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

RNAT • CROSS-PLATFORM MOBILE ENGINEERING (IOS & ANDROID) **BEGINNER TO SENIOR REACT NATIVE & EXPO ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

React Native Training

Master production React Native & Expo from Hermes bytecode, JSI, Fabric & TurboModules to View/Text primitives, Props/Hooks, StyleSheet, Yoga Flexbox, TextInput, Pressable, FlatList/SectionList virtualization, Image caching, Native-Driver Animations, WebView security & GPS Geolocation — with 10×50 MCQ Mock Tests per topic and Spoken Voice + Gemini AI Interviews at ₹4,999.

Duration: 15–17 Weeks • Live + Hands-On Projects (207 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) • 17–19 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) • 19–21 Weeks Weekend Track)

Core Technologies & Engineering Stack: React Native 0.76+ (New Architecture) • Hermes Engine, JSI, Fabric & TurboModules • Expo SDK 52, Expo Router & EAS Build • Yoga C++ Layout Engine & StyleSheet • FlatList, SectionList & Pressable • Animated (useNativeDriver) & Reanimated • WebView Hardening, Alerts & Geolocation

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Explain React Native's New Architecture: how C++ JSI, synchronous Fabric Shadow Trees, TurboModules, and Hermes AOT bytecode replace the legacy async JSON bridge
2. Build responsive, notch-safe iOS & Android interfaces using View, Text, SafeAreaView, StyleSheet.create, useWindowDimensions, and Yoga Flexbox
3. Optimize 10,000-row virtualized FlatList/SectionList feeds, 60/120fps native-driver animations, hardened WebViews, and permission-safe GPS tracking
4. Clear 200 Timed Mock Exams (10,000 React Native MCQs) and 200 Spoken Voice + Gemini AI Mobile Engineering Interviews
5. Strict TypeScript React Native components with React Navigation native stacks, deep linking, and auth flows.
6. High-performance lists (`FlashList` / tuned `FlatList`), Reanimated UI-thread animations, and keyboard-safe forms.
7. Server-state caching & offline mutation queues via TanStack Query paired with lightweight Zustand client stores.
8. Native device integrations: biometrics, camera/files, push notifications, and encrypted Keychain/Keystore tokens.

Eligibility & Prerequisites: Working knowledge of modern ES6+ JavaScript and basic React fundamentals (JSX, components, hooks) • Any Windows, macOS, or Linux laptop with Node.js 20+ LTS, VS Code, and Expo Go / Android Studio / Xcode Simulator • No prior React Native 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: React Native New Architecture (Hermes, JSI, Fabric, TurboModules) & Expo Setup

Understand how React Native renders genuine native iOS (`UIView`) and Android (`android.view.View`) components via C++ JSI, Fabric, TurboModules, and Hermes AOT bytecode, and set up Expo Development Builds.

Objectives: ✓ Contrast the legacy asynchronous JSON Bridge with synchronous C++ JSI, Fabric Shadow Trees, and lazy TurboModules • ✓ Explain how Hermes precompiles JavaScript into compact bytecode (`hbc`) for instant cold-start TTI • ✓ Choose correctly between Expo Go sandboxes, Expo Prebuild (CNG / Development Builds), and bare `ios/` + `android/` workflows

1.1 React Native Overview — Native OEM Views (`UIView` / `android.view.View`), Hermes, JSI & Fabric [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native Overview — Native OEM Views (`UIView` / `android.view.View`), Hermes, JSI & Fabric in real React Native systems — specifically: Why R...

1.2 Environment Setup — Expo SDK (`create-expo-app`), Development Builds (CNG), Metro & Fast Refresh [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Environment Setup — Expo SDK (`create-expo-app`), Development Builds (CNG), Metro & Fast Refresh in real React Native systems — specifically: Boot...

Module 1 Hands-On Capstone Project (100 Marks): New Architecture Runtime Inspector & Expo Config Validator — Create a React Native diagnostics helper inspecting global JSI/Fabric/Hermes runtime flags (`global.HermesInternal`, `global.nativeFabricUIManager`) and validating `app.json` bundle identifi...

