eBey3 Frontend Architecture Audit

Generated March 01, 2026 — Covering Search Engine, SPA Navigation, and Scroll Restoration

---

AUDIT 1 — Search Engine Optimization & Best Practices

1.1 React InstantSearch Lifecycle: Input Debouncing

- **[CRITICAL] No debouncing on SearchBox.** The <SearchBox> at `search.tsx:207-216` has no `queryHook` prop. Every keystroke fires a network request to the Meilisearch proxy immediately, overwhelming the server and causing UI stutter on slow connections.

  Fix: *Add a* `queryHook` *with a 300ms debounce:* `<SearchBox queryHook={(query, search) => { clearTimeout(t); t = setTimeout(() => search(query), 300); }} />`

1.2 Routing & State Sync: History Stack Pollution

- **[CRITICAL] No writeDelay on InstantSearch routing.** At `search.tsx:413-433`, the routing uses raw `stateMapping` with no `historyRouter({ writeDelay })`. Every state change (keystroke, facet click) pushes a new browser history entry. This turns the Back button into a trap—the user must click Back dozens of times to escape the search page.

  Fix: *Wrap routing with* `history({ writeDelay: 400 })` *from* `instantsearch.js/es/lib/routers`*. This batches URL writes and uses replaceState for intermediate changes.*

- **[WARNING] Dual routing conflict.** `App.tsx:113-136` globally monkey-patches `history.pushState` and `history.replaceState` to inject `_scrollKey` metadata. InstantSearch's internal router also calls these methods. The two systems interfere: scroll keys are injected into InstantSearch state objects, and InstantSearch URL writes trigger App.tsx's scroll-save logic on every keystroke.

  Fix: *Instead of monkey-patching global history methods, listen for Wouter's route change events only. Use a Navigation API (or popstate + custom event) that doesn't intercept third-party routers.*

1.3 Meilisearch Index Configuration (Backend)

- **[WARNING] No custom rankingRules.** `meilisearch.ts:68-92` configures searchable/filterable/sortable attributes but never calls `index.updateRankingRules()`. Meilisearch defaults treat all attributes equally. For e-commerce, exact title matches should rank higher than description matches, and active listings above inactive ones.

  Fix: *Add custom ranking rules:* `['words', 'typo', 'proximity', 'attribute', 'sort', 'exactness', 'isActive:desc', 'views:desc']`*. The* `attribute` *rule respects the searchableAttributes order (title > description > brand > category).*

- **[WARNING] No Arabic/Kurdish stopWords.** Common words like ■■, ■■, ■, ■■, ■■■, ■■, ■■, ■■, ■ are not excluded. These short function words match nearly every listing and dilute relevance.

  Fix: *Call* `index.updateStopWords(['■■','■■','■','■■','■■■','■■','■■','■■','■■','■■■■■'])` *in initializeMeilisearch().*

- **[WARNING] No typoTolerance tuning.** Defaults allow up to 2 typos on any word. For a bilingual Arabic/Kurdish marketplace, this can produce false matches on short words.

Fix: *Configure* `index.updateTypoTolerance({ minWordSizeForTypos: { oneTypo: 4, twoTypos: 8 } })` *to prevent typo-matching on 2-3 character Arabic particles.*

1.4 Conditional Faceting & Dynamic Filters

- **[WARNING] Static filter panel—no category-aware facets.** `search.tsx:235-329` (FiltersSheet) renders the same four filters (category, saleType, condition, price) regardless of context. 22 specification-based filterable attributes (shoeSize, storage, ram, etc.) are configured in Meilisearch but never exposed in the UI.

  Fix: *Use InstantSearch's* `useRefinementList` *hook to read the current category refinement, then conditionally render spec-level RefinementLists (e.g., show shoeSize only when category = 'shoes').*

- **[WARNING] EmptySearchState receives empty suggestions and fallbacks.** At `search.tsx:166-176`, EmptySearchState is called with `suggestions=[]` and `fallbackListings=[]`. The component supports 'Did you mean?' suggestions and popular product fallbacks, but both are hardcoded to empty arrays. The user sees a bare empty state with no recovery path.

  Fix: *Populate* `suggestions` *from Meilisearch's typo-corrected query or a separate suggestions endpoint. For* `fallbackListings`*, fetch top-viewed active listings as a static React Query.*

1.5 Proxy Performance & Security

- **[OK] Master key is server-side only.** `meilisearch.ts:24` injects the Authorization header on the server. The client connects to `/api/meilisearch` with an empty API key.

- **[WARNING] No timeout on proxy fetch.** `meilisearch.ts (route):34` calls `fetch(url, init)` with no `AbortController` timeout. If Meilisearch Cloud hangs, the Express request hangs indefinitely, consuming a Cloud Run instance slot.

  Fix: *Add a 5-second AbortController:* `const ctrl = new AbortController(); setTimeout(() => ctrl.abort(), 5000); fetch(url, { ...init, signal: ctrl.signal })`

- **[WARNING] No rate limiting on proxy.** Combined with the undebounced SearchBox, a single user typing rapidly can generate 10+ requests/second through the proxy.

  Fix: *Add express-rate-limit to* `/api/meilisearch` *(e.g., 30 requests/10s per IP).*

- **[WARNING] No React error boundary around InstantSearch.** A global ErrorBoundary exists in App.tsx, but if the Meilisearch proxy returns a 502 or malformed JSON, InstantSearch may throw during render and crash the entire page tree. A local boundary would show a graceful fallback.

  Fix: *Wrap* `<InstantSearch>` *in a dedicated ErrorBoundary that renders a 'Search temporarily unavailable' message.*

AUDIT 2 — SPA Navigation & Scroll Architecture

2.1 History Stack Management: Push vs. Replace

- **[CRITICAL] InstantSearch spams pushState on every interaction.** Without `writeDelay`, typing 'Nike' generates 4 pushState calls (N, Ni, Nik, Nike), each of which also triggers App.tsx's monkey-patched pushState to save scroll + inject _scrollKey. The user must hit Back 4+ times just to leave the search page. This is the root cause of the 'Back button trap.'

  Fix: *Use* `history({ writeDelay: 400 })` *router from InstantSearch, which batches changes and uses replaceState for intermediate updates. Only the final settled state becomes a pushState entry.*

- **[OK] Wouter Link components use pushState correctly.** All internal navigation uses Wouter's `<Link>` component. No raw `<a href>` tags were found that would cause full-page reloads. Product cards, navigation links, and empty state buttons all route through the SPA.

- **[OK] replaceState is used correctly in non-search contexts.** Files like `main.tsx:21`, `use-auth.ts:57`, and dashboard pages use `window.history.replaceState()` for token cleanup and tab state—these don't pollute the history stack.

2.2 Scroll Restoration Implementation

- **[WARNING] Race condition: triple-setTimeout brute force.** At `App.tsx:156-158`, scroll restore fires at 50ms, 150ms, and 400ms blindly. None of these timeouts check whether the target page's DOM has actually rendered. If InstantSearch hits take 500ms+ to paint, all three attempts scroll to a collapsed layout and fail silently.

  Fix: *Replace setTimeout cascade with a MutationObserver or ResizeObserver that fires scrollTo once the content container reaches a minimum height, with a 2-second timeout fallback.*

- **[WARNING] __scrollRestoreHandled flag is checked but never set.** The page-level useLayoutEffect in search.tsx was planned to set `window.__scrollRestoreHandled = true` but this was lost during the Meilisearch merge. The flag is checked at `App.tsx:153` but never activated, so the App-level timeouts always run and may fight with any future page-level restore.

  Fix: *Either implement the page-level restore in search.tsx (set the flag in a useLayoutEffect after hits are painted) or remove the flag check entirely to reduce dead code.*

- **[OK] sessionStorage persistence is solid.** `scroll-storage.ts` correctly stores positions in sessionStorage with a 50-entry LRU cap. Positions survive tab refresh and back/forward navigation. Error handling silently degrades on quota exceeded or private browsing.

