FACEBOOK

ANDROID

Initial Interview Guide

What You'll Find in This Guide

Welcome to your prep guide for your Android software engineer initial interview at the Facebook company. Our Android software engineers put together this guide so you know what to expect and how to prepare.

Time

Preparation and how much time your interview will take

Prep Time

Start your prep early. Before your interview date, set aside a specific amount of time every week to practice your coding, algorithmic, and problem-solving skills.

Interview: 45 Minutes

Your conversation with an engineer will be divided into the following time blocks:

- **Introductions:** 5 – 10 minutes
    - We'll talk about us and you'll talk about who you are, where you've trained and worked, and what your areas of expertise are. Your interviewer will ask you Android domain questions.
- **Coding:** 25 – 35 minutes
    - Typically, you'll have questions that exercise your knowledge of data structures and algorithms. You might have one long question, or several shorter questions.
- **Your Questions:** 5 minutes
    - Take this brief opportunity to learn more about working at Facebook from an engineer's point of view. Think about what you find interesting and challenging about the work you'd be doing.

Interview overview

What you can expect in your initial technical screen

You'll interview with an engineer, and you'll be remote on **CoderPad**. Please know that you won't be able to compile your code, and you'll need to run test cases through your code manually to check for bugs.

Before your tech screen, make sure the device(s) you'll use to connect are fully charged, you have a reliable and strong internet connection, and you have a hands-free setup (headphones, speaker and quiet environment, etc.) so you can code.

Content

- **Android knowledge and wisdom questions** probe your familiarity with building for the Android platform: for instance, how to use key APIs and how to solve common problems you'll encounter as a developer. We expect that you'll be able to answer most of these questions from your day-to-day work and shouldn't need to study much. You should of course be familiar with the basics: **Activities, Views, Fragments, Services, memory management, scrolling performance,** and **RecyclerView**.
- You should know about concurrency, networking, storage, rendering, and everything else involved in making a mobile app. **Think about strengths, weaknesses, tradeoffs, and constraints.** You want to have a good breadth of Android knowledge.
- **Coding problems** are general Java coding problems. Typically, you'll have questions that exercise your **knowledge of data structures and algorithms**. You might have one long question or several shorter questions. Be prepared also for technical questions involving **design patterns in Java and runtime complexity**.
- It may also help to review core **computer science concepts** as well as subjects pertaining to the scale of our environment. We may ask you for the runtime analysis of each problem.

For the coding questions, we'll ask you to recommend a solution, code that solution, analyze the runtime of your code, and possibly optimize your solution if there's time. Interviewers are looking for insight into your thought process, for creative solutions, and for the ability to work out more than one way to solve a problem. We'll also ask you to talk through your rationale for choosing a certain approach.

We may pose some of the coding problems as ambiguous, real-world problems, and we'll ask you to interpret the problem and apply your knowledge to find a solution.

How to prep for your initial screen

From the big picture to the specifics, this is how our engineers suggest you prepare

As you begin preparing, please **watch this video** (password: fbprep), which gives a good example of what Facebook coding interviews are like. While the code in this example is Python, we use many different coding languages at Facebook, so practice in the coding language that you are most confident in. Then, to practice:

- Do as many coding questions as you can. Visit **LeetCode**, **HackerRank**, **Interview Cake** or another site listed in the appendix of this guide. The idea isn't to see every question but to become familiar with the pattern of interpreting a question, formulating a solution, and writing an efficient, bug-free program without a compiler.
- Practice on a whiteboard or with pencil and paper. Practice under time pressure: coding speed is important. The more rigorous your training, the easier you'll find the interviews. Try to complete two coding problems in 30 minutes. If you're able to solve the medium-level questions within 15 – 20 minutes, it's usually a good sign.
- Review 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, and heaps. Also sorts, searches, and traversals (BFS, DFS). Also review recursion and iterative approaches.
    - Our typical coding questions aren't phrased as "Implement 'X'"—they're "Solve this problem." You can pick from a number of approaches. (No one is going to ask you to "Implement Knuth-Morris-Pratt" or "Construct a 2-3-4 tree.")
    - Your reasoning is important. Software engineering is all about tradeoffs, so be able to discuss those.

What success looks like

Considerations that lead to a positive outcome

Interviewers will weigh the success of an interview based on your approach as much as your answer. They'll funnel your performance based on the following considerations:

- How well do you know Android?
- Do you listen carefully and comprehend the question?
- Do you ask any necessary questions before proceeding?
- Do you notice and follow hints the interviewer gives?
- Are you quick to comprehend and solve problems?
- Do you find multiple solutions before selecting the best one?
- Do you keep seeking out new methods to tackle the problem?
- Are you inventive and flexible in your solutions, and open to new ideas?

10 tips for your initial screen

Engineers share insider views on what works best

1. Avoid misunderstanding.

When we ask you to provide a solution, first define and develop a framework of the problem as you see it. Ask for help or clarification, and spend 2 – 5 minutes asking the interviewer about corner cases on the problem. This will ensure that you've understood the problem correctly.

2. Think out loud.

It helps your interviewer to follow along, learn about your problem-solving skills, and provide hints if needed.

3. Write a working solution and iterate.

It's better to have a non-optimal but working solution than random fragments of an optimal but unfinished solution.

4. Hints.

If your interviewer gives you hints to improve your code, please use them.

5. Don't worry about memorizing tables of runtimes or API calls.

It's always good to know how to figure out approximate runtimes, but the code you write is more important.

6. If your solution is getting messy, step back.

Most coding interview questions are designed to have reasonably elegant solutions. If you have if-else blocks and special cases everywhere, you might be able to take a better approach. Look for patterns and try to generalize.

7. Plan your approach.

Ensure that you spend time planning your approach, but remember you can always go brute force and then optimize from there.

8. Clarification.

Make sure you're asking clarifying questions as you go (ensure you have all the info you need).

9. The cool things.

Think of cool things that you've done in engineering—you'll likely talk about things you've made.

10. Questions.

You'll most likely have some time at the end for questions for your interviewer. Some people find it easier to come up with a few questions in advance rather than think of them on the spot.

Appendix / resources

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

Our engineers collected some helpful resources with content and activities for your initial tech screen. Take a look through the list as you prepare.

Prep tutorial videos

- **Cracking the Facebook Coding Interview—The Approach** (password: FB_IPS)
- **Cracking the Facebook Coding Interview—Problem Walk-Through** (password: FB_IPS)
- **How to Crush Your Coding Interview at Facebook**
- **Sample Facebook Coding Interview** (password: fbprep)

Coding questions and exercises

- To study runtime complexity: **Big O Cheat Sheet**
- To see some of the questions Facebook has asked: **Glassdoor Interview Questions**
- The **Algorithm Design Canvas** captures the thought process when solving an algorithmic problem.
- Gauge how prepared you are on CS fundamentals with more than **400 questions** that will take from less than a minute to about an hour to solve.
- Engineer favorites for practicing coding problems:
  - **LeetCode**
  - **HiredInTech**
  - **HackerRank**
  - **CareerCup**
  - **CodeChef**
  - **GeeksforGeeks**

Resources from facebook

- **Timed Releases for Mobile Apps**
- **Get that Job at Facebook**
- **Facebook Engineering Bootcamp**
- **Engineering Notes**
- **Facebook Newsroom**
- **Facebook Quarterly Earnings**

Helpful books

- **Cracking the Coding Interview**
- **Introduction to Algorithms**
- **Algorithms in C**

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