FACEBOOK MANAGEMENT INTERVIEWING 101

The Facebook management interview can seem intense, spanning everything from management to software engineering. However, the interview isn't designed to focus on what you don't know. Its goal is to find the best aspects of each candidate to make sure you will be set up for success at Facebook.

Each candidate's experience is key to this evaluation and we spend much of the interview asking for detailed analysis and examples that reflect your experience and problem-solving skills. The four areas we focus on are: *Basic software engineering, Management, System design and Project retrospective*.

To efficiently prepare for such a broad set of topics, this guide summarizes each topic area, the types of questions asked, the evaluation criteria, and how one can prepare for the interview.

While preparing for your interview, we do not recommend you learn entire new areas of computer science. You have worked in the field for many years and it that experience that we most want to understand. Therefore, *we recommend focusing your preparation work on refreshing your memory and then building out a set of "answers" that best communicate your knowledge in each interview area.*

OVERVIEW

Here is a synopsis of the four areas interview areas are:

- *Management:* This area assesses your capability in supporting organizational health, people management and planning. Showcase success stories and get to the situations that show how you as a leader navigate the complexity of developing, supporting, and scaling both your people and your teams.
- *Project Retrospective:* Using example(s) from your past, discuss the non-technical skills needed to deliver a project from start to finish. Areas we cover include: 1) Business understanding, product/project planning; 2) How you negotiated with stakeholders and peers; 3) Setting and driving success metrics; 4) Your understanding of design tradeoffs, business vs operations vs development considerations; 5) How you scale systems and business processes.
- *System Design:* Explore the design of a complex system and the tradeoffs within a design. The scope of the question can vary widely; it's a challenging and deep technical discussion around product ideas, usability issues, scalability, data structures and technologies used. For this interview, there is no right or wrong answer. The interviewer wants to observe how you design and architect a system.
- *Problem Solving and Coding*: Solve a basic computer science problem typical for a 1st- or 2nd-year computer science undergrad program. The problem will cover basics of algorithms, data structures, design patterns, system design, complexity and basic coding.

Most candidates are most concerned with the coding question — especially those who have not coded in many years. Our goal is not to see if you can still write production-quality, syntactically perfect code but to see how you apply basic CS principles to solve a concrete problem using a language of your choice. In the notes below, you will see that we recommend you do not spend more that 10-20% of your time preparing for the coding question.

Most candidates should focus on the system design questions. Facebook builds some of the most complex systems on the planet and we expect our managers to understand the technical areas they are responsible for, working closely with the senior technical leads to evaluate designs and guiding teams on how best to take a project from concept to production. In this area, we will focus on how you think though large-scale technical problems where the problem definition can be ambiguous and careful considering of tradeoffs are key to the ultimate success of the product.

Of equal importance are the management and project retrospective areas. As practitioners of software management, we are often very good at doing the job, but can sometimes be a bit rusty describing what it takes to manage people and projects. Reviewing basics management techniques will help you provide clear and concise explanations of the art of management; mapping those techniques onto your experience will enable you to discuss relevant examples that help demonstrate the real-life experience you have.

DETAILED GUIDE

MANAGEMENT INTERVIEW

# Interviews: 2

OVERVIEW

The purpose of the People Management interview is to assess your capability in supporting Organizational Health and People Management & Planning. Showcase success stories and get to the situations that show how you as a leader has navigated the complexity of growing, developing and supporting the people in their team/organization.

PREPARATION GUIDELINES

- Pick one or two of your favorite management books and skim them to refresh yourself w.r.t the various aspects of management.
- Outline or role play answers to the basic questions asked of managers (e.g., how to you deal with low performs, how to you recruit, what happens if your schedule slips).
- Please prepare a few examples for each theme using the STAR methodology.

DURING THE INTERVIEW

- Describe both the underlying management principle(s) AND provide brief examples of how you've applied them in real life.
- Describe both the strengths and the pitfalls of various techniques
- Be honest about areas where you do not have much experience (e.g., managing remote sites).
- Differentiate between your previous employer(s) management culture and other models of management. For example, if your previous organization has a top-down decision-making culture, ALSO discuss organizational models that are more bottom up and the tradeoffs.
- Don't hesitate to ask follow-up questions to help clarify what the interviewer is looking for.

