Faysal Aziz

Build Once, Reach Everywhere

A 3-Day Intensive in React Native with Expo, React Native Web, and Modern App Capabilities

FA-0399 · 3-Day

Software Development · Cybersecurity

Introduction

Modern app teams are under pressure to move faster without lowering quality. That is why React Native with Expo remains such a practical choice: it lets teams build native mobile experiences with a strong developer workflow, broad device API access, and a path to web delivery through React Native Web and Expo's universal app model. React Native Web remains the compatibility layer that lets React Native components render on the web, while Expo Router supports navigation across native and web in a unified way.

This course is designed as a focused 3-day industry workshop rather than an academic survey. It covers the foundations of React Native with Expo, introduces React Native Web as a practical extension of the same codebase, and then moves into the app capabilities teams routinely ask for in real projects: push notifications, deep linking, biometric authentication, file and document workflows, global state management, and TypeScript-first TSX development. Expo currently provides maintained support for notifications, document picking, file system access, biometric authentication, and TypeScript setup, which makes it a suitable training base for these topics.

The instructor for this course brings over 30 years of industry experience and will teach with real industry-demanded content, focusing on the way production teams actually structure apps, choose tooling, and balance speed with maintainability.

Learning outcomes

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

- Explain how React Native, Expo, and React Native Web fit together in a modern cross-platform app strategy.
- Create and structure a React Native application using Expo with TypeScript and TSX.
- Build core screens, components, layouts, and navigation flows for mobile and web using current React Native and Expo Router practices.
- Apply responsive and platform-aware techniques when sharing code between mobile and web through React Native Web.
- Implement push notification basics, including token flow, receiving notifications, and handling user interaction.
- Implement deep linking concepts for screen routing and app entry points.
- Add biometric authentication using Expo's local authentication capabilities.
- Handle documents and files using document picker and file system APIs.
- Compare global state approaches and choose between Context, Redux Toolkit, and Zustand based on project needs. Redux Toolkit is still the standard Redux approach, while Zustand remains a lightweight hook-based alternative.
- Organize a small production-style app with cleaner state boundaries, typed props, typed navigation inputs, and maintainable feature structure.

Prerequisites

- Experienced with JavaScript fundamentals
- Good React knowledge, including components, props, state, hooks, and JSX
- Familiarity with HTML and CSS concepts
- Advanced command line usage
- Prior exposure to npm or yarn package management
- TypeScript basics
- Mobile app development concepts such as navigation, forms, and API calls

Training outline

1. Course foundation and cross-platform application strategy

1. What React Native is and where it fits
2. Why Expo is commonly chosen as the framework entry point
3. The role of React Native Web in a shared-code architecture
4. Mobile-first versus universal app thinking
5. Native capabilities versus shared business logic

6. Understanding where one codebase helps and where platform divergence still matters

Current ecosystem direction

1. Framework-first React Native setup
2. Expo-managed workflows and development ergonomics
3. React Native Web as a compatibility layer
4. TypeScript-first development expectations

Performance and runtime considerations

1. Hermes direction in current React Native releases
2. New Architecture awareness at a conceptual level

2. Development environment and project setup

1. Creating a new Expo project
2. Project bootstrapping choices
3. Understanding the generated project structure

Running the app on

1. Android
2. iOS
3. Web

Development tools and workflows

1. Expo CLI basics
2. Local development and preview flows
3. Device and simulator testing
4. Web preview and debugging

Core configuration files

1. App configuration
2. TypeScript configuration
3. Babel and Metro awareness
4. Environment variable awareness

Working with dependencies

1. Expo-managed packages
2. Third-party React Native libraries
3. Version compatibility awareness

3. TypeScript and TSX for React Native

1. Why TypeScript matters in application teams
2. TSX syntax and component authoring
3. Typing props
4. Typing component state
5. Typing hooks and event handlers
6. Typing async functions and API responses
7. Shared types across features
8. Utility types for app development

