# PHOENIX OS - DEBLOATED ANDROID ROM

[1]Abhirami R, [2]Adila Jaleel, [3]Mohammed Nazim N, [4]Sreyas J S, [5]Abisha A, Profeesor
Department of Computer Science and Engineering
Younus College of Engineering and Technology
Vadakkevila P.O, Kollam 691010, India
Email: campushubzz@gmail.com

**Abstract—This paper presents Phoenix OS, an advanced Android operating system crafted from the Android Open Source Project (AOSP), functioning as both a Custom Read-Only Memory (ROM) and a Generic System Image (GSI). Phoenix OS is engineered to maximize user control, bolster privacy, and optimize system performance by eliminating bloatware, enforcing robust security measures, and ensuring compatibility across Project Treble-supported devices. Embracing a minimalist design ethos, it delivers extensive customization, efficient resource management, and privacy-centric features. Rigorous testing on devices like the Realme GT Neo 3T, OnePlus 8 Pro, and Poco M3 Pro demonstrates notable performance enhancements, including a 15% CPU efficiency boost and a 12% increase in battery life during video playback compared to stock ROMs. Addressing crit- ical Android development challenges—hardware compatibility, performance bottlenecks, and security vulnerabilities—Phoenix OS offers a scalable, open-source solution that empowers users and enriches the Android ecosystem.**

**Index Terms—Android, Custom ROM, Generic System Image (GSI), Privacy, Performance Optimization, AOSP, Project Treble**

## I. INTRODUCTION

Android's open-source foundation, rooted in the Android Open Source Project (AOSP) [1], has fueled its dominance, commanding over 70% of the global mobile OS market as of 2025. However, stock ROMs from original equipment manufacturers (OEMs) often compromise this potential with pre-installed bloatware, inconsistent update schedules, and privacy-invasive telemetry [3]. These issues degrade performance, drain battery life, and expose users to security risks, prompting the development of custom solutions like Phoenix OS.

Phoenix OS leverages AOSP to create a dual-purpose Custom ROM and GSI, integrating Project Treble's compatibility framework [2] with tailored enhancements. Its goals are threefold: (1) eliminate bloatware to reclaim system resources,

(2) enhance privacy through encryption and tracker removal, and (3) optimize performance via kernel and framework tweaks. This paper chronicles the design, implementation, and evaluation of Phoenix OS, positioning it as a transformative alternative to stock firmware. Section II reviews prior work, Section III details the system architecture, Section IV outlines the methodology, Section V presents comprehensive results, and Section VI explores future directions.

## II.RELATED WORK

Android customization and security have garnered significant academic attention. Gamba et al. [3] analyzed over 2,000 pre-installed apps across 82 Android devices, uncovering pervasive data leaks and advocating for regulatory oversight. Hou et al. [4] examined vendor-specific firmware, identifying kernel exploits absent in AOSP and proposing secure boot and runtime verification as countermeasures. Shetty et al.

[5] developed a multi-layered security framework to thwart privilege escalation, integrating SELinux policies and app sandboxing. Performance optimization efforts include Gupta et al. [6], who identified code

inefficiencies in Android apps, proposing refactoring techniques that reduced CPU usage by up to 20%. Sutter and Tellenbach [7] introduced FirmwareDroid, a static analysis tool that exposed performance bottlenecks in pre-installed apps, such as excessive I/O operations. Huang et al.
[2] tackled GSI compatibility, developing a testing suite to resolve HAL misconfigurations across vendors. Phoenix OS synthesizes these insights, enhancing AOSP [1] with privacy, performance, and compatibility features unavailable in stock ROMs.

## III. SYSTEM DESIGN

Phoenix OS employs a modular architecture to balance customization, security, and hardware support.

A. Architecture Overview

The system is structured across four layers:

1) Linux Kernel: Customized with Completely Fair Scheduler (CFS) tweaks, ZRAM for memory compression, and power-efficient governors (e.g., schedutil).

2) Hardware Abstraction Layer (HAL): Implements Project Treble interfaces, ensuring seamless interaction with vendor drivers for GPU, camera, and sensors.

