iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VIII **Month of publication:** Aug 2023

DOI: https://doi.org/10.22214/ijraset.2023.55187

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VIII Aug 2023- Available at www.ijraset.com

Comparative Study of Android Native vs Cross-Platform Mobile Application Development

Aditya Desai
Master of Computer Application, KLS Gogte Institute of Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India

Abstract: ***This paper provides a comprehensive comparison between two prominent approaches to Android app development: native and cross-platform. It thoroughly analyzes various aspects relevant to enterprise and general mobile application development. The study examines project size, developer work comfort, popularity, performance, and documentation. Regarding project size, both native and cross-platform approaches are evaluated to determine their suitability for small to large-scale projects. The study also delves into developers' work comfort, investigating the ease of use and convenience associated with each approach. Popularity and adoption rates among developers are analyzed to understand the industry's prevailing trends. Moreover, the research examines the performance capabilities of both approaches, considering factors like speed and responsiveness. Lastly, the availability and comprehensiveness of documentation for each tech stack are assessed. After analyzing the paper, readers can gain a clear overview of the strengths and weaknesses of native and cross-platform development. This enables them to make informed decisions on which tech stack to learn and use when building Android applications, whether it's for enterprise purposes or to simply pursue mobile application development skills in general.***
Keywords: native development, cross-platform development, React Native, Kotlin, Flutter.

I. INTRODUCTION

Android-based smartphones are in particular high demand today because they are so frequently utilized for daily communication. Numerous applications have been created to enhance people's lives and offer countless hours of entertainment. The Android operating system has experienced enormous growth in the smartphone industry.

Application development for the Android operating system, which was designed primarily for smartphones, is made simpler by the use of the Google-created Android Studio tool. Conversely, cross-platform mobile development is the process of creating mobile applications that can function with a single codebase across numerous platforms. Choosing which cross-platform solution to use to achieve certain user or company goals has become a big challenge given the plethora of options accessible online. Cross-platform tools offer a simple answer to the issue of platform fragmentation, despite the fact that they are still in development and have their drawbacks. It is feasible to create cross-platform apps using Model-Driven Engineering (MDE) principles, as well as to streamline the development procedure and expand the user base by applying domain concepts. It would be advantageous to expand this survey in the future by incorporating a wider variety of studied tools and factors.

II. LITERATURE REVIEW

In the present mobile application market, it can be challenging for developers to decide whether to build for iOS or Android, the two most widely used mobile operating systems. It can take a lot of time and effort for developers to become proficient in both Kotlin for Android and Swift for iOS, which are two different programming languages that must be used when writing code for separate applications for each platform.

The solution to this issue is cross-platform development. Because cross-platform development enables developers to write code once and distribute it across multiple platforms, it reduces the time and effort needed for development. Instead of maintaining separate codebases for Android and iOS, developers can use frameworks like React native, created by Facebook, and Flutter, created by Google.

III. COMPARATIVE ANALYSIS OF CROSS-PLATFORM AND NATIVE DEVELOPMENT

A. *Project Sizes*

Small IT businesses with up to 10 employees frequently choose cross-platform development. They often work on small and medium-sized projects that last three months to a year on average. Cross-platform technologies are an appropriate option for these projects because they typically do not have strict technological requirements. A decision like that is advantageous to both the client and the business.

By using cross-platform technologies, the number of required application developers can be reduced, often involving only one person. This streamlined approach makes it more convenient for the customer to communicate with a single developer instead of explaining to multiple developers how each screen should function. People have different perspectives, and they may interpret design and technical aspects differently, resulting in varying appearances and work speeds. Achieving a consistent result may take more time.

For small companies, adopting a cross-platform approach offers several advantages, including lower costs due to the reduced need for developers.

Creating separate applications for different platforms sequentially—first for one platform and then for another—is not a practical option. It would require significant time and monetary investments, making it less profitable for both parties involved.

Regarding quality, simple applications such as social networks or online shops that do not demand extensive use of smartphone resources are unlikely to require low-level interventions. Although React Native allows writing native modules, for such applications, the quality and performance will not be affected in any significant way.

