Software Engineer, Reality Labs

Technical Screen Interview Guide

Meta

What You'll Find in This Guide

Welcome to your prep guide for your Reality Labs Software Engineer interview with Meta. Our engineering leaders and recruiters put together this guide so you know what to expect and how to prepare. We recognize that interviewing can be stressful, so we hope this guide provides the information and resources you need. Remember, your recruiter is there to support you, so please reach out to them with any questions.

Before you get started, it's important to note that Meta is committed to providing reasonable support (called accommodations) in our recruiting processes for candidates with disabilities, long term conditions, mental health conditions or sincerely held religious beliefs, or who are neurodivergent or require pregnancy-related support. If you need support, please reach out to accommodations-ext@fb.com or your recruiter.

Interview Process Overview

What will your interview process be like?

Meta seeks specialized engineers who have the technical expertise and ability to solve deep, challenging problems and who can communicate in a clear and compelling way to a wide variety of stakeholders and across diverse subjects. This interview is designed to give you the opportunity to give us a sense of your technical skills.

Technical skills aren't the same as interview skills, so even the most experienced engineers need to prepare and practice to do well in an interview. It's difficult for interviewers to get a clear signal on coding ability from someone who hasn't practiced solving new problems under time constraints, so we recommend that even the most experienced engineers sharpen those skills in advance.

Meta conducts most interviews remotely. Make sure your video conference setup is comfortable, quiet, and free of distractions. Coordinate with your recruiter or a friend to do a practice video conference session. Check connectivity, audio, video, and lighting in advance of your actual interviews.

What we look for

This interview focuses heavily on Coding. You'll be assessed on how you solve the problem as well as the structure and style of your code. Additionally, we want to hear your thought process throughout, so be sure to provide a narrative as you solve and code up your solutions.

The language you will code in may vary depending upon the specific role for which you are interviewing (see below for a list of languages by role). Please confirm the coding language preference with your recruiter in advance:

- Audio, Camera, Compiler, Embedded, or Graphics: C, C++
- Gameplay: C#, C++
- OS Frameworks: C++, Java
- Security: C, C++, Rust, Java, Python
- Video: C, C++, Java

Your interviewer will be thinking about how your skills and experience might help Meta. In your coding interview, your interviewer will assess your performance on four focus areas:

- **Communication:** Are you asking for requirements and clarity when necessary, or are you just diving into the code? Your initial tech screen should be a conversation, so don't forget to ask questions and listen carefully to your interviewer.
- **Problem Solving:** How do you comprehend and explain complex ideas? Are you providing the reasoning behind a particular solution? Developing and comparing multiple solutions? Using appropriate data structures? Speaking about space and time complexity? Optimizing your solution?
- **Coding:** Can you convert solutions to executable code? Is the code organized and does it capture the right logical structure?
- **Verification:** Are you considering a reasonable number of test cases or coming up with a good argument for why your code is correct? Are you able to walk through your own logic to find bugs and explain what the code is doing?

How to Prep

Your interview will focus on technical coding questions. In addition to the preparation guidance below, this video will give you an example of what to expect during your technical screen.

1. Before you practice, plan!

Be honest with yourself—only you know how much prep time you'll need. Make the most of your prep time by following these steps to plan your approach before you start practicing.

- **Schedule time to study and practice.** Block out time every day to write code. Target medium and hard problems.
- **Prioritize breadth over depth.** It's much better to practice solving fewer example problems of many problem types than to become very familiar with one type at the expense of the others.
- **Set aside time to review what you've practiced.** As you solve problems, make cheat sheets or flash cards to review later. Revision and repetition will strengthen your understanding of core concepts.
- **Remember your goal.** Aim for confidently solving two questions—while thinking aloud—in about 35 minutes.

2. Use key practice strategies to practice effectively

Reading through sample questions, recognizing concepts, and having a vague understanding of these concepts won't be enough to help you shine. You need to practice! Make sure you're setting your practice sessions up for success by following these tips from engineers who've been through the process.

