “done” means
- What evidence means
- What accountability means

This is your **Task Ontology Spec v0.1** (which you already have).

What NOT to freeze now

- Exact column names
- Database normalization
- Field data types beyond basics
- How reports are rendered

💡 *Ontology survives migrations. Schemas don't.*

---

2. Design for Additive Expansion Only

A flexible system allows **adding**, not **changing**.

Rule of thumb

> You should almost never need to *remove* or *reinterpret* an existing field.

Examples:

- Start with Task Type
- Later add Sub-Type, not replace it
- Start with Evidence (text/link)
- Later add structured artifacts

This avoids painful refactors.

---

3. Separate "Core" from "Extensions" Explicitly

Even in a spreadsheet.

Core fields (never change semantics)

- Task ID

- Objective
- Assignee
- Dates
- Status
- Deliverable
- Evidence

Extension fields (allowed to grow)

- Risk level
- Confidence
- AI usage
- Dependencies
- Tags

Make this separation **explicit in your document**.

---

4. Use IDs Everywhere (even in spreadsheets)

This sounds boring — it's not.

Why IDs matter

- Easy migration to databases
- Stable references across systems
- Safer joins later

Every entity gets an ID:

- Task ID
- Project ID
- Report ID

IDs = portability.

---

5. Treat Reports as Views, Not Objects (at first)

To stay flexible:

- Reports **pull from tasks**
- Reports do NOT own data
- Narrative is layered on top

Later, if needed, reports can become first-class entities — but not now.

---

6. Log Events Instead of Over-Structuring State

State is brittle. Events are flexible.

Instead of:

- complex status trees

Prefer:

- simple status + event log

This lets you:

- reconstruct timelines

- change interpretations later
- add analytics without refactoring

---

7. Delay Hard Relationships

Some relationships are obvious (Task → Project).

Others are tempting but dangerous early.

Delay these:

- full dependency graphs
- many-to-many role hierarchies
- advanced knowledge graphs

Keep relationships “soft” first:

- tags
- references
- links

You can harden later.

---

8. Make Expansion a First-Class Use Case

In your **design doc**, explicitly add:

> “Expected future expansions”

Examples:

- AI-assisted summarization
- Knowledge reuse
- Cross-project analytics
- SOP enforcement

This changes how people design today.

---

9. Pick Tech That Supports Change, Not Perfection

You already hinted at this well.

Spreadsheet → Firebase → Postgres path works because:

- tabular thinking stays consistent
- IDs migrate cleanly
- schemas can evolve

Avoid:

- over-normalized schemas
- rigid ORMs too early
- “perfect” data models

---

10. Use a “Flexibility Check” Before Adding Anything

Before adding a field or module, ask:

1. Does this help reporting now?

2. Can it be ignored without breaking anything?
3. Can it be derived later?
4. Can it be added without touching old data?

If “yes” to most — it’s safe.

A Simple Mental Model for the Team

You can explain flexibility to the group like this:

> “We’re not building a finished system.
> We’re building a **language for work** that can live in many tools.”

That framing reduces fear and resistance.

One concrete suggestion (do this)

Add a short section to your core document:

“Rules for Safe Expansion”

- New fields must be optional
- No field may change meaning
- Old data must remain valid
- Reports must not break

This becomes your **guardrail**.

Rules for Safe Expansion

Rules for Safe Expansion

This system is intentionally designed to evolve over time. To ensure that future enhancements do not create disruption, rework, or loss of institutional knowledge, the following rules shall govern all expansions, modifications, and extensions of the system.

1. Meaning Is Frozen; Structure Is Flexible

The meaning of core concepts (e.g., *task*, *completion*, *evidence*, *accountability*) shall not be altered once defined.

However, the way these concepts are represented in tools, schemas, or databases may evolve.

No expansion shall reinterpret an existing field or concept in a way that invalidates past data.

---

2. Additive Changes Only

All expansions must be **additive**, not substitutive.

- New fields may be added.
- New modules may be introduced.
- Existing fields shall not be removed.
- Existing fields shall not change their original meaning.

If a new requirement overlaps with an existing field, a new field shall be created rather than repurposing the old one.

---

3. Core Fields Are Stable and Backward-Compatible

The following core task attributes must always remain valid for historical records:

- Task identity
- Objective

- Responsibility (assignee and accountable owner)
- Dates (start and completion)
- Status
- Deliverable and evidence

All future designs must ensure that tasks created under earlier versions of the system remain interpretable and reportable.

---

4. Optional First, Mandatory Later

New fields and modules shall be introduced as **optional by default**.

A field may only become mandatory if:

- its purpose is well understood,
- its value is proven in practice,
- and its absence does not invalidate historical records.

This prevents early over-structuring and reduces resistance from users.

---

5. Reports Must Not Break

All expansions must preserve the ability to generate existing reports.

- Historical reports must remain reproducible.
- New data may enrich reports, but not alter past outputs.
- Report definitions should reference meaning, not implementation details.

If a change risks breaking reporting, it must be deferred or redesigned.

---

6. Stable Identifiers Are Non-Negotiable

All core entities (e.g., tasks, projects, reports) shall use stable identifiers.

Identifiers must:

- remain constant across tools and migrations,
- allow linking between modules,
- support future database transitions.

Identifiers are the primary mechanism for long-term portability.

---

7. Events Are Preferred Over Complex State

Where possible, changes in work should be recorded as **events** rather than encoded into increasingly complex task states.

This enables:

- timeline reconstruction,
- auditability,
- future reinterpretation without data loss.

State may evolve; events preserve history.

---

8. Soft Relationships Before Hard Dependencies

Early versions of the system shall favor:

- tags,
- references,
- links,

- notes

over rigid dependency graphs or enforced hierarchies.

Relationships may be formalized only after their necessity and stability are demonstrated in practice.

---

9. Expansion Must Be Justified by Use

No module, field, or automation shall be added solely because it is *possible*.

Every expansion must be justified by at least one of the following:

- reporting needs,
- compliance requirements,
- reduction of cognitive load,
- demonstrated user pain points.

This keeps the system purposeful and humane.

---

10. The System Must Remain Understandable

At every stage of expansion, the system must remain explainable to a non-technical user.

If an expansion increases complexity, it must also provide:

- clearer documentation, or
- tangible reduction in manual effort.

A system that cannot be understood cannot be trusted.

---

11. Expansion Is a Deliberate Act

All significant expansions should be:

- discussed explicitly,
- documented clearly,
- and reviewed for alignment with the system's primary purpose.

The goal is not to grow fast, but to grow *well*.

Closing Principle

This system is not designed to be finished.

It is designed to remain **coherent**, **defensible**, and **useful** as it grows.

Expansion is a feature — chaos is not.

Stress Testing

Stress Test 1 — Real AIPO Workflow Chaos

Scenario

AIPO runs multiple simultaneous streams:

- licensing negotiation
- IP audit training
- survey thematic analysis
- partner coordination
- internal policy work

…and work is often interrupted.

What breaks first

Task granularity + task type consistency.

