Operating System

Assignment 3

Session-Fall-24

Submitted by:

Name: Muhammad Usman

ID: 221403

Section: BSCS-5-C

Submitted to: Miss Warda Aslam

Department of Computer Science

Faculty of Computing and AI

Air University, Islamabad

Android vs iOS

Comparative Analysis of Android vs iOS Through Operating System

Introduction

Android and iOS are the two dominant mobile operating systems, each with distinct architectural and functional attributes. This report delves into their comparative analysis through key OS concepts: process management, memory management, file systems, security, and CPU scheduling. By dissecting these elements, we gain insights into their operational efficiency, design philosophies, and user experiences.

Overview of the Selected Operating Systems

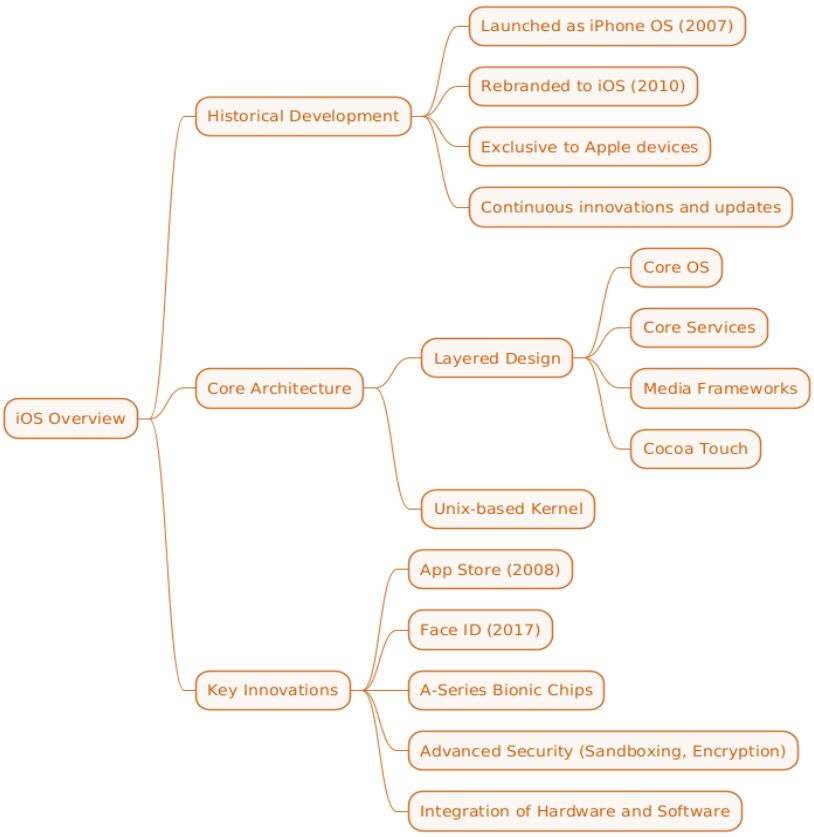
iOS Overview:

Historical Development and Core Architecture:

iOS, first launched as iPhone OS in 2007, marked a revolutionary shift in mobile technology with its pioneering multi-touch interface. Developed exclusively by Apple Inc., it introduced a tightly integrated ecosystem of hardware and software. The rebranding to "iOS" in 2010 reflected its broader applicability across devices like the iPhone, iPad, and iPod. Its layered architecture includes the Core OS, Core Services, Media, and Cocoa Touch frameworks, each fulfilling essential roles like multitasking, network access, and UI management. This architecture enables developers to create efficient, secure, and user-friendly applications.

Key Features and Innovations:

