When Your Product Needs a Design System

Repeated interface decisions are a signal to create shared foundations, components and practical guidance.

Mavocado Design | 23 October 2026 | Design Systems

Look for repeated decisions

A design system can help when teams repeatedly solve the same interface problem in different ways. Similar buttons, inconsistent form states and competing layout patterns can indicate a need for shared decisions.

Start by reviewing what already exists. Identify patterns used across several journeys and separate intentional differences from accidental variation. A useful system should reflect the product's real needs, rather than a large collection of hypothetical components.

Build from a small, useful foundation

Agree on the basics that influence many screens: typography, spacing, colour roles, interaction states and naming. Then choose a small group of components that will be used in actual work.

Document how a component behaves, not just how it looks. A form field may need default, focus, error, disabled and loading-related behaviour. A shared visual appearance is only one part of a consistent experience.

Connect design with implementation

Designers and developers should agree on how patterns are named and represented. Discuss the situations in which a component should be reused, extended or replaced by a more suitable pattern.

Include examples from the product. A component shown in context helps the next person understand its purpose and constraints. Keep guidance short enough to use during everyday work.

Plan for maintenance

A library without ownership can become another source of inconsistency. Decide who reviews changes, how updates are communicated and how older patterns are retired.

Treat adoption as part of the work. Begin with a real feature, identify what the team struggled to use and improve the system from that experience. The goal is to make recurring decisions clearer, not to make every screen identical.

A Clear Next Step

Start a design system when shared decisions can solve repeated product problems. A small, maintained system used by the team is more useful than a large library that nobody trusts.