GRIFFIN CSS

UI/UX Design Case Study

A lightweight, modern CSS framework designed to help developers build beautiful, responsive interfaces without writing custom CSS from scratch.

PROJECT

Griffin CSS — Component Library & Design System

RESEARCHER & UI/UX DESIGNER

Ali Ahmad Khan

PLATFORM

griffin-css.netlify.app

ROLE

End-to-end UI/UX Design, Frontend Development

Project Overview

Griffin CSS is a pure CSS framework offering 130+ ready-to-use components across 14 categories — buttons, cards, forms, navigation, modals, dashboards, and animated elements. The goal of the project was to design a system that is instantly usable (a single CDN link, zero build tools), visually distinctive, and fully themeable through CSS custom properties, while keeping the footprint under 8kb minified and gzipped.

130+	14	~8kb	0
Components	Categories	Min + Gzip	Dependencies

Deliverables

- A documented design token system (color, spacing, type, radius, shadow)
- 130+ production-ready UI components with copy-paste HTML
- Light & dark mode theming out of the box
- A marketing landing page and interactive component documentation site

The Problem

Developers building small-to-mid sized projects often reach for heavy frameworks like Bootstrap or Tailwind, which either come with significant setup overhead, opinionated build pipelines, or a steep learning curve for teams that just want a polished UI fast. There was a clear gap for a framework that is genuinely drop-in: one file, no configuration, and no JavaScript dependency, while still looking modern rather than generic.

Design Objectives

- Make setup frictionless — a single CDN <link> tag with zero build tools or npm.
- Keep the bundle lightweight (target: under 10kb minified + gzipped) without sacrificing component breadth.
- Build a token-based design system so any project can be re-themed by overriding a handful of CSS variables.
- Ship dark mode as a first-class citizen, not an afterthought.
- Maintain accessibility basics — visible focus states, sufficient contrast, ARIA-friendly markup patterns.
- Present the framework itself through a landing page and docs site that demonstrate the components in use.

Target Users

Frontend Developers	Solo Builders & Indie Hackers	Design-minded Engineers
Need fast, reliable components without maintaining custom CSS for every project.	Want a polished look for landing pages and MVPs without hiring a designer.	Want a system they can retheme quickly via design tokens for client work.

Design Process & Approach

01 **Audit & Benchmark**
Reviewed existing frameworks (Bootstrap, Tailwind, Bulma) to identify friction points — build tooling, bundle bloat, and generic default styling — that Griffin CSS needed to avoid.

02 **Design Tokens First**
Defined the entire visual language as CSS custom properties before building components: accent color, background/surface/text colors, border radius, shadow, spacing scale, and font families. This became the single source of truth for theming.

03 **Component Architecture**
Grouped components into 14 categories (Buttons, Cards, Forms, Navigation, Modals, Feedback, Layout, Dashboard, Landing Pages, Animated, and more), using a consistent .g- prefix naming convention for predictability and to avoid class collisions.

04 **Dark Mode by Design**
Built every component to respond automatically to a single data-theme="dark" attribute, rather than retrofitting dark styles after the fact.

05 **Landing Page & Docs**
Designed the marketing site to double as a live demo — every section on the homepage uses the framework's own components, and a dedicated documentation site shows live previews with copy-ready code.

06 **Performance Pass**
Trimmed unused rules and consolidated repeated patterns to bring the final file down to roughly 8kb minified and gzipped, with zero external dependencies.

Core Design Tokens

Every visual property in Griffin CSS is exposed as a CSS custom property, letting any project retheme the entire component set by overriding a small set of variables in :root.

Token	Purpose
--g-accent / --g-accent-dark	Primary brand color and its hover/active state
--g-bg / --g-surface	Page background and elevated surface (cards, modals)
--g-text / --g-text-muted / --g-text-hint	Text hierarchy — primary, secondary, tertiary
--g-border / --g-radius / --g-shadow	Structural styling — borders, corner rounding, elevation
--g-font-sans / --g-font-display / --g-font-mono	Typography roles across body, headings, and code
--g-success / --g-warning / --g-danger / --g-info	Semantic feedback colors
--g-space-4 / --g-text-base	Base spacing and type scale units

Component Categories

Buttons	Cards	Forms & Inputs	Navigation
Modals	Feedback / Alerts	Layout / Grid	Dashboard blocks
Landing sections	Animated components	3D / Perspective	Typography

Key UI/UX Decisions

- **.g- namespace:** every class is prefixed to prevent collisions with a host project's existing CSS.
- **One-line integration:** the entire framework loads from a single CDN link, removing setup friction identified as the top adoption barrier for CSS libraries.
- **Live-in-context docs:** component examples are shown with the exact HTML snippet beside the rendered preview, reducing the gap between reading documentation and shipping code.
- **Accessible defaults:** focus rings, adequate contrast ratios, and semantic HTML patterns are built into components rather than left to the implementer.

Results

- Shipped a complete framework — 130+ components across 14 categories — at roughly 8kb minified and gzipped, meeting the lightweight-footprint objective.
- Achieved true zero-config adoption: a single CDN link is the only integration step, with no build tools or JavaScript dependency required.
- Delivered a token-driven theming system, allowing any project to fully retheme the UI by overriding a small set of CSS variables in :root.
- Dark mode ships as a native, first-class mode across every component via one data attribute.
- The landing page and documentation site double as a working demonstration of the framework, with every section built using Griffin CSS's own components.

Reflection

Designing Griffin CSS reinforced that the biggest UX win for a developer-facing tool is often removing friction rather than adding features — the single-line CDN install and token-based theming did more for adoption than any individual component. Treating the documentation site as a live product surface, rather than a static reference, also made the design system easier to trust: what is documented is exactly what renders.

Tools & Skills

Design	Development	Delivery
Design tokens, component architecture, light/dark theming	HTML, CSS (custom properties), responsive/mobile-first layout	Netlify hosting, jsDelivr CDN, GitHub-based distribution