Engineering Cheat Sheet

LATENCY — ORDERS OF MAGNITUDE

L1 cache reference	~1 ns
Main memory reference	~100 ns
Read 1 MB from RAM	~100 µs
SSD random read	~100 µs
Same-datacenter round trip	~0.5 ms
Redis GET (same DC)	~0.5–1 ms
Read 1 MB from SSD	~1 ms
Postgres indexed read	~1–5 ms
HDD seek	~10 ms
Cross-region RTT (US–EU)	~80–150 ms
Cross-continent (US–APAC)	~150–250 ms
Human "instant" threshold	~100–200 ms

Memory is ~100× faster than SSD; SSD ~10–20× faster than disk; **the network dominates everything**. A cross-region hop costs more than 1,000 local disk reads — that is why replication topology, not code, decides p99.

BACK-OF-ENVELOPE FORMULAS

- Seconds/day ≈ 86,400 → **round to 100k**
- 10M requests/day ≈ **100 QPS** average
- Peak ≈ **2–5× average** (spiky consumer: 10×)
- DAU → QPS: `DAU × actions ÷ 100k`
- Storage: `items/day × bytes × days × 3` (replication)
- Bandwidth: `QPS × payload`. 10k × 10 KB ≈ 0.8 Gbps
- Cache 80/20: hot 20% of the working set serves ~80% of reads

Round aggressively. The number justifies a choice; it is not the answer.

CONSISTENCY & THE TRADE YOU CAN'T DODGE

CAP	On partition, choose Consistency *or* Availability. Partitions are not optional.
PACELC	The honest version: **E**lse (no partition) you still trade **L**atency vs **C**onsistency — every day, not just in a crisis.
Strong	Reads see the last write. Costs a quorum round trip.
Eventual	Replicas converge, eventually. Cheap, and surprising to users.
Read-your-writes	The one users actually notice. Usually enough.

EXACTLY-ONCE IS A MYTH End-to-end exactly-once delivery does not exist. What ships: **at-least-once delivery + an idempotent consumer** keyed on a dedupe ID. Anyone selling exactly-once is selling this.

SCALING MOVES, IN THE ORDER YOU MAKE THEM

- **Vertical** — bigger box. Buys a year, has a ceiling.
- **Stateless + LB** — horizontal, only once state is out of the box.
- **Cache** — the highest leverage per hour of work.
- **Read replicas** — reads scale; writes still don't.
- **Async / queue** — take work off the request path.
- **Shard** — last, and hardest to undo. Choose the key carefully.

Shard key rule: pick the field you filter by. Wrong key = every query is a scatter-gather, and the fix is a migration.

CACHING — THE THREE FAILURES

Stampede	Key expires, 10k requests hit the DB at once → lock, or early-recompute.
Penetration	Queries for keys that don't exist → cache the negative, or Bloom filter.
Avalanche	Many keys expire together → jitter the TTLs.

Invalidation: TTL is not a strategy, it is a delay. Write-through for correctness; cache-aside for simplicity — but then two writers can race, so version the entry.

DATA-STRUCTURE REACHES

Bloom filter	"Definitely not present" in ~1 byte/item. No false negatives.
HyperLogLog	Count distinct in ~1.5 KB, ~2% error. Unique visitors.
Count-min sketch	Heavy hitters / trending, fixed memory.
Consistent hash	Add a node, move 1/n keys — not all of them.
LSM tree	Write-optimized (Cassandra). B-tree = read-optimized (Postgres).

FAILURE IS THE DEFAULT STATE

- **Timeout** everything. No timeout = a hang that becomes an outage.
- **Retry with backoff + jitter.** Naked retries synchronize and DDoS you.
- **Circuit breaker** — stop calling the dead thing; fail fast.
- **Bulkhead** — one pool per dependency, so one slow call can't eat the threads.
- **Idempotency key** on every mutating endpoint. The retry is coming.

p99 is the number users feel. An average latency of 50 ms with a p99 of 3 s means 1 in 100 requests is a bad review.

KEYS → ADDRESS (WHY IT ALL WORKS)

```
entropy → BIP-39 words → PBKDF2(2048) → seed
  → BIP-32 master key → BIP-44 path
     m/44'/60'/0'/0/0
  → pubkey → keccak256(pub)[12:] = address
```

- **Passphrase enters as PBKDF2 salt** → a completely different wallet. No checksum protects it; a typo yields a valid empty wallet, not an error.
- Same key, **same address on every EVM chain** — only chainId (EIP-155) separates networks.
- Address = hash of pubkey → **quantum-safe until you spend**. The first signature publishes the pubkey, permanently.

THE STANDARDS THAT MATTER

EIP-155	chainId replay protection — the cheque with the country on it
EIP-191	personal_sign — `0x19` makes messages ≠ transactions
EIP-712	typed data — signing a labeled form, not a napkin
ERC-1271	`isValidSignature` — ask the company's front desk
ERC-6492	signatures for wallets not yet deployed
EIP-2612	`permit()` — a signed letter of authorization
ERC-4337	account abstraction via an alt-mempool
EIP-7702	your EOA runs code (tx type 0x04)
ERC-7579	modular accounts — the app store API

Engineering Cheat Sheet

WHAT CHANGES IN PRODUCTION

- The model is **non-deterministic; your system can't be.** Constrain, validate, measure — don't try to eliminate variance.
- **Failures are silent.** Code crashes loudly; an LLM returns something plausible and wrong. Without traces you don't know you're failing.
- The demo distribution is not the real one. Production brings typos, other languages, and adversaries.
- **Cost and latency are features.** A perfect answer in 45 s at $0.30 loses to a good one in 2 s at $0.002.
- **Prompts are code.** Version, review, roll back. A prompt edit is a deploy.

```
PRODUCT       chat · API · jobs
ORCHESTRATION workflows · agents · HITL
CONTEXT       prompts · memory · RAG · tools
MODEL         routing · fallback · caching
QUALITY ————— evals · tracing · guardrails
              (crosscutting, not an afterthought)
```

Read the stack **bottom-up when debugging** ("why is quality bad?"), **top-down when designing**.

