3 Ways to Ace Your System Design Interview

Posted by *Sam Gavis-Hughson*

System design interview questions are one of the least understood type of any type of question out there. The reason why people get so hung up on them is that there is so much that you need to know but also very little information out there about how companies actually evaluate these questions. In my opinion there are 3 things you can do to ace your system design interview questions and it all starts with a good strategy and approach.

Thankfully, companies are not going to judge your entire interview based solely on how you do with the system design questions. You will do the technical interview as well, so system design is just going to be one piece of the puzzle. If you bomb system design it might hurt a little. But if you do really well it will only help you a little. It will not make or break your interview. That's one big thing we can get out of the way upfront and it should be comforting to you.

Know what to prepare

Preparing for the system design interview is really important. However, how do you prepare effectively when there's an infinite amount of information that you could know?

For example in thinking about how many different database technologies there are out there. There's no possible way that you could know all of them and it's really not necessary. How then do you determine what you do and don't need to know?

The key to this is to start with the knowledge that you already do have. Chances are good that you won't be asked to a system design interview if you're fresh out of college. If you're answering these questions, you've probably been working for several years as a software engineer already and if you're at that point then you've been using these technologies already. You've used database technologies, software design patterns, and you can draw on these in your system design interview.

The key with studying is to start with what you know and then fill in the gaps. Let me explain. There are broad categories of different things that you need to know about for your system design interview. There may be load balancers, message handlers, database technologies, etc. The key is to know at least one thing in each of these larger areas. If we take message handlers for endings for example, you might have Kafka,

RabbitMQ, or a million other things. But you don't need to know all of those. You just need to know generally what a message handler is and how it works. After that you just need to know one specific technology that you can use in your interview.

This allows you to focus your efforts efficiently in this one area so that you don't spend too much time covering this one small topic. It will give you more time to cover everything else in your interview. Another good thing to know for these broader categories is to understand what are the trade-offs. If you are using different database technologies what are the trade-offs between a SQL and a non-SQL database?

As long as you know what those are and can talk intelligently about them that is the most important thing in your interview. Not that you pick the perfect technology, because everyone's going to have a different experience. If you go through all of these broad categories of technologies that you need to know and know something from each one, you will be successful in your system design interview.

Understand the interview question

The second thing that you should do in preparing for your system design interview is to really understand the problem. I don't mean just knowing the problem and knowing how to solve it, I mean deeply understand the problem. This means understanding the constraints, who the users are and the size of the user base. Understanding what something is being used for makes it so much easier for you to make assumptions about what all of these other constraints need to be.

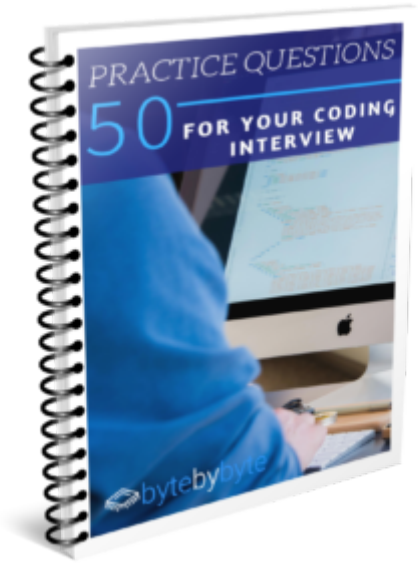
As an example we could take a messaging app like Facebook Messenger. In this example you are going to have billions of users who are only messaging their friends, but they are using it to share multimedia, links, all sorts of things like that. This is one sort of messaging app. Another example is a messaging app for doctors in hospitals. Security is really important but you don't need to share multimedia.

If you really understand what is the core use case, you can define the priorities of your design and where you are going to focus your efforts as efficiently as possible. This will also help eliminate the need to ask lots of questions to your interviewer because you can already make reasonable assumptions based on how this is going to be used.

Breaking down the problem

The last thing you can do to prepare is to understand how to breakdown the problem. You don't necessarily need to know exactly what they are looking for since every company is looking for something slightly different. One company may be looking for a high level design, one may want you to actually write what all the objects are going to look like for some sort of service you're building, and one might want to focus on what the data store is going to look like. You won't know until you are actually in your system design interview which is why the most important thing you can do is to really breakdown the problem and engage with your interviewer to figure out what they're looking for.

Going into the interview you need to really understand the problem and then break it into pieces. You are going to have the data store, server, and message handler. Using all of these different components you will start with a broad picture of what it will look like. Then from there you can engage with your interviewer to figure out where they want you to go deeper. It is so important to engage and talk to the person conducting the interview. You can say, okay I have this large structure, where would you like me to go deeper? They may tell you to choose and then you can pick whatever you think is going to be the most interesting or easiest for you. Or they may have something specific in mind that they want you to do. Breaking down the problem not only helps you to understand it a lot better but it makes it much easier for your interviewer to get what they want out of the interview.



Sam Gavis-Hughson

Sam, founder of Byte by Byte, helps software engineers successfully interview for jobs at top tech companies. Sam has helped thousands of students through his blog and free content -- as well as 400+ paying students -- land jobs at companies such as Google, Amazon, Microsoft, Bloomberg, Uber, and more.

← Finding a brute force solution

FAANG Interview – Big 5 Coding Prep →

The only 10% of information you need to know to ace your interview – forget all the useless fluff

Enter your email below and get instant access to your free Dynamic Programming guide.

GET THE FREE GUIDE

RECENT POSTS

How To Nail the Amazon Interview: A Practical Guide

The Ultimate Guide to Dynamic Programming

Behavioral Interviews for Software Engineers

Acing the Google Interview: The Ultimate Guide

INTERVIEW CAKE

Interview Cake is an awesome resource for more practice interview questions. Get 50% off for a limited time.

CRACKING THE CODING INTERVIEW

Check out my hands down favorite resource for coding interview prep here.

sam@byte-by-byte.com

Youtube

Made in NYC

Privacy Policy
Terms and Conditions

Sam Gavis-Hughson is a software engineer based in New York City. Through Byte by Byte, he publishes regular coding interview question videos, demonstrating proper interview techniques. He also helps many students by offering practice coding interviews to help them get jobs at Google, Facebook, and other exciting tech companies.