

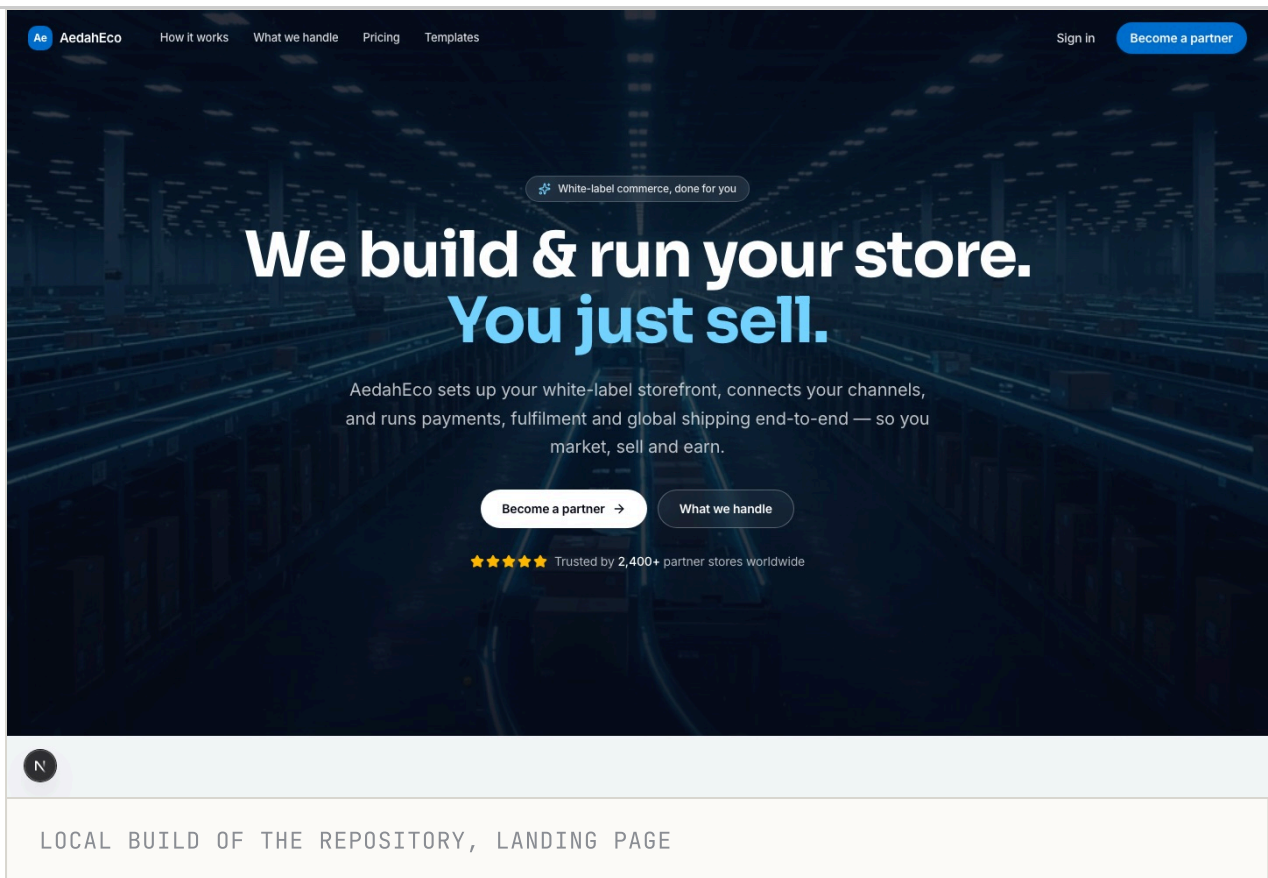
Aedah Commerce Suite: designing a merchant-of-record marketplace, economics first

Most marketplace prototypes start with screens. This one started with the money. Aedah Commerce Suite (internally AedahEco) is a B2B commerce platform where the platform is the merchant of record between product suppliers and the...

Role: Sole architect and product designer

Stack: Next.js 16, React 19, Tailwind CSS v4, shadcn/ui

Status: Frontend product demonstration with fully mocked data. There is no backend yet. This page describes design and frontend work, and is careful not to describe it as a live system.



Context

Most marketplace prototypes start with screens. This one started with the money. Aedah Commerce Suite (internally AedahEco) is a B2B commerce platform where the platform is the merchant of record between product suppliers and the channels where products are sold: suppliers list products, resellers distribute them through many sales channels, every order is collected centrally, and the platform settles money outward while keeping a margin. Before the first component was built, the business model was written down as a versioned set of documents, and the frontend was built to make that model tangible.

The model, as designed

Five actors, deliberately separated

- ♦ **Admin.** Platform operator: tenants, merchants, partners, customers, orders, payments, settlement, CMS, audit logs.
- ♦ **Tenant.** The supplier, manufacturer or distributor who is the real source of products. Hidden from partners.
- ♦ **Merchant.** A brand or store listed under a tenant. This is what partners browse and resell.
- ♦ **Partner.** An affiliate-style API reseller who sells the catalogue through their own channels for commission.
- ♦ **Affiliate.** A referral marketer working with links and commissions.

Tenants are hidden from partners on purpose. If a reseller could see who sources a brand, they could go direct and cut the platform out. The documentation calls this what it is: the supplier relationship is the platform's moat, so the information architecture protects it.

Markup economics

Retail price equals the tenant's base cost plus the platform markup. From every order, the tenant is paid base cost minus a supplier-side fee, the partner's commission is paid out of the markup, and what remains of the markup is platform profit. Plans and setup fees exist to cover fixed costs and gate resource usage; they are not the profit centre. The settlement and payout logic, including tax handling, is specified in its own documents.

Channels as configuration

Every sales channel maps to one of five connection archetypes, so building five integrations covers dozens of channels. OAuth-install channels include Shopify, BigCommerce, Wix, Amazon, eBay, Walmart, Noon, Etsy, TikTok, QuickBooks, Xero and Square. Credential-paste covers WooCommerce and custom APIs. Managed feeds cover Google Merchant, Meta, Pinterest, Snapchat, WhatsApp, YouTube and Microsoft. The design principle in the docs is "adding a channel is config, not code".

Storefront pricing with a performance unlock

Partners on hosted web storefronts pay a monthly plan that guarantees a capacity floor (products and merchants), and trailing three-month sales can unlock capacity beyond the paid tier. Effective capacity is the greater of the plan floor and the sales unlock. The entry plan is priced for emerging-market partners.

What was built

A complete Next.js 16 frontend for all five portals plus the public storefront, onboarding, pricing and templates, running on mocked data so every screen can be walked through end to end. Eight numbered design documents and a changelog live in the repository alongside the code, so the reasoning is versioned with the product.

Why it belongs in the portfolio

The value here is not the UI. It is the demonstration that I design the marketplace economics, the information asymmetry that protects them, and a scalable integration architecture before writing the screens, and that I document the reasoning so a team could build the backend from it. I would rather present it accurately as a design and frontend artefact than imply a production system that does not exist yet.