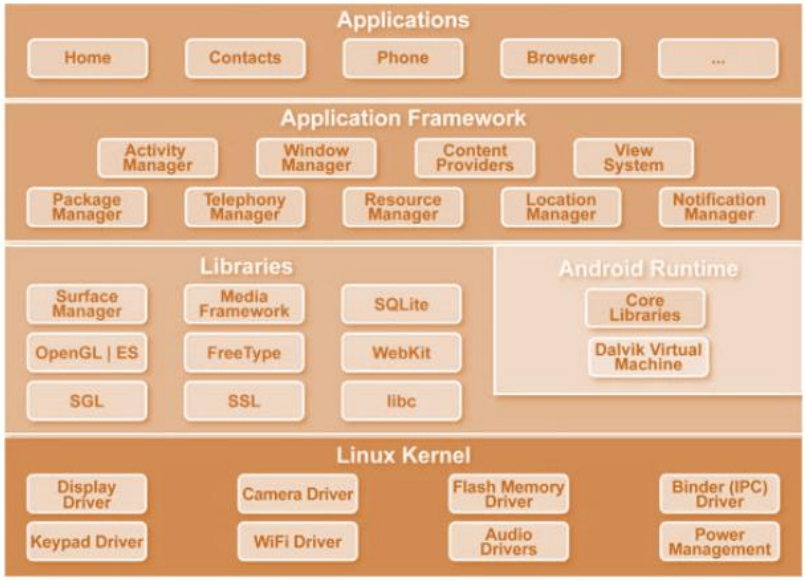
Notable innovations in iOS include the introduction of the App Store in 2008, which transformed app distribution and monetization. Apple's emphasis on security is evident through features like application sandboxing and encryption mechanisms. Cutting-edge technologies, such as Face ID and integration of hardware-specific machine learning via A-series Bionic chips, have made iOS synonymous with innovation.



Android Overview:

Historical Evolution and Architecture:

Android, launched by Google in 2008, quickly became the world's leading mobile OS. Its open-source nature, based on the Linux kernel, allows for extensive customization by developers and manufacturers. The OS has evolved through numerous versions named after desserts, with each iteration introducing significant enhancements in user experience and system efficiency. Android's layered architecture consists of the Linux Kernel, Native Libraries, Android Runtime, Application Framework, and Applications, collectively enabling comprehensive app support and user functionality.



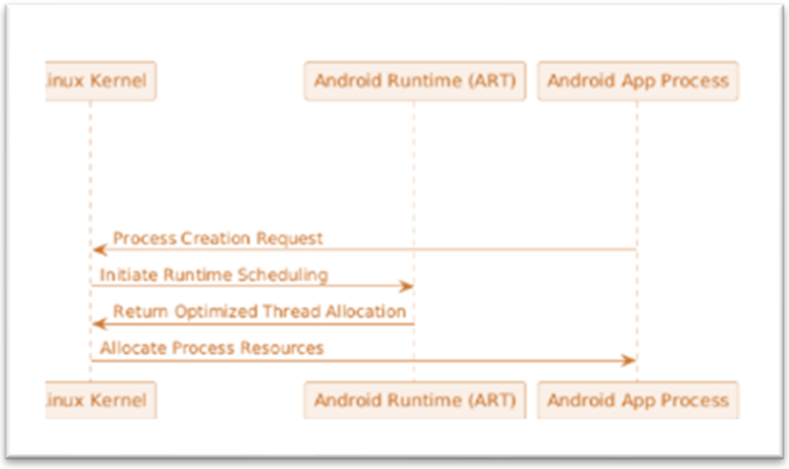
Open-Source Nature and Impact on Mobile Computing

The open-source model of Android has catalyzed its adoption across diverse hardware platforms, making it accessible at various price points. This adaptability has expanded Android's reach, fostering innovations in areas like IoT and wearables. However, it has also led to fragmentation, complicating updates and security management. Android's deep integration with Google services, such as Google Assistant and Google Play, has established it as a central pillar in mobile computing.

Process Management

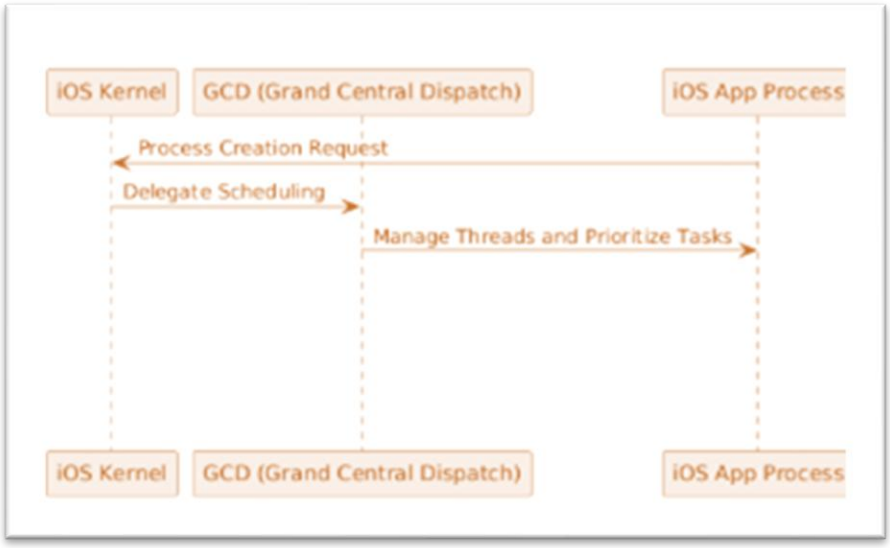
Android:

