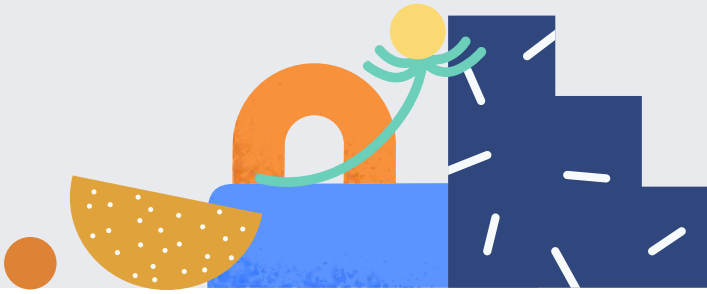
FACEBOOK

SOFTWARE ENGINEERING MANAGER

Onsite Interview Guide



What You'll Find in This Guide

Welcome to your prep guide for your engineering leadership onsite interview at the Facebook company. Our engineering leaders and recruiters put together this guide so you know what to expect and how to prepare.

Interview overview

The structure of your onsite interview

Your onsite interview is sectioned into the following 45-minute interviews, some of which will vary depending on your area of expertise:

- People management
- Project retrospective
- Career conversation
- System design & architecture
- Product design & architecture
- Machine Learning (if applicable)
- Coding

If you're unclear on what interviews are scheduled for your onsite interview, please contact your recruiter.

People management

How to prep for this section of your interview

This is a fairly unstructured interview, and is generally focused on philosophy, process, and strategy around your management experience. Be prepared with a few stories that highlight your people management, coaching / mentorship, team-building and talent acquisition skills. Think of examples that show both successes and failures in which you've navigated the complexity of growing, developing, and supporting the people in your team and / or organization.

Your interviewer will likely ask you to describe:

- A work experience that you consider to be most interesting, challenging, and perhaps relevant to what you view as an opportunity at Facebook.
- A situation in which you worked cross-functionally to remove a significant barrier for your team, or a situation in which you were able to really maximize the productivity of a team by digging into its inner workings and dynamics.

Example questions

- How do you evaluate a good team?
- How do you course-correct a team that's become unhealthy?
- How do you help develop people in your team?

Project retrospective

How to prep for this section of your interview

This discussion-based interview covers project implementation. The interviewer will ask you to present a project you've previously worked on, providing enough detail to explain the project at both high and detailed levels. We won't expect a presentation and we discourage slide decks, however we accept examples and diagrams.

What we look for

Using an example from your past, your interviewer will ask you to discuss the skills needed to deliver a project, to get a sense of your abilities in these areas:

- High-level product / project requirements, planning, and general business understanding.
- Negotiating with stakeholders and peers.
- Setting and driving success metrics.
- Understanding of design tradeoffs, business versus operations versus development considerations.
- Scaling systems and business processes.

How to prepare

Recall one or two different projects you've been responsible for and map out the various aspects of the project. Focus on how to best describe the high and detailed levels of the project.

During the interview

- **Be clear and concise.** The interviewer won't have any background knowledge of your example(s), so please make sure to provide enough context. Avoid overly complex examples that are difficult to describe or require too much context (e.g. company-specific knowledge for which your interviewer doesn't have a frame of reference).
- **Map your project retrospective** back to basic management principles.
- **Provide clear examples** that don't require too many details. You'll have roughly 40 minutes to talk about the project, but assume there'll be many questions. 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.

Career conversation

How to prep for this section of your interview

This interview helps us assess your capability in supporting organizational health and influencing the technical direction of your project / company. Your interviewer will ask you to showcase success stories and situations that present how you, as a leader, have navigated the complex business problems that affected the company at large.

How to prepare

- This interview will focus on your history, resume, and motivations. Review your resume so that you're prepared to discuss key events in your work history.
- Have concrete examples or anecdotes ready so you can support each question with examples.
- Prepare a few examples for each theme using the STAR methodology (Situation, Task, Action, Result).
- Taking a bit of time to organize your thoughts before the interview can help you structure your answers, allowing you to relate them back to basic leadership principles and demonstrate your experience by coupling them with solid examples from your past.
- Prepare answers for the following questions ahead of time and review the day before the interview:
    - How do you deal with conflict? Have examples prepared.
    - How and why did you become a people manager?

