Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: ANDR • **Updated:** 01 Nov 2026
https://mockattempt.com/courses/android-training

ANDR • NATIVE MOBILE ENGINEERING (KOTLIN & ANDROID) **BEGINNER TO SENIOR ANDROID KOTLIN ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Android Training

Master production Android development from ART Runtime, AndroidManifest & Activity/Fragment Lifecycles to Intents, RecyclerView DiffUtil, TabLayout, DataStore/SharedPreferences, SQLite/Room, JSON Networking, Foreground Services, Notifications & Firebase Google Sign-In — with runnable Kotlin programs, 10×50 MCQ Mock Tests per topic, and Spoken Voice + Gemini AI Interviews at ₹4,999.

Duration: 15–17 Weeks • Live + Hands-On Projects (212 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

Fixed Non-Negotiable Tuition Fee: Base Fee **₹4,999** + 18% GST (₹900) = **Total Incl. GST: ₹5,899**

Live Batch Tracks: Weekday Live Online Batch (Mon – Fri) (Monday to Friday (5 Live Interactive Classes / Week) • 1.5 Hours / Day (60 Mins Live Architecture & Coding + 30 Mins Hands-On Lab) • 18–20 Weeks Fast-Track) | Weekend Intensive Live Online Batch (Sat – Sun) (Saturday & Sunday (Designed for Working Professionals & College Students) • 3.0 Hours / Day (2 Hrs Deep-Dive Live Coding + 1 Hr Guided Project Build) • 20–22 Weeks Weekend Track)

Core Technologies & Engineering Stack: Kotlin 2.0 & Coroutines • Android SDK (API 35) & ART • Activities, Fragments & Intents • RecyclerView, DiffUtil & Material 3 • DataStore, SharedPreferences & SQLite/Room • Volley / Retrofit & JSON Parsing • Services, BroadcastReceivers & Firebase Auth

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Understand the Android Linux Kernel, HAL, ART AOT/JIT compilation pipeline, UID app sandboxing, and AndroidManifest component declarations
2. Architect lifecycle-resilient Activities, Fragments, Explicit/Implicit Intents, ViewBinding UI layouts, and 60fps RecyclerView lists with DiffUtil
3. Persist offline data safely using SharedPreferences/DataStore and parameterized SQLite/Room transactions while consuming REST JSON APIs asynchronously
4. Clear 200 Timed Mock Exams (10,000 Android MCQs) and 200 Spoken Voice + Gemini AI Native Mobile Technical Interviews
5. 100% Kotlin + declarative Jetpack Compose Material 3 UI with unidirectional `StateFlow` / `UiState` architecture.
6. Clean Architecture layers (UI → ViewModel → UseCase → Repository) wired via Hilt Dependency Injection.
7. Offline-first local persistence with Room SQLite, Retrofit + OkHttp networking, and Kotlin Coroutines/Flows.
8. Reliable background execution with WorkManager, lifecycle awareness, and runtime permission best practices.

Eligibility & Prerequisites: Basic familiarity with OOP fundamentals (variables, classes, functions) — Kotlin syntax and Android Studio setup are covered step-by-step • A 64-bit Windows, macOS, or Linux machine (8 GB+ RAM recommended) capable of running Android Studio Ladybug+ or physical USB debugging • No prior Android industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Basic computer & internet familiarity on a Windows, macOS, or Linux laptop (8 GB+ RAM recommended for smooth live coding labs). • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED MODULE & TOPIC-WISE SYLLABUS (16 MODULES • 42 LIVE TOPICS)

MODULE 1 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Android Ecosystem, Linux Kernel Foundations & ART Software Stack

Explore the Android Open Source Project (AOSP) architecture, SDK API level targeting, Linux kernel UID sandboxing, Hardware Abstraction Layer (HAL), and Android Runtime (ART) DEX compilation.

Objectives: ✓ Differentiate minSdk, targetSdk, and compileSdk in Gradle Kotlin DSL builds • ✓ Trace how Kotlin bytecode compiles into DEX files executed by ART via AOT, JIT, and Baseline Profiles • ✓ Explain Linux UID process isolation, Binder IPC, and Zygote memory zygoting across Android apps

1.1 Android Platform Overview — AOSP, Gradle SDK Levels & App Packaging (APK / AAB) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Platform Overview — AOSP, Gradle SDK Levels & App Packaging (APK / AAB) in real Android systems — specifically: How Android powers billion...

