

30-DAY ANDROID INTERVIEW PLAN

Your daily roadmap to crack Android
interviews.



BY: ANDROID ENGINEERS

<https://androidengineers.in>

HOW TO USE THIS PLAN

- Spend 1-2 hours daily.
- Tick off each topic as you finish it.
- By Day 30, you'll have covered the most common Android interview areas.
- For explanations & mock interviews → check my 1:1 Mentorship or Masterclass (links at end).



WEEK 1: ANDROID BASICS & COMPONENTS

Day 1: Activity & Fragment Lifecycle

- ☐ Revise Activity lifecycle methods (onCreate, onStart, onResume, etc.)
- ☐ Understand Fragment lifecycle differences
- ☐ Explore what happens during configuration changes (rotation)
- ☐ Note common interview Qs: “How do you handle screen rotation?”
- ☐ Practice by drawing lifecycle diagram from memory

Day 2: Intents & Navigation

- ☐ Understand Explicit vs Implicit Intents
- ☐ Learn how to pass data using Intent extras
- ☐ Review startActivityForResult vs Activity Result API
- ☐ Explore Android Navigation Component basics
- ☐ Practice: Open phone dialer using Intent

Day 3: Services & Broadcast Receivers

- ☐ Difference between Service, IntentService & JobIntentService
- ☐ Foreground Service vs Background Service (when to use each)
- ☐ Dynamic vs Manifest-declared Broadcast Receivers
- ☐ Understand restrictions on background services in modern Android
- ☐ Practice: Build a simple Music Player Service



WEEK 1: ANDROID BASICS & COMPONENTS

Day 4: Content Providers & Storage

- SharedPreferences — when to use
- Room Database basics (Entity, DAO, Database)
- DataStore (vs SharedPreferences)
- Content Provider basics (why they exist, when used)
- Practice: Store and retrieve a list of tasks with Room

Day 5: Permissions & Security Basics

- Dangerous vs Normal permissions
- Runtime permissions flow (request → callback)
- Best practices for handling “Don’t ask again”
- Storage permission changes in Android 11+ (Scoped Storage)
- Interview Q: “How do you secure sensitive data in Android?”

Day 6: Testing Basics

- JUnit basics (assertions, test annotations)
- Mockito/MockK basics (mocking dependencies)
- Espresso basics (UI testing selectors)
- Understand difference between Unit Test & Instrumentation Test
- Practice: Write a simple unit test for a Calculator function



WEEK 1: ANDROID BASICS & COMPONENTS

Day 7: Weekly Review + Quick Quiz

- ☐ Summarize lifecycles, Intents, Services
- ☐ Revise differences: Service vs IntentService, SharedPreferences vs DataStore
- ☐ Take a small mock quiz (10 Qs from above topics)
- ☐ Self-reflect: Which 2 topics felt hardest? (mark for revision)
- ☐ If stuck → [schedule a mock session / 1:1 review](#)



WEEK 2: ADVANCED UI & DATA HANDLING

Day 8: Jetpack Compose Basics

- Understand @Composable, Composition, and Recomposition
- Learn basic primitives: Text, Image, Button, Spacer
- Preview & recomposition triggers (remember vs state)
- Understand Modifier chain basics (padding, size, background, click)
- Practice: Build a simple “Hello + Counter” screen (increment, reset)

Day 9: Compose Layouts (Row, Column, Box)

- Master alignment/arrangement (horizontal/vertical)
- Use weight, fillMaxWidth/Height wisely
- Box with contentAlignment + stacking elements
- Intrinsic sizing & when to avoid fixed heights
- Practice: Create a card with avatar, title, subtitle, and CTA row

Day 10: State & State Hoisting

- Understand mutableStateOf, remember, rememberSaveable
- One-way data flow; lift state to parent (state hoisting)
- Avoid state duplication; single source of truth
- Side effects overview: LaunchedEffect, rememberUpdatedState (just awareness)
- Practice: Build a SearchBar that hoists query to parent