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

Module 2: Core Native Primitives — Hello World Entry Point, `<View>`, `<Text>` & Safe Areas

Build your first cross-platform screen without HTML tags (`<div>`/`<p>` do not exist!), master the strict `<Text>` string rule, notch-safe `SafeAreaProvider` / `useSafeAreaInsets`, and `StatusBar`.

Objectives: ✓ Register root components via Expo Router or `AppRegistry.registerComponent` cleanly • ✓ Prevent fatal `Text strings must be rendered within a <Text> component` crashes including the `{count && <Badge />}` `0`-falsy bug • ✓ Configure `<View>` accessibility (`accessible`, `accessibilityRole`), `pointerEvents`, and `react-native-safe-area-context` insets

2.1 First React Native App — `<Text>` Containment Rule, Falsy `0` Crash & `SafeAreaProvider` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply First React Native App — `<Text>` Containment Rule, Falsy `0` Crash & `SafeAreaProvider` in real React Native systems — specifically: Understand w...

2.2 React Native `<View>` — Native Mapping, View Flattening (`collapsable`), `pointerEvents` & `onLayout` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<View>` — Native Mapping, View Flattening (`collapsable`), `pointerEvents` & `onLayout` in real React Native systems — specifically:...

Module 2 Hands-On Capstone Project (100 Marks): Notch-Aware Course Welcome Card with Zero Falsy-Zero `<View>` Crashes — Build a reusable `<CourseHeroCard>` using `<View>`, nested `<Text>` style inheritance, safe boolean rendering (`count > 0 ? ... : null`), and dynamic top/bottom safe-area inset padding.

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

Module 3: Component Architecture — Unidirectional Props, Memoization & React Hooks State

Compose reusable mobile UI components with read-only `props`, default destructuring, `React.memo`, and immutable state updates via `useState`, functional updaters `setCount(c => c + 1)`, and `useReducer`.

Objectives: ✓ Design clean, composable mobile card/header components with default parameter values and `children` composition • ✓ Prevent stale-closure bugs in rapid tap handlers and timers using functional state updaters `setState(prev => ...)` • ✓ Avoid mutating arrays/objects in place so `Object.is()` reconciliation triggers native Fabric updates reliably

3.1 React Native Props — Read-Only Data Flow, ES6 Default Params, `children` & `React.memo` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native Props — Read-Only Data Flow, ES6 Default Params, `children` & `React.memo` in real React Native systems — specifically: Configure mob...

3.2 React Native State — `useState`, Functional Updaters, Immutable Arrays & `Object.is` Batching [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native State — `useState`, Functional Updaters, Immutable Arrays & `Object.is` Batching in real React Native systems — specifically: Drive r...

Module 3 Hands-On Capstone Project (100 Marks): Memoized Quiz Question Card & Stale-Closure-Safe Timer Hook Reducer — Build a memoized `<QuestionOptionCard>` paired with a pure quiz state reducer (`SELECT_OPTION`, `TICK_SECOND`, `SUBMIT_QUIZ`) that guarantees immutable transitions and zero stale-closure sco...

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

Module 4: `StyleSheet` Architecture, Cross-Platform Shadows & Responsive Screen Dimensions

Write clean, static `StyleSheet.create()` definitions, unify iOS shadow props with Android `elevation`, and build foldable/rotation-responsive layouts using `useWindowDimensions()` and `PixelRatio`.

Objectives: ✓ Organize reusable styles with `StyleSheet.create()`, array composition `[styles.card, isSelected && styles.active]`, and `StyleSheet.hairlineWidth` • ✓ Handle platform-divergent shadow properties (`shadowColor`/`shadowOffset`/`shadowOpacity` on iOS vs `elevation` on Android) • ✓ Use the reactive `useWindowDimensions()` hook instead of static module-top `Dimensions.get("window")` snapshots

4.1 React Native StyleSheet — `StyleSheet.create`, Array Merging, `hairlineWidth` & Platform Shadows [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native StyleSheet — `StyleSheet.create`, Array Merging, `hairlineWidth` & Platform Shadows in real React Native systems — specifically: Styl...

