

PR: Fix sticky form validation on Project Showcase form (flat sibling → nested hierarchy)

Bug finder & solution author: ctaxnagomi **Source:** Bug report `flat sibling structure bug.md` + console log `krackeddevs.com-1788636959266.log` **Target site:** krackeddevs.com — "Add / Edit Project Showcase" form (`.sf-card`) **Status:** ⚠️ **DevTools simulation only — not a deployed intervention** **Environment bug encountered on:** Desktop · Windows 11 · Dual-boot setup with WSL2 (browser: Chrome DevTools on Windows side)

⚠️ Scope & Ownership Disclaimer

This PR is prepared from a **client-side simulation performed in browser DevTools**. The DOM restructuring (nesting `[data-slot="form-item"]` containers) was applied live on the page to *validate the diagnosis and prove the fix works* — it was **never written into the site's source codebase, never committed, and never deployed**. Every visitor's browser still receives the original flat structure.

This document therefore serves as a **hand-off artifact**: it proves the root cause and the shape of the fix, but the decision to implement it — and the reliability, uptime, and correctness of krackeddevs.com in general — remains with **whoever owns, handles, and manages the site's codebase and infrastructure**. The bug was found and the proposed solutions were authored by **ctaxnagomi**, who is not intervening in the production site in any way.

🐛 Bug Report

Component: Project Showcase form (`textarea[name="description"]` inside `.sf-card` , Shadcn/Radix `data-slot="form-item"` pattern)

Symptom: The inline error *"Description must be at least 10 characters"* persists even when the textarea contains valid input (e.g. 71 characters). The `aria-invalid` attribute stays `true` . The user cannot submit the form despite meeting the requirement.

Diagnosis (DevTools-verified): The `[data-slot="form-item"]` elements — Project Title, Project Summary, Project Description — are **flat siblings**. With the Radix/Shadcn context-based validation pattern, sibling items collide in the same validation context:

- The validator appears to aggregate `textContent` of the sibling containers (labels + hints + inputs), producing the nonsensical "1 character group" symptom.
- Validation state becomes **sticky** — it triggers on `blur` / `submit` but never re-clears for the specific input, because state is not isolated per field.

Test matrix (observed on live page)

Structure	Input	<code>aria-invalid</code>	Error cleared?
Flat siblings	0 chars	<code>true</code>	✅ (correct)
Flat siblings	71 chars	<code>true</code>	❌ BUG
Nested hierarchy	0 chars	<code>true</code>	✅ (correct)
Nested hierarchy	11 chars	<code>false</code>	✅ FIXED

✅ Fix Applied (simulation)

Transition from a flat list to a **nested parent→child chain**, isolating each field's validation context:

```

<!-- Before (broken) -->

<div class="sf-card">

  <div data-slot="form-item"><label>Project Title</label><input name="title"></div>

  <div data-slot="form-item"><label>Project Summary</label><input name="shortDescription">

  <div data-slot="form-item"><label>Project Description</label>

    <textarea name="description"></textarea>

    <p data-slot="form-message">Description must be at least 10 characters.</p>

  </div>

</div>

<!-- After (fix) -->

<div class="sf-card">

  <div data-slot="form-item" class="sf-field">

    <label>Project Title</label>

    <input name="title">
  
```

```

<div data-slot="form-item" class="sf-field" style="padding-left: 20px;">

  <label>Project Summary</label>

  <input name="shortDescription">


  <div data-slot="form-item" class="sf-field" style="padding-left: 20px;">

    <label>Project Description</label>

    <textarea name="description" minlength="10"></textarea>

    <p data-slot="form-message">Description must be at least 10 characters.</p>

  </div>

</div>

</div>

</div>

```

- Nesting creates clear parent-child chains → validation contexts no longer collide → state clears correctly at ≥ 10 characters.
- `padding-left: 20px` preserves visual hierarchy so the nesting is not visually confusing.

Recommended Implementation Checklist (for the maintainer)

1. **Refactor the DOM** in the actual component source: flat `[data-slot="form-item"]` list → nested hierarchy (adapt to your framework's component tree, not literal HTML).
2. **Add** `minlength="10"` on the textarea as a native HTML5 fallback layer.
3. **Tighten selectors:** validation JS should target `textarea[name="description"]` by `name`, never the generic `data-slot` container.
4. **Re-test** the blur/submit validation path and screen-reader announcements (`aria-invalid` transition `true → false`).

