www.ijarp.com ISSN 2456-9992 Page: 01-14

International Journal Advanced Research Publications

Review Article
Volume: 02
Issue: 08

JAVA-BASED ANDROID APPLICATION DEVELOPMENT: A SYSTEMATIC REVIEW OF DEVELOPMENT TECHNIQUES, CHALLENGES AND EMERGING TRENDS

***Prof. Indu L. Gadappa**

Master of Computer application, Ballarpur Institute of Technology Gondwana University.

Article Received: 11 July 2026, Article Revised: 31 July 2026, Published on: 20 August 2026

***Corresponding Author: Prof. Indu L. Gadappa**

Master of Computer application, Ballarpur Institute of Technology Gondwana University.

Doi: https://doi-doi.org/101555/ijarp.2017

ABSTRACT

Android has emerged as one of the most widely used platforms for mobile application development, resulting in continuous evolution of programming languages, development tools, software architectures, testing approaches, and security practices. Java has historically been one of the principal programming languages used for Android application development because of its mature ecosystem, object-oriented programming features, extensive libraries, large developer community, and compatibility with existing Android applications. In recent years, however, the increasing adoption of Kotlin has significantly influenced the Android development environment and has raised questions regarding the continuing relevance, advantages, limitations, and future role of Java. This paper presents a systematic literature review of Java-based Android application development with the objective of examining major development techniques, tools, frameworks, architectures, challenges, and emerging trends reported in existing studies. Relevant literature concerning Java-based Android development, Java and Kotlin comparisons, application performance, maintainability, security, programming practices, and development technologies is analyzed and synthesized. The review indicates that Java continues to provide important advantages through its mature ecosystem, extensive existing code base, established development practices, and interoperability with modern Android technologies. At the same time, existing comparative studies indicate that Kotlin can provide more concise source code and advantages in addressing certain programming pitfalls. The literature also highlights continuing challenges related to application security, performance, maintainability, software evolution, language migration, and compatibility with changing Android technologies. Based on the reviewed

literature, the Java-to-Kotlin transition, secure Android application development, automated software analysis, modern application architectures, testing, and development automation represent important areas for continued research. This review provides a consolidated understanding of the evolution and continuing relevance of Java in Android application development and identifies research opportunities for improving the development, maintenance, performance, and security of Android applications.

KEYWORDS: Java, Android, Android Application Development, Kotlin, Software Engineering, Mobile Applications

1. INTRODUCTION

Mobile applications have become an important part of modern software systems, supporting communication, education, entertainment, business, healthcare, banking, and many other domains. Among mobile operating systems, Android has developed into a major platform for application development and has supported a large ecosystem of applications and developers. The development of Android applications has also evolved over time through changes in programming languages, development tools, software architectures, testing techniques, and security practices.

Java has played an important role in the historical development of Android applications. Its object-oriented programming model, mature development ecosystem, extensive libraries, and large developer community made it a widely used language for Android application development. Java-based Android development also benefited from the availability of established development tools and a large amount of existing source code. Consequently, a substantial body of Android applications and development knowledge has been built around Java.

However, the Android development environment has changed significantly with the increasing adoption of Kotlin. Java and Kotlin can coexist within the same Android application, making gradual migration possible. Empirical research comparing the two languages has examined factors such as maintainability, code size, development effectiveness, and programming pitfalls. Ardito et al. reported that Kotlin produced more concise code and showed advantages in avoiding certain common programming problems, while Java received better perceptions regarding IDE support. Their study also found no significant difference in maintainability between the two languages.

[1]

The transition from Java to Kotlin has also introduced new research questions concerning software security. Mazuera-Rozo et al. examined security weaknesses in Java and Kotlin Android applications by analyzing 681 commits and surveying 43 Android developers. Their work demonstrates that security remains an important concern across both languages and highlights the need for continued investigation of security weaknesses in Android applications during the transition between programming languages.**[2]**

Although individual studies have investigated specific aspects of Android development, such as programming-language comparison, security, maintainability, and development effectiveness, these findings are distributed across different research areas. A consolidated examination of Java-based Android development is therefore useful for understanding its development techniques, supporting technologies, challenges, and changing role within the modern Android ecosystem.