- **Process Creation:** Android is based on the Linux kernel, employing processes that handle application components such as Activities, Services, Broadcast Receivers, and Content Providers. Processes are created when an application is launched, and each app typically runs in its own Linux process.
- **Scheduling:** Android uses the Linux CFS (Completely Fair Scheduler) for CPU scheduling, prioritizing foreground processes.
- **Multitasking:** Android supports multitasking by keeping background processes in a Least Recently Used (LRU) cache.
- **Inter-Process Communication (IPC):** Binder IPC facilitates communication between processes, ensuring efficient data transfer and execution.



iOS:

- **Process Creation:** iOS restricts background processes to preserve battery life and system performance. Applications not in the foreground are suspended.
- **Scheduling:** iOS uses a proprietary scheduler optimized for responsiveness and energy efficiency.
- **Multitasking:** Limited to specific background tasks (e.g., audio playback, location tracking) with stringent control over resource allocation.
- **IPC:** Mach ports are used for IPC, ensuring secure and fast communication between processes.



Memory Management

Android:

- **Allocation & Deallocation:** Managed by the Dalvik/ART (Android Runtime) VM, employing garbage collection to free unused memory.
- **Virtual Memory:** Uses Linux's paging mechanism but avoids excessive swapping to minimize latency.
- **Caching & Protection:** Shared memory pages, Zygote process for app spawning, and ashmem for memory sharing.

iOS:

- **Allocation & Deallocation:** Utilizes Automatic Reference Counting (ARC) for memory management, minimizing the need for explicit deallocation.
- **Virtual Memory:** Employs compressed memory rather than paging to optimize performance.
- **Caching & Protection:** Sandboxing ensures applications do not interfere with each other, enhancing memory protection.

File System

Android:

- **File Storage:** Uses ext4 file system, supporting internal and external storage.
- **Organization:** Public and private directories, with apps storing data in sandboxed areas.
- **Access:** External storage is accessible by multiple apps, posing potential security risks.

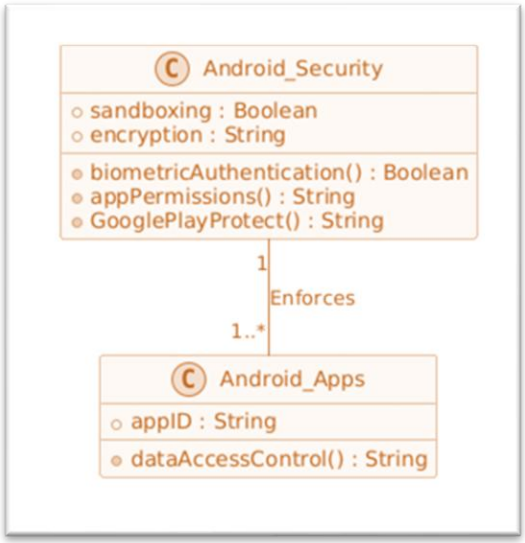
iOS:

- **File Storage:** Employs APFS (Apple File System), optimized for flash and SSD storage.
- **Organization:** Strict sandboxing, each app has isolated storage.
- **Access:** Limited file sharing between apps unless explicitly allowed through app extensions.

Security

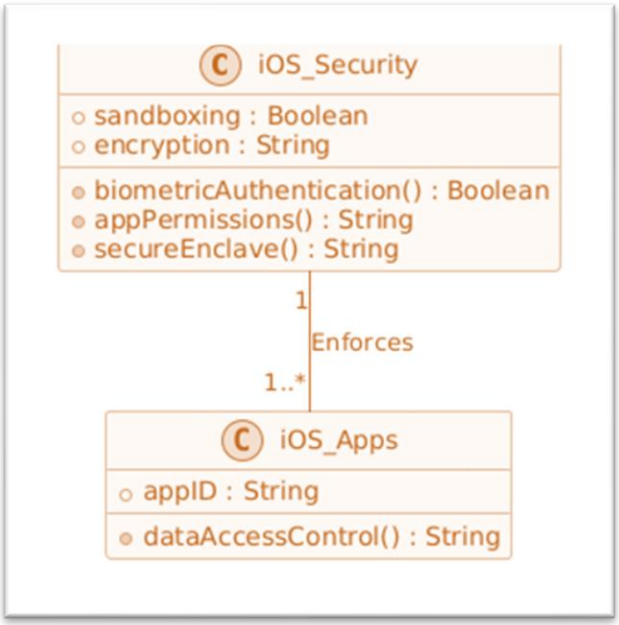
Android:

