3. REVIEW AND COMPARISON OF MOBILE APP DEVELOPMENT FRAMEWORKS BY COMPARISON OF THEIR KEY FEATURES AND WHERE THEY CAN BE USED.

Framework	Language	Performance	Cost & Time to Market	UX & UI	Complexity	Community Support	Where it can be used
React Native	JavaScript	**High**: Utilizes native components via JavaScript bridge. Performance can suffer for complex UI animations.	**Moderate**: Single codebase for iOS and Android reduces development time. Some platform-specific customization may be needed.	**Good**: Native-like feel with platform-specific UI components.	**Moderate**: Learning curve due to JavaScript and platform-specific nuances.	**Large**: Extensive community support with active development and frequent updates.	Cross-platform (iOS, Android)
Flutter	Dart	**High**: Compiles to native code, providing near-native performance.	**Low**: Hot reload feature speeds up development. Single codebase for iOS and Android reduces time to market.	**Excellent**: Customizable widgets for pixel-perfect designs.	**Moderate**: Learning curve for Dart language and Flutter's reactive programming model.	**Growing**: Active community with increasing adoption and support.	Cross-platform (iOS, Android)
Xamarin	C#	**High**: Compiles to native code, providing near-native performance.	**Moderate**: Shared codebase reduces development time, but platform-specific adjustments may be required.	**Good**: Native UI with Xamarin. Forms or platform-specific UI with Xamarin.iOS and Xamarin.Android.	**Moderate**: Learning curve for C# and Xamarin APIs.	**Large**: Established community with strong Microsoft support.	Cross-platform (iOS, Android)
Ionic	HTML/CSS/JS	**Moderate**: Performance depends on the performance of the	**Low**: Rapid development with web technologies. Easy deployment to	**Good**: Utilizes web technologies for flexible and responsive UI.	**Low**: Easy to learn for web developers. Limited native features may	**Large**: Active community with extensive documentation and plugins.	Cross-platform (iOS, Android)

		WebView component.	multiple platforms.		require additional plugins.		
NativeScript	JavaScript	**High**: Direct access to native APIs for optimal performance.	**Moderate**: Shared codebase reduces development time. Platform-specific adjustments may be needed.	**Good**: Native UI components for seamless user experience.	**Moderate**: Learning curve due to JavaScript and native platform APIs.	**Moderate**: Supportive community with regular updates and plugins.	Cross-platform (iOS, Android)
Swift (iOS)	Swift	**High**: Native code execution ensures optimal performance on iOS devices.	**High**: Swift's modern language features and native development environment can speed up development but limited to iOS only.	**Excellent**: Leverages native iOS UI components for best user experience.	**High**: Swift can have a steep learning curve for developers new to the language and iOS development.	**Large**: Robust community supported by Apple's ecosystem.	iOS only
Kotlin (Android)	Kotlin	**High**: Kotlin compiles to bytecode and runs on the JVM, providing performance comparable to Java.	**High**: Kotlin's concise syntax and interoperability with Java can accelerate development.	**Excellent**: Utilizes modern Android UI toolkit (Jetpack) for flexible and intuitive UI.	**Moderate**: Kotlin's syntax may require adjustment for developers familiar with Java.	**Large**: Kotlin is officially supported by Google with growing community adoption.	Android only

References:

1. https://theonetechnologies.com/blog/post/flutter-vs-react-native-vs-xamarin-top-cross-platform-mobile-app-development-framework
2. https://surf.dev/mobile-app-development-frameworks/