



The ML System Design Method

System design 01 | Nine steps for interviews and real projects: clarify, requirements, back-of-envelope estimates, data and labels, model, serving, evaluation, monitoring, iteration; a 45-minute time budget and the scoring rubric, on one running example (push-notification ranking)

#SystemDesign #Interview #Estimates #Method

The method in plain words

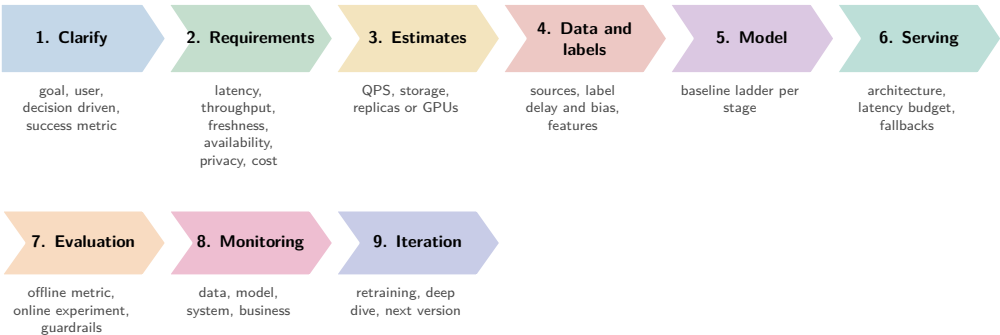
A doctor's consultation		An ML system design
Asking what hurts, since when, how much	≈	Clarifying questions and requirements
Weight, pulse, blood pressure	≈	Back-of-envelope estimates
Ordering the right tests	≈	Choosing data and labels
Diagnosis	≈	Model choice
Treatment plan and dosage	≈	Serving design and latency budget
Checking the treatment worked	≈	Offline and online evaluation
Follow-up visits	≈	Monitoring and iteration

WHERE THE ANALOGY BREAKS A patient reports back when it hurts; a model's users rarely complain, so you must build the measurements that tell you it is failing.

The nine steps work like a careful doctor's consultation

Prerequisites and scope

- Recall from mlops 01: requirements split into functional and non-functional; inference is batch, streaming or online; Little's law $L = \lambda W$ sizes concurrency. Not re-taught here.
- Metrics (machine-learning 07, metrics-and-kpis 03), A/B tests (machine-learning 16), serving and monitoring (mlops 04-05).



The nine steps, in interview order. In a project the same steps run over weeks and loop back

Steps 1-2: clarify and set requirements

Ask	Why it matters	Push example answer
What decision does the model drive?	defines the output and the label	send now, delay or drop each candidate notification
What is success for the business?	picks the online metric	daily active users without more opt-outs
How many users, items, events?	drives estimates and architecture	50 M daily users, ~20 candidates each
When is the answer needed?	batch, streaming or online	within seconds of the triggering event
What do errors cost?	sets thresholds and guardrails	a bad push costs an opt-out, which is permanent
Any constraints?	privacy, regulation, devices, explainability	consent per channel, quiet hours by time zone

Clarifying questions that change the design, with what each answer changes

Requirement	Write it as	Typical range
Latency	p99 under X ms at peak QPS	10-200 ms online; minutes for streaming
Throughput	peak requests/s or events/s	derive it in step 3
Freshness	max age of a feature or model	seconds (sessions) to a day (profiles)
Availability	share of time the SLO is met	99.9 % = 43 min down per 30 days

Cost	cost per 1,000 predictions or per user per month	set by the business case
Privacy	which data may be used, retention days	consent, minimisation, deletion
Non-functional requirements written as numbers you can test		

