CRACKING THE SOFT SKILLS

Created By Gayle Laakmann McDowell

Must Knows

- "So, tell me a bit about yourself…"
- "Why do you want to work here?"
- "Why should we hire you?"
- "Why are you leaving your current job?"
- "Where do you see yourself in 5 years?"
- "What do you do outside of work?"
- "What are your strengths and weaknesses?"

Your Two-Minute Pitch

"I am a ____ at _____."

"In college, I studied ___ at ____."

"Then I worked for … where I …."

"Then I worked for … where I …."

"In my current job, I've accomplished…."

"Also, outside of work, I …."

Preparing for Behavioral Questions

Create Resume Grid → Pick Five Key Stories → Diagram Your Stories

Themes	Job 1	Job 2
Leadership & Influence	*story*	
Mistakes & Failures		*story*
Challenges	*story*	
Teamwork	*story*	*story*
Successes		*story*

Coder? *Add: Bugs, Architecture, Optimization, Scaling.*

Stories	Story 1	Story 2
THEME(S)	*e.g., Leadership, Challenge, …*	
Nugget	*"Sure, let me tell you about when I …"*	
Situation	*Only the basics needed*	
Action(s)	*Expand here!*	
Result	*Prove it!*	
THE POINT	*What does it say about you?*	

Structured Answers

N **Nugget.** "Sure, let me tell you about the time that I …" This focuses you and your interviewer on what you're about to say.

S **Situation.** Explain just basics. The interviewer only needs enough details to understand what you did. Most people spend too much time here.

A **Action(s).** Detail what actions you took. "First, I …. Then, I …. And finally, I …." This is where you should spend most of your time. Speak in bullets!

R **Result.** Succinctly explain the result of your efforts were. Prove that the impact was good with numbers or a clear success metric.

Books by Gayle

Check Your Stories

- Are they substantial?
- Are they understandable?
- Have you explained *why* you did it this way?
- What do they say about you?
- Are they really about you (not your team)?
- Have you covered all the themes?
- Can you answer, "What would you do differently?"?

Questions For Your Interviewer

- Things you want to know.
- Things that show passion/interest.
- Things that show skills.

System Design Interview at Facebook

One of the goals of this interview is to design a Facebook-scale system. Interviewers will be evaluating your ability to determine what you should be building, and ability to solve large problems.

What we ask

Some example questions are:

- Architect a world-wide video distribution system
- Build Facebook chat
- Design a mobile image search client
- Build an API to power a crowd sourced address book

If you have deep, specialized knowledge (in fields like kernels, file systems, networking systems, JavaScript), the question might be rooted in that area. You'll be asked to design something you've never built before, and it will be of large enough scope that you won't be able to cover everything in perfect detail.

What we look for

As you're designing the system, among the things we're looking for is to see if you can:

- Ask clarifying questions to determine what the goals and requirements of the system are.
- Determine which parts of the problem are important and will affect the overall design.
- Describe the system at a high level, explaining the overall architecture. Explain and focus which pieces of the system are most interesting or difficult to design.
- Draw diagrams that clearly describe the relationship among different system components.
- Identify trade-offs in your design (such as consistency, availability, partitioning, performance), and describe how you make a decision around them.
- Calculate back-of-the-envelope and physical resources necessary.
- Adjust the design of your system when requirements or constraints change.
- Determine how your system will perform at Facebook's scale, and identify any bottlenecks and limitations in your design.
- Explain how your system handles both success and failure cases.

How to prepare

To get ready for architecture interviews, prepare and practice to familiarize yourself with the format.

Take any well-known app and imagine you're going to be building the primary feature. For example, you're going to be building video distribution for Facebook Video, or group chat for WhatsApp. Now figure out how you would build the various pieces out:

- How would you build your backend storage? How does that scale to Facebook's size?
- How would you lay out the application server layer? What are the responsibilities of the various services?
- How would you design your mobile API? What are the hard problems in representing the data being sent from server to client?
- How would you structure your mobile client? How do low-end devices and poor network conditions affect your design?

As you're designing these systems, run through the list of things we're looking for, and make sure for each piece of each app, you're able to answer all of them.

On-site Interview Logistics

- 4-5 hours long, encompassing 4-6 sections (~45-60 minutes/ section) in addition to a lunch break. Of the 4 sections, at least 2 will be coding-problem-solving questions on a white board (just like the tech-screen), at least 1 section will be a Systems Design exercise (also on a white board), and 1 section will focus on behavioral/ career questions.

