Sunday, 18 February 2024 2:36 AM

System Design Interviews

System design is a very extensive topic and system design interviews are designed to evaluate your capability to produce technical solutions to abstract problems, as such, they're not designed for a specific answer. The unique aspect of system design interviews is the two-way nature between the candidate and the interviewer.

Expectations are quite different at different engineering levels as well. This is because someone with a lot of practical experience will approach it quite differently from someone who's new in the industry. As a result, it's hard to come up with a single strategy that will help us stay organized during the interview.

Let's look at some common strategies for system design interviews:

Requirements clarifications

System design interview questions, by nature, are vague or abstract. Asking questions about the exact scope of the problem, and clarifying functional requirements early in the interview is essential. Usually, requirements are divided into three parts:

1. **Functional requirements**

   These are the requirements that the end user specifically demands as basic functionalities that the system should offer. All these functionalities need to be necessarily incorporated into the system as part of the contract.

   For example:

   - "What are the features that we need to design for this system?"
   - "What are the edge cases we need to consider, if any, in our design?"

2. **Non-functional requirements**

   These are the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. They are also called non-behavioral requirements. For example, portability, maintainability, reliability, scalability, security, etc.

   For example:

   - "Each request should be processed with the minimum latency"
   - "System should be highly available"

3. **Extended requirements**

   These are basically "nice to have" requirements that might be out of the scope of the system.

   For example:

   - "Our system should record metrics and analytics"
   - "Service health and performance monitoring?"

Estimation and Constraints

Estimate the scale of the system we're going to design. It is important to ask questions such as:

- "What is the desired scale that this system will need to handle?"
- "What is the read/write ratio of our system?"
- "How many requests per second?"
- "How much storage will be needed?"

These questions will help us scale our design later.

Data model design

Once we have the estimations, we can start with defining the database schema. Doing so in the early stages of the interview would help us to understand the data flow which is the core of every system. In this step, we basically define all the entities and relationships between them.

- "What are the different entities in the system?"
- "What are the relationships between these entities?"
- "How many tables do we need?"
- "Is NoSQL a better choice here?"

API design

Next, we can start designing APIs for the system. These APIs will help us define the expectations from the system explicitly. We don't have to write any code, just a simple interface defining the API requirements such as parameters, functions, classes, types, entities, etc.

For example:

```
createUser(name: string,email: string): User
```

It is advised to keep the interface as simple as possible and come back to it later when covering extended requirements.

High-level component design

Now we have established our data model and API design, it's time to identify system components (such as Load Balancers, API Gateway, etc.) that are needed to solve our problem and draft the first design of our system.

- "Is it best to design a monolithic or a microservices architecture?"
- "What type of database should we use?"

  Once we have a basic diagram, we can start discussing with the interviewer how the system will work from the client's perspective.

Detailed design

Now it's time to go into detail about the major components of the system we designed. As always discuss with the interviewer which component may need further improvements.

Here is a good opportunity to demonstrate your experience in the areas of your expertise. Present different approaches, advantages, and disadvantages. Explain your design decisions, and back them up with examples. This is also a good time to discuss any additional features the system might be able to support, though this is optional.

- "How should we partition our data?"
- "What about load distribution?"
- "Should we use cache?"
- "How will we handle a sudden spike in traffic?"

  Also, try not to be too opinionated about certain technologies, statements like "I believe that NoSQL databases are just better, SQL databases are not scalable" reflect poorly. As someone who has interviewed a lot of people over the years, my two cents here would be to be humble about what you know and what you do not. Use your existing knowledge with examples to navigate this part of the interview.

Identify and resolve bottlenecks

Finally, it's time to discuss bottlenecks and approaches to mitigate them. Here are some important questions to ask:

- "Do we have enough database replicas?"
- "Is there any single point of failure?"
- "Is database sharding required?"
- "How can we make our system more robust?"
- "How to improve the availability of our cache?"

  Make sure to read the engineering blog of the company you're interviewing with. This will help you get a sense of what technology stack they're using and which problems are important to them