How to Approach System Design Questions during Interview?

System design questions are popular during a technical round of an interview at top tech companies and large-scale tech startups. These types of interviews are open-ended conversations where candidates are expected to take the lead and guide the discussion with the interviewer. In this blog, we will discuss critical steps that candidates can follow to solve system design questions during interviews.

Step 1: Requirements clarifications

Since system design questions are often open-ended and do not have a single correct answer, it is important to ask questions and clarify any ambiguities early in the interview process. Candidates who take time to understand the end goals of the system are more likely to be successful.

To gather requirements and understand the scope of the problem, it can be helpful to ask questions such as:

- Who will be using the system?
- How will they be using it?
- How many users are there?
- What does the system do?
- What are the inputs and outputs of the system?
- How much data is expected to be handled?
- How many requests per second are expected?
- What is the expected read-to-write ratio?

Step 2: Back-of-the-envelope estimation

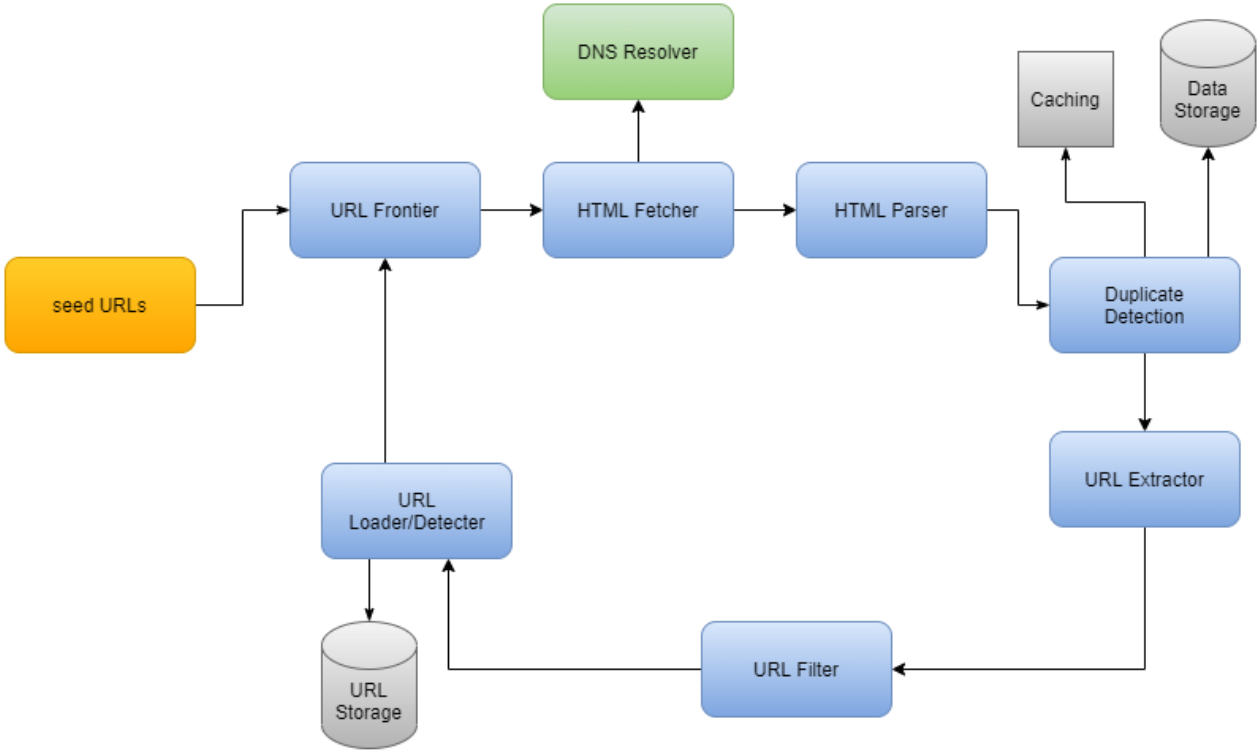
When designing a system, it is important to start by defining the system scale. This includes considering factors such as read-to-write ratio, number of concurrent requests, number of monthly users, number of daily

active users, required storage capacity, bandwidth requirements, and any data limitations. Once these parameters have been discussed with the interviewer, you can start to think about the best way to design a system that will work well at scale.

Step 3: Create a high-level design

To begin the design process, it can be helpful to outline a high-level design with all of the critical components and try to draw a diagram representing the system's core components. This will help you to identify all components that are needed to solve the problem from end to end.

By outlining the interaction of key components, you can better explain the overall structure of the system and how various components will work together to achieve desired results. This can help to ensure that your design is comprehensive and meets all requirements and constraints of the problem.