Step 3: back-of-envelope estimates

U = 50 M

daily users

a = 20

actions / user / day

p = 3

peak ÷ average

s = 1 KB

payload / action

D = 30 d

retention

W = 50 ms

latency / request

μ = 2 000 req/s

per replica

estimate	formula, numbers substituted	result
1 Average QPS	$\lambda = U \times a / 86\,400$ = 50 M × 20 / 86 400	11 574 req/s
2 Peak QPS	$\Lambda = \lambda \times p$ = 11 574 × 3	34 722 req/s
3 Storage / year	$S = U \times a \times s \times 365$ = 50 M × 20 × 1 KB × 365	365 TB 1 TB / day
4 Storage kept	$K = U \times a \times s \times D$ = 1 TB / day × 30 d	30 TB
5 Peak bandwidth	$B = \Lambda \times s$ = 34 722 req/s × 1 KB	34.7 MB/s 278 Mbit/s
6 Replicas	$N = \lceil \Lambda / \mu \rceil$ = $\lceil 34\,722 / 2\,000 \rceil$	18 replicas at peak, no headroom
7 In flight (Little)	$L = \Lambda \times W$ = 34 722 req/s × 0.05 s	1 736 requests concurrent at peak

Where U: daily active users (users/day); a: actions per user per day; 86 400: seconds per day; p: peak-to-average ratio; λ, Λ: average and peak requests per second; s: payload per action (KB, 1 KB = 1 000 B); S: storage per year; 365: days per year; B: peak bytes per second (× 8 for bits); D: retention (days); K: storage kept; N: replicas; μ: requests per second one replica serves; $\lceil x \rceil$ rounds up; W: latency per request (s); L: requests in flight (Little's law).

Push-notification ranker. Assumptions stated aloud: 50 M daily users, 20 candidate notifications scored per user per day, peak 3x average, 1 KB logged per score, logs kept 30 days, 50 ms budget per scoring call, one CPU replica scores 2,000 candidates/s (GBDT, batched)

Quantity	Formula	Rule of thumb
Average QPS	$\bar{q} = \frac{U \cdot a}{86\,400}$	1 M events/day $\approx$ 12 /s
Peak QPS	$q_{peak} = f \cdot \bar{q}$	f = 2-5 for daily cycles
Storage	$S = U \cdot a \cdot s \cdot d$	1 KB × 1 B/day = 1 TB/day
Replicas	$R = \lceil q_{peak} / (t \cdot \rho) \rceil$	ρ = 0.5-0.7 target utilisation

where U = daily active users (users/day); a = actions (requests, scores) per user per day (count); f = peak factor, peak divided by average traffic (ratio); s = size of one logged record (bytes); d = retention (days); t = throughput of one replica at the latency target (requests/s); ρ = target utilisation of a replica (0-1)

The four formulas behind every estimate (redo them by hand: 50 M × 20 / 86,400 $\approx$ 11,600 /s; peak $\approx$ 35,000 /s; 1 TB/day; $\approx$ 18 replicas before headroom)

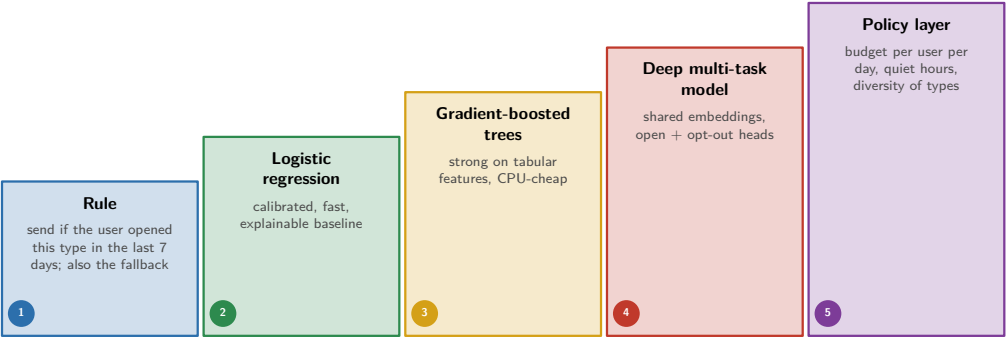
💡 Numbers worth memorising

86,400 s per day ($\approx 10^5$). 1 M requests/day $\approx$ 12 /s average. A GBDT scores thousands of rows/s per CPU core; a mid-size transformer ranker hundreds to a few thousand per GPU. Memory read $\approx$ 100 ns, same-zone network round trip $\approx$ 0.5 ms, cross-region $\approx$ 50-150 ms. Use them as orders of magnitude, then measure.

