cyber-wyse

// COMPANION TOOLKIT · ARTICLE 15

The Producer's Co-Pilot

Every prompt, template and checklist for running real project-management work with AI — the WBS, the Gantt, the budget, resourcing, risk and status — while keeping the judgement firmly human. Copy, paste, adapt.

DISCIPLINE	TOOLING	VERSION	PAIRS WITH
Production / PM	Any chat LLM	v1.0 · Jun 2026	work-ai-pm

// HOW TO USE THIS TOOLKIT

One principle, six artefacts

The whole toolkit rests on a single rule: the model drafts, you decide. Every prompt here produces a first draft of an artefact in seconds; your time goes on editing the judgement calls, not formatting cells. Anything with a number a stakeholder will act on — a date, a cost, a capacity figure — gets verified by you before it leaves the building.

> // THE ONE HABIT THAT MATTERS MOST
>
> Always ask the model for structured output — a pipe-delimited or CSV table, never a bulleted essay. The same rows then paste into a sheet, feed the next prompt, and import into Jira, Asana or Monday with no retyping. Prose is a dead end; data flows.

// WHAT'S INSIDE

01
Brief → WBS
Turn a messy brief into a structured work breakdown you prune instead of build.

02
Tasks → Gantt
Sequence and date the work; get a render-ready Mermaid chart back.

03
Schedule → Budget
Cost the plan with contingency, then flex it into comparable scenarios.

04
Resourcing
Build an allocation grid and surface every over-100% week before it bites.

05
Risk register
Derive the register from the plan itself — dependencies are risks.

06
Status & comms
One set of facts, three audiences: client, exec, team.

07
Templates
Copy-paste skeletons incl. a formula-based budget sheet.

08
Checklists
The verify-before-send list and the human-in-the-loop line.

> // A NOTE ON TOOLS
>
> Nothing here needs a special platform. Every prompt works in any chat-style LLM, and every template targets a plain spreadsheet. Mermaid charts render in most modern editors and on the cyber-wyse article pages. The point is portability — no lock-in, low operational overhead.

// PLAYBOOK 01

Brief → work breakdown

The blank WBS is the most expensive blank page in the job — everything downstream inherits its structure. Hand the model the raw brief plus your house phases and ask for structured rows, not prose.

// THE MESSY BRIEF THAT GOES IN

"We need to launch the new customer portal before the autumn campaign. It should let people log in, see their orders, and update payment details. Marketing wants it looking on-brand. There's some integration with the existing billing system. Oh, and legal need to sign off on the data handling. Can we get this done in about two months?"

PROMPT — BRIEF TO STRUCTURED WBS

```
Role: You are a senior producer building a work breakdown.

Input: the brief below, plus our standard phases
  (Discovery, Design, Build, QA, Launch, Hypercare).

Task: Decompose into phase > deliverable > task.
For each task output a row:
  phase | deliverable | task | rough_effort_days | depends_on | owner_role

Rules:
- Effort is a rough order of magnitude, not a commitment.
- Mark anything you inferred (not stated in the brief) with [ASSUMED].
- List open questions separately at the end. Do not invent dates.
- Output as a pipe-delimited table and nothing else.

Brief:
"""<the brief above>"""
```

↓ WHAT THE MODEL RETURNS — POPULATED WBS

PHASE	DELIVERABLE	TASK	DAYS	DEPENDS ON	OWNER
Discovery	Research	Stakeholder interviews	5	—	PM
Discovery	Research	Requirements synthesis	3	interviews	PM
Design	UX/UI	Concept directions	8	synthesis	Design
Design	UX/UI	Brand alignment review [ASSUMED]	2	concepts	Design
Build	Portal	Auth + login	6	concepts	Eng
Build	Portal	Order history view	5	auth	Eng
Build	Portal	Payment details update	5	auth	Eng
Build	Integration	Billing system link	6	payment	Eng
QA	Test	Test & fix cycle	10	integration	QA
Launch	Go-live	Legal data sign-off	3	QA	Legal

PHASE	DELIVERABLE	TASK	DAYS	DEPENDS ON	OWNER
Launch	Go-live	Launch + hypercare	5	sign-off	PM

```
open questions: 1) which billing system / does it have an API? 2) brand guidelines available or to be created?
3) is "two months" a hard date tied to the campaign launch?
```

```
// THE TWO DETAILS THAT CARRY IT
```

The `[ASSUMED]` flags are your edit list — the model inferred a brand-review task nobody asked for. And the open questions become your kickoff agenda instead of silently-wrong tasks.