   Safer patterns for

   1. Forms
   2. Lists
   3. Reusable UI components
   4. State stores

   Migration mindset

   1. Moving from JSX to TSX
   2. Incremental typing strategies
   3. Common mistakes and anti-patterns

4. React Native fundamentals with Expo

Core primitives

1. View
2. Text
3. Image
4. ScrollView
5. TextInput
6. Pressable
7. FlatList

Styling model

1. StyleSheet
2. Inline styles
3. Composition patterns
4. Platform-specific style handling

Layout system

    1. Flexbox in React Native
    2. Sizing and spacing
    3. Alignment patterns
    4. Scroll and content organization

  Component architecture

    1. Presentational and container boundaries
    2. Reusable UI patterns
    3. Feature-based structure

  State and rendering model

    1. Local component state
    2. Derived state
    3. Controlled inputs
    4. Rendering performance awareness

  Hooks in application development

    1. useState
    2. useEffect
    3. useMemo
    4. useCallback
    5. Custom hooks

5. Navigation and routing in modern Expo apps

1. Navigation concepts in mobile and web
2. File-based routing with Expo Router
3. Route organization
4. Nested routes
5. Dynamic routes
6. Layout routes
7. Tabs, stacks, and modal patterns
8. Passing data and route parameters
9. Navigation state awareness
10. Web route behavior and URL alignment
11. Guarded routes and authenticated flows
12. Route-based code organization

6. React Native Web fundamentals

1. What code sharing really means
2. How React Native Web maps components to the browser
3. Shared UI versus platform-specialized UI

   Designing components that work across

   1. Touch devices
   2. Desktop browsers
   3. Different viewport sizes
5. Styling considerations for web compatibility
6. Responsive design patterns

   Interaction differences

   1. Touch
   2. Mouse
   3. Keyboard
8. Accessibility awareness for universal apps
9. Web-specific constraints and opportunities
10. Handling web-only and native-only concerns cleanly
11. Universal navigation and route consistency

7. Data flow and state management strategy

Understanding state layers

1. Local UI state
2. Shared feature state
3. Server state awareness
4. Session and auth state

Choosing the right state tool

1. When local state is enough
2. When Context becomes difficult
3. When Zustand is a good fit
4. When Redux Toolkit is justified

Zustand

1. Store fundamentals
2. Hook-based access patterns
3. Slices and modular store structure

4. Type-safe store definitions
5. Actions and selectors
6. Persisting lightweight app state awareness

Redux Toolkit

1. Store setup
2. Slices
3. Actions
4. Reducers
5. Middleware awareness
6. Async logic concepts
7. Typed hooks
8. Feature-based Redux organization

Comparative decision-making

1. Boilerplate versus control
2. Team scalability
3. Debugging and traceability
4. Learning curve
5. Suitability for small, medium, and larger applications

State management anti-patterns

1. Over-globalizing state
2. Duplicated state
3. Mutable update mistakes
4. Navigation and state coupling issues

8. Working with APIs and asynchronous application behavior

1. Fetching data in Expo apps
2. Organizing service layers
3. Async patterns with TypeScript
4. Loading, error, and empty states
5. Caching awareness
6. Optimistic UI awareness
7. Authentication and session-aware requests
8. Coordinating async data with global state

9. Push notifications

1. Notification concepts and app use cases
2. Expo notifications capability overview

   Notification lifecycle

   1. Permission request flow
   2. Token acquisition
   3. Registration concepts
   4. Delivery concepts
   5. Foreground and background behavior awareness
4. Local notifications versus push notifications
5. Notification payload awareness
6. Handling notification responses
7. Routing users from notifications into screens

   Platform considerations

   1. Android awareness
   2. iOS awareness
   3. Device testing constraints

   UX and product considerations

   1. Permission timing
   2. Relevance and personalization
   3. Avoiding notification fatigue
10. Debugging and validation workflow

10. Deep linking

1. Why deep linking matters in modern app journeys
2. URL schemes and app entry concepts
3. Deep linking versus universal links and app links awareness
4. Linking fundamentals in React Native
5. Route mapping with Expo Router

