The AI Prompt Playbook

100 prompts, 5 categories, and the principles behind why they actually work

By @the.anadi

Before you use this

Most people get bad output from AI tools for one reason: they ask vague questions and expect specific answers. "Help me with my code" and "write me a caption" are requests, not prompts. A prompt is a request plus the context, constraints, and format that let the model actually do the job well.

Every good prompt in this document is built from the same four parts:

1. **Context** — what's actually going on (your situation, your level, your constraints)
2. **Task** — the specific thing you want done, not the general area
3. **Constraints** — what to prioritize, avoid, or rank by
4. **Format** — how you want the answer shaped, so you're not editing structure later

You'll see this pattern repeat 100 times below. That repetition is the point — once you internalize the shape, you stop needing a list like this at all.

> **How to use this document:** each prompt is copy-paste ready. Swap anything in `[brackets]` for your real situation. Every entry is tagged for who it's built for — **Professional**, **Student**, or **Both** — and includes one line on *why* it works, not just what it does. Read those lines. They're the actual lesson; the prompts are just the delivery mechanism.

Five categories, 20 prompts each:

1. Engineering & Coding
2. Learning Anything
3. Design (Product, UI/UX & Visual)
4. Content & Growth
5. AI Image, Video & Animation Generation

1 Engineering & Coding

treat the model like a senior collaborator you're handing context to, not an autocomplete engine. The quality gap between "fix this" and a well-scoped prompt is enormous, and it's almost entirely about how much real context you hand over up front.

1.1 Turn a rough idea into a spec

BOTH

```
I want to build [idea]. Ask me the 5 most important clarifying questions before proposing any solution — things like scope, edge cases, and what 'done' looks like.
```

Why it works — forces requirements-gathering before code, which is the step almost everyone skips.

1.2 Pre-mortem a build

PROFESSIONAL

```
I'm about to build [feature]. Play devil's advocate: what are the 3 most likely ways this breaks in production, and what assumption am I making that's probably wrong?
```

Why it works — surfaces risk while it's still cheap to fix.

1.3 Senior-level code review

BOTH

```
Review this code like a senior engineer doing a PR review, not a linter. Flag anything that'll bite me in 6 months — bad naming, hidden coupling, missing error handling. Be blunt. [paste code]
```

Why it works — pushes past syntax-checking into actual judgment.

1.4 Debug without over-explaining

BOTH

```
Here's an error and the relevant code. Before suggesting a fix, tell me what you think is actually happening, and ask if there's context you're missing. Error: [x] Code: [y]
```

Why it works — stops guessed fixes built on incomplete information.

1.5 Explain your own decision back to you

PROFESSIONAL

```
I just made this architecture decision: [decision]. Explain it back to me like you're teaching someone else. If my reasoning has a gap, point it out instead of agreeing.
```

Why it works — catches rationalizations disguised as reasoning.

1.6 Surface edge cases you missed

BOTH

```
List every edge case this function needs to handle that I probably haven't thought of. Rank them by how likely I am to have missed them. [paste code]
```

Why it works — a prioritized list beats an exhaustive, unranked one.

1.7 Refactor without breaking behavior

BOTH

```
Refactor this for readability only — no behavior changes, no new features. List exactly what changed and why so I can verify nothing broke. [paste code]
```

Why it works — keeps refactors auditable instead of a leap of faith.

1.8 Write the tests a rushed engineer skips

PROFESSIONAL

```
Write test cases for this function, prioritizing boundary values, null inputs, and one adversarial case. Skip the obvious happy path — I've got that. [paste code]
```

Why it works — targets the gaps, not the tests you'd write anyway.

1.9 Compress a long thread into decisions

PROFESSIONAL

```
Read this and extract only the decisions made, who owns what, and open questions. Skip the discussion. [paste doc/thread]
```

Why it works — turns meeting sprawl into an action log in seconds.

1.10 Sanity-check a plan before committing

BOTH

```
Here's my plan for [project]: [plan]. Where's the weakest assumption? If this fails, what's the most likely reason?
```

Why it works — one targeted weak point beats generic encouragement or generic criticism.