On-site Interview Breakdown (~4 Sections)

- **Coding (**2-3 Sections, 45 minutes each**)**
    - This will be just like the tech-screen: Given the problem(s), choose an algorithm + data structure that will solve it, and then convert that into working code.
    - Once you come to a solution, I always recommend asking your interviewer if (s)he wants you to test, debug, or optimize your code in anyway.
    - Like the tech-screen, ask clarifying questions before diving into a solution, articulate your thoughts while you work (think out loud), and be conscious of time/ speed; it's better to finish the problems quickly and debug them afterwards.
    - Engineers that work here recommend the following resources:
        - **Leetcode**
        - **Interview Bit**
        - **HackerRank**
- **Lunch Break (**~45 minutes**)**
    - This is NOT an interview. This is an opportunity for you to stretch your legs, grab something to eat, and talk to an engineer here in a non-interview format.
- **Systems Design (**1-2 sections, 45 minutes**)**
    - We will test how you build systems or services at scale that support an end user product or service, so this is a "boxes and arrows" interview.
    - Example questions might be:
        - Design a key-value store
        - Design a Google search
        - Architect a world-wide video distribution system
        - Design Facebook Chat or design Facebook Newsfeed
    - What we're looking for:
        - Strong communication + problem solving skills:
            - Can **lead** a design discussion and show a clear understanding of the problem.

        - Can reason about tradeoffs regarding: consistency, availability, partitioning, performance, latency, scale
        - Can talk about any piece of the proposed design in detail
    - Strong Design + Architectural Skills:
        - Can propose a design for a system that breaks the problem down into components and that can be built independently
        - Can draw diagrams that clearly describe the relationship between the different components in the system
        - Calculate (back-of-the-envelope) the physical resources necessary to make this system work, and can give ballpark numbers on things like throughput/capacity for RAM, hard drive, network
        - Identify the bottlenecks as the system scales and understand the limitations in your design
        - Understand how to adapt the solution when requirements change
- **Engineers that work here recommend the following resources:**
    - **https://github.com/donnemartin/system-design-primer**
    - **https://www.interviewbit.com/courses/system-design/**
- **Another exercise that can help:**
    - To practice, take any well-known app and imagine you work for a competitor. Your job is to figure out:
        - Where most of their money is spent (compute? people? bandwidth? storage?)
        - The fundamental bottleneck of their system.
    - Answering those two questions will force you to think about how a system is actually implemented. Thinking about the costs and bottlenecks is more important than the nerdy details of the design.
    - For example, YouTube spends a ton of money on bandwidth, and secondarily on storage and compute. On the other hand, their long-tail traffic pattern means that their fundamental bottleneck is random disk seeks. Netflix also is a bandwidth hog but most of their traffic is at night (when it's cheap) and their library of videos is much smaller, so disk-seeks are probably not an issue at all.
    - Work out the above problems on a paper and think about the ways to break them down. It also helps to read up on

common large scale systems, like watching public videos about memcached and learning how search engines work. But during the interview, don't parrot back what you read; make sure your solution actually answers the question being asked.

- **Behavioral (**1 sections, 60 minutes**)**
  - 45 minutes for questions about how your work experience and how you work with other people, and 15 mins for a short coding question at the end
  - Example questions/ topics:
    - What kinds of teams and environments do you like to work in?
    - Tell me a technical accomplishment that you're proud of
    - Tell me about difficult work relationships
    - What challenging problems have you solved in the past?
    - How do you like to give and receive feedback?
  - Read through your resume and have concrete anecdotes/ examples for these types of questions (failures/ successes, challenges, why you've made certain career moves).
  - Understanding about the work at Facebook is also important, so try to be ready to address what it is you like about Facebook, our product, the technical challenges we face here, and your feedback on the product.
  - Typical questions to be ready for are:
    - Why do you want to come to Facebook?
    - What features would you improve on Facebook?
    - What would you like to do at Facebook?
    - What product or work-related project are you most proud of?

Systems Design Interview (45 minutes)

We will test how you build systems or services at scale that support an end user product or service, so this is a "boxes and arrows" interview.

What are we looking for?

- Take initiative and drive the discussion. Assume this is a one-way conversation and the interviewer will ask specific questions as needed and they can validate/clarify question as well.
- Provide a solution, as well as potential alternative approaches and reflect on their tradeoffs with respect to your design.
    - Tradeoff concepts to consider:
        - Performance vs. Scalability
        - Latency vs. Throughput
        - Availability vs Consistency
- Consider and discuss in detail the strengths and weakness of the system you've designed.
- Anticipate bottlenecks and other problems.
- Respond to changing requirements

What topics to cover?

Design Principles to Review:	These topics are fair game in the systems design interview:	You will NOT be penalized for not for having **in-depth** experience with these topics:
• Load Balancing • Caching • Sharding • Database Partitioning/Replication • CAP Theorem	• Data storage, data aggregation • QPS capacity or machine estimates (back of the envelope estimations) • Cross-dc network traffic considerations • Byte Size estimation	• Mobile APIs • Pagination • Tradeoff's between markup, json, and html responses • Optimizing product based on user feedback

How to prepare?

1) Review design principles (do you know them? Can you explain them and apply them?)
2) Practice the below questions with a friend
    a. Design Facebook Chat
    b. Architect a world-wide video distribution system
    c. Design Google search
3) *Review two sample problems in this website: https://www.hiredintech.com/classrooms/system-design/lesson/61*
4) Read attached document on design interviews as a frame of reference

Lastly, if you'd like to go through an in-depth guide on systems design, check out this site and mark off the areas you know and don't know: Github Systems Design Primer

    a. *Review and grasp the ones you don't know*
    b. *Fine tune the areas you do know*