

A Polished Web App is a Detour When the User Lives on Their Phone

18-35 year old women want effortless social photos. The loop is pick, generate, save, share. Mobile-native is the only way.



	PWA (Web)	Expo (Native)
Camera Access	✗ Sandboxed & Clunky	✓ Direct Native Camera Roll
File Storage	✗ Cannot save to album easily	✓ Auto-cache to filesystem
Share Sheet	✗ Non-existent	✓ 1-Tap WeChat Moments

The Hardest Technical Hurdle Was a Known iOS Streaming Trap

The React Native Bug

fetch + ReadableStream = null body on iOS. (A known, multi-year runtime limitation).



The Evaluated Alternatives

- **react-native-sse** fails (GET-only, we need POST).
- **react-native-fetch-api** fails (Expo 55 compatibility).

The Old-School Savior

Use XMLHttpRequest to enable progressive text accumulation via onprogress.



Buffer chunks → split by double-newline

→ Parse JSON. Result: 100% reliable streaming without native module patches!

Sometimes, Mobile Development is Wonderfully Trivial



Web Implementation (Tailwind)

- Requires **dark**: prefixes everywhere.
- Complex CSS variables.
- Time taken: Days of tweaking.

Mobile Implementation (Expo SDK 55)

- **userInterfaceStyle:** 'automatic'
- Simple ThemeProvider lookup table.
- Component reads from `useTheme()` (`colors.background`).
- Time taken: 2 Hours.

No conditional `className` logic.
Just direct style object application.
The champagne gold brand accent
remains perfectly consistent.

The Logic Transfers, But the Fingers Behave Differently



Photo Selection

Web: Drag-and-drop dropzones.

Mobile: expo-image-picker with compact headshot/body preview slots.

Inspiration Matching

Web: Hover-to-preview responsive grid.

Mobile: Tap-to-select, long-press-to-preview sliding modal. AI auto-matches and collapses the picker. (One fewer tap!)

The Waiting Game

Mobile Exclusive: Skeleton shimmer animations via react-native-reanimated with rotating, cozy waiting copy (‘光正在寻找最好的角度’) to keep the user engaged.

A 35% Size Ratio Tells the Real Story of Code Reuse



Transferred (100%): Prompt logic, AI pipelines, Muse Card data (Server-side heavy lifting).



Transferred (85% - 95%): API contracts, TS types, Zustand store architecture.



Rewritten (40%): UI components (DOM vs View), Expo file-based Routing.

Web Codebase: ~**12,000** lines

Mobile Codebase: ~**4,200** lines

Mobile is just 35% the size of the web app.

**New AI features ship to both platforms simultaneously.
The intelligence lives on the server; the clients are just beautiful windows.**