People will create tasks that are:

- too big (“Finish IP audit for Partner X”)
- too vague (“Work on report”)
- too cross-cutting (“Coordinate with DOST” with no scope)

Does the current map handle it?

✅ Mostly yes, because:

- Task module has objective/activity/output
- Projects contain reporting scope

- Events capture history

⚠️ But you still need one more **MVP-safe guardrail**:

Add one field to Task (optional but HIGH impact):

- **Workstream / Work Package** *(small controlled list; e.g., Licensing, TT Capacity Building, Compliance, Research Support, Operations)*

Why: it prevents "everything is Coordination" and makes quarterly narrative assembly coherent.

> This is not scope creep. It's a reporting alignment tag.

Outcome: Without this, reporting narratives will be harder to shape.

Stress Test 2 — Cross-Project & Institutional Work

Scenario

One activity supports 2–3 grants or is institutional (e.g., "AIPO strategy workshop," "IP policy revision").

What breaks first

Project association.

You already added:

- Primary Project (required)
- Secondary Project(s) (optional)

✅ This is the correct design.

⚠️ But you must specify a policy for reporting collisions:

Add a rule in Reporting module (policy, not code):

- If task has multiple projects, it can appear in multiple reports **only if flagged**.

To implement minimally, add **one field**:

- **Multi-report Allowed? (Yes/No)** *(default No)*

Why: prevents accidental double-counting.

Stress Test 3 — Audit & Defensibility (“Show me proof”)

Scenario

An auditor asks:

- “When exactly was this completed?”
- “Who reviewed it?”
- “Where is the evidence?”
- “Why was this decision made?”

What breaks first

Evidence linkage discipline.

Your design supports:

- Events with actor/timestamp
- Evidence artifacts linked to tasks
- Decision records with evidence

✅ Strong.

⚠️ Two subtle audit gaps remain:

Gap A: Evidence without access stability

A “Storage Link” might point to:

- personal drive
- expiring link
- moved file

MVP-safe fix: add to Artifact:

- **Storage Location Type (Drive/Notion/Email/Local/Other)**

This helps future migrations and audit retrieval.

Gap B: Decision provenance

Decisions need to connect to *what triggered them*.

Fix: ensure Decision module includes:

- **Decision Trigger / Context (Meeting / Task / Issue)** *(can just be a relation or text)*

You already link decisions to tasks/meetings; just make it explicit in the template.

Stress Test 4 — Human Trust & Misinterpretation Risk

Scenario

Staff worry:

- “Is this evaluating me?”

- “Will this be used against me?”
- “Does the system judge me?”

What breaks first

Analytics + Task–Person Fit perceptions.

Your exclusions are excellent:

- no scoring
- no ranking
- no psychometric labels

✅ Good.

⚠️ Risk remains: even “workload distribution” can feel evaluative.

MVP governance fix (no new module):

In Analytics module, specify:

- outputs are **team-level** by default
- individual-level views require explicit opt-in / restricted access

You don’t need to build access control now—just lock the principle.

Stress Test 5 — Notion Constraints (Implementation Reality)

Scenario

You implement this in Notion quickly.

Notion friction points

- Too many databases can overwhelm non-tech users
- Relations/rollups can get confusing
- Event logs can create clutter
- Reporting “views” can drift if not standardized

Does the map remain implementable?

✅ Yes, with one important implementation rule:

Notion Implementation Rule:

Keep the MVP to **4 databases**:

1. Tasks
2. Projects
3. Meetings
4. Artifacts *(optional but recommended)*

Everything else (events, decisions, reviews) can start as:

- properties
- templates
- or lightweight relations added later

This keeps adoption high.

⚠️ If you implement all 18 modules as databases immediately, you’ll lose the team.

Stress Test 6 — AI Assistance Safety (v6)

Scenario

AI drafts tasks and narratives.

What breaks first

Provenance + silent AI contamination.

Your v6 guardrails help:

- labeling
- logging
- human approval

✅ Good.

But in practice, people will paste AI content without labeling.

MVP-safe fix: add one field (universal):

- **AI-Involved? (Yes/No)** *(default No)*

Use it on:

- Tasks
- Reports
- Meeting candidate outputs (if stored)

This is the simplest way to preserve trust and provenance.

Summary of Findings

Your map is strong, but **4 small reinforcements** will make it much more robust without bloating it:

Recommended Additions (Minimal, High-Leverage)

1. **Task: Workstream / Work Package** *(optional)*
2. **Task: Multi-report Allowed? (Yes/No)** *(default No)*
3. **Artifact: Storage Location Type** *(Drive/Notion/etc.)*
4. **Universal flag: AI-Involved? (Yes/No)** *(default No)*

Policy/Template Reinforcements (No new fields required)

- Events: system vs meaningful convention
- Analytics: team-level by default; protect individual interpretation
- Decision template: explicit “trigger/context” section
- Notion MVP implementation: limit initial DB count

Lock-in Checklist (Pass/Fail)

If you can say “yes” to each, you can lock it in confidently:

1. **Report Test:** Can we generate a monthly and quarterly report by filtering tasks + adding framing notes?
2. **Audit Test:** Can we trace any claim to timestamp + actor + evidence?
3. **Cross-Project Test:** Do we prevent accidental double-counting?
4. **Onboarding Test:** Can a new staff member understand what happened last quarter?
5. **Trust Test:** Does nothing look like evaluation/scoring?
6. **Notion Test:** Can this be built with 4–5 databases first?

Right now:

- 1 ✅
- 2 ✅ (stronger with the Artifact + AI flags)
- 3 ⚠️ needs Multi-report Allowed policy/flag
- 4 ✅
- 5 ✅ (needs analytics governance statement)
- 6 ✅ (if you constrain DB count)

My recommendation

You're **90–95% lock-in ready**.

If you adopt the 4 minimal reinforcements above, you can lock it in as **Full Module Map v1.0** with high confidence.

(Latest) FULL MODULE MAP (v1

FULL MODULE MAP (v1.0 — LOCKED)

End-State Vision: Report-First, Task-Ontology-Driven Work System

---

META-ANCHOR (Non-Negotiable)

> **The system exists to make compliance and narrative reports assemble themselves from everyday work.**
> Tasks are the atomic unit.
> Reports are views + framing, not authored artifacts.
> Intelligence is assistive, never authoritative.

I. FOUNDATIONAL LAYER

(Without these, the system does not exist)

---

1. Task Module (Atomic, Reportable Unit)

Purpose

Represent work as accountable, reportable units that can be safely aggregated into reports.