For clients initiating their own business with an existing website, cross-platform application development proves cheaper and faster compared to native development during project initiation. However, this primarily applies to small projects that do not require extensive support or further development. In the case of projects with long development cycles, continuous improvement, and expanding functionality, native development outperforms cross-platform development.

In large-scale projects, significant emphasis is placed on design, including object animations. However, practical experience has shown that implementing non-trivial designs for two operating systems often leads to numerous bugs, increasing the time required to resolve errors. The user interface may display differently, necessitating additional adaptation time. Issues frequently arise with animations, clicks, and scrolling, potentially causing the application to freeze. While the user interface is developed in HTML, achieving native platform performance may take months. In such cases, the native development languages for Android and iOS, such as Java, Kotlin, Objective-C, and Swift, are more effective tools compared to cross-platform technology APIs.

Developing an interface that is intuitive for both iOS and Android users is crucial. Failure to do so will result in an application built according to iOS Human Interface OS Guidelines being inconvenient for Android users, requiring additional time to enhance the user experience.

Long-term projects are not primarily focused on resource savings. Instead, customers prioritize performance, quality, and reliability. For large-scale projects, native development is the optimal solution. Creating a high-quality application in this context requires developers with extensive experience and deep knowledge in Android (Java or Kotlin) and iOS (Objective-C and Swift) development. Compared to cross-platform technologies, native development may involve a larger number of developers, whose expertise commands higher costs. However, given the substantial investment typically associated with large projects, these expenses are justified. Native development ensures application speed, as the compiled code is optimized for the specific platform, benefiting from full hardware support and utilizing multithreading for complex tasks. When updates are released for the programming language or hardware functionality, new features become immediately available, without the need to wait for implementation and adaptation in cross-platform development technologies.

The most significant advantage of native development is its absence of restrictions, making it an ideal choice for the development of extensive projects. Native development leverages system software functions, such as the camera, microphone, geolocator, accelerometer, calendar, and media, with maximum flexibility. Furthermore, native applications are designed to consume memory and device battery resources sparingly.

B. *Work Comfort*

When selecting a technology, it's critical to take the developer into account in addition to the project needs. The ease of learning is an important factor to take into account. The quantity of online courses, tutorials, and articles generally reflects how well-liked a technology is. Native development has a benefit in this sense. Consider Kotlin as the official language for creating Android applications, for instance.

Google's endorsement of Kotlin stems from observing its rise in popularity in recent years. Google openly describes Kotlin as an impressive, concise, powerful, and enjoyable language to work with. Kotlin offers increased performance, with program code being, on average, 40% shorter compared to other languages. Moreover, Kotlin helps in avoiding certain errors in code. One significant factor contributing to Kotlin's popularity with Google is its compatibility with Java, which is widely used in Android app development. More Kotlin libraries, tools, and instructional resources are expected to be created in the future. This will aid in the language's continuous development and make it simpler to discover answers to any potential issues. Because Google is a prominent player in the IT sector and actively promotes and popularizes Kotlin, many other businesses are influenced to do the same. As a result, several organizations are releasing Kotlin training classes and other resources. Additionally, there are more and more posts about Kotlin on websites like Habr and Stack Overflow, where this technology is being used to address a wide range of issues.