1.2 Android Software Stack — Linux Kernel, Binder IPC, HAL, Zygote & ART Runtime [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Software Stack — Linux Kernel, Binder IPC, HAL, Zygote & ART Runtime in real Android systems — specifically: Deep dive into Android's 5-la...

Module 1 Hands-On Capstone Project (100 Marks): Android SDK Compatibility & ART Runtime Diagnostics Inspector — Build a Kotlin diagnostic utility module (`SdkInspector`) that evaluates `minSdk`, `targetSdk`, and device API levels, selects safe backward-compatible execution paths without `NoSuchMethodE...

MODULE 2 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Core Android Building Blocks & AndroidManifest Security Contracts

Master Android's four foundational application components (Activities, Services, BroadcastReceivers, ContentProviders), Application Context vs Activity Context, and strict `AndroidManifest.xml` declarations.

Objectives: ✓ Distinguish the roles and lifespans of Activity, Service, BroadcastReceiver, and ContentProvider • ✓ Prevent static Context memory leaks by choosing Application Context vs Activity UI Context properly • ✓ Configure `AndroidManifest.xml` permissions, intent filters, and mandatory `android:exported` flags on API 31+

2.1 Android Core Building Blocks — Activities, Services, BroadcastReceivers & ContentProviders [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Core Building Blocks — Activities, Services, BroadcastReceivers & ContentProviders in real Android systems — specifically: How Android dec...

2.2 AndroidManifest.xml — Component Registration, `android:exported` & Permission Enforcement [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply AndroidManifest.xml — Component Registration, `android:exported` & Permission Enforcement in real Android systems — specifically: Configure `<mani...

Module 2 Hands-On Capstone Project (100 Marks): AndroidManifest Security Auditor & Component Export Validator — Engineer a static verification engine that inspects parsed Android component descriptors (`Activity`, `Service`, `Receiver`, `Provider`), flags missing `android:exported` attributes on compo...

MODULE 3 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Activity Lifecycle State Machine & Implicit / Explicit Intent Architecture

Master the 7 Activity lifecycle callbacks, configuration changes (rotation/multi-window), `ViewModel` + `SavedStateHandle` survival, and Android `Intent` action/category/URI resolution.

Objectives: ✓ Trace `onCreate` -> `onStart` -> `onResume` -> `onPause` -> `onStop` -> `onDestroy` across rotation and process death • ✓ Preserve UI state across configuration changes and OS background process eviction • ✓ Construct Implicit Intents with `ACTION_VIEW` / `ACTION_SEND`, MIME types, and `ActivityResultContracts`

3.1 Activity Lifecycle — 7 Callbacks, Configuration Changes, `ViewModel` & Process Death [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Activity Lifecycle — 7 Callbacks, Configuration Changes, `ViewModel` & Process Death in real Android systems — specifically: Understand visible vs...

3.2 Android Intents — Implicit vs Explicit Resolution, Intent Filters, Flags & `FileProvider` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Intents — Implicit vs Explicit Resolution, Intent Filters, Flags & `FileProvider` in real Android systems — specifically: How `Intent` mes...

Module 3 Hands-On Capstone Project (100 Marks): Configuration-Change Resilient Exam Timer & Safe Intent Dispatcher — Implement an Activity lifecycle state tracker paired with a `SavedState` exam timer that survives screen rotation and process recreation, plus a safe Implicit Intent URI dispatcher that guar...

MODULE 4 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Explicit Navigation Contracts, `@Parcelize` Payloads & Modular Jetpack Fragments

Pass strongly-typed Kotlin data across screens using Explicit Intents and `@Parcelize`, and build adaptive phone/tablet UIs with Jetpack Fragments, `FragmentManager`, and `viewLifecycleOwner`.

Objectives: ✓ Create type-safe Explicit Intent factory methods (`newIntent()`) and `@Parcelize` DTOs over slow Java `Serializable` • ✓ Manage Fragment transactions (`commit`, `replace`, `addToBackStack`) via `supportFragmentManager` • ✓ Eliminate Fragment View memory leaks by scoping `LiveData`/`Flow`

collection strictly to `viewLifecycleOwner`

4.1 Explicit Intents — Type-Safe Screen Navigation, Companion `newIntent()` & `@Parcelize` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Explicit Intents — Type-Safe Screen Navigation, Companion `newIntent()` & `@Parcelize` in real Android systems — specifically: Navigate determinis...