1.11 Turn a vague bug report into a repro

PROFESSIONAL

```
Here's a vague bug report: [report]. Write minimal repro steps, and list what info I'd need to ask the user for if this isn't enough.
```

Why it works — converts a complaint into an actionable ticket.

1.12 Explain code like a PR description

BOTH

```
Explain what this code does like you're writing the PR description for someone who's never seen it — what changed, why, what to watch for in review. [paste code]
```

Why it works — forces plain-language clarity before you ship, not after someone asks.

1.13 First-project architecture check

STUDENT

```
I'm building my first [type of project] for a class/portfolio. Here's my planned structure: [structure]. What would a professional flag as over-engineered, and what am I under-engineering?
```

Why it works — calibrates a student's instinct for scope, which usually skews in one direction or the other.

1.14 Learn a new language fast by translating

STUDENT

```
I know [language A] well. Translate this snippet into [language B] line by line, explaining the idiomatic difference at each step, not just the syntax swap. [paste code]
```

Why it works — builds real fluency instead of copy-paste familiarity.

1.15 Simulate a technical interview

STUDENT

```
Run me through a technical interview for a [role] position. Ask one question at a time, wait for my answer, then critique it like a real interviewer would before moving to the next.
```

Why it works — practice under realistic pressure beats reading interview guides.

1.16 Diagnose a slow query or function

PROFESSIONAL

```
Here's a slow [query/function] and its execution context. Identify the most likely bottleneck before suggesting an optimization, and tell me how you'd confirm it. [paste code + context]
```

Why it works — diagnosis before treatment — the same discipline as debugging.

1.17 Design an API before building it

PROFESSIONAL

```
I need an API for [use case]. Propose the endpoints, request/response shapes, and error cases — then tell me the one design decision most likely to need reworking later.
```

Why it works — front-loads the decisions that are expensive to change after the fact.

1.18 Explain an error like I'm five commits behind

STUDENT

```
Explain this error message as if I just started learning [language/framework] — what it means, why it's happening in my code specifically, and the underlying concept I'm missing. [paste error + code]
```

Why it works — teaches the underlying concept, not just the one-line fix.

1.19 Compare two technical approaches

BOTH

```
I'm choosing between [approach A] and [approach B] for [problem]. Give me a real trade-off table — not a listicle — and tell me which you'd pick for a team of [size] shipping in [timeframe].
```

Why it works — a contextualized recommendation beats a generic pros/cons list.

1.20 Write a commit message that explains "why"

PROFESSIONAL

```
Here's my diff: [diff]. Write a commit message that explains why this change was made, not just what changed.
```

Why it works — future-you (or your teammate) needs the reasoning, not a restatement of the diff. ---

2 Learning Anything

most people use AI as an answer-dispenser when learning, which feels productive but builds nothing. Used as a tutor — one that quizzes, questions, and pushes back — it builds understanding you can actually retrieve later, under pressure, without the tool in front of you.

2.1 Build a first-principles explanation

BOTH

```
Explain [concept] to me from first principles, as if I know nothing about the field it comes from. Use one real-world analogy, then show where the analogy breaks down.
```

Why it works — analogies teach faster; knowing where they fail prevents new misconceptions.

2.2 Socratic tutor mode

STUDENT

```
Teach me [topic] using the Socratic method — ask me questions that lead me to the answer instead of explaining it outright. Only give me the answer if I get stuck twice in a row.
```

Why it works — active recall beats passive reading, every time.

2.3 Test understanding, not memory

STUDENT

```
Quiz me on [topic] with questions that test whether I understand it, not whether I memorized it — application and edge-case questions, not definitions.
```

Why it works — surfaces shallow understanding before an exam does it for you.

2.4 Explain it three ways

BOTH

```
Explain [concept] three different ways: to a total beginner, to someone with related background knowledge, and to an expert looking for nuance. Show me all three.
```

Why it works — reveals which level you're actually operating at right now.

2.5 Build a learning roadmap with checkpoints

BOTH

```
I want to learn [skill] in [timeframe] starting from [current level]. Build me a week-by-week roadmap with a concrete checkpoint project at the end of each week.
```

Why it works — structure beats an open-ended pile of course links.

