### **Global Frontend Road-Map (12 clear steps)**

Follow the sequence top-to-bottom; each step produces tangible artefacts you can commit and tag. Citations show where every requirement comes from.

| **#** | **Deliverable** | **Concrete actions (✅ = merge gate)** | **Key sources** |
| --- | --- | --- | --- |
| **0 · Read-in** | *One-hour orientation sprint* | • Open *Navigation Map* and *Technical Spec* side-by-side. • Skim next-enterprise repo tree. | Routes list, stack spec, repo files |
| **1 · Repo hygiene** | Clean baseline | • pnpm install & pnpm lint must pass. • Delete boiler-plate pages except /app/page.tsx. | ESLint setup |
| **2 · Monorepo skeleton** | Feature-folders | • Create /modules/{ekoh,ethikos,…}. • Add /shared (layout, theme, hooks). ✅ commit “feat: scaffold feature folders”. | Folder scheme mirrors modules list |
| **3 · Theme + layout** | <MainLayout> & sidebar | • Build Ant-D based sidebar & top-bar. • Wire getLayout pattern (already planned). • Add <IntentCardGrid> placeholder for /. ✅ Storybook snapshot. | Routes for global shell |
| **4 · Route stubs** | 24 Next.js pages | • Generate one page file per path (/konsensus/page.tsx, …). • Each returns <ComingSoon> inside MainLayout. ✅ Playwright navigation test (no 404). | Route list |
| **5 · API baseline** | Typed hooks infra | • Install @tanstack/react-query. • Codegen TypeScript types from DRF schema (/api/schema/) or hand-write for now. • Create shared/api.ts: createClient(baseURL). ✅ useHealth() hook calls /api/health/ and passes. | Health route exists |
| **6 · Insights module (reference)** | Full vertical slice | • Implement components & pages exactly as doc 5.1 (charts, hooks, WS). • Use this as coding pattern for all other modules. ✅ /reports/\* pages render live dummy data. | Detailed spec |
| **7 · Replicable module loop** | *Run for each of the remaining 8 feature modules* → 1 Scaffold API hooks (useTopics, useProjects, …) from api\_router.py paths. 2 Build “dumb” UI components (cards, tables). 3 Compose route pages. 4 Add Storybook stories & unit tests. ✅ Playwright happy-path per module. | Functionality inventory & specs |  |
| **8 · State & caching rules** | React-Query policies | • Add default query client (staleTime 5 min, retry 1). • Module-specific overrides as in Insights (e.g. /reports/\* TTL 10 min). | Caching rules doc 5.2 |
| **9 · Observability hooks** | OTel browser traces | • Import @vercel/otel registration (already present) and wrap API calls. ✅ traces visible in local console. | instrumentation.ts exists |
| **10 · Access control** | Auth guard HOC | • Read JWT from cookie, redirect unauthenticated. • Protect pages flagged “ADMIN only” (e.g. /admin, /reports/export). | Endpoint ACLs |
| **11 · CI pipeline update** | Full green build | • Add pnpm test, pnpm playwright test, storybook-build. • Bundle-size check already scripted. ✅ GitHub Actions badge green. | Jest/Playwright scripts |
| **12 · Staging deploy** | Vercel or Docker image | • next build → static output + server bundle. • Smoke test health page & all routes. ✅ tag frontend-v14-alpha. | Deployment notes in cookiecutter stack (standard) |

#### **How to use the table**

1. **Print or pin it** in your project tracker – each row is a milestone.
2. **Never jump** ahead until the ✅ of the current row is merged; this keeps the tree clean.
3. After step 6 you have a working pattern; steps 7–9 are rinse-and-repeat for every module.
4. Flags like *ADMIN only* or special TTL are taken directly from backend docs, so you stay aligned without re-reading later.

This roadmap is **self-contained**; every reference is to documents already in your repo, so you can proceed without losing the thread.

Below is a concise “boiler-plate handbook” for **next-enterprise** in the Konnaxion context.  
 It confirms that the roadmap you drafted is compatible with the template and highlights extra guard-rails the template itself expects you to follow.

