

INTERNAL TECHNICAL REPORT

Tarsco Bolted Tank

UX/UI Assessment

This assessment walks the running Tarsco Bolted Tank application with an estimator and sales lens rather than scanning code alone. The foundations are strong. The opportunity is not visual cleanup. It is interaction quality and role fit.

01 Executive Summary

Read this application as one shared tool serving two distinct audiences. **Estimators are engineers:** technically savvy, comfortable with domain jargon, and the owners of the pricing formulas. **Sales are roughly half of daily users,** less technical, and they live in the post-lock stages, the proposal, scope of work, and delivery. The product was built as an engineering tool, but half the people using it every day are not engineers. That single tension shapes the highest-leverage recommendations below.

The visual layer is in good shape: a clean Optics token system, few hardcoded values, and a polished, on-brand customer Proposal. Those are assets to protect. The work that matters most sits in interaction: how the app starts a task, reports progress and errors, keeps a user safe from a destructive click, and who it quietly leaves out.

0

ARIA attributes across
338 views

15

hardcoded values across
35 stylesheets

17

findings documented

2

findings corrected by live
testing

Why the confirmed set is trustworthy

Two initial findings were corrected by testing live: form density is acceptable for engineers, and the processing guard does not eject users to the dashboard. The findings below reflect the actual application, not static assumptions.

The single most important recommendation: treat estimators and sales as two audiences of one shared tool, beginning with a role and stage aware estimate view and a validated sales workflow. In parallel, a cluster of low-effort, high-trust fixes and a broad accessibility gap are worth addressing regardless of appetite.

02

How This Was Done

The review combined breadth and reality. Four parallel code-scan agents swept the codebase, then every candidate finding was walked live in the running application against seeded data, under an estimator and sales user lens.

- **First impressions and visual coherence.** Button styles, typography, color usage, spacing, and iconography across many screens, plus token and hardcoded-value coverage.
- **Feedback and system communication.** How the app acknowledges actions, reports loading and errors, and distinguishes success from failure.
- **Navigation, consistency, and cognitive load.** Wayfinding, the sidebar hierarchy, and whether screens ask for more mental effort than the task requires.
- **Accessibility.** ARIA, keyboard and focus support, semantic landmarks, color-only meaning, and labeling of controls.

The live walkthrough is what separates this report from a static scan. Walking the estimate and tank edit forms, setting an estimate to processing and triggering a guarded action, and deleting a comment in the running app corrected two findings that a code scan alone would have reported incorrectly. Those corrections are documented in full later in this report.

03 The Users

Three roles use the application. Understanding who they are, and where they work in the lifecycle of an estimate, is the frame for every finding that follows.

A	E	S
Admin	Estimator	Sales
Effectively a senior estimator. Full access, owns configuration and pricing rules.	An engineer. Technically savvy, owns the pricing formulas, and builds the estimate through the active stages.	Less technical. Works the post-lock stages: proposal, scope of work, deliver to customer.

A representative is an external commissioned rep, not an application user. Estimator and sales volume are roughly equal day to day, which is the crux: the tool is engineering-first, but about half of the people in it are not engineers.

03.01

The 16-Stage Status Pipeline

An estimate moves through sixteen statuses. The pipeline locks at **bid_review**: everything before it is active estimating work, and everything from the lock onward is sales-facing and no longer editable in the same way. Estimators own the stages before the lock; sales own the stages after it.



☐ Active, estimator-owned
 ☒ Lock point (bid_review)
 ☐ Locked, sales-facing

03.02

The Core Tension

The estimate view is identical for both roles at every status. It interleaves estimator knobs (margin, contingency, commission, field wage) with sales and customer data on one undifferentiated sheet. Domain jargon such as hoppers, portal frames, rings, and BOM, BOL, and RFQ is acceptable for estimators but raises the barrier for sales. The product never adapts to who is looking or what stage the estimate is in. That is the root cause behind several findings, and the reason a role and stage aware view is the lead recommendation.

04 Findings

Seventeen findings, grouped into four themes. Each carries a severity label, the observation, the user impact, and a file and line reference. Confidence is noted on every finding: verified live in the running app, confirmed in code, or, for the one strategic finding, flagged as needing validation.

04.01

Getting Started and Flow

HIGH

Landing page has no "start here" call to action

Observation. The dashboard is home for both roles, yet it offers no New Estimate or New Site action, and its empty state is a dead end.

User impact. A user lands with nowhere obvious to begin. The primary task, building an estimate, has no visible entry point from the first screen.

app/views/dashboards/show.html.slim:33

VERIFIED LIVE

HIGH

Estimate view is one screen serving two jobs

Observation. The estimate view interleaves estimator knobs (margin, contingency, commission, field wage) with sales and customer data on one undifferentiated sheet, identical for both roles and every status.

User impact. Sales users wade through engineering controls they never touch, and estimators get no emphasis on what matters at their stage. One layout serves neither audience well.

app/views/estimates/show.html.slim

VERIFIED LIVE

HIGH

Submit-only validation and reload loop on long forms

Observation. There is no inline validation, and HTML5 validation is globally disabled (browser_validations=false). An error on a long form costs a full reload and a scroll-hunt for the offending field.

User impact. Correcting a single mistake means resubmitting the whole form and losing your place. The cost compounds on the longest, most detailed forms.

CORRECTED LIVE. The forms themselves are well-structured and fine for engineers. The friction is the submit and reload loop, not form density.

`config/initializers/simple_form.rb:140`

CONFIRMED IN CODE

MEDIUM

Tank saves with a blank name

Observation. Tank#name carries only a uniqueness check, no presence validation. Clearing the name and saving produces a nameless tank.

User impact. Blank rows appear in lists, making tanks hard to tell apart and easy to mis-select.

`app/models/tank.rb:55`

VERIFIED LIVE

PATTERN

Inconsistent modal versus full-page for adding child records

Observation. Assemblies are added in a modal, while tanks and estimates force full-page navigation. There is no single "add a child" interaction model.

User impact. The interaction is unpredictable. Users cannot build a reliable habit for adding related records.

`app/views/estimates/show.html.slim:37`

CONFIRMED IN CODE

04.02

Feedback and Safety

HIGH Processing state: no progress, no proactive lockout, reactive-only refusal

Observation. A static "Processing, please wait" message shows with no spinner. All actions stay enabled, and guarded actions are refused only after a click, via a transient alert.

User impact. Users cannot tell whether work is happening or how long it will take, and they only learn an action is blocked after attempting it.

CORRECTED LIVE. The guard uses `redirect_back_or_to(root_path)`. Users are not ejected to the dashboard as first assumed. They stay on the referring page with a transient alert.

```
app/controllers/concerns/processing_guard.rb:8
```

```
app/views/estimates/_header.html.slim:11
```

VERIFIED LIVE

HIGH Pricing shows a bare "Loading" and a dead-end "Calculation Error"

Observation. The app's primary output renders as unstyled placeholder text with no skeleton or aria-busy state, and calculation failures render "Calculation Error" with no cause and no retry.

User impact. The most important number in the app appears as plain "Loading" text, and when it fails the user has no way to understand why or recover.

```
app/view_components/lazy_calculation_view.rb:18
```

```
app/views/lazy_calculations/show_error.html.slim:2
```

CONFIRMED IN CODE

HIGH Flash auto-dismisses in about five seconds; errors look like successes

Observation. There is no dismiss control and no persistent variant. Success and error differ only by color, and all messages use `role="alert"`.

User impact. An error message can vanish before it is read, and a colorblind user cannot distinguish a failure from a success.

```
app/views/application/_flash.html.slim:3
```

VERIFIED LIVE

HIGH Comment delete has no confirmation

Observation. Comment delete is a plain delete with no turbo_confirm, while every other delete in the app confirms first. It is one click and permanent.