WEEK 2: ADVANCED UI & DATA HANDLING

Day 11: RecyclerView vs LazyColumn

- When Compose lists (LazyColumn/LazyRow) beat RecyclerView
- items vs itemsIndexed; keys and why keys matter
- Item reuse, sticky headers basics, item separators
- Performance tips: avoid heavy recomposition, use derivedStateOf if needed
- Practice: Build a LazyColumn of articles with thumbnail + title + time

Day 12: Networking with Retrofit/OkHttp + Coroutines

- Create Retrofit service with suspend functions
- OkHttp interceptors (logging, headers)
- Coroutine scopes on Android (ViewModelScope)
- Error handling: try/catch + Result wrapper
- Practice: Call a public API and log the response safely



WEEK 2: ADVANCED UI & DATA HANDLING

Day 13: JSON Parsing & API Integration

- Choose parser (Moshi/Kotlinx Serialization) & set up adapters
- Map DTO → domain model; avoid leaking API models to UI
- Handle nullability, default values, and versioned fields
- Loading/Error/Empty UI states pattern
- Practice: Display API data in a LazyColumn with loading + retry

Day 14: Mini Project — News App in Compose

- Screen 1: Headlines list (image, title, source, time)
- Screen 2: Details page (open on click)
- Pull-to-refresh + basic pagination (load more)
- Handle offline/slow network: show cached/placeholder states
- Extras (optional): Save article (in-memory or Room), share intent



WEEK 3: OOP, KOTLIN & ARCHITECTURE PATTERNS

Day 15: OOP in Kotlin (Encapsulation, Inheritance, Polymorphism, Abstraction)

- Data class vs class vs object vs companion object
- Encapsulation: visibility modifiers (private/internal/protected) + immutability
- Inheritance & composition: prefer composition; sealed class hierarchies
- Polymorphism: interfaces vs abstract classes; override/implement
- Abstraction in Android: Repository, DataSource contracts, Navigator interfaces
- Practice: Model a Result hierarchy (Success/Error/Loading) with sealed classes

Day 16: Kotlin Coroutines Basics (suspend, launch, async)

- suspend functions & structured concurrency basics
- Coroutine builders: launch vs async (when to use await)
- Dispatchers: Main, IO, Default; switching with withContext
- Scope rules: viewModelScope, lifecycleScope; cancelling onCleared
- Exception handling: try/catch, CoroutineExceptionHandler, supervisorJob
- Practice: Call a suspend API from ViewModel, expose state to UI



WEEK 3: OOP, KOTLIN & ARCHITECTURE PATTERNS

Day 17: Coroutines Advanced (flows, structured concurrency)

- Flow vs Channel vs LiveData; cold vs hot streams
- Key operators: map, flatMapLatest, debounce, distinctUntilChanged, catch
- StateFlow vs SharedFlow: replay, conflation, configuration
- Backpressure & performance: buffer, conflate, throttle/debounce wisely
- Testing flows: Turbine (or manual) + TestScope/TestDispatcher
- Practice: Search-as-you-type with debounce + cancel previous calls

Day 18: Architecture — MVVM/MVI

- MVVM essentials: View → ViewModel → Repository boundaries
- UI state modeling: single immutable data class for screen state
- One-time events: Event wrappers or SharedFlow for navigation/snacks
- MVI basics: Intent → Reducer → New State (when to prefer MVI)
- Error/loading handling patterns (Result, Either)
- Practice: Convert a screen to MVVM with StateFlow + a clear reducer



WEEK 3: OOP, KOTLIN & ARCHITECTURE PATTERNS

Day 19: Clean Architecture Basics + SOLID Principles

- Layering: presentation, domain (use cases), data (repo/datasources)
- Dependency direction: domain is pure; UI/data depend on domain
- Use Cases: single responsibility, pure business logic, testable
- SOLID quick tour:
 - S: Single Responsibility (one reason to change)
 - O: Open/Closed (use interfaces, sealed)
 - L: Liskov Substitution (respect contracts)
 - I: Interface Segregation (small, focused interfaces)
 - D: Dependency Inversion (depend on abstractions)
 - □ Practice: Extract “SaveTodo” & “GetTodos” use cases; unit test them

Day 20: Dependency Injection (Hilt/Dagger/Koin Basics)

