# Major Types of Mobile Applications: Review and Comparison

An application is a software that lets you exchange information with customers and help them complete specific tasks. Different types of applications, or apps, are based on their development method and internal functionality. Web apps are delivered over an internet browser. Users don't need to install them on their devices. Native apps, on the other hand, are built for a specific platform or device type. The user must install the appropriate software version on their device of choice. Hybrid apps are native applications with a web browser embedded inside them

* 1. **Native Mobile Apps**

Native apps are software applications built specifically for a particular operating system (OS), like iOS or Android, and are designed to leverage the device's hardware and software for optimal performance and a seamless user experience

Native apps are developed using programming languages and tools specific to the target OS, such as Swift and Objective-C for iOS or Java and Kotlin for Android.

This means that if you want to have an app published on Google Play and App Store, you will have to program it twice, once for each of the languages mentioned above.

Native apps are typically distributed through app stores like the Apple App Store or Google Play Store.

Popular examples of native apps include Instagram, Google Maps, Uber, and Spotify.

* + 1. **Advantages of Native Applications**

**Performance and User Experience:**

Native apps are optimized for the specific platform, resulting in faster performance, smoother transitions, and a more intuitive user experience.

**Offline Functionality:**

Native apps can store data locally and access features even without an internet connection, offering a more reliable and seamless user experience.

**Seamless Integration:**

Native apps can seamlessly integrate with the operating system's features, such as notifications, camera, geolocation, and other device functionalities.

**Access to Platform-Specific Security Features:**

Native apps have access to platform-specific built-in security features, enhancing the security of user data.

**Widgets:**

A native app can benefit from widgets. That is, it can incorporate a thumbnail view of the app on the smartphone's homepage.

* + 1. **Disadvantages of Native Apps**

**Development Time and Cost**: Creating individual apps for iOS and Android can be both costly and time-consuming. This is due to the requirement of maintaining two separate codebases, one for each platform.

**Skill Requirements**: Building native apps requires platform-specific programming languages and frameworks. Developers need to master platform-specific languages (Swift/Objective-C for iOS, Java/Kotlin for Android). Acquiring proficiency in multiple languages poses a learning curve.

**Updates and Maintenance**: Maintaining two codebases means that updates and bug fixes must be implemented separately for iOS and Android. This can lead to longer development cycles and higher maintenance costs.

* 1. **PWAs (Progressive Web Apps)**

Progressive Web Apps (PWAs) are web applications built with web technologies that offer a native-app-like experience, including the ability to be installed, work offline, and receive push notifications, all while being accessible through a web browser.

They are called "**progressive**" because they gradually absorb functionalities from Native apps

They are cheaper and faster to develop than native apps.

Example of PWA Apps are**: Pinterest and Trivago, Tinder, Starbucks**

**Advantages Of PWAs**

**Installable:**

Users can install PWAs on their devices, making them easily accessible and giving them the appearance of a native app.

**SEO-Friendly:**

PWAs are built using web technologies, making them easily indexable by search engines.

**Cross-Platform Compatibility:**

PWAs can run on multiple platforms and devices (smartphones, tablets, desktops) from a single codebase, simplifying development and reducing costs.

**Lower Development Costs:**

PWAs are generally less expensive and faster to develop compared to native apps, as they leverage existing web technologies and a single codebase.

**No App Store Dependence:**

PWAs can be installed directly from a browser and don't require users to go through app stores, simplifying the installation process.

**Disadvantages Of PWAs**

### Compatibility with iOS

Since iOS 11.3, it’s been possible to run PWAs on Apple devices, but you can forget about compatibility with older devices. What’s more, Apple doesn’t allow PWAs to access many important features, including Touch ID, Face ID, ARKit, Bluetooth, serial, Beacons, altimeter sensor, and even battery information.

**Hardware Limitations:**

PWAs, being web applications, may not have full access to device features like NFC, Bluetooth, advanced camera controls, fingerprint scanners, or proximity sensors, as native apps do.

**Legacy Device Compatibility:**

PWAs may not be fully compatible with older devices or browsers that don't support the latest web standards.

**Performance:**

PWAs, built with web technologies like JavaScript, HTML, and CSS, might not perform as efficiently as native apps, especially for complex tasks or resource-intensive applications.

* 1. **Hybrid Apps**

Hybrid mobile apps combine elements of native and web apps, using web technologies (HTML, CSS, JavaScript) wrapped in a native app shell to enable cross-platform functionality with a single codebase.

**How they work:**

They are built using web technologies (HTML, CSS, JavaScript) and then packaged within a native app container, allowing them to access device features and run on multiple platforms (iOS and Android) with a single codebase.

**Advantages of Hybrid Apps**

### One codebase to rule them all:

Hybrid apps use one codebase and can work across multiple platforms.

### Quicker build:

A hybrid app is quicker to build because it doesn’t have any native requirements and uses the most basic coding languages; HTML, CSS, and JavaScript.

**Disadvantages of Hybrid Apps**

## ****Dependency on Third-Party Plugins****

Hybrid apps require **plugins** (e.g., Cordova plugins) to access native features. If a required plugin is outdated or unsupported, development becomes challenging.