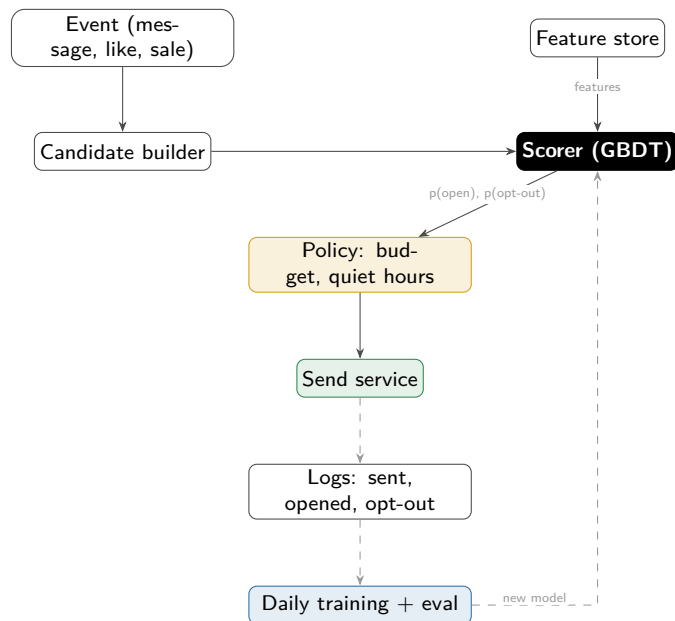
📖 Steps 4-6: data, model, serving

Source	Delay	Bias to name	Push example
Implicit (clicks, opens)	seconds-minutes	only shown items get labels; position and time effects	opened within 1 h
Explicit (ratings, reports)	minutes-days	few users give them; negative ones are rarer	"turn off this type"
Outcome (purchase, churn)	days-months	confounded by everything else	active 7 days later
Human annotation	days	guidelines drift, cost per label	rarely needed here

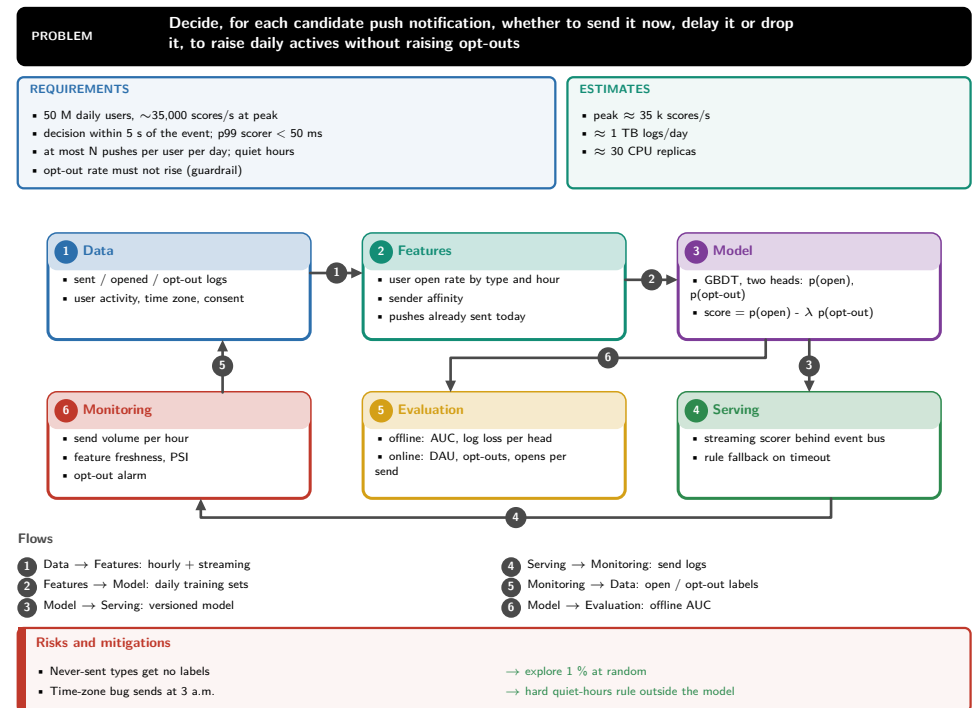
Label sources: pick one per objective and say its delay and its bias



