BRYCE PAGEL

CS 499 Milestone One

Dr. Sherin Aly

7/27/2025

Milestone Two

1. **Briefly describe the artifact. What is it? When was it created?**

The artifact I enhanced is batchRename (now renamed as Barron Pagel File Utilities), a Python‐based utility originally developed around March 2025 to streamline the bulk renaming of discovery response files in my family-law paralegal firm.

In this second enhancement, I introduced a command-pattern Undo system managed by a global stack in undo\_commands.py, and ensured collision-safe dry-run mapping via a seen\_targets dictionary. I extracted all file-system enumeration into folder\_file\_logic.py for cleaner, unit-testable logic. Beyond the core algorithm work, I polished the toast notifications so they stack rather than overwrite, display a progress bar indicating their lifespan, and animate off-screen with a wipe to the left. I also added a settings menu that generates a config.json in a hidden .bpfu folder in the user’s home directory, centralizes backups under .bpfu/backups by default, lets the user choose or reset the backup path, and offers a one-click clear-all-backups action while relocating the About dialog there. Finally, I refactored OS-specific code into isolated functions, added Windows .ico and macOS .icns icons with a PNG fallback, and updated the README.md to reflect the new branding, multi-tool UX, UI enhancements, cross-platform support, and future roadmap.

1. **Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

I selected this artifact because it embodies both algorithmic rigor and polished user experience. The Undo system demonstrates mastery of the command pattern for reversible operations and robust status reporting, while the seen\_targets hash-map approach prevents filename collisions in constant time. By decoupling file I/O from rename logic, I achieved a clean architecture that supports pure, easily testable code. The UI refinements (stacking toasts with progress indicators and smooth wipe animations) show attention to user feedback, and the new settings menu with persistent configuration illustrates sustainable design and user empowerment and gives a central point to increase that customizability in the future, enhancing scalability. Cross-platform considerations and distinct icon resources further showcase professional engineering foresight. The updated README.md itself serves as evidence of clear, audience-appropriate documentation, guiding users through installation, the revamped main menu, feature flows, config options, and build steps.

1. **Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

I fully met the outcome of designing and evaluating computing solutions by crafting an Undo stack and collision-safe mapping that handle edge cases and failures gracefully. I demonstrated well-founded and innovative techniques by applying the command pattern, leveraging Python’s native data structures, refactoring for modularity, and enhancing UI feedback mechanisms. The revamped documentation and in-app messaging satisfy the requirement for professional-quality communications, delivering concise instructions and visual cues tailored to both technical and non-technical users. My security mindset remains in progress: input validation and safe rollback logic now guard against unintended data loss, and the upcoming database module will further demand rigorous sanitization and secure handling of user metadata.

1. **Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

Building the Undo feature underscored the elegance of the command pattern but forced me to wrestle with comprehensive conflict detection and multi-state reporting. Isolating filesystem interactions into a dedicated module improved testability but required reframing familiar code into pure functions. Polishing the toast system pushed me to learn animation timing and state management within CustomTkinter, ensuring messages queue neatly and expire predictably. Designing the settings menu involved deciding sensible defaults for backup storage, determining an intuitive layout for path selection and cleanup actions. Preparing for cross-platform support led to abstracting OS-specific behaviors and integrating the correct icon formats. Finally, updating the README demanded concise technical writing to capture all these enhancements – branding, UI flows, settings, cross-platform notes, and build instructions – in a coherent, user-friendly document. This week’s work sharpened both my engineering and communication skills, yielding a tool ready for real-world deployment and future growth.