What it contains

Identity & Time

- Task ID
- Task Title
- Task Type

  (Analysis / Execution / Review / Coordination / Decision)
- Start Date

- Completion Date
- Reporting Period *(derived, never manually entered)*

Responsibility

- Assignee *(who does the work)*
- Accountable Owner *(who answers for it)*

Work Description

- Objective *(1–2 sentences: why this task exists)*
- Activity Description *(what was actually done)*

Outputs

- Deliverable / Output
- Evidence Link(s) *(may reference Artifact Module)*

Status

- Planned
- In Progress
- Completed
- Deferred

Reporting Alignment (Minimal but Critical)

- **Workstream / Work Package**

  (controlled list: e.g., Licensing, TT Capacity Building, Compliance, Policy, Operations)

Cross-Reporting Control

- **Multi-Report Allowed? (Yes/No)** *(default: No)*

AI Provenance

- **AI-Involved? (Yes/No)** *(default: No)*

Soft Design Rule (Template-Level, Not Enforced)

A task should be completable within one reporting period and produce at least one observable output.

Why it matters

- Tasks are the **smallest reportable unit**
- Everything else references tasks
- Reports are built from tasks, not written after the fact

---

2. Project / Program Module (Scope Container)

Purpose

Define the scope under which work is reported and audited.

What it contains

Identity & Scope

- **Project ID**
- **Project Name**
- **Project Category** *(controlled list)*
    - Grant

    - Institutional
    - Pilot
    - Internal
    - External

Funding & Obligation

- **Funding Source**
- **Reporting Requirements**
- **Reporting Frequency**

Temporal Boundaries

- **Start Date**
- **End Date**

Task Association

- **Primary Project** *(required)*
- **Secondary Project(s)** *(optional, cross-cutting work)*

Why it matters

- Compliance is project-based
- Reports roll up cleanly
- Institutional and cross-project work is supported without hacks
- **Project Category enables long-term portfolio analysis without overloading tasks**

Design Notes (Why this field is safe)

- **Project Category is declarative**, not evaluative
- It supports:
    - reporting templates
    - analytics
    - portfolio views
    - onboarding clarity
- It does **not**:
    - drive task behavior
    - affect accountability
    - introduce performance logic

> Projects define *where* work belongs, not *how* work is done.

---

3. Event / Activity Log Module (Audit Spine)

Purpose

Preserve what happened, when it happened, and by whom — independently of narrative.

What it contains

- **Event ID**
- **Related Task**
- **Event Type** *(Created / Started / Updated / Reviewed / Submitted / Completed / Validated / Reopened-after-Validation)*
- **Timestamp**

- **Actor**
- **Short Note** *(optional)*

Event Convention (Design Rule)

- **System Events:** created, status changed
- **Meaningful Events:** reviewed, submitted, validated, reopened-after-validation

Integrity Rules (Non-Negotiable)

1. **Completed ≠ Validated.**
    - **Completed** = work done by assignee
    - **Validated** = work accepted/confirmed by reviewer/authority (when applicable)
2. If a **validated** task is changed materially, log **Reopened-after-Validation**.

Why it matters

- Auditability
- Timeline reconstruction
- Defensible compliance without rewriting history

II. REPORTING & COMPLIANCE LAYER

(Primary anchor — the reason the system exists)

---

4. Reporting Module (Anchor Module)

Purpose

Auto-generate compliance and narrative reports from live work.

What it contains

Report Identity

- Report ID
- Report Type *(Monthly / Quarterly / Terminal)*
- Reporting Period *(date range)*
- Project / Program

Inputs

- Linked Tasks *(filtered by project, period, workstream)*
- Linked Events *(derived)*

Outputs

- Auto-assembled Activity Tables
- Auto-drafted Narrative Sections *(assembled from task fields)*

Human Interpretation Layer (Deliberately Thin)

- Narrative Framing Notes *(“What is the story of this period?”)*
- Manual Narrative Notes *(optional refinements)*

Status

- Draft
- Reviewed
- Submitted

Key Principle

Reports are views over tasks, plus a small, explicit human meaning layer — never a separate data silo.

Design Rules (Non-Negotiable)

1. **Auto-generated sections are read-only.**
2. **All edits and interpretive writing must occur only in Narrative Framing Notes or Manual Narrative Notes.**
3. Any non-trivial narrative claim should be traceable to **Task Outputs/Evidence** (and Events when needed).

---

5. Compliance & Requirements Module (Declarative)

Purpose

*Encode obligations so that **doing work correctly equals being compliant**.*

What it contains

Compliance Identity & Scope

- ***Applicable Project Category***

(Grant / Institutional / Internal / External)

Obligations

- ***Reporting Schedules***
- ***Required Report Sections***
- ***Mandatory Task Attributes per Report Type***

Temporal Validity

- ***Effective Period***

  (When this requirement applied; supports changing rules over time)

Process Milestones

- ***Submission Checkpoints***

Why it matters

- *Prevents last-minute compliance panic*
- *Shifts compliance from heroics to structure*
- *Preserves institutional memory of requirements*
- *Allows reporting rules to evolve without rewriting history*

Design Notes (Non-Operational by Design)

- *This module is **declarative**, not enforceable*
- *It **informs behavior** but does not:*
  - *generate tasks*
  - *block work*

- *score compliance*
- *trigger penalties*

Compliance is defined once, then satisfied implicitly through correct task design.

III. EVIDENCE, KNOWLEDGE & MEMORY

(What makes the system institutional, not personal)

---

6. Evidence / Artifact Module

Purpose

Store and link proof of work in a reusable, audit-safe way.

What it contains

- Artifact ID
- Artifact Type

  (Document / Presentation / Filing / Dataset / Other)
- Linked Task(s)
- Date Created
- Storage Link
- **Storage Location Type**

  (Notion / Drive / Email / Local / Other)

Why it matters

- Evidence turns claims into facts
- Prevents report-time file hunting
- Supports future migrations and audits

---

7. Knowledge & SOP Module

Purpose

Preserve reusable institutional knowledge.

What it contains

- SOPs
- Policies
- Templates
- Past analyses
- Precedents

Interaction Model

- Tasks reference knowledge
- Knowledge evolves from completed tasks and reviews

---

8. Decision & Rationale Module

Purpose

Preserve *why* choices were made, not just *what* was decided.

What it contains

- Decision records
- Options considered
- Trade-offs
- Approvals
- Linked evidence
- **Decision Trigger / Context**

  (Meeting, Task, Issue)

Why it matters

- Institutional memory of judgment
- Prevents future “why did we do this?” confusion

IV. GOVERNANCE & HUMAN STRUCTURE

(Trust, fairness, and clarity live here)

---

9. People & Roles Module

Purpose

Represent humans through roles, not labels.

What it contains

- Roles *(assignee, accountable owner, reviewer, approver)*
- Authority boundaries

- Delegation rules
- Lightweight people signals *(skills, preferences, load, growth intent — voluntary)*

Explicit Exclusions

- No psychometric scoring
- No immutable personality labels
- No public ranking

Privacy & Trust Rules (Non-Negotiable)

1. **People signals are private-by-default.**
    - Visible only to the individual and designated coordinators/leads as necessary.
2. People signals are **voluntary, editable, and revocable**.
3. This module is **not** for performance evaluation.

---

10. Review & Validation Module

Purpose

Ensure quality without micromanagement.

