LEGACY PROJECT

Product Requirements Document

Date: February 20, 2026

Status: Initial Release

Author: DT7A1

GitHub: github.com/DT7A1/LegacyProject

License: MIT — Open Source

An open-source, community-driven platform for the preservation of classic gaming heritage through On-Device Static Recompilation and High-Level Emulation — a permanent gift to the global community.

—. Table of Contents

1. Executive Summary
2. Project Scope & Hardware Support
3. System Architecture
4. Graphics Architecture & GPU Strategy
5. Critical Technical Challenges
6. Functional Requirements
7. Non-Functional Requirements
8. Security & Legal Compliance
9. Development Strategy
10. Roadmap
11. Hardware Tier Classification
12. Approval

1. Executive Summary

Vision

To establish the Ultimate Preservation Platform for classic gaming hardware using modern open-source technologies to achieve native performance through On-Device Static Recompilation, surpassing traditional emulation in both performance and legal standing.

Mission

Enable users to run legally owned classic game libraries on modern hardware with high fidelity, full legal compliance, and community ownership. Released under MIT License as a permanent gift to the global community.

Core Philosophy

- 100% open-source — no proprietary components from core engine to UI.
- On-Device compilation only — all recompiled binaries are device-bound and non-transferable.
- No copyright infringement — no BIOS distribution, no cryptographic circumvention, no ROM hosting.
- Community ownership — the project belongs to everyone, serves no commercial interest.
- Preservation over perfection — long-term heritage conservation is the primary goal.

What This Project Is NOT

Not a traditional emulator. Not software porting. It is an On-Device Native Runtime Compiler: reads the user's legally owned game binary, recompiles it locally via LLVM using Hybrid Static+Micro-JIT strategy, and executes it as a device-bound native library. Output is cryptographically signed to the originating device and cannot be extracted or distributed.

2. Project Scope & Hardware Support

2.1 Supported Hardware — Tier Classification

Tier classification is determined primarily by GPU abstraction complexity. Static Recompilation via LLVM resolves most CPU architecture differences.

Tier	Platform	CPU Architecture	GPU	Priority
1	Sega Dreamcast	Hitachi SH-4 (RISC)	PowerVR NEC CLX2	FIRST — Launch Target
1	PlayStation 1	MIPS R3000	GPU (Custom)	High
1	Nintendo 64	MIPS R4300	RDP (Custom)	High
1	Nintendo DS	ARM7 + ARM9 (Native)	Custom 2D/3D	High
1	PlayStation Portable	MIPS R4000	GE (Custom)	High
2	GameCube	PowerPC Gekko	Flipper GPU	Medium
2	PlayStation 2	MIPS R5900 (EE)	Graphics Synthesizer	Medium
2	Xbox Original	Intel Celeron x86	NVidia GeForce 3	Medium
2	Nintendo Wii	PowerPC Broadway	Hollywood GPU	Medium
3	Nintendo Wii U	PowerPC Espresso	Latte GPU	Deferred
3	Nintendo 3DS	ARM11 + ARM9	PICA200 (DMP)	Deferred
4	Sega Saturn	SH-2 Dual	VDP1 + VDP2	Long-term
4	Xbox 360	PowerPC Xenon	Xenos GPU (Custom)	Long-term
4	PlayStation 3	Cell/PowerPC + SPE x7	RSX (NVidia)	Long-term

2.2 Target Platforms

- LegacyDroid: Android OS (Smartphones, Tablets, Foldable Devices) — API Level 31+.
- LegacyPC: Windows (x86_64 and ARM64 / Snapdragon X Elite), Linux (Desktop and Raspberry Pi 4/5).

2.3 Permanent Exclusions

- Modern Windows PC games (NovaDroid project suspended).
- Apple ecosystems (iOS / macOS) — JIT restrictions and philosophical position.
- BIOS files or copyrighted ROMs — relies on HLE or user-provided files.
- Official server connectivity (PSN / Xbox Live).
- Generation 8+ hardware (PS4, Xbox One, Nintendo Switch) — commercially active, permanently excluded.