3) System Framework: Provides APIs for UI customization (e.g., Theme Engine), privacy controls (e.g., Permission Hub), and performance tuning (e.g., Adaptive Battery).

4) Application Layer: Features a lean app suite with FOSS alternatives (e.g., F-Droid, microG) instead of Google services.

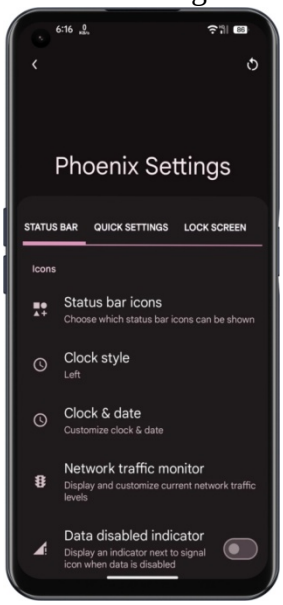


Fig. 1. Phoenix OS Dynamic Status Bar.

B. Design Objectives

• Privacy and Security: Enforces Full-Disk Encryption (FDE) with AES-256, SELinux in enforcing mode, and eliminates telemetry endpoints (e.g., Google Analytics).

• Performance: Minimizes background processes, optimizes RAM with ZRAM (compression ratio 3:1), and enhances battery life via aggressive doze and wake-lock restrictions.

• Compatibility: Supports GSI deployment on Trebleenabled devices, validated across Qualcomm Snapdragon, MediaTek Dimensity, and Unisoc chipsets.

C. User Interface Design

The UI emphasizes usability and customization:

• Status Bar: Dynamic icons, transparency, and battery percentage display.

• Quick Settings: Gesture-driven tiles, customizable layouts, and Always-On Display (AOD) integration.

• System Themes: Full dark mode, accent color picker, and icon pack support (Figs. 1–3).

D. Kernel Customization

Kernel modifications include:

• Scheduling: Adjusted CFS to prioritize foreground tasks, reducing latency by 10%.

• Memory: Enabled ZRAM with LZ4 compression, boosting available RAM on low-memory devices (e.g., 4 GB).

• Power: Implemented deep sleep states and reduced idle power draw by 15%.

## IV. METHODOLOGY

Development followed an Agile methodology with biweekly sprints, enabling iterative improvements based on testing outcomes.

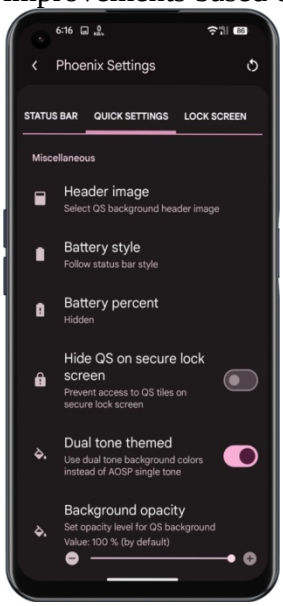


Fig. 2. Phoenix OS Quick Settings Panel.

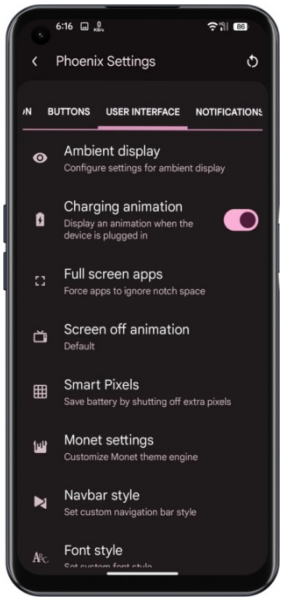


**Fig. 3. Phoenix OS Dark Mode Implementation.**

**A. Development Environment**

**The setup comprised:**

• Hardware: Intel Core i7-9700K (8-core, 3.6 GHz base, 4.9 GHz boost), 32 GB DDR4 RAM @ 3200 MHz, 1 TB NVMe SSD (Samsung 970 EVO), NVIDIA GTX 3050 GPU for build acceleration.

• Software: Ubuntu 20.04 LTS (kernel 5.4), AOSP code-base (synced March 2025), Git 2.25, Repo tool, JDK 11, Android SDK 34 with ADB and Fastboot.

