GPT-5.6 Sol across every effort level

VulcanBench Frontier v4 | Code quality protocol v3.7 | September 2026

Abstract

GPT-5.6 Sol ran the 23-task VulcanBench Frontier v4 suite through the Codex CLI on a ChatGPT Pro subscription on September 17 to 18, 2026, once per task at each of the five effort levels its API offers, 115 runs in all. Code quality carries 33% of the combined score and is judged for a named human reader by Muse Spark 1.3 (Meta) and Grok 4.6 (xAI) under the same frozen protocol as the other Frontier v4 reports, with a ground-truth intent-recovery probe. The combined score rises from 69.05 at Low to 87.18 at Max, and most of that range is the first step: 13.5 points from Low to Medium, then 3.2 to High, 0.7 to Extra-high and 0.7 to Max. Sol passes 6 of 23 tasks at Low, 20 of 23 at High and 21 of the 22 judged at Max. Code quality stays between 64.98 and 68.19 at every effort: the knob buys correctness, not readability. Cost per task does not follow the ladder: $1.36 at Low, $2.05 at High, the most expensive level, and $1.77 at Max.

One Max run, codeccore, has no published Code quality score: Grok 4.6's intent probe quoted an excerpt absent from the code on both attempts and the frozen protocol accepts no invented characters, so the Max cell's Code quality and combined score cover 22 of its 23 runs. The run passed its tests and is counted in every runtime, token and cost figure. Under the prior 50/15/15/20 profile applied to the same Code quality scores the ladder has the same shape, so the picture does not depend on the weight change.

Table 1. Combined score, Code quality, tasks passed and runtime by effort

Model	Effort	Combined (33%)	SE	Combined (20%)	Code quality	Min/task	Passed
Sol	Low	69.05	3.50	72.24	64.98	9.8	6/23
Sol	Medium	82.56	1.25	85.33	68.19	11.1	12/23
Sol	High	85.76	0.98	88.51	67.65	12.0	20/23
Sol	Extra-high	86.47	0.80	89.23	67.31	10.6	21/23
Sol	Max	87.18	0.62	89.90	67.58	11.6	21/22

Combined (33%) = 0.50 functional + 0.085 lint and complexity + 0.085 security + 0.33 Code quality. Combined (20%) applies the prior 50/15/15/20 profile to the same Code quality scores for comparison. SE is one sample standard error across the cell's tasks, not judge uncertainty or a significance test. Passed counts judged tasks with a perfect functional score; at Max that is 21 of the 22 judged runs (22 of 23 in the sweep). Runtime is solver wall-clock per task over every run in the cell; judging is excluded.

The Code quality protocol

Two failure modes motivated the reviewed score. The lint and complexity metric rewards compression: the maintainability index carries a lines-of-code term and per-function complexity stays low when each dense line does something different, so five statements on a line with names like x and k can outscore the same logic written for a person. And a large model reading compressed code pays almost nothing to parse it; when a frontier model was calibrated as a readability judge, it rated a squashed six-line function the same as its formatted copy. Since September 7, 2026 Code quality carries 33% of the combined score and lint and complexity and security carry 8.5% each. The prior profile is reported beside the new one on every table.

Protocol v3.7 is the same protocol applied to this population. Nothing in the rubric, controls, quirk keys, gates, repeats, seed or judge settings changed; both judges retook the calibration exam under v3.7 before any counted call. The population froze on September 18 with all five cells complete, none missing and none excluded. A submission without a valid answer-key match from every passing panel is left unpublished by the frozen summary; that rule applied once, described under Results in detail.

The rubric

Every judge receives the same instructions, frozen by hash before any review. The prompt names the reader it scores for: an engineer who has never seen the code, reads it top to bottom without running it, and must make a correct change in one sitting. It tells the judge that its own ease at parsing dense code is not evidence of readability. Six dimensions are scored 0 to 4 in half steps against written anchors, each with an exact excerpt and a concrete consequence for that reader.