- What were some successful collaborations you've had?
- What motivates you as a person and in your career? What motivates you as a manager?
- Can you tell me about four people whose careers you've fundamentally improved?
- Describe a few of your peers at your company and what type of relationship you have with each of them.
- What did you do on your very best day at work?
- What does office politics mean to you, and do you see politics as your job?
- Tell me about a project that you led that failed. Why did it fail and what did you learn?

- Focus on teamwork, leadership, and mentorship qualities. Be open and honest about your successes and failures.

System design & architecture

How to prep for this section of your interview

This interview covers 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 an existing system like Twitter, Facebook, Google, Uber, etc.

For this interview, there's no right or wrong answer. The interviewer will observe how you design and architect a system. We'll 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.

How to prepare

- Refresh your senior-level and/or master's-level CS background (e.g. operating systems, networking, distributed systems). Most people haven't covered all areas, so focus on those you've already studied or worked in.
- Recall the design of a few systems you worked on in the past and try to answer the questions below for those systems (see "During the Interview"). We won't 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'd answer these types of questions.
- Read about how the hyper-scale companies have built their systems (e.g. Google File System, Facebook TAO cache, Amazon S3). The goal isn't to learn each of these systems in detail, but rather 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 online and try to answer them on paper in roughly 30 minutes per question.
- Avoid trying to master entire new areas of computer science. For example, if you are a kernel person, don't try to learn the details of distributed systems.
- Avoid focusing only on optimal solutions. We're much more interested in seeing how you think through basic tradeoffs (e.g. determining whether to use a relational database or a NoSQL data store) and how you think through those tradeoffs.

Example question

Google's index-building layer has many components for document understanding. It needs components for extracting deep links, contact information, referrals (for page rank). On the other hand, Twitter's index building should be simpler due to small-size tweets and some rich media information for attached media. Twitter's search is head heavy, so a bulk of engineering efforts in designing their search should go to rapidly indexing new tweets and making them searchable.

Work out the problems above on a paper and think about the ways to break them down. It also helps to read up on common large-scale systems, for example, watch the public videos about Memcached and learn how search engines work. However, during the interview, avoid repeating back what you read, but rather make sure your solution actually answers the question we've asked.

Other examples:

- Design a key value store.
- Design Google Search.
- Architect a worldwide video distribution system.
- Build Facebook Chat.

We may focus on specific types of systems (ads, distributed learning systems) or we may be more product focused. We won't expect you to know algorithms that you likely wouldn't know off the top of your head (e.g. quad trees or Paxos).

During the interview

Demonstrate an understanding of the low-level restraints and how they affect the high-level goals, and be sure to talk through your solutions while keeping in mind different tradeoffs. When analyzing your system, you should discuss several of the following areas:

- **Testability:** Formally, or otherwise, gaining confidence that the components of the software system are correct.
- **Usability:** The customer's experience with the software system and how to evolve it quickly.

- **Extensibility:** Changing the software system over time.
- **Security:** How the system can survive DDOS, spoofing, tampering, repudiation, information disclosure, or elevation of privilege attacks.
- **Portability:** How the system can execute in different environments, on heterogeneous tiers, on different hardware, or even operating systems.
- **Availability:** How the system can survive failures.
- **Scalability:** How the system can grow over time.
- **Operational characteristics:** Diagnosing or debugging problems when they occur. Disruption of service avoidance or mitigation.

We're looking for how you can take an abstract problem definition in a field somewhat new to you. Your interviewer will be exploring your ability to design a software component or system while looking at a number of dimensions of the design, including:

- **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.
- **Tradeoffs:** Discuss the positives and negatives of the design that was proposed.
- **System evolution:** Show good intuition and judgment regarding degrees of freedom within the system that are introduced. Evolve the software system in a coherent way as you introduce additional requirements.

Product design & architecture

How to prep for this section of your interview

This interview is product focused, and your interviewer will focus on the more holistic parts of building a software solution and less on the back-end components required.

An example question might be: "Tell me how you'd design a client-server API to build a rich document editor." Some questions you might want to consider:

- How does the client request data on the document from the server, especially as the document gets large enough that we wouldn't want to download it in a single request?
- How do we represent the rich document aspects like bold and italics in our API response?
- How do we design the system so that we can add new features on the server without breaking older clients?

