extend your methodology to develop the full plan to move the project to the ujust submodules structure,

also create a set of task prompts (start with that) for an appropriate expert instance to:

research https://github.com/ublue-os/image-template
https://github.com/opencontainers/image-spec/blob/main/spec.md
https://copr.fedorainfracloud.org/
https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository
https://edu.chainguard.dev/open-source/sigstore/cosign/how-to-install-cosign/#installing-cosign-with-the-cosign-binary
https://github.com/ublue-os/image-template/blob/main/.github/workflows/build-disk.yml
https://artifacthub.io/
https://github.com/m2giles/m2os
https://github.com/bsherman/bos
https://github.com/astrovm/amyos

create a project brand, theming, banner styles, unique icon and font schema, as a modification of the bazzite/universal blue theming AND develop a unique bootc image, bespoke to our needs, which once again would address the needs of the community. The installer idea is great, what if we could ship the project just as a rock-solid image, implementing our unique workflows

also a prompt to create and maintain a community + associated/upstream project need + changelog + bug tracker + project watch library, which will, as a published perplexity labs page, interactively show the needs of the community (forum/reddit posts.. etc) and the work being done (by us, and others) to address these needs

Here are expert-grade task prompts and an extended methodology for modernizing and future-proofing this project into a proper ujust submodules-based, Universal Blue-aligned pipeline.

Task Prompts for Bootc/ujust Submodules Rollout & Brand/Community Ops

1. Technical Research Activation (ujust/bootc/integration)

- **Study Universal Blue bootc submodule structure:**
  - Analyze:
    - Universal Blue image-template
    - Image specification: OCI spec
    - Official Universal Blue [Justfile/readme/CONTRIBUTING.md]
- **Create & Document a new submodule:**
  - Fork image-template and create a submodule for your toolkit.
  - Model `Containerfile`, `Justfile`, and modular scripts after Universal Blue best practices.
  - Document using AGTL/mithril patterns (REF-tag metadata, decision history, readable YAML).
  - Add bootc support: ensure all deployment steps are safe on atomic/immutable distros.
- **Setup secure supply chain:**
  - Sign OCI images:
    - Research and implement cosign for Github Actions.
    - Add image signing steps in build.yml.
- **Research and document third-party repo integration:**
  - Review Fedora COPR best practices for 3rd-party RPM support, add safe enable/disable hooks.
- **Automate disk and ISO builds:**
  - Study build-disk.yml workflow.
  - Extend pipeline to produce ready-to-run ISOs/boot media for testing and preview.

2. Brand, UX, and Visual Identity

- **Design project brand and theming:**
  - Audit current Bazzite/Universal Blue banners and icons; list the common themes (blue gradients, retro gaming, pixel styles, flat design).
  - Draft a new logo with a unique palette and a modern typeface mixed with retro pixel/console vibes; give it a clear wordmark and a memorable pictogram.
  - Pick and license a distinct open-source font for CLI/UI.
  - Create SVG/PNG banners for docs, forum, social preview (OG), and the image boot splash.
- **Develop a branded bootc image from scratch:**
  - Fork image-template; apply the artwork/logo, start with the needed workflow scripts baked in.
  - Document the complete pipeline, including:

        - Custom installer script (as wrapper, or for user-space dry run)
        - Theming and wallpapers set by default
        - Pre-pinned help/manual links, forum, diagnostics

3. Community & Knowledge Ecosystem

- **Create and organize a community portal:**
    - New repo for "community-hub": README, live issue tracker, public changelog, project board.
    - Install/operate a static knowledge site (e.g., Perplexity Labs, GitHub Pages, or ArtifactHub).
    - Script or automate community needs intake:
        - Scrape relevant forum/reddit posts about Bazzite/Universal Blue feature requests (monitor with GitHub Actions or a webhook).
        - Present an auto-updated NEEDS.md file or dashboard.
        - Run a prompt chain ("Summarize top 3 open requests and current work in flight").
    - Host/track upstream dependency status (e.g., for Flatpak, COPR, upstream repos listed above).

4. Workflow & Project Management

- **Changelog, Bug Tracker, Project Library:**
    - Draft a well-structured CHANGELOG.md with each release/change.
    - Integrate labels for "upstream", "needs-in-community", "in-progress", "next".
    - Add GitHub project board for feature tracking.
    - Use Discussions and Issues; auto-link with forum/Reddit insights (prompt agents to collect and surface insights weekly).
    - Align documentation with MIT/Apache open governance.

5. Brand & UX Prompt for Designers/Teams

"Develop a flagship theme, logo, and icon set for our Bazzite-derived project that balances visual continuity with Universal Blue—abstract geometric forms, 'electric' blues, pixel gradients, and modern CLI typography. Integrate the look into a boot splash, desktop wallpaper, portal header, and GitHub social preview. Select and license a legible, accessible monospace font fit for both code and branding, and apply to CLI outputs and docs."

6. Community/AI Dashboard Prompt

"Create a public dashboard (Perplexity Labs/Obsidian/Docsify/ArtifactHub) which auto-publishes the current top-10 community and upstream needs, shows actionable changelog diffs, and cross-links all our and upstream Universal Blue/Bazzite projects. Add an auto-updating embedded bug tracker and changelog with direct links to commits, PRs, and forum/Reddit discussions. Make status and watchlists easily scannable for contributors and maintainers."

Extended Methodology for Submodule Move

I. Repo & Submodules

- Fork from [ublue-os/image-template][1]
- Each tool, recipe, or utility = a directory/submodule (e.g. `/modules/network`, `/modules/gaming`)
- Use root `Containerfile`, `Justfile`, and standardized `/docs`, `/branding`, `/installer`

II. Core Structure & Directories

Keep Bazzite/Universal Blue conventions:

- `/usr/share/ublue-os/just/` for ujust commands (`80-pro-network.just` etc.)
- `/var/lib/bazzite-pro/` for persistent state
- `~/.bazzite-pro/` & `~/.config/bazzite-pro/` for per-user configs, logs, sessions
- All hooks, scripts, helpers as `/modules` subfolders

III. Build, Sign, Distribute

- Use `podman build` and GitHub Actions for image builds
- Sign artifacts/images with Cosign
- Push to GHCR (container registry)
- Set up branch protection and signed releases (security, audit)
- Register on [ArtifactHub.io] for distribution/visibility[2]

IV. Integration & Testing

- Implement automated unit and integration testing (GitHub CI, bats for bash scripts)
- Compose sample profiles for gaming, dev, workstation
- Simulate updates/rollbacks on Fedora Atomic, Bazzite, Bluefin test VMs

V. Documentation & Community

- Wiki: full install/usage docs, FAQ, troubleshooting, privacy, compatibility (reference Universal Blue docs for style)
- Publicly visible changelog, bug tracker, board
- Make contribution guidelines explicit, linked to the most common forum questions/issues

VI. Theming & Brand

- Derive color palette, icon, wallpaper from Universal Blue but with distinct new brand/OG image
- Integrate into the default bootc Containerfile (wallpaper, plymouth splash, help links)

References

This approach ensures a robust, immutable, branded, and community-integrated toolset/image, and supports smooth growth through modular, reproducible pipelines and evidence-driven development cycles.

⁂

1. https://github.com/ublue-os/image-template
2. https://www.semanticscholar.org/paper/1465cb2748d808aba6d2ce7c355f1fb21fb4ac3c
3. bazzite-automation-architecture.md
4. mithril-bazzite-refactor.md
5. directory-structure.md