Top 10 System Design Interview Questions and Answers

anuupadhyay

In the software engineering interview process system design round has become a standard part of the interview. If you want to get your dream job in some big tech giant companies (especially as a senior engineer) then you need to tell your approach to build a complex large scalable system. There is no standard or accurate answer to the design interview questions.

You may have different conversations with different interviewers. Due to the lack of experience in building a large-scale system and the open-ended nature of the system design round a lot of candidates struggle with this round. Not just junior and mid-level developers but also experienced developers feel uncomfortable in the system design round. If you are preparing yourself for this round then we recommend you to read the blog How to Crack System Design Round in Interviews. Along with this, if you also want to crack Architect, SDM, SDE, and SSE jobs then you can check out **the System Design – Live course** and optimize your approach towards getting an amazing job.

We will discuss some common design interview questions asked by the interviewers frequently in big tech companies. We will also discuss some key points, that you should consider while designing the system and explaining the answer to the interviewer.

1. Design a URL Shortening Service (TinyURL)

URL shortening service allows users to enter a long URL, and then it returns a shorter, unique URL. These services generate a short URL if the user gives a long URL and if the user gives a short URL then it returns the original long URL.

Things to discuss and analyze:

- Given a long URL, the service should generate a shorter and unique alias for it.
- When the user hits a short link, the service should redirect to the original link.
- Consider scalability if 1000’s URL shortening requests come every second.
- Service handle redirects.
- Support for custom short URLs.
- Track click stats.
- Delete expired URLs.
- The system should be highly available.

You need to consider three things while designing this service.

1. **API(REST API)** – Discuss how the client will follow an approach to communicate with the service along with the load balancer which is the front end of the service.
2. **Application Layer** – Discuss how the worker thread or hosts that will take the long URL, generate the tiny URL and how it will store both of the URLs in the database.
3. **Persistence Layer** – Database

2. Design YouTube/Netflix (A Global Live Video Streaming Service)

Design a video streaming service like Youtube/Netflix where users can upload/view/search videos. The service should be scalable where a large number of users can watch and share the videos simultaneously. It will be storing and transmitting petabytes and petabytes of data.

Things to discuss and analyze:

- Approach to record stats about videos e.g. the total number of views, up-votes/down-votes, etc.

- Adding comments on videos in real-time.

Components:

1. **OC** – Clouds like AWS, and OpenConnect act as a content delivery network.
2. **Backend-**Database
3. **Client-** Any device to use Youtube/Netflix

3. Design Facebook Messenger or WhatsApp (A Global Chat Service)

Things to discuss and analyze:

- Approach for one-on-one text messaging between users.
- Approach for extending the design to support group chats.
- Delivered and read status
- What action needs to be taken if the user is not connected to the internet?
- Push notifications
- Sending media like images or other documents
- Approach for providing end-to-end message encryption.

4. Design Quora/Reddit/HackerNews (A Social Network + Message Board Service)

These services allow users to post questions, share links and answer the questions of other users. Users can also comment on questions or shared links.

Things to discuss and analyze:

- Approach to record stats of each answer such as the number of views, up-votes/down-votes, etc.
- Follow options should be there for users to follow other users or topics.
- News feed generation which means users can see the list of top questions from all the users and topics they follow on their timeline.

5. Design Search Typeahead

Typeahead service allows users to type some query and based on that it suggests top searched items starting with whatever the user has typed.

Things to discuss and analyze:

- Approach to storing previous search queries
- Real-time requirement of the system
- Approach to keep the data fresh.
- Approach to find the best matches to the already typed string
- Queries per second are to be handled by the system.
- Criteria for choosing the suggestions.

- A total number of data to be stored.
- Approach to find the best matches to the already typed string

6. Design Dropbox/Google Drive/Google Photos (A Global File Storage and Sharing Service)

Design a file or image hosting service that allows users to upload, store, share, delete, and download files or images on their servers and provides synchronization across various devices.

Things to discuss and analyze:

- Approach to upload/view/search/share/download files or photos from any device.
- Service should support automatic synchronization between devices, i.e., after updating a file on one device, it should get synchronized on all devices.
- ACID (Atomicity, Consistency, Isolation, and Durability) properties should be present in the system.
- Approach to track permission for file sharing.
- Allowing multiple users to edit the same document.
- The system should support storing large files up to a GB.

7. Design a Web Crawler

Design a Web Crawler scalable service that collects information (crawls) from the entire web and fetches hundreds of millions of web documents.

Things to discuss and analyze:

- Approach to finding new web pages.
- Approach to prioritize web pages that change dynamically.
- Ensure that the crawler is not unbounded on the same domain.

8. Design Facebook, Twitter, or Instagram

You need to design a social media service for billions of users. Most of the interviewers spend time discussing news feed generation services in these apps.

Features to be considered:

- Some of the specific Twitter/Facebook/Instagram features to be supported.
- Privacy controls around each tweet or post.
- Users should be able to post tweets also the system should support replies to tweets/grouping tweets by conversations.
- Users should be able to see trending tweets/posts.
- Direct messaging
- Mentions/Tagging.
- The user should be able to follow another user.

Things to analyze:

- The system should be able to handle the huge amount of traffic for billions of users.
- Number of followers
- The number of times the tweet has been favorited.

Components:

- News feed generation.
- Social graph (Friend connection networking between users or who follows whom?—? especially when millions of users are following a celebrity
- Efficient storage and search for posts or tweets.

9. Design Uber or Lyft (A Ride-Sharing Service)

Design a service where a user requests a ride from the app, and a driver arrives to take them to their destination. A frequently asked interview question in the system design round of interviews.

Architecture: Monolithic/Microservices (real-time service, Front-end (Application), and database)

Things to analyze and discuss:

- The backend is primarily serving mobile phone traffic. uber app talks to the backend over mobile data.
- How dispatch system works (GPS/ location data is what drives the dispatch system)? How efficiently can the user match request with nearby drivers?
- How do maps and routing work in Uber? How ETAs are calculated?
- An efficient approach to store millions of geographical locations for drivers/riders who are always on the move.
- Approach to handle millions of updates to driver location.
- Dispatch is mostly built using Node.js
- Services: Business logic services are mostly written in Python.
- Databases: Postgres, Redis, MySQL.

10. Design an API Rate Limiter(Github)

Design a service or tool that monitors the number of requests per window of time a service agrees to allow. If the number of requests exceeds the rate limit blocks all the excess calls.

Things to analyze and discuss:

- Limiting the number of requests an entity can send to an API within a time window, for example, twenty requests per second.

- Rate limiting should work for a distributed setup, as the APIs are available through a group of servers.
- How to handle throttling (soft and hard throttling etc.).

Last Updated : 19 May, 2023

61

Similar Reads

1. Top 50 C++ Interview Questions and Answers (2023)
2. Top 25 Android Interview Questions and Answers For Experienced
3. Top 10 Most Commonly Asked Web3 Interview Questions and Answers
4. 20 Top Situational Interview Questions and Answers
5. Top 50 Android Interview Questions & Answers - SDE I to SDE III
6. Top 50 Data Mining Interview Questions & Answers
7. Top 25 Behavioral Interview Questions (With Sample Answers)
8. What is System Design - Learn System Design
9. Design BookMyShow - A System Design Interview Question
10. Design Twitter - A System Design Interview Question

Previous

Software Design Patterns

Article Contributed By :

anuupadhyay

anuupadhyay

Vote for difficulty

Current difficulty : Hard

Easy Normal Medium Hard Expert