Software Engineer, Machine Learning

Onsite Interview Prep Guide

Meta

What You'll Find in This Guide

Welcome to your prep guide for your interview process with Meta. Our engineers and recruiters put together this guide so you know what to expect and how to prepare. This document is quite lengthy but is designed to answer most of the questions you may have about the role, the interview process, and how to set yourself up for success by preparing.

Before you begin prep, one important piece of advice for your interview: it's ok if you don't know! No one who works at Meta is an expert in all things, and we don't look for perfection in the people we interview. If you aren't sure if something is true or if it's the best solution then say that. Explain what you do know, and your interviewer will ask you follow-up questions.

Now, please take some time to review the information and think about how much time you may need to really prepare for your interviews.

Interview Process Overview

What will your interview process be like?

Your interview process will include a number of 45-minute interviews. Each interviewer should also leave a few minutes at the end for your questions.

- **Coding**
- **System Design**
- **Machine Learning Design**
- **Behavioral**

Coding Interview

What can you expect?

This interview focuses heavily on coding. You'll be assessed on how you solved the problem as well as the structure and style of your code. It's best to avoid bugs, but the interviewer will not compile your code so don't worry about making minor mistakes. Finding and catching bugs in your code is a positive sign!

Additionally, they'll want to hear your thought process throughout, so be sure to provide a narrative as you go through the code. You're welcome to code in whatever language you feel most comfortable, but choosing one that is going to assist in getting an optimal solution in the most speedy and efficient manner is key.

Have questions about the format or details of your specific interview? Your recruiter can help!

What do we look for?

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:

- **Problem Solving:** We're evaluating how 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 have 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? If your solution has bugs, are you able to walk through your own logic to find them and explain what the code is doing?
- **Communication:** Are you asking for requirements and clarity when necessary, or are you just diving into the code? Your coding interview should be a conversation, so don't forget to ask questions.

How to prep

Interviewers can only assess your skills and abilities based on what you show them during your interview, so it's important to plan and prepare to best showcase your strengths.

As you begin preparing, please reference your Career Profile for role-specific prep materials. We use many different coding languages at Meta so practice in the coding language that you are most confident in unless otherwise noted based on your specific role or informed by your recruiter. Scripting languages (Python, Perl, JS, etc.) tend to be easier to use in coding interviews, but stick with a language you are comfortable with rather than attempting to learn a new one.

You should expect to solve about two problems in the course of about 40 minutes. When you have a solution, be sure to review it. Make sure that it's correct, that you've considered the edge cases, that it's efficient, and that it clearly reflects the ideas that you're trying to express in your code.

In addition to reviewing the above fundamentals, these tips may be helpful:

- **Schedule time to study and practice.** Block out time every day to write code. You'll need to practice writing code without executing it - without the help of a compiler and debugger. This will simulate a timed interview environment.
- **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.
- **Think about different algorithms and algorithmic techniques** (e.g., iteration, sorting, divide-and-conquer, recursion). We do not ask dynamic programming questions so don't spend time prepping for that technique and focus your efforts elsewhere.
- **Think about data structures**, particularly the ones used most often. For instance, array, stack (or queues), hashtable, trees (including specialized trees like binary trees and binary search trees), graphs, and heaps.
- **Analyze the problem** and make sure that you fully understand it before jumping in. It's OK to ask clarifying questions during the interview to ensure you understand the exact problem you are trying to solve. Also, breaking down the problem into smaller pieces can be helpful.
- **Get comfortable with the medium you'll use in the interview.** In other words, try writing code with just a plain text editor with no compiler, linter, syntax highlighter, autocomplete, and so on. Please also practice talking through a problem before you start coding. Your interviewer will be evaluating how you explore the problem space and weigh different possible solutions so it's crucial to help the interviewer understand your choices.
- **Keep things simple.** If you find the solution getting excessively complex, step back and ask if there's a simpler way to solve it. It may also be helpful to write everything out so you can see insights and bugs faster and make fewer mistakes.

You will find an appendix at the end of this guide with specific resources and tools that you may reference and utilize as you prepare for your interview.

System Design Interview

What can you expect?