4.2 Android Fragments — Dual Lifecycles (`this` vs `viewLifecycleOwner`) & `FragmentManager` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Fragments — Dual Lifecycles (`this` vs `viewLifecycleOwner`) & `FragmentManager` in real Android systems — specifically: Master modular UI...

Module 4 Hands-On Capstone Project (100 Marks): Adaptive Master-Detail Course Viewer with Fragment BackStack & Parcelable Args — Build an adaptive navigation controller supporting single-pane phone backstack navigation and dual-pane tablet split-views using zero non-default Fragment constructors and safe `Bundle` argu...

MODULE 5 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Material 3 UI Widgets, ViewBinding & Measure/Layout Rendering Mechanics

Build accessible Material 3 UI screens using `ViewBinding`, density-independent units (`dp`/`sp`), input validation widgets, and shallow view trees with `LinearLayout` weights vs `ConstraintLayout`.

Objectives: ✓ Replace legacy null-unsafe `findViewById()` calls with compile-time type-safe `ViewBinding` • ✓ Use `dp` for spatial dimensions, `sp` for user-scalable typography, and 48dp minimum touch targets • ✓ Optimize the 3-phase Android rendering pass (`measure` -> `layout` -> `draw`) by avoiding nested `layout_weight` traps

5.1 Android UI Widgets — `ViewBinding`, Material 3 Inputs, `dp` vs `sp` & Accessibility [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android UI Widgets — `ViewBinding`, Material 3 Inputs, `dp` vs `sp` & Accessibility in real Android systems — specifically: Work cleanly with `Mat...

5.2 LinearLayout & View Hierarchy Performance — Orientation, `layout_weight` & Measure Passes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply LinearLayout & View Hierarchy Performance — Orientation, `layout_weight` & Measure Passes in real Android systems — specifically: Align child view...

Module 5 Hands-On Capstone Project (100 Marks): Accessible Material 3 Enrollment Form with ViewBinding & Layout Pass Profiler — Create a responsive student registration screen validator using `dp`/`sp` density converters, Material `TextInputLayout` error state reducers, and a layout tree analyzer that flags exponenti...

MODULE 6 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: High-Performance Lists with `RecyclerView`, `ListAdapter` (`DiffUtil`) & `TabLayout`

Render 10,000+ dynamic items at a locked 60/120fps using the `ViewHolder` recycling pool, background Myers diffing via `ListAdapter` + `DiffUtil`, and swipeable `ViewPager2` + `TabLayoutMediator`.

Objectives: ✓ Implement `ListAdapter<T, VH>` with `DiffUtil.ItemCallback<T>` instead of full-list `notifyDataSetChanged()` • ✓ Keep `onBindViewHolder()` allocation-free and hoist click listeners to `onCreateViewHolder()` • ✓ Synchronize `TabLayout` tabs and lazy-loaded `ViewPager2` Fragments via `TabLayoutMediator`

6.1 RecyclerView & `ListAdapter` — `ViewHolder` Pooling, `LayoutManager` & `DiffUtil` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply RecyclerView & `ListAdapter` — `ViewHolder` Pooling, `LayoutManager` & `DiffUtil` in real Android systems — specifically: How `RecyclerView` infla...

6.2 TabLayout & `ViewPager2` — `FragmentStateAdapter` & `TabLayoutMediator` Synchronization [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply TabLayout & `ViewPager2` — `FragmentStateAdapter` & `TabLayoutMediator` Synchronization in real Android systems — specifically: Combine Material `...

Module 6 Hands-On Capstone Project (100 Marks): 60fps Mock Exam Leaderboard with DiffUtil Patching & Categorized ViewPager2 Tabs — Build an immutable list diffing engine modeling `DiffUtil.ItemCallback` (`areItemsTheSame` vs `areContentsTheSame`) paired with a `ViewPager2` + `TabLayout` category pager for Top Scorers, R...