2.6 Find the gap in my mental model

STUDENT

```
Here's how I currently understand [concept]: [your explanation]. Find the gap or misconception in my understanding before correcting it.
```

Why it works — targeted correction instead of a full re-teach you don't need.

2.7 Turn a chapter into flashcards

STUDENT

```
Read this and generate 15 flashcards — question on one side, answer on the other — prioritizing the ideas most likely to be tested, not the most memorable trivia. [paste text]
```

Why it works — optimizes for exams, not novelty.

2.8 Explain why the "obvious" answer is wrong

BOTH

```
For [topic], what's the answer most beginners confidently get wrong, and why does it feel right even though it isn't?
```

Why it works — targets the single highest-value misconception directly.

2.9 Debate both sides to understand a topic

BOTH

```
Argue the strongest case for [position] on [topic], then argue the strongest case against it. Don't tell me which is right — let me decide after seeing both.
```

Why it works — understanding a real debate teaches more than being handed a conclusion.

2.10 Learn from a mistake, not just the correction

STUDENT

```
Here's a problem I got wrong and my incorrect answer: [problem + your answer]. Don't just give me the right answer — explain the specific reasoning error that led to mine.
```

Why it works — fixes the thought process, not just this one instance.

2.11 Compress a dense paper into what matters

PROFESSIONAL

```
Summarize this paper/article into: the core claim, the evidence for it, and the biggest limitation the authors admit or gloss over. [paste text]
```

Why it works — teaches critical reading, not just summarizing.

2.12 Build intuition before the formalism

STUDENT

```
Before showing me the formal definition or formula for [concept], build my intuition for why it exists and what problem it solves.
```

Why it works — formulas stick far better once you know why they exist.

2.13 Practice explaining it back

BOTH

```
I'm going to explain [concept] back to you in my own words. Listen, then tell me specifically what I got right, what I got wrong, and what I left out. Here's my explanation: [your explanation]
```

Why it works — the Feynman technique, on demand, with an actual critic.

2.14 Design a spaced-repetition schedule

STUDENT

```
I need to retain [topic] long-term, not just for a test. Design a spaced-repetition review schedule and tell me what to actively test myself on at each interval.
```

Why it works — retention beats cramming, and most people never plan for it.

2.15 Turn a skill gap into a project

BOTH

```
I want to get better at [skill] through building something real, not tutorials. Suggest 3 project ideas at increasing difficulty that would each force me to learn a specific missing piece.
```

Why it works — project-based learning beats tutorial hell, reliably.

2.16 Explain how experts actually think about this

PROFESSIONAL

```
How does someone with 10+ years of experience in [field] actually think about [problem], as opposed to how it's taught to beginners?
```

Why it works — closes the real gap between textbook framing and practice.

2.17 Stress-test my understanding with a hard question

BOTH

```
Ask me the single hardest, most clarifying question you could ask about [topic] to reveal whether I actually understand it or just recognize the vocabulary.
```

Why it works — one sharp question beats ten easy ones.

2.18 Explain a mistake in my own work

STUDENT

```
Here's my [assignment/solution]: [paste it]. Don't grade it — just tell me where my reasoning breaks down, in the order I'd have discovered the problems myself while working through it.
```

Why it works — mirrors real discovery and builds your own debugging instinct.

2.19 Connect a new topic to what I already know

BOTH

```
I already understand [topic A] well. Explain [new topic B] by mapping it onto what I already know about A — where the mapping holds, and where it misleads.
```

Why it works — learning by analogy to your own existing knowledge, not a stranger's.

2.20 Prep for an oral exam or viva

STUDENT

```
I have an oral exam/viva on [topic]. Play the examiner — ask me increasingly specific follow-up questions on my answers the way a real examiner would to probe for real understanding.
```

Why it works — simulates the actual pressure format, not just the content. ---

3 Design (Product, UI/UX & Visual)

good critique locates a specific problem; good creative direction imposes a specific constraint. Vague prompts get vague design feedback ("looks clean!") and vague creative options. Specific prompts get you something you can actually act on.

