Faysal Aziz

Mastering Modern iOS Dev

with SwiftUI, Concurrency & AI - 3 days

FA-0312 · 3 days

Software Development

Introduction

Build scalable, high-performance iOS applications using modern Swift architecture, SwiftUI, and AI-powered capabilities, guided by a veteran instructor with over 30 years of real industry experience.

Modern iOS development has reached a point where performance, architecture, and developer productivity intersect. SwiftUI, Swift Concurrency, and new frameworks such as SwiftData are shaping how scalable applications are designed. At the same time, advances in AI integration are transforming mobile apps from static tools into intelligent systems capable of assisting users, predicting behavior, and automating workflows.

This three-day intensive program is designed for developers who want to move beyond basic app development and adopt the practices used in professional production environments. Participants will explore modern Swift language capabilities, advanced SwiftUI design patterns, concurrency, scalable architecture strategies, and SDK design principles.

A major differentiator of this training is the instructor. The course is delivered by a professional with **over 30 years of industry experience**, bringing practical insights from real software projects, enterprise architecture work, and large-scale mobile deployments. Instead of academic theory, the training emphasizes **real-world engineering practices, industry demanded content, and practical architectural thinking** that professionals apply when building production iOS systems.

The training style is also intentionally unique. Rather than simply presenting slides, the instructor focuses on explaining **why architectural decisions matter, how modern Apple frameworks work internally, and how experienced engineers approach system design**. This results in a deeper understanding that developers can immediately apply in their own projects.

The technical scope of the training also aligns with the advanced topics outlined in the provided requirements document, including protocol-oriented programming, SwiftUI architecture, concurrency, scalable application design, and SDK library design.

Learning outcomes

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

- Design scalable iOS applications using modern Swift architecture patterns.
- Apply Protocol-Oriented Programming and generics to build reusable components.
- Build complex UI systems using SwiftUI's declarative architecture.
- Implement modern state management patterns using the Observation framework.
- Use Swift 6 structured concurrency for safe, performant asynchronous programming.
- Design scalable architectures using MVVM, Coordinators, and modular architectures.
- Implement dependency injection strategies to improve testability.
- Persist application data using SwiftData and Core Data.
- Build modular SDK libraries and reusable frameworks.
- Optimize application performance using profiling and testing tools.
- Implement CI/CD pipelines for automated builds and testing.
- Integrate AI capabilities into iOS applications using modern AI services and frameworks.
- Apply industry best practices learned from real production environments.

Prerequisites

Participants should have the following background:

- Basic to intermediate Swift programming knowledge
- Experience building at least one iOS application
- Familiarity with Xcode development environment
- Basic understanding of object-oriented programming
- Understanding of REST APIs and networking concepts
- Familiarity with UIKit or basic SwiftUI concepts

Recommended but not required:

- Experience with MVVM architecture
- Basic knowledge of Git and version control

Training outline

1. Foundations of Modern Swift Language Architecture

Advanced Swift Language Capabilities

Protocol-Oriented Programming

- Protocol composition
- Protocol extensions
- Conditional conformance
- Designing flexible architectures using protocols

Generics and Type Constraints

- Generic functions
- Generic data structures
- Associated types
- Opaque types using some
- Existential types using any
- Designing reusable abstractions

Property Wrappers and Swift Macros

- Custom @propertyWrapper development
- Eliminating boilerplate using wrappers
- Swift 6 compile-time macros
- Code generation through macros
- Comparison between macros and runtime wrappers

Memory Management

- Automatic Reference Counting deep dive
- Retain cycles in closures
- Weak vs unowned references
- Memory leak detection strategies
- Best practices for long-lived objects

---

2. SwiftUI Architecture and UI System Design

SwiftUI Fundamentals

- The View protocol
- Declarative UI architecture

- The body property lifecycle
- View identity and rendering

Layout and View Composition

Layout containers

- HStack
- VStack
- ZStack

Advanced layout systems

- Grid API
- Lazy layouts

Responsive layout design

- GeometryReader
- LayoutPriority
- ViewThatFits

Efficient rendering strategies

- List
- LazyVStack
- LazyVGrid

State Management in SwiftUI

Local state

- @State
- @Binding

The Observation framework

- @Observable
- differences from ObservableObject

Dependency injection with SwiftUI

- @Environment
- .environment()

- Designing predictable data flow
- MVVM in SwiftUI

Navigation and Interaction Design

Navigation architecture

- NavigationStack
- NavigationPath

- Programmatic navigation
- Deep linking

Modal presentation

- .sheet
- .fullScreenCover
- .popover

Gesture system

- Tap gestures
- Long press gestures
- Drag gestures
- Magnification gestures

---

3. Data Persistence and Framework Integration

SwiftData Framework

- SwiftData architecture
- The @Model macro
- Querying data using property wrappers
- Model context and lifecycle
- Integration with SwiftUI views
- Data migration strategies
- Comparison with Core Data

SwiftData introduces a Swift-native approach to persistence, allowing developers to define models using macros and interact with them through SwiftUI-friendly APIs.