Therefore, this study presents a systematic review of Java-based Android application development. The review focuses on four major dimensions: development techniques, tools and technologies, challenges and limitations, and emerging trends and research gaps. The objective is not to argue that Java is superior to Kotlin, but rather to examine the existing evidence and determine the continuing relevance and challenges of Java in Android application development.

The following research questions guide the study:

RQ1: What development techniques are commonly used in Java-based Android application development?

RQ2: What tools, frameworks, architectures, and technologies support Java-based Android application development?

RQ3: What challenges and limitations are reported in Java-based Android application development?

RQ4: What emerging trends and research gaps are associated with Java-based Android development?

2. Research Methodology

This study adopts a systematic literature review approach to identify, analyze, and synthesize existing research related to Java-based Android application development. The review focuses on development techniques, tools and technologies, challenges, and emerging trends

associated with Java-based Android applications. A structured search and screening process was followed to improve the relevance and consistency of the selected literature.

2.1 Research Questions

The review was guided by the following research questions:

RQ1: What development techniques are commonly used in Java-based Android application development?

RQ2: What tools, frameworks, architectures, and technologies support Java-based Android application development?

RQ3: What challenges and limitations are reported in Java-based Android application development?

RQ4: What emerging trends and research gaps are associated with Java-based Android development?

2.2 Literature Search Strategy

Relevant studies were identified through scholarly digital libraries and academic search platforms, with particular emphasis on IEEE Xplore, ScienceDirect, and Google Scholar. These sources were selected because they provide access to research in software engineering, mobile application development, programming languages, and information technology.

The search terms were developed according to the research questions. The major keywords included “Java”, “Android”, “Android application development”, “Java-based Android development”, “Android development techniques”, “Android security”, “Java Kotlin Android”, “Android architecture”, “Android performance”, and “Android application challenges”. Boolean combinations of these keywords were also considered to obtain relevant studies.

2.3 Inclusion Criteria

Studies were considered for inclusion when they satisfied the following criteria:

1. The study was related to Android application development.
2. The study discussed Java, Kotlin, or Java-based Android development in a meaningful manner.
3. The study addressed development techniques, tools, architectures, performance, security, maintainability, programming practices, or related challenges.
4. The study was published in a recognized academic publication or scholarly digital library.

5. The study was available in English.

Studies published primarily between 2015 and 2026 were considered to capture both established Java-based development practices and recent developments in the Android ecosystem.

2.4 Exclusion Criteria

Studies were excluded when:

1. They were unrelated to Android application development.
2. They did not provide meaningful information relevant to the research questions.
3. They were duplicate publications.
4. They were non-scholarly web content, advertisements, or general technical articles.
5. The full text or sufficient bibliographic information was unavailable.
6. The study focused exclusively on unrelated mobile platforms or programming technologies.

2.5 Data Extraction and Analysis

For each selected study, relevant information was extracted and organized according to predefined categories. These categories included publication year, programming language, development technique, tools and technologies, application area, evaluation approach, reported advantages, reported challenges, and major findings.

The extracted information was subsequently mapped to the four research questions. Studies related to development techniques and implementation practices were analyzed under RQ1, while studies discussing tools, frameworks, architectures, and supporting technologies were considered under RQ2. Reported technical, security, performance, and maintenance challenges were analyzed under RQ3. Finally, studies addressing Java-to-Kotlin migration, modern Android development practices, automation, security analysis, and other evolving technologies were considered under RQ4.

The findings were synthesized qualitatively to identify common themes, differences between studies, continuing challenges, and potential research gaps in Java-based Android application development.

