Faysal Aziz

iOS Apps with React Native

from First Screen to On-Device AI - 3 days

FA-0266 · 3 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

A 3-day industry-focused program that takes developers from React Native fundamentals to production-ready iOS apps with cloud AI and Apple on-device intelligence.

This course is designed as a practical, market-facing training rather than an academic survey. It is led by an instructor with over 30 years of industry experience, using real industry-demanded content, workflows, and architectural decisions that reflect how modern mobile teams actually build and ship React Native iOS products with AI capabilities.

React Native currently recommends a framework-based approach for most new apps, OpenAI recommends the Responses API for new integrations, and Apple now provides both Foundation Models for on-device language features and Core ML for broader local inference and optimization.

Learning outcomes

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

- Explain how React Native fits into modern iOS development and how its current architecture affects app design and performance.
- Set up and structure a React Native iOS project using current recommended workflows.
- Build core iOS app features with React Native, including navigation, state management, forms, networking, storage, and device integration.
- Understand the role of native iOS modules, JSI, TurboModules, and bridging when React Native apps need platform-specific capabilities.

- Design and implement API-based AI features in a React Native app using contemporary OpenAI platform patterns, including the Responses API, structured outputs, and realtime interaction models where appropriate.
- Evaluate when to use server-hosted AI versus on-device AI for latency, privacy, cost, resilience, and user experience.
- Integrate on-device AI on Apple platforms through Foundation Models and Core ML, including the implications of adapters, model packaging, performance, and OS-version compatibility.
- Plan production considerations for intelligent iOS apps, including observability, testing, safety, privacy, app lifecycle, and deployment readiness.

Prerequisites

- Basic JavaScript knowledge
- Familiarity with React fundamentals
- Comfortable using the command line and npm
- Basic understanding of API requests and JSON
- Some exposure to mobile app concepts is helpful
- Access to a Mac suitable for iOS development
- Xcode and Apple development tooling installed
- Interest in building production-grade apps rather than tutorial-only prototypes

Training outline

1. Course orientation and technical framing

1. What React Native is in the current iOS landscape
2. Why React Native remains relevant for iOS product teams

   Current state of React Native releases and architecture

   1. Stable release direction
   2. New Architecture as the default path forward
   3. Implications for long-term maintainability

   The spectrum of intelligence in mobile apps

   1. Rules-based features
   2. Cloud AI features
   3. On-device AI features
   4. Hybrid AI architectures

   Course delivery model

1. Industry-first mindset
2. Production-oriented decision making
3. Real-world engineering expectations

2. React Native foundations for iOS development

React Native mental model

1. JavaScript and native runtime relationship
2. Declarative UI principles
3. Component-driven app construction

React Native project models

1. Framework-based React Native workflow
2. Bare workflow considerations
3. Criteria for choosing the right starting point

iOS development context for React Native engineers

1. Xcode project structure
2. Simulators and device builds
3. Bundling, signing, and platform constraints

Local development environment

1. Toolchain requirements
2. Node and package management
3. CocoaPods and iOS dependency management
4. Simulator and physical device setup