This quick rise in popularity suggests that Kotlin greatly lowers the entry barrier for newcomers, making it simpler to learn thanks to the wealth of available learning resources. On the other hand, Google has also unveiled Flutter, a very promising cross-platform development framework. Unlike Xamarin and React Native, Flutter follows a distinct methodology. Flutter renders each element independently by creating a window on the screen of the smartphone rather than turning the source code into native code that is run by the platform. Flutter uses Dart, a language that Google developed as a better alternative to JavaScript. However, as of 2020, Flutter was still considered relatively new and in its early stages. The lack of training resources for Flutter at the time, as compared to more established technologies, was one disadvantage. There weren't as many learning materials for Flutter as would have been ideal, and there weren't as many supplementary libraries. The learning curve involved in implementing a certain technology is another factor to take into account. In other terms, it refers to the amount of time and effort needed to learn enough to carry out a straightforward project. Cross-platform development appears to be advantageous from this angle. Consider the Facebook product React Native as an illustration. A JavaScript framework called React Native enables programmers to create natively displayed Android and iOS apps. It is based on Facebook's React JS toolkit, which is frequently employed in web development to build user interfaces. As a result, programmers with web development experience—especially with ReactJS—can easily switch to developing mobile apps using React Native. ReactJS and React Native share 80% of their code bases, making it incredibly quick and easy to develop straightforward mobile applications like a business card app. Moreover, there is a platform called Expo that significantly simplifies working with React Native, especially for beginners. Expo provides a comprehensive set of tools, including pre-configured Android Studio and XCode configurations, management of Apple and Google certificates, and push notifications. The Expo development team has also created a fantastic tool called Snack, which allows developers to write and test code directly in the browser without any prior installation or setup of development environments. As a result, the barrier to entry for React Native becomes very low. Additionally, Facebook's documentation for React Native is well-documented, making the transition from web development to mobile development a breeze. It's worth noting that the documentation caters to both developers experienced in native development, recommending the use of React Native CLI, as well as newcomers to mobile development, who are encouraged to use Expo CLI. Overall, the combination of React Native's familiarity for web developers, the Expo platform with its user-friendly tools, and the comprehensive documentation provided by Facebook make React Native an appealing choice for those new to mobile development.

C. Popularity

When it comes to top 1000 apps then clearly Kotlin is the most preferred language for android development.It’s also estimated that 80% of the top 1,000 Android apps used Kotlin code, including over 60 Google apps. Moreover, PYPL ranked Kotlin as the 12th most popular programming language in the world. Besides Android, its multi-platform capabilities allow programmers to share code, logic, and data across several platforms such as iOS, Web, and more.

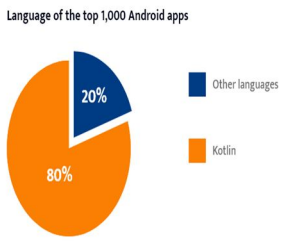


Fig 1: Language used by top 1000 android apps

Companies that use Kotlin include Pinterest, Netflix, Slack, Trello, and Airbnb.
Flutter is becoming one of the most well-liked cross-platform frameworks.

1) *Some Benefits of Utilizing Flutter Include*

a) *Free and Open Source:* An open-source framework called Flutter enables extensive customization. This indicates that you will have free access to the necessary original code and documentation. The documentation for Flutter is simple to use and navigate.

b) *Simpleness of usage and Learning:* The Flutter/Dart combination is simple to use and quick to learn for coders. Dart and Flutter are easily learnable by programmers who have experience with Swift, React Native, and C-like syntax languages within a few weeks. The Kotlin-like Flutter framework guarantees a quick installation that doesn't take longer than 30 minutes and enables you to write a lot less code to create fully working apps.

c) *The Ability to use Hot Reloads:* Developers may view the output in real time thanks to Flutter. Therefore, while an application is running, software engineers can amend their code, make changes to the front-end or back-end, and immediately see the changes. Therefore, when you develop UI, add features, repair issues, etc., there is no need to completely reload the program. One of the most significant benefits of cross-platform programming over native development is this.

d) *Faster Time to Market:* A powerful and affordable platform for creating apps is called Flutter. It facilitates a speedier product delivery and shortens the time to market because to the shared codebase it offers.

The following companies employ Flutter as part of their tech stack: BMW, eBay, Square, Groupon, CapitalOne, etc.

