PROJECT DETAIL

Post-Deployment Remediation Script Development

SUMMARY

Built and deployed a remediation script for live post-deployment incidents. It targeted profile corruption, cache drift, and launch inconsistency. The script moved from testing into production use and became part of the wider stabilization effort.

HOW IT WAS BUILT

- **Development workflow:** Built with AI-assisted development support and standard tooling
- **Testing path:** Moved from non-production testing to test-device validation and then live deployment
- **User targeting:** Separated device logon user from session user to hit the correct profile
- **Cache cleanup:** Covered both version-specific and legacy cache variants
- **Shortcut control:** Consolidated validated shortcut artifacts into one canonical launch surface
- **Binary checks:** Confirmed binaries and detected reparse-point anomalies
- **Status output:** Returned JSON-compatible state summaries for downstream parsing

WHAT WE FOUND

- **Main pattern:** Incidents clustered in upgrade scenarios
- **Control case:** Clean first-time installs did not show the same failure profile
- **Main causes:** Cache drift, user-profile contamination, mixed shortcut paths, duplicate variants, and validation gaps
- **Field behavior:** Production incidents often involved stale shortcuts, legacy cache corruption, and precise per-profile targeting
- **Production result:** Validation confirmed normalized application state
- **Boundary:** Remaining failures fell outside remediation scope

HOW IT WAS USED

- **Evidence role:** Served as the main evidence item in the reformatted deployment package wrapper
- **Problem link:** Tied each remediation action back to observed symptoms
- **Approval:** Approved for controlled live use on single endpoints
- **Project effect:** Turned incident success into evidence for root-cause analysis and package hardening
- **Scope boundary:** Full wrapper lifecycle testing remained separate from script effectiveness evidence

WHY IT HELPED

- Each remediation action targeted a real observed failure rather than a theoretical fix
- Refinement decisions were based on test-device and production outcomes
- Defensive design handled cloud desktop variants, registry-redirected paths, and reparse-point binaries
- Rapid problem breakdown supported cache, shortcut, and user-context diagnosis
- AI-assisted code generation became production-grade PowerShell
- Confirmed outcomes stayed separate from assumptions that needed more evidence

ATTRIBUTION NOTE

This portfolio description was prepared with AI-assisted drafting support. Repository artifacts, validation records, and operational observations were used to summarize the work while preserving confidentiality.