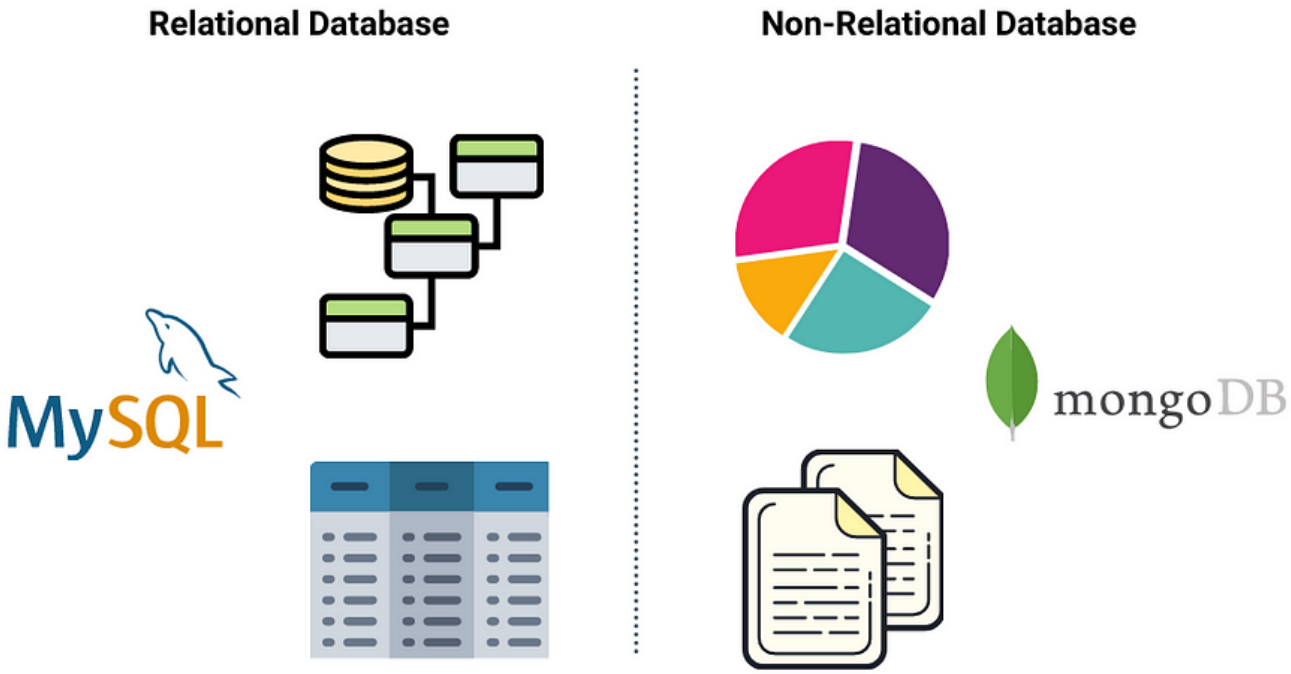


Step 4: Database Design

Before designing a hypothetical system, it is important to define how data

will be processed. This includes identifying the inputs and outputs of the system, how they will be stored, and how data will flow through the system. Determining which database would be the best fit for the problem can also be helpful at this stage.

You can try to explore this blog: SQL vs NoSQL Comparison.



Step 5: Design core components

Once you have outlined the high-level structure of the system, it can be helpful to discuss the major components in more detail. At this stage, interviewer input can often help guide you to parts of the system that need more attention. It is important to present various approaches, benefits, and drawbacks of each option and justify why you have chosen a particular approach. For example, if you were asked to design Tiny-URL, then discuss these things:

Generating and storing a hash of the full URL

- MD5 and Base62
- Hash collisions
- SQL or NoSQL

- Database schema

Translating a hashed URL to the full URL

- Database lookup
- API and object-oriented design

Step 6: Scale the design

To ensure that your system is able to meet the scale requirements of the problem, it is important to identify key concepts relevant to scalability. This can include considering approaches such as load balancing, horizontal scaling, caching, database sharding, replication, etc. to address scalability issues.

By considering these options and determining which are most appropriate for your system, you can ensure that your design is able to handle the expected workload and meet the needs of your users.

Step 7: Identifying and resolving bottlenecks

To ensure that your system is robust and able to handle unexpected challenges, it is important to discuss as many potential bottlenecks as possible and consider different approaches to mitigate them. Some questions to consider might include:

- Is there any single point of failure in the system? What steps can be taken to mitigate this risk?
- Are there enough replicas of data to ensure that users can still be served if a few servers are down?
- Are there enough copies of different services running to ensure that a few failures will not cause a total system shutdown?
- How is the performance of the service being monitored? Are there alerts in place to notify the team when critical components fail or their performance degrades?

By considering such issues, you can ensure that your system is well-

equipped to handle unexpected challenges and continue serving your users effectively.

System design questions to practice the steps

- Design Key-Value Store
- Typeahead System Design
- Pastebin System Design
- Designing API Rate Limiter
- Designing Web Crawler
- Youtube System Design
- Designing Google Docs

If you have any queries or feedback, please write us at contact@enjoyalgorithms.com. Enjoy learning, Enjoy system design!