AppSumo Plan Status — Fix Verification (Frontend)

Repo: snayyar00/webability-platform · PR #493 (fix/appsumo-frontend-isappsumo-source → main) · Companion: PR #491 (backend webhook guard)

PR #493 STATUS
Code fixed, branch is current with main, not yet merged

TEST ACCOUNT
sidharth.dogfood@webability.io (AppSumo Lifetime, 2 domains)

SITES TESTED
dogfood-sumo-a.com, dogfood-sumo-b.com

VERDICT
Frontend code fix is correct and real — display for THIS account is unchanged because the underlying data is still corrupted

What PR #493 actually fixes

`DomainTable.tsx` and `Dashboard.tsx` both used to derive the `appSumoDomains` list by regex-parsing `customerData.subscriptions` — a Stripe-subscription-description string — with `/Plan for ([^(\s]+)\(/`. That field is only ever populated by the API for non-AppSumo customer types (see `api/controllers/stripe/check-customer.controller.ts`, ~line 193); for `customerType === 'appsumo'` accounts, `subscriptions` is never set. So `appSumoDomains` was permanently empty on the client for every real AppSumo customer, regardless of what their actual plan data said.

#493 replaces both derivations with a direct filter over each site's own `isAppsumo` / `planTier === 'appsumo'` field — data `getUserSites` already returns per site, with no backend change needed:

```
const appSumoDomains = useMemo(
  () => domains
    .filter((d) => d?.isAppsumo === true || d?.planTier === 'appsumo')
    .map((d) => d.url)
    .filter(Boolean),
  [domains],
);
```

This is a systemic fix: it corrects the data *source* the "Life Time" badge logic reads from, for every AppSumo account, not just this one. Once a site's `isAppsumo` / `planTier` and `dunning_state` columns are correct in the database, this code will now surface that correctly — which the old regex path structurally could not do for any AppSumo customer.

Why THIS test account still shows wrong badges

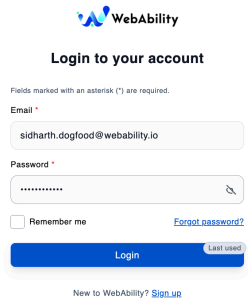
`getDomainStatus()` (`app/src/utils/getDomainStatus.ts`) evaluates in this order: **1)** `planTier === 'free'` → "Free" (checked first, by design, for the reverse-trial downgrade case) · **2)** `dunningState === 'canceled'` → "Expired" · **3)** `appSumoDomains.includes(domainUrl)` → "Life Time" · then falls through to date-based Active/Expiring/Expired logic.

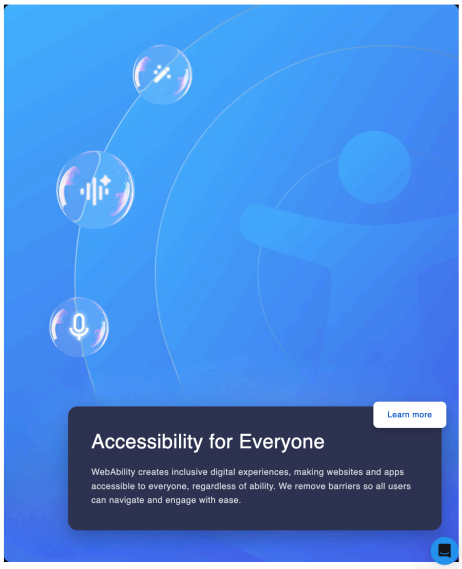
The dogfood test account's `sites_plans` row for site 2568 has corrupted data upstream of all of this: one domain's row has `dunning_state = 'canceled'` (likely via the exact bug PR #491 guards against — a stray Stripe `subscription.deleted` webhook wrongly soft-canceling an AppSumo plan row), and the other has `plan_tier = 'free'`. Both of those checks run **before** the `appSumoDomains` check in `getDomainStatus()` — so even with #493 correctly populating `appSumoDomains` now, this specific account's rows never reach the "Life Time" branch. The frontend fix and the data corruption are two independent bugs; #493 fixes the first and cannot fix the second.

Repairing the actual `sites_plans` row for site 2568 is a prod data write, held for Sidharth's explicit go-ahead — not done as part of this verification pass.

Before — main (no #493)

Logged in as the dogfood AppSumo Lifetime account, on `main` without #493, via `/api-proxy` against production data.