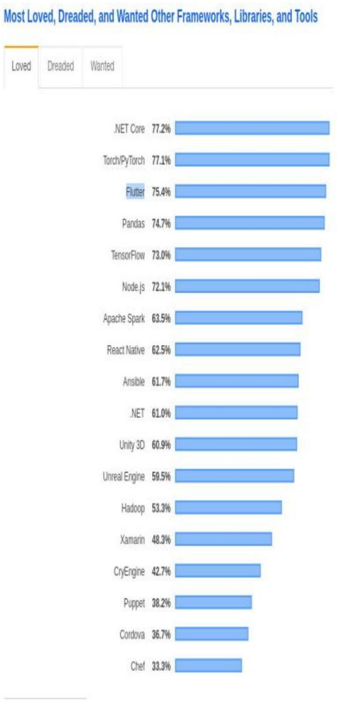


Fig 2: Most loved cross platform languages in 2023

2) *Talking about React Native, Some of the Pros Are*

a) *Live-tracking:* Working with this framework will be enjoyable for you, especially since it has a function known as "Hot Reload." By enabling an automatic refresh of the application and its interface, this feature enables programmers to observe real-time changes in the application immediately after writing the code.

b) *Third-party Plugins:* Software developers can use third-party plugins with React Native to achieve a higher level of flexibility, functionality, and customisation than they can with the majority of languages and frameworks (including Flutter), which forbid the usage of plugins. Additionally, developers regard it as React Native's most impressive feature.

c) *Modular Architecture:* React Native's modular programming enables developers to separate different functionalities into blocks or modules. Every block contains all the data needed to execute a certain function of the app. On top of that, the framework permits flexible and optimized coding flow as part of its declarative coding feature.

d) *Higher Performance:* Another undeniable benefit of React Native is its high performance and cross-platform feature. You can have one team that will build products for Android and iOS with just one codebase. Infrastructure and feature development are performed simultaneously. Plus, the framework highlights errors on the move, provides pre-built components, decreases debugging time, and allows 90% reusability of the code, all contributing to increased performance and shorter development time.

e) *More Experienced Developers:* React developers, whose numbers are constantly increasing in the market, may pick up React Native quickly and effectively. Additionally, they have the option to build code using native frameworks and then include it into a hybrid React Native codebase. In conclusion, it might be a terrific career path for developers and a viable choice for product owners.

Instagram, Salesforce, Tesla, Skype, Discord, and other businesses employ React Native as part of their software stack.

D. Performance

There are several factors while comparing kotlin, react native and flutter:

When comparing the performance of Kotlin, React Native, and Flutter, it's important to consider several factors:

1) *Native vs. Cross-Platform:* Kotlin is a native language for Android development, which means it can fully leverage the underlying platform's capabilities and optimizations. This often results in better performance compared to cross-platform frameworks like React Native and Flutter, which introduce an additional layer of abstraction.
2) *Rendering Approach:* React Native uses a bridge to communicate between JavaScript and native components, which can introduce slight performance overhead compared to Kotlin and Flutter. Flutter, on the other hand, renders its own UI elements directly on the screen, bypassing the need for a bridge, which can lead to excellent performance.
3) *Just-In-Time (JIT) vs. Ahead-of-Time (AOT) Compilation:* Both Kotlin and Flutter make advantage of ahead-of-time (AOT) compilation, which turns the program's source code into machine code before it is put into use. Performance is enhanced and starting times are reduced as a result. The just-in-time (JIT) compilation method is used by React Native, which may result in somewhat longer startup times and potential runtime performance penalties.
4) *Animation and UI Interactions:* Particularly noticeable is Flutter's performance in terms of animations and UI interactions. Flutter uses its own rendering engine and renders the user interface (UI) by default at 60 frames per second (FPS), producing responsive and fluid animations. React Native may produce animations with good performance, but because to its bridge-based connection with native components, it may occasionally have frame rate decreases.
5) *Optimization Opportunities:* Kotlin, being a native language, allows developers to fine-tune performance optimizations specific to the Android platform. This includes leveraging platform-specific features and libraries. Flutter, with its direct rendering approach and Dart's efficient runtime, provides good performance out of the box. React Native's performance optimizations heavily depend on the underlying native components and the JavaScript engine it uses.
6) *Application Complexity:* Kotlin, React Native, and Flutter's performance disparities could be more noticeable in complex apps with computationally demanding processes. While React Native and Flutter may occasionally encounter performance snags as a result of the bridge or rendering technique, Kotlin's native capabilities allow for quick processing.

