Introduction

> **Canonical source:** This Markdown file.
> Word and PDF versions in `docs/handbook/` are derived artefacts and may be regenerated.
> They may be committed for convenience, but the Markdown source remains authoritative.

The CloudPedagogy Course Engine is designed for people who are making **real, accountable decisions** about AI-supported course and curriculum design.

In recent years, generative AI tools have made it easier than ever to produce educational content quickly. At the same time, they have made it harder to see where **judgement, responsibility, and intent** actually sit. Content can now be generated at scale, but accountability often becomes blurred — particularly in institutional, educational, and governance-sensitive contexts.

This handbook exists to support a different way of working.

Rather than treating AI as an author or decision-maker, the Course Engine treats AI as an **assistant within a structured, human-led design process**. It is designed to help educators, learning designers, technologists, and institutions use AI **without losing control of purpose, quality, or accountability**.

At a practical level, the CloudPedagogy Course Engine is a **structured course compilation system**. It turns a single, explicit course specification into **consistent, auditable learning artefacts** that can be reviewed, updated, and justified over time. This makes it particularly suitable for contexts where courses must stand up to internal review, quality assurance, governance and audit processes, and long-term maintenance and revision.

Rather than generating content directly, the Course Engine treats **course intent and structure as first-class design elements**. Courses are compiled from a single source of truth, produce reproducible outputs across formats such as web, PDF, and markdown, and generate machine-readable metadata to support inspection, reporting, and audit.

Where capability frameworks are used, the system supports three related but distinct mechanisms:

- **Framework alignment** (v1.6+): a *declaration* of which framework and domains the course intends to reference (informational, non-enforced).
- **Capability mapping** (v1.1+): structured *coverage/evidence metadata* that can be reported on, and (optionally) validated against policy rules.
- **Design intent** (v1.12+): a *declaration* of rationale, AI positioning and boundaries, and governance/review context (informational, non-enforced). Design intent is recorded in `manifest.json` with a stable hash and surfaced via `course-engine explain`.

The Course Engine is intentionally **non-prescriptive**. It supports reflection, review, traceability, and transparency, but it does not evaluate pedagogical quality, determine academic merit, or decide whether a course should be approved or adopted. Those responsibilities remain firmly with the human user.

This handbook explains why the Course Engine was created, how it is intended to be used, what it deliberately does *not* automate, and where responsibility remains human-owned. It does not assume prior technical expertise, but it does assume professional judgement.

The sections that follow are written to be practical rather than promotional, explicit rather than implicit, and stable over time — even as the software itself continues to evolve.

In short, the CloudPedagogy Course Engine is **infrastructure for responsible course production**, not an automated teaching system.

STAGE 1 — ORIENTATION & PURPOSE

Why this software exists, and why it looks the way it does

1. What This Handbook Is For

This handbook is written to support **responsible, confident use** of the CloudPedagogy Course Engine.

It is intended for people who are using, or considering using, the Course Engine to design courses, modules, or other structured learning materials with AI assistance — and who remain **accountable for the outcomes** of that work.

In practice, many tools explain *what they can do* but say very little about *how they should be used*, *what assumptions they make*, or *where responsibility actually sits*. This handbook exists to fill that gap.

Its purpose is to help you:

- understand **why** the Course Engine was created
- understand **what problems** it is designed to address
- use the system effectively **without surrendering professional judgement**
- recognise **what the system deliberately does not automate**

- make decisions about AI-supported course design that are **defensible, explainable, and reviewable**

This is not a technical manual, and it is not a marketing document. You do not need to understand the internal mechanics of the software to use it well.

Instead, this handbook is a **use-and-rationale guide**. It is designed to support thoughtful practice, particularly in contexts where quality, accountability, and governance matter.

2. What the Course Engine Is (and Is Not)

What the Course Engine *is*

The CloudPedagogy Course Engine is a **capability-aware system for designing structured learning artefacts**.