3. System Architecture

3.1 Three-Layer Architecture

LegacyCore is the fully headless, platform-agnostic engine — no awareness of any OS. LegacyDroid and LegacyPC are platform shells wrapping LegacyCore. This mirrors Linux Kernel and LLVM design philosophy.

3.2 LegacyCore — Platform-Agnostic Engine

Language: C++23 — core logic, recompilation pipeline, hardware abstraction.

Compiler Backend: LLVM — optimized machine code for ARM64 and x86_64. Each platform needs only a dedicated LLVM frontend; backend is shared across all targets.

Recompilation Strategy : Hybrid: Static Recompilation (AOT via LLVM) for ~95% of game code + Micro JIT engine activated only when Self-Modifying Code is detected at runtime.

Graphics API: Vulkan — exclusive unified translation target for all GPU HLE layers.

Safety Layer: Rust — ISO parsing, P2P network, cryptographic device binding.

Output Format: Per-game device-bound shared libraries (.so / .dll). Cryptographically signed to originating hardware fingerprint + optional user account.

3.3 Logic-Assets Separation

The compiler translates game logic only into a compact shared library. Media assets (textures, audio, video) are NOT embedded — the native library reads them directly from the user's original ISO at runtime.

- Significantly reduces compilation time and storage footprint.
- Strengthens legal position — the original ISO remains the authoritative source.
- Logic patches can be distributed without requiring asset re-extraction.

3.4 Frontend & User Interface

Framework: Flutter (Dart) — shared UI codebase across Android and Desktop.

Rendering Engine: Impeller — consistent 120fps UI performance.

Native Binding: dart:ffi — zero-overhead direct communication between Dart and LegacyCore.

Display Strategy: Zero-Copy Texture Sharing — Vulkan framebuffer direct to Flutter widget tree.

Launcher Model: Unified game library with cover art, metadata, compatibility status, device-bound library links.

4. Graphics Architecture & GPU Strategy

4.1 The Core GPU Challenge

Vulkan unifies modern GPU communication but does not resolve understanding what legacy GPUs did internally. Each platform requires its own reverse-engineered HLE translation layer. This is the primary engineering challenge of the entire project.

4.2 HLE as Primary GPU Strategy

HLE intercepts high-level rendering commands and translates them directly to Vulkan calls, bypassing low-level hardware simulation. The final output is a Native Vulkan Binding library speaking directly to the modern GPU with no emulation overhead at runtime.

- HLE covers approximately 80-90% of titles per platform without per-game intervention.
- Remaining edge cases addressed via Cloud Game Configs on a per-title basis.
- HLE carries no legal equivalence to BIOS emulation.
- Vulkan translation implemented independently of any proprietary GPU firmware.

4.3 Per-Platform GPU Abstraction & Known Edge Cases

Platform	Legacy GPU	Primary HLE Challenge	Known Edge Case Risk	Complexity
Dreamcast	PowerVR NEC CLX2	Tile-Based Deferred Rendering	Undocumented modifier volumes	Medium-High
PlayStation 1	Custom GPU	Fixed-function pipeline, dithering	PGXP perspective correction exploits	Low-Medium
Nintendo 64	SGI RCP / RDP	Microcode-based rendering	Custom microcode programs per game	Medium
Nintendo DS	Custom 2D/3D	Dual-screen compositing	3D-to-2D layer blending edges	Low
PSP	GE (Graphics Engine)	Compressed textures, morphing	CLUT palette manipulation	Low-Medium
GameCube	ATI Flipper	TEV texture combiner stages	Multi-stage TEV combiner sequences	Medium
PlayStation 2	Graphics Synthesizer	VRAM framebuffer reads	Direct VRAM framebuffer sampling	Medium-High
Xbox Original	NVidia GeForce 3	Closest to modern GPU model	Pixel shader v1.1 quirks	Low
Wii	ATI Hollywood	Extension of Flipper	EFB copy-to-texture operations	Medium

4.4 AMD FSR 3.0

- Upscaling: spatial upscaling to 1080p and 4K output.
- Frame Generation: optical flow AI interpolation — 30fps content to 60fps+ visually.
- Supports 120Hz and 144Hz displays — legacy titles can far exceed original framerate specs.