MODULE 7 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Offline-First Persistence — Preferences / DataStore & SQLite / Room Databases

Store lightweight settings asynchronously using Preferences/Jetpack DataStore and relational offline datasets using parameterized SQLite (`SQLiteOpenHelper`) and Room DAO transactions.

Objectives: ✓ Contrast `SharedPreferences` (`apply()` vs blocking `commit()`) with coroutine-safe Jetpack `DataStore` • ✓ Design normalized SQLite schemas, `onUpgrade()` migrations, and SQL-injection-immune `?` placeholders • ✓ Batch bulk SQLite inserts inside `beginTransaction()` / `setTransactionSuccessful()` for 50x faster disk throughput

7.1 SharedPreferences & Jetpack DataStore — `apply()` vs `commit()`, Encrypted Storage & Flow [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply SharedPreferences & Jetpack DataStore — `apply()` vs `commit()`, Encrypted Storage & Flow in real Android systems — specifically: Persist user fla...

7.2 Android SQLite & Room Foundations — `SQLiteOpenHelper`, WAL Mode, Transactions & Migrations [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android SQLite & Room Foundations — `SQLiteOpenHelper`, WAL Mode, Transactions & Migrations in real Android systems — specifically: Manage embedde...

Module 7 Hands-On Capstone Project (100 Marks): Offline Mock Exam Attempt Vault with Transactional SQLite & Safe Schema Migrations — Build an offline persistence layer storing user theme/session flags in async key-value preferences and queued quiz answers in a versioned relational SQLite store with lossless `v1 -> v2` mig...

MODULE 8 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Defensive JSON Parsing & Asynchronous HTTP Networking (Volley / Retrofit)

Parse untrusted server JSON defensively without `JSONException` crashes, enforce HTTPS TLS cleartext bans, and manage request queues, retries, and lifecycle cancellation with Volley / Retrofit.

Objectives: ✓ Parse nested JSON payloads safely using `optString()`/`optInt()` or Kotlinx Serialization default values • ✓ Manage singleton `RequestQueue` connection pools, timeouts, and tag-based request cancellation in Volley • ✓ Understand `NetworkOnMainThreadException` and Android 9+ HTTPS-by-default `network_security_config.xml` rules

8.1 Android JSON Parsing — `JSONObject` / `JSONArray` (`opt*` vs `get*`) & Kotlin Serialization [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android JSON Parsing — `JSONObject` / `JSONArray` (`opt*` vs `get*`) & Kotlin Serialization in real Android systems — specifically: Extract nested...

8.2 Volley HTTP Networking — Singleton `RequestQueue`, `JsonObjectRequest`, RetryPolicy & HTTPS [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Volley HTTP Networking — Singleton `RequestQueue`, `JsonObjectRequest`, RetryPolicy & HTTPS in real Android systems — specifically: Fetch remote J...

Module 8 Hands-On Capstone Project (100 Marks): Resilient REST Course Sync Client with Singleton RequestQueue & Defensive JSON Mapper — Implement a fault-tolerant JSON mapper that transforms partial/nullable API payloads into clean Kotlin domain models without throwing on missing fields, paired with a tagged HTTP request que...

MODULE 9 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Background Execution — Foreground Services, WorkManager & `BroadcastReceiver`s

Execute background work within modern Android 8–15 execution limits using Foreground Services (`foregroundServiceType`), `WorkManager`, and secure `BroadcastReceiver` telephony/system event handling.

Objectives: ✓ Choose correctly between Kotlin Coroutines, `WorkManager` deferrable jobs, and user-visible Foreground Services • ✓ Comply with the 5-second `startForeground()` contract and Android 14+ mandatory `foregroundServiceType` declarations • ✓ Handle system events in `BroadcastReceiver.onReceive()` within the 10-second ANR window and Android 14 export flags

9.1 Android Services — Started vs Bound Services, Foreground Service Types & `WorkManager` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Services — Started vs Bound Services, Foreground Service Types & `WorkManager` in real Android systems — specifically: Understand critical...

9.2 BroadcastReceiver & Call State Events — Dynamic Registration, `RECEIVER_NOT_EXPORTED` & `TelephonyCallback` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply BroadcastReceiver & Call State Events — Dynamic Registration, `RECEIVER_NOT_EXPORTED` & `TelephonyCallback` in real Android systems — specifically...