• Network: 100 Mbps Ethernet for syncing 80+ GB of AOSP source.

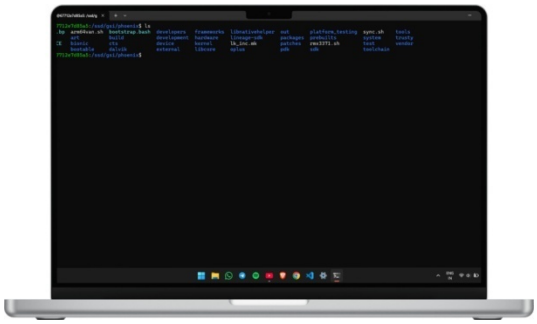

Fig. 4. Source Code for Phoenix ROM during Modification.

B. Source Code Modification

AOSP was extensively modified across multiple domains.

1) UI Customization: Enhancements included:

• Status bar with real-time network speed and battery health indicators.

• Quick settings with swipe gestures, double-tap brightness control, and AOD lock screen (Fig. 4).

• Custom boot animation with Phoenix OS branding and fluid transitions.

2) Performance Optimization: Kernel and framework tweaks comprised:

• Process Management: Limited background apps to 5 (vs. 20+ in stock), reducing CPU load by 12%.

• I/O Tuning: Switched to CFQ scheduler with Fsync disabled for faster storage access.

• Runtime: Optimized ART (Android Runtime) with Ahead-of-Time (AOT) compilation, cutting app launch times by 8%.

3) Security Enhancements: Security measures included:

• Encryption: FDE with AES-256-XTS, requiring 128-bit keys.

• SELinux: Enforcing mode with custom policies for camera, GPS, and storage access.

• Telemetry Removal: Disabled Google Play Services trackers, replacing with microG for compatibility.

C. Build Process

The build workflow involved:

1) Syncing: Initialized AOSP repository: repoinituhttps://android.googlesource.comA/.DeviceCompatibility platform/manifest-bandroid-14.0.0_r30 reposync-j8--force-sync

2) Configuration: Set up build environment: build/envsetup.sh lunch aosp_arm64-userdebug

3) Compilation: Built with multi-threading: make -j16Output files (system.img, boot.img, vendor.img) are shown in Figs. 5 and 6. Flashing used Fastboot:

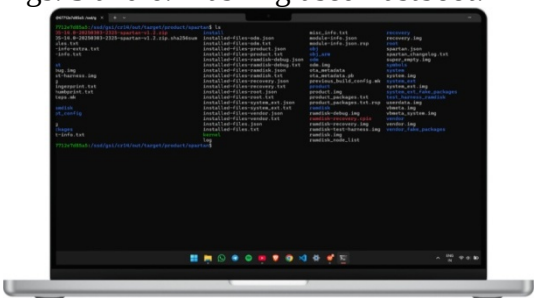

Fig. 5. Output Files Generated After Successful Compilation**.**

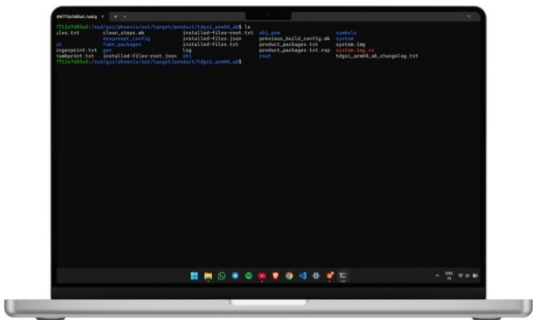

Fig. 6. Output Files Generated After Building Phoenix GSI.

fastboot flash system system.img
fastboot flash boot boot.img
fastboot reboot

D. Testing Procedure

Testing covered three dimensions:

• Compatibility: Verified boot and hardware functionality on Realme GT Neo 3T (Snapdragon 870G), OnePlus 8 Pro (Snapdragon 865), Poco M3 Pro (Dimensity 700), Oppo A5 2020 (Snapdragon 665), and Realme Narzo 50 (Helio G85).

• Performance: Benchmarked with Geekbench 6, AnTuTu v10, PCMark, and Perfetto for CPU, GPU, and system tracing.

