

PACEDALL **ACADEMY**

PRACTICAL FIELD GUIDE

Best Prompts for Platform Builders

Seven copy-ready prompt patterns for product discovery, architecture, security, evaluation and implementation quality.

BUILT FOR

Builders who want repeatable, evidence-led assistance from language models.

INSIDE

The CLEAR framework plus ready-to-adapt prompts, guardrails, checklists and decision rules.

How to use this guide

Replace every bracketed field with real context. Treat outputs as draft analysis, verify consequential claims and keep a record of the prompt and model version used.

Contents

- 01 The CLEAR prompt framework
- 02 Opportunity assessment
- 03 Product brief
- 04 Architecture decision review
- 05 Threat modelling
- 06 Prompt and output evaluation
- 07 Implementation review

Pacedall principle

A useful AI platform is a complete service: product, workflow, data, safeguards, operations and commercial value. The model is only one component.

The CLEAR prompt framework

Give the model enough structure to succeed.

CLEAR stands for Context, Lens, Expected outcome, Arrangement and Rules. Add examples and evaluation criteria when consistency matters.

Separate stable instructions from user data. Ask for structured output when another system will consume the result.

Practical checklist

- Context: relevant situation and inputs
- Lens: role or point of view
- Expected outcome: decision or artefact
- Arrangement: structure and format
- Rules: constraints, exclusions and quality bar

Copy-ready template

```
Context: [relevant background]
Lens: Act as [specific expertise].
Expected outcome: Produce [decision or artefact].
Arrangement: Return [sections / schema / length].
Rules: [constraints, evidence, tone, exclusions].
Before answering, identify missing critical information.
```

Decision rule

A longer prompt is not automatically better. Every instruction should change the expected result.

Opportunity assessment

Test whether an idea deserves to become a platform.

Use this before choosing features. It forces attention onto the user, current alternatives, frequency, consequence and defensibility.

Practical checklist

- Replace assumptions with explicit unknowns
- Distinguish user value from novelty
- Identify the smallest validation test

Copy-ready template

Act as a sceptical product strategist.
 Assess this AI platform idea: [IDEA].
 User: [USER]. Current workaround: [WORKAROUND].
 Evaluate problem severity, frequency, willingness to change,
 prompt-replacement risk, data advantage, workflow advantage,
 distribution and revenue.
 Return: scorecard, top assumptions, kill criteria and
 a 14-day validation plan. Do not invent evidence.

Decision rule

Do not use model enthusiasm as evidence of market demand.

Product brief

Turn a validated problem into a bounded build.

A good brief aligns product, design and engineering around the intended outcome and the deliberate exclusions.

Practical checklist

- State the target outcome
- Name non-goals
- Define measurable acceptance criteria
- Expose risks and dependencies

Copy-ready template

Create a product brief for [PRODUCT].
 Inputs: [RESEARCH / CONSTRAINTS].
 Include problem, primary user, job to be done, desired outcome,
 current journey, proposed journey, MVP scope, non-goals,
 functional requirements, trust requirements, metrics, risks,
 open questions and launch criteria.
 Mark every unsupported claim as an assumption.

Decision rule

Scope should be the smallest coherent service, not the smallest collection of screens.

Architecture decision review

Use AI to challenge a design, not rubber-stamp it.

Provide known scale, data sensitivity, team capability, cost limits and operational constraints. Request trade-offs and failure modes.

Practical checklist

- Separate facts from assumptions
- Compare at least two viable options
- Include security and operability
- Record the reversal cost

Copy-ready template

Review this architecture decision: [DECISION].
 Context: [USERS, SCALE, DATA, TEAM, BUDGET, DEADLINE].
 Options considered: [OPTIONS].
 Analyse benefits, failure modes, privacy, security, latency, cost, operability, vendor lock-in and migration path.
 Return a recommendation, rejected alternatives, safeguards, validation tests and conditions that would reverse the decision.

Decision rule

The preferred architecture must be justified against the actual constraints.

Threat modelling

Identify abuse before launch.

Cover users, administrators, integrations, model inputs, retrieved data, tools and output handling. Keep mitigations actionable.

Practical checklist

- List protected assets
- Map trust boundaries
- Consider prompt injection and data leakage
- Prioritise by likelihood and impact

Copy-ready template

```
Threat-model [FEATURE / WORKFLOW].
Data handled: [DATA]. Actors and integrations: [ACTORS].
Map assets, entry points, trust boundaries and abuse cases.
Cover identity, permissions, prompt injection, data leakage,
unsafe tool use, denial of service and logging exposure.
Return a prioritised risk register with mitigations, owners,
verification steps and residual risk.
```

Decision rule

A threat model is useful only when each material risk has an owner and mitigation.

Prompt and output evaluation

Test performance with a repeatable set.

Define dimensions before judging examples. Include ordinary, ambiguous, adversarial and out-of-scope cases.

Practical checklist

- Use representative inputs
- Score correctness and usefulness separately
- Track refusals and unsafe compliance
- Compare versions blind where possible

Copy-ready template

Design an evaluation plan for this prompt: [PROMPT].
 Purpose: [PURPOSE]. Users: [USERS]. Risks: [RISKS].
 Create test categories, 20 representative cases, expected traits,
 pass/fail rules and a scoring rubric for correctness, relevance,
 completeness, groundedness, safety, tone, latency and cost.
 Include regression and adversarial cases.

Decision rule

Do not ship a prompt change without comparing it against the current baseline.

Implementation review

Ask for findings that engineers can act on.

Give the model the relevant code, architecture context and standards. Require references to exact locations and avoid broad rewrites without evidence.

Practical checklist

- Prioritise correctness and security
- Explain user impact
- Propose the smallest safe fix
- Request missing tests

Copy-ready template

Review the supplied implementation for [FEATURE].
Focus on correctness, security, privacy, accessibility, performance, maintainability and test coverage.
For each finding return severity, location, evidence, impact, smallest safe fix and a test that proves the fix.
Do not report style preferences as defects. If context is insufficient, ask a focused question rather than guessing.

Decision rule

A useful code review separates defects from preferences.

Keep building

Use this guide as a working artefact. Record decisions, test assumptions with real users and update the document as evidence changes.

Learn more at academy.pacedall.com