It's crucial to remember that performance can change based on the particular use case, the caliber of the code, and the optimization strategies used by developers. Insights into the performance characteristics of each technology can be gained more precisely by conducting performance tests and benchmarks that are suited to the project needs.

Overall, while Kotlin's native nature often leads to superior performance, both Flutter and React Native have made significant strides in optimizing their performance. Flutter, with its direct rendering and efficient rendering engine, generally offers excellent performance for animations and UI interactions. React Native can deliver satisfactory performance, but slight performance overhead may be observed due to its bridge-based communication.

E. Documentation

It is evident by comparing the level of documentation for Flutter, React Native, and Kotlin that each technology provides useful resources for developers.

Google-supported Flutter offers thorough and organized documentation. Developers may readily obtain detailed explanations and examples to comprehend and successfully implement Flutter's features thanks to frequent updates and a wide variety of subjects covered. Google has been actively involved in maintaining and enhancing the framework, which is reflected in the documentation.

Facebook's React Native also comes with extensive documentation. While React Native documentation may not have as much official support as Flutter, it is frequently updated and offers comprehensive explanations. Developers can effectively use the framework by locating installation instructions, fundamental ideas, APIs, and platform-specific details.

Kotlin benefits from the substantial documentation that JetBrains and the Kotlin community provide as the preferred language for Android development. The language's syntax, common library functions, Android-specific subjects, and recommended practices are all covered in the documentation. Kotlin's concepts are simpler for developers to understand and incorporate into their applications thanks to the availability of explicit explanations and examples.

All three technologies also have active communities that provide extra learning materials. These include articles created by communities, blogs, forums, and tutorials. Users of Flutter, React Native, and Kotlin can gain knowledge from these resources, which include insightful information, advice, and practical use examples. These extra learning resources improve the overall documentation experience and give developers access to a wider variety of tools for developing their knowledge and abilities.

Overall, Flutter, React Native, and Kotlin offer solid documentation that meets the needs of developers. Whether it's Google's involvement in Flutter, Facebook's support for React Native, or the dedicated efforts of JetBrains and the Kotlin community, developers have access to clear explanations, examples, and supplementary resources to support their learning and implementation processes.

