UI/UX Design and Style Standardization

From Phong Trinh Ha Cameron Proulx Jacob Beaumont Gabriel Laboy

Date Mar 28, 2025

Color Schemes

This is the color scheme. It will be codified into more extensive shadcn/ui theme variables. We use the Geist font family. We use TailwindCSS variables for padding and margins (e.g., *p-1*, *p-2*, *p-4*, *p-8*) and *gap-** in *flex* and *grid* layouts for consistent spacing between elements.

Component Behavior

The default hover effects, transitions, and animations from shadcn/ui are adequate but may be subject to change as we get a better idea of the app's look and feel. Components outside of shadcn/ui are to be designed in congruence with shadcn/ui's hover effects, transitions, and animations.

Accessibility Standards

Since we are using shadcn/ui components, accessibility is achieved through shad's usage of Radix UI primitives which prioritize accessibility. According to Radix, "components adhere to the WAI-ARIA design patterns where possible." Beyond components provided by shad, we try to use common design patterns as well as accessibility tools such as the *htmlFor* element attribute, the *alt* image attribute, and Radix's *asChild* prop.

Responsive Design

Responsive design is achieved through the usage of TailwindCSS, shadcn/ui, and regular testing. By combining the strategies below, we hope to achieve a consistent and user-friendly experience across a wide range of devices and screen sizes.

- TailwindCSS provides responsive utility classes like *sm*, *md*, *lg*, and *xl* that can be used to specify which styles apply to which screen widths. It also comes with *flex* and *grid* utilities that make it easy to create responsive layouts that dynamically reorder and resize components.
- shadcn/ui components are already responsive using Tailwind. They are also composable and customizable, making it easy to adjust them for our own needs.
- For mobile development, our approach to responsive design is not too different. Instead of TailwindCSS we use NativeWind, and instead of shadcn/ui we use React Native Reusables. Both are analogs of their React counterparts, but for React Native. There are of course mobile-specific considerations to responsive design that we will encounter during development.
- Regular testing on different screen sizes and devices is critical. We use Vite to launch local development servers and view UI changes as we write them. Along with developer tools built into many Chromium browsers, UI testing on different screen sizes is a breeze. For mobile, Expo's device simulators and Expo Go will be used to validate responsiveness.