AI Token Cost Projection Framework

A methodology for forecasting enterprise LLM API spend

Provider-agnostic · Bottom-up + Top-down · Companion Excel model

Smit Jani, CAITL

AI Governance & Data Strategy Consultant

June 2026

Version 1.0 · June 2026

Contents

1. Purpose and scope

Goal. This framework gives a finance, platform, or AI-enablement team a repeatable way to estimate and forecast the cost of large-language-model (LLM) usage across an organization. It is deliberately **provider-agnostic**: you plug in whatever token prices apply to your contracts, and the structure works for any vendor or self-hosted model.

What it covers. Usage-based API/token cost: the dominant variable cost of running LLM features and assistants. It produces a steady-state run-rate, a time-phased projection that accounts for adoption, and a low/base/high scenario envelope.

What it does not cover. Per-seat SaaS licenses (e.g. flat-fee assistant subscriptions), model fine-tuning projects, GPU capital for self-hosting, data-pipeline engineering, and headcount. These can be layered on as separate line items; the token model is the piece that scales non-linearly with usage and is hardest to estimate by intuition.

All pricing figures in this document are benchmarks captured in June 2026 and should be treated as verify-required inputs. LLM pricing changes frequently; always confirm against each provider's current pricing page before committing a budget.

2. The core cost equation

Every estimate in this framework reduces to one relationship. For a single workflow, monthly cost is:

Cost = (Input tokens × Input price) + (Output tokens × Output price)

Expanded to the drivers you actually control:

Monthly cost = Users × Interactions per user per day × Working days × (Tokens per interaction) × Effective price per token

Effective price is the list price adjusted for two structural discounts most providers offer: prompt caching (cached input tokens are heavily discounted) and batch/async processing (typically around half price). The companion model computes this blended rate automatically per workflow.

Why output dominates. Output tokens are normally priced several times higher than input tokens (often 5×). A workflow that returns long answers can cost far more than its input volume suggests, so estimate input and output token counts separately — never as a single blended number.

3. Cost drivers

Eight variables explain almost all of the spend. Sorting an estimate by these is the fastest way to see where the money goes and which levers matter.

Active users	Headcount actually using the feature, not licensed seats	Adoption rarely starts at 100%; the ramp is a major driver of year-1 cost
Interactions / user / day	Calls or turns per active user	Heavy tools (coding, BI) can be 10–30× lighter ones
Input tokens / call	Prompt + context + retrieved documents	RAG and long context inflate this quickly
Output tokens / call	Length of the model's response	Priced highest; the single most sensitive lever
Model tier	Which model serves the workflow	Frontier vs economy models differ ~5–50× in price
Cache hit rate	Share of input served from prompt cache	Cuts the input portion of cost sharply
Batch share	Share of volume run async/batched	~50% discount where latency is tolerable
Working days	Days per month the workflow runs	Converts daily usage to a monthly figure

4. Two estimation methods

Estimate cost two independent ways and reconcile them. Agreement builds confidence; a large gap tells you an assumption is wrong before it reaches a budget.

4.1 Bottom-up (use-case build)

Enumerate each AI workflow and estimate its drivers individually, then sum. This is the more accurate and defensible method, and the one to lead with.

Tip: ground token counts in real samples. Run a handful of representative prompts through the provider's tokenizer rather than guessing — estimates made from word counts are often off by a wide margin.

4.2 Top-down (per-seat)

Group the population into a few personas (power users, knowledge workers, light users), assign each a typical tokens-per-active-seat-per-day, and multiply by headcount. This is coarser but fast, and serves as a sanity check on the bottom-up figure.

Reconciliation rule. Aim to keep the two methods within roughly ±25% of each other. If they diverge more, revisit the assumption you are least sure of — usually tokens per interaction or the active-user share.

5. Time-phasing: the adoption ramp

Steady-state run-rate is not what you spend in year one. Usage ramps as the rollout proceeds, so the projection scales the steady-state figure by the share of target users active each month.

The practical consequence: budget the ramped figure for year one, but plan capacity and guardrails around the higher steady-state run-rate you will reach later.

6. Price and efficiency trends

Token prices have trended down materially over time as competition and efficiency improve, while capability per dollar has risen. A multi-year projection that holds today's prices flat will usually overstate later-year cost.

7. Levers to reduce cost

Once the model shows where spend concentrates, these are the highest-leverage ways to bring it down, roughly in order of impact:

8. Scenario and sensitivity analysis

Single-point estimates create false precision. Bracket the forecast with three scenarios driven by multipliers on the most uncertain inputs — adoption/usage, token price, and tokens per interaction.

Low	Conservative adoption, lower prices, leaner prompts	Floor for budget; best case for cost
Base	Most-likely assumptions	Primary planning number
High	Aggressive adoption, higher prices, richer prompts	Guardrail / contingency ceiling

Read the spread, not just the midpoint. If the high case is multiples of the low case, the forecast is dominated by a few uncertain assumptions — invest in measuring those (run a pilot, instrument real token usage) before committing.

9. Governance and ongoing monitoring

A projection is only useful if it is checked against reality. Establish a monthly loop:

Over a few cycles this turns the model from a one-off estimate into a living forecast that gets more accurate as real data accrues.

10. Using the companion Excel model

The framework ships with a parameterized workbook, AI_Token_Cost_Model.xlsx. It implements everything above and is built to be a reusable template — change the blue input cells and every projection updates.

Guide	How to use the workbook, color legend, token rule of thumb
Rate Card	Provider-agnostic model tiers with editable $/1M prices and discount levers
Assumptions	Global drivers: org size, working days, adoption ramp, price/usage trends, overheads
Bottom-Up	Per-use-case build and steady-state monthly cost (the core calculation)
Top-Down	Per-seat cross-check with automatic variance vs bottom-up
Projection-12mo	Month-by-month year-one forecast with the adoption ramp
Projection-36mo	Three-year forecast including price and usage trends
Scenarios	Low / base / high envelope from driver multipliers
Dashboard	Headline KPIs and cost-by-use-case chart

Workflow: set your Rate Card prices → set Assumptions for your org → fill in the Bottom-Up workflows → check the Top-Down variance → read the projections, scenarios, and dashboard. The pre-filled numbers are illustrative defaults for a ~5,000-user organization; replace them with your own.

Appendix: reference pricing benchmarks (June 2026)

Indicative public API list prices, $ per 1,000,000 tokens, captured June 2026. These move frequently — verify before use. Batch processing is commonly ~50% off and cached input ~90% off across providers.

Anthropic Claude Opus 4.8	5.00	25.00	Frontier High
OpenAI GPT-5.5	5.00	30.00	Frontier High
Anthropic Claude Sonnet 4.6	3.00	15.00	Frontier Mid
OpenAI GPT-5.4	2.50	15.00	Frontier Mid
Google Gemini 3.1 Pro	2.00	12.00	Frontier Mid
Google Gemini 3.5 Flash	1.50	9.00	Mid / Fast
Anthropic Claude Haiku 4.5	1.00	5.00	Economy
OpenAI GPT-5	0.63	5.00	Economy
Google Gemini Flash-Lite	0.10	0.40	Ultra-economy

Sources: Anthropic, OpenAI, and Google public pricing pages and pricing-aggregator summaries, accessed June 2026. Figures are approximate and provided only as starting defaults for the model's editable Rate Card.

About the author

Smit Jani is a management consultant specializing in AI governance, responsible AI, and enterprise data strategy in regulated financial services, with engagements spanning GenAI adoption, audit remediation, and enterprise LLM platform delivery at major North American financial institutions. He holds the CAITL certification and is the author of the forthcoming book *Human in the Lead*.