Sub-score	Dimension	What is assessed
Human readability	Naming	Identifiers say what things are in the task's domain
Human readability	Presentation	Statements per line, nesting, function length, how much the reader must hold at once
Human readability	Intent	Constants, thresholds and quirks explained by names or accurate comments that say why
Maintainability	Structure	Coherent responsibilities, no duplicated policy, no abstraction beyond the task's scale
Maintainability	Changeability	A named plausible change, every edit site traced, scored on locality and safety
Maintainability	Verifiability	Explicit state, failures that name what went wrong, seams to test one rule alone

The layers under the 33 points

Reviewed panel, 24 points. The six-dimension rubric above, averaged across both judges. **Intent recovery, 9 points.** Every task in this suite is a rewrite of a retired binary whose real behaviour departs from its written specification in documented ways. The judge sees only the specification and the code, never the issue text, and lists where the code departs from the specification; a separate call matches that list against the frozen answer key. The denominator is the quirks the submission actually passed tests for, so a functional failure is not punished twice; a submission that passed none would have no intent-recovery score and its Code quality would be the reviewed score alone, which no Sol run needed. **Measured maintenance, designed for 12 points, not yet built.** Until it exists the pre-registered split above is in force and is stated on the card.

Judges and calibration

The scored panel is Muse Spark 1.3 through the Muse CLI on its Standard tier, which does not train on prompts, and Grok 4.6 through the Cursor CLI at medium effort. Neither lab has a model on this board, so neither judge grades a relative. Every session is fresh, tools are disabled, the workspace is empty, model identity is checked per call from the CLI's own records, and solver labels are withheld.

The calibration exam

Before scoring a single submission each judge reviews ten held-out programs that implement the same ledger specification: clear, compressed, compressed then auto-formatted, verbose with duplicated policy, needlessly abstracted, misleadingly commented, narrated with a comment on every line, a documented legacy quirk, an embedded instruction to give full marks, and hidden module state. Each program is reviewed five times in a seeded order. Twenty gates fixed in advance check that the judge sees the construct; a judge may miss at most one gate by at most half a point.

Table 2. Calibration verdicts

Judge	Protocol	Calls	Result	Allowance used	Failing gates
Muse Spark 1.3	code-quality-maintenance-v3.7	80	passed	no	none
Grok 4.6	code-quality-maintenance-v3.7	80	passed	no	none

Both judges passed every gate with no allowance used. Under v3.5 and v3.6 Muse Spark 1.3 had needed the pre-registered allowance on the repeatability gate; under v3.7 it did not.

Table 3. Control means on five of the ten calibration programs

Judge	Control	Naming	Present.	Intent	Struct.	Change.	Verif.
Muse	clear	3.90	4.00	3.50	4.00	3.90	3.50
Muse	compressed	1.00	1.90	1.10	3.70	2.70	3.00
Muse	formatted	1.30	3.40	1.30	3.90	2.90	3.10
Muse	misleading comments	4.00	3.70	0.40	3.90	2.40	3.40
Muse	hidden state	2.30	3.00	1.90	1.50	2.50	1.80
Grok	clear	3.60	3.50	3.30	4.00	3.90	3.50
Grok	compressed	1.20	1.50	1.40	3.20	2.60	2.90
Grok	formatted	1.50	2.50	1.60	3.20	2.70	3.00
Grok	misleading comments	3.50	3.10	1.00	3.70	2.40	3.10
Grok	hidden state	3.00	2.80	2.40	2.00	3.00	2.00

Both judges separate the clear program from the compressed one by more than two points on naming and by 2.0 to 2.1 points on presentation, credit formatting mainly on presentation, penalise misleading comments on intent, and penalise hidden state on verifiability. Full gate values are in calibration.json.