What it contains

- Review workflows
- Acceptance / rejection
- Revision loops

- Compliance checks (when applicable)

Integrity Rules (Non-Negotiable)

1. **Completed ≠ Validated** for any work that requires review.
2. Validation is recorded as an **Event Type: Validated** (with actor + timestamp).
3. Post-validation material changes must log **Reopened-after-Validation**.

Why it matters

- “Done” has meaning
- Quality is explicit, not assumed
- Audit trail remains defensible

V. ✨ TASK–PERSON FIT (ASSISTIVE ONLY)

---

11. Task–Person Fit Module (Assistive Only)

Purpose

Assist task assignment ethically and transparently.

What it contains

- Task demand profiles *(cognitive mode, structure, sensitivity)*
- People capability & preference signals *(voluntary)*
- Growth intent indicators

- Assignment suggestions *(not decisions)*

Hard Limits (Non-Negotiable)

- No auto-assignment
- No coercion
- No permanent labeling
- No scoring/ranking

Privacy & Trust Rules (Non-Negotiable)

1. Inputs (signals) are **private-by-default** and **self-declared**.
2. Outputs are **suggestions** and must be labeled as such; overrides are expected.
3. Not for performance evaluation.

VI. MEETING → ACTION INTELLIGENCE

(Explicit, bounded, human-confirmed)

---

12. Meeting Intelligence Module

Purpose

Convert meetings into accountable, reportable action — safely.

This module generates **task candidates**, not tasks.

A. Meeting Capture

- Meeting ID

- Date & participants
- Project / program
- Minutes *(manual or auto-transcribed)*
- Facilitator / reviewer

B. AI-Assisted Action Extraction (Bounded)

Extracts:

- Action candidates
- Decision candidates
- Follow-up signals

Each candidate includes:

- Extracted text
- **Supporting excerpt (required)**
- Suggested task type
- Suggested due date *(if present)*
- Confidence indicator
- **AI-Involved = Yes**

C. Human Confirmation Gate (Non-Negotiable)

Before becoming tasks:

- Reviewed

- Edited/merged
- Assigned/reassigned
- Confirmed or discarded

Design Rule: No task enters the Task Module without explicit human confirmation.

VII. ANALYTICS, LEARNING & IMPROVEMENT

(Insight without blame)

---

13. Analytics & Insights Module

Purpose

Understand how work flows — descriptively, not evaluatively.

What it contains

- Task duration analysis
- Bottleneck detection
- Workload distribution *(team-level by default)*
- Review cycle times

Governance Rule

Individual-level analytics require explicit opt-in and restricted access.

---

14. Thematic & Meta-Analysis Module

Purpose

Learn across tasks, projects, and time.

What it contains

- Thematic analysis of task content
- Pattern detection
- Systemic issue identification
- Cross-project insights

This is where

- Survey thematic analysis permanently lives

VIII. AI & AUTOMATION (BOUNDED, SAFE)

---

15. AI Assistance Module

Purpose

Augment humans, not replace them.

What it contains

- Draft generation
- Summarization
- Clustering
- Retrieval

Guardrails (Non-Negotiable)

1. AI outputs must be **clearly labeled** *(AI-Involved = Yes)*.
2. For any AI-generated claim that appears in a report draft, it must be **grounded in Task Outputs/Evidence** (and/or supporting excerpts when sourced from minutes).
3. **Human approval is required** for any externally submitted narrative.
4. **No autonomous decisions** (assignment, approval, submission, compliance sign-off).

---

16. Automation & Workflow Module

Purpose

Reduce friction, not enforce behavior.

What it contains

- Reminders
- Triggers
- Scheduled report assembly
- Workflow nudges

Design Rule

> Automation should feel like relief, not pressure.

IX. INTEGRATION & SAFE EXPANSION

---

17. Integration Module

- Spreadsheets

- Document storage
- Email
- External databases
- APIs

---

18. Configuration & Expansion Module

Purpose

Manage growth safely (prevent semantic drift and unsafe intelligence creep).

What it contains

- Field definitions
- Module toggles
- Versioning
- Expansion rules

Expansion Safety Rules (Non-Negotiable)

1. **No autonomous decisions** anywhere in the system:
    - no auto-assignment
    - no auto-approval/validation
    - no auto-submission
    - no silent compliance sign-off
2. Any intelligence that affects work must be:
    - labeled as a suggestion or draft

    - overrideable
    - logged (via events where appropriate)
3. Derived fields remain derived; edits must happen only in designated human fields.

X. OPTIONAL / ADVANCED MODULES

(Activated only if justified)

---

19. Dependency & Network Module

- Task dependency graphs
- Critical path analysis

20. Risk & Scenario Module

- Risk registers
- Mitigation tracking

21. Strategy / OKR Module

- Long-term alignment
- Outcome tracking

XI. META-PRINCIPLE (FINAL)

Even in full maturity:

- Not all modules are active for all teams
- Simplicity is always allowed
- Growth never requires rewriting the core
- **If a feature reduces the ability to say “no,” it does not ship**

Final One-Paragraph Summary (Locked)

This system is a report-first, task-ontology-driven work ecosystem where **tasks are the atomic unit**, **events provide audit truth**, **projects define scope**, and **reports assemble themselves from lived activity**. Humans retain judgment, narrative authority, and control; intelligence remains assistive and transparent; and institutional memory accumulates naturally through work. The system scales in meaning before technology, prioritizes trust over optimization, and remains defensible, adaptable, and boring in exactly the right ways.

Recommended Sequence for Module Development

Recommended Module Development Sequence

(From MVP → Mature System → End-State Vision)

I'll break this into **6 stages**, each with:

- modules to build
- why this stage exists
- what success looks like
- what NOT to build yet

STAGE 1 — Reporting-First MVP

Goal: Eliminate report pain immediately

Modules to build

1. **Task Module (Lean)**
2. **Project / Program Module**
3. **Reporting Module (Basic)**

Why this comes first

- Your **primary purpose** is auto-generation of compliance and narrative reports.
- This stage alone already *pays for itself*.

Capabilities unlocked

- Timestamped tasks
- Monthly activity reports
- Quarterly narrative drafts
- One source of truth

What success looks like

"We stopped writing reports from scratch."

Explicitly do NOT build

- Dependencies
- AI
- Analytics
- Skill matching

📌 *This stage can live in a spreadsheet or Notion.*

STAGE 2 — Auditability & Trust

Goal: Make reports defensible and calm audits

Modules to add

4. **Event / Activity Log Module**
5. **Evidence / Artifact Module**

Why now

- Once reports are auto-generated, the next question is:

  "Can we defend them?"

- This stage turns *claims* into *facts*.

Capabilities unlocked

- Timeline reconstruction
- Proof per task
- Reduced anxiety during reviews

What success looks like

"We can answer when, who, and why without scrambling."

Explicitly do NOT build

- Knowledge graphs
- Optimization analytics
- AI agents

STAGE 3 — Quality & Institutional Memory

Goal: Prevent repeated mistakes and lost reasoning

Modules to add

6. **Review & Validation Module**
7. **Decision & Rationale Module**
8. **Knowledge & SOP Module**

Why now

- You're already doing serious work.