Module 9 Hands-On Capstone Project (100 Marks): Audio Viva Recording Foreground Service & Call-Interruption Broadcast Coordinator — Design a state coordinator for an active spoken AI interview session that runs as a typed `microphone` Foreground Service and automatically pauses audio recording when an incoming phone call...

MODULE 10 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Notification Channels (API 26–35) & Firebase Google Sign-In (`CredentialManager`)

Deliver rich actionable notifications with `NotificationChannel`, `POST_NOTIFICATIONS` (API 33+), and `PendingIntent.FLAG_IMMUTABLE`, and authenticate users securely via Android `CredentialManager` + Firebase Auth.

Objectives: ✓ Register idempotent `NotificationChannel`s (API 26+) and request `POST_NOTIFICATIONS` gracefully on Android 13+ • ✓ Enforce `PendingIntent.FLAG_IMMUTABLE` across notifications to prevent third-party intent mutability hijacking • ✓ Implement Google Sign-In with Jetpack `CredentialManager` (`GetGoogleIdOption`) + `FirebaseAuth.signInWithCredential` and server-side ID token verification

10.1 Android Notifications — `NotificationChannel`, `POST_NOTIFICATIONS` & `FLAG_IMMUTABLE` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Android Notifications — `NotificationChannel`, `POST_NOTIFICATIONS` & `FLAG_IMMUTABLE` in real Android systems — specifically: Build rich notifica...

10.2 Firebase Authentication & Google Sign-In — `CredentialManager`, SHA-1/SHA-256 & ID Tokens [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Firebase Authentication & Google Sign-In — `CredentialManager`, SHA-1/SHA-256 & ID Tokens in real Android systems — specifically: Authenticate And...

Module 10 Hands-On Capstone Project (100 Marks): Mock Exam Reminder Notification Engine & Firebase Google Auth Token Verifier — Build a notification dispatcher enforcing API 26+ channels, API 33+ runtime permission checks, and `FLAG_IMMUTABLE` PendingIntents, paired with a Firebase Google ID-token session state manag...

MODULE 11 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Offline-First Sync: Room Migrations, Paging 3 & WorkManager Queues

Production-grade Android Training engineering module covering Offline-First Sync: Room Migrations, Paging 3 & WorkManager Queues with hands-on labs and capstone build.

Objectives: ✓ Single-Source-of-Truth Repository (Room Flow + RemoteMediator) — Purpose: Keep screens usable inside elevators and rural 2G zones with seamless background sync. • ✓ Paging 3 Infinite Lists & Room Auto/Manual Schema Migrations — Purpose: Scroll 10,000+ catalog items smoothly and upgrade local SQLite schemas without data loss. • ✓ Guaranteed Background Sync with WorkManager Constraints & Backoff — Purpose: Queue pending orders/photos reliably until battery and Wi-Fi/network constraints pass. • ✓ DataStore Proto/Preferences & Encrypted Token Persistence — Purpose: Replace blocking SharedPreferences with async, corruption-safe Jetpack DataStore.

11.1 Single-Source-of-Truth Repository (Room Flow + RemoteMediator) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Single-Source-of-Truth Repository (Room Flow + RemoteMediator) in real Android systems — specifically: Keep screens usable inside elevators and ru...

11.2 Paging 3 Infinite Lists & Room Auto/Manual Schema Migrations [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Paging 3 Infinite Lists & Room Auto/Manual Schema Migrations in real Android systems — specifically: Scroll 10,000+ catalog items smoothly and upg...

11.3 Guaranteed Background Sync with WorkManager Constraints & Backoff [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Guaranteed Background Sync with WorkManager Constraints & Backoff in real Android systems — specifically: Queue pending orders/photos reliably unt...

11.4 DataStore Proto/Preferences & Encrypted Token Persistence [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply DataStore Proto/Preferences & Encrypted Token Persistence in real Android systems — specifically: Replace blocking SharedPreferences with async, c...

Module 11 Hands-On Capstone Project (100 Marks): Offline-Resilient Field Sales & Inventory Android App — Design, implement, test, and document 'Offline-Resilient Field Sales & Inventory Android App' applying every production concept from Module 11 (Offline-First Sync: Room Migrations, Paging 3...