5. Critical Technical Challenges

Critical engineering challenges that require explicit solutions before or during Phase 1. These require explicit engineering solutions before or during Phase 1.

5.1 Self-Modifying Code (SMC) [CRITICAL]

Static Recompilation assumes code immutability. Some Tier 1 games on PS1 and N64 used Self-Modifying Code at runtime to optimize performance. SMC is fundamentally incompatible with pure AOT Static Recompilation.

Risk Level: HIGH — affects unknown percentage of Tier 1 titles

Affected Platforms: PS1 (MIPS R3000), N64 (MIPS R4300), potentially Dreamcast (SH-4)

Solution: Hybrid Recompilation + Micro JIT :

- Static Recompilation (AOT via LLVM) for ~95% of game code — full native performance.
- Micro JIT engine embedded — activates only on SMC detection, no penalty for normal execution.
- SMC detection via write-protection monitoring on compiled code memory regions.
- Micro JIT handles affected regions dynamically; returns to static execution when region stabilizes.
- Mirrors Apple Rosetta 2 hybrid strategy for dynamic code handling.

5.2 Undocumented Hardware Behavior

Some games relied on GPU behaviors never officially documented. These were discovered empirically by developers and exploited for visual effects. HLE cannot cover what is unknown until a failure is observed.

Mitigation: Cloud Game Configs acts as the live patch database for discovered undocumented behaviors. Community Compatibility Database surfaces these cases. Per-game HLE patches are distributed as config updates — no app update required.

5.3 Multi-threading Strategy

Legacy games designed for single-threaded execution. LLVM auto-vectorization alone is insufficient to parallelize game logic.

Strategy: Preserve single-threaded execution for game logic. Parallelize independent subsystems: audio pipeline, shader compilation, asset streaming, and network stack run on separate threads. Game logic thread pinned to a performance core.

5.4 Device Binding v2 — User Experience Problem Resolved

Strict hardware-fingerprint-only binding creates unacceptable UX: a broken phone means losing all compiled game libraries. This is a critical user experience issue that must be resolved.

Solution: Dual-Layer Binding System:

- Primary: cryptographic hardware fingerprint — used for offline play (existing approach).
- Secondary: optional User Account — compiled libraries additionally linked to user account.
- Account migration: user can re-authorize libraries on a new device after identity verification.
- Maximum 3 simultaneous active devices per account — prevents abuse, allows legitimate multi-device use.

- Legal position maintained: the same user is always the consumer — no third-party redistribution possible.

5.5 Redump Checksum Verification

Optional ISO integrity verification against the Redump database of known-good disc dumps to assist legitimate users and strengthen legal standing.

- Verifies SHA-1/MD5 checksums of imported ISOs against bundled Redump database (offline).
- Flags known-bad dumps with guidance to re-dump from original disc.
- Does NOT block unverified files — informational only.
- Demonstrates the project is designed for legitimate disc owners.
- Redump database distributed as a bundled, periodically updated data file — no network dependency.

6. Functional Requirements

6.1 Performance & Visual Quality

- AMD FSR 3.0 Upscaling and Frame Generation (see Section 4.4).
- Async Shader Compilation — eliminates gameplay stuttering during shader cache generation.
- Per-Game Shader Cache — distributed via Cloud Game Configs to eliminate first-run stutter.

6.2 Audio System

- Low Latency: Oboe library (C++) on Android — bypasses Java audio stack.
- Time Stretching: smooth audio sync during frame drops to prevent crackling.

6.3 Input & Sensors

- Smart Sensor Fusion: gyroscope/accelerometer mapped to motion controls (Wii Remote, Sixaxis).
- Haptic Feedback: HD Rumble support on compatible devices.
- Input Mapping: full Bluetooth/USB controller support with per-game profiles.

6.4 Thermal & Power Management

- Thermal Throttling Manager: automatically reduces FSR quality or limits refresh rate to maintain stability and prevent thermal shutdown.

