What Is a System Design Interview?

Learn about system design interviews (SDIs) and how to strategically approach them.

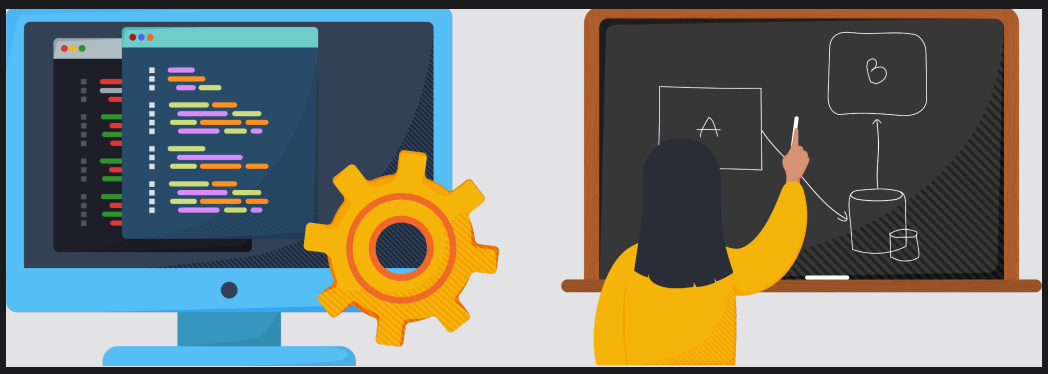
We'll cover the following

Our system design course is equally useful for people already working and those preparing for interviews. In this chapter, we highlight the different aspects of a system design interview (SDI) and some helpful tips for those who are preparing for an upcoming interview. We encourage learners to read this chapter even if they aren’t preparing for an interview because some of the topics covered in this chapter can be applied broadly.

How are SDIs different from other interviews?

Just like with any other interview, we need to approach the system design interviews strategically. SDIs are different from coding interviews. There’s rarely

any coding required in this interview.



Other interviews versus a system design interviews

An SDI takes place at a much higher level of abstraction. We figure out the requirements and map them on to the computational components and the high-level communication protocols that connect these subsystems.

The final answer doesn't matter. What matters is the process and the journey that a good applicant takes the interviewer through.

> **Note:** As compared to coding problems in interviews, system design is more aligned with the tasks we'll complete on our jobs.

How do we tackle a design question?

Design questions are open ended, and they're intentionally vague to start with. Such vagueness mimics the reality of modern day business.

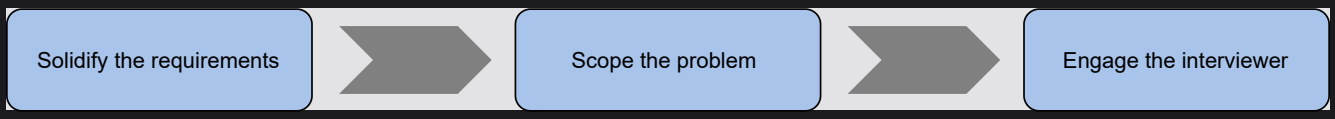
Interviewers often ask about a well-known problem,—for example, designing WhatsApp. Now, a real WhatsApp application has numerous features, and including all of them as requirements for our WhatApp clone might not be a wise idea due to the following reasons:

- First, we'll have limited time during the interview.

- Second, working with some core functionalities of the system should be enough to exhibit our problem-solving skills.

We can tell the interviewer that there are many other things that a real WhatsApp does that we don't intend to include in our design. If the interviewer has any objections, we can change our plan of action accordingly.