Results in detail

Table 4. Code quality components at each effort level

Component	Low	Medium	High	Extra-high	Max
Code quality	64.98	68.19	67.65	67.31	67.58
Human readability	57.4	61.2	58.8	59.2	59.8
Maintainability	66.9	69.8	67.8	67.8	68.3
Intent recovery	72.4	75.3	79.3	77.4	77.1
Reviewed score	62.2	65.5	63.3	63.5	64.0
Rated by Muse Spark 1.3	60.2	64.3	62.0	61.8	63.0
Rated by Grok 4.6	64.1	66.8	64.6	65.3	65.1
Standard error of Code quality	1.56	1.30	1.17	1.39	1.39
Judged runs	23	23	23	23	22

Code quality sits between 64.98 and 68.19 at every effort, with the two judges within four points of each other in every cell and Grok 4.6 the higher rater at each level. Intent recovery, the ground-truth layer, rises from Low to High as more of the hidden contract is implemented and holds there; human readability and maintainability do not move with the knob.

Table 5. Four factors by effort, mean score out of 100

Model	Effort	Functional	Lint, complexity	Security	Code quality
Sol	Low	64.77	79.05	100.00	64.98
Sol	Medium	89.70	79.15	99.78	68.19
Sol	High	96.68	77.82	99.78	67.65
Sol	Extra-high	98.41	77.52	99.57	67.31
Sol	Max	99.68	77.43	99.55	67.58

Functional scores retain partial credit. Lint and complexity and security are the sweep's automated measurements. Code quality is the protocol's score. Means at Max cover the 22 judged runs.

The unpublished Max row

On codeccore at Max, Grok 4.6's intent probe quoted `memo = record[31:46].rstrip(".","")` on both attempts where the source line is `memo = record[31:46].rstrip(".,")`. A character is inserted inside the string literal; it is neither a re-wrap, an omission nor an escape spelling, so no recovery rule accepts it and none was added. No answer-key match was made and the frozen summary, which requires a valid match from every passing panel, publishes no Code quality or combined score for the run. Both judges' reviews and the Muse probe are retained in the harness archive. The run passed its tests, so the sweep itself passed 22 of 23 at Max; its runtime, tokens and cost are in every economics figure. The operator's finding is recorded beside the two attempts and its hash is in provenance.json.

Operator record

Under v3.7 Muse Spark 1.3 made 440 calls (80 in calibration, 115 primary reviews, 5 repeats, 10 pairwise checks, 115 intent probes, 115 answer-key matches) and Grok 4.6 made 439, with 114 matches. Muse needed a second attempt on nine calls: four unsupported excerpts and five malformed JSON responses. Grok needed a second attempt on nine: five unsupported excerpts (including the codeccore probe above), one malformed JSON probe, two transport faults retried under the network-fault rule, and one calibration probe the provider blocked before any model output, retried once under a new wrapper rule recorded in the operations log. Neither judge produced a reviewer fallback. Every attempt is archived beside its replacement in the harness run directory.

Cost and tokens across effort levels

The same 115 runs priced from their Codex receipts at list API rates, cache-aware, solver inference only, judging excluded. Sol ran on a ChatGPT Pro subscription, so these are API-equivalent estimates rather than bills. Cost and tokens do not follow the ladder: High is the most expensive and slowest level, and Extra-high and Max cost less than it while passing more tasks. Every run is priced, including the Max run without a published Code quality score.

Table 6. API-equivalent cost, raw tokens and runtime by effort

Effort	Runs	$/task	Level total	Tokens/task	Min/task
Low	23	$1.36	$31.26	1.79M	9.8
Medium	23	$1.72	$39.45	2.26M	11.1
High	23	$2.05	$47.04	2.83M	12.0
Extra-high	23	$1.61	$37.01	1.91M	10.6
Max	23	$1.77	$40.79	2.11M	11.6
Full sweep	115	$1.70	$195.55	251M	21.1 h

Rates checked 2026-09-11: Sol at $4.00 input, $0.40 cached input and $20.00 output per million tokens. Codex receipts report input, cached input, output and reasoning tokens per run, so cached input is billed at the cache-read rate and the rest at the standard rate. The sweep stamped each run at these rates at run time; the export recomputed every run from its receipt and matched every stamp. Tokens are raw solver totals including cache reads. Per-run estimates and the rate table are in economics.json and runs.csv.