- **Practice coding the way you'll code during your tech screen**. Use CoderPad.io if your interview is via phone or video call or use a whiteboard or pen and paper if your interview will be in person. Check with your recruiter if you're not sure which format you'll use.
- **Set a time constraint when you practice problems**. In your tech screen, you'll be asked to solve two problems in under 35 minutes. Practice coding solutions to medium and hard problems in less than 15 minutes each to help you be ready for the constraints during the interview.
- **Code in your strongest language**. Provide the most efficient solution and find and fix the bugs yourself.
- **Practice talking through the problem space and possible solutions before you dive in and talk through your decisions out loud as you code.** Interviewers will be evaluating your thought process as well as your coding abilities. Explaining your decisions as you code is crucial to helping them understand your choices. The more you practice this, the more natural it will feel during the interview.

3. Understand the types of problems you may encounter

Practice a variety of different problems—and understand why we ask them—so you're prepared to solve them during your interview.

- **Don't be surprised if the questions sound contrived**. Problems may be different than what you're probably tackling in a day-to-day job. We don't ask "puzzle" questions, but questions may be different than real-world problems because they need to be described and solved in 10-20 minutes.
- **Problems may assess the depth of your knowledge and your versatility**. For example, your interviewer might ask you to solve a problem any way you want. Then, they could add constraints on the running or space characteristics and ask you to solve it again.
- **Problems may focus on edge cases**. You might be asked to parse some data format or mini language. Your answers demonstrate your ability to handle multiple states in your head.
- **Problems may test how well you know how things work under the hood**. For example, you might be asked to implement well-known library functions.

How to approach problems during your interview

1. Before you code

- **Ask clarifying questions.** Talk through the problem and ask follow-up questions to make sure you understand the exact problem you're trying to solve before you jump into building the solution.
- **Let us know if you've seen the problem previously.** That will help us understand your context.
- **Present multiple potential solutions, if possible.** Talk through which solution you're choosing and why.

2. While you code

- **Don't forget to talk!** While your tech screen will focus heavily on coding, the engineer you're interviewing with will also be evaluating your thought process. Explaining your decisions and actions as you go will help the interviewer understand your choices.
- **Be flexible.** Some problems have elegant solutions, and some must be brute forced. If you get stuck, just describe your best approach, and ask the interviewer if you should go that route. It's much better to have non-optimal but working code than just an idea with nothing written down.
- **Iterate** rather than immediately trying to jump to the clever solution. If you

can't explain your concept clearly in five minutes, it's probably too complex.

- **Consider** (and be prepared to talk about):
  - Different algorithms and algorithmic techniques, such as sorting, divide-and-conquer, recursion, etc.
  - Data structures, particularly those used most often (e.g., array, stack/queue, hashset/hashmap/hashtable/dictionary, tree/binary tree, heap, graph)
  - Memory and runtime complexity for your algorithm expressed as big-O notation
- Generally, avoid solutions with lots of edge cases or huge if/else if/else blocks. Deciding between iteration and recursion can be an important step.

3. After you code

- **Validate your solution**. Think like a compiler. Walk through the code using test cases to evaluate correctness.
- **Expect questions**. The interviewer may tweak the problem a bit to test your knowledge and see if you can come up with another answer and/or further optimize your solution.
- **Take the interviewer's hints** to improve your code. If the interviewer suggests or asks a question, listen fully so you can incorporate any hints they may provide.
- Ask yourself if you would **approve your solution** as part of your codebase. Explain your answer to your interviewer. Make sure your solution is correct and efficient, that you've considered edge cases, and that it clearly reflects the ideas you're trying to express in your code.

Appendix / Resources

Links to exercises, information, and guides to help you prepare

About Meta

- About Meta website
- Meta Newsroom website
- Meta Life website
- Meta Diversity website

Coding Resources

- Software Engineering Interview Q+A Video
- Cracking the Meta Coding Interview Videos (The Approach and Problem Walk-through)(password: FB_IPS)
- HackerRank website
- CoderPad website
- Topcoder website
- GeeksQuiz website
- CareerCup website

Thank you for taking the time to review this guide!