https://doi.org/10.46335/IJIES.2025.10.6.5 **e-ISSN: 2456-3463**
Vol. 10, No. 6, 2025, PP. 23-27

International Journal of Innovations in Engineering and Science, www.ijies.net

Comparative Analysis of Kotlin, Kotlin Multiplatform, Swift, Flutter, and React Native in Cross-Platform App Development

Dr. Neha Gupta[1], Atharva Gour[2], Nishita Raghvendra[3], Rohit Manna[4]

[1]*Deputy Director, Symbiosis University of Applied Sciences, Indore, India, 453112,* *0000-0003-0052-2653*
[2] *Student, Symbiosis University of Applied Sciences, Indore, India, 453112, 2022btcs013@student.suas.ac.in,*
[3] *Student, Symbiosis University of Applied Sciences, Indore, India, 453112, nishitaraghvendra@gmail.com,*
[4] *Student, Symbiosis University of Applied Sciences, Indore, India, 453112, rohitmanna55@gmail.com*

Email of Corresponding Author: ***neha.gupta@suas.ac.in***

Received on: *28 April, 2025* ***Revised on:*** *02 June,2025* ***Published on:*** *06 June, 2025*

Abstract *–This paper investigates and compares five prominent technologies in the modern app development ecosystem: Kotlin (and its multi-platform extension), Swift, Flutter, and React Native. The study examines their application in mobile, watch, and laptop (desktop) app development with respect to performance, speed, compatibility, learning curve, and current as well as future market demand.* [1] [2] [3] [4] *By analyzing both technical attributes and ecosystem trends, this research aims to provide developers and decision-makers with insights into which framework or language might best suit different project requirements.*

Keywords- Kotlin, multi-platform extension, Swift, Flutter, and React Native

INTRODUCTION

The rapid evolution of app development platforms has resulted in a diverse range of languages and frameworks. Traditionally, native development required separate codebases for different platforms. [5] Today, frameworks such as Kotlin Multiplatform, Flutter, and React Native provide cross-platform capabilities, while languages like Swift (for iOS/macOS) and Kotlin (for Android and beyond) continue to be critical in native environments. [6] [7] This study outlines the advantages and trade-offs of each approach:

- **Kotlin**: A modern programming language known for its expressiveness, safety, and strong support on Android. [8]
- **Kotlin Multiplatform (KMP):** An extension of Kotlin that enables code sharing across platforms without sacrificing native performance.
- **Swift:** Apple's native language designed for iOS, watchOS, and macOS, known for its performance and seamless integration within the Apple ecosystem.
- **Flutter:** Google's UI toolkit that leverages the Dart language to deliver high-performance, visually appealing apps across mobile, web, and desktop.
- **React Native:** A JavaScript-based framework that allows developers to build cross-platform mobile applications with a single codebase.

METHODOLOGY

Our comparative analysis is structured around the following dimensions:

- Platform Domain: Mobile apps, watch apps, and laptop/desktop apps.
- Performance: Runtime speed, memory management, and UI responsiveness.
- Development Speed and Compatibility: Ease of integration, native API support, and cross-platform consistency.

https://doi.org/10.46335/IJIES.2025.10.6.5 e-ISSN: 2456-3463
Vol. 10, No. 6, 2025, PP. 23-27

International Journal of Innovations in Engineering and Science, www.ijies.net

- Learning Curve and Ecosystem Demand: Developer adoption, community support, and projected market trends.

1. Platform Domain Support

Mobile Apps

- Kotlin & Kotlin Multiplatform:
  - Kotlin is highly optimized for Android native development.
  - Kotlin Multi-platform allows sharing business logic across Android and iOS while still enabling native UI implementation.
- Swift:
  - Primarily used for iOS, watchOS, and macOS, offering seamless integration with Apple hardware.
- Flutter:
  - Delivers a consistent UI across Android and iOS using its own rendering engine.
- React Native:
  - Leverages native components to provide a near-native look and feel on both Android and iOS.

