NUSA DESIGN SYSTEM

Setup guide

How foundations, tokens, components, patterns, themes, and AI handoff files fit together - and how to reuse the system in another web project.

10 foundations Color through Accessibility.	**3 token layers** Primitive -> Semantic -> Component.
36 components Typed production primitives.	**14 patterns** Reusable product compositions.

VERSION

Nusa v2.4 / Documentation generated 20 July 2026

Source project

lawndry-system - Next.js 16.2 / React 19.2 / Tailwind CSS 4 / TypeScript 5

Use this document as the implementation map. The live documentation remains the source of truth for interactive states and API details.

SECTION 01

The system at a glance

A single dependency direction prevents local styling from drifting away from the product language.

01	**FOUNDATIONS**	Visual and behavioral decisions: color, type, rhythm, layout, radius, borders, depth, icons, motion, accessibility.
02	**TOKENS**	Named contracts translate those decisions into primitive, semantic, and component variables.
03	**COMPONENTS**	Typed controls consume component tokens and own states, focus behavior, and accessibility.
04	**PATTERNS**	Product compositions reuse components. Patterns should not invent private controls or duplicate token values.

Core rule

Change raw values at the primitive or preset layer. Use semantic intent in product code. Add component tokens when a reusable component needs a stable contract. Compose patterns from existing components.

Do not skip layers

A one-off hex color in a CTA or a new radius inside a form creates a second, invisible design system. If a visual decision repeats, name it in tokens; if a behavior repeats, add it to the component API.

SECTION 02

Foundations

Ten decisions define the visual grammar before a component is styled.

Foundation	Contract	Implementation cue
Color	Semantic roles and restrained brand emphasis	Use brand/status aliases; never signal status with color alone
Typography	Heading, body, label, caption, and code hierarchy	Use the shared type classes and font variables
Spacing	8px rhythm with 4px half-steps	Choose from the token scale; avoid arbitrary gaps
Layout	Containers, measures, columns, responsive composition	Content-first breakpoints and no horizontal overflow
Radius	6 / 12 / 16 / 18 / full	Pill is reserved for compact controls and badges
Borders	Subtle / default / strong / focus	Cards use borders before shadows
Elevation	Four deliberate shadow levels	Only genuinely floating surfaces get prominent depth
Icons	Lucide, consistent size and stroke	Decorative icons are hidden from assistive technology
Motion	90 / 150 / 240 / 360 / 520ms	Use shared easing and reduced-motion behavior
Accessibility	Keyboard, focus, contrast, semantics, targets	Accessibility is part of the component contract

Signature palette

Warm paper neutrals, ink-like text, one orange brand accent, and muted supporting status colors.

Signature structure

Friendly rounded controls, border-led hierarchy, generous section rhythm, and restrained shadows.

SECTION 03

Token architecture

Tokens are dependencies, not a flat bag of CSS variables.

Layer	Owns	May reference	Product code?
Primitive	Raw ramps and scales	Literal values	No
Semantic	Intent for light/dark UI	Primitive tokens	Yes
Component	Stable component contracts	Semantic tokens	Via component APIs

Example dependency chain

```
brand-600: #c94e0c
brand-primary (light): brand-600
button-primary-background: brand-primary
<Button variant="primary">Save</Button>
```

Primitive files

src/tokens/primitives.ts and src/tokens/typography.ts

Semantic file

src/tokens/semantic.ts defines intent and light/dark mappings

Component file

src/tokens/components.ts defines reusable control contracts

CSS output

src/tokens/css.ts and src/app/globals.css expose the variables to Tailwind

Rule of thumb

If the name describes a hue or measurement, it is probably primitive. If it describes intent, it is semantic. If it describes a reusable component part, it belongs to the component layer.

SECTION 04

Themes and presets

Theme changes mode; preset changes brand personality. Components remain unchanged.

Preset	Personality	Notable override
Nusa	Warm orange, paper-like, generous	Default token values
Ocean	Crisp blue, tighter surfaces	Blue ramp and smaller radii
Ember	Warm orange, pill controls	Orange ramp and full control radius
White & Black	Monochrome and compact	Grayscale brand ramp and compact corners

Theme preference

system, light, or dark; persisted under nusa:theme.

Preset preference

nusa, ocean, ember, or white-black; persisted under nusa:preset.

DOM contract

data-theme-preference, data-theme, and data-preset live on the root html element.

No component fork

A preset changes CSS variables; Button, Input, Card, and CTA do not need preset-specific props.

Add a preset

Copy one entry in src/tokens/presets.ts, give it a stable id, override primitive or component aliases, then regenerate/use the existing CSS pipeline. Prefer the smallest override set that expresses the new identity.

SECTION 05

Use Nusa in another project

Today the repository is private, so source transfer is the reliable path. The package path becomes valid after @nusa/react is published.

Step	Action	Result
1	Confirm Next 16.2+, React 19.2+, Tailwind 4, TypeScript 5	Compatible runtime
2	Copy the foundation file set before any component	Token dependency graph is intact
3	Install only dependencies required by selected components	No unnecessary package weight
4	Import tokens before @theme inline in globals.css	Utilities resolve semantic variables
5	Copy selected UI components and their transitive dependencies	Reusable production primitives
6	Add root providers and pre-hydration theme initialization	Correct focus, overlay, and theme behavior
7	Run typecheck, lint, production build, keyboard and responsive QA	Verified integration

Required foundation files

```
src/tokens/{types,primitives,semantic,typography,components}.ts
src/tokens/{presets,registry,css,index}.ts
src/app/globals.css
src/lib/{cn,motion,theme}.ts
src/components/theme/theme-provider.tsx
src/app/layout.tsx  # integration reference
```

Distribution decision

Source transfer works now and keeps ownership explicit. When a real package is published, install @nusa/react and import its token stylesheet instead; do not assume the package exists merely because documentation shows the future package name.