// PLAYBOOK 02

Tasks → Gantt & timeline

With effort and dependencies in hand, the schedule is arithmetic over a calendar. Give the model real constraints and ask for dated rows plus a chart. It returns three things: a dated schedule table, a render-ready Mermaid source, and — once rendered — the timeline below.

PROMPT — SCHEDULE THE TASKS

```
Task: From the WBS task rows, build a schedule.

Constraints:
- Start 2026-06-15. Mon–Fri working days. UK bank holidays off.
- Respect every depends_on. Flag any circular dependency.
- Hard milestone: client review on/ before 2026-08-07.

Output: 1) table task | start | end | duration | depends_on
        2) a Mermaid gantt chart of the same schedule.

If the milestone cannot be met, say so explicitly and show
the critical path that breaks it — do not silently compress tasks.
```

↓ OUTPUT 1 — DATED SCHEDULE

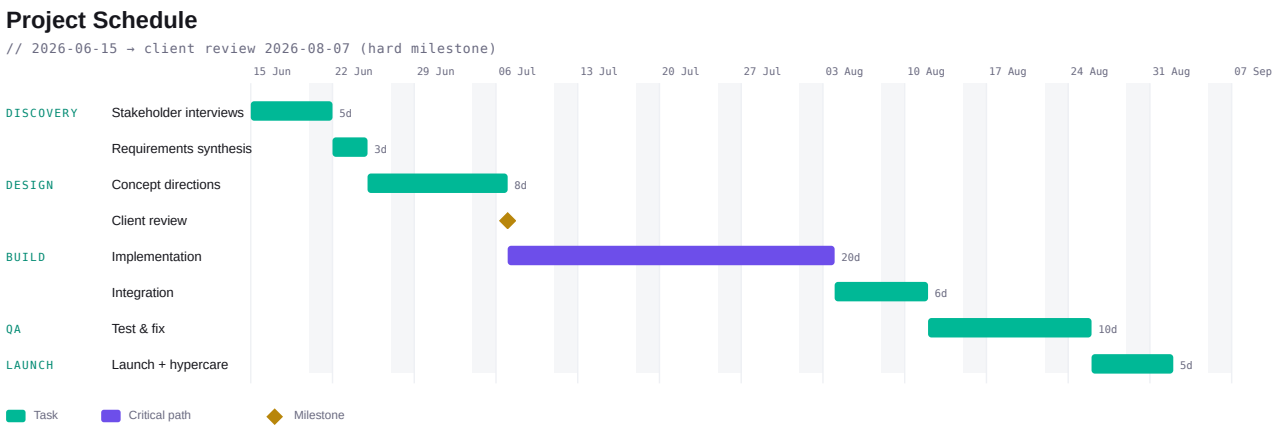
TASK	START	END	DUR	DEPENDS ON
Stakeholder interviews	15 Jun	19 Jun	5d	—
Requirements synthesis	22 Jun	24 Jun	3d	interviews
Concept directions	25 Jun	06 Jul	8d	synthesis
Client review (milestone)	07 Jul	07 Jul	0d	concepts
Implementation	07 Jul	03 Aug	20d	concepts
Integration	04 Aug	11 Aug	6d	implementation

milestone check: client review lands 07 Jul — well inside the 07 Aug hard date. ✓ met

// THE INSTRUCTION THAT PREVENTS A LIE

Left to please you, a model quietly shrinks durations to hit the date you wanted. "Refuse and show the critical path instead" turns it from a yes-man into a scheduler that reports bad news — the only kind worth having.