Core Data Integration

- Core Data fundamentals
- Managing object graphs
- Data synchronization strategies
- SwiftData vs Core Data usage scenarios

UIKit Interoperability

- Integrating legacy UIKit views
- UIViewRepresentable
- Hybrid UI strategies
- Migration approaches from UIKit to SwiftUI

---

4. Swift 6 Concurrency and Safe Parallel Programming

Modern Concurrency Model

- Structured concurrency
- Task lifecycle
- TaskGroup
- async let

Actor-based Isolation

- Actors
- @MainActor
- Global actors
- Actor isolation rules

Data Race Prevention

- Sendable protocol
- Immutable data patterns
- Thread safety design

Modern Swift concurrency improves application safety by introducing compile-time checks and actor-based isolation to prevent race conditions.

Concurrency Integration in SwiftUI

- .task modifier
- Async data loading
- UI responsiveness strategies
- Background task management

---

5. Scalable iOS Architecture and Design Patterns

Advanced MVVM Architecture

- ViewModel responsibilities
- State management
- Data binding strategies

MVVM-Coordinator Architecture

- Navigation separation
- Flow coordination
- Decoupling UI and navigation logic

Unidirectional Data Flow Architectures

- Redux-style architectures
- The Composable Architecture concepts
- Predictable state management

Dependency Injection

- Protocol-based injection
- Container-based injection
- Testable architecture design

Modular Application Design

- Swift Package Manager modules
- Feature-based modularization
- Dependency boundaries
- Build time optimization

Modern iOS architecture increasingly combines SwiftUI, concurrency, and modular design to improve scalability and maintainability.

---

6. SDK Library Architecture and Framework Design

Designing Reusable SDKs

- Public vs internal APIs
- Stable interface design
- API versioning strategies

Modular Library Design

- Avoiding dependency conflicts
- Dependency isolation
- Library integration strategies

Framework Packaging

- Framework vs XCFramework
- Binary distribution
- Multi-platform compatibility

Resource Management

- Resource bundling
- Managing images and assets
- Core Data models within libraries

Privacy and Compliance

- Privacy manifests

- App Store privacy requirements
- Data transparency practices

---

7. Quality Engineering and Performance Optimization

Test-Driven Development

- Unit testing
- Integration testing
- UI testing
- Swift Testing framework

Performance Profiling

- Instruments overview
- Memory leak detection
- CPU profiling
- Main thread analysis

Continuous Integration and Delivery

- Xcode Cloud pipelines
- GitHub Actions
- Automated testing workflows
- Build automation strategies

---

8. Integrating Artificial Intelligence into iOS Applications

AI Opportunities in Mobile Applications

- Intelligent assistants
- Predictive recommendations
- Personalization engines
- Natural language interfaces

AI Architecture for Mobile Apps

- On-device vs cloud AI
- Privacy considerations
- Model deployment strategies

Using Apple AI Frameworks

- Core ML fundamentals
- Natural Language framework

- Vision framework

Integrating Large Language Models

- AI powered chat interfaces
- prompt engineering for mobile apps
- building AI powered features

AI-Driven App Features

- smart search systems
- AI assisted content generation
- recommendation engines
- automation workflows

Performance and Cost Considerations

- model optimization
- offline inference
- hybrid AI architectures

AI integration is becoming one of the defining trends in modern iOS development, enabling applications to become intelligent, context-aware systems.

---

9. Industry Engineering Practices from a 30-Year Veteran Instructor

Real-World Engineering Decision Making

- Architecture trade-offs
- Choosing frameworks wisely
- Avoiding common enterprise mistakes

Production-Ready Code Practices

- Code maintainability strategies
- Scaling development teams
- Engineering discipline and architecture governance

Lessons from Large Production Systems

- Real industry challenges
- Scaling mobile applications
- Managing technical debt

The instructor's **30+ years of industry experience** ensures that every topic is taught through the lens of real engineering practice rather than academic theory. Participants gain insight into how professional engineers design systems, solve architectural challenges, and deliver reliable production software.

© 2026 Faysal Aziz · Kuala Lumpur

https://faysalaziz.com/courses/mastering-modern-ios-dev/