3.1 Critique like a senior designer

BOTH

```
Critique this design like a senior designer in a design review, not a fan. Focus on hierarchy, clarity, and one thing you'd cut entirely. [describe/attach design]
```

Why it works — cuts the flattery, keeps the useful part.

3.2 Find the confusing moment

PROFESSIONAL

```
Walk through this flow as a first-time user would. Tell me the exact point where you'd hesitate or feel confused, and why. [describe flow]
```

Why it works — locates real friction instead of collecting general opinions.

3.3 Justify a design decision under scrutiny

BOTH

```
I chose [design decision] for [reason]. Poke holes in that reasoning like a skeptical stakeholder would in a review.
```

Why it works — pressure-tests decisions before a real critique does it for you.

3.4 Generate constraints, not options

BOTH

```
I'm designing [thing] for [audience/context]. Instead of giving me options, give me the 3 hardest constraints I should be designing against.
```

Why it works — constraints produce sharper work than open-ended brainstorms.

3.5 Name the emotional tone before the visuals

BOTH

```
Before I choose any colors or fonts for [project], help me name the emotional tone it should evoke in one sentence, then tell me what visual choices would contradict that tone.
```

Why it works — prevents style-first, purpose-later design.

3.6 Compare two directions honestly

PROFESSIONAL

```
Here are two design directions for [thing]: [A] and [B]. Don't average them — tell me which one better serves [specific goal] and why the other one fails at it.
```

Why it works — forces a real decision instead of a watered-down compromise.

3.7 Explain accessibility gaps in plain terms

BOTH

```
Review this design for accessibility issues — color contrast, touch target size, screen-reader logic — and explain each issue in terms of a specific user who'd be blocked by it. [describe/attach design]
```

Why it works — makes abstract accessibility rules concrete and hard to dismiss.

3.8 Simplify without losing meaning

BOTH

```
Here's a screen/document with too much on it: [describe]. Tell me what to cut, what to combine, and what has to stay non-negotiably.
```

Why it works — ruthless prioritization beats surface-level tidying.

3.9 Design system consistency check

PROFESSIONAL

```
Compare these two components/screens against our existing design patterns: [describe/paste]. Flag any inconsistency in spacing, naming, or interaction that a user would subconsciously notice even if they couldn't name it.
```

Why it works — systemic thinking catches what one-off polish misses.

3.10 Build a moodboard brief from a vague ask

STUDENT

```
A client/professor asked for [vague creative brief]. Turn this into 5 concrete visual directions I could actually mock up, each with a one-line rationale.
```

Why it works — converts a vague request into buildable options instead of paralysis.

3.11 Reverse-engineer why something works

STUDENT

```
Here's a design/interface I admire: [describe/link]. Break down the specific decisions that make it work — not just 'it's clean,' but the actual mechanics.
```

Why it works — builds design vocabulary through analysis, not imitation.

3.12 Write empty state and error state copy

BOTH

```
Write the empty state and error state copy for [feature]. Make it useful, not just apologetic — tell the user what to do next in both cases.
```

Why it works — edge states are often the most neglected part of any UX.

3.13 Pressure-test a portfolio piece

STUDENT

```
Here's a project in my portfolio: [describe]. If a hiring manager only had 30 seconds on this piece, what would make them stop scrolling, and what would make them skip it?
```

Why it works — optimizes for real reviewer behavior instead of completeness.

3.14 Design for the worst-case user

BOTH

```
Design [flow/feature] assuming the user is distracted, on a bad connection, and has never used a product like this before. What breaks first?
```

Why it works — stress-tests against real conditions instead of ideal ones.

3.15 Get feedback framed as questions, not verdicts

BOTH

```
Review this design, but instead of telling me what's wrong, ask me the questions that would make me realize the problems myself. [describe/attach design]
```

Why it works — builds your design judgment instead of just fixing this one file.

3.16 Translate a business goal into a design principle

PROFESSIONAL

```
The business goal is [goal]. Translate that into one concrete design principle I can actually apply, not a restatement of the goal.
```

Why it works — bridges strategy and execution, which usually don't talk to each other.

3.17 Audit visual hierarchy in isolation

BOTH

```
If I squint at this design so I can only see shapes and contrast, not the actual content, what does my eye go to first, second, third? Does that order match what should matter most? [describe/attach design]
```

Why it works — isolates hierarchy from content bias.