In practical terms, it supports the creation of materials such as short courses, modules, handbooks, guides, and curriculum documentation — particularly where coherence, traceability, and reviewability are important.

The Course Engine is designed to help you:

- clarify and structure ideas before substantial content is generated
- maintain coherence across sections, themes, and learning intent
- build reflection and revision into the design process
- produce outputs in multiple formats suitable for different contexts
- keep key design decisions visible rather than hidden

It supports **thinking, drafting, and iteration**. It is designed to help you work *with* AI while retaining ownership of direction, emphasis, and judgement.

What the Course Engine *is not*

Just as important as what the Course Engine does is what it **deliberately does not do**.

It is not an "AI writes my course" tool.
It is not a curriculum auto-designer.
It is not a learning management system.
It is not a pedagogical authority.
It is not a compliance or approval engine.

The Course Engine does not decide:

- what is pedagogically appropriate
- what is ethically acceptable
- what should or should not be taught
- how learners should be assessed

Those decisions remain **human-owned**.

This boundary is intentional. The Course Engine is designed to support professional work, not to replace professional responsibility. Where judgement matters, the system is designed to **surface decisions**, not make them on your behalf.

3. Why This Software Exists

The problem it responds to

AI-supported course design often fails not because educators lack expertise or care, but because of **how tools are introduced into the design process**.

Common patterns include generating content before intent is clarified, producing large volumes of material before structure exists, or considering governance only once something has already been published. In these situations, AI output can appear authoritative simply because it looks complete.

Over time, these patterns tend to produce courses that feel fragmented, contain hidden assumptions, or are difficult to justify under review. Accountability becomes unclear, and revision becomes harder rather than easier.

The design response

The Course Engine exists to support a different approach.

Instead of starting with content, it starts with **intent**.
Instead of scaling quickly, it prioritises **structure and coherence**.
Instead of hiding decisions, it keeps them visible.
Instead of treating publication as an endpoint, it treats it as a moment in an ongoing process.

In short, it is designed to make **good practice easier**, without pretending to automate expertise, responsibility, or professional judgement.

4. Who the Course Engine Is Designed For

The Course Engine is designed for people who design learning in **real institutional or professional contexts**.

This includes educators, learning designers, learning technologists, and researchers who are accountable for the quality, coherence, and impact of the materials they produce. It is particularly relevant where courses must be reviewed, updated, reused, or justified over time.

Typical users include those who need outputs they can:

- review critically
- adapt collaboratively
- explain to others
- maintain over multiple iterations

The Course Engine is **not optimised** for mass-content generation or consumer-grade "one-click" publishing. Its value lies in supporting careful design rather than rapid output.

5. The Core Design Philosophy

At the heart of the Course Engine are five guiding principles.

First, **humans decide; AI assists**. The system is built on the assumption that responsibility remains with the human user.

Second, **structure shapes quality**. Clear structure improves coherence, interpretability, and long-term maintainability.

Third, **transparency beats optimisation**. The system favours explainable processes over hidden automation.

Fourth, **governance is a design feature**, not an afterthought. Reviewability and traceability are treated as part of the system's purpose.

Finally, **courses are never finished — only published for now**. Iteration, revision, and renewal are expected parts of responsible practice.

Every major design choice in the Course Engine follows from these principles.

6. How to Read and Use This Handbook

This handbook is not intended to be read once and put aside.

You may find it useful to read the early sections to understand intent and boundaries, then return to later sections while actively using the Course Engine. Some sections may become more relevant when you update or expand existing work.

You do not need to read the handbook linearly in order to use the software effectively. It is designed to support use **over time**, not just at the point of first contact.

7. How This Handbook Will Grow Over Time

This handbook is a **living document**.

As the Course Engine evolves, new sections may be added to describe additional workflows, outputs, or features. Guidance may also be extended to reflect emerging practices around governance, review, and responsible AI use.

These changes will appear as **additions or extensions**, not as rewrites of the core rationale.