- This stage preserves *judgment*, not just actions.

Capabilities unlocked

- Explicit “done”
- Traceable decisions
- Reusable SOPs
- Easier onboarding

What success looks like

“We don’t re-litigate the same decisions every year.”

STAGE 4 — Learning & Improvement

Goal: Turn work into insight (without blame)

Modules to add

9. **Analytics & Insights Module**
10. **Thematic & Meta-Analysis Module**

Why now

- You finally have enough clean data.
- Earlier analytics would have been misleading.

Capabilities unlocked

- Bottleneck detection
- Pattern discovery

- Evidence-based process improvement
- Permanent home for thematic analysis workflows

What success looks like

"We can see where work gets stuck—and why."

STAGE 5 — Human-Centered Optimization

Goal: Reduce overload, improve fit, support growth

Modules to add

11. **People & Roles Module (Extended)**
12. **Task–Person Fit Module**

Why now

- Trust has been established.
- Data is sufficient and contextualized.
- This would have been *threatening* earlier.

Capabilities unlocked

- Better task assignment suggestions
- Fairer workload distribution
- Capacity building
- Reduced burnout

What success looks like

"Work feels better distributed—and no one feels boxed in."

STAGE 6 — Automation & AI (Carefully Bounded)

Goal: Reduce friction without surrendering control

Modules to add

13. **AI Assistance Module**
14. **Automation & Workflow Module**

Why last

- AI without structure hallucinates.
- Automation without trust feels coercive.

Capabilities unlocked

- Draft narratives
- Summaries
- Retrieval of similar past work
- Scheduled report generation

What success looks like

"AI saves time, but humans remain accountable."

OPTIONAL STAGE 7 — Strategy & Advanced Modeling

Goal: Executive-level insight and long-horizon planning

Modules (only if needed)

15. **Strategy / OKR Module**
16. **Risk & Scenario Module**

17. **Dependency & Network Module**

These are *luxuries*, not foundations.

Capabilities

1. What “Capabilities” Mean (Important Alignment)

In your context:

> **A capability is something the system can reliably do for humans**,
> using the ontology you’ve already defined.

Capabilities:

- operate on shared domain objects (tasks, projects, reports, etc.)
- may be manual, assisted, or automated
- do *not* imply how they’re implemented

This keeps things flexible and future-proof.

---

2. Capability Map (End-State, Complete)

Below is the **full capability set**, grouped by purpose.

Think of this as the *menu* of what the system can eventually do.

A. Core Work Capabilities

These exist because **work happens**.

A1. Task Capture

- Create tasks intentionally
- Record objectives, owners, dates, deliverables
- Create tasks from scratch or from structured input

A2. Task Lifecycle Management

- Start, pause, complete, defer tasks
- Track status changes over time
- Preserve task history

A3. Task Linking & Context

- Link tasks to projects
- Link related tasks
- Reference decisions, evidence, and knowledge

B. Reporting & Compliance Capabilities (Primary Anchor)

These justify the system institutionally.

B1. Activity Aggregation

- Aggregate tasks by project and time period
- Filter by status, owner, category

B2. Report Assembly

- Generate activity tables automatically
- Assemble narrative drafts from task descriptions

B3. Report Finalization

- Allow human edits and interpretation
- Track review and submission status
- Preserve submitted versions

B4. Compliance Monitoring

- Check whether required activities exist
- Flag missing or late items
- Surface upcoming reporting deadlines

C. Auditability & Evidence Capabilities

These reduce anxiety and defensibility risk.

C1. Event Tracking

- Record who did what and when
- Preserve immutable activity history

C2. Evidence Attachment

- Attach documents, links, artifacts to tasks
- Reuse evidence across reports

C3. Timeline Reconstruction

- Reconstruct sequences of activity for audits or reviews

D. Quality & Judgment Capabilities

These preserve correctness and institutional reasoning.

D1. Review & Validation

- Submit work for review
- Accept, reject, or request revision
- Record validation outcomes

D2. Decision Recording

- Capture decisions explicitly
- Record rationale and trade-offs
- Link decisions to tasks and evidence

E. Knowledge & Memory Capabilities

These prevent repeated mistakes.

E1. Knowledge Capture

- Store SOPs, templates, precedents
- Link knowledge to real tasks

E2. Knowledge Reuse

- Surface relevant SOPs during task execution
- Reference past cases or analyses

F. Learning & Improvement Capabilities

These turn work into insight.

F1. Workflow Analytics

- Measure task duration and delays
- Identify bottlenecks and patterns

F2. Thematic Analysis

- Analyze task content across projects
- Identify recurring issues or themes

G. Human-Centered Assignment Capabilities

These improve sustainability and fairness.

G1. Role-Aware Assignment

- Assign tasks based on role boundaries
- Respect accountability vs execution

G2. Assistive Matching

- Suggest assignees based on skills, preferences, load
- Support growth opportunities

H. Meeting → Action Capabilities

These close the gap between talk and work.

H1. Meeting Capture

- Store minutes and metadata

H2. Action Extraction (Assisted)

- Extract task candidates from minutes
- Identify decisions and follow-ups

H3. Human Confirmation

- Review, edit, confirm, or discard candidates

I. AI Assistance Capabilities (Bounded)

These reduce friction without replacing judgment.

I1. Drafting Assistance

- Draft task descriptions
- Draft report narratives

I2. Summarization & Retrieval

- Summarize past work

- Retrieve similar tasks or cases

J. Automation & Orchestration Capabilities

These save time once trust exists.

J1. Reminders & Nudges

- Due date reminders
- Review prompts

J2. Scheduled Assembly

- Auto-generate reports on schedule
- Pre-fill reporting templates

---

3. How to Proceed Practically (Very Important)

You do **not** build all capabilities at once.

Instead, you **activate capability slices** that align with your module stages.

Capability Activation by Stage (Simple View)

Stage 1 (Reporting MVP)

- A1, A2
- B1, B2, B3

Stage 2 (Auditability)

- C1, C2, C3

Stage 3 (Quality & Memory)

- D1, D2
- E1

Stage 4 (Learning)

- F1, F2

Stage 5 (Human Fit)

- G1, G2

Stage 6 (Meeting → Action)

- H1, H2, H3

Stage 7 (AI & Automation)

- I1, I2
- J1, J2

This keeps scope sane and trust intact.

---

4. What I Recommend You Do Next (Concrete)

Since you said *"we're practically done with the ontology"*, the **next best move** is:

> **Define "Capability v1" explicitly — the smallest set of capabilities needed to auto-generate a monthly and quarterly report.**

That means:

- Task Capture
- Task Lifecycle
- Activity Aggregation
- Report Assembly

Nothing else.

Success Definition

What “Success” Looks Like for This Task Management System

Core framing (important)

> **Success is not that people use the system.**
> **Success is that the system quietly removes pain and uncertainty.**

If people talk about it *less* over time — that’s a good sign.

