Faysal Aziz

Kotlin Multiplatform Mastery

for Modern Mobile Engineering - 3 days

FA-0031 · 3 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

Build Once, Deliver Native Experiences Across Platforms — with AI-Enhanced Android Apps

Mobile development has entered a phase where efficiency, maintainability, and cross-platform delivery are no longer optional—they are competitive requirements. Kotlin Multiplatform has emerged as one of the most strategic technologies for mobile teams seeking to share core logic across Android and iOS while still maintaining the power and performance of native applications. Organizations increasingly adopt it to reduce duplicated codebases and accelerate feature delivery without sacrificing platform-specific capabilities.

This intensive **3-day professional training program** is designed to equip developers with practical, production-ready skills for building Kotlin Multiplatform applications that integrate modern architectures, robust networking layers, shared persistence, and scalable CI/CD pipelines. The course goes beyond theory by focusing on the exact tooling, libraries, and patterns currently used in the industry.

Participants will learn from an **instructor with over 30 years of industry experience** who has trained thousands of engineers and led real production systems across enterprise environments. The instructor's approach is intentionally **industry-driven rather than academic**, emphasizing practical implementation patterns, real architecture decisions, and lessons learned from large-scale software systems. The training style blends conceptual clarity with real-world engineering discipline, allowing participants to understand not just *how* something works but *why experienced engineers design systems that way*.

The course also introduces **modern AI capabilities in Android applications**, including integration with generative AI models such as Gemini and on-device AI features using ML Kit APIs—capabilities increasingly expected in modern mobile products.

Learning outcomes

By the end of this training, participants will be able to:

- Understand the architecture and capabilities of Kotlin Multiplatform (KMP)
- Configure and manage a full Kotlin Multiplatform project targeting Android and iOS
- Implement shared business logic using modern Kotlin language features and coroutines
- Build cross-platform networking layers using Ktor
- Implement shared persistence layers using SQLDelight or Room Multiplatform
- Use Kotlin serialization for consistent JSON handling across platforms
- Apply modern multiplatform architecture patterns such as MVVM and MVI
- Implement dependency injection strategies across platforms
- Integrate Kotlin shared modules with **SwiftUI and Jetpack Compose**
- Work effectively with the **Expect/Actual pattern for platform-specific logic**
- Manage Swift interoperability, bridging Kotlin types into Swift environments
- Understand Kotlin/Native memory management and performance considerations
- Build shared navigation and state management layers
- Integrate third-party C/C++ libraries using Kotlin Native C-Interop
- Optimize binary sizes and build artifacts
- Implement CI/CD strategies for Kotlin Multiplatform delivery pipelines
- Distribute shared modules using KMMBridge
- Integrate analytics, remote configuration, and cross-platform services
- Integrate **Generative AI and on-device AI capabilities into Android Kotlin applications**

Prerequisites

Participants should have:

- Experience and intermediate know-how with **Kotlin**
- Expertise in **Android development**
- Working understanding of **object-oriented programming**
- Working knowledge of **REST APIs**
- Familiarity with **Gradle build systems**
- Working understanding of **mobile application architectures**

Recommended (strongly):

- Basic Swift or iOS development knowledge
- Experience with Jetpack Compose

Training outline

1. Kotlin Multiplatform Foundations

Kotlin Multiplatform ecosystem overview

- Cross-platform development strategies
- Native vs hybrid vs multiplatform approaches
- Advantages of Kotlin Multiplatform for mobile teams
- Production adoption patterns in modern organizations

Evolution of Kotlin for multiplatform development

- Kotlin 2.x ecosystem evolution
- Improvements introduced by the K2 compiler
- Compiler architecture and performance benefits
- Multiplatform language features

---

2. Kotlin Multiplatform Environment & Tooling

Kotlin/Native architecture

- LLVM compilation pipeline
- Kotlin Native runtime
- Binary generation for iOS platforms

Development tooling and environment configuration

- Android Studio and IntelliJ support for KMP
- Multiplatform project wizard and plugin ecosystem
- Gradle configuration for multiplatform targets

Multiplatform project structure

- Shared module architecture
- androidApp module organization
- iosApp module integration

Source set architecture

- Common source sets
- Platform-specific source sets
- Test source sets

Expect/Actual mechanism

- Platform abstraction patterns

- Implementing platform-specific services
- Device APIs and OS capabilities
- File system access implementations
- Device identifiers and system-level services

