Mobile App Developer interview questions

careertips.com.au: common questions and what they're really testing

What to expect

Interviews for Mobile App Developer roles usually mix technical depth (language and platform knowledge) with practical judgement on how you handle the messiness of real devices, real users and shifting requirements. Expect a technical component alongside standard behavioural questions.

- **Technical:** Questions on Swift, Kotlin, React Native or Flutter, plus how you structure and debug mobile code.
- **Process/workflow:** How you handle device testing, API integration and app store deployment in practice.
- **Behavioural:** Past examples of collaborating with designers, handling bugs under deadline, or dealing with ambiguous requirements.
- **Scenario/judgement:** Hypothetical situations, such as a critical bug found post-release, to see how you triage and respond.

Most interviews open with a walkthrough of your background and recent projects, move into technical questions (sometimes a live coding or whiteboard exercise, or a take-home task reviewed in the interview), then cover behavioural and scenario questions, and close with your questions about the team and tech stack.

1. Walk me through how you'd take a feature from design handoff to a released build.

Why they ask

Tests whether you understand the full mobile development lifecycle, not just how to write code.

How to structure your answer

Describe each stage in order: reviewing the design and clarifying edge cases with the designer, building the UI, integrating any required APIs, testing across devices, then handling app store submission and post-release monitoring.

Example answer

“I start by reviewing the Figma file with the designer to flag anything that won't translate well to a responsive layout, like fixed pixel spacing. Once requirements are clear, I build the UI first against static data, then integrate the API for real data and sync. Before raising a pull request I test on a couple of physical devices and at least one older OS version, since simulators don't always catch everything. After code review and merge, I keep an eye on crash reporting and user feedback for the first few days after release in case anything slipped through QA.”

2. Tell me about a time you had to fix a bug that only showed up on certain devices or OS versions.

Why they ask

Device fragmentation is one of the defining challenges of mobile development, and this checks your debugging process under a real constraint.

How to structure your answer

Use STAR: situation, task, action, result.

Example answer

“A user reported that a form screen was rendering with overlapping fields, but only on one specific Android screen size. I couldn't reproduce it on my usual test devices, so I pulled the exact device model from the crash reporting tool and tested on an emulator matching that resolution. It turned out a

hardcoded height value wasn't scaling properly. I replaced it with a constraint-based layout and retested across several screen sizes to confirm the fix held. The bug reports for that screen stopped after the next release."

3. How do you decide between native development and a cross-platform framework like React Native or Flutter for a given project?

Why they ask

Checks your practical understanding of the tradeoffs, since this is a real decision teams face, not just theoretical knowledge.

How to structure your answer

State the key factors you weigh, then give a concrete example of how you'd apply them.

Example answer

"It depends on how performance-sensitive the app is and how much platform-specific functionality it needs. If the app is heavy on custom animations or needs deep integration with device hardware, I'd lean native with Swift and Kotlin. For a more standard app where speed of development and a shared codebase across iOS and Android matter more, Flutter or React Native usually makes sense. I've worked in both, and I try to make the call based on the product's actual requirements rather than defaulting to whichever framework I know best."

4. A critical bug is found in the app the day after a major release. How do you handle it?

Why they ask

Tests judgement under pressure and understanding of release and rollback processes.

How to structure your answer

Talk through immediate triage, communication, and the fix and release process, in that order.

Example answer

"First I'd confirm the severity and how many users it's affecting, using crash reports or analytics if available. If it's serious, I'd flag it to the team immediately so we can decide whether to roll back or push a hotfix. I'd reproduce the bug locally, write a fix, and get it reviewed quickly rather than skipping review just because of time pressure. Once tested, I'd push it through an expedited release process and keep monitoring after it goes live to confirm the issue is actually resolved."

5. How do you approach testing an app across multiple devices and OS versions with limited time?

Why they ask

Directly reflects one of the core tasks of the role and checks whether you have a sensible, risk-based approach rather than testing everything exhaustively or nothing at all.

How to structure your answer

Describe your prioritisation logic and any tools you use.

Example answer

"I look at analytics to see which devices and OS versions our actual users are on and prioritise those first. I keep a small set of physical devices covering the oldest supported OS version and the most common screen sizes, and use emulators or a device cloud service for anything outside that. I focus manual testing on the areas most affected by the current change, and rely on automated tests for regressions elsewhere."

6. Describe a time you disagreed with a designer or product manager about how a feature should work.

Why they ask

Mobile developers work closely with UX and product, so this checks collaboration style and how you handle pushback.

How to structure your answer

STAR, with emphasis on how you communicated the disagreement.

Example answer

“A designer wanted a multi-step animated transition between screens that I was concerned would hurt performance on older Android devices. Instead of just pushing back, I built a quick prototype showing the frame rate on a low-end test device, which made the tradeoff concrete rather than abstract. We ended up agreeing on a simplified version of the animation that kept most of the visual intent without the performance hit.”