3. Literature Review and Results

The reviewed literature demonstrates that Java has played a significant role in the development of Android applications and has contributed to the evolution of Android

software engineering practices. The studies examined in this review can broadly be categorized into four areas: Android development techniques, development tools and technologies, challenges associated with Android application development, and emerging trends in programming languages and development practices.

3.1 Development Techniques in Java-Based Android Application Development

Java-based Android development traditionally relies on object-oriented programming principles and the Android Software Development Kit (SDK). Android applications are generally organized into components such as activities, services, broadcast receivers, and content providers. These components communicate through intents and provide the fundamental structure required for developing Android applications.

Java has supported a wide range of Android application development practices, including user-interface development, database management, network communication, background processing, application testing, and integration with external services. The availability of extensive Java libraries and development resources has contributed to its continued use in Android projects.

Software engineering practices such as modular design, object-oriented programming, reusable components, exception handling, and testing have also been applied to improve the maintainability and reliability of Android applications. Java-based applications can additionally integrate with web services and remote databases, allowing Android applications to support data-driven functionality.

Comparative research has examined Java and Kotlin in practical Android development tasks. Ardito et al. investigated the effectiveness of Kotlin and Java in Android application development and found that Kotlin can provide more concise code, while Java continues to provide advantages related to developers' perceptions of development tools. Their findings indicate that programming-language selection can influence development characteristics without necessarily making one language universally superior to the other. [1]

3.2 Tools, Frameworks and Technologies

The Android development ecosystem has traditionally provided strong support for Java through the Android SDK and integrated development environments. Android Studio provides facilities for source-code development, debugging, application building, testing, and performance analysis. Java-based Android applications can also use libraries and APIs for database access, networking, multimedia, security, and communication with external services.

Application architecture has also evolved from relatively simple activity-based applications toward more structured architectural approaches. Separation of user-interface components, application logic, and data management can improve maintainability and facilitate testing. The use of architectural patterns and reusable components has therefore become an important aspect of Android software development.

Android applications frequently communicate with remote servers through web services and APIs. Java-based applications can implement networking functionality and exchange data using commonly used data formats and communication mechanisms. Database technologies such as SQLite have also traditionally supported local data storage within Android applications.

The development ecosystem has further expanded through the coexistence of Java and Kotlin. Since the two languages can interoperate, existing Java-based Android applications can gradually incorporate Kotlin components rather than requiring complete redevelopment. This interoperability has contributed to the gradual evolution of existing Android applications.

3.3 Challenges in Java-Based Android Development

Although Java provides a mature development environment, the literature identifies several challenges associated with Android application development.

One important challenge is application security. Android applications may contain vulnerabilities resulting from inappropriate handling of data, insecure communication, improper resource management, or weaknesses introduced through third-party components. Research examining Java and Kotlin Android applications has demonstrated that security weaknesses remain an important concern across both programming languages.

Performance is another important consideration. Android applications operate on devices with different hardware configurations, memory capacities, processors, and operating-system versions. Developers therefore need to consider memory consumption, CPU utilization, application response time, network usage, and application package size.

Maintainability is also an important challenge. Android applications frequently evolve through changes in requirements, operating-system versions, libraries, and application programming interfaces. Large and complex code bases can increase maintenance effort. Java's comparatively verbose syntax can also result in larger amounts of source code for implementing equivalent functionality when compared with Kotlin.

Another challenge is the transition from Java to Kotlin. While Kotlin provides several modern language features, organizations maintaining existing Java applications may face

migration costs, learning requirements, compatibility considerations, and changes to development and analysis tools.

3.4 Java and Kotlin Comparison

Several studies have directly compared Java and Kotlin for Android development. The comparison is important because Kotlin has increasingly influenced modern Android development while Java continues to be present in existing applications.

A comparative study published through IEEE in 2020 evaluated Java and Kotlin applications using source-code characteristics, application performance, and data usage. The study reported that Kotlin provided advantages in terms of code conciseness and data usage, whereas Java demonstrated advantages in initial compilation time and application package size. These findings suggest that the choice between Java and Kotlin can depend on the specific requirements and priorities of an Android application [3].