CONTEXT ENGINEERING

- **Context rot:** effective attention degrades as the window fills. Recall drops, instructions get ignored.
- **Lost in the middle:** content in the middle is recalled worst. Put rules at the *end*.
- Irrelevant context isn't neutral — **it actively hurts**. A plausible distractor is worse than nothing.
- Goal: the **smallest set of high-signal tokens** that solves the task. More context is a cost, not an upgrade.

Compact at ~65% of the window, not 95% — compaction is itself a model call and needs room to run. Preserve decisions, constraints, IDs, open questions. Discard pleasantries and raw tool dumps.

RAG VS STUFFING VS FINE-TUNING

Stuff it	Corpus < ~50K tokens, stable, needed whole. Cache it and repeated calls get cheap.
RAG	Large, changing, per-tenant, or needs access control. Retrieval quality becomes your ceiling.
Fine-tune	Teaches *style/format*. Terrible for facts — stale instantly, no citations, no access control.

The production answer is the hybrid: stuff the stable thing (cached), retrieve the specific thing.

THE FILTER IS THE WHOLE BALLGAME Vector search without a tenant filter returns the nearest neighbour **from another customer's documents** — semantic similarity knows nothing about tenancy. The model then launders it into a fluent, well-cited answer. Nothing in your logs looks like a breach; it looks like a good response.

WORKFLOW OR AGENT?

Workflow	You know the steps. Code owns control flow; the model fills in the gaps. Predictable, testable, cheap. **Start here.**
Agent	The model chooses the steps and the tools. Use only when the path genuinely can't be known up front.

"Agent" is not a maturity level. Most shipped features are workflows, and the ones that aren't are usually harder to debug than the problem they solved.

- **Cap the loop.** Max turns, max spend, max wall-clock. Every one of them.
- **Every tool is idempotent** or takes a dedupe key — the model *will* retry.
- **HITL on the irreversible.** Refunds, sends, deletes. Reversible = let it run.
- Watch **turns per task**. Creeping up = context rot or a tool regression.

COST & LATENCY LEVERS, BY ROI

1. **Prompt caching** — order stable→volatile. ~1.25× to write, ~0.1× to read. One field moved, ~90% off.
2. **Cascade** — cheap model first, escalate on low confidence. ~80% saved at a 15% escalation rate.
3. **Context diet** — cost bugs are almost always context-bloat bugs.
4. **Output caps** — output tokens cost more than input.
5. **Batch** anything not user-facing (~50% off).

Silent cache killers: a timestamp or UUID early in the prompt; unsorted `JSON.stringify` keys; a prefix below the model minimum (~2048 tok Sonnet / ~4096 Opus) — the marker is accepted and does nothing. Verify with `cache_read_input_tokens`; if it stays 0, your prefix isn't stable.

GUARDRAILS & EVALS

- **Guards are code, not prompts.** "Please don't reveal card numbers" is a suggestion; a Luhn check that blocks the response is a control.
- **Lethal trifecta** — never combine in one agent: private data + untrusted content + ability to act/exfiltrate. Break it architecturally.
- The model **proposes**; your code **disposes**. Limits and ownership checks live in the executor, never in the prompt.
- **Evals before vibes.** Ship the eval set with the feature; LLM-as-judge needs its own calibration.

The test: "Assume the model does exactly what the attacker asked — what stops it?" If the answer names a prompt, you have no control. If it names a schema bound, an ownership check, or a human, you do.

ERC-4337 — THE CAST

The restaurant: a **UserOperation** is an order slip. The **bundler** is the waiter collecting slips and firing them at once. The **EntryPoint** is the kitchen pass — every plate crosses it. Your **smart account** is a membership card with its own rules. The **paymaster** is the company card behind the bar tab.

- Account decides **if the op is valid** · paymaster decides **who pays** · bundler **gets it on-chain**.
- A UserOp is **not a transaction** — it rides a separate mempool.
- **7702 + 4337 is the 2026 architecture**: your existing address, delegated to a modular implementation, with sponsored gas.

SIGNATURE VERIFICATION

ECRECOVER FAILS OPEN Hand-rolled `recoverMessageAddress` works for an EOA and is **wrong for every contract wallet** (Safe, Coinbase Smart Wallet, any 4337 account). It doesn't error — it returns a confident wrong address. Use `verifyMessage`, which covers EOA + 1271 + 6492. And no nonce = a captured signature is a permanent login.

L2 & THE 7-DAY NUMBER

- **Rollup** = execute elsewhere, post data + commitment to L1 → inherits Ethereum's security.
- **Sidechain** = its own validators; only a bridge touches L1 → you trust them, fully.
- Optimistic withdrawal **~7 days** — that is the fraud-proof window, i.e. **load-bearing security**, not a delay to optimize. "Instant withdrawal" = a liquidity provider fronting you and taking the wait.
- ZK: minutes–hours; L1 verifies math, not people.

7702 CHAINID 0 Signs a delegation valid on **every chain, forever** — including chains that don't exist yet, where the same address may hold different code. Scope it.