Limitations recorded with the estimates

• Codex receipts report input, cached input, output and reasoning tokens per run; cached input is billed at the cache-read rate and the rest of the input at the standard rate. Cache writes are free on this API and are not modelled.

• Standard tier, short-context rates. Per-request context sizes are not exposed by the receipts, so no long-context premium is applied.

• No Batch, Flex, Fast or priority pricing is applied.

• The estimate covers the solver's exposed receipts, not an independently observed API invoice.

• The sweep stamped each run at run time at the published Sol list rates; the export recomputes every run from its receipt and matched every stamp.

Evidence and reproduction

Public files: `runs.json` and `runs.csv` with every run's factors, both judges' six-dimension sub-scores and intent recovery for each published run; `groups.json`; `calibration.json` with every gate value and control mean; `judge-protocols.json` with the exact rubric, system text, probe and match instructions, schemas, weights and the finding on the unpublished row; `economics.json` with cost and token aggregates, the rate table and pricing limitations; and `provenance.json` with source hashes and limits. https://github.com/morganlinton/VulcanBenchCOM/tree/main/assets/data/swe-v4-sol-v37

Withheld: raw prompts and responses; submitted patches and reconstructed sources; quirk answer keys (they describe hidden-test behaviour); reviewer session identifiers and usage receipts.

Recompute the published numbers

Each run's Code quality is (0.24 x reviewed score + 0.09 x intent recovery) / 0.33, where both terms are the equal mean of the two judges, or the reviewed score alone where the submission passed no quirk family. The combined score is 100 x (0.50 F + 0.085 Q + 0.085 S + 0.33 C / 100) with F, Q, S on a 0 to 1 scale. Group means weight the cell's judged tasks equally. The export script recomputes every row from the frozen summary, re-prices every run from its receipt, asserts that exactly one row is unpublished and that it is the row carrying the operator's invalid-probe marker, and refuses to write if any value differs.

What this report does not claim

No human rated anything; the scores are model judgment for a human reader, not human validation. Standard errors describe task sampling only. The measured-maintenance layer is unbuilt, so the 33 points are 24 reviewed plus 9 intent recovery. Runs were not repeated and effort labels are Codex's own. The Max cell's Code quality and combined score are means over 22 of 23 runs. Hash checks bind the export to frozen files; they do not prove that judges were unbiased or that no training overlap exists.

Source hashes

Frozen artifact	SHA-256
v3.7/summary.json	042ed949f87f33f4a79347fc4c183eae…
v3.7/protocol.json	6f78884f8164f60ad93e3f981cf61fb6…
v3.7/private-manifest.json	c22c68d47b339a8cc50f479d350fda4a…
v3.7/calibration-muse.json	274549630d7372653e53dca8ddcbcd3a…
v3.7/calibration-grok.json	da500c1fd8a1c7d96792d87064ff5919…
v3.7/calls/grok/probe/submission-023/operator-invalid.json	8762293d4d4268021c1c74b6bfccae6b…
comparison.json	d87d8e8ab958c098908078fe1edc3afe…

The protocol documents, controls, quirk-key format, runner and operator wrapper are in the VulcanBench repository under docs/judging and harness. The quirk answer keys themselves describe hidden-test behaviour and are not published.