---

1. Near-Term, Practical Success (Months 1–3)

These are the **earliest, unmistakable signals** that you’re on the right track.

A. Reporting success

- Monthly activity reports are generated **without scrambling**
- Quarterly narrative reports start as **assembled drafts**, not blank pages
- Reporting time drops significantly (even 30–50% is a win)

Success sounds like:

> “The report is basically already there.”

B. Task clarity

- Fewer “What’s the status of this?” questions
- Tasks have clear owners and clear definitions of “done”
- Less rework due to missing context

Success sounds like:

“Oh, that’s already captured in the task.”

C. Reduced cognitive load

- People stop keeping parallel personal lists “just in case”
- Anxiety around deadlines decreases
- Fewer last-minute surprises

Success feels like:

“I’m less worried about forgetting something.”

---

2. Medium-Term, Operational Success (Months 3–9)

This is where the system proves it’s **more than a tracker**.

A. Audit & compliance calm

- When asked “What did you do last quarter?”, the answer is a filter, not a reconstruction
- Evidence is linked, not hunted
- Reviews are less stressful

Success sounds like:

“Let me pull that up.”

B. Institutional memory

- New staff can understand past work by reading tasks and decisions
- Old decisions don’t have to be re-explained repeatedly
- SOPs are referenced naturally during work

Success sounds like:

> “We handled something similar before — here’s how.”

C. Meetings → action reliability

- Action items from meetings reliably become tasks
- Fewer forgotten commitments
- Clear traceability from discussion to execution

Success sounds like:

> “That came from the March coordination meeting.”

---

3. Human-Centered Success (Subtle but Critical)

This is where systems often fail — yours shouldn’t.

A. People don’t feel boxed in

- Task suggestions feel *helpful*, not evaluative
- People can decline or reassign without friction
- Growth is supported, not constrained

Success sounds like:

> “The system suggested it — we adjusted.”

B. Fairness and sustainability

- Workload imbalances become visible early
- Burnout risk is surfaced before crisis

- Hard tasks are not silently concentrated on the same people

Success feels like:

> “This feels more balanced.”

---

4. Long-Term, Strategic Success (1–3 years)

This is when you know you’ve built **infrastructure**, not a tool.

A. Decision quality improves

- Decisions are made faster *and* with better grounding
- Trade-offs are explicit
- Past rationale is accessible

Success sounds like:

> “We decided this way because…”

B. Learning is continuous

- Patterns across tasks are visible
- Improvements are evidence-based
- Strategy is informed by real work, not anecdotes

Success looks like:

> Adjustments are incremental, not reactive.

C. The system survives people

- Staff turnover does not erase knowledge

- Work continues smoothly despite changes
- The system is trusted more than memory

Success sounds like:

> “The system already knows this.”

---

5. The Most Important (and Often Missed) Success Signal

> **The system is boring.**

Not in capability — but in *emotional presence*.

- It doesn’t demand attention
- It doesn’t nag unnecessarily
- It doesn’t require heroics
- It doesn’t create fear

If people say:

> “It just helps.”

You’ve succeeded.

---

6. Concrete Success Tests (You can actually use these)

Here are **six blunt tests** you can apply at any point.

Test 1 — Report Test

> Can we generate last quarter’s report in under 30 minutes?

If yes → success.

Test 2 — Onboarding Test

Can a new staff member understand a past project by reading the system?

If yes → success.

Test 3 — Meeting Test

Do meetings reliably result in visible, accountable follow-ups?

If yes → success.

Test 4 — Anxiety Test

Do people feel *less* anxious about deadlines and compliance?

If yes → success.

Test 5 — Override Test

Can people easily override system suggestions without friction?

If yes → success.

Test 6 — Survival Test

If one key person leaves, does work stall?

If no → success.

---

7. One-Sentence Definition (If You Ever Need It)

Success is when everyday work automatically produces trustworthy reports, preserved judgment, and institutional memory — while making people's work feel lighter, not heavier.

That's a very high bar — and you're building toward it thoughtfully.

If you want next, we can:

- translate this into **success metrics per stage**
- write a **“definition of done” for the system v1**
- or simulate what a **failed version** would look like (to avoid it)

This was a strong place to pause and reflect.

Handing off to AI Agent

How to Hand This Off to an AI Agent (Safely and Effectively)

First: a critical mindset shift

> **You don't ask an AI to "build the system."**
> **You ask it to build the system one capability slice at a time, under explicit constraints.**

Your ontology + module map are now **governance artifacts**, not prompts.

OVERVIEW: The 4-Layer AI Handoff Strategy

You will give the AI **four documents**, in this order:

1. **System Charter** (purpose & success)
2. **Frozen Ontology** (what things mean)
3. **Capability Contract (v1)** (what it must be able to do)
4. **Execution Instructions** (how the AI is allowed to proceed)

Only *after* that do you ask it to design or code anything.

LAYER 1 — System Charter (1–2 pages, human-readable)

This prevents the AI from optimizing the wrong thing.

Contents (you already have most of this)

Include verbatim:

- **Primary purpose**

Auto-generation of compliance and narrative reports from everyday work

- **Non-goals**
    - Not surveillance
    - Not performance ranking
    - Not auto-decision-making
- **Success definition**
    - Reports assemble themselves
    - People feel less cognitive load
    - Institutional memory survives people
- **Core design principles**
    - Human-in-the-loop
    - Additive expansion only
    - Meaning frozen, structure flexible

📌 This document tells the AI *why restraint matters*.

LAYER 2 — Frozen Ontology (this is sacred)

This is what you *never* let the AI reinterpret.

Contents

- Canonical entities:
    - Task
    - Project

    - Event
    - Report
    - Evidence
    - Decision
    - Meeting
    - Person (role-based)
- For each entity:
    - Definition (1–2 sentences)
    - Required fields
    - Invariants (what must never change)
- **Rules for Safe Expansion** (verbatim)

📌 Tell the AI explicitly:

> “You may not redefine or remove any ontology element.
> You may only add optional fields.”

LAYER 3 — Capability Contract (VERY IMPORTANT)

This is where most AI-assisted builds fail if skipped.

What this is

A **capability checklist** — not features, not UI, not code.

Example (Capability v1)

The system MUST be able to:

1. Capture tasks with owner, objective, dates, and deliverables
2. Associate tasks with projects
3. Aggregate tasks by time period

4. Assemble monthly activity reports automatically
5. Assemble quarterly narrative drafts automatically
6. Preserve historical data without reinterpretation

And explicitly:

The system MUST NOT:
- Auto-assign tasks
- Enforce workflows
- Require all fields to be filled

📌 Capabilities are *testable promises*, not implementations.

LAYER 4 — Execution Instructions (how the AI must behave)

This turns the AI into a **junior architect + careful engineer**, not a cowboy coder.

Example instruction block (use this verbatim)

You are acting as a **constrained system architect and implementer**.