4.2 Height & Width — DP Units, Flex Proportions, Percentage Sizing & `useWindowDimensions()` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Height & Width — DP Units, Flex Proportions, Percentage Sizing & `useWindowDimensions()` in real React Native systems — specifically: Choose betwe...

Module 4 Hands-On Capstone Project (100 Marks): Cross-Platform Shadow Elevation Token System & Foldable Breakpoint Hook — Create a design-token helper generating unified iOS + Android elevation style objects for levels `1..5`, paired with a `useWindowDimensions`-compatible breakpoint classifier handling portrai...

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

Module 5: Yoga C++ Flexbox Engine & Controlled Mobile `<TextInput>` Forms

Master Meta's C++ **Yoga** Flexbox defaults (`flexDirection: "column"`, `gap`, `flex: 1` shorthand) and build keyboard-aware `<TextInput>` forms with `KeyboardAvoidingView` and focus ref chaining.

Objectives: ✓ Contrast React Native Yoga defaults (`flexDirection: "column"`, `alignContent: "flex-start"`, `flexShrink: 0`) with browser CSS Flexbox • ✓ Use `gap`, `rowGap`, `columnGap`, and `flexShrink: 1` to prevent long text inside row cards from pushing icons off-screen • ✓ Build mobile-native `<TextInput>` flows with `keyboardType`, `autoCapitalize="none"`, `secureTextEntry`, `returnKeyType`, and `KeyboardAvoidingView`

5.1 Layout & Yoga Flexbox — `flexDirection: "column"` Default, `gap` & Text `flexShrink: 1` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Layout & Yoga Flexbox — `flexDirection: "column"` Default, `gap` & Text `flexShrink: 1` in real React Native systems — specifically: Master Yoga C...

5.2 React Native `<TextInput>` — `onChangeText`, IME `keyboardType`, Focus Refs & `KeyboardAvoidingView` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<TextInput>` — `onChangeText`, IME `keyboardType`, Focus Refs & `KeyboardAvoidingView` in real React Native systems — specifically:...

Module 5 Hands-On Capstone Project (100 Marks): Keyboard-Safe OTP & Login Screen State Controller with Yoga Row Text-Wrap Guard — Create a controlled login form validation hook handling email normalization (`trim` + lowercase), password masking toggles, and platform-specific `KeyboardAvoidingView` behavior (`"padding"`...

MODULE 6 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Touch Interactions — Native `<Button>` Limits vs Modern `<Pressable>` & `hitSlop`

Understand why `<Button>` is only for quick prototypes and why production design systems standardize on `<Pressable>` (replacing legacy `TouchableOpacity` / `TouchableHighlight`) with `android_ripple` and `hitSlop`.

Objectives: ✓ Contrast OS-styled `<Button>` limitations (no `style` prop, inconsistent iOS text vs Android filled look) with custom `<Pressable>` components • ✓ Implement tactile feedback across both platforms using `<Pressable style={({ pressed }) => ...} android_ripple={{ ... }}>` • ✓ Expand small icon tap zones without shifting visual layout using `hitSlop={12}` and `pressRetentionOffset`

6.1 React Native `<Button>` — Native `UIBarButton` / Material Button Rendering & Customization Limits [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<Button>` — Native `UIBarButton` / Material Button Rendering & Customization Limits in real React Native systems — specifically: Use...

6.2 Touchables & `<Pressable>` — `TouchableOpacity` vs Future-Proof `<Pressable>`, `android_ripple` & `hitSlop` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Touchables & `<Pressable>` — `TouchableOpacity` vs Future-Proof `<Pressable>`, `android_ripple` & `hitSlop` in real React Native systems — specifi...

Module 6 Hands-On Capstone Project (100 Marks): Design-System `<AcademyButton>` Built on `<Pressable>` with Ripple, Loading & `hitSlop` — Engineer a reusable cross-platform button state helper supporting `primary`, `outline`, and `ghost` variants, pressed-opacity scaling on iOS, native Material `android_ripple` on Android, `di...

