How to prepare for your software engineering interview at Meta

Meta

Interview Guide

Whether you're actively interviewing at Meta or just curious about the process, it’s important to know what to expect so you have a well-informed, positive interview experience. We’ve broken down Meta’s software engineering interview process along with tips and insights to help you prepare and do your best.

Step 1: initial technical screen

This initial interview serves as a screening step to determine whether to continue with a full interview cycle.

The initial technical screen determines whether you'll continue with the full interview cycle. This screen will be with a Meta engineer and is primarily a coding interview. Your recruiter will let you know whether your initial tech screen will be conducted over video chat or in person. The interview will include three sections, explained below:

Introductions (5 minutes)

We want to know more about you and how your experiences and interests align with Meta's mission. Prepare a concise, interesting description of who you are, where you've trained and worked, and what your areas of expertise are.

Coding (35 minutes)

You'll solve two coding problems focused on computer science fundamentals like algorithms, data structures, recursions, and binary trees. If your technical screen is not in person, the engineer will send you a collaborative editing tool. If it is in person, you'll use a whiteboard.

Answering your questions (5 minutes)

Come prepared with questions to learn about working at Meta as an engineer. Take this brief opportunity to learn more from an engineer's point of view. Think about what you find interesting and challenging about the work you'd be doing here or what challenges you're most interested in solving.

Here’s what you can expect during the initial technical interview, how to prepare and some tips for a successful interview.

- Introductions
- Coding: The next 35 minutes will be spent on coding.
    - This takes place in an online collaborative editor shared between you and the interviewer, or on a whiteboard if you do the initial interview in person.
    - You’ll be given one or more coding questions to complete in this editor. We ask questions that are short enough to explain in a few minutes and to solve in 10-30 minutes.
    - In this section, we try to understand your approach to problem solving.
    - We typically don't ask estimation or trick questions - nothing about how many ping pong balls can fit into a swimming pool.
    - You could be asked to solve a problem in any way you choose, and then the interviewer could add further constraints or requirements.
- Questions

How to prepare for Meta’s technical screen

Invest time in preparing. It's important for any engineer, even senior ones, to brush up on their interviewing skills, coding skills, and algorithms. You can practice by answering different kinds of coding questions. For example, practice answering a coding question with the most efficient bug-free solution without using a compiler.

- Write code in a simple text editor. During the interview, you'll write your code in a similar environment without syntax highlighting or auto-completion.
- Practice coding by hand. If your interview is conducted in person, be prepared to work on a whiteboard. Practice some of the questions with a whiteboard or pen and paper to help prepare.
- Practice under time pressure. You'll have a limited time for the coding question, so it will be important to finish it in time. If possible, have a mock interview with a friend to simulate the interview experience.
- Go over data structures, algorithms and complexity. Be able to discuss the big-O complexity of your approaches. Don't forget to brush up on your data structures, like lists, arrays, hash tables, hash maps, stacks, queues, graphs, trees, heaps, sorts, searches, and traversals (BFS, DFS). Also review recursion and iterative approaches.

Step 2: full interview

Interviewing with any company can be stressful, so we want to provide some insight as to how our process works. The full interview consists of the following:

- The coding interview: where you'll solve general coding questions.

- The design interview: where you’ll be asked to show off your design skills. The design question will be focused on either systems or product, depending on your background.
- The behavioral interview: where you’ll talk through your previous work experience, motivations, and a number of other behavioral questions.

Unless you've scheduled your interview for very early or very late in the day, someone from engineering or recruiting will take you to lunch. This will give you a chance to ask lots of questions of someone who isn't interviewing you.

Full interview: coding—what to expect

The coding interview is typically harder than the initial interview: We ask more difficult questions and have a more exacting evaluation. The 45 minute interview consists of the following:

- Introductions (5)
- 35 minutes of coding
- Questions (5)

How to prepare for the coding interview:

- Think out loud. We pay a lot of attention to the way you solve problems, which can be as important as having the right answer. Thinking out loud gives the interviewer insight into your thinking process and can also help them follow along with your solution.

- If you're interviewing remotely, locate a good interview spot. Choose a quiet place and ensure that you have a good internet connection and strong phone reception. Headphones will leave both hands free for coding.
- Speak clearly. Ensure you are speaking clearly and likewise, if you can't hear the interviewer clearly, let them know so they can accommodate.
- Unless otherwise informed by your recruiter, use the programming language you're best at. It's important to write your solution correctly and in time, so use the language you are most familiar with unless a specific language is required.
- Manage your time effectively. Spend some time figuring out the ideal solution to the question. Don't jump too quickly into brute forcing the first solution that comes to mind. If you can't find a better solution in a reasonable time, start writing a working solution, then iterate and improve it as you go. Some interviews end without any coding because the interviewee couldn't find the ideal solution. It's better to have non-optimal but working code than just an idea. Once you have a working solution, you can then try to improve its efficiency, code design, or any other aspect of it.
- Share your reasoning: Make sure you can talk about your solution because you will probably be asked to explain it.
- Find and fix the bugs by yourself. Don't wait for the interviewer to find them for you.
- If you are stuck, ask questions. Usually, the interviewer knows the question well enough to provide information that may help you move forward.

Full interview: design—what to expect

The design interview is 45 minutes long. These almost never involve coding. Instead, you'll spend the interview talking and drawing on the whiteboard. As with all interviews, the interviewer will typically save the last five minutes for your questions.

The purpose of the interview is to assess the candidate's ability to solve a non-trivial engineering design problem. To that end, your interviewer will ask you a very broad design problem and evaluate your solution.