Project bootstrap and directory strategy

1. App entry points
2. Feature-based folder organization
3. Shared code versus native code boundaries

Type safety and maintainability

1. TypeScript adoption strategy
2. Interface design for scalable apps
3. Domain modeling for UI and data

3. Core React Native UI and app structure

Essential primitives

1. Views

2. Text
3. Images
4. Scroll containers
5. Lists

Styling system

1. Flexbox for mobile layouts
2. Design consistency
3. Reusable style patterns
4. Theming foundations

Component architecture

1. Presentational and container components
2. Reusable UI systems
3. Screen composition patterns

Input and interaction

1. Touch handling
2. Gestures
3. Keyboard handling
4. Accessibility-aware interactions

Navigation architecture

1. Stack navigation
2. Tab navigation
3. Nested navigators
4. Deep linking concepts

6. Screen lifecycle and focus handling

Form design

1. Validation
2. Submission flow
3. Error states
4. Input ergonomics on iOS

4. State, data flow, and application logic

State fundamentals

1. Local state
2. Derived state
3. Shared state

React patterns for mobile apps

1. Effects and side effects
2. Memoization
3. Context boundaries

Scalable state management

1. Lightweight state patterns
2. Store-based state management
3. Domain-centric data organization

Asynchronous workflows

1. Fetching
2. Caching
3. Refresh strategies
4. Optimistic UI

Error handling strategy

1. Transport errors
2. Business logic errors
3. User feedback surfaces

Configuration management

1. Environment variables
2. Secrets strategy
3. Build-specific configuration

5. Networking and backend connectivity

API fundamentals in mobile applications

1. REST integration
2. Streaming responses
3. Pagination
4. Retry behavior

Authentication patterns

1. Token handling
2. Session lifecycle
3. Secure transmission principles

Data layer design

1. API clients
2. Service abstractions

3. Request orchestration

Offline-aware considerations

1. Connectivity changes
2. Sync patterns
3. Deferred actions

5. File and media transfer basics
6. Background-friendly communication patterns

6. Device capabilities and iOS integration

1. Working with iOS-specific behavior in React Native

Permissions strategy

1. Camera
2. Microphone
3. Photos
4. Notifications
5. Location

Local storage and persistence

1. Key-value persistence
2. Secure storage
3. Caching layers

Media capabilities

1. Image input
2. Audio input
3. Capture workflows

5. Notifications and user re-engagement
6. Share sheets and inter-app workflows

App lifecycle awareness

1. Foreground and background transitions
2. Resource management
3. Resume strategies

Performance fundamentals

1. Render cost
2. List performance
3. Memory pressure
4. Startup optimization

7. Native extension points for advanced iOS work

1. Why and when React Native needs native code

   Native module fundamentals

   1. Data flow between JavaScript and Swift
   2. Capability exposure patterns

   React Native New Architecture concepts

   1. TurboModules
   2. JSI
   3. Fabric overview
   4. Codegen role

   Designing native boundaries

   1. Stable interfaces
   2. Async contracts
   3. Error propagation

5. Packaging native functionality for reuse
6. Maintaining mixed JavaScript and Swift codebases

8. Production-quality iOS app engineering

App architecture for scale

1. Layered design
2. Domain separation
3. Feature modularity

Observability

1. Logging
2. Error reporting
3. Performance tracing

Quality engineering

1. Unit testing
2. Component testing
3. Integration testing
4. End-to-end testing

Build and release management

1. Debug and release configurations
2. Environment promotion

3. CI/CD concepts

App Store readiness

1. Metadata planning
2. Privacy disclosures
3. Performance and stability expectations

9. Introduction to AI in mobile products

AI product patterns for iOS apps

1. Chat interfaces
2. Summarization
3. Classification
4. Extraction
5. Recommendation
6. Voice experiences

Choosing the right AI architecture

1. API-based AI
2. On-device AI
3. Hybrid orchestration

Decision criteria

1. Latency
2. Privacy
3. Cost
4. Reliability
5. Battery and compute constraints

UX for intelligent features

1. Human-in-the-loop design
2. Transparency
3. Failure handling
4. Confidence-aware interfaces

5. Safety and policy-aware product thinking

10. Integrating API-based AI into React Native apps

Cloud AI architecture fundamentals

1. Client app

2. Secure backend
3. AI provider integration
4. Response pipeline

Why AI API calls should be brokered through a backend

1. Credential protection
2. Rate limiting
3. Auditing
4. Centralized policy enforcement

Modern OpenAI platform patterns for new applications

1. Responses API as the recommended integration path
2. Model selection strategy
3. Stateful versus stateless request design

Prompt design for application features

1. Instruction design
2. Context packaging
3. Role separation
4. Output control

Structured outputs in mobile workflows

1. Schema-first response design
2. Typed UI rendering
3. Safer downstream handling

Multi-turn app experiences

1. Conversation state
2. Session continuity
3. Memory boundaries

Streaming and responsive UI

1. Partial response rendering
2. Loading states
3. Interruptibility

Realtime and multimodal feature concepts

1. Speech-driven experiences
2. Low-latency assistant scenarios
3. Audio interaction pipelines

Grounding and tool use

1. External data retrieval
2. Function calling concepts

3. Action-taking workflows

Cost and reliability controls

1. Token budgeting
2. Caching
3. Fallback behavior
4. Model tiering

11. Backend patterns for AI-enabled mobile apps

Service boundary design

1. Mobile client responsibilities
2. API gateway responsibilities
3. AI orchestration layer

2. Authentication and authorization for AI features
3. Secure prompt and context assembly
4. Content filtering and safety checks
5. Usage metering and quota management
6. Persistent conversation design
7. Retrieval-augmented patterns for app knowledge
8. Data privacy and compliance-aware architecture

Monitoring AI quality in production

1. Latency
2. Failure modes
3. Output drift
4. User correction loops

12. Building intelligent UX in React Native

AI-first interaction design

1. Input capture patterns
2. Result presentation patterns
3. Progressive disclosure

Managing asynchronous intelligence in the interface

1. Waiting states
2. Interruptions
3. User edits

4. Confirmation patterns

Trustworthy response presentation

1. Structured cards
2. Explanatory affordances
3. Source-aware design

4. Voice and multimodal UI concepts
5. Local-first versus cloud-first interaction flows
6. Designing for graceful degradation

13. On-device AI on Apple platforms

The Apple on-device AI landscape

1. Foundation Models framework
2. Core ML
3. Create ML
4. Apple silicon execution model

Foundation Models framework fundamentals

1. Role of the on-device language model
2. App-specific intelligent tasks
3. Tool-calling potential
4. Prompt-driven local experiences