## ****App Store Rejections****

Some hybrid apps get **rejected** by the **App Store (iOS)** if they don’t meet performance standards or rely too much on WebViews.

Rely on system browser security

1. **Mobile Application Programming Languages: Review and Compare**

Various programming languages dominate the dynamic world of mobile application development, each tailored to specific platforms or offering cross-platform capabilities. From **Java** and **Kotlin** for Android development to **Swift** and **Objective-C** for iOS, developers have an array of options to choose from based on project requirements and platform preferences.

This section discusses and compares the different mobile app programming languages mentioned above in terms of syntax, performance characteristics, optimisations available and benchmarks showing strengths and weaknesses.

* 1. **Java**

Java is a widely used, versatile, object-oriented programming language, often used for developing mobile applications, particularly Android apps, due to its mature ecosystem and powerful framework.

With its object-oriented approach, Java facilitates modular and reusable code, simplifying the development process and enhancing code maintainability

**Features of Java**

**Object Oriented:** In Java everything is an Object. Java Can be easily extended since it is based on an Object Model

**Platform Independent:** Unlike other programming languages, when Java is compiled, it is not compiled into a platform-specific machine, but rather into a platform independent byte code.

**Simple**: Java is designed to be easy to learn. If you understand the basics of OOP, it would be easy to Master

**High Performance**: With the use of Just-in-time compilers, Java enables high performance.

**Secure**: With Java’s secure feature, it enables the development of Virus free, tamper free systems

**Advantages of Java**

* **Multithreaded –** It is the potential for a program to perform numerous assignments simultaneously. at long last time showed up to become familiar with the ideas of Multithreading in Java.

 **Java is fast and high-performance** – Uses JIT compilation and efficient memory management for optimized execution speed.

 **Java supports functional programming** – Features like lambda expressions and streams enable concise, expressive, and parallel-friendly code.

 **Java has expert leadership** – Maintained by Oracle and a strong community, ensuring regular updates and long-term support.

 **Java is fast and high-performance** – Already mentioned; Java optimizes execution with JIT and efficient garbage collection.

 **Java is open source** – OpenJDK allows free use, modification, and contributions, fostering innovation and broad adoption.

**Disadvantages of Java**

 **Non-native performance** – Java runs on the JVM, adding an abstraction layer that makes it slower than compiled languages like C or Rust.

 **Lack of low-level programming support** – Java doesn’t allow direct memory manipulation like C/C++, limiting its use in system programming and hardware-level applications.

 **Threat of vendor interference** – Oracle owns Java and can impose licensing or policy changes that impact developers and businesses.

 **Confusing licensing models** – Java has multiple versions (Oracle JDK, OpenJDK) with different licensing terms, making it tricky to choose the right one.

 **JDK distro proliferation** – Multiple JDK distributions (Oracle, OpenJDK, Amazon Corretto, etc.) lead to fragmentation and confusion about which version to use.

* 1. **Swift(IOS development)**

Swift is a general-purpose, multi-paradigm, object-oriented, functional, imperative, and block-structured language. Swift is the result of the latest research on programming languages and is built using a modern approach to safety, and software design patterns by Apple Inc. for iOS applications, macOS applications, watchOS applications, tvOS applications.

**Key Feature Of Swift**

* Generics that are powerful and simple to use
* Protocol extensions that make writing generic code even easier
* First-class functions and a lightweight closure syntax
* Fast and concise iteration over a range or collection
* Tuples and multiple return values
* Structs that support methods, extensions, and protocols
* Enums can have payloads and support pattern matchin

**Advantages of Swift**

 **Fast Performance** – Swift is optimized for speed, often outperforming Objective-C due to LLVM compilation.

 **Easy to Read & Write** – Its clean, concise syntax makes code more readable and maintainable.

 **Memory Management with ARC** – Automatic Reference Counting (ARC) efficiently manages memory, reducing memory leaks.

 **Safe & Secure** – Features like optionals and strong type safety help prevent crashes and runtime errors.

 **Open Source** – Available to the public, with active community contributions and improvements.

**Disadvantages of Swift**

 **Smaller Talent Pool** – Fewer Swift developers compared to Java, Kotlin, or JavaScript, making hiring more challenging.

 **Frequent Updates** – Swift evolves rapidly, sometimes causing compatibility issues with older versions..

 **Limited Support for Backend Development** – While Swift can be used for backend (e.g., with Vapor), it’s not as popular as Node.js or Python.

 **Relatively New Language** – Introduced in 2014, Swift is still evolving, leading to occasional instability in frameworks.

* 1. **Kotlin**

Kotlin is an open-source, statically-typed programming language that supports both object-oriented and functional programming. Kotlin provides similar syntax and concepts from other languages, including C#, Java, and Scala, among many others. Kotlin does not aim to be unique—instead, it draws inspiration from decades of language development. It exists in variants that target the JVM (Kotlin/JVM), JavaScript (Kotlin/JS), and native code (Kotlin/Native).

