Gmail **sivaddi RangareddyGari <sivareddy.jnta@gmail.com>**

Siva, Facebook onsite prep material

1 message

Priti Kadam <prikadam@fb.com> Thu, Feb 13, 2020 at 12:05 PM
To: sivaddi <sivareddy.jnta@gmail.com>
Cc: Nitin Lamba <nitinlamba@fb.com>

Hi Siva,

It was great connecting with you this afternoon! Congratulations again on securing your onsite interview at Facebook! Per discussion, below is the summary of our discussion. Please review it and let me know if you have any question. I'll get back to you on interview date/s and location shortly.

Please see below to help you prepare for your interview:

You will have up to 5 interviews on the day, each lasting between 45 mins.

Your interviews will consist of 3 different formats:

- Career/Motivations x 1
- Coding x2
- Product Design x1
- 5th interview can be either based on career/motivations or product design.

Before you start looking at each format, please read *Cracking the Coding Interview*. Twice if possible.

1. Career/ Motivations - focused on personal attributes and behaviors. In this interview, we are looking for you to give specific examples when answering questions.

- We'll be looking at your leadership experience, this could consist of a number of things from technical leadership, direction of projects or products, making architectural and design decisions to mentoring engineers and people management.
- We're interested in your passions and what you're motivated by, things like, what would you like to work on at Facebook if you could work on anything the and problems you enjoy solving.
- We're looking at your current role, what you're doing, your ownership and the impact you have had.
- Conflict management – How you deal with conflict and disagreements with colleagues and managers.

- Where you have learnt from past mistakes, where you have taken feedback on-board and acted on it.
- We're interested in things that didn't go so well and what you learned and took away from the experience. This is a critical area to be able to demonstrate as Facebook embraces failure and being reflective.
- We're interested in how you could cope with our unstructured environment and can come up with your own work.

Many people make the mistake of not preparing for this interview, please make sure you prepare!

Some examples questions are:

- Can you give an example of a valuable piece of feedback that you have received?
- Describe a time where you had a disagreement or conflict within your team
- How would your co-workers describe you?
- Can you describe a situation where you had to work with a decision that you didn't agree with?
- Describe a technical mistake you have made recently? What did you learn?
- What's the most difficult/challenging problem you have had to solve?
- If things aren't going to plan, how do you move yourself/your projects forward?
- How do you seek out opportunities? (we really like candidates who are proactive in fixing/improving things)
- How do you communicate, both within your team and with other teams?
- What is your biggest failure? What did you learn from this?
- When did you receive constructive feedback? How did you act upon this?

You can expect to be asked "what would you like to do at Facebook?" so be prepared to talk specifically about your interests and how they relate to Facebook as a tech company and/or as a product.

We are looking for go-getters / people who can direct themselves and seek out opportunities. Think of some examples where you have been proactive on seeking out opportunities/solving problems without being told to do this.

You can also expect 1 short additional coding problem during this interview.

2. Coding

At least 2x 45mins interviews where you need to know your data structures and algorithms, it's similar to the phone interview. The only difference is you will be coding on a whiteboard at all coding interviews. We need to know you can translate your thoughts to code and work around any constraints.

You should:

- PRACTISE PRACTISE PRACTISE!
- Use the Careercup and codelab links provided below for example questions.
- Put yourself under time constraints, speed is important in the interview.

Remember:

- We can only evaluate you on the code you write.
- To talk about what you are doing throughout the interview. If you need to be quiet to think, that's great – just let the interviewer know.

7 steps to success in your coding interview:

1) Do not jump straight into coding, take a few mins to understand the problem and ask any clarifying questions (but not too long!).

2) Describe your solution to your interviewer and get their thoughts on your solution.

3) Think about your algorithm(s) including the complexity and runtime. Ask the interviewer if it's ok or if you should think about something more optimized. Then figure out your data structure and implement.

4) Run through your code verbally with your interviewer. This is really important at this point. Does it really do what you think it does? Make sure to read what is there, not what you think is there.

5) Test your code, put in an input to see what happens. We're looking for you to find the bugs yourself and fix anything that comes up

6) Restate the complexity. Is it the same, or different to your initial thinking based on what you have actually coded up? Make sure you're thinking about both space and time

7) Optimize. Proactively suggest ways to optimize to the interviewer and get their feedback to ensure what you're trying to do is not overly complex and is correct then code it up.