The orientation and purpose set out in this stage are intended to remain **stable across versions**, providing a consistent foundation even as the software develops.

For example, v1.6 introduced support for **external lesson source files**, **lesson source provenance**, and a clearer separation between **declared framework alignment** and **capability mapping evidence**; this handbook reflects that change through targeted additions rather than by rewriting its core rationale.

STAGE 2A — HOW THE COURSE ENGINE WORKS (CONCEPTUAL)

Understanding the workflow before using the software

8. A Deliberate, Human-Centred Workflow

The CloudPedagogy Course Engine follows a **deliberate, human-centred workflow**.

Unlike many AI tools, it does not begin by generating content. Instead, it begins by asking you to clarify **what you are trying to build** and **why**. This is not a technical constraint; it is a design choice intended to protect judgement, intent, and accountability.

At a conceptual level, the workflow moves through five broad phases: clarifying intent, defining structure, using AI assistance within that structure, reviewing and refining the result, and publishing outputs that remain editable and reviewable. Each phase exists to protect a different aspect of responsible course design.

The order matters. Skipping steps may be faster in the short term, but it usually creates problems later — particularly when a course needs to be reviewed, adapted, or justified.

9. Starting With Intent, Not Output

Many AI-enabled tools begin by asking what you want them to generate. The Course Engine begins by asking something more fundamental: **what are you trying to create, and for what purpose?**

From **v1.12**, this "start with intent" step can also be captured explicitly in the course specification itself, using an optional `design_intent` block in `course.yml`. This metadata is **informational** (it does not affect build or validation behaviour), but it allows authors to record the course's rationale, AI positioning, decision boundaries, and review context in a way that is **auditable and revisitable** over time. When present, it is recorded in `manifest.json` and surfaced by `course-engine explain` in both JSON and text reports.

This distinction is important because courses are not simply collections of text. They are learning journeys, arguments about what matters, and structured commitments made to learners and institutions. When intent is unclear, even well-written content can become incoherent or misleading.

By encouraging you to articulate intent early, the Course Engine helps prevent common failure modes such as unfocused courses, contradictory sections, or content that appears polished but lacks direction.

10. Why Structure Comes Before Writing

Once intent is clear, the Course Engine focuses on **structure**.

Structure includes decisions about sections and sub-sections, narrative flow, emphasis, progression, and boundaries between what is included and what is not. These decisions are made before substantial text is generated.

This ordering is intentional. Structure shapes meaning. It constrains ambiguity, improves coherence, and makes revision possible. Without structure, content tends to grow unevenly and becomes difficult to review or adapt.

11. The Role of AI Within the Workflow

Only after intent and structure are established does AI enter the process.

Within the Course Engine, AI is used as an **assistant**, not an authority. It can help draft explanations, elaborate on ideas, or suggest alternative phrasing, but its output is always provisional and subject to human review.

AI does not define scope, set priorities, determine values, or decide what is pedagogically appropriate. Those decisions remain with the human user.

12. Iteration as a Normal Part of Design

The Course Engine assumes that first versions are incomplete and that clarity improves through revision.

Rather than treating iteration as an exception or a failure, the workflow is designed to support **ongoing refinement**. You are expected to revisit earlier decisions, adjust structure, refine emphasis, and expand or contract sections as your understanding develops.

13. Review and Reflection Built Into the Process

A key part of responsible course design is pausing to reflect.

Throughout the workflow, the Course Engine encourages review by keeping content transparent and editable rather than hiding decisions behind automation. This makes it easier to ask whether the material still reflects the original intent, whether anything is overstated or underspecified, and where learners might misunderstand key ideas.

14. Publishing Without Lock-In

When you publish outputs from the Course Engine, you are not locking your work into a single system or format.

Outputs are designed to be editable, portable, and reusable. They can be shared for review, adapted collaboratively, or maintained over time without dependence on the engine itself.

15. What This Workflow Is Designed to Protect

The workflow used by the Course Engine is designed to protect several things at once.