MODULE 7 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Scroll Containers (`<ScrollView>`) vs High-Speed Virtualized `<FlatList>` Feeds

Know exactly when to use eager `<ScrollView>` (`contentContainerStyle`, `RefreshControl`) vs windowed `<FlatList>` (`keyExtractor`, `getItemLayout`, `windowSize`) to scroll 10,000+ rows at 60/120fps.

Objectives: ✓ Distinguish `<ScrollView>` `style` (outer viewport frame) from `contentContainerStyle` (inner scrollable children padding/alignment) • ✓ Prevent the #1 React Native performance bug: never nest a vertical `<FlatList>` inside a same-axis `<ScrollView>` • ✓ Tune `<FlatList>` with stable `keyExtractor`,

memoized `renderItem`, and O(1) `getItemLayout` jump offsets

7.1 React Native `<ScrollView>` — `contentContainerStyle`, `keyboardShouldPersistTaps` & Eager Mounting [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<ScrollView>` — `contentContainerStyle`, `keyboardShouldPersistTaps` & Eager Mounting in real React Native systems — specifically: U...

7.2 React Native `<FlatList>` — `VirtualizedList` Windowing, `keyExtractor` & `getItemLayout` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<FlatList>` — `VirtualizedList` Windowing, `keyExtractor` & `getItemLayout` in real React Native systems — specifically: Render mass...

Module 7 Hands-On Capstone Project (100 Marks): O(1) `getItemLayout` Virtualized Question Navigator & Pull-to-Refresh Feed — Implement `<FlatList>` optimization helpers including deterministic `getItemLayout` offset calculation (accounting for row height + `ItemSeparatorComponent` height) and infinite-scroll curso...

MODULE 8 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Grouped `<SectionList>` Headers & High-Performance `<Image>` Caching

Render sticky-header grouped curricula/contacts using `<SectionList>` (`sections: [{ title, data }]`), and load local/remote images cleanly without layout shifts or memory spikes.

Objectives: ✓ Format grouped datasets into `<SectionList>`'s mandatory `{ title, data: [...] }` schema with `stickySectionHeadersEnabled` • ✓ Understand why remote `<Image source={{ uri }}>` requires explicit `width` + `height` (or `aspectRatio`), unlike static `require("./logo.png")` • ✓ Optimize image decoding, blurhash placeholders, and disk/memory caching (`expo-image` / `Image.prefetch`)

8.1 React Native `<SectionList>` — `{ data: [...] }` Schema, `renderSectionHeader` & Sticky Headers [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<SectionList>` — `{ data: [...] }` Schema, `renderSectionHeader` & Sticky Headers in real React Native systems — specifically: Displ...

8.2 React Native `<Image>` — Static `require()` vs Remote `{ uri }`, `resizeMode` & `expo-image` Caching [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<Image>` — Static `require()` vs Remote `{ uri }`, `resizeMode` & `expo-image` Caching in real React Native systems — specifically:...

Module 8 Hands-On Capstone Project (100 Marks): Grouped Curriculum `<SectionList>` Transformer & Remote `<Image>` Dimension Guard — Build a flat-to-grouped curriculum transformer outputting valid `<SectionList>` `sections` arrays with module completion counts, paired with a remote `<Image>` source validator enforcing HTT...

MODULE 9 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: 60/120fps Native-Driver Animations (`Animated`) & Hardened `<WebView>` Integration

Drive silky-smooth 60/120fps animations off the JS thread using `Animated.timing` / `Animated.spring` with `useNativeDriver: true` (and `LayoutAnimation` / Reanimated), and embed secure two-way `<WebView>` flows.

Objectives: ✓ Keep `Animated.Value` stable across renders using `useAnimatedValue(0)` or `useRef(new Animated.Value(0)).current` • ✓ Know exactly which style properties support `useNativeDriver: true` (`opacity` and `transform`) vs layout properties (`width`, `height`, `padding`) • ✓ Harden `react-native-webview` using `originWhitelist`, `onShouldStartLoadWithRequest`, and structured JSON `postMessage` / `onMessage` bridges

9.1 React Native Animations — `Animated.Value`, `useNativeDriver: true`, `interpolate` & Reanimated [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native Animations — `Animated.Value`, `useNativeDriver: true`, `interpolate` & Reanimated in real React Native systems — specifically: Anima...