   Deep linking into

   1. Public screens
   2. Authenticated screens
   3. Content detail screens
   4. Marketing and onboarding flows

7. Parameter handling and validation
8. Handling cold start and warm start flows
9. Notification-to-screen linking patterns
10. Testing and troubleshooting deep links
11. Security and misuse awareness

11. Biometric authentication

1. Authentication layers in mobile apps
2. Biometrics as a local authentication capability
3. Expo local authentication overview
   Platform support awareness
   1. Face ID
   2. Touch ID
   3. Fingerprint prompt

   Basic biometric flow
   1. Capability check
   2. Enrollment awareness
   3. Prompt invocation
   4. Success and failure handling
6. Biometrics and session design
7. Re-authentication patterns
   UX considerations
   1. Fallbacks
   2. Retry rules
   3. User trust and consent

   Security boundaries
   1. What biometrics do and do not secure
   2. Local auth versus server auth
   3. Sensitive action confirmation patterns

12. File and document handling

1. Common document workflows in business apps
2. Using document picker in Expo
3. Selecting files from device providers

4. Understanding file metadata
5. Reading and storing file references

    File system fundamentals with Expo

    1. Local storage concepts
    2. Reading files
    3. Writing files
    4. Downloading files
    5. Temporary versus persistent file awareness

    Import workflows

    1. Picking documents
    2. Validating file types
    3. Size awareness
    4. Error handling

    Export workflows awareness

    1. Creating downloadable content
    2. Sharing concepts

    File lifecycle concerns

    1. Cleanup
    2. Permissions awareness
    3. Cross-platform inconsistencies

10. User experience considerations for document-driven mobile apps

13. Authentication and protected app flows

1. Authentication concepts in Expo applications
2. Session handling awareness
3. Protected routes with Expo Router
4. Login flow structure
5. Token storage awareness
6. Re-entry and app resume considerations
7. Combining auth flows with biometrics
8. Redirect logic and guarded navigation
9. Web and native parity considerations

14. Platform awareness and app quality

1. iOS, Android, and web differences that matter
2. Input and interaction differences
3. Permission handling patterns
4. Error boundaries and resilience
5. Offline and weak-network awareness

   Performance thinking

   1. Rendering discipline
   2. List performance awareness
   3. Memoization awareness
   4. Avoiding unnecessary re-renders

   Accessibility basics

   1. Labels
   2. Focus awareness
   3. Touch target sizing
   4. Screen reader awareness

   Debugging workflow

   1. Logs
   2. Device testing
   3. Cross-platform verification

9. Release-readiness mindset

15. Application architecture for maintainability

1. Folder structures for real projects
2. Feature-first organization
3. Co-locating UI, hooks, services, and types
4. Shared component libraries
5. Separating domain logic from presentation
6. Managing configuration and constants
7. Error and status modeling
8. Reusable utility modules
9. Team conventions and code quality expectations

16. Capstone-oriented integration topics

1. Bringing routing, state, auth, notifications, and file handling together
2. Designing a small production-style app shell
3. Shared layout and navigation structure
4. Feature composition strategy
5. Typed data flow across screens
6. Cross-platform testing checklist

   Common integration pitfalls

   1. Route-state mismatch
   2. Permission sequencing issues
   3. Platform-specific behavior drift
   4. Type mismatches in async flows

17. Delivery, deployment, and next-step awareness

1. Development build awareness
2. Preview and internal distribution awareness
3. Web deployment awareness
4. App store preparation awareness
5. Production configuration awareness
6. Monitoring and maintenance mindset
7. Evaluating when to stay managed and when to expand native customization

Suggested Positioning for the Course

This outline fits a **3-day intensive hands-on program** for developers, web engineers, or internal app teams who want a practical entry into Expo-based React Native development while still touching the capabilities that usually appear in real delivery work. It is especially suitable for teams that want to build toward a shared Android, iOS, and web strategy using TypeScript and current Expo tooling.

© 2026 Faysal Aziz · Kuala Lumpur