How to prepare

Take a product or service that you use (e.g. Twitter) and talk through your thought process as you design that product in 30 minutes, from the core experiences and APIs, while keeping in mind other considerations like security.

Common errors

Refrain from trying to memorize how various products / services / APIs are currently designed and implemented.

Machine learning

How to prep for this section of your interview

The machine learning (ML) practical design interview focuses on your ability to build ML systems at Facebook scale.

What we look for

- Can you visualize the entire problem and solution space?
- Are you good at feature engineering?
- Can you detect flaws in machine learning systems and suggest improvements?
- Can you design consistent evaluation and deployment techniques?
- Do you understand architecture requirements (storage, perf, etc.) of your system?
- Can you model product requirements into your ML system?

A good design will touch on the following components:

- Problem formulation.
    - Optimization function.
    - Supervision signal.
    - Feature engineering.
        - » Data source.
        - » Representation.
    - Model architecture.
    - Evaluation metrics.

Example exercise and questions

Google's index-building layer has many more components for document understanding. It would need components for extracting deep links, contact information, and referrals (for page rank). On the other hand, Twitter's index building should be simpler due to small-size tweets and some rich media information for attached media. Twitter's search is head heavy. So a bulk of engineering efforts in designing its search should go to rapidly indexing new tweets and making them searchable.

Work out the problems above on paper and think about the ways to break them down. It also helps to read up on common large-scale systems, for example, watch the public videos about Memcached and learn how search engines work. However, during the interview, don't just repeat what you've read, but rather make sure your solution answers the question being asked.

Other examples:

- Design a key value store.
- Design Google search.
- Architect a worldwide video distribution system.
- Build Facebook Chat.

Your interviewer might focus on specific types of systems (ads, distributed learning systems, etc.) or they may be more product focused. We don't expect you to know complex algorithms that you likely wouldn't know off the top of your head (i.e. quad trees or Paxos).

How to prepare

- Take any well-known app and pick a system that can benefit from ML. Consider that system is built using a handful of rules for a small set of people. Now, consider you want to deprecate those rules and want to take advantage of ML, so you can easily extend that functionality to millions of people.
- Brush up on basic ML theory and be comfortable with concepts like overfitting and regularization.
- Practice your ability to convert intuitive ideas to concrete features. For example, number of likes is a good idea, but a better feature would use normalization, smoothing and bucketing.
- Think about the problem end to end. What will you do after you train the model and the model doesn't perform well? How do you go about debugging an ML model? How do you evaluate and continuously deploy an ML model?
- Be ready to analyze your approach. Having a good toolset of several different algorithms and understanding the tradeoffs is helpful. For example, be able to example advantages of logistic regression compared to SVM.

Coding

How to prep for this section of your interview

For this interview, be prepared for one or two basic coding questions covering CS coding, algorithms, data structures, design patterns, system design, and scalability. Your interviewer will ask you to 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. For example, we won't focus on syntactically perfect code nor advanced language usage (e.g. C++ templates and multiple inheritance).

How to prepare

Review basic problem-solving skills: how do you break down a problem, how do you map a problem to practice, how do you express a solution in code, how do you test, what is a good API (and why), what are the tradeoffs you are making in your solution?

Practice using 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 dataset(s) to test and debug your solution. We don't expect you to learn a new language just for the interview.

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. It's good to adjust and work through problems to show the interviewer your problem-solving abilities. If there's something you don't know, admit it and move on to keep your focus on speed.

What we look for

- **Communication:** Think out loud while writing code. Ask questions to confirm understanding.
- **Problem solving:** Check your approach before writing code.
- **Coding:** Check for bugs. Try to not over-complicate your code.
- **Complexity analysis:** Time and space complexity.
- **Debugging:** Check your code proactively.

Appendix / resources

Links to exercises, information and guides to help you prepare

Here are some resources to learn more about Facebook.

About Facebook

- **Life at Facebook as an Engineering Manager**
- **Founder's Letter**

Interviewing at Facebook

- **What to Expect During the Recruiting Process**
- **Interview Advice from a Facebook Engineer**
- **Mock Interviews**
- **Team Selection & Bootcamp Training**

Coding practice

- **HackerRank**

Design practice

- **HiredInTech—Algorithm Design Practice**

Thanks for taking the time to review this guide and good luck in the interview - you'll do great!