

The current version is #ident

"@(#)\$Format:LocalFoodAI_lanfr144:Recommendations.md:Francois

Lange:lanfr144@school.lu:2026/06/17 12:00:57:Francois

Lange:lanfr144@school.lu:2026/06/17 12:00:57:Not Committed Yet:local:none\$"

Recommendations for Future Agile Projects

Based on the challenges and technical resolutions encountered during the engineering and deployment lifecycle of this project, the following guidelines are recommended to optimize future implementations:

1. Version Control and Git Attributes Governance

- **Define Binary Assets Early:** Ensure that `.gitattributes` is defined and committed at the project's inception. Specifically, explicitly declare non-textual formats (e.g., `*.pdf binary`, `*.ttf binary`, `*.zip binary`, `*.docx binary`) to prevent line-ending renormalization from corrupting binary compiles.
- **Filter Script Ordering:** During cache resets or clean checkouts, isolate filters to prevent smudge/clean scripts from failing if dependency helpers are not yet available on disk.

2. Document Generation & PDF Portability

- **Prioritize Core Standard Fonts:** To prevent "Cannot extract embedded font" errors in external PDF viewers (such as Adobe Acrobat Reader), avoid custom font embeddings. Rely on built-in PDF standard fonts (such as `Helvetica` for sans-serif body text and `Courier` or `Courier New` for monospaced code blocks).
- **Code Block Wrapping & Margin Safety:** Avoid extremely long lines in code blocks. Manually wrap long command lines with the bash backslash `\` to prevent visual page overflows and copy-paste gluing.
- **Avoid Hard Page-Break Constraints:** Do not force `page-break-inside: avoid` on large block flowables. If a block's height exceeds the remaining page layout or the viewport boundary, the rendering engine may drop the block entirely or insert empty pages.
- **Portal and Index Structure:** For static document ecosystems, use relative links (`.md` targets) to cross-reference guides. Ensure the compilation script translates them to `.pdf` targets dynamically, creating a fully linked portable PDF ecosystem.
- **Auditable Page Layout Headers/Footers:** Use programmatic post-processing in PyMuPDF to draw clean page headers (logo image, document title, project code) and footers (author ID, Page X of Y) for institutional quality.

3. Shell Scripting and Configuration Streamlining

- **Dynamic Variable Formatting:** When presenting shell commands in documentation code blocks, avoid backslash-escaping environment variables (such as `$USER` or `$WSL_DISTRO_NAME`) as they render literally in the code block and break copy-paste execution.
- **Separation of Interactive Streams:** For stream editors (like `ed`), keep all commands

and markers on their own separate lines. Never join commands (like `.w`) on a single line. Use single quotes `'` for `echo` input streams to avoid escaping variables with backslashes.

- **Home Directory Alignment:** Always prefix repository clone commands with `cd ~` to ensure that deployment environments remain aligned relative to user home folders.