
BLACKTECHSTARTUP EDUCATION

Choose the Cheapest AI Architecture That Can Actually Do the Job

The most expensive model is not automatically the best product. Match task difficulty, retrieval, tools, structured output, latency and error tolerance to the architecture before you burn margin.

The most expensive model is not automatically the best product. Match task difficulty, retrieval, tools, structured output, latency and error tolerance to the architecture before you burn margin.

Start with the job and the cost of being wrong

Before selecting a model, write down the task, allowed latency, data sensitivity, output format and consequence of failure. Extracting five known fields from a standard invoice is a different problem from researching a complex acquisition. A customer-support classifier is different from a legal drafting assistant. The more structured and repeatable the task, the more you should test whether a smaller or cheaper model can handle it. Reserve expensive reasoning capacity for steps where it materially improves outcomes. This is how a founder protects gross margin instead of routing every request to the most powerful model by habit.

Separate knowledge problems from reasoning problems

If the model lacks current or private facts, a larger model may not solve the problem. Retrieval, database access or web search may be the missing architecture. If the model has the facts but struggles to compare tradeoffs or plan across constraints, stronger reasoning may help. Diagnose the failure before upgrading. Ask: did the model not know, not understand, not follow format, or fail to execute an action? Each failure points toward a different fix—better context, clearer instruction, structured output, a tool or a different model.

Use structured outputs when software consumes the result

If the next step is code, do not rely on prose such as 'please return valid JSON.' Modern model APIs can constrain output to a schema or function argument shape. That reduces parsing errors and makes downstream validation easier. Still validate values and business rules after generation. For example, a schema can guarantee that `amount` is numeric, but it cannot guarantee the amount is correct. Structured output improves interface reliability; it does not replace factual evaluation.

Route by difficulty and verification cost

A practical system can send simple classification, rewriting or extraction to a lower-cost model and escalate only ambiguous or high-value cases. Routing can be based on task type, document length, user tier or an initial confidence signal that you have validated. But routing itself needs evaluation: a cheap first model that misroutes difficult cases can create hidden failures. Measure total cost per successful task, including retries, human review and tool calls. A ten-cent answer that needs ten minutes of staff correction is not cheaper than a one-dollar answer that passes.

Know when prompting has reached its limit

If the task requires proprietary facts, add retrieval. If it requires an action, add a tool. If the output must obey a machine schema, use structured outputs. If a repeated style or domain behavior remains unstable across many examples, evaluate whether fine-tuning is appropriate. If latency is unacceptable, change model size or workflow. If the task is deterministic arithmetic or business logic, write code. A common AI architecture mistake is trying to prompt around a missing system component because prompts are easy to edit.

Build the architecture scorecard

For each candidate design, score task success, severe-error rate, median and tail latency, cost per successful task, human-review minutes, implementation complexity and vendor dependency. Test at least two architectures on the same evaluation set. Example: small model + retrieval versus large model with long context; or small model for intake plus reasoning model for escalation. Choose the lowest-cost design that clears your quality and risk threshold with room for production variance. That is the architecture you can price sustainably.

Run a three-architecture bake-off

Take one real use case and implement the smallest possible version of three designs. Design A: a low-cost model with concise instructions. Design B: the same model plus retrieval or a tool that supplies missing facts. Design C: a stronger reasoning model used only on cases the first path cannot handle. Run all three on the same evaluation set. For every attempt, record provider cost, total latency, pass/fail, human correction minutes and number of tool calls. Then calculate cost per accepted outcome. Include engineering cost qualitatively: a design that saves \$200 per month but requires a custom orchestration layer nobody can maintain may not be cheaper. Also test failure recovery. What happens if retrieval returns nothing, the tool times out or the large model is unavailable? The bake-off reveals whether intelligence, information or system design is the actual constraint. It also gives you pricing evidence. If the average successful support case costs nine cents, you can model gross margin from expected usage; if complex research cases cost \$1.80 and require review, price them differently. Architecture becomes a business decision when model quality and unit economics are measured together.

SOURCE DESK

Research behind this guide

Use the primary sources below to verify current rules, eligibility and program details before acting. Program terms can change.

OpenAI — Model selection and current model guidance



OpenAI — Structured outputs



OpenAI — Function calling



Google Cloud — Generative AI model selection and evaluation



Important: This article is educational information, not individualized legal, tax, investment or compliance advice. For decisions with legal or financial consequences, use a qualified professional who can review your facts.

SHARE THIS GUIDE ↗

POWERED BY: BLACK TECH STARTUP

blacktechstartup.com