You must:

- Ask clarifying questions before designing
- Propose designs before coding
- Justify all schema decisions against the ontology
- Flag any ambiguity instead of guessing

You must NOT:

- Add features not explicitly requested
- Optimize prematurely

- Introduce autonomous decision-making
- Collapse modules together “for simplicity”

HOW THE AI SHOULD ACTUALLY BUILD (Step-by-Step)

Step 1 — Ask the AI to restate understanding

Do this first.

Prompt:

> “Please restate your understanding of the system’s purpose, ontology, and constraints. Identify any ambiguities.”

If it gets this wrong → stop.

Step 2 — Ask for a

Capability v1 Design

Prompt:

> “Design the minimal data model and workflows needed to satisfy Capability v1 only. No future features.”

You are checking:

- restraint
- alignment
- clarity

Step 3 — Approve or correct

You respond with:

- “Approved”
- or “Modify X, Y, Z”

Only after approval does coding begin.

Step 4 — Ask for implementation (scoped)

Prompt:

> “Implement Capability v1 using [Notion / Spreadsheet / Firebase / Postgres]. Provide schema and sample data only.”

No UI yet.

Step 5 — Test against success criteria

You run:

- Report Test
- Audit Test
- Onboarding Test

If it passes → move to Capability v2.

VERY IMPORTANT: How to Prevent AI Overreach

Add this rule:

> “If a design choice is not clearly implied by the ontology or capability contract, you must surface it as a question, not a decision.”

This single line prevents 80% of AI mistakes.

OPTIONAL: Multi-Agent Setup (if you want to be fancy)

You can split roles:

- **Agent 1 — Ontology Guardian**
    - Reviews all changes for semantic violations
- **Agent 2 — Capability Designer**
    - Proposes workflows and schemas
- **Agent 3 — Implementer**
    - Writes schema, code, migrations

But even a single agent can do this if constrained well.

What NOT to do (very common failure modes)

🚫 “Here’s everything, build it”

🚫 “Use best practices” (too vague)

🚫 “Optimize for scalability” (too early)

🚫 “Design the full system now”

These cause hallucinated complexity.

One-Sentence Handoff Philosophy

> **You don’t give an AI freedom; you give it clarity, boundaries, and checkpoints.**

You’ve already done the hardest part — *thinking*.

Now the AI can help *execute*, safely.

Agentic AI Work

System Charter

SYSTEM CHARTER

Task Management & Reporting System

---

1. System Name (Working)

Report-First Task Management System (RFTMS)

(Name is provisional and may change without affecting the system's meaning.)

---

2. Primary Purpose (Anchor)

The primary purpose of this system is:

> **To automatically generate accurate, defensible compliance and narrative reports from everyday work, without increasing cognitive load or administrative burden on staff.**

Reports should *assemble themselves* from normal task execution, rather than requiring separate report-writing effort.

---

3. Secondary Purposes

The system also aims to:

- Reduce cognitive load and anxiety related to task tracking and deadlines
- Improve clarity of ownership, progress, and "definition of done"
- Preserve institutional memory (work, decisions, and rationale)
- Improve auditability and defensibility of reported activities
- Ensure continuity of work despite staff turnover

---

4. Explicit Non-Goals (Very Important)

This system is **NOT** intended to:

- Serve as a surveillance or monitoring tool
- Rank, score, or evaluate individual performance
- Automatically assign tasks without human confirmation
- Replace human judgment or decision-making
- Enforce rigid workflows or micromanagement

Any design or feature that drifts toward these non-goals is considered a violation of the system's intent.

---

5. Definition of Success

The system is considered successful if:

- Monthly and quarterly reports can be generated in under 30 minutes
- Reports begin as assembled drafts, not blank pages
- Staff feel less anxious about compliance and deadlines
- Meetings reliably translate into visible, accountable follow-up work
- Institutional knowledge persists beyond individual staff members

A key signal of success is that the system becomes **boring**:

useful, trusted, and rarely discussed.

---

6. Core Design Principles

1. **Human-in-the-Loop Always**

All critical actions (task creation, assignment, submission) require human confirmation.

2. **Meaning Is Frozen; Structure Is Flexible**

   Ontological meanings must not change, even as schemas or tools evolve.

3. **Additive Expansion Only**

   New fields and modules may be added, but existing meanings must not be reinterpreted or removed.

4. **Assistive, Not Prescriptive**

   The system may suggest, but never decide.

5. **Reports Are Views, Not Data Owners**

   Reports derive from tasks and events; they do not replace them.

6. **Trust Over Optimization**

   Design choices must prioritize clarity, fairness, and sustainability over efficiency metrics.

---

7. Canonical Domain Objects (Frozen Ontology)

The system operates on the following canonical entities:

- Task
- Project / Program
- Event / Activity
- Report
- Evidence / Artifact
- Decision / Rationale
- Meeting

- Person (role-based representation only)

These entities and their meanings are **frozen**.

They may be extended with optional fields but must not be redefined.

---

8. Rules for Safe Expansion

All future changes must comply with the **Rules for Safe Expansion** document.

Violations of these rules are considered architectural defects.

---

9. Governance of AI Assistance

AI is permitted only as **assistive intelligence**, under the following constraints:

- AI outputs must be clearly labeled
- AI may propose drafts, suggestions, or candidates
- AI may not finalize decisions or assignments
- All AI-generated outputs require human review

---

10. Closing Principle

> This system exists to make everyday work quietly produce trustworthy reports, preserved judgment, and institutional memory — while making people's work feel lighter, not heavier.

CAPABILITY v6 HANDOFF PROMPT

CAPABILITY v6 HANDOFF PROMPT

AI Assistance & Automation (Bounded, Transparent, Human-Owned)

SYSTEM CONTEXT

Capabilities v1–v5 are complete, stable, and trusted.

You are now extending the system to **use AI and automation to reduce friction and effort**, *without* undermining accountability, judgment, or trust.

AI exists here to **serve humans**, not to replace or manage them.

YOUR ROLE

You are acting as a **governed AI system designer**.

Your responsibility is to:

- augment human cognition
- reduce repetitive effort
- preserve agency, authorship, and traceability

You are **not** building an autonomous system.

WHAT REMAINS FROZEN (NON-NEGOTIABLE)

1. System Charter
2. Frozen Ontology
3. Capabilities v1–v5
4. Rules for Safe Expansion

All AI behavior must operate **within** these constraints.