→ Output 2: the rendered timeline



// the critical path (purple) is the Implementation run — the task with no slack against the milestone

→ Output 3: the Mermaid source behind it

GANTT.MMD — RENDER-READY, EDIT AND RE-RENDER

```
gantt
  title Project Schedule
  dateFormat YYYY-MM-DD
  section Discovery
    Stakeholder interviews   :a1, 2026-06-15, 5d
    Requirements synthesis   :a2, after a1, 3d
  section Design
    Concept directions       :b1, after a2, 8d
    Client review            :milestone, m1, after b1, 0d
  section Build
    Implementation           :crit, c1, after b1, 20d
    Integration              :c2, after c1, 6d
  section QA
    Test & fix               :d1, after c2, 10d
```

// ROUND-TRIP IT

Mermaid is text, so you can hand the source back to the model after a slip — "Implementation ran 4 days long, reflow everything downstream and tell me if the milestone still holds" — and get an updated chart plus the impact in one move.

// PLAYBOOK 03

Schedule → budget & scenarios

A budget is the schedule seen through money: roles × rates × effort, plus contingency, plus non-labour. The model is excellent at the assembly and the re-runs — the "what if we drop a region?" you used to rebuild by hand.

PROMPT — COST THE PLAN, THEN FLEX IT

```
Inputs:
- Task rows with owner_role and effort_days (below).
- Rate card: PM £650/day, Design £700, Eng £750, QA £500.
- Contingency: 15% on Build & QA, 10% elsewhere.
- Non-labour: licences £2,400, testing incentives £1,500.

Task: Budget table grouped by phase:
  phase | role | days | rate | line_cost
then subtotal, contingency, non-labour, grand total.
Show the arithmetic — I need to audit each line.

Then three scenarios as a comparison table:
  Baseline | Lean (−1 region) | Plus (added hypercare)
with grand total and the key trade-off for each.
```

→ Worked example: phase budget

PHASE	ROLE	DAYS	RATE	LINE COST
Discovery	PM	8	£650	£5,200
Design	Design	14	£700	£9,800
Build	Eng	30	£750	£22,500
QA	QA	10	£500	£5,000
Subtotal (labour)				£42,500
Contingency (blended)				£5,475
Non-labour				£3,900
Grand total				£51,875

→ Scenario comparison

SCENARIO	GRAND TOTAL	KEY TRADE-OFF
Baseline	£51,875	Full scope, both regions.
Lean (−1 region)	£41,200	~21% cheaper; second market deferred, not delivered.
Plus (hypercare)	£58,600	+1 month support; lower post-launch risk, higher cash now.

// VERIFY THE MATHS — DO NOT SKIP THIS

Models are confident arithmeticians and occasionally wrong ones. For anything client-facing, have the model output the budget as a sheet with live formulas, not pre-computed numbers — so totals are computed by the spreadsheet, not the model. Template on the next page.

// TEMPLATE

Formula-based budget sheet

Ask the model to fill columns A–D only, and to leave E and all totals as formulas. The spreadsheet does the arithmetic, so a wrong number from the model can't reach the client.

BUDGET-TEMPLATE — PASTE INTO A SHEET, MODEL FILLS A–D

```
     A          B         C       D          E (formula — do not type a number)
1    Phase      Role      Days    Rate       Line cost
2    Discovery  PM        8       650        =C2*D2
3    Design     Design    14      700        =C3*D3
4    Build      Eng       30      750        =C4*D4
5    QA         QA        10      500        =C5*D5
6
7    Subtotal                                =SUM(E2:E5)
8    Contingency  (blended %)                =E7*0.129
9    Non-labour                              3900
10   GRAND TOTAL                             =E7+E8+E9
```

// PROMPT TO ENFORCE IT

"Output the budget as CSV. Columns A–D are values; column E and all totals must be spreadsheet formulas as literal text (e.g. `=C2*D2`), never computed numbers. I will paste this into Sheets and the totals must calculate themselves."