MODULE 12 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Clean Architecture Modularization, Hilt DI & Kotlin Coroutines/Flow

Production-grade Android Training engineering module covering Clean Architecture Modularization, Hilt DI & Kotlin Coroutines/Flow with hands-on labs and capstone build.

Objectives: ✓ Feature/Core Multi-Module Gradle Setup (:app, :core:network, :feature:pay) — Purpose: Slash incremental Gradle build times and enforce strict layer boundaries. • ✓ Dagger Hilt Scopes (@Singleton, @ViewModelScoped) & TestReplacements — Purpose: Inject repositories and clocks cleanly across ViewModels, Workers, and UI tests. • ✓ StateFlow, SharedFlow, repeatOnLifecycle & Unidirectional MVI State — Purpose: Eliminate lifecycle crashes and duplicate snackbar/navigation events on screen rotation. • ✓ Retrofit OkHttp Interceptors, Certificate Pinning & Token Refresh Mutex — Purpose: Block MITM attacks and refresh expired JWTs once across parallel mobile calls.

12.1 Feature/Core Multi-Module Gradle Setup (:app, :core:network, :feature:pay) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Feature/Core Multi-Module Gradle Setup (:app, :core:network, :feature:pay) in real Android systems — specifically: Slash incremental Gradle build...

12.2 Dagger Hilt Scopes (@Singleton, @ViewModelScoped) & TestReplacements [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Dagger Hilt Scopes (@Singleton, @ViewModelScoped) & TestReplacements in real Android systems — specifically: Inject repositories and clocks cleanl...

12.3 StateFlow, SharedFlow, repeatOnLifecycle & Unidirectional MVI State [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply StateFlow, SharedFlow, repeatOnLifecycle & Unidirectional MVI State in real Android systems — specifically: Eliminate lifecycle crashes and duplic...

12.4 Retrofit OkHttp Interceptors, Certificate Pinning & Token Refresh Mutex [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Retrofit OkHttp Interceptors, Certificate Pinning & Token Refresh Mutex in real Android systems — specifically: Block MITM attacks and refresh exp...

Module 12 Hands-On Capstone Project (100 Marks): Multi-Module Clean Architecture FinTech Android Client — Design, implement, test, and document 'Multi-Module Clean Architecture FinTech Android Client' applying every production concept from Module 12 (Clean Architecture Modularization, Hilt DI &...

MODULE 13 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Jetpack Compose Performance, Baseline Profiles, Biometrics & Push

Production-grade Android Training engineering module covering Jetpack Compose Performance, Baseline Profiles, Biometrics & Push with hands-on labs and capstone build.

Objectives: ✓ Compose Stability, @Immutable Models, derivedStateOf & Layout Inspector — Purpose: Eliminate dropped frames and jank caused by unnecessary recompositions. • ✓ Baseline Profiles & R8/ProGuard Code Shrinking & Obfuscation — Purpose: Boost cold-start speed by 30% and shrink APK/AAB download size dramatically. • ✓ Android Keystore, BiometricPrompt Auth & Play Integrity Checks — Purpose: Protect UPI/wallet confirmations with hardware-backed fingerprint/face verification. • ✓ Firebase Cloud Messaging (FCM) Channels & Compose Deep-Link Navigation — Purpose: Land users on the exact order or chat screen tapped from a lock-screen push alert.

13.1 Compose Stability, @Immutable Models, derivedStateOf & Layout Inspector [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Compose Stability, @Immutable Models, derivedStateOf & Layout Inspector in real Android systems — specifically: Eliminate dropped frames and jank...

13.2 Baseline Profiles & R8/ProGuard Code Shrinking & Obfuscation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Baseline Profiles & R8/ProGuard Code Shrinking & Obfuscation in real Android systems — specifically: Boost cold-start speed by 30% and shrink APK/...

13.3 Android Keystore, BiometricPrompt Auth & Play Integrity Checks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Android Keystore, BiometricPrompt Auth & Play Integrity Checks in real Android systems — specifically: Protect UPI/wallet confirmations with hardw...

13.4 Firebase Cloud Messaging (FCM) Channels & Compose Deep-Link Navigation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Firebase Cloud Messaging (FCM) Channels & Compose Deep-Link Navigation in real Android systems — specifically: Land users on the exact order or ch...