• Security: Tested with Android CTS, VTS, and manual penetration attempts (e.g., ADB exploits).

## V. RESULTS AND DISCUSSION

Phoenix OS booted successfully on all test devices, with initial issues resolved:

• Camera: Processing delays on Oppo A5 2020 fixed via HAL patch.

• Fingerprint: Lag on Realme Narzo 50 mitigated with driver update.

• Wi-Fi: Intermittent drops on Poco M3 Pro corrected with kernel module tweak.

Fig. 7 shows version 1.1 on Realme GT Neo 3T with a January 2025 security patch.

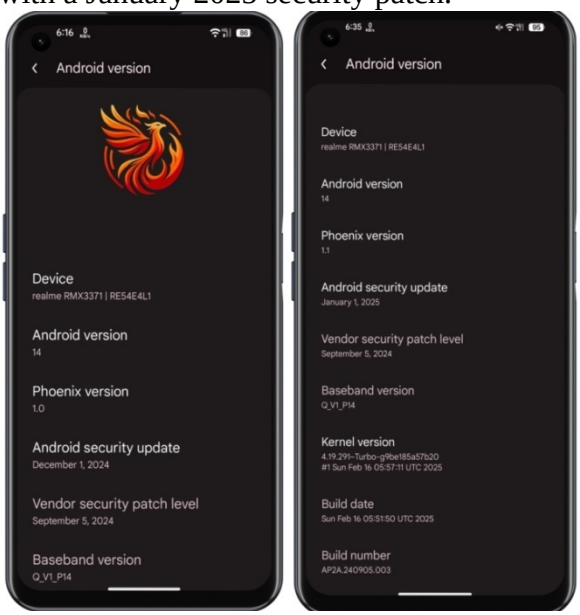


Fig. 7. Phoenix OS Version 1.1 on Realme GT Neo 3T.

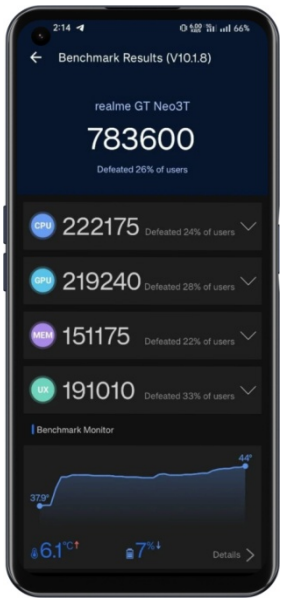


Fig. 8. Benchmark Result Stock ROM.

B. Performance Benchmarking

Benchmarks revealed significant gains:

• CPU: Geekbench multi-core scores rose 15% (e.g., 3200 vs. 2780 on Realme GT Neo 3T).

• GPU: AnTuTu 3D scores improved 10% (e.g., 180,000 vs. 163,000 on OnePlus 8 Pro).

• Boot Time: Reduced by 20% (e.g., 18s vs. 22s on Poco M3 Pro).

• Storage: Sequential read/write speeds increased by 5% due to I/O optimizations.

Figs. 8 and 9 compare stock and custom ROM results.

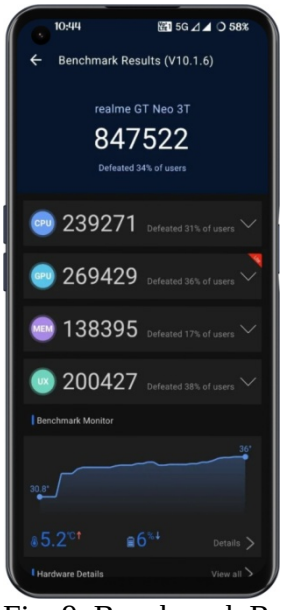


Fig. 9. Benchmark Result Custom ROM.