6.5 Smart Features

- Cloud Game Configs: community-maintained database applying optimal settings and GPU HLE patches automatically.
- Community Compatibility Database: open per-title compatibility reporting — modeled after Dolphin/PCSX2 wikis.
- BYO-Cloud Save: Google Drive and Dropbox integration for save state sync across devices.
- Dual Screen Management: adaptive layouts for DS, 3DS, Wii U (Split View, PiP, Foldable native).
- Redump Checksum Verification: optional ISO integrity check against known-good database.

7. Non-Functional Requirements

7.1 Extensibility

- Modding API: Lua scripting for external asset injection and logic modifications without altering originals.
- Telemetry (Opt-in): anonymous crash dumps and compatibility statistics.

7.2 Debug Toolchain

A dedicated debug toolchain is mandatory from the start of Phase 1. Games will not run correctly on first attempt. Without diagnostic tools, debugging becomes guesswork that wastes months of development time.

- Instruction Tracer: records executed instructions with register and memory state snapshots.
- Reference Comparator: compares LegacyCore state against a known-good emulator (e.g., Flycast) to pinpoint divergence.
- GPU Command Logger: records all HLE-translated Vulkan calls for frame-by-frame analysis.
- SMC Monitor: tracks memory write operations on compiled regions to detect Self-Modifying Code.
- Performance Profiler: identifies LLVM-generated hotspots for manual optimization.

7.3 Open Source Compliance

- All components released under MIT License.
- No proprietary dependencies permitted anywhere in the stack.
- All third-party components must be MIT-compatible.
- Complete source tree publicly available on GitHub from Day 1.

8. Security & Legal Compliance

8.1 Technical Security

- No Root Requirement: operates within standard user privileges on all platforms.
- In-Memory Execution: compiled code in RAM via memfd_create on Android — W^X compliance.
- SAF Compliance: full Android Storage Access Framework adherence.
- Device Binding v2: dual-layer — hardware fingerprint + optional user account.
- Networking: multiplayer restricted to P2P only — no centralized game servers.

8.2 Legal Boundaries — Permanent Non-Negotiable Constraints

- No cryptographic circumvention — excludes project from DMCA Section 1201 liability.
- No BIOS distribution — relies exclusively on HLE for system firmware functions.
- No ROM or ISO distribution of any kind.
- No support for Generation 8+ hardware — commercially active platforms permanently excluded.
- Term 'Porting' prohibited — correct term is On-Device Native Runtime Compilation.
- Users solely responsible for the legal status of game files they provide.
- All reverse engineering performed independently — no proprietary firmware or microcode used.
- Process documentation maintained to demonstrate no circumvention occurred at any stage.

9. Development Strategy

Development strategy designed to maximize probability of reaching a working proof-of-concept.

9.1 Bottom-Up Build Order

Priority	Component	Rationale
1st	LegacyCore CLI — command line only, no UI	Validate recompiler correctness before any UI exists
2nd	Debug Toolchain	Essential to diagnose failures — build before attempting any game
3rd	Single frame GPU output	Validate HLE pipeline with a static frame before animation
4th	Audio pipeline	Add audio only after video is stable
5th	Flutter Launcher UI	Build the interface last — after the hard parts work

9.2 MVP Progression Ladder

Each MVP level must be achieved and stabilized before proceeding. Skipping levels leads to compounded debugging complexity.

MVP	Target	Success Criteria
MVP-0	Dreamcast Homebrew 'Hello World'	SH-4 binary compiled via LLVM, static text via PowerVR HLE on Vulkan
MVP-1	Simple 2D Dreamcast Homebrew	2D sprite rendered correctly at stable framerate — validates basic HLE pipeline
MVP-2	Sonic the Hedgehog (Dreamcast)	Commercial game boots to gameplay — validates full Tier 1 pipeline
MVP-3	Shenmue or SoulCalibur (Dreamcast)	Complex 3D title running — validates HLE under real GPU load
MVP-4	PS1 commercial title	Validates MIPS R3000 recompiler and second GPU HLE layer
MVP-5	Full Tier 1 on LegacyDroid	All 5 Tier 1 platforms playable — triggers LegacyPC development

9.3 Community Growth Timeline

- Months 1-18: solo development — do not expect meaningful external contributions.
- Month 6: publish MVP-0 or MVP-1 on GitHub to establish public presence.
- Month 12: publish MVP-2 demo video (Sonic on LegacyDroid) — primary community recruitment moment.
- Month 18+: structured contributor onboarding with documented tasks and CONTRIBUTING.md.
- Year 2+: Cloud Game Configs contributions, HLE patches, and Compatibility DB entries from community.

