

Practice sheet

A 45-minute rehearsal. Make the design choices before you compare your reasoning with a worked answer.

The 45-minute plan

TIME	STEP	WHAT TO SAY OR DRAW
5 min	Clarify the problem	Who uses it? What must it do? What quality target matters most? State the assumptions the design depends on.
5 min	Estimate the load	Work out average and peak requests, data growth, and read-to-write ratio. Use the numbers to identify the first constraint.
10 min	Draw the main paths	Show clients, APIs, stores, caches, and asynchronous work. Trace one request and one write from start to finish.
15 min	Choose one hard part	Compare two plausible designs for the hardest requirement. Pick one and name the cost you accept.
5 min	Break and change it	Trace a crash, duplicate message, timeout, or traffic spike. Then change one requirement and adapt the design.
5 min	State the guarantees	Say what the design guarantees, where it can fail, the first likely bottleneck, and what you would change next.

Quick capacity math

Average requests/s	requests per day / 86,400
Peak requests/s	average requests/s x peak factor
Raw storage per year	writes per day x bytes per record x 365
Example	10 million requests/day is about 116 requests/s on average. At a 5x peak, plan for about 580 requests/s.

Write the estimate that changes your design: _____

Assumptions I need to say out loud

1. _____
2. _____

Five questions, five design challenges

Practise one question each day, or choose the one that exposes a gap in your reasoning.

Choose a system

QUESTION	WHAT THE INTERVIEWER CAN TEST
Design a URL shortener	Estimate read-heavy traffic. Handle unique codes, fast redirects, cache invalidation, and click counting off the redirect path.
Design an API rate limiter	Compare fixed windows, sliding windows, and token buckets. Decide how servers share counters and what happens when the limiter fails.
Design a news feed	Choose where fan-out happens. Handle celebrity accounts, inactive readers, deleted posts, and a feed assembled from multiple sources.
Design a payment system	Keep an order correct when a charge times out after succeeding, a retry arrives, or provider webhooks arrive twice or out of order.
Design a distributed job queue	Make accepted work durable. Handle worker crashes, duplicate delivery, retries, poison messages, and growing backlogs.

Before you finish

☐ I clarified requirements before drawing boxes.	☐ I used at least one estimate to justify a design choice.
☐ I traced a request through each service and data boundary.	☐ I compared an alternative and named its cost.
☐ I explained how the design recovers from a failure or duplicate.	☐ I stated the guarantee precisely and named what can still go wrong.

After the timer

Compare your assumptions, tradeoffs, and failure handling with a complete walkthrough. Then repeat the question later and see whether you can explain the design more clearly.

Free interactive questions and worked answers: sysgeeks.com/investigations