→ Rate-card block (keep alongside)

ROLE	DAY RATE	NOTES
Producer / PM	£650	Edit to your real card before each estimate.
Design	£700	Blend snr/jnr if needed.
Engineering	£750	—
QA	£500	—

// PLAYBOOK 04

Resourcing & hidden over-allocation

The schedule says the work fits. The resource view often says it doesn't — because one person sits on the critical path of three things in the same week. A model catches this instantly; it's just overlapping bookings per person per period.

PROMPT — FIND AND FIX OVER-ALLOCATION

```
From the scheduled tasks (task | owner | start | end | effort_days):

1) Build a weekly allocation grid: rows = people, cols = ISO weeks,
   cells = % allocated (assume 5-day weeks, effort spread evenly).
2) Flag every cell > 100% separately: who, which week, what clashes.
3) For each clash propose 2 levelling options — shift a non-critical
   task, or split effort — and name the cost of each (slip, extra spend).
   Recommend nothing yet. Just lay out the choices.
```

→ Worked example: allocation grid

PERSON	W24	W25	W26	W27
Producer	60%	80%	80%	40%
Designer	100%	**140%**	**120%**	60%
Engineer	40%	90%	100%	100%

// red cells = over capacity. W25–W26 designer is the clash to resolve.

// WHY "RECOMMEND NOTHING YET"

Levelling trades slip against cost against who's already had a brutal month — context the model doesn't have. Its job is to surface the clash and frame the options cleanly. The call is yours.

// PLAYBOOK 05

The plan is the risk register

The best first-pass register is hiding inside the plan. Every dependency is a slip risk. Every `[ASSUMED]` tag is an assumption risk. Every 100%-loaded person is a single-point-of-failure. Don't brainstorm from nothing — read the risks out of the artefacts.

PROMPT — DERIVE THE REGISTER FROM THE PLAN

```
Read the WBS, schedule and resource grid. Produce a register:
  id | risk | source | likelihood(1-5) | impact(1-5) | score | mitigation | owner

Derive risks from the artefacts themselves:
- every [ASSUMED] task → an assumption risk
- every long dependency chain → a cascade risk
- every >90% allocated person → a single-point-of-failure risk
- the milestone with least slack → a schedule risk

Score = likelihood × impact. Sort high to low.
Concrete mitigations only — no "monitor closely".
```

→ Worked example: register

ID	RISK	L	I	SCORE	MITIGATION
R1	Designer over-booked W25–26	4	4	**16**	Pre-book backup designer for those two weeks now.
R2	Third-party API docs thin (assumed complete)	3	4	12	Spike against the API in W24; escalate if blocked by W25.
R3	Client review milestone has zero slack	3	3	9	Pre-circulate concepts 3 days early for async sign-off.

// BAN THE DEAD VERB

Forbidding "monitor closely" matters more than it looks. Registers rot into a column of meaningless verbs. Forcing concrete mitigations — "pre-book a backup designer for W25–26" — makes the register a thing you act on, not a thing you present.

// PLAYBOOK 06

One truth, three audiences

Status is where producers lose hours to translation. The client wants reassurance and the headline; the exec wants one line and the number; the team wants specifics and blockers. Same facts, three registers — which a model does almost too well.

PROMPT — ONE UPDATE, THREE REGISTERS

```
Facts (single source of truth):
- Design complete, signed off 2 days early.
- Build 60% done. Risk: third-party API docs are thin.
- Need client decision on payment provider by Fri or Build slips a week.

Produce three versions, no new facts:
1) Client email — warm, confident, one clear ask (the Fri decision).
2) Exec one-liner — RAG status + the single number that matters.
3) Team stand-up note — blunt, specific, blockers first.

Keep each self-contained. Do not soften the Friday deadline anywhere.
```

OUTPUT 1 — CLIENT EMAIL

Subject: Portal update — great progress, one quick decision needed

Hi Sarah,
Good news this week: design is complete and signed off two days ahead of plan, and the build is now 60% of the way there. We're on track.