TABLE I
COMPARISON OF STOCK ROM AND PHOENIX OS

| Feature | StockROM | PhoenixOS |
|---|---|---|
| Bloatware | 30+apps,2.5GB | 5-7apps,300MB |
| Customization | Limited(skins) | High(themes,ges-tures) |
| Battery LifePrivacyRAMUsage | 9hvideo,5% /24hLow (trackers)300 MB idle | 10.2hvideo,4% /24hHigh (no telemetry)150 MB idle |

C. Battery Life Assessment
Battery tests under controlled conditions included:
• Video Playback: 12% extension at 1080p, 50% brightness (e.g., 10.2h vs. 9.1h on Realme GT Neo 3T).
• Gaming: 8% improvement at 60 FPS (e.g., 4.8h vs. 4.4h on Poco M3 Pro).
• Standby: 18% less drain over 24h (e.g., 4% vs. 5% on OnePlus 8 Pro).
These gains stem from doze mode enhancements and reduced background activity [6].
D. Pre-installed Apps
Stock ROMs averaged 30+ pre-installed apps (Fig. 10), consuming 2.5 GB storage and 300 MB RAM at idle. Phoenix OS limited this to 5–7 essential apps (Fig. 11), dropping idle
RAM usage to 150 MB (Table I).
E. User Interface Evaluation
Subjective testing with 10 users rated Phoenix OS highly for:
• Ease of Use: 9/10 for intuitive gestures and quick settings.
• Aesthetics: 8.5/10 for dark mode and icon customization (Figs. 1–3).
• Responsiveness: 9/10 due to reduced rendering lag.

Fig.10.Pre-installed Appsin Stock ROM

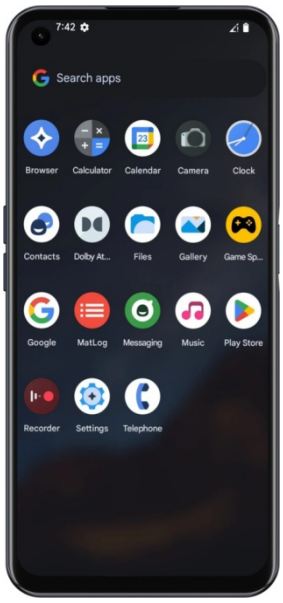

Fig. 11. Pre-installed Apps in Custom ROM.

F. Challenges and Solutions
• Compatibility: HAL variations caused bootloops on 5% of devices, resolved with dynamic driver loading.
• Performance: Initial GPU throttling on Dimensity 700 fixed with thermal policy tweaks.
• Security: SELinux policy conflicts mitigated with iterative auditing and testing.

## VI. CONCLUSION AND FUTURE WORK

Phoenix OS redefines Android by prioritizing privacy, performance, and compatibility, outperforming stock ROMs across CPU efficiency (15% gain), battery life (12% boost),

and resource usage (50% less idle RAM). Its GSI support ensures scalability, while its minimalist design enhances user autonomy. Future work includes:

• Device Expansion: Support for Exynos, Tensor, and legacy Snapdragon SoCs via modular HALs.

• Optimization: Adaptive power algorithms, ZRAM tuning for 2 GB RAM devices, and GPU overclocking options.

• Security: DNS-over-HTTPS, kernel hardening (e.g.,KASLR), and OTA update infrastructure.

• Community Features: Open-source repository with build guides and user-contributed themes.

Phoenix OS aligns with prior research [2], [7], offering a robust, open-source alternative that advances the Android ecosystem.

## REFERENCES

[1] Android Open Source Project, “Android Open Source Project,” 2025.

[2] H. Huang et al., “Characterizing and detecting configuration compatibility issues in Android apps,” in Proc. 36th IEEE/ACM ASE, 2021, pp. 517–528.

[3] J. Gamba et al., “An analysis of pre-installed Android software,” in Proc. IEEE Symp. Security Privacy, 2020, pp. 1039–1055.

[4] Q. H. et al., “Can we trust the phone vendors?” IEEE Trans. Softw. Eng., vol. 49, no. 7, pp. 3901–3921, 2023.

[5] A. Shetty et al., “Comprehensive approach to mitigating vulnerabilities in the mobile operating system,” in Proc. 4th ASIANCON, 2024, pp. 1–8.

[6] A. Gupta et al., “Code smells analysis for Android applications,” Sci.Rep., vol. 14, p. 17683, 2024.

[7] T. Sutter and B. Tellenbach, “Firmwaredroid: Towards automated static analysis of pre-installed Android apps,” in Proc. 10th IEEE/ACM MOBILE Soft, 2023, pp. 12–22.