Watch Apps

- Kotlin/KMP:
  - Currently less prevalent in watch app development compared to native solutions; Kotlin Multiplatform projects may require additional libraries for watchOS.
- Swift:
  - The dominant language for watchOS apps, offering deep integration with Apple Watch features.
- Flutter & React Native:
  - While both frameworks have growing community efforts, native development (Swift) remains the standard due to performance and OS-specific requirements.

Laptop/Desktop Apps

- Kotlin & Kotlin Multiplatform:
  - Kotlin can target JVM-based desktop applications. Kotlin Multiplatform extends support to desktop environments using shared logic and native wrappers.
- Swift:
  - Swift is extensively used in macOS app development.
- Flutter:
  - Recently expanded support for desktop apps (Windows, macOS, Linux) with promising performance.
- React Native:
  - Primarily focused on mobile; however, community-driven projects (e.g., React Native for Windows/Mac) are emerging but still less mature.

2. Performance, Speed, and Compatibility

Runtime Performance and Speed

- Kotlin (Native):
  - Benefits from direct compilation to JVM or native binaries; offers excellent performance on Android.
- Kotlin Multiplatform:
  - By sharing business logic while maintaining native UI, it strives for near-native performance on each platform.
- Swift:
  - Highly optimized for Apple devices, delivering top performance and efficient memory management.
- Flutter:
  - Uses a high-performance rendering engine (Skia) to ensure smooth animations; however, some overhead exists due to its abstraction layer.
- React Native:
  - Offers acceptable performance by bridging JavaScript with native code; however, complex UIs and heavy computations can sometimes experience latency.

Compatibility

- Kotlin/KMP:
  - Excellent for Android; KMP is evolving to provide robust cross-platform compatibility.
- Swift:

https://doi.org/10.46335/IJIES.2025.10.6.5 e-ISSN: 2456-3463
Vol. 10, No. 6, 2025, PP. 23-27

- Offers tight integration within the Apple ecosystem.
- Flutter:
  - Maintains consistent UI and behavior across supported platforms.
- React Native:
  - Leverages native UI components but sometimes suffers from compatibility issues with newer OS versions or custom native modules.

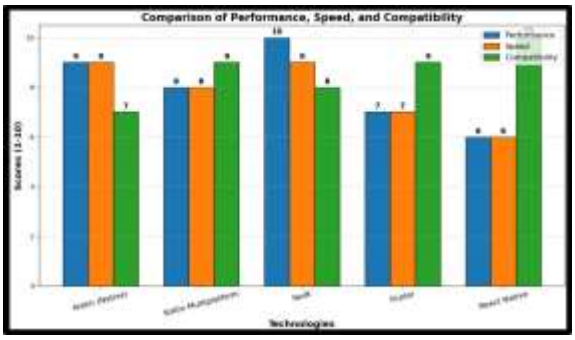


Figure 1: Graphical Representation of Performance, Speed and Compatibility of Languages/ Platform.

3. Learning Curve and Market Demand

Learning Speed

- Kotlin:
  - Simple syntax and interoperability with Java make it relatively easy for developers with an Android background.
- Kotlin Multiplatform:
  - Introduces additional complexity in managing shared code and platform-specific modules.
- Swift:
  - Designed to be modern and safe, but its unique paradigms require time to master for developers coming from other languages.
- Flutter (Dart):

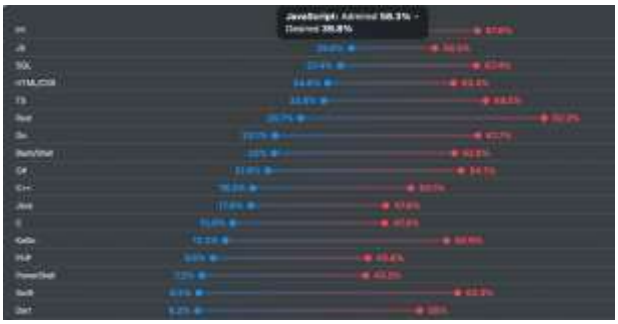
Figure 2: Graphical Representation of Learning Curve, Market Trend and Future Scope of Languages/ Platform.