3.18 Critique a rebrand or redesign choice

PROFESSIONAL

```
We're considering changing [brand element] from [old] to [new]. What would a loyal existing user assume this change signals, even if that's not our intent?
```

Why it works — anticipates the unintended reads before you ship them publicly.

3.19 Build a critique rubric before presenting

STUDENT

```
I'm about to present this design in a crit. Give me the 5 questions a tough critic is most likely to ask, so I can prepare answers in advance.
```

Why it works — preemptive defense beats reactive scrambling in the room.

3.20 Design the first 10 seconds

BOTH

```
Design the first 10 seconds of onboarding for [product] assuming the user will abandon it if they don't immediately understand the value. What's the one thing they need to see or do first?
```

Why it works — forces onboarding to earn attention instead of trying to explain everything. ---

4 Content & Growth

content and outreach fail for the same reason — genericness. A hook that could apply to any topic stops no one; an outreach message that could go to anyone gets deleted by everyone. Specificity is the entire game in both.

4.1 Turn one idea into a week of content

BOTH

```
I have one core idea: [idea]. Break it into 5 distinct pieces of content for [platform], each with a different angle so they don't feel repetitive.
```

Why it works — one idea, multiple entry points, instead of one post and a blank calendar.

4.2 Write the hook first, everything else second

BOTH

```
Write 10 different hooks/opening lines for content about [topic], optimized for someone scrolling past in under a second. Rank them by how likely they are to stop the scroll.
```

Why it works — the hook decides whether the rest of the content ever gets seen.

4.3 Find the angle nobody else is using

PROFESSIONAL

```
Everyone talks about [topic] the same way: [common angle]. Give me 3 contrarian or unexpected angles on the same topic that are still true.
```

Why it works — differentiation without needing to be dishonest.

4.4 Turn a personal experience into a lesson

BOTH

```
I went through [experience]. Help me turn this into content that leads with the specific, relatable detail before the lesson — not the lesson first.
```

Why it works — specificity builds trust before advice does.

4.5 Write cold outreach that isn't generic

PROFESSIONAL

```
I want to reach out to [person/role] at [company] about [reason]. Write an opener that references something specific and true about them, not a template compliment.
```

Why it works — personalization beats volume, every time it's tested.

4.6 Draft a follow-up that doesn't sound needy

PROFESSIONAL

```
I sent this message [timeframe] ago with no response: [message]. Write a follow-up that adds new value or information instead of just 'checking in.'
```

Why it works — every follow-up should earn a reply, not just request one.

4.7 Turn an objection into content

BOTH

```
The most common objection I hear about [product/idea/topic] is [objection]. Turn addressing that objection into a piece of content, not a defensive rebuttal.
```

Why it works — pre-empts sales conversations before they happen.

4.8 Write a caption that ends with a real CTA

BOTH

```
Write a caption for this post about [topic] that ends with a call-to-action tied to something the reader actually wants, not a generic 'follow for more.'
```

Why it works — a specific incentive beats a generic ask, always.

4.9 Build a content calendar from constraints

PROFESSIONAL

```
I can realistically post [frequency] on [platform] about [niche]. Build me a content calendar for the next [timeframe] that mixes formats so it doesn't feel repetitive.
```

Why it works — a sustainable cadence beats a sporadic burst that burns out in two weeks.

4.10 Qualify a lead before reaching out

PROFESSIONAL

```
Here's what I know about this lead: [details]. Based on this, is this a good-fit prospect for [product/service], and what's the single most relevant thing to lead with if I reach out?
```

Why it works — focuses effort before it's spent, not after.

4.11 Write a subject line that gets opened

PROFESSIONAL

```
Write 8 subject lines for an email about [topic/offer]. Optimize for curiosity or specificity, not urgency or hype.
```

Why it works — curiosity earns opens without feeling spammy.

4.12 Repurpose one piece across formats

BOTH

```
Here's a piece of long-form content: [paste/describe]. Turn it into a short social post, a one-line quote graphic, and a 3-slide carousel — each rewritten for how people actually consume that format, not just cut down.
```

Why it works — format-native repurposing beats copy-pasting and shrinking.

