My Ramp Process: Illustrative Prompts

▼ Architectural Analysis

@codebase I need a rapid architectural onboarding for this repo.

Please analyze the workspace and provide an executive summary covering:

1. The Stack & Infrastructure: Confirm the backend framework (NestJS?), frontend framework (Qwik?), and database/ORM strategy.
2. The Dependency Graph: Briefly explain how `apps/frontend` communicates with `apps/backend` (is it REST?)
3. Key Logic Location: Find the exact file where the "Shorten URL" logic lives in the backend, and the corresponding form component in the frontend.
4. Constraint Check: Since this uses Qwik, are there specific serialization or "resumability" patterns I need to be aware of before creating new UI components?

Goal: I am about to build a client-side "Ephemeral Note" feature and need to know where to safely inject the route.

▼ Iterative Planning

I am building a "Client-Side Ephemeral Note" feature in this Qwik/NX repository.

Please generate a step-by-step Implementation Plan consisting of 4-5 distinct, copy-pasteable prompts that I can feed back to you to build the feature iteratively.

Feature Specs:

1. Architecture: Stateless. The note content is compressed using `lz-string` and stored in the URL hash (e.g., `/note#compressed_data`).
2. Stack: Qwik (Frontend), Tailwind (UI).
3. Workflow:

- User toggles "Ephemeral Note" mode on the landing page vs "Shorten URL" mode.
- User types text → App compresses it → App updates the "Shortened URL" input with the hash link.
- Recipient opens link → App reads window.hash → Decompresses → Shows text.

Constraints to handle in the prompts:

- Qwik Serialization: anticipating issues with closures, ensure prompts ask for proper `$(...)` QRL wrapping.
- Hydration: Ensure the "Read" logic uses `useVisibleTask$` since `window.location` is not available on the server.
- Validation: The existing "Shorten" button likely rejects `localhost` or `note` paths. One prompt must specifically address patching this validation.

Output Format:
Please provide the response as a list of "Prompt 1", "Prompt 2", etc., where each prompt leads to a code block I can immediately implement.