9.2 React Native `<WebView>` — `react-native-webview`, `originWhitelist` & `postMessage` Bridge Security [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `<WebView>` — `react-native-webview`, `originWhitelist` & `postMessage` Bridge Security in real React Native systems — specifically:...

Module 9 Hands-On Capstone Project (100 Marks): Native-Driver Progress Spring Interpolator & Secure `<WebView>` Origin Guard — Build an `Animated` interpolation mapper converting `[0, 1]` progress values into `scale` and `translateY` transforms compatible with `useNativeDriver: true`, plus a `<WebView>` navigation g...

MODULE 10 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Native OS Dialogs (`Alert.alert`) & Permission-Safe GPS Geolocation

Trigger native `UIAlertController` / `AlertDialog` confirmation prompts cleanly across iOS & Android (`Alert.alert` 3-button Android rule!), and acquire live GPS coordinates with foreground/background permission flows and cleanup.

Objectives: ✓ Build destructive action confirmations with `Alert.alert`, understanding Android's max-3-button limit and iOS-only `Alert.prompt` • ✓ Request iOS (`NSLocationWhenInUseUsageDescription`) and Android (`ACCESS_FINE_LOCATION`) runtime permissions gracefully before reading GPS • ✓ Compute Haversine geofence distances and clean up `watchPosition` subscriptions inside `useEffect` return functions to prevent battery drain

10.1 React Native `Alert` — Imperative `Alert.alert()`, Android 3-Button Limit & Destructive Styles [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native `Alert` — Imperative `Alert.alert()`, Android 3-Button Limit & Destructive Styles in real React Native systems — specifically: Invoke...

10.2 React Native Geolocation — `expo-location` / Geolocation Permissions, Timeouts & `watchPosition` Cleanup [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply React Native Geolocation — `expo-location` / Geolocation Permissions, Timeouts & `watchPosition` Cleanup in real React Native systems — specifical...

Module 10 Hands-On Capstone Project (100 Marks): Cross-Platform Exam Exit Confirmation & Geofenced Attendance Verifier — Build a cross-platform `Alert.alert` builder enforcing Android's 3-button ceiling (`[Neutral, Cancel, Destructive]`) and a Haversine GPS distance verifier with automatic `watchPosition` subs...

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

React Native New Architecture (Fabric, JSI, TurboModules) & Navigation

Production-grade React Native Training engineering module covering React Native New Architecture (Fabric, JSI, TurboModules) & Navigation with hands-on labs and capstone build.

Objectives: ✓ Bridgeless JSI Synchronous Calls & Fabric Concurrent Renderer — Purpose: Understand why modern React Native achieves native-grade startup and gesture latency. • ✓ Typed React Navigation Stacks, Tabs, Modals & Universal Deep Links — Purpose: Route WhatsApp/SMS links directly to authenticated product or order screens. • ✓ TanStack Query + Zustand + MMKV Synchronous Persistence — Purpose: Read cached mobile state 30x faster than legacy AsyncStorage on app launch. • ✓ KeyboardAvoiding Patterns, SafeAreaInsets & Responsive Breakpoints — Purpose: Prevent notches and on-screen keyboards from covering input fields across devices.

11.1 Bridgeless JSI Synchronous Calls & Fabric Concurrent Renderer [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Bridgeless JSI Synchronous Calls & Fabric Concurrent Renderer in real React Native systems — specifically: Understand why modern React Native achi...

11.2 Typed React Navigation Stacks, Tabs, Modals & Universal Deep Links [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Typed React Navigation Stacks, Tabs, Modals & Universal Deep Links in real React Native systems — specifically: Route WhatsApp/SMS links directly...

11.3 TanStack Query + Zustand + MMKV Synchronous Persistence [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply TanStack Query + Zustand + MMKV Synchronous Persistence in real React Native systems — specifically: Read cached mobile state 30x faster than lega...

