Frontend Architecture

Frontend Architecture Document

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Purpose & Scope

This document defines the frontend architecture for **{{PROJECT_NAME}}**. It covers framework decisions, folder structure, rendering strategy, routing, build configuration, performance targets, and operational considerations.

TODO: Define scope boundaries — which client applications are covered (web app, admin panel, marketing site, etc.).

---

2. Framework Choice & Rationale

Criterion	Weight	Next.js	Nuxt 3	SvelteKit	Remix
Team expertise	30%	{{SCORE}}	{{SCORE}}	{{SCORE}}	{{SCORE}}
Ecosystem maturity	25%	{{SCORE}}	{{SCORE}}	{{SCORE}}	{{SCORE}}
Performance	20%	{{SCORE}}	{{SCORE}}	{{SCORE}}	{{SCORE}}

Criterion	Weight	Next.js	Nuxt 3	SvelteKit	Remix
DX / tooling	15%	{{SCORE}}	{{SCORE}}	{{SCORE}}	{{SCORE}}
License / cost	10%	{{SCORE}}	{{SCORE}}	{{SCORE}}	{{SCORE}}

Selected Framework: `{{FRAMEWORK}}`

Rationale:

> TODO: Write 3-5 sentences explaining the final decision.

Node/runtime version: `{{NODE_VERSION}}` **Package manager:** `{{npm | yarn | pnpm | bun}}`

---

3. Project Folder Structure

```
{{PROJECT_NAME}}/
├── src/
│   ├── app/              # Route segments (App Router) or pages/
│   ├── components/
│   │   ├── ui/           # Primitive design system components
│   │   ├── features/     # Feature-scoped composite components
│   │   └── layouts/      # Page layout wrappers
│   ├── hooks/            # Shared custom hooks / composables
│   ├── lib/              # Pure utilities, third-party wrappers
│   ├── stores/           # Client state (Zustand / Pinia slices)
│   ├── services/         # API client, data-fetching layer
│   ├── styles/           # Global CSS, theme tokens
│   ├── types/            # TypeScript interfaces & enums
│   └── constants/        # App-wide constants
├── public/               # Static assets served at root
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── .env.example
├── next.config.ts        # (or framework config file)
└── package.json
```

TODO: Update tree to match actual project structure.

---

4. Rendering Strategy

Page Type	Strategy	Rationale
Marketing / landing	SSG	Maximum caching, no auth dependency
Dashboard / app views	SSR	Fresh data, auth-gated
Listing pages	ISR (revalidate: 60s)	Balance freshness vs performance
User-specific views	CSR	Dynamic, personalized data
API routes	Server	Thin BFF layer

Hydration approach: `{{Partial hydration | Full hydration | Islands}}`

TODO: Map every top-level route to a rendering strategy.

---

5. Routing Architecture

5.1 Route Organization

```
/                       → Home (SSG)
/app/                   → App shell (SSR, auth-required)
/app/dashboard          → Dashboard
/app/[resource]/[id]    → Dynamic resource detail
/api/                   → API routes (BFF)
/auth/login             → Login (CSR)
/[...catchAll]          → 404 handler
```

5.2 Route Guards / Middleware

```
Middleware execution order:
1. Request logging
2. Authentication check (JWT validation / session lookup)
3. Redirect unauthenticated → /auth/login
4. Role-based route protection
```

```
5. Locale detection / redirect
```

TODO: List all protected route patterns and their required roles.

---

6. Build & Bundle Configuration

6.1 Key Config Options

Option	Value	Reason
Output	`standalone`	Docker-optimized deployment
Image optimization	Enabled	Next/Image with defined domains
Bundle analyzer	`ANALYZE=true`	On-demand local analysis
Source maps	Production: hidden	Security — upload to Sentry only
Compression	gzip + brotli	CDN-level, not server-level

6.2 Code Splitting Strategy

- **Route-level splitting:** Automatic per page/route segment
- **Component-level splitting:** `dynamic()` / `defineAsyncComponent()` for modals, heavy widgets
- **Third-party splitting:** Vendor chunk isolation for large deps (charts, editors)

TODO: Identify and document specific lazy-loaded components.

