

Android developer (fresher)

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Full-time | CTC: [CTC for the full-time role]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

The [Team] team at [Company name] needs a fresher Android developer. You will build and fix features in our Android app: turning designs into screens, connecting them to our APIs, reading crash reports and testing on phones as well as the emulator. A senior developer reviews your work and explains how the app is built and released. The role suits someone who likes seeing their work run on a phone and wants to find out why an app misbehaves on one device.

What you will do

- Build and change screens from Figma designs or image mockups using [Jetpack Compose with Kotlin / XML layouts with Kotlin / XML layouts with Java / XML layouts in a mixed Kotlin and Java project], and compare each screen with the design on a small and a large phone
- Call the app's REST APIs off the main thread in the way the code base already does, against a test server with sample data until a senior approves access to real user data. Turn the JSON into the app's own data classes and show loading, empty and error states for every call
- Store settings, saved lists and offline data on the phone with the local storage the code base uses, and check that the data is still there after the app is closed and reopened
- Handle runtime permissions such as camera, location and notifications, so that the app still works, and explains itself, when the user says no
- Make each screen survive being turned sideways, being sent to the background and being brought back, without losing what the user typed
- Read Logcat and crash reports, for example from Firebase Crashlytics or the Play Console with the read-only access a senior gives you, reproduce the crash, fix the cause and write the cause in the ticket; keep personal data, passwords and tokens out of the logs you write
- Write unit tests for the logic you change and add a test for each bug you fix
- Test every change on an emulator and on the team's test phones: a small and a large screen, an older Android version and, where our users have them, a slower phone and a poor network; note what you checked
- Build the test version of the app for QA with the team's build steps and fix what QA reports; a senior handles the release to the Play Store
- Keep app-signing private keys and server credentials in the team's approved secret store, outside source control and the app package. Treat values compiled from build settings or environment variables as readable. Handle user access and refresh tokens obtained at runtime through the app's approved authentication and secure-storage code

What we are looking for

- Kotlin or Java fundamentals, as the post states, with object-oriented basics: classes, interfaces, collections, exceptions and null handling
- Android Studio: creating a project, running it on an emulator or a phone, and using the debugger

- One Android app you built and can run and explain screen by screen, shown through a repository or a short demonstration, or talked through if the work is private
- The screen lifecycle in plain words: what happens when a screen opens, goes to the background, comes back or is rotated
- REST APIs and JSON: calling an endpoint, reading the response and handling a slow or failed call
- Reading Logcat and a stack trace to find the failing line
- Git basics: commits, branches, pull requests and merge conflicts

Good to have

- Practice with the UI toolkit the post names, Jetpack Compose or XML layouts
- Retrofit or OkHttp for API calls, and Kotlin coroutines for background work
- Room or DataStore for saving data on the phone
- MVVM with a ViewModel, or another way of keeping screen code apart from data code
- JUnit unit tests, and Espresso or Compose UI tests
- Accessibility basics: TalkBack, content descriptions and large font sizes

Who can apply

Final-year students and graduates of any stream can apply, including B.E./B.Tech, BCA, B.Sc, MCA and diploma holders in computer science, IT or electronics. Judge every applicant on the same skills: Kotlin or Java basics, Android Studio and one app they can explain.

How to apply

Apply through this job post by [\[Apply by\]](#). Shortlisted candidates will hear from us about the next steps.

Android developer intern

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Internship | Duration: [Internship duration] | Stipend: [Stipend for the internship]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

[Company name] offers an Android internship with a mentor. You will build one screen from a design, connect it to a test API with fictional sample data, fix small bugs and check your work on [an emulator and a company test phone / an emulator only]. Interns use test builds and sample data only; senior developers handle releases. [Company name] must approve any publication or portfolio use of your work.

What you will do

- Build one screen from a design in the toolkit your mentor names, and compare it with the design on two screen sizes
- Connect that screen to a test API that returns fictional sample data, and show loading, empty and error states
- Fix a few small bugs picked by your mentor and write the cause of each in the ticket
- Find the crash in a practice app that your mentor has broken on purpose, locate the line in Logcat and fix it
- Add unit tests for one small class, such as a price formatter or a form validator
- Turn the phone sideways, send the app to the background and bring it back, then list what breaks and why

What we are looking for

- A current student or recent graduate of any stream
- Knows the basics of Kotlin or Java: variables, functions, classes and lists
- Has run an Android app from Android Studio on an emulator or a phone, even from a course
- Can describe an app or a screen they built, even if the code is private
- Knows what a commit and a branch are in Git, or is ready to learn both in week one

Who can apply

Final-year students and graduates of any stream can apply, including B.E./B.Tech, BCA, B.Sc, MCA and diploma holders in computer science, IT or electronics. Judge every applicant on the same skills: Kotlin or Java basics, Android Studio and one app they can explain.

How to apply

Apply through this job post by [Apply by]. Shortlisted candidates will hear from us about the next steps.

Screening questions to add to your job post

1. **What happens to your screen and to what the user typed when the phone is rotated, and when the app goes to the background?** Look for explaining that rotation normally recreates the activity and that user input needs preserving. Accept ViewModel, saved state, automatic View state restoration or rememberSaveable, with an explanation of what survives. Also accept an accurate explanation of an app that handles configuration changes itself. For backgrounding, look for recognising possible process death: a ViewModel alone does not survive it. Plain words are enough; lifecycle method names are not required.
2. **Why should a network call not run on the main thread, and how do you keep it off in your own code?** Look for explaining that a blocking network call can freeze drawing and touch handling. Accept an asynchronous networking call, an executor or a coroutine that moves blocking work to an appropriate dispatcher. A main-safe library call also counts. Saying 'use coroutines' alone is incomplete; ask how the candidate's approach keeps blocking work off the main thread.
3. **Your app crashes only on one colleague's phone. What do you check?** Look for reading the crash report or Logcat first, then comparing that phone with the others: Android version, screen size, memory, permissions and free storage. Accept a plan to borrow the phone or to match its settings in an emulator.
4. **A request to the server fails or times out. What does the user see, and what can they do?** Look for a message in plain words, a retry option, saved data still shown where it exists, and different messages for no connection and for a server error. Accept a snackbar, a full-screen message or a dialog.
5. **A response from the server leaves out a field your screen needs. What happens in your code, and what should happen?** Look for explaining whether the missing field causes parsing to fail or reaches the screen as a missing value. Optional information may use an agreed default or be hidden; missing required information should produce a clear error state rather than an invented value or a crash. Accept Kotlin or Java and an explanation based on the library the candidate used.
6. **Pick one screen of an app you built and tell us where its data comes from and what it shows while the data loads. Show it through a repository or a short demonstration, or talk us through it if the work is private.** Look for a clear path from the data source to the screen, and a loading state. Accept a course project that the candidate has changed and can explain.
7. **Which screen sizes or Android versions have you run your app on, on an emulator or a phone, and what looked different?** Look for at least one real comparison, such as text cut off on a small screen or a feature that behaves differently on an older version. Accept emulators; owning several phones is not needed.