4.13 Diagnose why content underperformed

PROFESSIONAL

```
This piece of content underperformed: [describe/paste]. Compared to what I know performs well for this audience, what's the most likely reason — hook, format, timing, or relevance?
```

Why it works — diagnosis before you repeat the same mistake next week.

4.14 Write a bio that says what you actually do

BOTH

```
Here's a rough description of what I do: [description]. Write 3 versions of a short bio for [platform], each emphasizing a different angle (expertise, personality, outcome).
```

Why it works — gives you positioning options instead of one guess.

4.15 Turn a case study into a story

PROFESSIONAL

```
Here's the raw outcome of a project: [details]. Turn this into a short story with a real before/after, not a bullet list of results.
```

Why it works — narrative holds attention far longer than stats alone.

4.16 Write a comment-keyword CTA that doesn't feel gimmicky

BOTH

```
I'm giving away [resource] in exchange for a comment with the word [keyword]. Write a CTA line that explains this naturally, without sounding like a growth hack.
```

Why it works — transparency preserves trust in the mechanic itself.

4.17 Build a student's first content plan

STUDENT

```
I'm a student starting to build an audience around [interest/niche]. I have no existing following. Build me a realistic first-month content plan that accounts for having zero credibility yet.
```

Why it works — starts from real constraints instead of aspirational ones.

4.18 Write a pitch that leads with their problem

PROFESSIONAL

```
I want to pitch [service/product] to [type of prospect]. Write an opening paragraph that starts with their specific problem, not my solution.
```

Why it works — problem-first pitches get read further before being dismissed.

4.19 Turn negative feedback into a response

BOTH

```
Someone left this critical comment/review: [paste]. Write a response that acknowledges the specific point without being defensive or over-apologizing.
```

Why it works — calibrated tone protects your reputation either way this goes.

4.20 Find the one metric that actually matters this week

PROFESSIONAL

```
Here's what I'm tracking: [list metrics]. If I could only look at one number this week to know if things are working, which should it be, and why?
```

Why it works — forces focus instead of dashboard-staring. ---

5 AI Image, Video & Animation Generation

generators respond to concrete, describable detail — subject, action, environment, style, lighting, camera, mood — not adjectives like "cinematic" or "beautiful" on their own. The less you leave to the model's default interpretation, the closer the output lands to what you actually wanted. These prompts are written to work across tools rather than one specific app.

5.1 Build a full visual prompt from a vague idea

BOTH

```
I want an image of [vague idea]. Turn this into a full prompt specifying subject, action, environment, lighting, camera angle, and art style — ask me which style direction I want before finalizing.
```

Why it works — forces the specificity generators actually need to work well.

5.2 Diagnose why a generated image looks off

BOTH

```
Here's the prompt I used and a description of what came out wrong: [prompt] → [what's wrong]. Tell me which part of the prompt is most likely causing that specific problem.
```

Why it works — debugs the prompt instead of retrying blindly and hoping.

5.3 Write a consistent character description

PROFESSIONAL

```
I need the same character to appear across multiple generated images. Write a locked character description — physical details, outfit, distinguishing features — precise enough to stay consistent across separate generations.
```

Why it works — consistency requires precision, not just repeating a name.

5.4 Translate a mood into visual language

BOTH

```
I want an image that feels like [mood/emotion], not that depicts it literally. Translate that feeling into concrete visual choices: color palette, lighting direction, composition, and camera distance.
```

Why it works — moods have to become describable specifics before a generator can render them.

5.5 Storyboard before generating video

PROFESSIONAL

```
I want a short video of [concept]. Break it into a shot-by-shot storyboard first — what's in frame, camera movement, and duration per shot — before writing generation prompts for each.
```

Why it works — plans the sequence before you spend generations discovering it doesn't work.

5.6 Write a camera-language prompt

BOTH

```
Rewrite this prompt using proper camera/cinematography language — shot type, lens feel, camera movement — instead of vague descriptors like 'cinematic.' [paste prompt]
```

Why it works — specific camera terms consistently outperform vague style words.