There are two types of design interviews: systems design and product design. We know that candidates come from all sorts of backgrounds: some build complex user interfaces, others build network libraries, others build platform APIs for third parties. We expect every candidate to be familiar with basic design principles, but we'll try to match you with an interviewer of a similar background (systems or product) so that the conversation is as engaging as possible.

We aim to have the conversation be based in an area slightly familiar to you, allowing you to demonstrate your design abilities on an unfamiliar problem, but in a space that is not completely foreign to you.

Full interview: systems design—how to prepare

- Improving upon a design. Think about and review the complex systems you've already designed. What would you change about your approach if you did it all over again? What worked well?

- Designing from the ground up. Think about how you'd design a system that Meta already has. It's a good exercise to think through the complicated, high-scale systems that you already use every day. How would you design it from the ground up?
- Read up. Read engineering blogs about approaches that work and the false starts that big companies have had along the way.
- Start with requirements. Your interviewer might ask, "How would you architect the backend for a messaging system?" At the onset, this question sounds vague. Where do you even start? You could start with some requirements:
    - How many users are we talking about?
    - How many messages sent?
    - What are the latency requirements for sender->receiver message delivery?
    - How are you going to store messages?
    - What operations does this data store need to support?
    - What operations is it optimized for?
    - How do you push new messages to clients? Do you push at all, or rely on a pull based model?
- Ask us anything: The last five minutes are for questions. This is a great opportunity to get an insider's perspective directly from an engineer.

Full interview: product design—how to prepare

- Reflect on your projects. Think about the projects you've built. What was easy, and what was difficult?
- Your interviewer might ask, "Tell me how you'd design an email server." Some questions you might want to think about:
    - How do you store mail, especially as the system grows larger?
    - How do you handle mailing lists with large numbers of recipients?

    - How do you handle people abusing the system for spam?
    - How do you guarantee the reliability of the system in the face of potential system failures?
- A different interviewer might ask, “Tell me how you'd design a client-server API to build a rich document editor.” Some questions you might want to think about:
    - 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 new features can be added on the server without breaking older clients?
- Start with requirements: Your interviewer might ask, “How would you architect the backend for a messaging system?” This question is quite broad, so how would you begin to answer it? You could start with some requirements:
    - How many users are we talking about here?
    - How many messages sent?
    - How many messages read?
    - What are the latency requirements for sender->receiver message delivery?
    - How are you going to store messages?
    - What operations does this data store need to support?
    - What operations is it optimized for?
    - How do you push new messages to clients? Do you push at all, or rely on a pull-based model?

More tips for the systems design and product design interviews:

- Outline your high-level approach: When you take your approach, outline it then think about how it can be broken down into subparts.
- Identify your focus: There's not enough time to discuss every detail of the design, and find the interesting and hard problems to focus the discussion on.
- Navigate the holistic and the details: Move effortlessly from the goals to the high-level approach to the precise details and back again. A good solution covers both high-level ideas as well as low-level specifics. Talk about what components you'll use and how they fit together - "Responsibilities will be divided as so between Service A and Service B..." Also, describe the implementation details - "A pub-sub queue makes sense here because…"
- Explore the inherent tradeoffs: In any engineering problem, you will need to make intelligent decisions about tradeoffs. A good solution compares and contrasts different approaches. It explores the tradeoffs present in any complex engineering problem and it makes intelligent, reasoned decisions about those tradeoffs.
- Drive the discussion: Part of the signal the interviewer hopes to gather is whether you've learned how to build large systems through hard experience. Your ability to anticipate and work around typical problems is part of that signal.
- Make it a conversation: Be sure to ask clarifying questions. But make sure you drive towards a good solution.

Full interview: behavioral—what to expect

Meta's behavioral interview is 45-minutes long and is designed for your interviewer to learn more about your background, what you're passionate about in tech, and what kind of impact you want to make. The purpose of the behavioral interview is to assess if you would thrive in the company's fast-paced engineering organization.

Full interview: behavioral—how to prepare

- Know yourself. Take the time to review your own resume as your interviewer will almost certainly ask questions about key events in your work history.
- Be honest. Not every project is a runaway success and we may not always interact perfectly with our peers. Being transparent in these situations won't be counted against you in the interview. In fact, sharing and discussing how you learned, improved, and grew from your past experiences is valued.
- Use the S.T.A.R. (Situation, Task, Actions, Results) method to mentally organization your thoughts. This will provoke a well-thought-out and chronological action of events. Easy to describe, easy to follow.
- Have concrete examples or anecdotes. Support each question with practical experiences and examples. Avoid theoretical answers - if you go into a theoretical tangent, your interviewer will redirect you to provide a concrete example.

  Some typical behavioral interview questions are:

  - What are some of the best things you've built?
  - What are you most proud of?
  - What could you have done better?
  - What were some excellent collaborations you've had?
  - Tell me about a time when you advocated for and pushed your own ideas forward despite opposition.
  - How do you deal with conflict?
  - How do you like to give and receive feedback?
  - What kinds of technologies are you most excited about?

Some final tips for your software engineering interview:

- Be yourself. This means being open and honest about your successes and failures.
- Be humble and focus on teamwork, leadership, and mentorship qualities.
- Familiarize yourself with our six core values. These values influence how we work together to fulfill our mission of bringing the world closer together.
    - Move fast
    - Focus on the long-term impact
    - Build awesome things
    - Live in the future
    - Be direct and respect your colleagues
    - Meta, Metamates, me

Good luck with your interview.