User impact. A single misclick destroys a comment with no prompt and no undo.

```
app/views/comments/_comment.html.slim:18
```

VERIFIED LIVE

HIGH Zero-result index renders a blank page

Observation. A trailing `if @companies.any?` suppresses the helper's built-in empty state, so an index with no records shows nothing at all.

User impact. A user with no data sees a blank page and no guidance, and cannot tell whether the app is broken or simply empty.

```
app/views/companies/index.html.slim:8
```

VERIFIED LIVE

MEDIUM Delete confirmations are the least specific

Observation. Deletes prompt with a generic "Are you sure?" while duplicate and revise actions already name the record.

User impact. The most destructive action gives the least context, exactly where naming the record would prevent a mistake.

```
app/helpers/context_menu_helper.rb:50
```

CONFIRMED IN CODE

04.03

Navigation and Wayfinding

HIGH Flat 14-item sidebar; hierarchy collapses to one indicator; no search or recents

Observation. The daily workflow entry, Sites, is visually identical to eleven reference and configuration screens. Site, Estimate, and Tank all light up the single Sites link, and there is no global search or recents.

User impact. Users cannot separate work from configuration at a glance, cannot tell how deep they are, and have no fast path back into a recent estimate.

```
app/views/application/_sidebar.html.slim:6
```

VERIFIED LIVE

HIGH Wide tables are not wrapped for scroll; the app is largely non-responsive outside the sidebar

Observation. The `resource_table` helper emits a bare table, and only a handful of views opt into a scroll wrapper. Application-level breakpoints appear almost exclusively in the sidebar stylesheet.

User impact. Wide pricing tables clip on smaller screens, and the main content does not adapt away from a desktop width.

`app/helpers/view_component_helper.rb:31`

CONFIRMED IN CODE

04.04

Accessibility

HIGH Icon-only controls expose no accessible name; raw ligature text and tooltips

Observation. There are zero ARIA attributes across 338 views. The icon helper renders Material Symbols ligature text as content, so screen readers announce "edit_note", and the title attribute defaults to the raw icon name.

User impact. Screen-reader users hear icon codenames instead of actions, and sighted users see tooltips built from internal names.

`app/helpers/icon_helper.rb:24`

VERIFIED LIVE

HIGH No keyboard or focus support in custom JS; focus rings only on buttons; no main landmark or skip-nav

Observation. No custom Stimulus controller handles keydown, focus, or tabindex. The `:focus-visible` style exists only on buttons, so tabs, links, selects, and menu triggers show no focus. There is no main landmark and no skip-nav.

User impact. A keyboard-only user cannot see where focus is or operate custom controls, and cannot skip repeated navigation.

`app/javascript/controllers/active_item_controller.js:7`

CONFIRMED IN CODE

HIGH Out-of-spec values are conveyed by red text alone

Observation. Usable volume below the customer minimum is shown as red text only, with no icon and no label.

User impact. A colorblind user, or anyone scanning quickly, can miss that a value is out of spec, on a number that drives the estimate.

```
app/views/short_sheets/_show.html.slim:13
```

```
app/assets/stylesheets/components/details-sheet.scss:61
```

VERIFIED LIVE**MEDIUM Index search inputs are placeholder-only, with no label**

Observation. The hand-rolled index search forms rely on placeholder text with no associated label. The main resource forms use `simple_form` correctly.

User impact. Screen-reader users get no name for the search field, and the placeholder disappears once typing begins.

```
app/views/materials/index.html.slim:5
```

VERIFIED LIVE

05 What We Corrected

Two findings from the initial code scan were wrong, and the live walkthrough caught them. They are documented here in full rather than dropped silently, because knowing what the app does not do is as valuable as knowing what it does.

Correction 1: Form density is fine for engineers

The initial read was that the build forms are too dense and overwhelming. Walking the estimate and tank edit forms live corrected this. The forms are acceptable for estimators, who are engineers, and the tank edit form is a well-structured model. The real friction is the submit-only validation and reload loop, not the density of fields.