The system design interview is 45 minutes and does not involve coding. Instead, you'll spend the interview talking and using the virtual whiteboard as a visual aid (think: box and arrows diagrams, function signatures, or API specifications).

What do we look for?

The purpose of this interview is to assess your ability to solve a non-trivial engineering design problem. The interviewer is trying to determine if you can architect a solution to a higher-level problem that requires connecting multiple concepts. The question will be something very high-level and it will be up to you to drive the conversation. In this interview, your interviewer will assess your performance on four focus areas:

- **Problem Navigation:** Are you demonstrating your ability to organize the problem space, the constraints, and the potential solutions? You should be asking questions to reduce ambiguity, identify the most critical problems, understand what is needed for quantitative analysis, and define a requirement set to design to.
- **Solution Design:** We are evaluating your ability to design a working solution that addresses either the complete problem/design or pieces of the problem/design. Are you able to consider the big picture in your design?
- **Technical Excellence:** Can you dive into the technical details when needed? Can you identify and articulate the dependencies and trade-offs? How are you mitigating potential risks and failure points?
- **Technical Communication:** Can you articulate your vision and technical ideas clearly? We are assessing your ability to communicate your reasoning as well as understand and address feedback from the interviewer.

How to prep

This may be a challenging interview to prep for given it is so interactive and there's no "right answer" for a design problem. Use the prep tips here and the resources listed in the appendix.

- **Improve 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?
- **Design from the ground up.** Think about how you'd design a system that Meta (or another large tech company) 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 scratch? What are the bottlenecks to your solution? How would you address them?
- **Keep in mind inherent tradeoffs.** Explore competing solutions and speak to all their major tradeoffs. Demonstrate that you can make intelligent decisions about how to balance each of those tradeoffs.
- **Do some research.** Read engineering blogs about approaches that have worked for big companies along the way. Read about the false-starts too!
- **Be holistic and detailed.** Practice moving effortlessly from high-level information to the precise details. Outline the high-level requirements by describing what components you may need, how they fit together, and how to get to that solution.
- **Start with the requirements.** Your interviewer might ask: "How would you architect the front-end for a messaging system?" Obviously, this question is extremely vague. Where do you even start? You could start with some requirements:

- How many users are we talking about?
- What should the experience be while you're waiting for confirmation?
- How will you show error states?
- What are the latency requirements for sender–receiver message delivery?
- How are you going to store messages?
- What kind of features are we going to need to support?
- What operations does this data store need to support?
- How do you push new messages to clients? Do you push at all, or rely on a pull-based model?

Machine Learning (ML) Practical Design Interview

What can you expect?

The ML practice design interview will consist of a 45-minute session and focuses on your ability to build ML systems at scale. A strong performance in this interview indicates to your interviewer that you'd be successful in most applied ML problems here at Meta. You can expect your interviewer to ask:

- Design a personalized news ranking system.
- Design a product recommendation system.
- Design an evaluation framework for ads ranking.

What do we look for?

The purpose of this interview is for the candidate to pick any product feature and understand how to model it using ML. We're not looking for you to know and memorize every ML algorithm that may exist. Instead, you should be able to use your existing toolsets to model the problem and break down the components involved in building a large-scale ML system.

We assess candidates on the following signals for ML design:

- **Problem Navigation:** Can you visualize and organize the entire problem and solution space? Can you connect the business context and needs to the ML decisions?
- **Training Data:** How would you identify methods to collect training data? How do you look at the constraints / risks with a proposed method?
- **Feature Engineering:** Can you come up with relevant ML features for your model? How do you identify the most important features for the specific task?
- **Modeling:** How do you explain modeling choices? Are you able to justify the decision to use a specific model? Can you explain the training process? Can you anticipate risks and how do you mitigate those risks?
- **Evaluation & Deployment:** Can you design consistent evaluation & deployment techniques? How do you justify and articulate your choice of metrics to track?