It protects **intent**, by making it explicit early in the process. It protects **judgement**, by keeping humans in control of key decisions. It protects **quality**, by enforcing structure before scale. It protects **governance**, by avoiding opaque automation and preserving traceability. And it protects **longevity**, by producing outputs that can be maintained and revised over time.

16. What the Workflow Does Not Guarantee

It is equally important to be clear about what the Course Engine does not guarantee.

It does not ensure that a course is pedagogically sound, that content is appropriate for a particular audience, that ethical decisions are correct, or that learning outcomes will be effective. Those judgements require context, expertise, and professional responsibility.

Stage 2A Summary

By the end of this stage, you should have a clear understanding of how the Course Engine is intended to be used at a conceptual level. You should understand why it starts with intent and structure, how AI assistance is deliberately limited, why iteration and reflection are expected, and where responsibility remains human-owned.

STAGE 2B — CAPABILITY-AWARE COURSE DESIGN

Designing courses that preserve judgement, responsibility, and quality

17. What "Capability-Aware" Means in Practice

When this handbook refers to **capability-aware course design**, it is not referring to a technical feature or a metadata requirement. It is referring to a way of thinking about what a course is *for*, and what it must *support* in the people who use it.

A capability-aware approach begins with the recognition that courses do more than transmit information. They shape how learners think, decide, and act. They also shape how educators justify their choices, how institutions demonstrate responsibility, and how decisions stand up to review over time.

18. Capability Is Not the Same as Content Coverage

A common mistake in course design is to treat capability as a list of topics to be covered.

From a capability-aware perspective, this is insufficient. Capability is not simply about whether a concept appears somewhere in the material. It is about whether learners are supported to understand, interpret, apply, question, and revisit ideas in meaningful ways.

19. Designing With Capability in Mind From the Start

Capability-aware design starts early in the workflow, long before detailed content is written.

When you clarify the purpose of a course, define its structure, and decide how ideas will be introduced and revisited, you are already making capability decisions.

20. Capability and Human Judgement

A capability-aware course recognises that **judgement cannot be automated**.

In AI-supported design, there is a temptation to treat outputs as authoritative simply because they are fluent or comprehensive. The Course Engine resists this by maintaining clear boundaries between assistance and decision-making.

21. Making Capability Visible Without Enforcing It

In many institutional contexts, there is a need to explain how a course supports particular capabilities, frameworks, or priorities. At the same time, over-enforcement can be counterproductive, reducing complex judgement to checklist compliance.

The Course Engine is designed to support **visibility without enforcement**.

In v1.6 this is expressed through a clearer distinction:

- **framework_alignment**: declares intent and context (informational; supports inspection and reporting even without mapping)
- **capability_mapping**: expresses structured coverage/evidence (enables coverage reports and validation rules)

22. Capability, Coherence, and Progression

Capability-aware design is closely linked to coherence and progression.

Courses that support capability well tend to have a clear sense of how ideas build on one another, how complexity increases, and how learners are expected to engage more critically over time.

23. Capability and Responsibility to Learners

Designing for capability is also a matter of responsibility to learners.

Courses implicitly communicate what is valued, what is expected, and where responsibility lies. A course that presents AI-generated content as unquestionable fact sends a very different signal from one that models uncertainty, reflection, and critical engagement.

24. Capability Over Time, Not Just at Publication

Capability does not develop instantly, and courses are rarely static.

A capability-aware approach recognises that courses may need to be revisited, updated, or reinterpreted as contexts change.

25. What Capability-Aware Design Does Not Mean

It is important to be clear about what capability-aware design does *not* imply.

It does not mean that a course must align to a particular framework.
It does not mean that capability can be measured automatically.
It does not mean that quality can be inferred from metadata alone.
It does not mean that responsibility can be delegated to a system.

Stage 2B Summary

By the end of this stage, you should understand capability-aware course design as a way of thinking about purpose, structure, judgement, and responsibility — not as a technical feature or compliance exercise.