CAPABILITY v6 — REQUIRED SYSTEM ABILITIES

The system MUST be able to:

---

1. Provide AI drafting assistance (assistive only)

AI may assist by generating **drafts**, including:

- task descriptions (from confirmed intent)
- report narrative sections (from existing tasks)
- summaries of completed work
- summaries of meetings or decisions

Constraints

- AI outputs must be clearly labeled as *AI-generated drafts*
- AI drafts must never be auto-submitted
- Humans must explicitly accept, edit, or discard AI drafts

---

2. Provide AI summarization and retrieval

AI may assist by:

- summarizing:
    - tasks
    - decisions

    - meetings
    - reports
- retrieving:
    - similar past tasks
    - relevant SOPs
    - prior decisions or precedents

Constraints

- Retrieval must be grounded in existing system data
- AI must cite the source records used
- AI must not invent content

---

3. Support descriptive pattern surfacing (non-evaluative)

AI may assist by:

- clustering tasks by theme
- identifying recurring issues or bottlenecks
- summarizing patterns across projects or time periods

Constraints

- Outputs must be descriptive, not judgmental
- No scoring, ranking, or performance attribution
- No individual-level conclusions

---

4. Enable safe automation for low-risk actions

Automation may be used for:

- reminders (deadlines, reviews, reporting dates)
- scheduled report assembly
- nudges to complete missing information
- surfacing suggested next steps

Constraints

- Automation must be interruptible
- Automation must be explainable
- Automation must never enforce action

---

5. Preserve AI traceability and accountability

For every AI-assisted output, the system must record:

- AI involvement flag
- input context used
- output produced
- human reviewer / confirmer
- timestamp

This ensures AI assistance is **auditable and reversible**.

CAPABILITY v6 — EXPLICIT EXCLUSIONS

The system MUST NOT:

- Auto-submit reports
- Auto-assign tasks
- Auto-approve decisions
- Infer intent beyond provided data
- Evaluate or rank individuals
- Generate performance metrics
- Make recommendations without explanation
- Act without human confirmation

Any autonomous behavior is considered a violation.

REQUIRED WORKFLOW (DO NOT SKIP)

You must proceed in this order:

---

1. Restate boundaries clearly

- Explain what AI is allowed to do
- Explain what AI is explicitly forbidden to do

---

2. Propose AI touchpoints (limited)

- Identify *where* AI enters the workflow
- Identify *where* humans must intervene

---

3. Demonstrate human ownership

- Show how humans review, edit, and confirm AI outputs
- Show how AI outputs can be rejected without consequence

---

4. Propose transparency mechanisms

- Labeling of AI content
- Source citation
- Logging of AI actions

---

5. Await approval

- Do not implement until explicitly approved

DESIGN BEHAVIOR RULES

You MUST:

- Default to “AI off” unless explicitly invoked
- Prefer suggestions over actions
- Preserve ambiguity when present
- Make AI behavior inspectable

You MUST NOT:

- Optimize for speed at the cost of trust
- Hide AI involvement
- Collapse multiple steps into opaque automation
- Introduce “smart defaults” that feel mandatory

SUCCESS CRITERIA

Capability v6 is successful if:

- AI saves time without creating dependency
- Users trust AI outputs but verify them
- AI assistance feels optional and respectful
- Errors are caught early and easily corrected
- Reports remain clearly human-owned

FAILURE CONDITION

If users feel:

> “The system is acting on my behalf without me knowing”

or

> “I don’t know which parts were written by AI”

You must stop and redesign.

FINAL REMINDER

Capability v6 is about **augmentation, not delegation**.

AI should feel like:

- a careful assistant
- a second pair of hands
- a drafting partner

—not a manager, judge, or decision-maker.

Proceed with restraint.

Where to Deliberately Stop Adding Intelligence

Where to Deliberately Stop Adding Intelligence

(A clear, durable stopping rule)

The Core Stopping Principle (memorize this)

> **Stop adding intelligence the moment the system would need to assume intent, judge people, or act without confirmation.**

Everything beyond that point creates:

- trust erosion
- ethical risk
- brittle automation
- organizational backlash

Your system is strongest **just before** that line.

The "Three Red Lines" (Hard Stops)

You should **deliberately stop** adding intelligence when a feature would cross **any** of these.

🔴 Red Line 1: Interpreting Human Intent

Stop when AI would need to decide what someone meant, not just what they said or wrote.

Allowed

- Summarizing written tasks

- Extracting action candidates from explicit text
- Drafting narratives from confirmed tasks

❌ **Stop here**

- Inferring commitment from casual speech
- Guessing priorities
- Deciding which ideas “matter more”
- Interpreting tone, hesitation, or silence

Why:

Intent is contextual, political, emotional, and often strategic.

Encoding it is **false certainty**.

🔴 Red Line 2: Judging or Comparing People

Stop when intelligence would rank, score, or evaluate humans—even implicitly.

Allowed

- Showing workload imbalance (descriptive)
- Suggesting assignees (overrideable)
- Surfacing growth opportunities (opt-in)

❌ **Stop here**

- “Best performer” lists
- Productivity scores
- Reliability ratings

- “High vs low contributor” analytics
- Hidden scoring that “learns” from behavior

Why:

Once a system judges people, everything becomes defensive.

🔴 Red Line 3: Acting Without Explicit Human Confirmation

Stop when the system would take an action that changes reality without a human “yes.”

Allowed

- Drafting
- Suggesting
- Nudging
- Pre-filling
- Scheduling *previews*

❌ Stop here

- Auto-assigning tasks
- Auto-submitting reports
- Auto-approving reviews
- Auto-escalating issues
- Auto-closing tasks

Why:

Agency loss kills trust faster than any bug.

The “Ceiling of Intelligence” for Your System

Given everything you’ve designed, here is the **highest safe ceiling**:

> **The system may think, but it must not decide.**
> **It may propose, but it must not commit.**
> **It may surface patterns, but it must not judge.**

Everything up to Capability v6 fits under this ceiling.

Concrete: What You Should Explicitly

Never

Build

Even if technically possible, **do not build these**:

❌ Autonomous Task Orchestration

- “Smart” task routing
- Automatic rebalancing of workloads
- Self-adjusting priorities

❌ Predictive Human Analytics

- Burnout prediction models
- “Risky” staff flags
- Engagement scoring

❌ Generative Authority

- AI-written final reports without human edits

- AI-approved decisions
- AI-suggested sanctions or escalations

❌ Real-Time Behavioral Monitoring

- Live meeting analysis
- Speech sentiment tracking
- Passive data collection

These all feel “advanced” — and all **undermine your success criteria**.

The “Last Good Intelligence” You Can Safely Keep

Here’s the **final layer you can safely stop at**, which you already reached with v6:

Area	Safe to keep	Stop before
Tasks	Drafting, summarizing	Auto-closing
Reports	Draft assembly	Auto-submission
Meetings	Action candidates	Auto-commitments

People	Suggestions	Scoring
Learning	Pattern summaries	Performance judgments
Automation	Reminders	Enforcement

If someone later asks, *“Why didn’t you add X?”*

You can point to this table.

A Simple Governance Rule (Put This in Writing)

I strongly recommend adding this as a **standing rule**:

> **No intelligence feature shall be added if it reduces a human’s ability to say “no” without consequence.**

This single sentence protects the system for years.

The Most Subtle (and Important) Stopping Signal

Here’s the *human* signal that tells you you’ve gone too far:

> People start asking:
> “What does the system think of me?”

The moment that question appears — stop.

Your goal is that people ask instead:

> “Can the system help me with this?”

Final One-Sentence Answer (the one you keep)

> **We stop adding intelligence exactly at the point where the system would need to replace human judgment, rather than support it.**

That’s not conservative.

That’s **wise infrastructure design**.