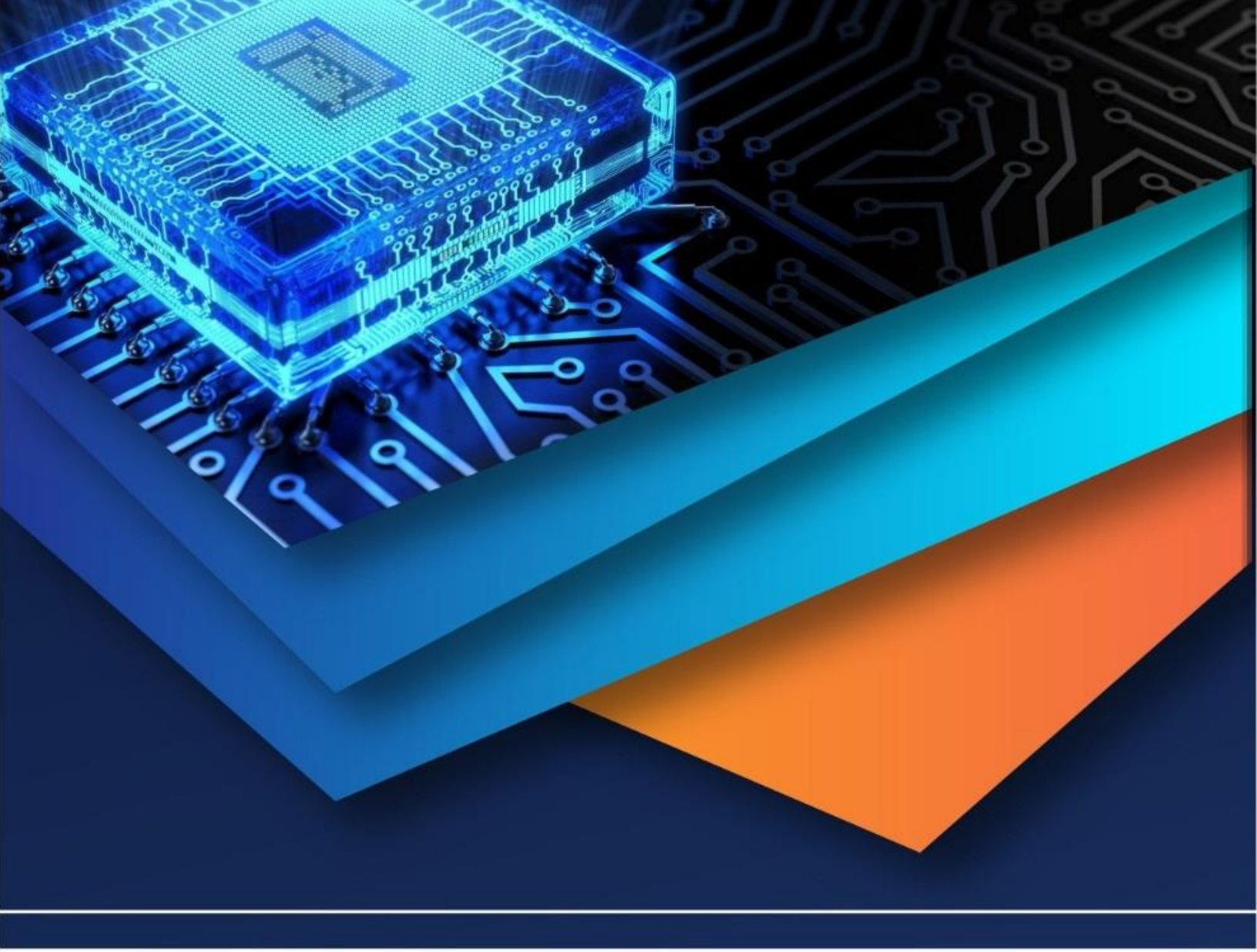
IV. CONCLUSION

When it comes for building android application for enterprise it actually depends on the requirement of enterprise as well as infrastructure. When it comes to learning android development, I believe starting with Native android development is better because it will make us familiar with android specific concepts such as activities, fragments, view model, layout etc only prerequisite is one must have knowledge of at-least one programming language.

On the other hand if someone is familiar with web development especially in framework such as React Js then learning React Native will be piece of cake of him because 80% of the concepts are same. Also with the help of tools such as Expo building mobile application is very easy using React native.

REFERENCES

[1] A Students' Perspective of Native and Cross-Platform approaches for Mobile Application Development by Paulo Meirelles and Carla S. R. Aguiar.
[2] Cross platform development vs native development by Nikita A. Shevtsiv and Andrii M. Striuk.
[3] CROSS PLATFORM APP A COMPARATIVE STUDY by Paulo R. M. de Andrade and Adriano B. Albuquerque.
[4] Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps by Neha Verma1, Sarita Kansal and Huned Malvi.
[5] Research on the Development Technology of Cross Platform Hybrid Mobile Application Based on HTML5 by Qiao Yan, Guyu Hu1 , Guiqiang Ni1, Jinsong Jiang and Junxian Long.
[6] Comparison of Native, Cross-Platform and Hyper Mobile Development Tools Approaches for iOS and Android Mobile Applications by Shan Jiang.
[7] Review On Cross-Platform Mobile Application Development by Pranjali Ravindra Hiwale.
[8] Native Vs Cross-Platform Development: How To Choose By Anastasiya Marchuk.
[9] Mobile Application Development Using Flutter Development: A Review Study by Akash Dhingra.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY