Faculty of Engineering and Technology
Electrical and Computer Engineering Department

OPERATING SYSTEMS
ENCS3390

Answers of assignment 1

Prepared by:
Student Name: Raghad Murad Buzia
Student #: 1212214

Instructor: Abdel Salam Sayyad

Section: 1

Date: 24/10/2023

Problem 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?)

Answer:

a. Customizability varies between Android and iOS. Android provides extensive customization options for both users and developers, allowing for modifications to the user interface, the installation of apps from sources outside of Google Play, and comprehensive system modifications. In contrast, iOS is less customizable, with limited options for customization and strict criteria for app installation through the App Store.

b. In terms of performance, iOS is known for its high performance and smooth responsiveness. Apple designs hardware and system together, enabling high compatibility and enhanced performance. From this we conclude that iOS applications work well and smoothly on most devices. As for the Android operating system, performance varies depending on the device and version. Due to the wide variety of Android devices, you may experience different performance from one device to another. High-end devices provide excellent performance, while devices with lower specifications may be less powerful.

c. Android makes money for Google through several revenue streams, including the Google Play Store, where it takes a portion of revenue from apps and in-app purchases. In addition, Google receives revenue from in-app advertising and Google services built into Android such as Gmail and Google Drive. As for iOS, Apple's profits are mainly due to the App Store, where it takes a percentage of revenue from apps and in-app purchases. In addition, Apple benefits from iCloud music and cloud storage services and also from sales of its own devices such as the iPhone and iPad.

Problem 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?

Note:

- **ahead-of-time compilation (AOT compilation)** is the act of compiling an (often) higher-level programming language into an (often) lower-level language before execution of a program, usually at build-time, to reduce the amount of work need to be performed at run time.

- **just-in-time compilation (JIT compilation)** (also dynamic translation or run-time compilations) is compilation (of computer code) during execution of a program (at run time) rather than before execution.

Answer:

a. The Android system uses ahead-of-time compilation (AOT) for many advantages, such as:

   Improving performance, as ahead-of-time compilation (AOT) allows for improved overall performance because pre-compiled applications can run more quickly because they do not need translation operations during execution, and this is important in high-performance applications. It also helps reduce memory consumption, as it converts the code in advance, which enables the device to reduce memory consumption by a certain amount because it does not require keeping unused code in memory. In addition, ahead-of-time compilation (AOT) increases the security of applications because pre-compiled code is difficult to exploit compared to code compiled in real time, "Security vulnerabilities such as executing malicious code during runtime are difficult to exploit."

   However, there are some cases where Android uses just-in-time (JIT) compilation.

b. **For ahead-of-time compilation (AOT compilation):**

advantages	disadvantages
Predictable Performance: AOT compiles code before execution, leading to consistent and predictable performance, making it ideal for critical applications and real-time systems.	**Limited Adaptability:** AOT-compiled code can't adapt to changing runtime conditions, potentially resulting in suboptimal performance in dynamic environments.
Optimal Memory Usage: AOT-compiled apps generally use less memory since they don't need to retain source code or an interpreter in memory.	**Extended Development Time:** AOT compilation adds an extra step in the development process, which can increase project timelines.
Enhanced Security: AOT-compiled code is less vulnerable to runtime attacks as it's not stored in memory in a readable format.	**Platform Dependence:** AOT compilation often involves compiling code for specific target platforms, which might hinder cross-platform development.
Simplified Distribution: AOT-compiled programs are self-contained, simplifying deployment and distribution.	

For ahead-of-time compilation (AOT compilation):

advantages	disadvantages
Dynamic Performance: JIT compilation optimizes code as it runs, adapting to the system's specific conditions, which can lead to improved performance in real-time.	**Delayed Startup:** JIT introduces a slight delay at the beginning of application execution because it compiles code on-the-fly.
Cross-Platform Portability: JIT allows writing code that works on different platforms without modification. The compiler customizes it to each platform when executed.	**Resource Intensive:** JIT requires additional memory and processing power to maintain an interpreter alongside the compiled code.
Efficient Memory Usage: JIT saves memory by loading only the parts of code in use, unlike AOT, which compiles everything.	**Security Concerns:** JIT-compiled code stored in memory might be susceptible to certain security vulnerabilities.
Developer-Friendly: JIT makes development faster and more flexible because you can write, test, and debug code without waiting for a separate compilation phase.	

