

1. **Generalisasi masih terbatas pada satu studi kasus.** Naskah sendiri menyatakan fokus pada Koperasi Pusdatin Kemenhan dan generalisasi ke koperasi lain hanya sebatas saran. Ini membuat kontribusi teoretis kurang kuat jika dibandingkan kontribusi praktisnya.
2. **Stack teknis “PHP native” berpotensi lemah dari sisi maintainability & scalability.** Teknologi yang dipakai disebutkan PHP native + MySQL + Bootstrap. Untuk sistem ERP/keuangan yang berkembang, arsitektur ini rawan technical debt jika tidak disertai pembahasan desain modular, security hardening, dan rencana skalabilitas.
3. **Evaluasi non-fungsional tidak terlihat kuat.** Evaluasi sudah bagus di akurasi/efisiensi/transparansi, tetapi tidak tampak pengujian performa (load/concurrency), keamanan (OWASP-style checks), dan reliability (backup–restore, audit log tamper resistance). Untuk sistem keuangan, ini aspek penting.
4. **Aspek sosioteknis & integrasi legacy belum digarap.** Di bagian saran disebut sebagai peluang riset berikutnya (adopsi, change management, integrasi SIMPEG/payroll, keamanan kompleks, multi-tenant). Artinya hal ini belum jadi bagian penelitian sekarang.
5. **Ada gap kecil pada fitur pendukung.** Misalnya disebutkan fitur ekspor PDF belum tersedia sehingga skor pelacakan riwayat transaksi tidak maksimal. Ini minor, tapi menunjukkan requirement belum 100% tuntas.

Perbaikan yang harus dilakukan

Prioritas perbaikan (urut dari paling berdampak):

1. **Perkuat kontribusi ilmiah & transferabilitas.**
 - Tambahkan pembahasan “lesson learned” dan model konseptual ERP-keuangan koperasi yang bisa direplikasi di koperasi lain (mis. faktor keberhasilan, prasyarat organisasi, batas adaptasi).
 - Kaitkan lebih kuat ke teori digital maturity / ERP success (bukan hanya implementasi teknis).
2. **Tambahkan evaluasi non-fungsional minimal:**
 - **Security testing:** role-based access test, uji input validation, logging integrity, mitigasi SQLi/XSS.
 - **Performance & concurrency:** simulasi multiuser transaksi, waktu respon halaman kritis.
 - **Reliability:** skenario recovery & backup.
Ini bisa ditulis sebagai pengujian tambahan atau keterbatasan yang jelas.
3. **Naikkan kualitas arsitektur perangkat lunak.**
 - Jelaskan modularisasi kode, separation of concerns, dan alasan tetap memakai PHP native atau rencana migrasi ke framework (Laravel/CI) untuk maintainability.
 - Pertimbangkan desain API-first agar mudah integrasi antar unit/legacy, selaras dengan peluang integrasi yang Anda catat.
4. **Lengkapi analisis manfaat bisnis.**
 - Konversi efisiensi waktu menjadi estimasi saving biaya/jam kerja, atau dampak pada ketepatan laporan keuangan/keputusan koperasi.

5. **Rapikan requirement traceability.**

- Buat matriks “kebutuhan → fitur → test case → hasil” supaya pembaca cepat melihat coverage kebutuhan.