One thing I need from you to keep that momentum: a decision on the payment provider by Friday. If it lands by then we stay on schedule; if it slips past Friday, the build slips by about a week. Happy to jump on a quick call if that helps.

OUTPUT 2 — EXEC ONE-LINER

Portal AMBER — build 60% done, on track *if* client confirms payment provider by Fri; one-week slip if not.

OUTPUT 3 — TEAM STAND-UP NOTE

Blocker: need payment-provider decision from client by Fri or Build slips a week — chasing today.
Risk: third-party API docs are thin; flagging now, may need a spike.
Done: design signed off (2 days early). In flight: build at 60%.

// THE GUARDRAIL

"No new facts." Notice all three say the same true things — 60%, Friday, the API risk — in three tones. A model asked for a reassuring client email will invent reassuring detail unprompted; pinning it to the fact list keeps every version honest.

→ Template: status fact-block

Write this once, plainly, before every update. Then let the model translate — never the other way around.

STATUS-FACTS.TXT

```
PERIOD:    <week ending>
DONE:      <what shipped / signed off>
IN FLIGHT: <what's mid-build, with % if known>
RISKS:     <the one or two that matter>
ASKS:      <decisions needed, with deadline + consequence>
RAG:       <Green | Amber | Red> — one-line why
```

// REFERENCE

Prompt library — quick grabs

Short, single-purpose prompts for the small jobs that come up mid-project. Paste, fill the angle brackets, go.

Meeting transcript → actions

ACTIONS.TXT

```
From this transcript, extract a table: action | owner | due | depends_on.
Only items someone actually committed to — not topics discussed.
List anything ambiguous (no clear owner) separately as "needs assignment".
```

Scope-creep check

CREEP.TXT

```
Here is the signed scope, and here is the latest client request.
Is the request inside scope or outside it? If outside, draft a one-paragraph
change note: what's new, rough effort, and the schedule/cost impact.
```

Estimate sanity-check

SANITY.TXT

```
Here's my estimate (tasks + days). Play sceptical senior reviewer:
where am I likely optimistic? Which tasks usually take 2× what's shown?
What's missing that always gets forgotten (handover, env setup, review cycles)?
```

Dependency / critical-path finder

CRITPATH.TXT

```
From these task rows with depends_on, identify the critical path
(longest chain by duration). List it in order, with total length in days,
and flag the 3 tasks where a slip does the most damage.
```

Retro synthesis

RETRO.TXT

```
Here are raw retro notes from the team. Cluster into themes.
For each theme: what happened, why it likely happened, and ONE concrete
change for next time. No platitudes — each change must be testable.
```

// GUARDRAILS

The two checklists that keep it safe

A Verify-before-send

Run this on anything AI-drafted before it reaches a stakeholder.

- [ ] **Every date checked** against the real calendar — holidays, leave, frozen periods.
- [ ] **Every total re-computed** by a spreadsheet formula, not trusted from the model's prose.
- [ ] **Every** `[ASSUMED]` **resolved** — confirmed with someone, or carried into the risk register.
- [ ] **Capacity figures sane** — no one silently booked over 100%.
- [ ] **No invented facts** in any client-facing comms — cross-check against your fact-block.
- [ ] **Names & rates current** — the model used last project's rate card unless you replaced it.

B The human-in-the-loop line

The model never owns these. If you catch it deciding rather than drafting, stop.

- [ ] Whether the scope is the right scope.
- [ ] Whether an estimate is one you'll actually commit to.
- [ ] Who can really do the work — given everything you know about the people.
- [ ] Which risk is the one that matters this week.
- [ ] What you say when a project is genuinely in trouble.

> // THE WHOLE POINT
>
> Everything here clears the mechanical work out of the way so you have more attention for the judgement, not less. Used like this, AI doesn't make you a lighter-touch producer — it makes you a faster one with more room to think.