# Evaluation of Operating Systems

RESEARCH REPORT

Mnqobi Lisbon Jeza
230878369 | OPERATING SYSTEMS 2

# Contents

# Declaration

I, (Mnqobi Lisbon Jeza), declare that the contents of this report represent my own unaided work, and that the report has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

| ML Jeza | 30 March 2024 |
| --- | --- |
| Signed | Date |

# 1. Introduction

The aren't much improvement in the industry of Operating Systems in the past few decades. The likes of iOS and Android are few of those popular operating systems that will be discussed in this report about the changes and how they have improved in many aspects differently from each other in recent years.

# 2. Android Evaluation

Google created the mobile operating system Android. It is primarily made for touchscreen mobile devices, such tablets, and smartphones, and is based on the Linux kernel. Android has grown to be one of the most widely used operating systems worldwide, powering a large variety of gadgets made by different companies.

## 2.1. User Interface

Over the past years, Android development has seen significant advancements and changes in user interface (UI) design and development which offers a user interface that is adjustable and supports different home screen layouts, shortcuts, and widgets. Users can add various icon packs, themes, and wallpapers to customize their smartphones. One of the UI interesting changes was the introduction of Dark Mode.

Dark mode has gained popularity as a user interface trend, providing users with a dark colour scheme for easier reading in low light and less eye strain. With the inclusion of built-in support for dark mode in many Android apps and the system itself, users may now choose between light and dark themes according to their preferences.

## 2.2. Process Management and Synchronization

In Android, process management and synchronization in multicore systems are crucial aspects of ensuring efficient utilization of hardware resources and providing a responsive user experience.

A process is an instance of an application that is currently running. To maintain security and isolation, every application operates within its own process. In process management they are three Lifecycles which are Active Lifecycle, Service Lifecycle and Process Lifecycle.

In Active Lifecycle the Android system is responsible for managing the lifespan of activities, which are the user interface elements of an application. Developers can efficiently manage their tasks by using different lifecycle callback methods.

With Service Lifecycle services are parts that operate in the background to manage network transactions or carry out lengthy tasks. Services have a lifespan that developers may control, much like activities do.

And in Process Lifecycle android controls all aspects of a process' lifespan, including its creation, pause, resume, halt, and destruction. The system has the ability to terminate processes in specific circumstances, such low memory.

In Thread Synchronisation multithreading is frequently used in Android to carry out background jobs and lengthy processes. It is important to employ appropriate synchronization techniques to guarantee data consistency and avoid race situations. Atomic variables, locks, semaphores, and synchronized blocks are some of the methods that can be used to accomplish this.

In terms of optimizing performance and preventing problems like UI freezes, ANR (Application Not Responding) errors, and excessive battery usage require proper process management and synchronization. Using tools such as Android Profiler, developers may profile their applications in order to find bottlenecks in performance and minimize resource utilization.

## 2.3. Memory Management

Android employs ART (Android Runtime, starting with Android 5.0) or Dalvik (up to Android 4.4) as a virtual machine to run the bytecode of applications. A garbage collector (GC) is used by Dalvik/ART to manage memory, automatically disposing of memory that is occupied by items that are no longer in use.

Memory leaks happen when objects are referenced even after they are no longer needed by an application, making it impossible for the garbage collector to recover the memory. Static references to objects, callbacks that aren't unregistered, and holding onto references to activities or contexts after they have ended their lifecycle are common sources of memory leaks in Android.

To prevent memory leaks, developers should follow best practices such as keeping in mind the references to situations or actions, particularly while dealing with durable items. Holding references to things that shouldn't impede garbage collection can be accomplished by using WeakReference or other techniques. Making certain that resources, such as files, streams, and cursors, are appropriately terminated after use.

In order to stop listeners and callbacks from keeping references indefinitely, unregister them.

## 2.4. Storage Management

Android handles a variety of storage formats, such as databases, external storage, and internal storage.

The device's internal memory houses internal storage, which is exclusive to the program. The internal storage directory of every Android application is private and can be used to store files that are only accessible by the program itself.

Several programs can access and share external storage. Users can save items including documents, films, and images on their SD card, which is a common external storage option on Android devices. However, the Storage Access Framework (SAF) or the MediaStore API are needed to access the majority of external storage locations as of Android 10 (API level 29).

Also, Android comes with an integrated relational database management system called SQLite. Structured data, including application, user, and cached data, is frequently stored in SQLite databases.

## 2.5. File System

The hierarchical structure of the Android file system is similar to that of other Unix-based operating systems. It includes storage that is both internal and external. Every application has its own internal storage, which is located in /data/data// and is solely accessible by that particular program. On the other hand, external storage is shared by several apps; this usually includes the SD card on the device and other shared destinations.