Another empirical study examined the effectiveness of Kotlin and Java in Android application development tasks. The researchers reported that Kotlin generated more concise implementations and showed advantages related to certain programming pitfalls, while Java received favorable perceptions regarding development-tool support. The study did not establish a universal maintainability advantage for either language **[1]** .

These findings indicate that Java should not simply be considered obsolete because of Kotlin's growing adoption. Instead, Java continues to be relevant for maintaining existing applications, supporting established development practices, and leveraging its mature ecosystem. At the same time, Kotlin provides characteristics that may improve developer productivity and reduce certain forms of programming complexity.

3.5 Security and Software Analysis

Security is an important research area in Android application development. Studies have examined security weaknesses in Android applications written using Java and Kotlin, demonstrating that programming-language choice alone does not eliminate security risks.

Research investigating security weaknesses in Java and Kotlin Android applications has analyzed real software-development changes and developer experiences. Such research demonstrates the importance of secure coding practices, security-aware development processes, and automated tools for identifying vulnerabilities [2] .

Software analysis is also becoming more challenging as Android applications use multiple programming languages and modern language features. Existing analysis techniques originally designed around Java may require adaptation when analyzing Kotlin applications

because Kotlin introduces language constructs and compiler-generated representations that differ from traditional Java code.

Consequently, future Android development research needs to consider security and software analysis across multilingual Android code bases rather than examining Java and Kotlin as completely isolated technologies.

3.6 Emerging Trends

The literature indicates several important trends in Android application development.

First, the transition from Java toward Kotlin represents a major change in Android programming practices. However, the interoperability between Java and Kotlin means that Java is expected to remain relevant in existing applications and mixed-language projects.

Second, Android development is increasingly influenced by modern software architecture and modular development practices. These approaches aim to improve maintainability, scalability, testing, and reuse.

Third, automated software analysis and testing are becoming increasingly important because of the complexity and scale of modern mobile applications. Automated techniques can assist developers in identifying security vulnerabilities, programming errors, performance problems, and maintainability issues.

Finally, modern Android development increasingly emphasizes secure development, efficient resource usage, cloud and web-service integration, automated testing, and development productivity. These trends provide opportunities for continued research into how Java-based applications can be maintained, modernized, and integrated with contemporary Android technologies.

3.7 Summary of Findings

The literature reviewed in this study indicates that Java remains an important technology in the Android ecosystem despite the increasing adoption of Kotlin. Java provides the advantages of maturity, extensive existing code, ecosystem support, and interoperability with Kotlin. However, challenges related to verbosity, application security, performance, maintainability, and technological transition remain important.

The evidence also suggests that Java and Kotlin should not necessarily be viewed as mutually exclusive technologies. Their interoperability allows organizations to maintain existing Java applications while progressively adopting newer technologies. Therefore, the continuing role of Java in Android development is closely connected with software maintenance, modernization, migration, security, and integration with contemporary development practices.

4. Research Gaps and Future Research Directions

The analysis of existing literature indicates that considerable research has been conducted on Android application development, programming-language comparison, security, maintainability, and software analysis. However, several areas remain insufficiently explored. The identified gaps provide opportunities for future research in Java-based Android application development.

4.1 Limited Long-Term Evaluation of Java and Kotlin

Several studies compare Java and Kotlin using specific development tasks, applications, or performance measurements. However, fewer studies provide long-term evaluations of projects maintained over several years. Future research could investigate how Java and Kotlin affect maintenance effort, defect rates, developer productivity, application evolution, and technical debt over extended development periods.

4.2 Need for Comprehensive Performance Studies

Existing comparisons have considered factors such as compilation time, application package size, data usage, and resource consumption. However, Android applications operate across a wide range of hardware and software configurations. More comprehensive studies are required to evaluate CPU usage, memory consumption, battery usage, network performance, startup time, and runtime behavior across different Android devices and operating-system versions.