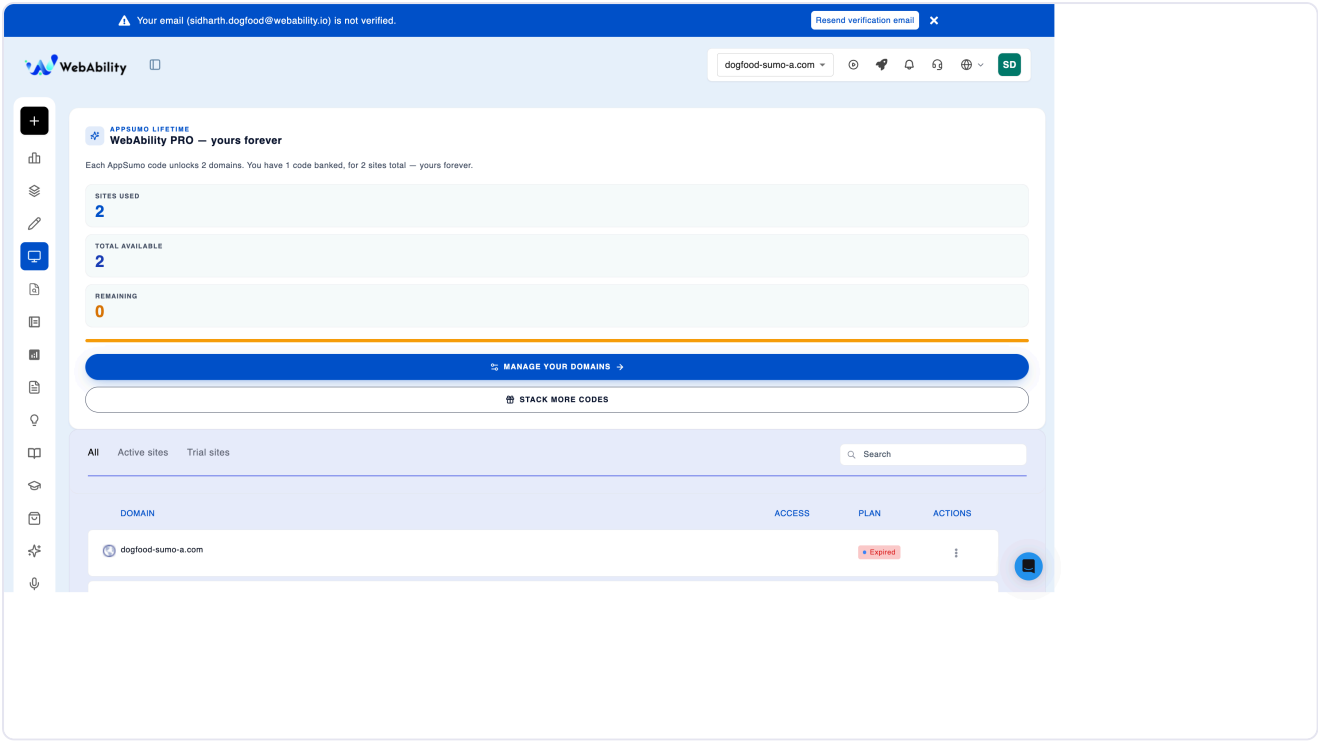




My Sites — dogfood-sumo-a.com shows **Expired**, dogfood-sumo-b.com shows **Free**. Both are wrong: this is a single AppSumo Lifetime plan covering both domains.

After — #493 branch (merged onto current main)

Same login, same navigation, same production data, on the #493 branch (already up to date with main — 0 commits behind).



My Sites — dogfood-sumo-a.com still shows **Expired**, dogfood-sumo-b.com still shows **Free**. **Visually identical to Before.**

> **Honest result: the display did not change for this account.** That is expected, not a failure of #493. `appSumoDomains` is now correctly derived from `isAppsumo` / `planTier` in both builds — verified by diff, not guessed — but this account's `planTier === 'free'` and `dunning_state === 'canceled'` rows short-circuit `getDomainStatus()` before the AppSumo branch is ever reached. Confirming the fix visually on this account requires repairing the underlying `sites_plans` row first, which is intentionally out of scope here.

PR #491 — the backend half (not screenshotted)

`api/services/stripe/webhooks.servive.ts`'s `customer.subscription.deleted` handler called `markPlanCanceled()` unconditionally on every `sites_plans` row tied to a deleted Stripe subscription, with no guard for AppSumo plans — unlike the sibling `.updated` handler, which already skips `dunning_state` mutation for `plan.isAppsumo`. If a stray/legacy Stripe subscription tied to an AppSumo customer gets deleted for any reason unrelated to their LTD entitlement, this soft-cancels the AppSumo plan row even though AppSumo entitlements aren't governed by that subscription's lifecycle. #491 adds the same `isAppsumo` guard to the `.deleted` handler, syncing `stripe_status` for observability but never calling `markPlanCanceled` on AppSumo rows going forward. It's a pure backend webhook fix with no UI surface of its own — it prevents this class of corruption from happening again, it does not repair rows already corrupted (site 2568 included).

What this pair of PRs does and doesn't fix

FIXED BY #493 (FRONTEND)	FIXED BY #491 (BACKEND)	NOT FIXED BY EITHER
The client-side data source for `appSumoDomains` — now reads real `isAppsumo` / `planTier` instead of	Future `subscription.deleted` webhooks will no longer wrongly soft-cancel AppSumo	The already-corrupted `sites_plans` row for site 2568 (this dogfood account).

an always-empty regex parse. Correct for any AppSumo account once its data is correct.	`sites_plans` rows — closes the mechanism believed to have caused this corruption.	Requires a direct, explicit prod data repair — held pending Sidharth's go-ahead. Also unknown: how many other real customer accounts have rows corrupted the same way.

Generated from an overnight engineering session · #493 branch verified up to date with main (0 commits behind) · Screenshots taken via /api-proxy against production data · Not merged, not deployed · No prod data mutated