OPERATING SYSTEMS

PREPARED BY: ODY SHBAYAH-1201462.

OS-HW

SECTION2.

DR:ABD-SALAM-SAYYAD

DATE: 10/23/2023.

1. **Compare and contrast the Android and iOS operating systems from the following viewpoints:**
    a. **customizability**
    b. **performance**
    c. **business model (how does iOS make money for Apple? How does Android make money for Google?**

we must say that the comaparison is depending on user choices, iOS and Android's level of customization can change. While iOS delivers a more regulated and controlled experience that is likewise valued by a separate group of users who favor consistency and security, Android offers a great level of customisation that many users prefer. Web searches and user reviews can shed light on how various user groups see and make use of these customisation possibilities.

Android:

1 . User-Driven Customization: Users like the freedom Android offers to customize their devices because of its high degree of customizability. Since customers can alter the home screen, add personalized widgets, and even utilize third-party launchers to entirely modify the appearance and feel of their smartphones, many users find this flexibility to be a big benefit.

2. Diverse App Sources: Users frequently value the option to download apps from places other than the Google Play Store. Although this gives consumers greater control, some may be wary owing to possible security issues. The large selection of third-party apps that are accessible to Android users is well appreciated.

3. Rooting The possibility to root an Android smartphone is viewed by experienced users as a potent means of controlling and deeply customizing the operating system. Users should take caution since this approach has the potential to void warranties and expose devices to security flaws.

iOS :

1. Consistency and User Experience: iOS is intended to provide a consistent user experience, and many consumers value the close supervision Apple gives to the appearance and performance of its products. For individuals who value a uniform experience, this technique offers a clean and consistent user interface, which may be considered as a positive feature.

2. Limited Customization: Although iOS's customization options have improved over time, it is still relatively less customizable than Android. For those who prefer a simpler and more controlled environment, users can alter wallpapers, rearrange app icons, and change some settings.

3.App Store Ecosystem: The Apple App Store is renowned for its meticulous review procedure, which contributes to ensuring the reliability and security of iOS apps. Although some users might wish for more flexibility in app installation, users appreciate the security and dependability of apps from the App Store.

2. **AOT vs. JIT**
    a. **Why does Android use Ahead-of-Time (AOT) compilation rather than Just-in-Time (JIT) compilation?**
    b. **What are the advantages and disadvantages of each method?**

A. Android prefers AOT compilation. due to its superior performance, battery efficiency, and security advantages, AOT makes sure that apps launch and function without interupts, saving battery life and lowering security risks. But it might lead to bigger app sizes and less adaptability.

JIT compilation offers more adaptability and smaller app sizes, but it can also result in longer startup times, more battery use, and security risks. In order to optimize the user experience, Android does use a combination of AOT and JIT techniques, such as profile-guided compilation, to strike a balance between performance and flexibility.

b.1 **Ahead-of-Time (AOT) Compilation**:

****Advantages:**

1. Performance: Because AOT compiles code before execution, apps launch more quickly and perform better overall.
2. Battery Efficiency: Because the code has already been precompiled, AOT uses less CPU and electricity, extending the battery life of portable electronics.
3. Security: By reducing the attack surface for potential weaknesses, AOT can enhance security.
4. Review of the App Store: AOT makes it possible for app shops to examine and validate code before to publication, improving compliance and safety.
5. Device agnosticism: During installation, apps can be compiled for a number of different device architectures, ensuring interoperability with a range of devices.

****Disadvantages:**

1. higher App Size: Because AOT compilation includes compiled code for many device architectures, it may lead to higher app files.
2. Less Runtime Adaptability: Because the code is precompiled, it could be less adaptive to runtime changes.

b.2 **Compiling Just-in-Time (JIT):**

****Advantages:**

1. JIT compiles code while it is being executed, which might result in reduced app sizes.
2. Better Runtime Performance is Achieved Through JIT's Capability To Optimize Code Execution Based On The Specific Device.
3. Flexibility: Since code is compiled as needed, it offers greater flexibility and adaptability.

****Disadvantages:**

1. Slower App Startup: JIT compilation builds code when you run the app, which can result in slower app startup times.
2. Battery Drain: JIT consumes more power and requires more CPU cycles, which reduces battery life.
3. Security Risks: Because vulnerabilities can be taken advantage of during compilation, there are security risks introduced during runtime.
4. Limited App Store Oversight: JIT compilation's code integrity is subject to less monitoring and supervision in app stores.

3. Native vs. cross-Platform

a. **What is the difference between native mobile code and cross-platform mobile code?**
b. **What are the advantages and disadvantages of each?**
c. **How can a programming framework like FLUTTER produce mobile apps that work on both Android and iOS?**

a.1 developing code especially for a certain mobile platform, such as Java or Kotlin for Android or Swift or Objective-C for iOS, is referred to as developing native mobile code.

a.2 Using frameworks like Flutter, React Native, or Xamarin, cross-platform mobile programming enables developers to create code once and run it on various platforms, such as Android and iOS.

b.1 **Mobile Native Code:**

****Advantages:**

1. High Performance: Because native code is platform-optimized, it often executes more quickly.
2. Full Access: Device-specific functionality and APIs are entirely accessible to developers.
3. Apps have a native appearance and feel, which can improve the user experience.

****Disadvantages:**

1. Platform-Specific: Each platform requires its own code to be built.
2. Time-consuming: Writing platform-specific code might slow down development.
3. Skills Necessary: Developers must be proficient in the programming language used by each platform.

b.2 **Mobile Cross-Platform Code:**

****Advantages:**

1. Code reuse: By using a large amount of the same code on both platforms, developers may speed up development.
2. Faster Development: When compared to creating distinct native apps, cross-platform development might be quicker.
3. Single Codebase: A single codebase makes maintenance easier.

****Disadvantages:**

1. Performance Slightly Lower: Because of the abstraction layer, cross-platform programs might not be as quick as native apps.
2. Access Restrictions: Some platform-specific features may have limited access.
3. Cross-platform tools may have trouble addressing particular device or platform capabilities due to compatibility issues.

c. In order to create mobile apps for both Android and iOS, Flutter compiles Dart code into native ARM code, uses widgets to provide a unified user interface, and plugins to get access to platform-specific capabilities. Cross-platform frameworks offers advantages over entirely native programming in terms of code reuse and speedier development, although there may be minor performance trade-offs. The precise project objectives and the trade-offs that best fit those requirements will determine whether to use native or cross-platform development.

4.Describe google Chrome as an example of multi-process, multi-threaded application.

The multi-process, multi-threaded design of Google Chrome is comparable to a well-run stage show. To maintain stability and security, each web page runs in a distinct "actor" (renderer process). The "roles" that threads play for these actors include rendering, networking, and storage.

There are several benefits to this design. Because problems on one online page won't effect the entire browser, it improves stability and security. Even with several open tabs, it minimizes resource utilization and offers a quick, responsive browsing experience.

Like a well-executed theatrical play, Chrome's architecture orchestrates a smooth, safe, and high-performance online surfing experience.