Alex Xu - System Design Interview (2nd ed.)

3. A Framework for System Design Interviews

A 4-step process for effective system design interview

Effective interview gives strong signals about abilities

- to collaborate
- to work under pressure
- to resolve ambiguity constructively
- to ask good questions

Red flags

- over-engineering
- narrow mindedness
- stubbornness

Step 1. Understand the problem and establish design scope

Answering without a thorough understanding of the requirements is a huge red flag as the interview is not a trivia contest

Good questions

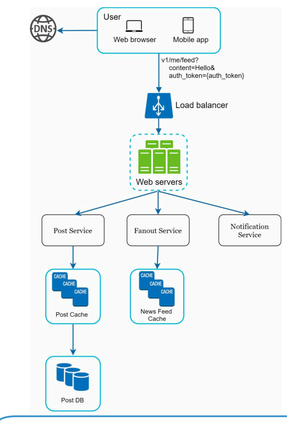
- What specific features are we going to build?
- How many users does the product have?
- How fast does the company anticipate to scale up? What are the anticipated scales in 3 months, 6 months, and a year?
- What is the company's technology stack? What existing services you might leverage to simplify the design?

Step 2. Propose high-level design and get buy-in

- Come up with an initial blueprint for the design
- Draw box diagrams with key components on the whiteboard or paper
- Do back-of-the-envelope calculations to evaluate if your blueprint fits the scale cons-

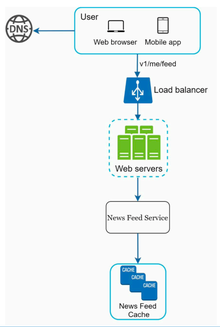
"Design a news feed system" example

Feed publishing: when a user publishes a post, corresponding data is written into cache/database, and the post will be populated into friends' news feed



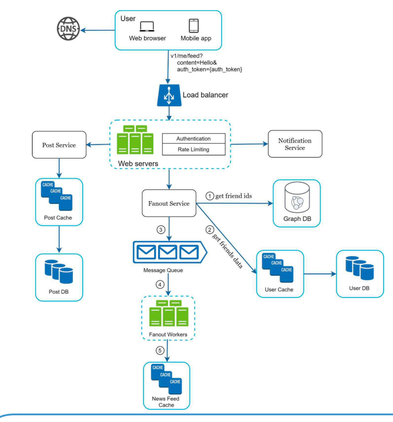
Q: Why we think we need caches for writing? Isn't it over-engineering at this stage?

Newsfeed building: the news feed is built by aggregating friends' posts in a reverse chronological order



Q: why we decide that feed is stored in cache and not in db? What if user want to see yesterday messages in his feed - cache could be expired already?

Q: why draw separate designs for separate scenarios? They should work together and should be in a single design!

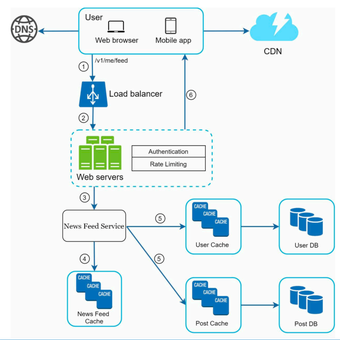


Step 3. Design deep dive

"Design a news feed system" example

Feed publishing

Q: why did we we select a graph db here?



Q: Why Auth/Rate limit have direct connectivity to User (arrow #6)? Why rate limit after load balancer - it should be stateful then?

Q: Should auth be stateless here (JWT-like)? BTW when it can be stateless and when it should be stateful?

Newsfeed building

Step 4. Wrap up

- Identify the system bottlenecks and discuss potential improvements
  - Never say your design is perfect and nothing can be improved!
- Give a recap of your design
- Error cases (server failure, network loss, etc.) are interesting to talk about
- Operation issues are worth mentioning. How do you monitor metrics and error logs? How to roll out the system?
- How to handle the next scale curve is also an interesting topic.
  - what changes do you need to make to support x10 more users
- Propose other refinements you need if you had more time

Dos and Donts

Dos

- Always ask for clarification
  - Do not assume your assumption is correct
- Understand the requirements of the problem.
- There is neither the right answer nor the best answer.
- Let the interviewer know what you are thinking.
- Suggest multiple approaches if possible.
- Once you agree with your intervie-wer on the blueprint, go into details on each component.
- Bounce ideas off the interviewer.
- Never give up.

Dont's

- Don't be unprepared for typical interview questions.
- Don't jump into a solution without clarifying the requirements and assumptions.
- Don't go into too much detail on a single component in the beginning
  - Give the highlevel design first then drills down
- If you get stuck, don't hesitate to ask for hints
- Again, communicate. Don't think in silence.
- Don't think your interview is done once you give the design
  - You are not done until your interviewer says you are done
  - Ask for feedback early and often

Time allocation on each step

- Step 1. Understand the problem and establish design scope
  - 3 - 10 minutes
- Step 2. Propose high-level design and get buy-in
  - 10 - 15 minutes
- Step 3. Design deep dive
  - 10 - 25 minutes
- Step 4. Wrap up
  - 3 - 5 minutes