How to prep

To practice, take any well-known app and pick a system that could benefit from ML. For instance, how would you design a friend recommendation system for Twitter? Consider that originally the system was implemented using a handful of static 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. From there, you should:

- **Brush up on basic ML theory and algorithm details.** Be comfortable with concepts like overfitting and regularization.
- **Practice converting intuitive ideas to concrete features.** For example, "number of likes" can be a good feature suggestion, but a better feature might also involve normalization, smoothing, and bucketing.
- **Think about the end-to-end problem in a production environment.** 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?
- **Consider what questions you may want to ask the interviewer about the problem.** What type of information will help you make better design decisions or to identify the topic you'd like to tackle?
- **Analyze your approach** and point out pros / cons. Having a good toolset of several different algorithms and understanding the tradeoffs is helpful. For example, be able to explain the advantages of logistics regression compared to SVM.

Work out the above problems on paper or a whiteboard and just think about the ways to break them down. Watch public videos and learn how Google search ranking works.

Behavioral Interview

What can you expect?

The behavioral interview will consist of a 45-minute session. Your interviewer will want to learn about your background, what you're passionate about in tech, and what kind of impact you want to make.

What do we look for?

The purpose of the behavioral interview is to assess if a candidate will thrive in Meta's fast-paced and highly unstructured environment. To that end, we assess candidates on five signals that correlate with success at Meta:

- **Resolving Conflict:** What kind of disagreements have you had with colleagues and/or managers? How have you resolved them? Can you empathize with people whose points of view differ radically from yours?
- **Growing Continuously:** Your interviewer will be assessing your aptitude for seeking out opportunities for growth and learning. Do you take constructive criticism as an opportunity to improve? How have you approached improving your skills?

- **Embracing Ambiguity:** How do you operate in an ambiguous and quickly changing environment? Are you comfortable making decisions and maintaining high levels of productivity when you are missing information or lack clarity? How did you react when you had to quickly pivot away from a project due to a shift in priority?
- **Driving Results:** We will be evaluating your experience pushing yourself and others to deliver against goals and objectives. How do you demonstrate your impact? Are you self-directed in reaching goals despite challenges and roadblocks?
- **Communicating Effectively:** How well do you communicate with teams? What about cross-functional partners? How do you tailor your communications based on the work and/or the audience?

We may ask you to:

- Discuss available details of past and current projects
- Provide specific examples about what you did and the resulting impact.
- Share what you learned from a past situation.
- Talk about what you like about your current role and/or being a developer.

How to prep

Just like with the coding and design interviews, it's important to prepare ahead of time for interviews designed to get to know you better.

- **Know yourself.** Take the time to review your own résumé as your interviewer will almost certainly ask 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 & discussing how you learned, improved, and grew from your past experiences is valued.
- **Use the S.T.A.R. method** to mentally organize your thoughts. This will provoke a well- thought-out and chronological action of events. Easy to describe, easy to follow.
    - **S**—One or two sentences about the **SITUATION**: What happened?
    - **T**—Describe the **TASK**: What was your specific goal?
    - **A**—**ACTIONS** you took to overcome the obstacles and complete your objective.
    - **R**—The tangible / quantifiable **RESULTS** of the situation: How did it help the team / company?
- **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.

Appendix / Resources

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

Below you will find some helpful resources for your interview. Take a look through the list as you prepare.

About Meta

- Meta Life
- 2021 Founder's Letter
- Meta Quarterly Earnings
- Join our Meta Careers Talent Community Page for the latest updates.
- Meta Newsroom

Meta's Technical Environment

- Meta Engineering Facebook page
- Meta Code videos
- Meta Open Source website
- Engineering Leadership at Meta blog
- Meta Engineering Bootcamp

Coding Resources

- Mock Coding Interview (Password: 2018prep)
- Coding Interview Example (Password: fbprep)
- Cracking the Meta Coding Interview Videos (The Approach and Problem Walk-through) (Password: FBprod)
- Topcoder, GeeksQuiz, CareerCup

System Design Resources

- Design Questions Online Tutorial
- Grokking the System Design Interview (Video)
- System Design Primer (GitHub)
- HiredInTech System Design for Tech Interviews Course

Machine Learning

- Facebook Research: Machine Learning
- High Scalability

Thank you for taking the time to review this guide!

Meta