**Features of Kotlin**

**Concise and Expressive Syntax:**

Kotlin offers a more compact and readable syntax compared to Java, reducing boilerplate code.

**Null Safety:**

Kotlin's type system prevents NullPointerException errors by explicitly handling null values, enhancing code reliability.

**Modern Language Features:**

Kotlin incorporates features like extension functions, data classes, smart casts, and lambda expressions, streamlining development.

**Advantages of Kotlin**

 **Concise and Readable Syntax** – Requires less boilerplate code compared to Java, improving readability and maintainability.

 **Null Safety** – Built-in null safety features prevent null pointer exceptions, reducing crashes.

 **Modern Features** – Includes functional programming, extension functions, and type inference for cleaner code.

**Fewer bugs.** If you code in Kotlin, the chances of making bugs are extremely low compared to other complex programming languages.

 **Officially Supported by Google** – Kotlin is the preferred language for Android development, ensuring long-term support.

**Disadvantages Of Kotlin**

 **Slower Compilation Speed** – While runtime performance is good, Kotlin’s compilation can be slower than Java.

 **Smaller Developer Community** – Fewer Kotlin developers and resources compared to Java.

 **Inconsistent Compilation Performance** – Kotlin’s build times can vary depending on project complexity and tooling.

* 1. **React Native(Hybrid)**

React Native also known as RN is popular JavaScript-based mobile app framework that allows you to build natively-rendered mobile apps for ios and Android. The framework lets you create applications for various platforms by using the same codebase.

**Advantages of React Native**

 **Cross-Platform Development** – Write one codebase that runs on both iOS and Android, saving time and effort.

 **Hot Reloading** – Instantly see code changes in the app without needing to restart, speeding up development.

 **Native Components** – Uses native UI components for better performance and a more native-like user experience.

**Limitations Of React Native**

 **Performance Limitations** – Not as fast as fully native apps, especially for heavy animations and complex calculations.

 **Large App Size** – React Native apps tend to have a larger file size compared to fully native apps..

 **Debugging Challenges** – Debugging React Native apps can be tricky due to issues with third-party packages and platform-specific bugs.

**Performance Optimisation**

**Performance Limitations → Optimize Code & Use Native Modules**

* Use the **Hermes engine** for faster app startup.
* Optimize rendering with **FlatList, VirtualizedList**, and avoid unnecessary re-renders.

**Large App Size → Reduce Bundle Size**

* Optimize images and assets using **WebP** format.
* Remove unused dependencies and fonts.

**Debugging Challenges → Use Proper Debugging Tools**

* Use **React Developer Tools**, Flipper, and Reactotron for debugging.
  1. Flutter

Flutter is an open source framework developed by Google to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile applications in Googles own language, Dart.

**Features of Flutter**

* Modern and Reactive framework
* Uses dart programming language and is easy to learn
* Fast Development
* Huge widget catalog

**Advantages Of Flutter**

* Dart has a large repository of software packages which lets you to extend the capabilities of your application
* Developers need to write just a single code base for both applications (both Android and iOS platforms).
* With Flutter developers have full control over the Widget and its layout

Disadvantages Of Flutter

* Since it is coded in Dart language, a developer needs to learn new language (though it is easy to learn)
* Modern framework tries to separate logic and UI as much as possible but, in Flutter, user interface and logic is intermixed

Comparisons

Here we are going to review the various mobile application programming based on the following metrics:

* Performance
* Development speed
* Cross platform support
* Use cases
* Memory management

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | Java | Kotlin | Swift | React Native | Flutter |
| Performance | High (compiled with JIT) | Slightly slower than Java but optimized | Faster than Java/Kotlin (compiled to native) | Near-native but slower than pure native apps | High performance (compiles to native) |
| **Development Speed** | Moderate (verbose syntax) | Faster than Java (less boilerplate) | Faster than Objective-C | Fast (Hot Reload, reusable code) | Fast (Hot Reload, fully customizable UI) |
| **Cross-Platform Support** | No (Android only) | No (Android only) | No (iOS only) | Yes (Android & iOS) | Yes (Android, iOS, Web, Desktop) |
| **Use Cases** | Android apps, enterprise applications, backend (Spring Boot) | Android apps, backend (Ktor), cross-platform (Kotlin Multiplatform) | iOS apps, macOS, watchOS, tvOS | Cross-platform mobile apps (Android & iOS) | Cross-platform (Android, iOS, Web, Desktop) |
| **Memory Management** | Automatic (Garbage Collection) | Automatic (Garbage Collection) | Automatic (ARC) | JavaScript-based, can cause memory leaks | Uses Dart’s Garbage Collection (optimized) |

Choosing the Right programming Language

 **Java**: Best for **legacy Android development** and enterprise applications.

 **Kotlin**: Modern alternative to Java, **official language for Android development**.

 **Swift**: The best choice for **iOS/macOS applications**.

 **React Native**: Ideal for **cross-platform apps** with a web development background.

 **Flutter**: Best for **high-performance cross-platform apps**, especially with a **custom UI**.