STAGE 2C — HUMAN–AI ROLES AND BOUNDARIES

Clarifying responsibility in AI-supported course design

26. Why Roles and Boundaries Matter

When AI is used in course design, the most important question is often not *what the system can do*, but *who remains responsible* for what is produced.

In many AI-supported workflows, boundaries between human judgement and automated assistance are left implicit. As a result, responsibility can drift.

The Course Engine is designed to avoid this ambiguity. It treats **role clarity as a design requirement**, not as an afterthought.

27. The Human Role: Ownership and Judgement

Within the Course Engine, humans retain ownership of all substantive decisions.

This includes decisions about purpose and scope, emphasis and prioritisation, pedagogical appropriateness, ethical framing and implications, and suitability for a particular audience or context.

28. The AI Role: Assistance, Not Authority

AI within the Course Engine plays a **supporting role**.

It can assist by drafting, exploring alternative explanations, supporting early sense-making, and accelerating routine drafting tasks.

AI does not approve, reject, or certify quality. Outputs remain provisional and subject to human review.

29. Avoiding the Illusion of Automation

A key risk in AI-supported design is the **illusion of automation** — the sense that because something has been generated smoothly or quickly, it must be correct, complete, or authoritative.

The Course Engine resists this by preserving explicit structure, explicit decisions, and inspectable outputs.

30. Boundaries as a Form of Protection

Clear human–AI boundaries protect learners, institutions, and the integrity of the design process.

For learners, boundaries help ensure that content reflects considered judgement rather than unexamined automation. For institutions, they make it easier to explain how courses were designed and who is accountable.

31. Responsibility Does Not Transfer to the System

Using AI assistance does not transfer responsibility to the system.

Even when AI contributes to drafting, the human user remains responsible for what is included, how it is framed, what is omitted, and how it may be interpreted.

32. Transparency Over Hidden Intelligence

The Course Engine prioritises transparency over "seamless intelligence".

This transparency allows users and reviewers to understand how outputs were produced, revisit earlier decisions, explain design choices to others, and adapt work without reverse-engineering it.

33. Human–AI Collaboration as an Ongoing Practice

Human–AI collaboration is an ongoing practice that develops over time.

There is no single "correct" balance. What matters is that the balance is **intentional, explicit, and revisitable**.

34. What Clear Boundaries Enable

Clear boundaries support reflective practice, collaboration, governance, and assurance — because accountability is traceable rather than assumed.

Stage 2C Summary

By the end of this stage, you should have a clear understanding of how human and AI roles are defined within the Course Engine, why these boundaries exist, and how they protect judgement, accountability, and trust.

STAGE 3 — WHAT THE COURSE ENGINE CAN DO

Capabilities that support responsible, reviewable course design

35. From Principles to Practical Capability

By this point, the design principles behind the Course Engine should be clear: intent before automation, structure before scale, and human judgement before optimisation.

This stage explains **what the Course Engine can actually do**, in practical terms, and how those capabilities support responsible course production rather than replacing professional decision-making.

36. Using a Single Source of Truth for Course Design

At the centre of the Course Engine is the idea of a **single source of truth** for a course: `course.yml`.

Rather than scattering course decisions across documents, tools, and interfaces, the Course Engine works from a single, explicit course specification that can be reviewed, updated, and reused.

37. Validating Structure and Assumptions

The Course Engine supports **validation of structure and metadata**, helping to surface missing elements, inconsistencies, or unclear relationships between parts of the course.

From v1.6, validation also includes explicit **fail-fast guardrails** around lesson definitions:

- A lesson must define **either** `source` **or** `content_blocks`.
- This prevents ambiguous authoring and makes errors actionable earlier.

38. Generating Consistent, Publishable Outputs

The Course Engine generates **consistent, publishable learning artefacts** from a single course specification.

Rather than manually recreating the same course in multiple formats, the engine compiles a Quarto project and can then render it to HTML (and other Quarto outputs where configured).