Appendix. Per task at Low and Max effort

Task	Low CQ	Max CQ	Low combined	Max combined
blendcore	60.5	67.6	74.33	87.83
cellarcore	51.9	70.8	57.05	88.14
codeccore	76.8	unpublished	65.84	unpublished
depotcore	64.9	55.3	71.92	79.40
freightcore	78.0	72.7	90.98	89.30
granarycore	63.6	56.4	60.52	82.17
hedgecore	66.8	77.2	62.45	90.81
lodgecore	52.3	65.3	36.44	86.26
matchcore-order-book	68.8	68.9	76.61	87.65
metercore	65.7	70.6	87.11	88.55
pacecore	73.6	76.3	39.80	90.64
paddockcore	59.5	56.5	47.04	82.16
payrollcore	66.7	66.2	82.31	87.12
qlite-store	69.8	73.5	38.31	89.37
queuecore	64.4	57.9	67.79	84.07
quotacore	72.7	73.5	89.38	89.58
reflowcore	53.1	67.0	83.05	87.69
schedcore	64.5	67.9	80.31	87.29
settlecore	70.9	68.9	63.57	87.69
snapcore	53.0	64.8	62.98	86.83
stampcore	61.4	70.8	85.75	88.85
tallycore	70.3	62.9	88.54	86.13
vaultcore	65.2	75.6	76.04	90.41

Per-task values from runs.json. Task names are shortened for width. The codeccore Max run is unpublished for the reason given on page 4.

VulcanBench

September 2026

VulcanBench Frontier v4: GPT-5.6 Sol across effort levels

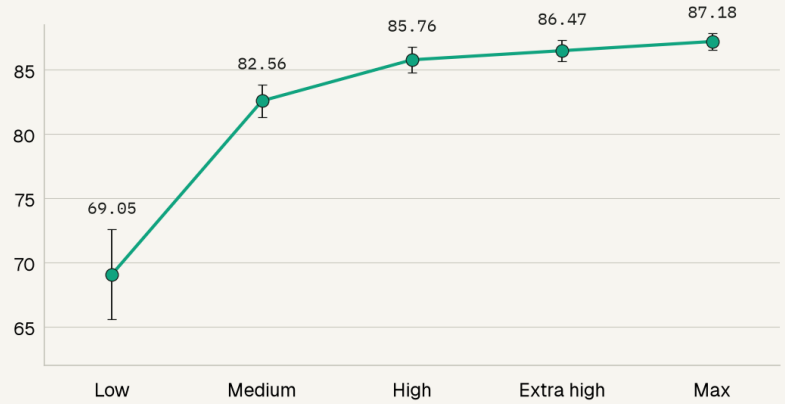
Combined score and runtime at every effort level, 23 tasks per effort. Code quality judged by Muse Spark 1.3 and Grok 4.6.

● **GPT-5.6 Sol · Codex**

n=23 at every effort, n=22 judged at max

Combined score

/100 · higher is better · focused scale



Mean runtime

Minutes per task · lower is better

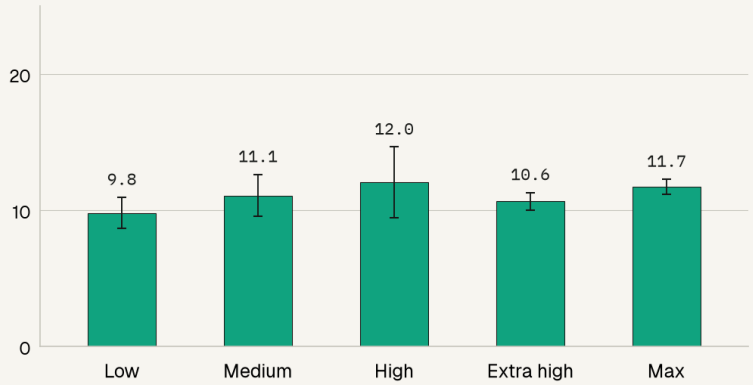


Table 1 | Code quality at each effort level

Component	Low	Medium	High	Extra high	Max
					/100, mean of both judges
Code quality	**64.98**	**68.19**	**67.65**	**67.31**	**67.58**
Human readability	57.4	61.2	58.8	59.2	59.8
Maintainability	66.9	69.8	67.8	67.8	68.3
Intent recovery	72.4	75.3	79.3	77.4	77.1
Rated by Muse Spark 1.3	60.2	64.3	62.0	61.8	63.0
Rated by Grok 4.6	64.1	66.8	64.6	65.3	65.1
Standard error of Code quality	1.56	1.30	1.17	1.39	1.39