11.4 KeyboardAvoiding Patterns, SafeAreaInsets & Responsive Breakpoints [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply KeyboardAvoiding Patterns, SafeAreaInsets & Responsive Breakpoints in real React Native systems — specifically: Prevent notches and on-screen keyb...

Module 11 Hands-On Capstone Project (100 Marks): Type-Safe Deep-Linked Mobile Commerce Shell — Design, implement, test, and document 'Type-Safe Deep-Linked Mobile Commerce Shell' applying every production concept from Module 11 (React Native New Architecture (Fabric, JSI, TurboModules...

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

60fps Lists (FlashList), Reanimated UI-Thread Gestures & Offline Sync

Production-grade React Native Training engineering module covering 60fps Lists (FlashList), Reanimated UI-Thread Gestures & Offline Sync with hands-on labs and capstone build.

Objectives: ✓ Cell Recycling with FlashList vs FlatList Memory Spikes — Purpose: Eliminate blank white cells while flick-scrolling thousands of catalog cards. • ✓ Reanimated Worklets & Gesture Handler on the Native UI Thread — Purpose: Run buttery 60/120fps swipe-to-pay and bottom-sheet animations even when JS is busy. • ✓ NetInfo Connectivity Queues & SQLite/WatermelonDB Offline Storage — Purpose: Queue field updates offline and replay idempotently once 4G/Wi-Fi returns. • ✓ Expo SecureStore, Biometric Auth & SSL Pinning — Purpose: Harden mobile sessions and sensitive healthcare/fintech payloads on device.

12.1 Cell Recycling with FlashList vs FlatList Memory Spikes [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Cell Recycling with FlashList vs FlatList Memory Spikes in real React Native systems — specifically: Eliminate blank white cells while flick-scrol...

12.2 Reanimated Worklets & Gesture Handler on the Native UI Thread [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Reanimated Worklets & Gesture Handler on the Native UI Thread in real React Native systems — specifically: Run buttery 60/120fps swipe-to-pay and...

12.3 NetInfo Connectivity Queues & SQLite/WatermelonDB Offline Storage [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply NetInfo Connectivity Queues & SQLite/WatermelonDB Offline Storage in real React Native systems — specifically: Queue field updates offline and rep...

12.4 Expo SecureStore, Biometric Auth & SSL Pinning [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Expo SecureStore, Biometric Auth & SSL Pinning in real React Native systems — specifically: Harden mobile sessions and sensitive healthcare/fintec...

Module 12 Hands-On Capstone Project (100 Marks): Fluid Gesture-Driven Feed & Offline Order Sync App — Design, implement, test, and document 'Fluid Gesture-Driven Feed & Offline Order Sync App' applying every production concept from Module 12 (60fps Lists (FlashList), Reanimated UI-Thread Ges...

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

Native Permissions, Camera/Media Uploads, Push & Native Module Config

Production-grade React Native Training engineering module covering Native Permissions, Camera/Media Uploads, Push & Native Module Config with hands-on labs and capstone build.

Objectives: ✓ Graceful Runtime Permission Flows (Camera, Location, Notifications) — Purpose: Explain permission value upfront to maximize user opt-in and pass App Store review. • ✓ Chunked/Presigned Multipart Media Uploads & Image Compression — Purpose: Upload compressed receipts and photos reliably over flaky mobile networks. • ✓ FCM / APNs Push Notification Channels, Badges & Background Tap Routing — Purpose: Re-engage users reliably across foreground, background, and cold-killed app states. • ✓ Expo Config Plugins & Custom Native Android/iOS Manifest Tuning — Purpose: Customize native Gradle/Podfile capabilities cleanly without losing upgrade velocity.

13.1 Graceful Runtime Permission Flows (Camera, Location, Notifications) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Graceful Runtime Permission Flows (Camera, Location, Notifications) in real React Native systems — specifically: Explain permission value upfront...

13.2 Chunked/Presigned Multipart Media Uploads & Image Compression [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Chunked/Presigned Multipart Media Uploads & Image Compression in real React Native systems — specifically: Upload compressed receipts and photos r...

13.3 FCM / APNs Push Notification Channels, Badges & Background Tap Routing [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply FCM / APNs Push Notification Channels, Badges & Background Tap Routing in real React Native systems — specifically: Re-engage users reliably acros...

