

The LLM Cost Engineering Handbook

Measure, Model, and Cut AI Spend Without Losing Quality

SmNix

Contents

Before You Start	3
Introduction: The Bill Nobody Owned	5
Chapter 1: Your Bill Is a Design Document	7
Chapter 2: What One Piece of Work Costs	11
Chapter 3: Where the Money Actually Goes	15
Chapter 4: Logging What Every Call Costs	21
Chapter 5: Knowing If You Broke Anything	27
Chapter 6: Stop Paying Twice for the Same Words	33
Chapter 7: Using a Cheaper Model, Properly	39
Chapter 8: Spend Less on Thinking	44
Chapter 9: Keeping the Conversation Short	48
Chapter 10: Work That Doesn't Need Answering Now	52
Chapter 11: Making the Agent Stop	55
Chapter 12: Should You Run Your Own?	60
Chapter 13: Catching It Before It Ships	64
Chapter 14: Splitting the Bill, and Defending It	68
Chapter 15: Six Ways It Goes Wrong	72
Chapter 16: When It Isn't Just Text	77
Appendix A: The Cost Model	82
Appendix B: Prices, and How to Rebuild Them	89
Appendix C: The 30-Day Plan	93
Appendix D: The Audit Checklist	96
Appendix E: Sources, and When They Were Checked	100
Glossary	102

Before You Start

Copyright © 2026 SmNix. All rights reserved.

No part of this book may be reproduced or transmitted in any form without written permission from the author, except for brief quotations in a review.

First edition, 2026. Version 1.1, September 2026.

Every provider figure in this book was checked against the provider's own documentation on the date recorded in Appendix E.

This book is general technical information, not financial, legal or procurement advice. Prices and provider behaviour change, so check current terms before you act on any figure in it.

Anthropic, Claude, OpenAI, Google, Gemini and DeepSeek are trademarks of their respective owners. They are named for identification only. This book is not affiliated with or endorsed by any of them.

Four things to know

The prices in this book are examples, not quotes. They were accurate when written and they will not stay that way. Every worked example states its assumptions at the top and shows the formula as well as the answer, so you can rebuild any of it with current numbers. Appendix B explains how, and it takes about twenty minutes.

The worked examples are models, not case studies. Nothing in this book is an account of a real company's incident, including the team in the Introduction. The arithmetic in every one is real and checkable, and the failure patterns in Chapter 15 are common enough that you'll recognise them. I'd rather hand you numbers you can verify than stories you have to take on faith.

Everything is priced per token. A token is the unit providers actually bill in: a fragment of text, usually a common word or a piece of a longer one. In ordinary English prose a token is roughly three quarters of a word, so a thousand words is about 1,300 tokens. That ratio shifts for other languages and falls apart for code, so don't estimate when the number matters. Providers give you a way to count exactly, and Appendix B, section B.1 covers it.

This book assumes you ship. It doesn't explain what a language model is or how to call one. It's for people who already have something running and have been asked what it costs.

How to read it

There are three parts, meant to be done in order, and a final chapter that stands on its own.

Part I, Chapters 1 to 5. Find out what you're paying for. What a unit of work costs, where the money goes inside a single run, logging that answers those questions continuously, and how to tell whether a change made the product worse. This is the part people want to skip, and skipping it is why so much cost work turns into a week of effort that saves nine percent.

Part II, Chapters 6 to 11. The six ways to spend less. One lever per chapter, ordered by how much they typically pay back on an agent workload. Each covers how it works, the arithmetic for deciding whether it applies to you, when it backfires, and how to confirm it worked.

Part III, Chapters 12 to 15. Keep it from creeping back. Whether owning the hardware would lock the savings in, then how to stop them decaying: cost tests in your build, splitting the bill across teams, and what a cost incident looks like when nobody was watching.

Chapter 16, When it isn't just text. Images, audio and video are counted differently and surprise people. Read it if you take any of them; skip it if your work is all text.

Appendix E lists every fact in this book that came from outside it, with the date it was last checked against the source. If a figure here matters to a decision you're making, check its date first.

Every chapter ends with **The Number**: one figure to work out for your own system before moving on. Do them. Appendix A collects them all into one cost model, and by the last chapter you'll have a complete picture of your own costs, which is worth more than having read about somebody else's.

If you want to start today rather than finish the book first, go to Appendix C. It's a thirty-day plan that assumes you're starting from nothing. If your waste multiple is high, day 10 is where most of the saving is.

Introduction: The Bill Nobody Owned

The feature worked.

That's the part people forget when they tell this story. The team had spent two months building an assistant that read support tickets, checked three internal systems, and drafted a reply. It was good. Response times dropped. Customers noticed. Nobody was thinking about money.

Then finance sent an email asking why the AI line had gone from four thousand dollars to thirty-one thousand in a single month, and whether that was expected.

The engineering lead pulled up the provider dashboard. It showed one number: total spend, for the whole account, for the month. No breakdown by feature. No breakdown by customer. No way to tell which of the six things the company had shipped was responsible, or whether thirty-one thousand was reasonable for the work being done.

They couldn't answer it, and carelessness had nothing to do with it. Nothing in the system had ever been built to answer that question.

That team is a composite, like every example in this book. The email is the part you'll recognise.

The gap this book fills

Think about what you already measure.

Latency has percentile dashboards and alerts that fire in seconds. Errors have exception tracking, ownership, and a runbook. Uptime has a target on a wall somewhere, and someone whose weekend gets ruined when it slips.

Cost has an invoice that shows up thirty days late, added up across your entire account, with no way to trace a single dollar back to a single decision.

That's the gap. Every other property of a production system gets measured continuously and owned by someone. Cost gets discovered afterward, by accident, usually by somebody outside engineering.

It's a strange thing to have left out, because cost now decides a lot of what gets built: which features ship, how many steps an agent is allowed, which model a prototype gets to keep. When the bill lands on finance first, it looks like a finance problem. It's an engineering problem that engineering never built the instruments to see.

What you'll be able to do

By the end of this book you'll be able to:

Say what one unit of your product actually costs. Not cost per token, which tells you almost nothing. The cost of one resolved ticket, one generated report, one active user for a month.

Find where the money goes, in a specific system, in about an hour.

Cut that bill by something like half without making the product worse, and prove the product didn't get worse.

Stop a code change from tripling your costs before it ships, the same way you'd stop a change that tripled your response times.