Max is judged on 22 of 23 runs: on one task (codeccore) Grok 4.6's intent probe quoted an excerpt absent from the code on both attempts, so the protocol publishes no Code quality score for that run. The run itself passed its tests and is priced on the economics card.

VulcanBench

September 2026

VulcanBench Frontier v4: GPT-5.6 Sol across effort levels

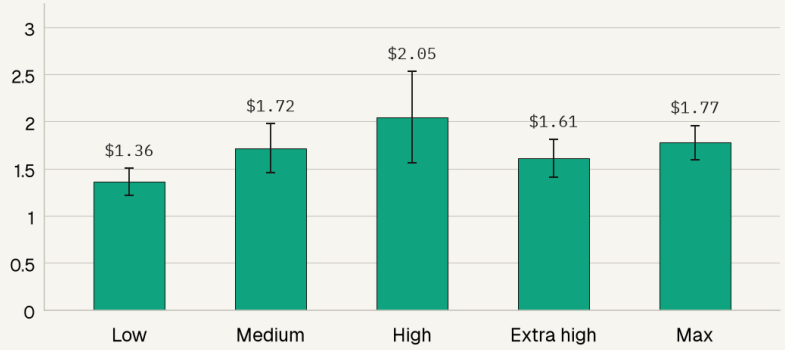
API-equivalent cost and token use at every effort level. All 115 runs of the sweep; solver inference only, judging excluded.

● **GPT-5.6 Sol · Codex**

n=23 at every effort

API-equivalent cost

USD per task at list prices · lower is better



Tokens per task

Millions of raw tokens, cache reads included · lower is better

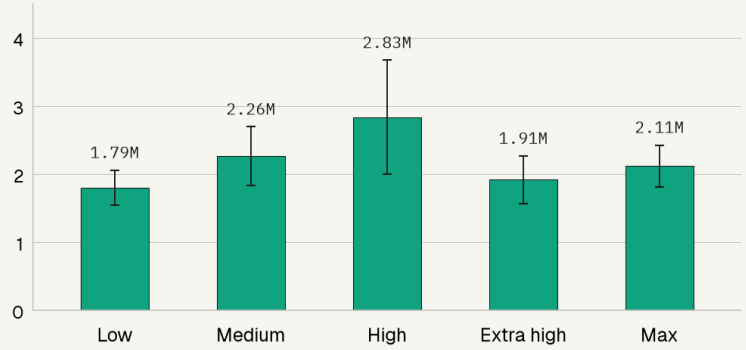


Table 1 | Cost, tokens and runtime at each effort level

Effort	Runs	$/task	Level total API-equivalent, list prices	Tokens/task raw, millions	Minutes/task solver wall-clock
Low	23	$1.36	$31.26	1.79M	9.8
Medium	23	$1.72	$39.45	2.26M	11.1
High	23	$2.05	$47.04	2.83M	12.0
Extra high	23	$1.61	$37.01	1.91M	10.6
Max	23	$1.77	$40.79	2.11M	11.6
Full sweep	**115**	**$1.70**	**$195.55**	**251M**	**21.1 h**

USD at list prices checked 2026-09-11, cache-aware (Sol $4.00 input, $0.40 cached input, $20.00 output per million), solver inference only; subscription bills differ.

All 115 runs are priced. The score card publishes 114: one Max run (codeccore) carries no Code quality score because a judge probe quoted an excerpt absent from the code on both attempts.

Whiskers are one task standard error. Tokens are raw solver totals including cache reads.