---

7. Performance Budget

Metric	Target	Tool
LCP (Largest Contentful Paint)	< 2.5s	Lighthouse, CrUX
FID / INP (Interaction to Next Paint)	< 200ms	CrUX
CLS (Cumulative Layout Shift)	< 0.1	Lighthouse
TTFB (Time to First Byte)	< 600ms	WebPageTest
Total JS bundle (compressed)	< 200 KB	Bundlesize CI check
First page load (mobile 4G)	< 3s	WebPageTest

Metric	Target	Tool
Lighthouse Performance Score	≥ 90	Lighthouse CI

CI enforcement: `TODO: link to bundlesize config or Lighthouse CI workflow`

---

8. Asset Management

8.1 Images

- Format: WebP primary, JPEG fallback, SVG for vector
- Optimization: Framework image component (`next/image` / `nuxt/image`)
- CDN: `{{CDN_PROVIDER}}` — origin: `{{STORAGE_BUCKET}}`
- Lazy loading: Native `loading="lazy"` + framework component

8.2 Fonts

- Self-hosted via `/public/fonts/` (no Google Fonts runtime requests)
- `font-display: swap` on all faces
- Subset to used characters if possible

8.3 Icons

- Library: `{{Lucide | Heroicons | custom SVG sprite}}`
- Delivery: Inline SVG via component (no icon font)

TODO: Finalize CDN domain and storage bucket configuration.

---

9. Internationalization (i18n) Strategy

Item	Decision
Library	`{{next-intl
Locale routing	`/[locale]/path` prefix
Default locale	`{{en}}`
Supported locales	`{{en, nb, de}}`
Translation format	JSON key-value (per locale, per namespace)

Item	Decision
Translation storage	`src/messages/[locale]/[namespace].json`
Pluralization	ICU message format
RTL support	`{{Yes

Translation workflow:

1. Developer adds key with English string
2. `TODO: extraction tool` generates translation keys
3. Translator fills missing keys in Phrase/Lokalise/manual JSON
4. CI validates no missing keys before deploy

---

10. Error Boundary Strategy

Level	Scope	Behavior
Global `error.tsx`	Full page crash	Show branded error page, report to Sentry
Layout boundary	Section crash	Isolate — rest of page remains usable
Async component	Data fetch failure	Skeleton → error state UI
Form submission	Mutation failure	Inline error message + retry

Error reporting: `{{Sentry | Datadog | custom}}` **DSN:** `{{SENTRY_DSN}}` (environment variable — never hardcode)

TODO: Implement and test each boundary level.

---

11. Environment Configuration

Variable	Dev	Staging	Prod	Description
`NEXT_PUBLIC_API_URL`	`http://localhost:4000`	`https://api-staging.{{DOMAIN}}`	`https://api.{{DOMAIN}}`	Backend API base URL
`NEXT_PUBLIC_APP_ENV`	`development`	`staging`	`production`	Runtime environment flag
`SENTRY_DSN`	optional	required	required	Error reporting (server-side)
`NEXT_PUBLIC_SENTRY_DSN`	optional	required	required	Error reporting (client-side)

Variable	Dev	Staging	Prod	Description
`ANALYZE`	`true/false`	—	—	Enable bundle analyzer

Secrets management: All non-public secrets stored in `{{Vault / AWS Secrets Manager / Vercel env}}`. **`.env.example`:** Must be kept up to date — CI validates no undocumented variables exist.

---

12. Dependency Management Policy

Rule	Detail
Approval required for new deps	PR must include: bundle size impact, license check, last release date
Allowed licenses	MIT, Apache-2.0, ISC, BSD-2, BSD-3
Security audit	`npm audit` / `pnpm audit` in CI — fail on high/critical
Update cadence	Minor/patch: monthly automated PR (Renovate); Major: manual + reviewed
Banned patterns	No `moment.js` (use `date-fns`), no `lodash` (use native / `lodash-es`)

TODO: Configure Renovate or Dependabot with appropriate grouping rules.

---

13. Architecture Diagram

```
graph TB
    subgraph "Client Browser"
        Browser["User Browser"]
    end

    subgraph "Frontend Application — {{FRAMEWORK}}"
        CDN["CDN Edge (Static Assets)"]
        SSR["SSR Server / Edge Runtime"]
        Pages["Route Segments / Pages"]
        Components["Component Tree"]
        State["State Layer (Server + Client)"]
        Services["API Service Layer"]
    end
```

```
    subgraph "Backend"
        API["REST / GraphQL API"]
        Auth["Auth Service"]
    end

    Browser --> CDN
    Browser --> SSR
    SSR --> Pages
    Pages --> Components
    Components --> State
    State --> Services
    Services --> API
    SSR --> Auth
```

TODO: Refine diagram to reflect actual deployed infrastructure (Vercel, AWS, self-hosted, etc.).

Approval

Role	Name	Date	Signature
Author			
Tech Lead			
Architect			
Engineering Manager			

Revision #4
Created 2026-02-24 14:52:51 UTC by John
Updated 2026-05-25 07:32:45 UTC by John