ChatGPT

PWA Forge testing – second cycle

Test environment & methodology

- **Version under test:** the repo snapshot provided as `pwa_forge-main.zip` (dated 2025-10-23). PyPI installation was still unavailable, so I ran the CLI directly from the extracted `src` directory via `PYTHONPATH` to avoid network-dependent dependencies. The installed version reports itself as **0.1.0**.
- **Execution:** All commands were exercised via the `cli.cli()` entrypoint. Because there is no graphical desktop, I focused on command-line behaviour and inspected generated files in `~/.local`.
- **Docs reviewed:** the `Implementation-specification.md` attached by the user and the upstream source at `raw.githubusercontent.com/bigr/pwa_forge/main`. Relevant excerpts are cited below.

Functionality that worked

Feature	Observations
Listing PWAs	With no apps registered, `pwa-forge list` showed "No PWAs found". After adding a PWA, the app appeared in the table and was marked **active**.
Configuration management	`pwa-forge config list` correctly printed the default configuration, including default browser (`chrome`) and directories for apps, icons, wrappers, etc.
Doctor command	`pwa-forge doctor` ran a comprehensive system check. It detected available tools (`xdg-mime`), identified browsers found on the system (only `/usr/bin/chromium` was present) and warned about missing ones (chrome, firefox, edge). This command also highlighted missing `update-desktop-database`.
Userscript generation	`pwa-forge generate-userscript --scheme ff --out /tmp/example.user.js` created a userscript and printed detailed installation instructions. The generated file exists and contains the expected boilerplate.
Adding a PWA (dry-run)	When an available browser was specified, e.g., `--browser chromium`, the dry-run mode successfully printed the paths it would create and exited without side-effects.
Adding a PWA (real)	Creating a real PWA using `chromium` succeeded. The command created the profile directory, wrapper script, `.desktop` file and manifest under `~/.local/share/pwa-forge`. `pwa-forge list` showed the app as active, and `pwa-forge audit <id>` reported that all nine audit checks passed. Removing the PWA in dry-run mode printed all the paths that would be removed.

Remaining issues & observations

1. **Outdated code in zipped version**
   The provided snapshot appears to pre-date the fixes referenced in the user's changelog. Critical improvements such as mutually exclusive verbosity flags, browser detection fallback, skipping browser validation in dry-run, and cross-platform file locking are not present in this version.
   Evidence:
2. The CLI does **not** validate mutually exclusive `--quiet` and `--verbose` options; running `pwa-forge list -v -q` exits with no error. The upstream `cli.py` now checks `if quiet and verbose > 0` and raises a usage error [1], but this check is absent in the tested code.
3. `add_app` in the snapshot calls `_get_browser_executable` before checking `dry_run`, so dry-run fails when the default browser (chrome) isn't installed. The upstream implementation has been fixed to skip browser validation and use a placeholder path when `dry_run` is `True` [2].
4. `_get_browser_executable` in the tested version only looks at hard-coded paths from the config, so specifying `--browser firefox` fails even though `firefox` may be in PATH. The upstream function now falls back to searching the system path with `shutil.which()` [3]. The default config still defines static paths [4], so the fallback is essential.

5. The registry in the snapshot uses only `fcntl.flock` for locking, limiting it to Unix. The upstream `registry.py` detects the platform and uses `msvcrt.locking` on Windows, and performs an atomic read-check-write inside the lock to prevent race conditions [5] [6].

6. **Installation still unclear**
   The usage guide now states that PWA Forge is not yet on PyPI and must be installed from GitHub (via `pip install git+https://github.com/bigr/pwa_forge.git` or cloning and running `pip install -e .`). However, in a restricted environment with no internet access this still isn't possible; running `pip install` from the zipped source fails because build dependencies cannot be downloaded. The documentation could mention that the tool can be run directly from the `src` directory (by adding it to `PYTHONPATH`), which is how this test was executed.

7. **Browser detection**
   Unless the updated fallback is used, the tool requires specifying the exact path of a supported browser. When the default `chrome` is missing, both normal and dry-run executions fail. With the upstream `shutil.which()` fallback [3], the CLI should automatically pick `chromium` or another installed browser. Testing this fallback was not possible because the snapshot lacked the new code.

8. **Missing** `update-desktop-database`
   `pwa-forge add` warns that `update-desktop-database` is not found. Without this utility the desktop database isn't updated; some desktop environments may not immediately see the new `.desktop` file. The `doctor` command already checks for this and hints that `xdg-utils` should be installed. If a portable fallback exists, it could be invoked or documented.

9. **Handler generation**
   The `generate-handler` command fails when the default browser isn't installed (e.g., it expects `/usr/bin/firefox` by default). Allowing `--browser` to select the available browser or using the browser detection fallback would resolve this.

10. **Legacy documentation fragments**
    Parts of the `USAGE.md` still reference `pip install pwa-forge` and a placeholder GitHub URL (`yourusername/pwa-forge`). The updated documentation should consistently instruct users to install from `bigr/pwa_forge` or via `pip install git+https://github.com/bigr/pwa_forge.git`.

Conclusion & suggestions

The core functionality (adding, listing, auditing, removing PWAs; generating userscripts) works well when an available browser (e.g., `chromium`) is specified. However, the provided code snapshot does not include several important fixes listed in the changelog. To fully verify the resolved issues, testing should be performed against the latest commit on the `main` branch. In that version, the following should be confirmed:

- The CLI should reject simultaneous `--quiet` and `--verbose` flags and display an appropriate error [1].
- Dry-run mode should skip browser checks and succeed even when no browsers are installed [2].
- Browser detection should fall back to searching `PATH` for common executable names [3].
- Registry operations should be safe on Windows and avoid race conditions by using platform-specific file locking and an atomic read-check-write [5] [6].
- The docs should provide clear installation instructions for offline environments and remove outdated placeholders.

Once the local installation is updated, a third test cycle should verify that these improvements behave as expected.

---

[1] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/cli.py

[2] [3] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/commands/add.py

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

[5] [6] raw.githubusercontent.com
https://raw.githubusercontent.com/bigr/pwa_forge/main/src/pwa_forge/registry.py