---

3. Shared Logic Development

Designing shared business logic

- Identifying reusable components
- Layered architecture in shared modules

Networking with Ktor

- Ktor client architecture
- Cross-platform HTTP engines
- API client abstraction design
- Authentication strategies
- Error handling patterns

Serialization across platforms

- kotlinx.serialization architecture
- JSON parsing across targets
- Custom serializers
- Data model versioning

Shared database persistence

- SQLDelight architecture
- Room Multiplatform concepts
- Schema design for shared modules
- Data access patterns
- Query abstraction layers

Concurrency and asynchronous programming

- Kotlin coroutines architecture
- Structured concurrency principles
- Multiplatform coroutine usage
- Coroutine scopes and lifecycle management
- Handling concurrency across Android and iOS
- Kotlin Native memory model considerations

---

4. The UI Layer

Native UI integration approach

- Shared logic with native UI frameworks
- Platform UI architecture patterns

Android UI implementation

- Jetpack Compose architecture
- State management in Compose
- Integration with shared ViewModels

iOS UI implementation

- SwiftUI architecture
- Shared business logic integration
- Data flow between SwiftUI and Kotlin modules

Compose Multiplatform (advanced)

- Compose Multiplatform architecture
- UI sharing strategies
- Platform-specific UI customization

Swift interoperability

- Kotlin to Objective-C compilation pipeline
- Swift interop architecture
- Mapping Kotlin types into Swift types
- Handling Flow and asynchronous streams

---

5. Advanced Swift–Kotlin Interoperability

Objective-C bridge architecture

- Interoperability constraints
- Swift compatibility considerations

SKIE framework usage

- Bridging sealed classes to Swift enums
- Enum interoperability patterns

KMP-NativeCoroutines

- Mapping Kotlin Flow to Swift AsyncSequence
- Asynchronous data streams in SwiftUI

Kotlin Native memory management

- Garbage collection in Kotlin/Native

- Memory profiling techniques
- iOS host process monitoring

---

6. Multiplatform Architecture & State Management

Architectural patterns for multiplatform systems

- MVVM in shared modules
- Model–View–Intent (MVI) architecture

Unidirectional data flow design

- Event processing pipelines
- State reducers

Navigation frameworks

- Decompose architecture
- Voyager navigation patterns
- Shared back-stack management

Resource management

- MOKO Resources framework
- Cross-platform resource sharing

Dependency injection strategies

- Koin multiplatform configuration
- Kodein DI patterns
- Platform-specific module loading

---

7. Advanced Multiplatform Engineering

Compiler plugins and code generation

- IR manipulation architecture
- Compose compiler plugin
- SQLDelight code generation pipeline

Native interoperability

- C-Interop fundamentals
- Interfacing with C/C++ libraries
- Wrapping libraries such as OpenSSL or FFmpeg

Performance optimization

- Binary size reduction strategies

- Dead Code Elimination (DCE)
- Linker configuration

---

8. AI Integration in Kotlin Android Applications

AI capabilities in modern Android applications

- Generative AI use cases in mobile apps
- On-device AI vs cloud AI architectures

AI frameworks for Android

- ML Kit architecture
- TensorFlow Lite integration patterns
- AI SDK ecosystems

Generative AI integration

- Gemini Developer API architecture
- Prompt-based application features
- Conversational interfaces in mobile apps
- Text generation and summarization capabilities

On-device AI with Gemini Nano

- ML Kit GenAI APIs
- Prompt API usage
- Privacy and latency advantages of local inference

AI feature implementation patterns

- Chat assistants inside mobile apps
- Smart summarization features
- Context-aware recommendations
- Image and multimodal AI features

AI architecture integration

- Integrating AI services into clean architecture
- AI service abstraction layers
- Observability and logging for AI systems
- Responsible AI considerations in mobile apps

---

9. Advanced CI/CD & Delivery

(Aligned with the requirements document)

Multiplatform build pipelines

- CI/CD architecture for KMP projects
- Automated build pipelines

KMMBridge integration

- Binary distribution strategies
- Delivering shared frameworks to iOS teams

Release management

- Versioning shared modules
- Dependency synchronization

Observability and analytics

- Cross-platform analytics wrappers
- Firebase integration patterns
- Amplitude integration strategies

Remote configuration systems

- Shared configuration architecture
- Feature flagging strategies

© 2026 Faysal Aziz · Kuala Lumpur