5.7 Generate style-transfer instructions

BOTH

```
I like the visual style of [reference, described in words]. Break that style down into its component parts — color, texture, linework, lighting — so I can apply it to a completely different subject.
```

Why it works — separates style from subject so it actually transfers cleanly.

5.8 Prompt for a specific animation principle

STUDENT

```
I'm animating [action/object] and want it to follow the animation principle of [e.g. squash and stretch, anticipation]. Describe what that would concretely look like frame-to-frame for this specific action.
```

Why it works — connects animation theory to a specific, executable frame breakdown.

5.9 Write a negative prompt that actually helps

BOTH

```
Here's my prompt: [prompt]. What are the 5 most likely unwanted elements a generator would add, so I can explicitly exclude them?
```

Why it works — anticipates common failure modes instead of reacting after a bad result.

5.10 Build a shot list for a screen-recorded video

BOTH

```
I'm making a video about [topic] using screen recordings and simple face-to-camera clips, no advanced editing skill. Build me a shot list that's realistic to actually film.
```

Why it works — keeps ambition matched to real production capability.

5.11 Match lighting across a sequence

PROFESSIONAL

```
I need multiple generated images to feel like they're lit by the same light source and time of day. Write a lighting description precise enough to stay consistent across separate prompts.
```

Why it works — visual consistency lives in the lighting spec, not luck.

5.12 Turn a product into a scene

PROFESSIONAL

```
I want to showcase [product] in a generated scene, not on a plain background. Suggest 3 environments that would make sense contextually and describe each in full prompt-ready detail.
```

Why it works — context sells a product better than isolation does.

5.13 Write a prompt for a specific emotion on a face

BOTH

```
I need a generated character/face expressing [specific emotion], not a generic expression. Describe the exact facial and postural details that would convey that emotion, not just the emotion's name.
```

Why it works — emotions need physical description, not a label, to render convincingly.

5.14 Explain why my video prompt produced weird motion

STUDENT

```
Here's my video generation prompt: [prompt]. The motion came out wrong in [specific way]. What part of the prompt likely caused that, and how should I rephrase the action description?
```

Why it works — diagnoses motion-description failures specifically instead of rewriting from scratch.

5.15 Build a moodboard prompt set

BOTH

```
I want to explore 5 different visual directions for [project] before committing. Write 5 distinct full prompts, each representing a genuinely different style, not minor variations of one idea.
```

Why it works — real range beats five near-duplicate options.

5.16 Design a simple animated loop

STUDENT

```
I want a short looping animation of [simple concept] for a portfolio piece. Describe the start frame, end frame, and what needs to match between them for the loop to feel seamless.
```

Why it works — loops live or die on the seam, so plan it explicitly instead of hoping.

5.17 Write a prompt using composition rules

BOTH

```
Rewrite this prompt to specify a composition rule — rule of thirds, leading lines, framing — instead of just describing the subject. [paste prompt]
```

Why it works — composition control is the detail most people skip entirely.

5.18 Match a brand's visual identity in a generated asset

PROFESSIONAL

```
Here's my brand's visual identity: [colors, fonts, tone]. Write a generation prompt for [asset] that would actually look on-brand, not just aesthetically nice.
```

Why it works — aesthetic quality isn't the same thing as brand fit.

5.19 Prompt for a specific texture or material

BOTH

```
I need [object] to look like it's made of [material] convincingly. Describe the specific surface qualities — reflectivity, texture, how light interacts with it — that sell that material.
```

Why it works — materials are sold through light behavior, not the material's name.

5.20 Critique your own generated output like a director

BOTH

```
Here's the image/video I generated and the prompt I used: [describe/paste]. Critique it like a director reviewing a shot — what's technically off, and what's the single next prompt revision that would fix the biggest problem first?
```

Why it works — one prioritized fix beats a list of everything that's wrong. ---

The one meta-skill

Every prompt above is a variation on the same four moves: give real context, ask for something specific, tell it what to prioritize, and shape the output format. Once that's automatic, you don't need a document like this anymore — you'll just write prompts that way by default.

If this was useful, save it, and follow @the.anadi for more of these — free, no course, no waitlist.