- Why DI: testability, modularity, lifecycle-aware objects
- Hilt setup: Application, EntryPoints, @Module/@Provides/@Binds
- Scopes: Singleton, ActivityRetained, ViewModel, Activity/Fragment
- Qualifiers for multiple implementations (e.g., @IoDispatcher)
- Koin quick start (if preferred): modules, single/factory, get()
- Practice: Inject Repository, Dispatcher, and Logger into ViewModel via DI



WEEK 3: OOP, KOTLIN & ARCHITECTURE PATTERNS

Day 21: Mini Project — Todo App (MVVM + Hilt + Coroutines)

- Features: add/delete/toggle complete, filter (All/Active/Done)
- Layers:
 - Domain: Todo entity + use cases (Add, Toggle, GetAll)
 - Data: Room (DAO + DB), Repository (maps DAO ↔ domain)
 - Presentation: ViewModel with StateFlow (list, filter, isLoading, error)
 - UI: Compose list (LazyColumn), input row, empty state, undo via snackbar
 - DI: Hilt modules for DB, DAO, Repo, UseCases, Dispatchers
 - Concurrency: use IO dispatcher for DB; expose StateFlow to UI
 - Testing: unit test use cases; instrument DAO test with in-memory Room
 - Extras (optional): simple analytics event on add/delete; basic nav



WEEK 4: SYSTEM DESIGN & MOCK PREP

Day 22: Modularization in Android Projects

- Understand app, feature, library, and core modules (what goes where)
- Define clear boundaries: :app (composition), :feature:* (UI + VM), :core:* (common, network, db, design)
- Configure Gradle dependencies: API vs implementation, kapt/ksp placement
- Public vs internal APIs; keep domain independent of Android
- Build speeds: configuration cache, Gradle parallelism, dependency constraints
- Practice: Extract one screen into :feature:news with a ui + domain + data split

Day 23: Offline-first & Caching Strategies

- Cache layers: memory → disk (Room) → network (Retrofit)
- Single source of truth (SSOT) pattern: Room emits, network updates
- Stale-while-revalidate flow; cache invalidation policies (time-based, versioned)
- Paging (snapshot vs streaming), prefetch & placeholders basics
- Error modes: offline fallback, partial data UI, retry/backoff strategy
- Practice: Add cached feed with Room + periodic refresh via WorkManager



WEEK 4: SYSTEM DESIGN & MOCK PREP

Day 24: WorkManager & Background Tasks

- When to use WorkManager vs Foreground Service vs AlarmManager
- OneTime vs Periodic work; constraints (network, charging, idle)
- Chaining works, input/output data, unique work policies
- Backoff policies (linear/exponential); idempotency considerations
- Observing work state from UI (LiveData/Flow)
- Practice: Schedule periodic sync + retry with exponential backoff

Day 25: Feature Flags, Configs & Crash Reporting

- Local vs remote flags (Firebase Remote Config, custom backend)
- Safe rollout: defaults, kill switches, staged rollout, % experiments
- Separate runtime config from build-time (BuildConfig vs remote)
- Crash reporting setup: Firebase Crashlytics/Sentry basics; custom keys/breadcrumbs
- Non-fatal logging; ANR & cold start tracking (basic metrics)
- Practice: Guard a new feature with a flag + add a Crashlytics breadcrumb



WEEK 4: SYSTEM DESIGN & MOCK PREP

Day 26: Mock Interview — Resume Walkthrough

- Prepare a 90-second “pitch”: current role, 1-2 key projects, impact in numbers
- STAR stories (Situation, Task, Action, Result) for 3 projects
- Be ready to whiteboard your app architecture from memory
- Pre-empt gaps: “What I’d do differently now...” (shows growth mindset)
- Practice: Record yourself once; trim filler words; tighten to 90s

Day 27: Mock Interview — Explain Your Project Architecture

- Draw layers: UI ↔ ViewModel ↔ UseCases ↔ Repository ↔ DataSources
- Explain choices: MVVM + coroutines + DI; tradeoffs vs alternatives (MVI/Rx)
- Non-functional: caching, pagination, error handling, logging
- Testing: what is unit vs instrumentation vs UI; what you actually tested
- Practice: 5-minute crisp architecture walkthrough + Q&A on tradeoffs