9.4 Reference Projects (Do Not Reinvent)

- Flycast (MIT) — Dreamcast/NAOMI: SH-4 and PowerVR behavior reference.
- DuckStation (GPL-2.0) — PS1 MIPS and GPU reference.
- mupen64plus (GPL-2.0) — N64 RDP and RSP reference.
- PPSSPP (GPL-2.0) — PSP MIPS and GE reference.
- Dolphin (GPL-2.0) — GameCube/Wii PowerPC and Flipper/Hollywood reference.

- All reference usage must respect license terms. LegacyCore code written independently.

10. Roadmap

Planned with explicit MVP milestones and realistic single-developer timelines.

Phase 1	**The Foundation**	
Scope: LegacyCore CLI. Debug Toolchain. Dreamcast (SH-4) Static Recompiler via LLVM. Micro JIT for SMC. PowerVR HLE for Vulkan. Logic-Assets Separation. Android target only.	**Milestone:** MVP-2: Sonic the Hedgehog running on LegacyDroid.	
Phase 2	**The Expansion**	
Scope: PS1, N64, Nintendo DS, PSP. LegacyPC beta (Windows / Linux). Flutter Launcher UI. Vulkan renderer stabilization. Redump Checksum Verification.	**Milestone:** MVP-5: Full Tier 1. LegacyPC public beta.	
Phase 3	**The Sixth Generation**	
Scope: GameCube, PlayStation 2, Xbox Original, Nintendo Wii. AMD FSR 3.0 activation. Cloud Game Configs launch. Community Compatibility Database. Per-Game Shader Cache.	**Milestone:** Full Tier 2. Community DB live.	
Phase 4	**The Ecosystem**	
Scope: Device Binding v2 + User Account system. BYO-Cloud Save. P2P Netplay. Lua Modding API. Raspberry Pi 4/5 optimization. Contributor onboarding program.	**Milestone:** Complete feature parity. Active contributor community.	
Phase 5	**The Deferred**	
Scope: Wii U and Nintendo 3DS (PICA200 GPU research). Saturn, Xbox 360, PS3 feasibility study. Community GPU HLE contributions for Tier 4 research.	**Milestone:** Tier 3 support. Tier 4 research begins.	

11. Hardware Tier Classification — Rationale

GPU abstraction complexity is the primary classification factor. LLVM-based Static Recompilation largely neutralizes CPU differences.

Tier 1 — Launch Targets

Well-documented GPU architectures with manageable HLE surface area. Nintendo DS benefits from native ARM-to-ARM execution requiring no CPU recompilation. PSP shares MIPS with PS1/N64 enabling recompiler code reuse. Dreamcast is the mandatory first and only Phase 1 target.

Tier 2 — Second Phase

Moderate GPU complexity. Xbox Original's NVidia GeForce 3 is closest to a modern programmable pipeline — lowest HLE complexity in Tier 2. GameCube and Wii share ATI Flipper/Hollywood GPU family enabling substantial code reuse.

Tier 3 — Deferred

Wii U and 3DS deferred due to dual-screen complexity and, for 3DS, the PICA200 GPU which has no modern architectural equivalent and requires deep reverse engineering.

Tier 4 — Long-Term Research

Sega Saturn: dual SH-2 and dual-VDP require a fundamentally different parallel model. Xbox 360 and PS3: proprietary Xenos and RSX GPU architectures. PS3 Cell SPE x7 cores represent a parallel execution model with no direct LLVM target equivalent.

12. Approval

Role	Signature	Date
DT7A1 — Chief Technology Officer (CTO)	_____	_____

Legacy Project — Preserving gaming heritage for future generations. Built by the community, for the community.

Author: DT7A1 — github.com/DT7A1/LegacyProject