Problem 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?

Answer:

a.

Aspect	native mobile code	cross-platform mobile code
Development Languages	Platform-specific languages (e.g., Swift, Objective-C, Java, Kotlin)	Typically uses a single codebase with languages like JavaScript, C#, or Dart.
Platform Specificity	Separate codebases for each platform (iOS, Android, etc.).	One codebase for multiple platforms.
Performance	Excellent performance due to direct hardware access and platform-specific optimizations.	Performance may not match native apps in all cases due to additional abstraction layer.
Access to Native Features	Full access to device features and capabilities.	Provides access to some native features, but not all platform-specific capabilities.
User Experience	Follows platform-specific design guidelines, ensuring a consistent user experience.	Aims for consistency across platforms but might have some differences in look and feel.
Development Time	Time-consuming as you need to build and maintain separate codebases.	More efficient in terms of development time and cost savings due to code reuse.

b. For native mobile code:

advantages	disadvantages
Native apps are known for their superior speed and performance due to direct access to the device's hardware.	Creating native apps demands separate codebases for different platforms, leading to increased development time and costs.
They have unrestricted access to all device features and platform-specific APIs, making them ideal for apps that require deep integration.	Developers must be well-versed in platform-specific languages, requiring additional training and skills development.
Native apps follow the platform's design guidelines, ensuring a consistent and familiar user experience, enhancing user satisfaction.	Synchronizing multiple codebases can be challenging, especially when addressing platform-specific updates and bug fixes.
Native development enjoys strong ecosystem support, with abundant documentation, libraries, and developer communities for each platform.	
These apps can benefit from robust security features and encryption capabilities, suitable for applications handling sensitive data.	

For cross-platform mobile code:

advantages	disadvantages
A single codebase can be used across multiple platforms, saving time and effort by eliminating redundant development tasks.	Cross-platform apps may not match the performance of native apps due to the additional layer involved in interacting with device hardware.
Cross-platform development is generally faster as you can write and maintain a single codebase.	Cross-platform frameworks offer access to some native features and plugins but may lack coverage of all platform-specific capabilities.
Developers can leverage their expertise in common languages (e.g., JavaScript, C#, Dart) and apply these skills to multiple platforms.	While cross-platform apps aim for consistency, variations in look and feel can affect user satisfaction.
Cross-platform apps can reach a broader audience by running on different platforms with one codebase.	Your project's success depends on the health and support of your chosen cross-platform framework, which may have limitations or dependencies affecting development.
Changes and updates can be applied uniformly to all platforms, reducing the complexities of platform-specific maintenance.	

c. A programming framework such as Flutter simplifies the development of cross-platform mobile applications by enabling a unified codebase that compiles into native code for both Android and iOS platforms. This is accomplished by harnessing a powerful rendering engine and a collection of platform-specific widgets. By adopting Flutter, developers can write code just once and deploy it on multiple platforms, thus significantly reducing development timelines and expenses. Moreover, Flutter offers an extensive range of plugins and libraries for accessing platform-specific functionalities as required. This approach ensures that applications exhibit excellent performance and an authentic, native appearance on both Android and iOS.

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

Note:

- Multi-Process: Software architecture where different tasks or components run in separate, isolated processes, enhancing stability and security.

- Multi-Threaded: Software design in which a single program is divided into multiple threads that execute concurrently within a single process, improving program efficiency by enabling parallel execution of tasks.

Answer:

Google Chrome is a prime example of a multi-process, multi-threaded application. Unlike traditional web browsers, Chrome operates using a unique architecture that separates its processes into multiple independent components. Each open tab, extension, and plugin runs in its own isolated process. Within these processes, multiple threads handle various tasks, such as rendering, networking, and JavaScript execution. This design enhances both security and stability, as a problem in one tab or extension doesn't necessarily affect the entire browser. Additionally, Chrome's multi-threaded approach optimizes performance, allowing for smoother multitasking and quicker loading times. By effectively utilizing multiple processes and threads, Chrome offers a robust and responsive web browsing experience.