13.4 Expo Config Plugins & Custom Native Android/iOS Manifest Tuning [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Expo Config Plugins & Custom Native Android/iOS Manifest Tuning in real React Native systems — specifically: Customize native Gradle/Podfile capab...

Module 13 Hands-On Capstone Project (100 Marks): Media Capture, Push Alert & Native Payment Integration — Design, implement, test, and document 'Media Capture, Push Alert & Native Payment Integration' applying every production concept from Module 13 (Native Permissions, Camera/Media Uploads, Pus...

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

RNTL + Maestro E2E Testing, Hermes Profiling, Sentry & EAS OTA Releases

Production-grade React Native Training engineering module covering RNTL + Maestro E2E Testing, Hermes Profiling, Sentry & EAS OTA Releases with hands-on labs and capstone build.

Objectives: ✓ Component Specs with Jest & React Native Testing Library — Purpose: Test mobile screens via accessibility labels and user events cleanly. • ✓ Black-Box Mobile E2E Flows with Maestro / Detox — Purpose: Verify login, OTP, and cart checkout automatically on iOS Simulator & Android Emulator. • ✓ Hermes Bytecode Startup Profiling, Bundle Size & Sentry Crash Sourcemaps — Purpose: Slash Time-to-Interactive (TTI) and debug native + JS crashes from one console. • ✓ EAS Build Profiles, Store Submissions & Safe Over-the-Air (OTA) Updates — Purpose: Ship critical JS bugfixes to users in minutes while respecting native runtime compatibility.

14.1 Component Specs with Jest & React Native Testing Library [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Component Specs with Jest & React Native Testing Library in real React Native systems — specifically: Test mobile screens via accessibility labels...

14.2 Black-Box Mobile E2E Flows with Maestro / Detox [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Black-Box Mobile E2E Flows with Maestro / Detox in real React Native systems — specifically: Verify login, OTP, and cart checkout automatically on...

14.3 Hermes Bytecode Startup Profiling, Bundle Size & Sentry Crash Sourcemaps [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Hermes Bytecode Startup Profiling, Bundle Size & Sentry Crash Sourcemaps in real React Native systems — specifically: Slash Time-to-Interactive (T...

14.4 EAS Build Profiles, Store Submissions & Safe Over-the-Air (OTA) Updates [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply EAS Build Profiles, Store Submissions & Safe Over-the-Air (OTA) Updates in real React Native systems — specifically: Ship critical JS bugfixes to...

Module 14 Hands-On Capstone Project (100 Marks): Production Hermes Bundle with Maestro E2E & Instant OTA — Design, implement, test, and document 'Production Hermes Bundle with Maestro E2E & Instant OTA' applying every production concept from Module 14 (RNTL + Maestro E2E Testing, Hermes Profiling...

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

React Native New Architecture (JSI/Fabric), Reanimated & Offline Sync

Self-explanatory list-and-bullet production mastery module covering React Native New Architecture (JSI/Fabric), Reanimated & Offline Sync in React Native Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ React Native New Architecture: JSI, Fabric Synchronous Layout & TurboModules — Purpose: Understand how direct C++ JSI bindings replace legacy JSON bridge bottlenecks for native speed. • ✓ High-Speed Lists (FlashList), UI-Thread Gestures (Reanimated) & Keyboard Handling — Purpose: Build silky-smooth mobile feeds, bottom sheets, and chat screens on budget Android & iOS devices. • ✓ Offline Queueing, MMKV/SecureStore Token Safety & Universal Deep Linking — Purpose: Persist mutations across offline periods and protect sensitive credentials in Keychain/Keystore. • ✓ Build, verify, and document '60fps Gesture-Driven Offline-First React Native App with MMKV & FlashList' 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 React Native New Architecture: JSI, Fabric Synchronous Layout & TurboModules [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply React Native New Architecture: JSI, Fabric Synchronous Layout & TurboModules in real React Native systems — specifically: Understand how direct C+...