  - Dart is easy to learn, and Flutter's "hot reload" speeds up the development cycle.
- React Native (JavaScript/TypeScript):
  - JavaScript's ubiquity makes it accessible; however, managing the bridge between JS and native code can add complexity.

Market Demand and Future Trends

- Kotlin:
  - Strong demand in the Android market; expected to remain essential as Android evolves.
- Kotlin Multiplatform:
  - Growing interest as companies look to reduce code redundancy; future market penetration depends on ecosystem maturation.
- Swift:
  - Continues to dominate iOS and macOS markets with high demand in the Apple developer community.
- Flutter:
  - Rapidly growing in popularity due to its cross-platform capabilities and Google's backing; adoption is rising among startups and established companies alike.
- React Native:
  - Despite some performance trade-offs, its JavaScript base and mature community ensure its continued demand, particularly for rapid prototyping and projects with web–mobile convergence.

https://doi.org/10.46335/IJIES.2025.10.6.5 e-ISSN: 2456-3463
Vol. 10, No. 6, 2025, PP. 23-27



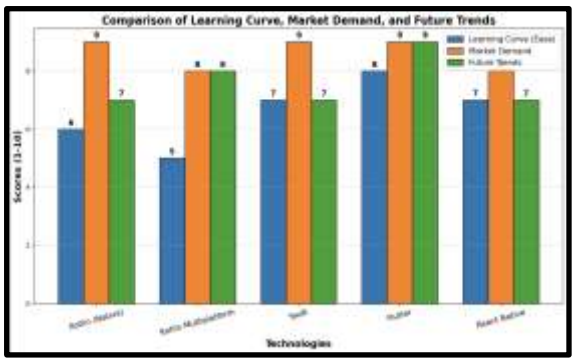


Figure 3: Demand and Admired Languages

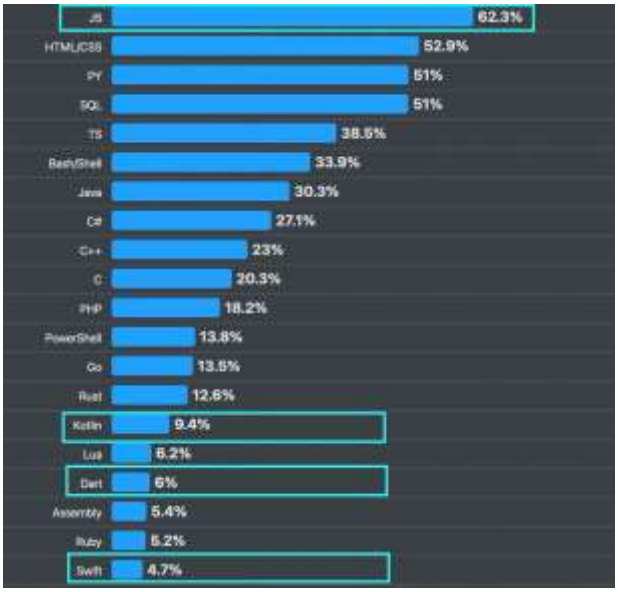


Figure 4: Current Usage of the Programming Language

RESULT & DISCUSSION

The comparative study reveals several key points:

- **Native vs. Cross-Platform Trade-offs:** Native languages like Swift and Kotlin offer unparalleled performance and integration, especially in domains like watch and desktop apps where platform-specific features are critical. Cross-platform frameworks like Flutter and React Native provide significant advantages in code reusability and rapid development.

- **Learning and Adoption:** Flutter and React Native, by virtue of their modern tooling and simplified development cycles, attract new developers. Kotlin and Swift, while requiring a deeper understanding of native paradigms, offer long-term benefits in performance and platform integration. Kotlin Multiplatform, although promising, currently faces challenges in developer education and tooling support.