COMMON ERRORS

Assuming that since you've managed for years, you know everything. Taking a bit of time to organize your thoughts before the interview can improve the structure of your answers, allowing you to relate them back to basic management principles AND demonstrate your experience by coupling them with solid examples from your past

ASSESSMENT CRITERIA

- Conflict Resolution
- Managing top and bottom performers
- Coaching and recruiting talent
- Cross Functional partnership
- *Feedback:* Things you could do better, areas where you failed and what you have learnt

SAMPLE QUESTIONS

- How do you measure your success?
- How do you describe your job to people outside the industry?
- Discuss our lack of an architect role.
- How do you build a team?
- How do you keep people motivated?
- What is the point of a 1:1?
- Discuss Facebook's trajectory as you see it. What choices do they think were good and bad? Would you have done things differently? What problems do you see coming up in the future?
- Think of a mistake or failure you’ve made in the past two years. What did you learn from it and/or do differently in the future?
- What would a report/peer/manager feedback on you be? Strengths, areas for development, etc.
- How would you assess if someone is a good manager?
- Why is management attractive to you?
- What in your mind are the responsibilities of a manager? What specific types of duties would you like to do help with at Facebook?
- What kind of process have you followed in the past and what has worked well for you?
- What hasn't worked?
- How do you institute the right process at the right time? How do you know when you need something more formal?
- Have they ever removed process? Why or why not?
- What are you looking for in an engineer when you recruit? Do they prefer to hire only experienced folks? Are they more comfortable with generalists or specialists?
- How have you optimized the recruiting process in the past? Have they thought about the recruiting funnel and how you can optimize different parts of it?
- What are their thoughts on structured interview loops vs. non-structured free-form loops?

- How do you deal with people performance issues?
- Have you ever had to implement a PIP (performance improvement plan)? What are your thoughts on it?
- What are the pros/cons of public vs. private titles?
- How do you identify engineers who could make good managers? How do you help them develop?
- What do you do when a team completely disagrees with the founder/VP on the direction of a product?
- How do you bootstrap technical leadership in an organization that has no public titles?
- How do you handle a great engineer with communication problems?
- How do you handle someone who really wants a promotion but isn't ready?

PROJECT RETROSPECTIVE INTERVIEW

# Interviews: 1

OVERVIEW

Using an example from your past, you are asked to discuss the non-technical skills needed to deliver a project. Areas we cover include:

- Business understanding, product/project planning
- How you negotiated with stakeholders and peers
- Setting and driving success metrics
- Your understanding of design tradeoffs, business vs operations vs development considerations
- How you scale systems and business processes.

PREPARATION GUIDELINES

- Recall one or two different projects you have been responsible for and map out the various aspects of the project, focusing on the questions outlined above.
- Map out a pithy description of the project, it's goals, and the various efforts.

DURING THE INTERVIEW

- Be clear and concise. Since interviewer will not have any background knowledge of your example, make sure to provide enough context. *BUT* watch out for overly complex examples that are difficult to describe or require too much context (e.g., watch out for company specific knowledge that will not make sense to the interviewer)
- Map your project retrospective back to basic management principles
- Provide clear examples that do not require too many details. You will have approx. 35-40 minutes to talk about the project - but assume there will be many questions, so try to map your answer into a 25-minute story.
- Focus on your ability to zoom out and look at each project/product more holistically

COMMON ERRORS

- Spending too much time on the technical aspects of the project
- Spending too much time on aspects that require too many details

INTERVIEW ASSESSMENT CRITERIA

- Business understanding, product/project planning
- How you handle cross-function relationships (e.g., stakeholders, partner teams, customers)
- Setting and driving success metrics
- Your understanding of design tradeoffs, business vs operations vs development considerations
- How you scale systems and business processes.

CODING INTERVIEW: (PLEASE DO NOT SPEND MORE THAN 10% OF YOUR PREPARATION TIME ON THIS)

# Interviews: 1

OVERVIEW

- Be prepared for 1-2 basic coding questions covering basic CS coding, algorithms, data structures, design patterns, system design and scalability.
- You will whiteboard your solution using a programming language of your choice.
- Solutions should focus on the basics in each area, not the more advanced and/or esoteric computer science skills — e.g., we are not focusing on syntactically perfect code nor advanced language usage (e.g., C++ templates, multiple inheritance).

PREPARATION GUIDELINES

- *Review* core CS concepts (data structures, binary trees, link lists, object-oriented analysis, etc.) that are typically covered in freshman/sophomore CS programming courses.
- *Practice:* Use your favorite language to solve some basic programming problems (e.g., print a binary tree in-order, add two binary numbers passed to a function as a string with the function call: string AddNumbers(string firstNum, string secondNum). Practice away from the computer, writing code and thinking out loud as you try to solve the problem, using basic data set(s) to test and debug your solution.