Baseline ladder: climb one rung only when the previous one is measured and beaten



Architecture of the running example: online path (solid) scores within seconds; learning path (dashed) runs daily



The running example on one canvas: what a strong candidate has on the board after 35 minutes

✓ Steps 7-9: evaluate, monitor, iterate

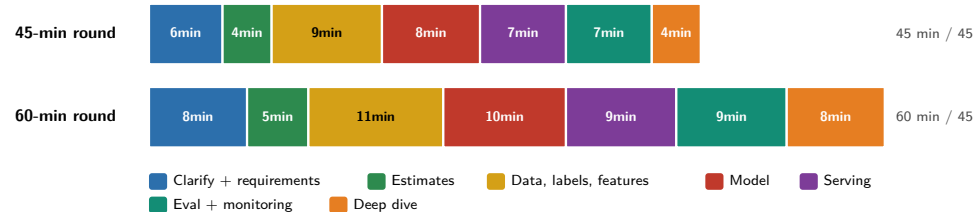
Rung	What you measure	Catches
Offline on a time split	AUC, log loss, NDCG on the most recent period	bugs, leakage, weak features
Replay or counterfactual	inverse-propensity estimate of the new policy on old logs	bias from logged decisions (needs logged probabilities)
Shadow	new model scores live traffic, decisions not used	latency, skew between training and serving features
A/B test	main metric + guardrails with confidence intervals	real user behaviour, novelty effects
Long-term holdout	a small group kept on the old system for months	slow harms: fatigue, opt-outs, churn

Evaluation ladder: each rung costs more and is closer to the truth

Layer	Signals	Push example alert
System	p99 latency, errors, fallback share	fallback share > 5 % for 10 min
Data	nulls, schema, freshness, PSI	open-rate feature older than 2 h
Model	score distribution, calibration, label-based AUC	mean p(open) shifts > 20 % day on day
Business	main metric, guardrails, segments	opt-outs up in one country

Monitoring in four layers; alert on the first two, review the last two weekly

🕒 The interview: time, scoring, answer



How to spend a 45-minute round (minutes). The deep dive goes where the design risk is, not where you feel safest

TARGET				
Criterion	Weight	Weak 1 pt	Pass 2 pt	Strong 3 pt
Problem framing	15%	jumps to a model	asks some questions	ties metric to the decision and business
Data and labels	20%	"we have data"	names sources	delay, bias, leakage, feedback loops
Modelling	20%	one fancy model	baseline + one upgrade	per-stage choices with trade-offs
Serving and scale	15%	no numbers	rough QPS	estimates, latency budget, fallbacks
Evaluation	15%	accuracy only	offline metric + A/B	guardrails, offline-online gap
Monitoring, iteration	10%	absent	mentions drift	alerts, retraining, exploration
Communication	5%	monologue	structured	drives, checks in, summarises

Total weight: 100%

What interviewers score (typical criteria; weights vary by company and level)

SENIOR ML ENGINEER

Q. Design the system that decides which push notifications to send.

Structure

1 Clarify goal and guardrail

2 Requirements and estimates

3 Labels and features

4 Model and policy layer

5 Serving and fallback

6 Evaluate, monitor, deep dive

Model answer

Goal: raise daily actives without raising opt-outs. ~35,000 scores/s at peak, decision within seconds. Label p(open within 1 h) and p(opt-out within 24 h) from send logs. GBDT with two heads, score = p(open) - λ · p(opt-out), then a policy layer: daily budget per user, quiet hours, type diversity. Streaming scorer with a rule fallback. Offline AUC and log loss on a time split, then an A/B test on DAU with opt-outs as guardrail and a long-term holdout for fatigue.