15.2 High-Speed Lists (FlashList), UI-Thread Gestures (Reanimated) & Keyboard Handling [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply High-Speed Lists (FlashList), UI-Thread Gestures (Reanimated) & Keyboard Handling in real React Native systems — specifically: Build silky-smooth...

15.3 Offline Queueing, MMKV/SecureStore Token Safety & Universal Deep Linking [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Offline Queueing, MMKV/SecureStore Token Safety & Universal Deep Linking in real React Native systems — specifically: Persist mutations across off...

Module 15 Hands-On Capstone Project (100 Marks): 60fps Gesture-Driven Offline-First React Native App with MMKV & FlashList — Architect, implement, test, and document '60fps Gesture-Driven Offline-First React Native App with MMKV & FlashList' applying every bullet-wise engineering concept from Module 15 (React Nati...

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

React Native Performance Profiling, E2E Testing & EAS App Store Delivery

Self-explanatory list-and-bullet production mastery module covering React Native Performance Profiling, E2E Testing & EAS App Store Delivery in React Native Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Hermes Bytecode Startup Tuning, Image Caching & Eliminating JS-Thread Frame Drops — Purpose: Cut cold TTI startup times and keep navigation transitions responsive under heavy state updates. • ✓ Component Testing with React Native Testing Library & Automated E2E Mobile Flows — Purpose: Verify login, checkout, and offline recovery journeys automatically before store submission. • ✓ App Store / Play Store Signing, OTA Update Rules & Mobile System Design Interviews — Purpose: Navigate native permissions, push notifications, and crash triage in React Native interviews. • ✓ Build, verify, and document 'Hermes Bundle Audit, Jest + Maestro/Detox Test Flow & Signed Release 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 Hermes Bytecode Startup Tuning, Image Caching & Eliminating JS-Thread Frame Drops [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Hermes Bytecode Startup Tuning, Image Caching & Eliminating JS-Thread Frame Drops in real React Native systems — specifically: Cut cold TTI startu...

16.2 Component Testing with React Native Testing Library & Automated E2E Mobile Flows [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Component Testing with React Native Testing Library & Automated E2E Mobile Flows in real React Native systems — specifically: Verify login, checko...

16.3 App Store / Play Store Signing, OTA Update Rules & Mobile System Design Interviews [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply App Store / Play Store Signing, OTA Update Rules & Mobile System Design Interviews in real React Native systems — specifically: Navigate native pe...

Module 16 Hands-On Capstone Project (100 Marks): Hermes Bundle Audit, Jest + Maestro/Detox Test Flow & Signed Release Pipeline — Architect, implement, test, and document 'Hermes Bundle Audit, Jest + Maestro/Detox Test Flow & Signed Release Pipeline' applying every bullet-wise engineering concept from Module 16 (React...

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

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

Project: RouteMate Real-Time Rider Delivery, Live Map & Order Wallet Mobile App • **Confirmed Role:** React Native & TypeScript Mobile Engineering Intern

Domain & Production Infra: Last-Mile E-Commerce & Hyperlocal Mobility Client (React Native + Expo/CLI + TypeScript + Zustand/TanStack Query + Reanimated + OTA Releases)

- Developed cross-platform iOS & Android screens in React Native + TypeScript with React Navigation stacks/tabs.
- Optimized FlatList/FlashList rendering, gesture interactions, and offline request queues with TanStack Query.
- Integrated native device permissions, secure token storage, push alerts, and biometric session locks.
- Shipped production mobile builds with crash telemetry and automated component test verification.

Aligned Global Vendor Certifications: **META-RN** (Meta React Native Cross-Platform Specialist Credential) • **META-FE** (Meta Certified Front-End & React Developer Credential)

Target Job Roles: React Native Mobile Developer, Cross-Platform iOS/Android Engineer, React & React Native Product Engineer, Mobile UI/UX Engineer

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹5.0 LPA – ₹9.5 LPA **Mid-Level React Native Engineer (1–3 Yrs):** ₹10.0 LPA – ₹20.5 LPA **Unified Web + Mobile Staff React Architect (3–6+ Yrs):** ₹21.0 LPA – ₹40.0+ LPA

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

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