DURING THE INTERVIEW

- Think out loud as you work through a solution, as the engineer/interviewer 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 it; it’s good to adjust and work through problems to show the interviewer your problem-solving abilities

- Don't try to fake it nor worry about speed. Manager's coding skills are often rusty and we don't expect you to write lots of code (most questions can be answered in under 25 lines of code) nor write code fast. We do expect you to be able to solve 1 or 2 basic coding problems in 45 minutes.
- Feel free to peruse the following coding resources that may give you a better idea of what you can expect and ways to prepare:
    - Cracking the Coding Interview (notably the best resource successful leadership candidates have used
    - Video on The Approach https://vimeo.com/157480836 Password: FB_IPS,
    - Geeks for , Topcoder, Problem solving with algorithms and data structures, Career cup, Code Chef, Project Euler.

COMMON ERRORS

- Trying to learn a new language
- Memorizing syntax rules, advanced language features, uncommon libraries

INTERVIEW ASSESSMENT CRITERIA

- *Communication:* Think out loud while writing code; Ask questions to confirm understanding
- *Problem Solving:* Check your solution before writing code
- *Coding*: Check for bugs, try to not overcomplicate your code
- *Complexity Analysis:* Time and space complexity
- *Debugging*: Check your code proactively

SYSTEM DESIGN INTERVIEW

# Interviews: 2

OVERVIEW

The design of complex systems and the tradeoffs within your design. The scope of the question can vary widely; it's a challenging and deep technical discussion around product ideas, usability issues, scalability, data structures and technologies used. One question will typically draw directly on your previous experience (e.g., file system design, device drivers, ad serving systems, web caching systems). Examples include: design a hotel reservation system, design Facebook's News Feed.

For this interview, there is no right or wrong answer; the interviewer wants to observe how you design and architect a system. We will look to you as the expert here and ask you to drive the design, starting with defining the high-level goals and then proposing a solution.

PREPARATION GUIDELINES

- Refresh your senior level and/or master's level CS background (e.g., operating systems, networking, distributed systems). Most people have not covered all areas — so focus on those you already have studied or worked in.
- Recall the design of a few systems you worked on in the past and try to answer for those systems the questions below (see "What to do in the interview bullet"). We will NOT be asking you for confidential information about your previous projects - but your experience makes them a good starting point for you to think about how you would answer these types of questions.
- Read a bit about how the hyper-scale companies have built their systems (e.g., Google File System, Facebook Tao Cache, Amazon S3). The goal is not to learn each of these systems in detail, but to build out a short list of the basic concepts used across many of these systems (e.g., sharding, containers and isolation, failure handling, durability). Those concepts are a good place to refresh your senior level CS background.
- Review the basics of hardware (e.g., cost, performance, limits) across CPU, network, and storage to give you a physical sense of what systems are currently capable of.
- Find a few system design questions on the internet. Take them and try to answer them on paper in 30 minutes per question.

DURING THE INTERVIEW

Be able to demonstrate an understanding of the low-level restraints and how they affect the high-level goals. and make sure to talk through your solutions keeping in mind different tradeoffs.

When analyzing your system, you should discuss several of the following areas:

- *Testability*: ability to formally, or otherwise, gain confidence that the components of the software system are correct.
- *Usability:* the customer's experience with the software system and the ability to evolve this quickly.
- *Extensibility:* the ability to change the software system over time.
- *Security:* ability of the system to survive DDOS, spoofing, tampering, repudiation, information disclosure or elevation of privilege attacks.
- *Portability:* ability of the system to execute in different environments, on heterogenous tiers, on different hardware, or even operating systems.
- *Availability:* ability for the system to survive failures.
- *Scalability:* ability for the system to grow over time.
- *Operational characteristics:* ability to diagnose or debug problems when they occur. Sev avoidance or mitigation.

COMMON ERRORS

- Trying to master whole new areas of computer science. For example, if you kernel person, do not try to learn the details of distributed systems.
- Focusing on optimal solutions - we are much more interested in seeing how you think though basic tradeoffs (e.g., should you use a hash table or an array) and how you think through those tradeoffs.

INTERVIEW ASSESSMENT CRITERIA

We are looking for how you can take an abstract problem definition in a field that is new to you. These interviews are intended to explore your ability to design a software component or system while looking at a number of dimensions of the design.

- *Problem exploration:* understand the problem and flesh out the requirements
- *Handle data:* segment the problem into pieces, build a logical and physical data model, and discuss the read/write/update path
- *Component responsibilities:* tackle the individual parts of the problem while keeping in mind the larger goals
- *Completeness of the solution:* generate a design for a system that addresses the requirements
- *Trade-offs:* discuss the positives and negatives of the design that was proposed
- *System evolution:* Show good intuition and judgement regarding what degrees of freedom within the system that are introduced. Evolve the software system in a coherent way as you introduce additional requirements
- *References*
    - https://github.com/donnemartin/system-design-primer
    - https://www.hiredintech.com/classrooms/system-design/lesson/52
    - https://www.educative.io/collection/5668639101419520/5649050225344512