38A. Supporting External Lesson Authoring (v1.6)

From v1.6, the Course Engine supports authoring lesson content as **external source files**, rather than requiring all lesson text to be embedded directly in `course.yml`.

This supports real-world maintenance where lesson files may be revised, reviewed, reused, or versioned independently over time.

39. Producing Auditable, Inspectable Metadata

Alongside human-readable outputs, the Course Engine produces **machine-readable metadata** in `manifest.json`.

This supports inspection, reporting, and audit without requiring users to understand internal code.

The engine also provides a **separate explainability interface** for inspecting inputs, resolution, and provenance **without making quality or compliance claims**.

From v1.6, the manifest can include:

- **framework_alignment** (declared)
- **design_intent** (v1.12+; rationale/boundaries summary + hash for auditability)
- **lesson_sources** (provenance summary)
- **render** (captured when rendering is run)

39A. Explainability for Governance and CI (v1.8+, stabilised v1.10.0)

The Course Engine includes an **explainability interface** intended for governance, QA, and automation contexts.

From **v1.8+**, explain outputs support structured review of both declared inputs and generated artefacts. In **v1.10.0**, the explain interface was **stabilised as a contract**: output semantics and format selection are explicit, deterministic (timestamps aside), and safe to rely on in CI and audit workflows.

Explain outputs are **descriptive, not evaluative**: they are designed to support inspection and review without asserting pedagogical quality, compliance, or approval.

39B. Capturing Design Intent as a Governance Signal (v1.12+)

From **v1.12**, the Course Engine supports an optional `design_intent` block in `course.yml`.

This is intended to capture the *why* behind a course, without turning that rationale into enforcement.

Design intent can include:

- a concise design rationale and positioning (what this course is trying to achieve)
- how AI is framed, used, or constrained (including decision boundaries)
- relevant frameworks, policy context, or intended review audience
- expectations for revision, stewardship, or review cadence

When present:

- it is recorded in `manifest.json` with a stable hash (supporting auditability and change detection)
- it is surfaced via `course-engine explain` (JSON and human-readable text)
- it remains **informational**: it is not validated as "correct", and it does not imply quality, compliance, or approval

This feature exists to support **transparent governance conversations**: it makes intent explicit, inspectable, and reviewable — while keeping responsibility with the human author.

40. Supporting Framework Alignment and Capability Mapping

The Course Engine supports two related layers of "framework-aware" authoring:

A. framework_alignment (v1.6)

- Declares the framework name and which domains the course references.
- Informational only.
- Shows up in `inspect` and `report`.
- Useful even when you are not ready to do structured mapping.

B. capability_mapping (v1.1+)

- Structured coverage and evidence metadata.
- Enables capability coverage reporting.
- Enables `validate` (policy/rule checking) when present.

41. Enabling Coverage Reporting and Review

Where capability mapping exists, the engine can produce a structured view of coverage and gaps.

Where capability mapping is absent but framework alignment exists (v1.6+), `report` will still produce a **declared alignment summary** rather than failing.

42. Supporting Validation Without Replacing Judgement

Validation mechanisms can check declared evidence/coverage against rules.

This is intentionally limited: validation is about defensibility and traceability, not pedagogical quality.

In v1.6, **validation requires capability mapping**. If only framework alignment exists, validation will explain that mapping is required and suggest using `report` instead.

43. Allowing Inspection and Explanation Without Building Outputs

Some modes allow inspection and explanation without full validation execution.

For example, policy resolution can be explained without requiring a built manifest (depending on command options).

44. Non-Destructive, Reversible Workflows

By default, builds do not overwrite existing outputs unless explicitly requested.

This protects earlier versions and supports cautious iteration.

45. What the Course Engine Does Not Do

The Course Engine does not:

- evaluate pedagogical quality
- judge academic merit
- determine learner effectiveness
- make approval or compliance decisions

These are deliberate exclusions that preserve human judgement.

Stage 3 Summary

By the end of this stage, you should have a clear picture of what the Course Engine can do and how its capabilities support responsible, reviewable course design.