The APK contains assets, which are raw files such as sounds and graphics. SQLite databases are used to store organized data and can be found in external and internal storage. A common interface for accessing structured data across apps is provided by content providers. Developers may effectively handle file access, data storage, and data security and integrity within Android applications by utilizing these components and APIs.

## 2.6. Security and Protection

To protect devices, apps, and user data from a wide range of possible threats, Android security takes a multipronged approach. Android's fundamental permission system allows apps to access certain device features and data only with the user's permission. Users are given control over their data privacy and transparency is ensured by this technique.

Additionally, every app on Android is isolated by its sandboxed environment, which limits the consequences of security breaches and stops unauthorized access to system resources. Android offers both file-based and full-disk encryption to prevent unwanted access in the event of device loss or theft.

Encryption is essential for safeguarding sensitive user data. To further support the integrity of the app ecosystem, Google Play Protect also checks apps for malware and other security threats. Continual protection updates are essential for fixing known vulnerabilities and maintaining device security against new dangers.

For businesses using Android smartphones in corporate environments, Android Enterprise also offers improved management tools and security measures. Android aims to provide a safe and reliable platform for its customers by raising user knowledge and encouraging recommended practices, such as obtaining apps from reputable sources and updating devices.

# 3. iOS Evaluation

Apple Inc. created the iOS mobile operating system specifically for use with its hardware, such as iPhones, iPads, and iPod Touch devices. With its svelte form factor, simple UI, and easy connection with Apple's ecosystem, iOS has grown to be one of the most well-liked and extensively utilized mobile systems globally.

## 3.1. User Interface

Smooth design, simple interactions, and a unified visual language across all of Apple's devices—iPhones, iPads, and iPod Touches—are what define the iOS user interface (UI). The iOS UI consists of key elements such as Home Screen, Dock, Control Center, Notifications, and others.

Control Center: Offers fast access to frequently used features and settings. To access the Control Center, users can swipe up from the bottom of the screen (on older devices) or down from the top-right corner of the screen (on newer devices). The screen brightness, volume controls, Wi-Fi, Bluetooth, Airplane Mode, Do Not Disturb, and shortcuts to frequently used apps, such as the flashlight and camera, are all included.

## 3.2. Process Management and Synchronization

Process control and synchronization are essential in iOS to guarantee that apps run smoothly, maximize system resources, and preserve a responsive user experience. iOS has advanced process management features, such as multitasking support, to facilitate smooth program switching while effectively allocating system resources. Applications are carefully controlled throughout their lifecycle; iOS decides whether to start, stop, continue, or end them depending on system parameters and user inputs. Strong APIs like Grand Central Dispatch (GCD) and Operation Queues help ensure thread synchronization, enabling developers to carry out tasks sequentially or concurrently while maintaining data integrity and preventing race situations.

Apps may connect and share data easily because to technologies like URL schemes and app extensions, which promote inter-process communication (IPC). Furthermore, iOS gives developers the ability to create cloud-based applications with capabilities like data syncing and conflict resolution by offering frameworks for data sharing and synchronization including CloudKit and Core Data. In general, iOS process management and synchronization are crucial elements of developing apps since they provide the best possible performance, responsiveness, and user experience on Apple's mobile devices.

## 3.3. Memory Management

Memory management is an essential part of developing applications for iOS in order to guarantee stable performance and effective use of resources. The main method used is Automatic Reference Counting (ARC), which inserts memory management instructions during compilation to automatically manage an object's lifecycle. To control object associations, developers use strong and weak references. Strong references maintain objects' lifetime whereas weak references enable deallocation when it's no longer required, preventing retention cycles and memory leaks.

Appropriate view controller lifecycle management guarantees timely resource deallocation, including views and observers, while lazy loading strategies save memory by loading resources only when needed. Memory profiling tools such as Instruments help developers find and fix memory-related problems, encourage efficient memory use, and transfer of ownership. iOS applications may preserve stability, responsiveness, and peak performance on a variety of Apple devices by following best practices and utilizing efficient memory management techniques.

## 3.4. Storage Management

Storage management in iOS is crucial to guaranteeing the accessibility and integrity of user data as well as effective use of device storage. iOS offers a number of storage management tools, such as data persistence strategies, external storage, and internal storage. Applications generally employ internal storage to hold user-specific information, preferences, and temporarily stored files in the sandboxed directory of the application. Users can save and access files across many devices with external storage, including iCloud Drive or third-party cloud storage services. Applications are able to safely save user preferences and structured data thanks to iOS frameworks like Core Data and UserDefaults that enable data persistence.

It is recommended that developers follow best practices for effective storage management, which include controlling cache expiration policies, optimizing file sizes, and utilizing iCloud integration for easy data syncing between devices. iOS users may also manage how much storage they use by utilizing settings like iCloud Storage Management, which offers information on how much storage is used as well as ways to maximize capacity by organizing media files, backups, and app data. Through the utilization of iOS storage features and the implementation of efficient storage management strategies, developers may guarantee optimal speed, data integrity, and user happiness in their applications.