## **1 Does the roadmap fit the boiler-plate philosophy?**

| **Road-map step** | **Boiler-plate expectation** | **Verdict** |
| --- | --- | --- |
| **Feature folders (/modules/\*)** | The repo is purposely minimal at root; adding domain-scoped sub-trees is the intended extension point. | ✅ |
| **<MainLayout> + getLayout wrapper** | Template provides only /app/layout.tsx as a starting point; wrapping individual pages is recommended for larger apps. | ✅ |
| **React-Query hooks & no Redux** | The boiler-plate is un-opinionated on state, but ships Jest/Playwright/Storybook; React-Query integrates cleanly without extra global stores. | ✅ |
| **Storybook + Playwright gates** | Both are already wired in package.json scripts (storybook, e2e:\*) so using them as merge gates is the canonical flow. | ✅ |
| **Bundle-analyser & size budget** | A bundle-analyser toggle (ANALYZE env) and script are pre-included; planning a CI budget check leverages that. | ✅ |

In short, nothing in your 12-step plan conflicts with the template.

## **2 Template-specific recommendations & pitfalls**

| **Area** | **Best-practice** | **Why / Where it is defined** |
| --- | --- | --- |
| **Environment variables** | Add new vars to **env.mjs** using zod validation rather than sprinkling process.env.\* through code. The helper createEnv will force correctness at build time . |  |
| **Runtime rewrites / proxy** | Keep all backend calls behind the existing rewrites() in **next.config.ts** to avoid CORS headaches; adjust only the destination, never the source paths . |  |
| **Scripts lifecycle** | Use the provided scripts for every check (lint, test, storybook, e2e:\*, analyze). CI should call exactly these so local ≅ CI ≅ production . |  |
| **ESLint** | Do **not** move Tailwind classes into plain CSS; the Tailwind ESLint config is tuned to detect class typos and order rules . |  |
| **OpenTelemetry** | The template already instruments fetches in instrumentation.ts; keep your custom API hooks thin so traces stay meaningful. |  |
| **Absolute imports** | Leave tsconfig.json baseUrl & paths untouched; add new aliases under the "paths" section instead of using long relative paths. |  |
| **Patch-package post-install** | The default postinstall runs *patch-package* (see package.json) – if you add library patches commit them under patches/, otherwise remove the hook to speed installs. |  |
| **Testing pyramid** | Stick to the hierarchy baked into scripts: (1) Jest unit, (2) Storybook component tests, (3) Playwright e2e. Mixing Cypress or Vitest would duplicate infra. |  |
| **Static assets** | Put module-scoped images/svg under /public/{module}; the template’s next-image config expects assets there for built-in optimisation. |  |
| **Incremental adoption of App Router features** | The boiler-plate is already App-Router native. Prefer server components for data-heavy dashboards (e.g. Insights) and client components only where interactivity is essential. |  |
| **Deployment knobs** | Use the ANALYZE flag (already zod-typed) to toggle bundle analysis in staging; never enable it in prod builds. |  |
| **Security headers** | Keep reactStrictMode: true and do not remove the logging fetch wrapper; it surfaces slow API calls in development. |  |

## **3 Typical mis-steps to avoid**

1. **Deleting boiler-plate health routes** – /api/health is used by *next.config.ts* rewrites and by Playwright smoke tests. Keep or update it.
2. **Hard-coding API URLs** – always read from env.NEXT\_PUBLIC\_API\_BASE (you’ll need to add that key in env.mjs) so Vercel preview URLs work.
3. **By-passing the ESLint flat config** – adding a .eslintrc file in a sub-folder will clash with the top-level flat config; extend the flat config instead.
4. **Disabling strict mode to “fix” hydration warnings** – fix the warning, don’t turn off the guard.
5. **Skipping Storybook for complex charts** – the template’s CI runs test-storybook; missing stories break builds.

### **Bottom line**

Yes – the step-by-step roadmap you drafted is fully compatible with the **next-enterprise** boiler-plate.  
 If you align with the specific guard-rails above (env validation, rewrites, scripts, linting, tracing), the template will **work for you rather than against you**, and upgrades of its dependencies will stay painless.