ChatGPT

Project Cone (pjcone_pwa) Web App Analysis

Project: Cone is a web-based messaging and collaboration app designed for small content-producing teams [1]. It is delivered as a **Progressive Web App** that runs in the browser (Chrome recommended) and can be installed as a standalone desktop or mobile application [2] [3]. All user data (chats, tasks, files) are stored client-side and sync via user-run private servers; there is **no official public service server** provided [4]. This architecture allows complete self-hosting and even offline-first usage: most app functions work offline, with changes synchronized to the server when a user goes online [5] [6]. Users log in with an email/password pair, but the system does not verify emails, so one can use any dummy address (e.g. `aa@aa.aa`) without hindrance [7]. In short, Cone emphasizes **privacy and autonomy**: users are "configured to be completely anonymous" [7] and may connect to multiple servers concurrently for messaging and file sync [8].

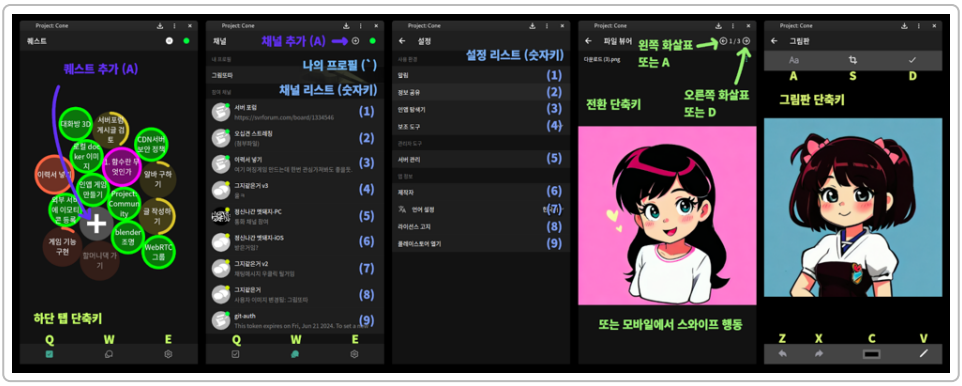
1. User Experience (UX)

Project Cone's UX aims to balance anonymity, offline resilience, and team communication. The app uses a **clean, dark-themed interface** with large icons and numeric shortcuts for keyboard power-users (e.g. pressing A to add, numbers for list items) [9]. Key usability points include:

- **Offline/Online Mode**: Users can work offline by default, then go online to sync or collaborate. The profile screen's login fields activate online mode [5]. This is a major UX strength: users remain productive even without network, reducing frustration during outages.
- **Anonymity and Security**: Signing up requires an email/password pair, but any email format is accepted [7]. This lowers friction (no email verification) and appeals to privacy-minded users, but can confuse users expecting account recovery or identity proof. (If a user forgets their password, they truly lose access.)
- **Learning Curve**: The app introduces some unconventional concepts (e.g. Quests for tasks, separate Channels) and uses Korean by default. However, a detailed user guide is available [1] [5]. Keyboard shortcuts and on-screen hints (like "퀘스트 추가 (A)") help; for example, pressing the + adds a quest or channel. Still, new users may initially find navigation non-intuitive (e.g. numeric keys to select items) until they learn the controls.

Overall, feedback from the documentation suggests users can quickly communicate in channels or manage tasks once acquainted. The offline-first approach and simplicity of login enhance satisfaction, but the anonymity and server requirements may pose challenges to casual or non-technical users [5] [7].

2. UI Layout Appropriateness



The Project Cone UI maintains a consistent visual style across screens. All views share a top app bar ("Project Cone") and a dark background with green accent colors for interactive elements (buttons, toggles). In the screenshots above, the **first panel** shows the Quest creation screen with colored task bubbles, the **second** is the Channel list, the **third** is the Settings list (numbered for shortcut keys), and the **fourth/fifth** panels show the Drawing/Editor tool for annotating images. Element placement is logical: menus and lists span the left side, action buttons are clearly labeled (often with an explicit hint of their keyboard shortcut in parentheses), and icons (e.g. gear for Settings) are consistent throughout.