## 3.5. File System

The fundamental framework in charge of handling and organizing data on Apple's mobile devices is called the iOS file system. Each iOS app operates in a sandboxed environment with a specified directory structure, protecting user privacy and security by limiting access to system files and other apps.

Starting with the root directory ("/"), the file system hierarchy consists of folders like "/Applications" (which holds installed software bundles), "/Documents" (which holds user-generated material), "/Library" (which holds app-specific items like preferences and caches), and "/tmp" (which holds temporary storage). Every iOS application is bundled into a bundle that includes resources, information, and executable code. This bundle is stored in the "/Applications" directory and is signed by Apple to ensure its integrity.

Apps are able to manage structured data more effectively because to data persistence mechanisms such as property lists, SQLite databases, and Core Data. File coordination APIs guarantee safe access to and modification of shared files, while integration with iCloud facilitates easy data and file synchronization across devices. All things considered, the iOS file system offers a safe and well-organized setting for managing app data, guaranteeing data confidentiality, integrity, and smooth syncing within Apple's ecosystem.

## 3.6. Security and Protection

iOS prioritizes security and safety, protecting against different threats and placing a high value on user privacy. The strong design of iOS, which includes features like Secure Boot, which makes sure that only reliable firmware and applications are loaded at startup to prevent tampering and unauthorized alterations, is essential to the system's security.

Furthermore, iOS uses advanced encryption methods to safeguard user data while it's in transit and at rest, making sure that private data is unreadable without the right decryption key. By ensuring that apps adhere to Apple's tight security requirements and reducing the likelihood of infection, the App Store's rigorous review process improves the platform's overall integrity. iOS also includes Touch ID and Face ID biometric authentication techniques, giving users easy-to-use yet safe ways to unlock their smartphones and verify transactions.

In addition, the platform receives frequent security updates to strengthen its defenses against new attacks and fix known vulnerabilities. iOS has a thorough security framework that puts user trust and secrecy first with features like App Transport Security, which ensures safe connections between apps and servers, and sandboxing, which isolates programs from the system and each other.

## 4. Conclusion

In summary, the comparison of iOS and Android over the last few years reveals notable developments and additions to both operating systems. iOS has cemented its standing as an intuitive platform that seamlessly integrates with Apple's ecosystem and boasts strong security measures and an easy-to-use UI. In the meantime, Android has developed into a flexible and popular operating system that works with a broad variety of devices and provides a wealth of customization choices.

Both systems are expected to push limits going forward, with iOS probably concentrating on improving security and privacy features while honing its seamless user experience. Conversely, Android might enhance system optimization for a wider range of hardware, broaden its ecosystem integration, and innovate in fields like augmented reality and AI-driven features. In the end, it is anticipated that the rivalry between iOS and Android will spur additional innovation, giving consumers access to ever-more-advanced features and an enhanced mobile computing environment.

# 5. References


- Bhatia, T., & Kaushal, R. (2017). Malware detection in Android based on dynamic analysis. *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*. https://doi.org/10.1109/cybersecpods.2017.8074847
- Data storage and SQLite operations. (2016). *Mobile Applications Development with Android*, 139-160. https://doi.org/10.1201/9781315367576-17
- Downey, E. (2016). Ios UI and storyboards. *Practical Swift*, 111-143. https://doi.org/10.1007/978-1-4842-2280-5_7
- Eve, M. P. (2011). Getting a custom sync service and adapter to show up under "Data and synchronization" on Android. https://doi.org/10.59348/9jxvd-6t191
- File system and data storage. (2011). *iPhone and iOS Forensics*, 55-77. https://doi.org/10.1016/b978-1-59749-659-9.00003-1
- Friedman, R., & Sainz, D. (2016). File system usage in Android mobile phones. *Proceedings of the 9th ACM International on Systems and Storage Conference*. https://doi.org/10.1145/2928275.2928280
- Interactive graphical user interface for performance and flight dynamics analysis of Battery-Powe... (2023). https://doi.org/10.2514/6.2023-4394.vid
- Qian, J., & Zhou, D. (2016). Prioritizing test cases for memory leaks in Android applications. *Journal of Computer Science and Technology*, *31*(5), 869-882. https://doi.org/10.1007/s11390-016-1670-2
- Relan, K. (2016). Ios security toolkit. *iOS Penetration Testing*, 73-96. https://doi.org/10.1007/978-1-4842-2355-0_5
- Sindhu, D., Sangwan, A., & Singh, K. (2015). An approach to process management using process synchronization. *International Journal of Computer Applications*, *128*(7), 18-22. https://doi.org/10.5120/ijca2015906604