Suitability analysis for Foundation Models

1. Private text generation
2. Summarization
3. Rewriting
4. Intent support
5. Local semantic assistance

Core ML fundamentals

1. Model formats
2. Conversion pipeline
3. Deployment pipeline
4. Execution on device

Choosing between Foundation Models and Core ML

1. Language-centric versus model-centric tasks
2. System model access versus custom model ownership
3. Development effort trade-offs

6. Create ML role in rapid model creation

On-device inference constraints

1. Model size
2. Memory use
3. Cold start considerations
4. Battery impact
5. Hardware variability

14. Bridging on-device AI into React Native

Native integration strategy for Apple AI frameworks

1. Swift wrapper layer
2. React Native bridge surface
3. JavaScript consumption model

Exposing Foundation Models capabilities to React Native

1. Request lifecycle
2. Result marshaling
3. Error handling
4. Capability checks

Exposing Core ML inference to React Native

1. Model loading
2. Input preprocessing
3. Inference execution
4. Output postprocessing

Bridging design decisions

1. Promise-based APIs
2. Event-based APIs
3. Streaming-capable interfaces

Performance-aware bridging

1. Minimizing serialization overhead
2. Background execution considerations
3. Keeping UI responsive

6. Packaging native AI functionality into reusable modules

15. Advanced on-device customization

Foundation Models adapters

1. What adapters are
2. When adapters are appropriate
3. Training workflow overview
4. Packaging and deployment implications

Adapter lifecycle management

1. System-model version coupling
2. OS-version compatibility planning
3. Re-training strategy
4. Entitlement requirements

Core ML optimization paths

1. Model conversion
2. Compression techniques
3. Stateful model execution
4. Multi-function models

Performance profiling on Apple platforms

1. Xcode model inspection
2. Performance reports
3. Instruments-based profiling

Security and protection of shipped models

1. Model packaging
2. Encryption considerations
3. Asset delivery strategy

16. Hybrid AI architectures in iOS apps

1. When to combine API AI and on-device AI

   Routing logic

   1. Privacy-first routing
   2. Latency-first routing
   3. Capability-based routing
   4. Cost-aware routing
3. Offline and degraded-mode user journeys
4. Local preprocessing with cloud completion

5. Cloud orchestration with local postprocessing
    Feature segmentation
        1. Which tasks belong on device
        2. Which tasks belong in the cloud
        3. Which tasks need both
7. Operational architecture for hybrid intelligence

17. AI security, privacy, and responsible shipping

1. Secret management and backend isolation
2. Minimizing sensitive data exposure
3. Privacy-first product design for on-device intelligence
4. Consent and permission-aware design
5. Guardrails for generated output
6. Misuse prevention
    Logging strategy for AI systems
        1. What to store
        2. What not to store
        3. Redaction principles
    Reliability engineering for AI features
        1. Fallback UX
        2. Failure containment
        3. Human override patterns

18. Testing AI-enabled mobile applications

1. Functional testing of AI pipelines
2. Deterministic testing around non-deterministic systems
3. Schema validation for structured outputs
4. Prompt regression testing
5. Latency and load testing
6. Device-specific testing for on-device inference
    Quality evaluation frameworks
        1. Accuracy
        2. Relevance
        3. Safety

4. UX consistency

8. Release gates for AI features

19. Capstone architecture and production roadmap

End-to-end intelligent iOS app blueprint

1. React Native front end
2. Native iOS AI bridge
3. Secure backend
4. Cloud AI integration
5. Optional on-device model path

2. Technical sequencing for building from zero to production

Team roles and collaboration model

1. Mobile engineer
2. Backend engineer
3. AI engineer
4. Product and design stakeholders

4. Migration path from simple mobile app to intelligent product
5. Common failure points in real projects
6. Readiness checklist for deployment and maintenance

Long-term evolution strategy

1. Upgrading React Native
2. Adapting to Apple AI platform changes
3. Managing model and API changes over time

This outline reflects current guidance from React Native, Apple, and OpenAI as of February 2026, including React Native 0.84, the New Architecture direction, OpenAI's recommendation to use the Responses API for new builds, and Apple's Foundation Models and Core ML pathways for on-device intelligence.

Disclaimer

This course outline is provided as a general guideline for planning and discussion purposes only. It is intended to present the proposed scope, sequence, and coverage of topics based on the current understanding of the training requirements. The content, depth, duration, delivery approach, and emphasis of any section may be revised, expanded, reduced, or reorganized at any time to align with participant profiles, organizational objectives, technology updates, scheduling considerations, or other evolving requirements. Final course coverage will be determined in consultation with the client and may be adjusted to ensure the training remains relevant, practical, and aligned with real-world industry needs.

© 2026 Faysal Aziz · Kuala Lumpur