STAGE 4 — USING THE COURSE ENGINE IN PRACTICE

Applying the system in real design, review, and governance contexts

46. From Capability to Day-to-Day Use

This stage describes common ways the Course Engine is used by educators, learning designers, and institutions.

These are not rigid workflows. They are typical patterns of use that illustrate how the system supports thoughtful, accountable course design in different contexts.

47. Designing a New Course

When designing a new course, the Course Engine is most effective when introduced early.

Users typically begin by clarifying purpose, audience, and scope, then translating intent into explicit structure. AI assistance can support drafting, but always within the structure that has already been decided.

48. Revising and Updating an Existing Course

Because structure remains explicit, revisions can be deliberate rather than ad hoc.

This supports defensibility under review and reduces the likelihood of "drift" over time.

49. Collaborative Course Design

The Course Engine supports collaboration by making course structure and intent visible in a shared, inspectable specification.

It becomes easier to discuss decisions, review sections, and document change rationales.

50. Supporting Review and Quality Assurance

In QA and governance contexts, the Course Engine is often used to support **inspection rather than production**.

Reviewers can inspect outputs and metadata, without the system claiming quality or compliance.

From **v1.12**, reviewers can also inspect declared **design intent** as part of the artefact metadata. This supports governance conversations about rationale, AI positioning, and decision boundaries without requiring reviewers to infer intent from content alone.

51. Using Framework and Capability Information in Practice

- Use **framework alignment** to declare intent and context.
- Use **capability mapping** when you want structured coverage/evidence and validation.

This supports conversation and reflection without collapsing judgement into automation.

52. Integrating the Course Engine Into Existing Workflows

The Course Engine is designed to complement existing tools and processes rather than replace them.

Outputs can be used in multiple contexts without requiring the Course Engine itself to become the platform of record.

53. Experimentation and Low-Risk Prototyping

Non-destructive workflows make the tool suitable for low-risk prototyping: you can iterate, compare builds, and refine structure without losing earlier versions.

54. Using the Course Engine Over Time

The Course Engine supports stewardship: courses can be revisited as contexts change, AI capabilities evolve, or institutional expectations shift.

55. What Good Practice Looks Like in Use

Good practice tends to share these characteristics:

- intent stays visible
- structure is revised deliberately
- AI assistance remains provisional
- inspection and reporting are used to support review
- publication is treated as "published for now", not "finished"

Stage 4 Summary

By the end of this stage, you should have a sense of how the Course Engine is used in real settings: designing new courses, revising existing ones, collaborating, supporting review, and stewarding learning materials over time.

STAGE 5 — LIMITS, RISKS, AND RESPONSIBLE USE

Understanding what the Course Engine cannot do — and how to use it well

56. Why Limits Matter

Responsible use begins with understanding limits.

The Course Engine is intentionally powerful in some areas and intentionally restrained in others. These limits are deliberate design choices to protect judgement, accountability, and trust.

57. The Course Engine Does Not Judge Quality

The Course Engine does not evaluate pedagogical quality, academic merit, or educational effectiveness.

Structured output is not the same as good pedagogy. Quality remains a matter of professional judgement and review.

58. The Course Engine Does Not Decide What Should Be Taught

It does not decide content scope, framing, or priorities.

Those decisions involve values, disciplinary norms, and ethical considerations.

59. Framework and Capability Information Is Not a Verdict

Framework alignment is *declarative*.

Capability mapping (and reports/validation) is *informational evidence*.

Neither is a substitute for judgement, peer review, or institutional approval.

60. Automation Can Create False Confidence

Fluent, structured outputs can create a false sense of certainty.

Users must remain critical, reflective, and willing to revise.

61. Responsibility Does Not Shift to the Tool

Even with AI assistance, responsibility for accuracy, appropriateness, and impact remains human-owned.

62. Risks of Over-Enforcement and Misuse

Over-enforcement can reduce complex judgement to box-ticking.