Module 13 Hands-On Capstone Project (100 Marks): 60fps Biometric-Protected Wallet & FCM Deep-Link Experience — Design, implement, test, and document '60fps Biometric-Protected Wallet & FCM Deep-Link Experience' applying every production concept from Module 13 (Jetpack Compose Performance, Baseline Pr...

MODULE 14 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Compose UI Testing, Turbine Flow Tests, Crashlytics & Play Store CI/CD

Production-grade Android Training engineering module covering Compose UI Testing, Turbine Flow Tests, Crashlytics & Play Store CI/CD with hands-on labs and capstone build.

Objectives: ✓ Unit Testing ViewModels & Flows with kotlinx-coroutines-test & Turbine — Purpose: Assert exact Loading -> Success/Error state emissions in milliseconds on JVM. • ✓ ComposeTestRule Semantic UI Tests & Robolectric/Espresso Suites — Purpose: Verify critical login and payment flows automatically on every pull request. • ✓ Firebase Crashlytics Mapping Upload, ANR Diagnostics & StrictMode — Purpose: De-obfuscate production stack traces and eliminate main-thread disk/network ANRs. • ✓ Signed Android App Bundles (.aab), Fastlane/GitHub Actions & Staged Rollout — Purpose: Ship 5% -> 25% -> 100% Play Store releases safely with instant halt controls.

14.1 Unit Testing ViewModels & Flows with kotlinx-coroutines-test & Turbine [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Unit Testing ViewModels & Flows with kotlinx-coroutines-test & Turbine in real Android systems — specifically: Assert exact Loading -> Success/Err...

14.2 ComposeTestRule Semantic UI Tests & Robolectric/Espresso Suites [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply ComposeTestRule Semantic UI Tests & Robolectric/Espresso Suites in real Android systems — specifically: Verify critical login and payment flows au...

14.3 Firebase Crashlytics Mapping Upload, ANR Diagnostics & StrictMode [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Firebase Crashlytics Mapping Upload, ANR Diagnostics & StrictMode in real Android systems — specifically: De-obfuscate production stack traces and...

14.4 Signed Android App Bundles (.aab), Fastlane/GitHub Actions & Staged Rollout [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Signed Android App Bundles (.aab), Fastlane/GitHub Actions & Staged Rollout in real Android systems — specifically: Ship 5% -> 25% -> 100% Play St...

Module 14 Hands-On Capstone Project (100 Marks): Automated Play Store AAB Release Pipeline with Staged Rollouts — Design, implement, test, and document 'Automated Play Store AAB Release Pipeline with Staged Rollouts' applying every production concept from Module 14 (Compose UI Testing, Turbine Flow Test...

MODULE 15 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Offline-First Android Architecture, Room Sync & Jetpack Compose Performance

Self-explanatory list-and-bullet production mastery module covering Offline-First Android Architecture, Room Sync & Jetpack Compose Performance in Android Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Unidirectional Data Flow (UDF), StateFlow UI State & Single-Source-of-Truth Room Sync — Purpose: Keep Android screens 100% usable on flaky mobile networks with automatic background conflict resolution. • ✓ Jetpack Compose Recomposition Optimization: @Immutable, derivedStateOf & Lazy Keys — Purpose: Eliminate dropped frames and jank on long scrolling feeds verified via Layout Inspector. • ✓ Encrypted DataStore/Keystore Credentials, Biometric Prompt & Deep-Link Navigation — Purpose: Protect user tokens on device hardware and handle notification/web deep links securely. • ✓ Build, verify, and document 'Offline-First Field Service Android App with Room + WorkManager + Compose Stability' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

15.1 Unidirectional Data Flow (UDF), StateFlow UI State & Single-Source-of-Truth Room Sync [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Unidirectional Data Flow (UDF), StateFlow UI State & Single-Source-of-Truth Room Sync in real Android systems — specifically: Keep Android screens...

15.2 Jetpack Compose Recomposition Optimization: @Immutable, derivedStateOf & Lazy Keys [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Jetpack Compose Recomposition Optimization: @Immutable, derivedStateOf & Lazy Keys in real Android systems — specifically: Eliminate dropped frame...

15.3 Encrypted DataStore/Keystore Credentials, Biometric Prompt & Deep-Link Navigation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Encrypted DataStore/Keystore Credentials, Biometric Prompt & Deep-Link Navigation in real Android systems — specifically: Protect user tokens on d...

