AI Builder Starter Kit for Design System CLI

Blueprint for AI-assisted UI Development

If your design system already has a CLI that scaffolds a web app and drops in components, you can layer "AI-generated development" on top by giving the AI a contract to target and a toolchain
that turns AI output into real, reviewable code.

1) Define the "Design System Contract"

Export tokens.json, components.manifest.json, layout primitives, and common UX patterns so the AI knows exactly what to use.

2) Choose an AI Output Format (strict!)

Use JSON or YAML DSL as structured output. Examples were provided.

3) Add a Codegen "AI Builder" to Your CLI

Extend your CLI with a subcommand that validates, translates, and writes files.

4) Give the AI Strong, Reusable Prompts

Use a system prompt with schema enforcement and user prompts for pages.

5) Guardrails & Quality Gates

Schema validation, linting, unit tests, visual regression, a11y checks.

6) Typical Dev Loop

Run ai:build with a spec or prompt, review diffs, regenerate if needed.

7) Advanced: Data-aware Components

Include dataSource schema for GraphQL/REST, with formatting instructions.

8) Repo Structure

apps/web/src/pages, design-system/tokens.json, scripts/ai-build/*, etc.

9) What this unlocks

Consistent, rapid, safe, diff-friendly code generation with DS compliance.

Ready-to-run AI Builder Starter

A starter kit was provided as a ZIP archive. Below is the summary.

Folder structure:
ai-builder-starter/

```
├─ package.json (ai:build, ai:demo, ai:validate scripts)
├─ README.md
├─ design-system/tokens.json
├─ design-system/components.manifest.json
├─ scripts/ai-build/schema/ui-spec.schema.json
├─ scripts/ai-build/translate.mjs
├─ scripts/ai-build/run.mjs
├─ examples/claims-dashboard.spec.json
├─ prompts/claims-dashboard.md
└─ src/pages/
```

Quick Start

1. Unzip into your repo root
2. Run npm i
3. Try: npm run ai:demo
   -> generates src/pages/ClaimsDashboard.jsx

How to use with your Design System CLI

- Keep contract files updated (manifest + tokens).
- Validate spec JSONs with ai:validate.
- Generate React code with ai:build.

Wire your LLM

Replace the stub in run.mjs with your provider call.
Use --prompt prompts/claims-dashboard.md to generate a spec via AI.

Customize next

- Extend schema with events/slots.
- Map dataSource props to GraphQL/REST hooks.
- Point manifest imports to your real DS packages.
- Add ESLint/TS, Playwright + axe, visual snapshots in CI.