Why This Fix Makes krackeddevs.com More Feasible Long-Term

- **Submission reliability = content pipeline reliability.** The showcase form is how projects enter the site. A validator that never clears means every user who types a valid description hits

a dead end — they churn before submitting, so the site's core content flywheel stalls. Fixing this removes a hard blocker on organic content growth.

- **Accessibility correctness.** Sticky `aria-invalid="true"` is a WCAG failure — screen-reader users are told the field is invalid forever. Fixing it keeps the site compliant as it grows an audience.
- **Trust signal.** Forms that reject valid input are the single strongest "this site is abandoned/buggy" signal to developers visiting a dev-community site like KrackedDevs. Long-term credibility of the brand depends on basics working.
- **Cheap now, expensive later.** This is a structural component fix — trivial today inside one form card. The longer it ships, the more copies of the broken pattern get forked into other forms (login, contact, apply), multiplying the remediation cost.
- **Preload hygiene (from the console log).** The exported log shows ~83 "preloaded but not used" warnings. Every wasted preload competes for bandwidth with the assets that matter; cleaning these up compounds with the validation fix into a measurably snappier, more professional site.

Data-Layer Recommendation: Scaling the Backend/Database

The same long-term lens should be applied behind the form. Recommendation for the submit → store → serve path:

Short term: SQLite (Cloudflare D1) is enough

- A project-showcase site's write volume is small (form submissions, not chat messages). **A single SQLite database — Cloudflare D1 if you're already on Workers — comfortably handles this at zero operational cost**, with point-in-time restore and no connection pooling headaches.
- Don't stand up a "fully manual" backend server (a VM + Postgres + a node service) for this. It adds patching, uptime, and backup burden with no benefit at this scale.

Per-user rows: hash, don't proliferate

- **Don't create one table (or one database file) per user.** That design collapses at scale: connection churn, migration fan-out, and impossible analytics.
- If the concern behind per-user isolation is **privacy or enumeration** (users guessing each other's IDs), keep a single table and **hash the identifiers**: store `user_id = sha256(raw_id + per-app-salt)` (or better, a random UUIDv4 issued at signup, never derived from email). Use a keyed hash (HMAC-SHA256) if the value must be unlinkable but still lookable-up.
- Result: **one flat projects table** (`id` , `owner_hash` , `title` , `short_description` , `description` , `created_at`) scales to millions of rows on one D1/SQLite instance — you will not outgrow it before the product itself needs re-architecture.

Mid term: keep the hybrid Workers + Workers AI pattern

The current hybrid — Cloudflare **Workers** for the request edge + **Workers AI** for inference — is the right direction and should be kept, because:

- Workers terminate close to users (low latency for both the form POST and AI calls).
- Workers AI runs inference without you operating GPU infrastructure; you pay per neuron and scale to zero.
- D1 (SQLite) sits beside it in the same runtime with zero network hops — the "frontend + AI + DB all at the edge" hybrid beats any self-hosted stack at this scale on cost and operational simplicity.

Migration path: manual backend → Workers (assets + API) → D1 (SQLite) for storage → Workers AI for any smart features (project tagging, description summarization) called directly from the Worker. One platform, one bill, no servers to babysit — which is exactly what "feasible long term" means for a small team running a community site.

Verification Steps for the Maintainer

1. Open the deployed form after applying the source fix.
 2. Leave Description empty, blur → error appears, `aria-invalid="true"`.
 3. Type 11+ characters, blur → error clears, `aria-invalid="false"`.
 4. Run Lighthouse/accessibility audit → no `aria-invalid` mismatch.
 5. Check DevTools console → confirm preload warnings shrink after pruning unused `<link rel="preload">` tags.
-

*Bug found by **ctaxnagomi** (DevTools observation on the live page), along with the proposed fix and data-layer suggestions. Simulation only; apply against your own component source. Site reliability remains owned by the krackeddevs.com maintainers.*