SECTION 06

Component inventory

Thirty-six components are organized by behavior. Select only what the target product needs, but keep shared contracts intact.

Actions / 4

Button, Icon Button, Link, Toggle Group

Forms / 9

Label, Input, Textarea, Select, Combobox, Checkbox, Radio Group, Switch, Rating

Display / 12

Badge, Kbd, Badge Notification, Avatar, Card, Alert, Toast, Skeleton, Progress, Spinner, Table, Separator

Navigation / 6

Tabs, Navigation Menu, Breadcrumb, Pagination, Accordion, Disclosure

Overlays / 5

Dropdown Menu, Popover, Tooltip, Dialog, Drawer

Component contract checklist

OK	Use typed variants and sizes instead of one-off styling props.
OK	Map backgrounds, content, border, radius, shadow, and motion to tokens.
OK	Preserve hover, active, focus-visible, disabled, loading, error, and open states where relevant.
OK	Forward refs and native attributes when the underlying element supports them.
OK	Use Radix primitives for focus management and ARIA behavior that should not be reimplemented.
OK	Document anatomy, usage, accessibility, API, and a live preview.

Source convention

Most components live at src/components/ui/{slug}.tsx. Badge Notification is the documented exception and maps to src/components/ui/notification-badge.tsx.

SECTION 07

Pattern inventory

Patterns combine components into reusable product decisions. They remain token-driven and do not duplicate component internals.

Pattern	Kind	Primary building blocks
CTA	Transfer-ready	Input, Button, Avatar
Bento Cards	Transfer-ready	Card, Badge, layout
FAQ	Transfer-ready	Accordion
Option with Avatar	Transfer-ready	Avatar, selection control
Docks	Transfer-ready	Icon Button, Tooltip
Animated Tags	Transfer-ready	Badge-like actions, Motion
Action Search Bar	Transfer-ready	Input, Button
Customize Design System	Recipe	Checkbox, sections, generation flow
Form Field	Recipe	Label, input control, message
Search and Filter	Recipe	Input, Select, removable filters
Empty State	Recipe	Icon, copy, action
Confirmation Dialog	Recipe	Dialog, Button
Settings Panel	Recipe	Form controls, save behavior
Table Toolbar	Recipe	Search, filters, bulk actions

Composition rule

If a pattern needs a new primitive behavior, improve the primitive first. The pattern may control layout and content flow, but the component continues to own states, keyboard behavior, and visual tokens.

SECTION 08

AI handoff workflow

Give an AI coding agent the system context before asking it to slice or customize external UI.

Order	Instruction for the agent
1	Read AGENTS.md and the relevant framework documentation before editing.
2	Inspect existing tokens, component APIs, navigation, docs format, and nearby patterns.
3	Inventory the requested reference: screenshot, URL, pasted code, or uploaded DESIGN.md / SKILL.md.
4	Map every visible part to an existing component; list genuine gaps before creating anything.
5	Implement by composing existing components and semantic tokens. Add missing reusable APIs at the component layer.
6	If a new component is necessary, add source, typed API, token contract, docs entry, preview, navigation, and accessibility notes.
7	Verify lint, typecheck, production build, keyboard, reduced motion, responsive layout, and theme/preset behavior.

Reference-image rule

Use screenshots for visual hierarchy, content, density, and composition - not as permission to hardcode colors or recreate existing controls. State what is observed, what is inferred, and which Nusa component/token implements each part.

Portable context files

```
AGENTS.md                # repository-specific rules
SKILL.md                 # when and how AI must use this system
DESIGN.md                # extracted visual and product rules
design-tokens.json       # machine-readable token values
src/data/design-system-transfer.ts  # transfer manifest
```

Non-negotiable prompt line

Reuse the existing Nusa design-system components and semantic tokens first. Do not rebuild Button, Input, Avatar, Card, text styles, or overlay behavior locally when an equivalent already exists.

SECTION 09

Verification and troubleshooting

A design system is portable only when its behavior survives integration, not merely when the first screenshot looks correct.

- OK Every selected component compiles with strict TypeScript and exposes no duplicate exports.
- OK ESLint and the production Next.js build pass.
- OK Light, dark, and all four presets update semantic and component colors.
- OK Keyboard traversal, focus return, Escape behavior, labels, and ARIA names work.
- OK Reduced-motion users do not receive decorative movement.
- OK Layouts remain usable at 320px, 390px, tablet, and desktop widths without horizontal overflow.
- OK Buttons, inputs, cards, and patterns use system tokens; no unexplained palette/radius copies remain.
- OK Interactive documentation shows real production components and mirrors current props.

Common failures

Symptom	Likely cause	Fix
Components are unstyled	Token CSS missing or imported too late	Import tokens before @theme inline
Borders appear black	Tailwind 4 currentColor default	Apply the shared base border rule
Preset does not affect CTA	CTA contains literal color or bypasses component tokens	Use semantic aliases and shared Button/Input
Theme flashes on load	Root theme attributes are applied after hydration	Run themeInitScript before interactive UI
Copied pattern diverges	Local primitive was recreated	Replace it with the existing component API

Where to continue

Open /docs/overview for searchable discovery, /docs/overview/view-all for the full visual inventory, /docs/installation for runtime setup, and /docs/the-design-system plus /docs/ai-usage-guide for transfer and AI workflows.

END OF GUIDE

Nusa Design System v2.4 - source-aligned setup reference