Dark Mode Toggle - Example Spec

- **Title & Context**
  - *Feature:* Dark-mode toggle for a Svelte Todo app.
  - *Why:* Reduce eye strain, meet user preferences, and align visuals across web/app.
  - *Stakeholders:* Frontend (Svelte), Design, QA, Accessibility.
- **Objective**
  - Provide a **persisted**, **accessible**, and **flash-free** theme switcher with modes: `light | dark | system`.
- **Scope**
  - In: UI control, theme state, persistence, prefers-color-scheme integration, CSS tokens, telemetry.
  - Out: Theming for 3rd-party iframes, printable styles, per-project branding.
- **Non-Goals**
  - No per-list/per-view overrides.
  - No user-account server sync in v1.
- **Constraints**
  - Works without JS errors in private mode (localStorage may be unavailable).
  - No layout shift on first paint (prevent “flash of wrong theme”).
- **Interfaces & Contracts**
  - *Global theme attribute:* `document.documentElement.dataset.theme ∈ {'light','dark'}`.
  - *System mode policy:* When mode = `system`, set `data-theme` based on `matchMedia('(prefers-color-scheme: dark)')` and update on changes.

- *Persistence key:* `localStorage['todo.theme']` with value:

  ```
  { "mode": "light" | "dark" | "system", "version": 1 }
  ```

- *JSON Schema (v1):*

  ```
  {
    "type":"object",
    "properties":{
      "mode":{"enum":["light","dark","system"]},
      "version":{"const":1}
    },
    "required":["mode","version"],
    "additionalProperties":false}
  ```

- *Svelte store contract (TypeScript):*

  ```
  export type ThemeMode = 'light' | 'dark' | 'system';
  export interface ThemeState { mode: ThemeMode }
  // API:
  // get(): ThemeState
  // set(mode: ThemeMode): void
  // subscribe(fn): Unsubscriber
  ```

- *Component contract:* `<ThemeToggle />`
  - Props: `mode: 'light'|'dark'|'system'`
  - Events: `on:change` → `{ mode }`
  - A11y: role `button`, `aria-pressed` for binary toggle, or `role="menu"` for 3-state menu; keyboard: Enter/Space; focus ring visible.

- **Behavior & State Transitions**

- Initial load:
  - If valid `todo.theme` → use it.
  - Else follow system (`system`).
- Mode changes:
  - `light` → set `data-theme="light"`, persist.
  - `dark` → set `data-theme="dark"`, persist.
  - `system` → compute from media query, persist, and react to changes.
- Optional cycle order for single button: `system → light → dark → system`.

- **CSS Token Contract**
  - Define tokens once; dark overrides only:

```
:root { --bg: #ffffff; --fg: #0a0a0a; --muted: #6b7280; }
[data-theme="dark"] { --bg: #0b0f14; --fg: #f3f4f6; --muted: #9aa4b
2; }
body { background: var(--bg); color: var(--fg); }
```

  - No color baked into components; use tokens only.
- **A11y Requirements**
  - Contrast ≥ 4.5:1 for text.
  - Visible focus states.
  - Toggle reachable via keyboard; screen-reader label “Theme: {Light|Dark|System}”.
- **Error Model & Guardrails**
  - If `localStorage` throws: fallback to in-memory store; do not block UI.
  - If schema invalid: ignore stored value and reset to `system`.
  - Refusal policy (for LLM codegen): do **not** access external APIs; only DOM, storage, media queries.

- **Performance**
  - First-paint snippet (inline, before CSS) to avoid theme flash:

```
<script>
 try {
  const raw = localStorage.getItem('todo.theme');
  const mode = raw ? JSON.parse(raw).mode : 'system';
  const dark = mode==='dark' || (mode==='system' && matchMedia
('(prefers-color-scheme: dark)').matches);
  document.documentElement.dataset.theme = dark ? 'dark' : 'light';
 } catch(_) {/* default light via CSS; no-op */}
</script>
```

  - Keep toggle JS <2KB.
- **Telemetry (optional)**
  - Events: `theme_set {mode}`, `theme_system_changed {to:'light'|'dark'}`.
  - Metric: % users by theme; FOUC rate (should be ~0).
- **Acceptance Criteria (v1)**
  - App shows correct theme on first paint with **no flash**.
  - Toggle cycles modes and persists across reloads.
  - `system` mode updates live when OS theme changes.
  - `data-theme` reflects effective theme at all times.
  - All pages/styles use tokens; no hardcoded colors.
  - Keyboard and screen-reader operation verified.
  - Corrupted storage self-heals to `system`.
  - Unit tests cover store; E2E covers first paint and cycling.
  - No console errors in private mode.
  - Lighthouse a11y score ≥ 95.

- **Example Tests**
  - *Unit (store):* set→persist; invalid JSON → default; system flip triggers update.
  - *E2E:* cold load with stored `dark` renders dark tokens; change OS theme in `system` updates within 200ms; Tab→Space toggles; localStorage blocked still operates.
- **Rollout**
  - Ship behind `feature:theme_toggle` flag.
  - Beta to 10% users; monitor errors/telemetry; then 100%.
- **LLM Handoff Snippet**
  - *Instruction:* "Generate Svelte code that satisfies the spec. Do not invent colors; only use CSS variables. Expose `<ThemeToggle />` with the stated props/events. Implement `theme` store per contract. Include the first-paint snippet. Return files as `{path, content}` JSON."
  - *Expected files:* `src/lib/theme.ts`, `src/lib/components/ThemeToggle.svelte`, `src/app.css` token updates, `src/routes/+layout.svelte` hook-up.
- **Versioning**
  - Spec `theme-toggle@1.0.0`. Any contract change (prop/event names, storage shape) requires a minor/major bump and migration step.