- **Permissions:** Users grant permissions at runtime, providing finer control.
- **Encryption:** Full-disk encryption is supported, with encryption keys protected by device credentials.
- **Authentication:** Pattern, PIN, and biometric authentication.



iOS:

- **Permissions:** Applications request permissions at installation or runtime, but are more restrictive.
- **Encryption:** End-to-end encryption by default, with hardware-based Secure Enclave for sensitive data.
- **Authentication:** Face ID, Touch ID, and two-factor authentication.



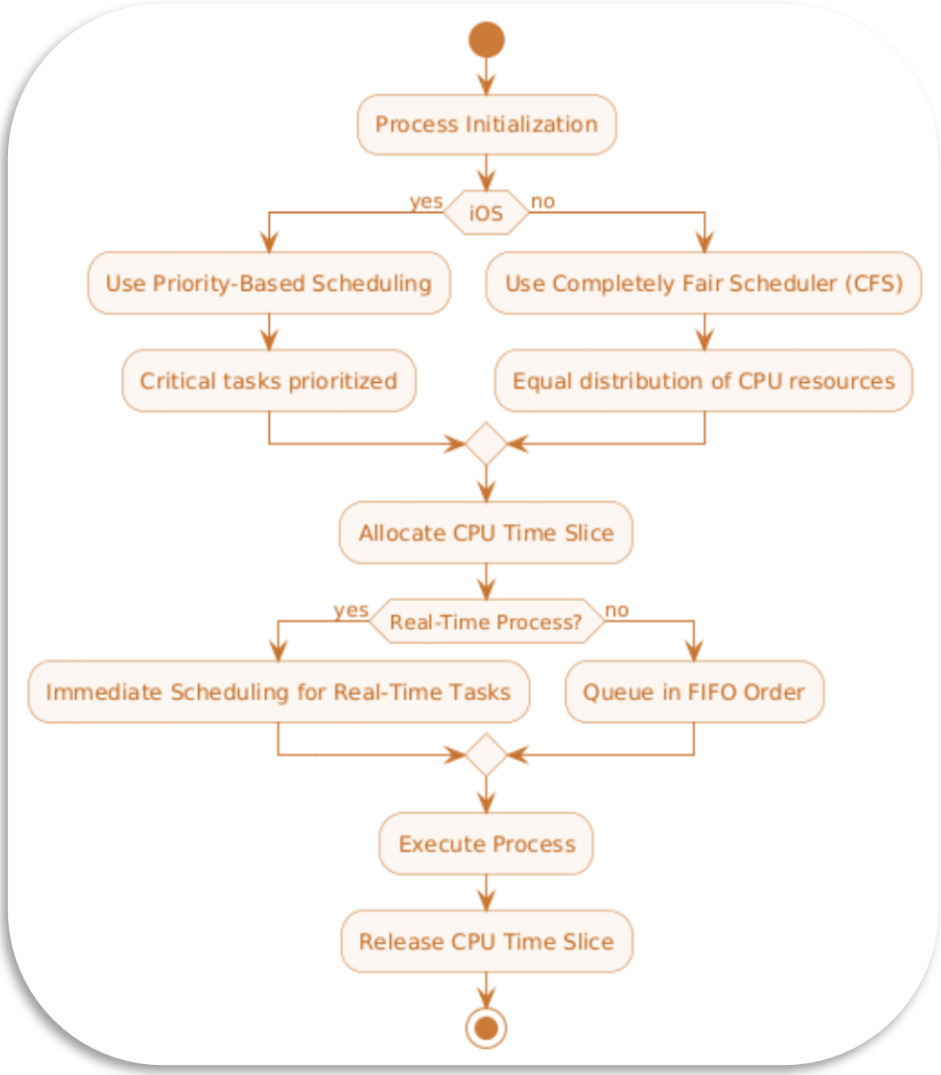
CPU Scheduling

Android:

- **Algorithms:** Uses CFS and real-time scheduling policies (SCHED_FIFO, SCHED_RR).
- **Real-time Processing:** Limited support for real-time processes; relies on Linux capabilities.
- **Handling Multiple Processes:** Prioritizes interactive processes, demoting long-running background tasks.

iOS:

- **Algorithms:** Proprietary algorithms focusing on real-time responsiveness.
- **Real-time Processing:** Superior real-time performance, ensuring smooth UI interactions.
- **Handling Multiple Processes:** Background tasks are strictly controlled, focusing on foreground performance.