Responsible use means choosing when to apply checks and when to rely on human review.

63. Transparency Requires Interpretation

Visibility does not remove the need for interpretation.

Reviewers must still ask critical questions and consider context.

64. Ethical and Contextual Judgement Remain Essential

The Course Engine cannot resolve cultural, ethical, or contextual appropriateness.

Users must apply professional judgement and domain expertise.

65. Designing for Review, Not Avoiding It

The Course Engine should be used to invite review, support discussion, and document decisions — not to avoid scrutiny.

66. Principles for Responsible Use

- Treat AI assistance as provisional
- Keep key decisions explicit and revisitable
- Use reporting and validation to support reflection
- Combine outputs with professional judgement and peer review
- Remain attentive to context, ethics, and unintended consequences

Stage 5 Summary

By the end of this stage, you should have a clear understanding of the limits of the Course Engine, the risks associated with misuse or over-reliance, and the responsibilities that remain human-owned.

CLOSING — USING THE COURSE ENGINE AS INTENDED

Responsibility, continuity, and confidence over time

Bringing the Pieces Together

This handbook has described the Course Engine as **infrastructure for responsible course design**.

Across the preceding stages, the emphasis has been on intent before generation, structure before scale, and judgement before optimisation. The Course Engine is designed to support these priorities by making design decisions explicit, reviewable, and revisitable.

The Course Engine in the CloudPedagogy Ecosystem

The Course Engine sits within a wider ecosystem that includes capability frameworks, reflective tools, and design methods.

Its role is not to define capability or determine governance decisions, but to produce artefacts that can be examined, discussed, and improved over time.

Using the Course Engine Over Time

The Course Engine is most effective when used **over time**.

Courses are expected to be revisited as contexts change, feedback is gathered, and expectations evolve.

Confidence Without Overreach

The Course Engine supports confidence through structure, auditability, and transparency — without claiming authority over quality, ethics, or effectiveness.

A Tool That Assumes Professional Judgement

The Course Engine assumes professional judgement: it is designed to support careful practice, not replace it.

This Handbook as a Living Document

Core principles and boundaries are intended to remain stable.

Workflow descriptions, examples, and feature-specific guidance will expand as the tool evolves.

Appendix — Getting Started Without Technical Overload

Purpose of This Appendix

This appendix helps readers take **practical first steps** with the Course Engine without turning the handbook into a command reference.

Detailed installation and command usage are documented separately in the repository documentation.

Step 1: Start With a Small, Real Use Case

Choose a small, concrete use case such as a short workshop, a single module, or a guidance document.

Starting small protects intent and coherence.

Step 2: Clarify Intent Before Writing

Ask:

- Who is this for?
- What should readers be able to do differently?
- What is in scope and out of scope?

Step 3: Sketch Structure First

Sketch major sections and progression before generating substantial text.

Step 4: Use AI Assistance Deliberately

Use AI for drafting, alternative phrasings, and clarification — not for making value-laden decisions.

Step 5: Review Early and Often

Use iteration intentionally. Do not wait until everything is written before reviewing structure and intent.

Step 6: Use Inspection and Reporting Thoughtfully

When appropriate:

- use **inspect** to understand what was built and what metadata was captured
- use **report** to surface framework alignment and (when present) capability mapping coverage
- use **validate** only when structured capability mapping exists and you need rule-based defensibility checks

Step 7: Publish as a Moment, Not an Endpoint

Treat publication as "published for now". Maintain the ability to revise, review, and improve over time.

Where to Find Technical Guidance

Detailed technical instructions — installation, prerequisites, CLI usage, and configuration — are intentionally kept out of this handbook.

Refer to the repository documentation, including:

- `README.md`

- `docs/END_USER_INSTRUCTIONS.md`
- `docs/POLICY_FILES.md`
- and any version-specific design notes (e.g., `docs/design/v1.6.0-design.md`, `docs/design/v1.12.0-design.md`)

Appendix — Licence and Use