4.3 Security of Java-Based Android Applications

Security remains an important challenge in mobile application development. Existing research has identified security weaknesses in Android applications and examined issues involving Java and Kotlin. However, further research is needed to develop practical security frameworks specifically designed for identifying and preventing vulnerabilities in Java-based Android applications throughout the software development life cycle.

4.4 Java-to-Kotlin Migration

The increasing adoption of Kotlin creates an important research opportunity concerning the migration of existing Java applications. Migration may involve source-code transformation, developer training, testing, compatibility issues, and changes in software-analysis tools. Future studies could develop systematic migration strategies and evaluate their effects on development cost, application quality, maintainability, and performance.

4.5 Automated Testing and Software Analysis

Modern Android applications are increasingly complex and may contain large amounts of code and third-party dependencies. Automated testing and software-analysis techniques can

help identify defects, security vulnerabilities, and performance problems. However, further research is required to improve automated techniques for mixed Java-Kotlin Android projects and to investigate how artificial intelligence and machine-learning techniques can support testing and code analysis.

4.6 Limited Integration of Multiple Development Factors

Many existing studies focus on a single dimension, such as performance, security, maintainability, or programming-language characteristics. A research gap therefore exists in studies that simultaneously evaluate these factors. Future empirical research could develop a comprehensive evaluation framework that considers performance, security, maintainability, development effort, application size, and developer productivity together.

4.7 Research on Legacy Java Android Applications

A significant amount of existing Android software has been developed using Java. However, research focusing specifically on the modernization and long-term maintenance of legacy Java Android applications is comparatively limited. Future research could investigate techniques for refactoring, modernization, dependency management, security improvement, and integration of legacy Java applications with contemporary Android technologies.

4.8 Future Research Directions

Based on the identified gaps, future research should focus on empirical comparisons of Java and Kotlin using larger and more diverse application datasets. Research should also investigate automated Java-to-Kotlin migration, secure coding practices, automated testing, performance optimization, and intelligent software-analysis techniques. In addition, studies involving real-world industrial Android projects could provide stronger evidence regarding the practical advantages and limitations of Java-based development.

Overall, future research should move beyond simple comparisons of programming languages and investigate the complete software development life cycle. Such research can provide a more comprehensive understanding of how Java-based Android applications can be developed, maintained, secured, modernized, and integrated with emerging Android technologies.

5. DISCUSSION

The findings of this review demonstrate that Java continues to have relevance in Android application development despite the increasing adoption of Kotlin. The transition toward Kotlin should not necessarily be interpreted as the complete disappearance of Java from the Android ecosystem. Instead, the literature indicates that both languages can coexist,

particularly because of their interoperability and the large number of existing Android applications developed using Java.

The findings related to RQ1 indicate that Java has supported a broad range of Android development techniques, including object-oriented programming, component-based development, database integration, network communication, application testing, and interaction with web services. These established techniques have contributed to the maturity of Java-based Android development. The continued availability of libraries, development resources, and existing source code also reduces the need for organizations to completely replace Java-based applications.

For RQ2, the literature indicates that Android development is supported by a broad ecosystem of development tools, software libraries, APIs, databases, architectural approaches, and testing technologies. Android Studio and the Android SDK provide the fundamental environment for application development, while modern architectural and modular approaches can improve application maintainability and scalability. The interoperability of Java and Kotlin further enables organizations to introduce modern technologies without necessarily rewriting an entire Java application.

The findings related to RQ3 show that Android developers continue to face challenges involving security, performance, maintainability, compatibility, and software evolution. Security is particularly important because mobile applications frequently process sensitive information and communicate with external services. Security weaknesses can result from programming practices, dependencies, data handling, communication mechanisms, and insufficient security testing. Therefore, secure application development should remain an important consideration regardless of the programming language used.