Comparison of OS Concepts

OS Concept	Android	iOS
Process Management	Uses Linux kernel, multitasking with LRU cache. Binder IPC for inter-process communication.	Suspends background apps to save battery. Mach ports for secure IPC.
Memory Management	Garbage collection by ART/Dalvik. Paging and shared memory.	Automatic Reference Counting (ARC). Compressed memory, no paging.
File System	ext4 file system, flexible storage. Public and private directories.	APFS, strict app sandboxing. Isolated app storage.
Security	Runtime permissions, customizable. Open-source, but vulnerable to malware.	Hardware-based encryption, strict. Secure Enclave, controlled ecosystem.
CPU Scheduling	CFS and Linux-based scheduling. Background tasks can run freely.	Proprietary real-time scheduler. Limits background tasks.

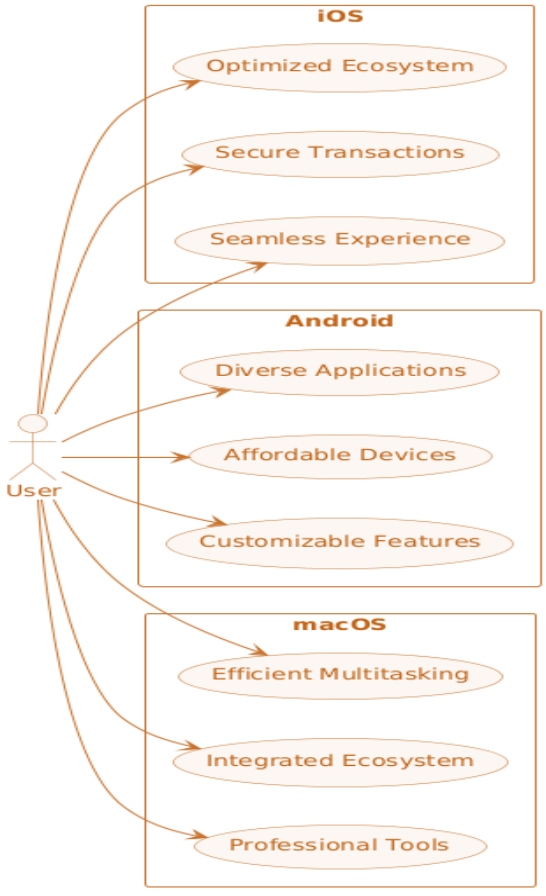
Creative Analogy

Imagine Android and iOS as two cities:

- **Android City** is like a bustling open market where vendors (apps) can set up stalls easily. There is freedom to decorate stalls (customize apps), but this means sometimes pickpockets (malware) sneak in.
- **iOS City** is like a gated community. Only verified vendors (apps) are allowed inside after strict checks. It is safer, but vendors must follow strict design rules, limiting creativity.

Insights and Personal Observations

- **Customization vs. Security:** Android's flexibility is great for tech enthusiasts who love tweaking their devices. However, this openness can introduce risks. iOS, on the other hand, prioritizes security and stability, which appeals to users valuing simplicity and reliability.
- **Performance:** iOS often feels smoother due to efficient CPU scheduling and better memory management. Android, while powerful, can sometimes lag on lower-end devices.
- **User Freedom:** Android offers more choices in file management and app sources, while iOS focuses on controlled environments for a seamless experience.



Conclusion

The comparative analysis highlights the strengths and trade-offs between Android and iOS. Android offers greater flexibility and customization, catering to diverse hardware and user needs, whereas iOS emphasizes security, performance, and seamless integration within its ecosystem. Understanding these core OS concepts helps in selecting the most suitable platform based on user preferences and application requirements.

GitHub Link:

usman11267/Comparative-Analysis-of-Mobile-OS-and-macOS-Through-Operating-System-Concepts