- **[OK] keepPreviousData prevents DOM collapse on filter changes.** `use-listings.ts:140` uses `placeholderData: keepPreviousData` from TanStack React Query. When filters change, the old grid stays visible during the fetch, preventing the flash-of-skeleton that would collapse the scroll height. Note: this only applies to the legacy listings hook; the InstantSearch page manages its own loading state.

2.3 Stale Layouts & Skeleton Loaders

- **[WARNING] InstantSearch skeleton collapses the DOM.** At `search.tsx:162-164`, when `status === 'loading'`, a `<ProductGridSkeleton count={12} />` replaces the hit cards. If this skeleton's total height differs from the previously rendered grid (e.g., different number of rows), the scroll restore target position may not exist yet, causing it to fail.

  Fix: *On popstate navigation, skip the skeleton entirely and keep the previous hits visible (similar to keepPreviousData). InstantSearch's* `stalledSearchDelay` *prop (default 200ms) can suppress brief loading states.*

2.4 Prefetching & Perceived Performance

- **[WARNING] No prefetching on product card hover.** When a user hovers over a product card (`search.tsx:70-156`), no data is preloaded. Clicking triggers a full fetch of the product detail from `/api/listings/:id`, resulting in a visible loading delay on the product page.

  Fix: *Add an* `onMouseEnter` *handler to ListingHitCard that calls* `queryClient.prefetchQuery({ queryKey: ['/api/listings', hit.id] })`. *The product page will then render instantly from cache.*

Summary: All Findings by Severity

Sev.	File : Line	Issue	Impact
CRIT	search.tsx:207	No SearchBox debouncing	Floods proxy with requests on every keystroke
CRIT	search.tsx:413	No writeDelay on routing	Back button trap; history spammed per keystroke
WARN	App.tsx:113-136	Monkey-patched pushState/replaceState	Conflicts with InstantSearch router
WARN	meilisearch.ts:68-92	No rankingRules configured	Suboptimal relevance for e-commerce
WARN	meilisearch.ts:68-92	No Arabic/Kurdish stopWords	Common particles dilute results
WARN	meilisearch.ts:68-92	No typoTolerance tuning	False matches on short Arabic words
WARN	search.tsx:235-329	Static filter panel	22 filterable specs never shown in UI
WARN	search.tsx:166-176	Empty suggestions/fallbacks	Bare empty state with no recovery
WARN	route meilisearch.ts:34	No fetch timeout on proxy	Hangs Cloud Run slot if Meilisearch stalls
WARN	route meilisearch.ts	No rate limiting on proxy	Amplifies undebounced client requests
WARN	search.tsx:405	No local ErrorBoundary	Proxy failure crashes entire page
WARN	App.tsx:156-158	Triple-setTimeout scroll restore	Misses DOM paint; brute-force approach
WARN	App.tsx:153	__scrollRestoreHandled never set	Dead code; planned feature not wired
WARN	search.tsx:162-164	Skeleton collapses DOM on back-nav	Breaks scroll restore target
WARN	search.tsx:70-156	No hover prefetching	Product page has visible load delay
OK	All .tsx files	No raw <a href> found	SPA transitions are clean
OK	scroll-storage.ts	sessionStorage with LRU	Survives refresh; graceful degradation
OK	use-listings.ts:140	keepPreviousData active	Prevents grid flash on filter change
OK	route meilisearch.ts:24	Master key server-side only	Client never sees credentials

Recommended Priority Order

- **P0 (ship-blocking):** Add SearchBox debounce + InstantSearch writeDelay. These two fixes eliminate the Back-button trap and reduce proxy load by ~90%.
- **P1 (next sprint):** Proxy timeout + rate limit. Ranking rules + stopWords. Error boundary around InstantSearch.
- **P2 (quality-of-life):** Scroll restore via MutationObserver. Hover prefetch on product cards. Dynamic filter panel. Populate empty-state suggestions.
- **P3 (polish):** Remove history.pushState monkey-patch. typoTolerance tuning. Synonyms dictionary.

End of audit — March 01, 2026