- **Market Demand and Future Trends:** Market trends suggest that while native development remains essential for high-performance applications, the demand for cross-platform solutions is rising. Flutter's strong performance and appealing developer experience position it well for future growth, whereas Kotlin Multiplatform is emerging as a robust solution for companies invested in both Android and iOS ecosystems.

Table 1: Compatibility of all Platform and Languages and their learning curve and Market Demand

Criterion	Kotin	KMP	Swift	Flutter	ReactNative
Mobile App	Excellent	Very Good	Excellent	Very Good	Good
Watch App	Limited	Limited	Excellent	Emerging	Emerging
Desktop App	Good	Emerging	Excellent	Good	Limited
Performance	High	High	Very High	High	Moderate=
Learning Curve	Moderate	Steeper	Moderate	Easy	Moderate
Market Demand (Now)	High	Growing	High	Rapidly Growing	High
Market Demand (Future)	Steady	Promising	Steady	Promising	Steady

https://doi.org/10.46335/IJIES.2025.10.6.5 **e-ISSN: 2456-3463**
Vol. 10, No. 6, 2025, PP. 23-27

International Journal of Innovations in Engineering and Science, www.ijies.net

Table 2: Comparison of the Languages and Platforms for Performance, Tools and IDEs and Community Support

Aspect	Kotlin/KMP	Swift	Flutter	React Native
Development	Fast	Very Fast	Fast	Fast
Tools & IDEs	Android Studio	Xcode	VS Code	VS Code
Community	Strong/Growing	Robust	Rapid Growing	Extensive

CONCLUSION

This study has provided a detailed comparison of Kotlin (native), Kotlin Multiplatform, Swift, Flutter, and React Native across multiple domains and criteria. While native solutions (Swift and Kotlin) deliver superior performance and system integration, cross-platform frameworks (Flutter and React Native) are catching up in terms of development speed and market appeal. Kotlin Multiplatform offers an intriguing middle ground by enabling code sharing without completely sacrificing native quality. Ultimately, the choice depends on project requirements, target platforms, and long-term maintenance considerations.

FUTURE SCOPE

- **Increased Adoption of Cross-Platform Frameworks**:
  With companies aiming to reduce maintenance costs and streamline development, frameworks like Flutter and Kotlin Multiplatform may see broader adoption, particularly as tooling and community support improve.
- **Enhanced Native Capabilities**:
  Swift and Kotlin will continue to evolve, integrating more features and improving performance. Their role will remain pivotal in applications that demand deep system integration, such as wearables and desktop apps.
- **Ecosystem Convergence**:
  Hybrid approaches that leverage the strengths of native and cross-platform paradigms will likely emerge, offering developers flexibility to optimize for both performance and speed.

REFERENCES

[1] "Kotlin Programming Language," [Online]. Available: https://kotlinlang.org/docs/home.html. [Accessed 01 03 2025].

[2] "Apple Inc.," [Online]. Available: https://developer.apple.com/swift/. [Accessed 20 02 2025].

[3] "Flutter Team," [Online]. Available: https://flutter.dev/docs. [Accessed 10 02 2025].

[4] "Meta Platforms, Inc.," [Online]. Available: https://reactnative.dev/docs/getting-started. [Accessed 24 01 2025].

[5] "Google Trends," Google, [Online]. Available: https://trends.google.com/trends. [Accessed 23 01 2025].

[6] "Flutter vs React Native vs Swift/Kotlin in 5 Minutes," YouTube, [Online]. Available: https://www.youtube.com/watch?v=8yImX_v8f-k. [Accessed 12 02 2025].

[7] M. M. T.F. Bernardes, "Cross-platform mobile development approaches: a systematic review," IEEE Latin American Transactions, vol. 14, no. 4, pp. 1892-1898, 2016.

[8] "React Native for Windows," Microsoft, [Online]. Available: https://github.com/microsoft/react-native-windows. [Accessed 10 02 2025].