- **Consistency**: All pages use the same layout and color scheme, making navigation predictable. Text size and spacing are comfortable for desktop use, and the app scales to different screen sizes (mobile vs PC) without major issues.
- **Responsive Controls**: The use of numeric keys (1–9) to jump to list items (as shown in the Settings list) speeds up navigation for power users [9]. The bottom navigation bar (visible on mobile width screens) provides quick access to Home/Channels/Quests/Settings. This is coherent and intuitive once learned.
- **Visual Design**: The aesthetic is clean and modern. Icons (e.g. for Add, Back) are familiar, and the task bubbles in the Quest screen use color and shape effectively to distinguish tasks and deadlines. However, some screens (like the drawing editor) have many small buttons which might require learning. Overall, the UI is appropriate for the app's goals, balancing functionality with simplicity.

3. Functional Suitability

Project Cone's features align well with its intended use as an internal team messenger with integrated task management:

- **Messaging (Channels)**: Channels serve as chat rooms. Users can create multiple channels (including a "self-chat" for personal notes) and invite others by sharing a channel ID or scanning a QR code [10]. This mirrors common chat apps but adds the ability to partition discussions and personal memos. Offline, messages are cached locally, and upon login each channel's chat history syncs with the server. Multiple server connections can be maintained simultaneously [8]. This setup meets the goal of flexible, distributed chat.
- **Task Management (Quests)**: A standout feature is the Quest system, which is a visual to-do list. Quests appear as circular widgets with progress arcs (indicating elapsed vs remaining time) [11] 【50†】. Users can attach files or assign tasks to others. This directly supports teamwork by integrating task tracking into the chat environment. It fills a functional niche not typically covered by plain chat apps.

- **File Sharing and Viewing**: The app's built-in file viewer supports text, images, audio, video, even 3D files (.blend, .pck) [12]. Crucially, every file opened can be **annotated** in-app via a simple "drawing pad" editor. For example, a screenshot or document can be marked up and saved back to chat【53†】. This synergy of chat + annotations is a strong fit for content teams needing feedback loops.

The functions generally meet the intended goals: providing a self-hosted team collaboration tool. The mobile and desktop (PWA) versions behave similarly, and users can transition between them. However, some expected features are not yet implemented (e.g. voice/video calls, advanced search) since the app is still in early development [1]. The documentation notes it is "in active development" with known missing core features [1]. For now, the app excels at text and task collaboration; other uses would need future updates.

4. Accessibility of Functions

The app offers several accessibility features but also has limitations:

- **Keyboard Support**: Project Cone is designed for keyboard-driven use. Menus and lists are fully navigable by keys (e.g. numeric keys, arrow keys, ESC to go back) [9]. This benefits power users and those with limited mouse ability.
- **Language and Contrast**: The interface defaults to Korean, but the settings allow choosing other languages (as indicated by a language dropdown in the Settings screen【42†】). The high-contrast dark theme aids visibility for many users, but there is no explicit indication of alternate high-contrast modes or text scaling beyond standard system fonts.
- **Offline Warning and Installation**: If JavaScript is disabled, the app shows a plain message: "Please enable JavaScript…" [13]. This is a basic fallback, but users relying on JS-blocking or older browsers would find the app unusable. The PWA install flow is standard (the browser prompts to install), which is intuitive.
- **Feedback and Errors**: The app provides visual cues (e.g. a green "online" indicator after login [14]). However, it may lack audible or haptic feedback cues. The keyboard shortcuts are shown in text, but there is no mention of screen-reader labels or ARIA support in the documentation.

In summary, Project Cone is **accessible to keyboard-centric users** and benefits from being a web app (usable on many devices). Its reliance on client-side JS and heavy use of color in task charts may pose issues for some accessibility tools. As it matures, the developer may need to address standards like WCAG (e.g. adding ARIA labels, colorblind-friendly indicators) to improve accessibility for all users.

5. Strengths and Weaknesses

Strengths	Weaknesses
• **Offline-first design** – Most features (chat, tasks, files) work without network [5]. Easy syncing when online.	• **Self-hosting requirement** – No official server means initial setup (or using a shared dev server) is needed [4]. May deter non-technical teams.
• **Anonymity/Privacy** – Users can join with any email, no tracking, and data stays on self-hosted servers [7].	• **Steep learning curve** – Custom UI (quests, numeric shortcuts) requires user orientation. First-time users may be confused.