Follow-ups

Q: How do you choose λ?

A: Sweep it in A/B arms or with offline replay; pick the value that maximises DAU subject to opt-outs not rising beyond the guardrail.

Q: A new notification type launches. How is it scored?

A: Cold start: share features across types (sender, hour), give it a small exploration budget, and fall back to rules until labels accumulate.

Q: Opens rise but uninstalls rise too. What happened?

A: The proxy (opens) diverged from the goal; add uninstall and opt-out to the score or the guardrails and check the long-term holdout.

Red flags

- no guardrail on fatigue
- random train/test split
- no fallback when the scorer is down

79 words · ≈ 36 s spoken

t = 60 · w / r; speaking time t (s) of the model answer, w: its words, r: speaking rate (130 words/min). Aim for 1-2 min, then stop and invite follow-ups.

The opening of a strong answer, then the follow-ups interviewers ask on this design

💡 Traps and recap

YOU MAY HEAR

"More components show more knowledge"

BUSTED

ACTUALLY

Each box must trace to a requirement

Interviewers ask why each box exists; an unjustified feature store or vector database loses credibility.

YOU MAY HEAR

"Estimates are a formality"

BUSTED

ACTUALLY

They decide the architecture

35,000 scores/s rules out a large transformer per candidate on CPU; 12 requests/s allows almost anything.

YOU MAY HEAR

"The best offline model ships"

BUSTED

ACTUALLY

The A/B test with guardrails decides

Logged data reflect the old policy; a model can win offline and lose online.

YOU MAY HEAR

"Monitoring means watching accuracy"

BUSTED

ACTUALLY

Labels are often late; monitor inputs and outputs first

Feature freshness, PSI and score distribution alert days before label-based metrics move.

Method myths that cost points

Tick a box only when you could explain the point to someone else

☐ Nine steps, always in order

clarify → iteration

☐ Estimate with stated assumptions

QPS, storage, replicas

☐ Climb the baseline ladder

rule first, it is also the fallback

☐ Spend the deep dive on the riskiest part

and summarise at the end

☐ Requirements as testable numbers

p99, QPS, freshness, availability

☐ Name label source, delay and bias

for every objective

☐ Evaluate offline, then online with guardrails

watch the offline-online gap

Tick each point only when you could explain it to an interviewer

✓ Self-test

Q1 50 M users x 20 events/day: average QPS?

A $50\text{ M} \times 20 / 86,400 \approx 11,600 / \text{s}$; with a peak factor of 3, about 35,000 /s.

Q2 Peak 35,000 /s, 2,000 /s per replica, 60 % target utilisation: replicas?

A $35,000 / (2,000 \times 0.6) \approx 29.2 \rightarrow 30$ replicas (plus one zone's worth for failover).

Q3 Why name the label delay?

A It sets how fast you can evaluate and retrain, and which proxy labels and monitors you need meanwhile.

Q4 What is the first rung of the baseline ladder and why keep it?

A A rule or heuristic: it sets the bar to beat and serves as the fallback when the model fails.

Q5 Why a long-term holdout?

A Some harms (fatigue, churn) appear after weeks; a short A/B test misses them.

Q6 Where should the deep dive go?

A To the component with the highest risk or uncertainty, usually labels, the ranking objective or latency.

Q7 What does 99.9 % availability allow per 30 days?

A About 43 minutes of SLO violation (0.1 % of 43,200 minutes).

Q8 Name three red flags in a design answer.

A No clarifying questions, random split on time-dependent data, no fallback or monitoring.

Q9 Why does a push ranker need a policy layer outside the model?

A Hard constraints (budgets, quiet hours, consent) must hold even when the model is wrong.