Here are some best practices that we should follow during a system design interview:



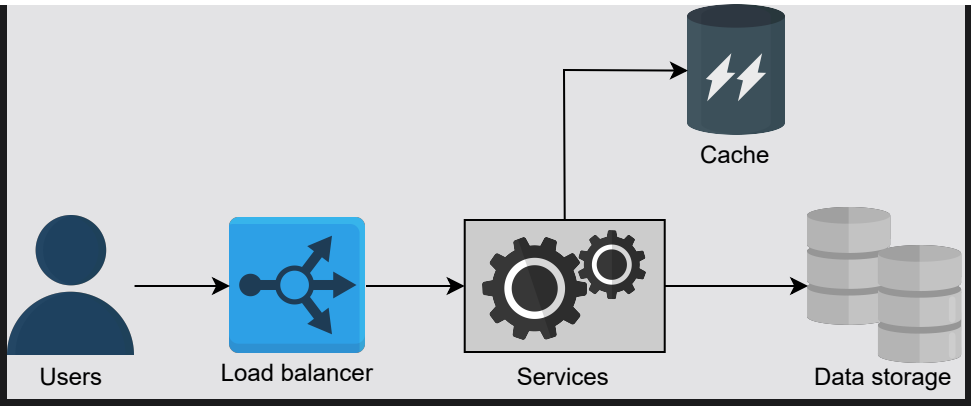
Best practices for system design interview

- An applicant should ask the right questions to solidify the requirements.
- Applicants also need to scope the problem so that they're able to make a good attempt at solving it within the limited time frame of the interview. SDIs are usually about 35 to 40 minutes long.
- Communication with the interviewer is critical. It's not a good idea to silently work on the design. Instead, we should engage with the interviewer to ensure that they understand our thought process.

Present the high-level design

At a high level, components could be frontend, load balancers, caches, data processing, and so on. The system design tells us how these components fit together.

An architectural design often represents components as boxes. The arrows between these boxes represent who talks to whom and how the boxes or components fit together collectively.



A sample design

We can draw a diagram like the one shown above for the given problem and present it to the interviewer.

Possible questions for every SDI

SDIs often include questions related to how a design might evolve over time as some aspect of the system increases by some order of magnitude—for example, the number of users, the number of queries per second, and so on. It's commonly believed in the systems community that when some aspect of the system increases by a factor of ten or more, the same design might not hold and might require change.

Designing and operating a bigger system requires careful thinking because designs often don't linearly scale with increasing demands on the system.

Another question in an SDI might be related to why we don't design a system that's already capable of handling more work than necessary or predicted.

Show the answer

The design evolution of Google

The design of the early version of Google Search may seem simplistic today, but it was quite sophisticated for its time. It also kept costs down, which was necessary for a startup like Google to stay afloat. The upshot is that whatever we do as designers have implications for the business and its customers. We need to meet or exceed customer needs by efficiently utilizing resources.

educative

Things will change, and things will break over time because of the following:

- There’s no single correct approach or solution to a design problem.
- A lot is predicated on the assumptions we make.

The responsibility of the designer

As designers, we need to provide fault tolerance at the design level because almost all modern systems use off-the-shelf components, and there are millions of such components. So, something will always be breaking down, and we need to hide this undesirable reality from our customers.

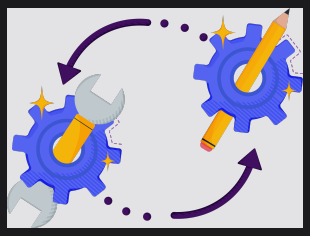
Who gets a system design interview?

Traditionally, mid-to-senior level candidates with more than two years of experience get at least one system design interview. For more senior applicants, two or three system design interviews are common.

Recently, large companies have also put forth system design questions to some junior candidates. It’s never too early to learn system design to grow or even expedite our careers.

Theory and practice

Most of the theory of system design comes from the domain of distributed systems. Getting a refresher on these concepts is highly recommended. Educative has an excellent course on distributed systems that we can use to refresh our knowledge of distributed systems concepts.



Distributed systems give us guideposts for mature software principles. These include the following:

- Robustness (the ability to maintain operations during a crisis)
- Scalability
- Availability
- Performance
- Extensibility
- Resiliency (the ability to return to normal operations over an acceptable period of time post disruption)

Such terminology also acts as a lingua franca between the interviewer and candidate.

As an example, we might say that we need to make a trade-off between availability and consistency when network components fail because the CAP theorem indicates that we can't have both under network partitions. Such common language helps with communication and shows that we're well versed in both theory and practice.

> **Remember:** It's a candidate's job to exhibit their skills to the interviewer.