Strengths	Weaknesses
• **Integrated task (Quest) system** – Unique visual to-do lists within chat support project management.	• **Feature gaps** – Lacks some expected features (voice/video calls, push notifications, threaded replies) as development is ongoing [1].
• **File annotation** – Built-in "drawing pad" allows marking up documents/images directly [15]【53†】, streamlining feedback.	• **Browser dependency** – Requires modern browsers and JS. Non-JS browsers cannot use the app at all.
• **Responsive UI with shortcuts** – Well-designed dark theme, consistent menus, and keyboard shortcuts [9] enhance efficiency.	• **Limited accessibility** – No explicit screen-reader support or high-contrast mode noted; heavy visual info (colored charts) might not suit all users.

These strengths align well with Project Cone's goals of secure, offline-capable team communication. The weaknesses largely stem from its early-stage status and target of developer-friendly environments rather than mass consumer polish.

6. Future Potential

Project Cone has significant room to grow. Key directions include:

- **Feature Completion**: Implement missing features such as push notifications, message search, file upload via camera, or real-time typing indicators to match user expectations.
- **Mobile Optimization**: Enhance the mobile PWA experience (e.g. easier channel navigation on small screens, native performance tweaks) and possibly native app wrappers for iOS/Android for push and offline behavior.
- **Server Ecosystem**: Provide easier server setup (perhaps containerized) or community-shared servers. An official or volunteer-run server could help onboard new users who cannot self-host, broadening adoption.
- **Integration and Extensions**: Add support for bots, webhooks, or integrations (e.g. GitHub, CI tools) to make it a central collaboration hub. The current tag list suggests future focus on plugins and scripting.
- **Accessibility & Localization**: Expand language support (UI translations) and improve accessibility (ARIA roles, color themes) to reach more users.
- **UX Refinements**: Continue polishing UI/UX based on feedback. For example, clarifying the onboarding flow, simplifying the UI for non-technical users, and optimizing keyboard navigation hints.

Given its open-source nature, community contributions could accelerate these improvements. The existing architecture (PWA + Nakama server) is modern and scalable, so performance and scalability can improve with iterative development.

7. Critical Points

Several potential issues could affect users:

- **Single Point of Failure (User-side)**: Since all data is stored in-browser, clearing browser data or deleting the PWA loses all history unless synced to a server. Users must remember to set up and trust a private server for persistence.

- **Security Considerations**: Offline data and anonymity mean there's no real user authentication. If an attacker gains access to the saved data or device, they could impersonate the user. The use of WebRTC suggests optional media features, which need careful security review.
- **Browser Compatibility**: The app seems optimized for Chromium-based browsers (Chrome). Some features (like the install prompt) may not work the same on Firefox or Safari. The developer notes they will fix issues in other browsers, but compatibility testing is critical for users on varied systems.
- **Scalability Limits**: For very large teams or heavy usage (many channels, large file history), the client-side architecture might strain browser memory. Performance testing is needed. Similarly, the Nakama server backend must be robustly configured for concurrent users.
- **User Support**: As a niche project, documentation is mostly in Korean and English; community support (forums, issue trackers) will be important for onboarding and troubleshooting. The developer encourages bug reports, which is good, but formal support channels are not evident.

In summary, **while Project Cone is promising**, users should be aware of its current development status and technical requirements. Teams with the ability to self-host and a tolerance for cutting-edge software will benefit most in the near term.

Sources: Official Project Cone documentation and guides [1] [16] [4], as well as APKPure listing for context [17]. These provide insights into the app's design, features, and intended usage. (Images above are drawn from the Project Cone user manual [11] [15].)

---

[1] [3] [5] [6] [7] [8] [9] [10] [11] [12] [14] [15] [16] Project: Cone 앱 사용 설명서 [v0.55.14] | 최성수
https://is2you2.github.io/posts/how-to-use-pjcone/

[2] 최성수
https://is2you2.github.io/

[4] Project: Cone 사설 서버 설명서 [v0.55.14] | 그림또따
https://is2you2.github.io/posts/how-to-use-pjcone-server/

[13] Project: Cone
https://is2you2.github.io/pjcone_pwa/

[17] Project: Cone APK for Android Download
https://apkpure.com/project-cone/org.pjcone.portal