The comparison between Java and Kotlin provides an important perspective on RQ3 and RQ4. Existing empirical studies indicate that Kotlin can reduce source-code verbosity and provide advantages in certain programming situations, whereas Java can provide benefits related to its mature ecosystem and existing development infrastructure. Comparative research has also demonstrated that the relative performance of the two languages can depend on the specific measurement and application context. Consequently, selecting a programming language should be based on project requirements rather than assuming that one language is universally superior.

The increasing adoption of Kotlin also creates challenges for organizations maintaining legacy Java applications. Migration requires planning, developer knowledge, testing, code transformation, and evaluation of existing dependencies. However, Java-Kotlin

interoperability provides an opportunity for gradual migration. Organizations can therefore modernize applications incrementally while continuing to maintain existing Java components. Another important observation is that existing research often examines individual aspects of Android development independently. For example, one study may focus on programming-language characteristics, while another focuses on security or software analysis. There is comparatively less integrated research evaluating development productivity, performance, security, maintainability, and migration effort together. This observation supports the research gaps identified in the previous section.

Overall, the review suggests that the future of Java in Android development is likely to involve coexistence with modern technologies rather than immediate replacement. Java remains particularly relevant for legacy application maintenance, existing enterprise applications, education, and projects with established Java code bases. At the same time, modern development practices require developers to understand newer languages, architectures, security techniques, automation tools, and software-analysis methods. Therefore, future research should investigate how Java-based applications can be effectively maintained and modernized while taking advantage of developments in the broader Android ecosystem.

6. CONCLUSION

Java has played a fundamental role in the development of Android applications and continues to remain relevant despite the rapid growth of Kotlin and other modern development technologies. This systematic review examined existing literature related to Java-based Android application development with particular emphasis on development techniques, supporting tools and technologies, challenges, and emerging trends.

The reviewed literature indicates that Java benefits from a mature ecosystem, extensive existing code bases, established development practices, and interoperability with Kotlin. These characteristics make Java particularly relevant for the maintenance and modernization of existing Android applications. At the same time, Kotlin provides advantages such as concise source code and improved support for certain modern programming practices. Therefore, the transition toward Kotlin should be viewed as an evolution of Android development rather than an immediate elimination of Java.

The review also identified several continuing challenges, including application security, performance optimization, maintainability, compatibility, software evolution, and Java-to-Kotlin migration. These challenges demonstrate that programming-language selection alone

does not determine the quality of an Android application. Development practices, architecture, testing, security mechanisms, tools, and maintenance strategies also play important roles.

The identified research gaps indicate the need for further empirical investigation of Java and Kotlin across real-world Android projects and longer development periods. Future research should particularly focus on secure Java-based development, automated testing and software analysis, Java-to-Kotlin migration, performance optimization, legacy application modernization, and the application of artificial intelligence to Android software engineering.

In conclusion, Java remains a significant part of the Android development ecosystem, particularly in existing and legacy applications. Its future relevance will depend on how effectively Java-based applications can be maintained, secured, modernized, and integrated with emerging Android technologies. Continued research in these areas can provide valuable guidance for developers, educators, researchers, and organizations involved in Android application development.

REFERENCES

1. L. Ardito, R. Coppola, G. Malnati, and M. Torchiano, “Effectiveness of Kotlin vs. Java in Android App Development Tasks,” *Information and Software Technology*, vol. 127, Art. no. 106374, 2020, doi: 10.1016/j.infsof.2020.106374.
2. A. Mazuera-Rozo, C. Escobar-Velásquez, J. Espitia-Acero, D. Vega-Guzmán, C. Trubiani, M. Linares-Vásquez, and G. Bavota, “Taxonomy of Security Weaknesses in Java and Kotlin Android Apps,” *Journal of Systems and Software*, vol. 187, Art. no. 111233, 2022, doi: 10.1016/j.jss.2022.111233.
3. “Comparative Analysis of Kotlin and Java Languages Used to Create Applications for Android Mobile,” in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, 2020, doi: 10.1109/ISRITI51436.2020.9315483.