One important tip to hack your interview, you should practice questions on careercup (link below) on a whiteboard before you come in, or on A3 paper. Please refer back to your initial interview prep to get ready for the coding interviews.

There will be questions like:

• Write the code to print the first element of each "row" of a binary tree. (want to see that you know how to convert a list to multi-dimensional array)

• Implement tic-tac-toe

• Write the code to show the number and type of permutations of a given string

Think out loud if you are working through a solution you are presented with as the Engineer will want to know how you approach and troubleshoot problems. If the interviewer gives you hints to improve your code, take them and run with them. It is good to adjust and work through the problems with the interviewer to show your thought process and problem-solving ability.

Other things to consider include knowing the complexity of the algorithm you are writing, its running time and whether your solution could be improved or optimized. You should also check for edge cases and be thinking about how you might test your code to prove it works. Specifics to consider:

- Your interviewer will be evaluating your code on efficiency, bugs, and timeliness.
- Take hints from your interviewer to showcase your thought process and problem-solving ability.
- Generally avoid solutions with lots of edge cases or huge if/else if/else blocks. Deciding between iteration and recursion is always an important step.
- Think about different algorithms and algorithmic techniques (sorting, divide-and-conquer, dynamic programming/memorization, recursion).
- Think about data structures, particularly the ones used most often (Array, Stack/Queue, Hashset/Hashmap/Hashtable/Dictionary, Tree/Binary Tree, Heap, Graph, Bloom Filter, etc.)
- Don’t worry about rote memorization such as runtimes or API/native calls. It's good to know how to figure out approximate runtimes on the fly but the code you write is more important.
- You will be asked about O(Memory) constraints, the complexity of the algorithm you are writing and its runtime - O(N^2), O(N) etc
- Make sure you review recursion.
- Make sure you know what your base case is. Bonus points available for talking about or implementing the dynamic programming solution.

What we’re looking for in a nutshell:

Basic coding hygiene. Can you whiteboard some code, test it and talk about its running time!

3. Design – based on a product design problem.

This is a good place to start:

https://github.com/donnemartin/system-design-primer

Please also see the attached docs and read the whitepapers for more detail and hints for preparation.

You will need to design a system or a product, the problem will be a broad and ambiguous one where you will need to create something, end-to-end, that will scale. You should not assume anything. You should pin down the requirements, we're looking for you to drive conversation throughout the interview. Be very vocal, say this is what you're doing/why/ your reasons. It should be your design, not half yours and half the interviewers. Try to cover both breadth and depth, we don't have architects at Facebook, so you will need to talk about both high level concepts and details associated.

- We want to understand how you reason through a problem that you've not necessarily encountered before.
- We are looking to get signal on both your technical and communication skills.
- We expect you to drive the design of your solution and lead the discussion after clarifying the initial problem.
- Be aware of trade-offs and alternative solutions and express clearly the decisions you make and justify why you made them.
- Go into a level of technical depth for each element or component of your solution.
- Think about designing at very large scale as this is what you will be doing at Facebook. This www may help: http://highscalability.com/blog/category/example
- Have a look at http://www.hiredintech.com/system-design

How we do it:

https://code.fb.com/?s=backend

https://code.facebook.com/posts/816473015039157/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/

Interview hacks to think about:

We don't expect you to know crazy algorithms that are domain-specific (like Quad Trees or Paxos). We do expect you to know that you have a variety of tradeoffs like consistency, availability, partitioning, etc. We also expect that you're working with a modern computer and know ballpark ideas on throughput/capacity for RAM, Hard Drive, Network, etc.

A good design shows that you:

- clearly understand the problem and break it down in a logical way
- think about a the high level design
- propose a design for a system that breaks the problem down into components, that can be built independently

- identify the bottlenecks as the system scales and can poke holes in the design
- think about all relevant trade-offs
- understand how to adapt the solution when requirements are changed
- 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

Try not to "one-off" stuff. Break things down into large, isolated components and drill in on things that you think are hard or critical problems.

