ChatGPT

Bug Report for PWA Forge

Environment

Item	Details
Test date	2025-10-23 (Europe/Prague)
Python version	3.x (container default)
OS	Linux container (no GUI)
Network restrictions	Outbound HTTP/HTTPS blocked for `git clone`/pip
Browsers installed	None (no Chrome/Chromium/Firefox/Edge)

Steps attempted

1. **Install from PyPI** – following the installation instructions in the *Usage guide* (`pip install pwa-forge` [1]). On the test system, `pip install pwa-forge` produced an error: *"ERROR: Could not find a version that satisfies the requirement pwa-forge"*. No versions of the `pwa-forge` package are published on PyPI, so installation fails.
2. **Install from source** – the documentation's example uses `git clone https://github.com/yourusername/pwa_forge.git` [2]. Attempts to clone `https://github.com/bigr/pwa_forge.git` or install with `pip install git+...` failed because outbound `git clone` is blocked in this environment; additionally, the documentation uses a placeholder `yourusername` instead of the actual `bigr` repository name.
3. **Run the CLI** – because installation failed, the `pwa-forge` CLI could not be executed. Running `pwa-forge --version` yielded a "command not found" error.
4. **Static review** – due to installation problems, static analysis of the source code and documentation was performed to identify potential issues.

Issues found

1. Package not available on PyPI

- The usage guide instructs users to install via `pip install pwa-forge` [1], but no `pwa-forge` distribution is available on PyPI. Attempting to install yields `ERROR: Could not find a version that satisfies the requirement pwa-forge`. This prevents normal installation and testing.
- **Recommendation:** publish the package to PyPI or update the documentation to reflect the current installation method (e.g., "install from source only until the package is released").

2. Incorrect repository URL in documentation

- The quick-start section says to clone `https://github.com/yourusername/pwa_forge.git` [2]. The actual project is hosted under `bigr/pwa_forge`. Users following the instructions will encounter a 404 or clone an incorrect repository.
- **Recommendation:** replace the placeholder with the correct repository URL.

3. Dry-run still requires a browser

- The `add_app` implementation invokes `_get_browser_executable()` to locate the browser binary **before** checking the `dry_run` flag [3]. If Chrome/Chromium/Firefox/Edge is not installed (common on servers or minimal containers), `_get_browser_executable()` raises `AddCommandError` [4], which stops the command even when `--dry-run` is specified. This defeats the purpose of dry-run and makes testing in headless environments impossible.
- **Recommendation:** only call `_get_browser_executable()` (and other OS-dependent routines) when `dry_run` is `False`, or allow specifying `--browser none` to skip browser detection for preview purposes.

4. Hard-coded browser paths and no fallback

- `BrowserConfig` defaults to `/usr/bin/google-chrome-stable`, `/usr/bin/chromium`, `/usr/bin/firefox` and `/usr/bin/microsoft-edge` [5]. `_get_browser_executable()` searches only those paths and a couple of additional hard-coded locations [4]. There is no use of `shutil.which` or `$PATH` to locate alternative browser binaries, so installations where browsers are installed in non-standard locations (e.g., `/snap/bin` or Flatpak) will result in "Browser not found" errors. The CLI fails even when the user sets a custom `--browser` value because the argument is validated against this fixed list.
- **Recommendation:** use `shutil.which()` to locate the requested browser in the user's `PATH` and provide a way to configure custom browser paths via the config file.

5. Dry-run still generates templates

- In `add_app`, the code calls `render_template(...)` and builds the wrapper/desktop contents regardless of the `dry_run` flag [6]. Although file writes are skipped, template rendering may still fail if required Jinja templates are missing or invalid. Rendering could be skipped entirely in dry-run mode to improve performance.

6. Platform-specific registry locking

- The `Registry` class uses `fcntl.flock` for registry file locking [7]. This mechanism only works on POSIX file systems and will not run on Windows. If cross-platform support is desired, an alternative locking mechanism should be considered.

7. Potential concurrency issues in registry

- The `_read` method returns an empty registry dictionary when the file does not exist [8]. If two processes call `add_app` simultaneously, both may read an empty registry and then write conflicting registries, overwriting each other's entries. The file lock prevents simultaneous writes, but two reads

followed by writes in separate processes could still race. Using atomic read-modify-write operations or storing apps keyed by ID would mitigate this.

8. Documentation refers to unreleased features

- The usage guide lists commands such as `generate-handler`, `install-handler` and `generate-userscript`, but the code for these commands was not reviewed and may be incomplete.
- The installation section emphasises that PyPI installation is only possible “when published”, but does not warn that the package is not yet published. This leads to user confusion.

9. Minor usability issues

- The CLI accepts both `--quiet` and `--verbose` options. There is no explicit check to prevent using them together; the resulting log level may be undefined. A mutually-exclusive group would improve clarity.
- `validate_url()` treats any message containing the word “Warning” as a warning [9]. If future error messages include the word “Warning”, they may be mis-interpreted. Checking explicit status codes instead of string matching is safer.

Summary

The **PWA Forge** project aims to turn web apps into native-like Linux launchers, but installation hurdles prevented full functional testing. The package is not yet available on PyPI, and cloning the repository is blocked in restricted environments, so the `pwa-forge` CLI could not be run. Static analysis highlighted several design issues: dry-run still requires a browser, browser paths are hard-coded, and file locking is platform-specific. Addressing the installation availability, correcting documentation, and relaxing environment assumptions (browser discovery, optional verification during dry runs) will make the project easier to install and test.

---

[1] [2] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/docs/USAGE.md

[3] [4] [6] [9] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/commands/add.py

[5] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/config.py

[7] [8] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/registry.py