Module 15 Hands-On Capstone Project (100 Marks): Offline-First Field Service Android App with Room + WorkManager + Compose Stability — Architect, implement, test, and document 'Offline-First Field Service Android App with Room + WorkManager + Compose Stability' applying every bullet-wise engineering concept from Module 15 (...

MODULE 16 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Baseline Profiles, Crash/ANR Diagnostics, Automated Testing & Play Release

Self-explanatory list-and-bullet production mastery module covering Baseline Profiles, Crash/ANR Diagnostics, Automated Testing & Play Release in Android Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Testing Kotlin Coroutines/Flows (Turbine) & Compose UI Semantics Tests — Purpose: Verify ViewModel state transitions and Compose screen accessibility deterministically in CI. • ✓ Fixing Memory Leaks (LeakCanary), ANRs, StrictMode & Baseline Profiles — Purpose: Accelerate cold app startup by 30%+ and eliminate main-thread disk/network stalls. • ✓ R8/ProGuard Shrinking, Play Integrity, Staged Rollouts & Android Interviews — Purpose: Ship compact, obfuscated Android App Bundles and ace Kotlin/Android architecture interviews. • ✓ Build, verify, and document 'Macrobenchmark Baseline Profile + Turbine/Compose UI Test Suite & Signed AAB Pipeline' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

16.1 Testing Kotlin Coroutines/Flows (Turbine) & Compose UI Semantics Tests [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Testing Kotlin Coroutines/Flows (Turbine) & Compose UI Semantics Tests in real Android systems — specifically: Verify ViewModel state transitions...

16.2 Fixing Memory Leaks (LeakCanary), ANRs, StrictMode & Baseline Profiles [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Fixing Memory Leaks (LeakCanary), ANRs, StrictMode & Baseline Profiles in real Android systems — specifically: Accelerate cold app startup by 30%+...

16.3 R8/ProGuard Shrinking, Play Integrity, Staged Rollouts & Android Interviews [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply R8/ProGuard Shrinking, Play Integrity, Staged Rollouts & Android Interviews in real Android systems — specifically: Ship compact, obfuscated Andro...

Module 16 Hands-On Capstone Project (100 Marks): Macrobenchmark Baseline Profile + Turbine/Compose UI Test Suite & Signed AAB Pipeline — Architect, implement, test, and document 'Macrobenchmark Baseline Profile + Turbine/Compose UI Test Suite & Signed AAB Pipeline' applying every bullet-wise engineering concept from Module 16...

3. LIVE CLIENT PRODUCTION PROJECT, GLOBAL CERTIFICATIONS & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: FieldTrack Offline-First Field Service & Barcode Inspection Android App • **Confirmed Role:** Native Android Kotlin & Jetpack Compose Intern

Domain & Production Infra: Logistics & Industrial Asset Inspection Client (Kotlin + Jetpack Compose + MVVM Clean Architecture + Room DB + Retrofit + WorkManager)

- Built responsive Material 3 screens in Jetpack Compose using unidirectional StateFlow and ViewModel architecture.
- Implemented offline-first caching with Room SQLite, Kotlin Coroutines, and background WorkManager sync.
- Integrated secure Retrofit + OkHttp REST clients with token refresh interceptors and dependency injection via Hilt.
- Delivered signed, ProGuard/R8-optimized release APKs/AABs verified against memory leaks and ANRs.

Aligned Global Vendor Certifications: **GOOGLE-AAD** (Android Kotlin & Jetpack Compose Developer Credential) • **1Z0-811** (Oracle Certified Foundations Associate, Java)

Target Job Roles: Native Android Developer (Kotlin), Mobile Software Engineer — Android, Jetpack Compose App Engineer, Associate Android SDE-1

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹5.0 LPA – ₹9.5 LPA **Mid-Level Android Product Engineer (1–3 Yrs):** ₹10.5 LPA – ₹21.0 LPA **Senior Mobile Architect / Staff Android Engineer (3–6+ Yrs):** ₹22.0 LPA – ₹42.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/android-training

Try 2 Free Topic-Wise 50-MCQ Mock Tests & Spoken Gemini AI Interviews daily on every topic above before enrolling.