Correction 2: The processing guard does not eject users to root

The initial read was that acting during processing throws the user out to the root or dashboard. Setting an estimate to processing and triggering a guarded Duplicate action live corrected this. The ProcessingGuard uses `redirect_back_or_to(root_path)`, so the user stays on the referring page with a transient alert. They are not ejected to root.

06 Recommendations and Roadmap

The work sorts into three effort tiers plus one discovery track. The quick wins are workflow-independent and worth doing regardless of appetite. The role-aware view and the accessibility sweep are the substantial investments. The sales-workflow discovery is a prerequisite, not a build.

06.01

Tier 1: Quick Wins

Low-effort, high-trust fixes. Each is small and independent, and together they remove the sharpest edges a daily user hits.

- + **Fix the blank empty state.** Remove the trailing `if @companies.any?` and let the helper render its built-in empty state. A one-line change.
- + **Confirm comment deletes.** Add the same data: `{ turbo_confirm: ... }` the rest of the app already uses, so a comment is not destroyed on a single misclick.
- + **Give flashes a dismiss control and persist errors.** Let users close a message, keep error messages on screen, and style success and error distinctly with the correct aria-live per type.
- + **Name records in delete confirmations.** Match the specificity duplicate and revise already provide.
- + **Add presence validation to tank names.** Stop nameless tanks from reaching lists.

06.02

Tier 2: Role-Aware Estimate View

The lead structural investment. Make the estimate view aware of who is looking and what stage the estimate is in. Group estimator knobs (margin, contingency, commission, field wage) separately from sales and customer data, and emphasize what each role needs at each stage. Surface report generation, a top sales goal, as a first-class action rather than an item buried in

a kebab menu. Add a New Estimate and New Site call to action to the dashboard so the primary task has a visible starting point.

06.03

Tier 3: Accessibility Sweep

A focused pass that closes the broad accessibility gap. The individual changes are small; the value is in doing them together across the app.

- + Mark decorative icons aria-hidden, and require an aria-label or hover text on every icon-only control.
- + Add keyboard handlers and focus management to custom controllers, a global :focus-visible style, a main landmark, and a skip-nav link.
- + Pair every color-coded status with an icon or a label, starting with out-of-spec values.
- + Add visually-hidden labels to the hand-rolled search forms.
- + Default the resource table to a scroll wrapper and add content breakpoints.

06.04

Discovery Track: Sales Workflow

HIGH

Sales-facing UX has never been validated against real sales workflow

Observation. About half of daily users are sales, yet the sales-facing experience has never been checked against how sales actually work.

Recommendation. Run a short discovery session with one or two sales users before building a sales landing: what they check on landing, when they generate versus send the proposal, and what is missing. A best-guess redesign can be sketched, but only as a hypothesis to confirm with sales first.

NEEDS VALIDATION

06.05

Roadmap at a Glance

Track	Addresses	Effort
Quick wins	Blank empty state, comment-delete confirm, flash dismiss and persistence, delete copy, blank tank name	Minutes to a few hours each
Role-aware estimate view	One-screen-two-jobs, no dashboard CTA, buried reports	Substantial, structural
Accessibility sweep	Icon names, keyboard and focus, color-only status, search labels, table scroll	Moderate, best done together
Sales-workflow discovery	Unvalidated sales UX	One short session first

07

Next Steps

The quick wins can begin immediately and independently. The role-aware view and the accessibility sweep are the investments worth planning around. Before either touches the sales experience, one conversation should come first.

Let us start with a short sales-user discovery session

We recommend a focused session with one or two sales users to watch how they actually work: what they check when they land, when they generate versus send the customer Proposal, and what the current view makes hard. That session turns the sales landing from a best guess into a validated design, and it is the smallest step with the largest downstream payoff.

We would welcome the chance to walk through these findings together and shape the sequence around what matters most to your team.