There's a good book called "Cracking the Technical Interview" that has a section called "Large Scale & Memory Limits" with some questions very similar to a design interview, but they're a little too coding focused where the design interview generally avoids coding but may ask data structures.

Probably the best way to study is to work out the below problems on a paper and just think about the ways to break them down.

Sample Questions:

- "Design a Newsfeed system"
- "Speed up this mobile app"
- "Implement Ads Targeting"
- "Design the backend for Google Photos"
- “Design a web crawler”

Recommended method:

First Step: Understand the Requirements

Spent 1-2 minutes clarifying the requirements. Either give the interviewer an example of what they are asking (e.g. “To make sure I understand the question -- I will design the newsfeed service in Facebook, so I will need to consider things like “what shows up in the feed”, “in what order”, “privacy”, “latency”, “scaling the systems to billions of requests”, “redundancy”, etc. -- Is that what you mean”. Alternatively you can ask them for an example of the product or feature they want you to design. Spend no more than 1-2 minutes here.

Some further examples:

- Are there any requirements on running time (online vs offline paths)
- Where are the users?
- How many users are there?
- Storage requirements?
- Data access or retention requirements?
- Security requirements?
- Mobile vs Web?

- Any APIs we need to externally expose? Any integration options?

Finally, before you proceed: ask which of the requirements are stronger than others? For instance, is there a strong requirement around data consistency? Latency? Reliability? Data Privacy? Can you write an ordered list of the priorities? Don't spend a lot of time here, but at least ask the questions -- it's important that you understand what tradeoffs exist when design systems. For instance, when speed and consistency are paramount, you should be thinking about synchronous calls. If some latency and variation in responses is tolerable, then asynchronous/queues are ok.

Second Step: Know the facts/figure that can help you estimate

Useful resource: https://gist.github.com/jboner/2841832

You could try putting these on note cards and memorizing them.

Useful resource 2: http://i.imgur.com/k0t1e.png

This makes it clearer that you want to be reading from SSD, not disk, and certainly not doing many data center round trips. And you also want to be careful about mutexs and access to shared resources.

Now you will want to estimate the scale of the system you will need -- even before you start to design it. Here are a few questions to ask:

• How many API requests will we expect? (e..g What is the QPS?)

• What data will be returned in these requests? (bytes or megabytes or gigabytes)

• Will there be read AND write operations or just read operations?

Chances are, you'll be given big numbers here. But it will be good to show that you understand that not every problem needs to be solved with a distributed, scaled system (sometimes things fit onto a single machine).

Last Step: Deep Dive/Design

• Everywhere that there is a question written below in this preparation email (e.g. "Which objects will be in the system?"), it's fine for you to just ask the question out loud during the interview. There are many right answers, and it's better to ask the question than jump straight to an answer.

• There are many things you may want to think about. You could go to the whiteboard write down the appropriate concepts, such as:

- Requirements
- Scaling
- Entity Design
- API Design
- Data Storage
- Security/Privacy
- Logging/Analytics
- Reliability

• These are a lot of the concepts that need to be covered in any design. You may not get to all of them, but it's important you show you understand the "big picture". Having the words written down can also help with the pace of the interview, and help you to remember to address as many of the concepts as you can.

• When thinking about entity modeling and design (Which objects will be in the system, and what relationships do they have with each other?), write down a few of the objects and relationships between them. When designing an API, make sure you point out that the API can be used by external AND internal developers (e.g. can be used by the mobile app, the web app, and packaged as an SDK for external developers). Think about what happens when this API is called? Here are two excellent articles on it:

http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/

http://blog.gainlo.co/index.php/2016/04/05/design-news-feed-system-part-2/

• Write out the overall system topology. It will almost always look like this at a high level:

https://gist.github.com/vasanthk/485d1c25737e8e72759f

• Data Storage (How will the data be stored physically on both the client and the server, and how will it be accessed). You will almost certainly be designing a distributed system, so you will want to think about how to distribute it (sometimes this is referred to loosely as "how to shard the solution". All this means is -- when you are given a request from a user, how will you decide which backend end server to send to the request? How will the "Load Balancer" in the above diagram work? Will you send it to a different server based on username? Geographic location? A combination of the two? The important thing here is to think about how the scale the requests evenly.

o Think also about caching: both on the client and server? What data will you cache? And why? How will you invalidate the cache? (will it be based on time? If so, how long?)

