Design Systems & Frontend Architecture
Component-Driven UI Development

I design and prototype responsive web interfaces using Figma's design tokens API and component libraries, translating high-fidelity mockups into production-ready codebases. Implementation follows atomic design principles with headless UI foundations (Radix UI, TanStack primitives) styled via Tailwind CSS or CSS-in-JS (Styled Components, Emotion, or Panda CSS) for type-safe, scalable theming.

Technical Specifications:

Design Ops Pipeline: Figma variables → Style Dictionary transformation → token distribution via npm for design-to-code parity
Framework Agnostic: React/Next.js, Vue/Nuxt, SvelteKit, or Astro implementations depending on hydration requirements and content complexity
Accessibility-First: WCAG 2.1 AA compliance baked into component specs; automated a11y testing via Storybook and playwright-axe
Performance Optimization: Sub-3s LCP targets via critical CSS extraction, font subsetting with next/font or unplugin-fonts, and image optimization pipelines (Sharp-based transforms, AVIF/WebP fallbacks)
Infrastructure & Deployment Orchestration
Static & Edge-Deployed Hosting Architecture

I configure modern hosting platforms—Vercel, Netlify, Cloudflare Pages, or AWS Amplify—to serve static and server-rendered applications via global edge networks. Deployment pipelines integrate Git workflows with preview environments, atomic rollbacks, and automated SSL management.

Technical Specifications:

CI/CD Configuration: GitHub Actions or GitLab CI pipelines handling build cache optimization, secret injection via Doppler/1Password Secrets Automation, and Lighthouse CI regression testing
Edge Function Integration: Middleware for geolocation-based routing, A/B testing experiments, and authentication gating at the CDN layer
Domain & DNS Management: IAC (Infrastructure as Code) via Terraform or Pulumi for reproducible Route 53/Cloudflare DNS configurations with automatic failover and health checks
Serverless Compute: API routes deployed as Vercel Functions, Netlify Edge Functions, or Cloudflare Workers with KV/R2/D1 storage bindings for stateful edge operations
Content Management & Continuous Operations
Decoupled CMS & Maintenance Workflows

I implement headless CMS architectures—Sanity, Strapi, Payload CMS, or Contentful—providing editorial teams with structured content schemas while maintaining frontend flexibility. Post-launch operations follow Git-based content workflows with automated dependency management and monitoring telemetry.

Technical Specifications:

Headless CMS Integration: GROQ/GraphQL API consumption with real-time listener hooks for ISR (Incremental Static Regeneration) or on-demand revalidation
Version Control for Content: Sanity Content Lake with document revision history; field-level validation and custom input components for complex editorial requirements
Dependency Automation: Renovate or Dependabot configuration for automated security patch PRs with merge queue protection and build verification
Observability Stack: Logtail, Datadog, or Vercel Analytics for Core Web Vitals tracking; Sentry error monitoring with source map uploads; UptimeRobot or Pingdom for synthetic monitoring
Editor Training & Documentation: Handoff packages including Notion/Github Wiki documentation for component usage, CMS field references, and emergency rollback procedures
Delivery Methodology

Projects follow trunk-based development with Conventional Commits and automated semantic versioning. I ship working prototypes within 48 hours, iterate via Vercel Previews or Netlify Deploy Previews with shareable URLs for stakeholder feedback, then cut production releases through protected branch policies with required status checks.

Migration & Legacy Support: I handle WordPress-to-headless migrations (via WP GraphQL), static site generators (Jekyll/Hugo → modern stack), and "brownfield" integrations where new components must coexist with legacy infrastructure during transitional phases.