WEEK 4: SYSTEM DESIGN & MOCK PREP

Day 28: Mock Interview — Live Coding (API + UI)

- ❑ Set up: Retrofit service + suspend call + DTO → domain mapping
- ❑ Display list with LazyColumn (loading/empty/error states)
- ❑ Add search or filter to show state handling and recomposition awareness
- ❑ Clean commit habits: small steps, meaningful messages
- ❑ Practice: Timebox to 30–40 minutes; narrate thought process as you code

Day 29: Mock Interview — Debugging Session

- ❑ Crash triage: read stack trace, reproduce, isolate minimal test case
- ❑ Network failures: timeouts, 4xx/5xx, malformed JSON → show graceful UI
- ❑ Perf issues: measure before guessing (Profiler basics, over-recomposition hints)
- ❑ Flaky UI: keys in Lazy lists, remember/derivedStateOf usage, lifecycle leaks
- ❑ Practice: Take a known bug and speak aloud your debugging steps



WEEK 4: SYSTEM DESIGN & MOCK PREP

Day 30: Final Review & Next Steps

- ☐ Revisit hardest topics; mark 3 to revise next week
- ☐ Polish your resume bullets with measurable outcomes (X% crash drop, Y ms startup)
- ☐ Update GitHub README/demo APKs; add screen recordings/GIFs
- ☐ Book one real mock (peer or mentor) & schedule 2 interviews
- ☐ Decide your path: self-study vs 1:1



<https://androidengineers.in>

CONGRATS! YOU COMPLETED THE 30-DAY ANDROID INTERVIEW PLAN ✓

Discipline beats motivation — you've proved it.

Path 1: Solo Ride (Free)

📖 Keep using this checklist

💡 Build projects

📺 Follow me on [LinkedIn](#) & [YouTube](#)

Path 2: Guided Growth (Faster)

🎨 1:1 Mentorship

📺 Masterclass

📍 ["Book Your Spot"](#)

★★★★★

"I had an exceptional experience with the 1:1 mentorship session. Akshay was incredibly friendly and provided invaluable guidance on focusing on long-term goals. They also gave great...

Nayab khan
11th Sep, 2024

★★★★★

"Cleared my major points for what I am missing in the resume and also suggested what I can work on for further growth in the career."

Ketan Chaurasiya
7th Aug, 2024

★ 5/5

He instantly got on a call and started noting down my intro details and started to figure out what my needs are. I am preparing for a Google interview and I got all the insights that I need right from how to prepare for the coding interview. Im a experienced android dev he also gave me good insights on the Android part as well. He asked me to concentrate more on coding round as I was very weak in it. He was very receptive in listening to all the questions that I had and have me proper answers for them as well... Thank you Akshay. Will prepare and might ask him to do mock interviews.

Anonymous
04 Nov 2024

★ 5/5

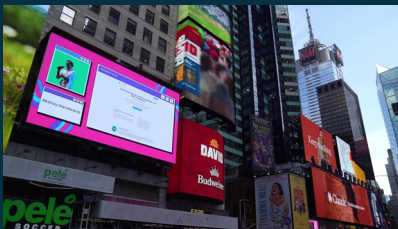
I recently had the pleasure of attending a Google interview consultation with Akshay, and it was an incredible experience. He was not only insightful and knowledgeable but also very approachable and open to questions. Akshay's guidance was clear, concise, and straight to the point, making complex topics easy to understand. I genuinely enjoyed the session and left feeling more confident and prepared. Highly recommend his expertise to anyone preparing for their next big opportunity!

Adithyan
04 Nov 2024

★ 5/5

What impressed me most was his personalized approach and practical tips that I could immediately apply. Akshay's guidance not only improved my technical skills but also boosted my confidence in navigating my career path. His insights and encouragement have been a game-changer for me. I highly recommend Akshay's mentorship to anyone looking to advance their Android development career.

Harsh Mahapatra
04 Aug 2024



From a one-room house in Udaipur to Times Square, my journey was built on community. Now you're part of it. 🚀



Akshay Nandwana
Senior Android Engineer