• Security/Privacy

- Who can see what? What about permissions?
- But what about employees? Which data do they have access to? Is there new types of data being introduced here?

- Does the API need any special key to work? Will the user be granting a permission to an external company? If so, how will we monitor for abuse?

• Logging Analytics

- What metrics do we care about? How will we log this data so that these metrics can be computed?
- How long do we retain the data?

• Reliability

- We need monitoring for the new services we introduced
- We need to be able to measure latency
- We need to publish service level agreements and metrics

Further Suggested Articles

• https://www.palantir.com/2011/10/how-to-ace-a-systems-design-interview/

• https://gist.github.com/vasanthk/485d1c25737e8e72759f

• http://highscalability.com/blog/2014/7/14/bitly-lessons-learned-building-a-distributed-system-that-han.html

We're looking to see how you handle thinking about the big picture? Do you know:

• How to formulate the goals, non-goals?

• How to scale the solution? (where are the bottlenecks? Is it memory, cpu, storage?)

• How to break down the processes?

• What about security?

• What about privacy?

• What about logging? And analytics?

• Any specific technologies that you'd invest in?

• What type of caching would you use? Why?

• Inter-server communication?

You should make sure to talk about all the points in the design, even if you can't elaborate on each one (e.g. Goals, Non-Goal, Security, Scalability, Privacy, Analytics/Logging, Mobile Consideration, Testing). Take your time at the board, describing everything that has to go into a "complete solution", even if its a bullet point (e.g. "We'll want to make sure we log the thing

we care about and have some sort of pipeline that can aggregate the data). Other things you should think about:

• Jeff Dean's "Numbers every computer scientists should know".

• Use the whiteboard -- make sure you draw what you are talking about. Make sure that you list every single element in the design headings.

• LEAD THE INTERVIEW: We want to see you asking clarifying questions. Particularly when it comes to requirements gathering. You'll be given a very high level problem to solve, and you must pick the kinds of questions that will lead towards solving things successfully.

Final things to not to forget!

- Load Balancers
- Sharding
- Caching / memache
- Grids
- Distribution of Data
- Speed / space / time trade-offs
- Pagination
- API's
- Client / Server relationships
- Partitioning

Finally (good job making it this far!)

Below I have highlighted 5 final key preparation areas for your interview:

1) Please thoroughly read this Blog: https://www.facebook.com/careers/life/preparing-for-your-software-engineering-interview-at-facebook

2) Here are some videos that give great advice around our coding interviews (the video password is FB_IPS):

- Cracking the Facebook Coding Interview - The Approach: https://vimeo.com/interviewprepsession/theapproach

- Cracking the Facebook Coding Interview- Problem Walk-Through: https://vimeo.com/interviewprepsession/problemwalkthrough

3) Example questions:

- https://www.careercup.com/page?pid=facebook-interview-questions
- https://codelab.interviewbit.com/registration/?type=professional

4) I would also highly recommend looking at Jackson Gabbard's Youtube videos. Jackson is an ex FB engineer who was here for approx 7 years and left FB recently to take some time out and travel. Jackson is really passionate about interviewing engineers and did approx 500+ interviews for us so has some really great insights into what we look for.

5) Finally, I would suggest signing up to FB live video interview prep session if you have not already. This happens bi-weekly, and you can sign in from home.
http://www.eventbrite.com/o/facebook-london-master-your-fb-interview-8037667565

6) Glassdoor is another great resource to prep for your interview

All the best!

Cheers,

Priti

FACEBOOK

Priti Kadam

Technical Recruiter | +65 8468 6271

9 Straits View, Marina One, Singapore 018937

Facebook |

to Work in Singapore!

---

7 attachments

- **Onsite - GFS Whitepaper.pdf**
  270K
- **Onsite - Haystack Whitepaper.pdf**
  312K
- **Onsite - Mapreduce Whitepaper.pdf**
  187K
- **Onsite - TAO Whitepaper.pdf**
  730K
- **Onsite Architecture Design Interview - tips 2.pdf**
  44K
- **Onsite Design interview - Facebook.pdf**
  53K
- **Onsite_Interview_Tips_for_Facebook.pdf**
  370K