

AfroZeal Technical Handover

Buyer-safe technical handover for AfroZeal.com

1. Product and system overview

AfroZeal is a single-page web application for a multi-provider African fashion marketplace. It contains four main areas:

- A public editorial website and searchable marketplace.
- Shopper accounts, cart, checkout, order history, profile, and discovery tools.
- Provider onboarding and a dashboard for storefronts, products, orders, payouts, billing, and assisted tools.
- An independent platform-owner console for operations, branding, provider pricing, requests, and owner credential management.

The browser application uses a hosted backend for data, files, server functions, and real-time updates. Account identity is handled by an external authentication service. Payments use a payment processor for customer checkout, provider subscriptions, connected-provider onboarding, and payouts. Assisted workflows call a generative service from server-side actions.

All external services must be placed under buyer-owned accounts during handover. Never reuse seller credentials in a production deployment.

2. Main application areas and roles

Public area

The public area includes the homepage, marketplace, product detail, artisan storefront, informational pages, contact form, newsletter form, personal styling flow, and bespoke commission form.

Shopper area

Authenticated shoppers can manage a cart, complete checkout, view order history, update profile and fit information, and manage account security through the authentication provider.

Provider area

Providers complete connected-payout onboarding, activate the provider plan, configure a storefront, manage products and images, review orders, update fulfilment status, and use assisted business tools.

Platform-owner area

The owner console uses independent credentials and sessions. It provides platform metrics, user status controls, commission and contact workflows, whitelabel settings, provider plan settings, and credential rotation.

3. Architecture overview

Browser application

- React 19 and TypeScript.
- Vite 8 for development and production builds.
- React Router 7 for public, shopper, provider, and owner routes.
- Tailwind CSS 4 for styling.
- Framer Motion and GSAP for interface motion.

Application backend

- Convex database and indexes.
- Queries and mutations for authenticated application data.
- Actions for Stripe and Gemini calls.
- Convex file storage and upload URLs.
- Public HTTP endpoint for payment webhooks.

Identity

- Clerk supplies shopper and provider authentication.
- The first successful sign-in requires a permanent shopper or provider role choice.
- Provider and shopper functions check the authenticated identity and ownership.
- The platform-owner console uses separate salted password hashes and hashed, expiring session tokens.

Payments

- Stripe Payment Element handles customer payment entry.
- Stripe PaymentIntents are created from a server-side cart snapshot.
- Stripe Connect Express handles provider onboarding and payout eligibility.
- Stripe Billing handles the recurring Artisan Pro plan, including a \$0 starting amount.
- A signed webhook confirms successful payments, creates provider-grouped orders once, and attempts eligible transfers.

Assisted tools

- Gemini 2.5 Flash is called only from a Convex action.
- Inputs and results are saved to the requesting user's history.
- Outputs are decision support and require human review.

4. Project structure

The transferred source package uses the following relative layout:

```

src/
components/ Shared, public, commerce, provider, and auth UI
pages/ Public, account, provider, and owner screens
App.tsx Route tree and route protection
main.tsx Browser service bootstrap
index.css Theme and shared styles
convex/
schema.ts Data model and indexes
users.ts Account records and permanent role selection
providers.ts Provider profile, publishing, payouts, and orders
products.ts Provider product management and public catalogue
cart.ts Shopper cart operations
payments.ts Checkout, billing, transfers, and payment actions
http.ts Public webhook route
ai.ts Assisted tool action and saved run history
admin.ts Owner operations
adminAuth.ts Owner credentials and sessions
seed.ts Protected, duplicate-safe installation seed
seedData.ts Demo settings, providers, and products
files.ts File upload helpers
public/ Static brand assets, product images, and hosting rules
docs/ Requirements, installation, and verification material
.env.example Names-only configuration contract
package.json pnpm scripts and dependencies
vite.config.ts Build configuration

```

Do not rename core folders until the application is running under buyer-owned services and a baseline build has passed.

5. Data model

The main Convex tables are:

- **users:** external identity, email, permanent role, status, and onboarding state.
- **providers:** storefront identity, story, specialties, publication state, payout state, and subscription state.
- **consumers:** shipping and optional style-profile data.
- **products:** catalogue details, prices, storage references, sizing, stock, provenance, status, and featured state.
- **cartItems:** shopper cart quantity and selected size.
- **checkoutIntents:** immutable checkout snapshot, total, shipping data, payment reference, and status.
- **orders:** provider-grouped items, totals, shipping details, payment and transfer references, fulfilment status, and payout status.
- **plans:** provider plan definition, amount, availability, and Stripe mapping.
- **subscriptions:** provider subscription status and period details.
- **platformSettings:** public name, tagline, logo, accent colour, contact email, and custom-order fee.
- **commissionRequests:** shopper briefs, budget, date, measurements, and workflow status.
- **aiRuns:** assisted-tool inputs and outputs owned by the requesting user.
- **webhookEvents:** processed payment event identifiers for idempotency.
- **adminCredentials:** salted owner password hash and username.
- **adminSessions:** hashed owner session token and expiry.

- `contactMessages`: public contact submissions.
- `newsletterSubscriptions`: newsletter email records.

Indexes support identity lookup, role access, public storefront and product lookup, provider-owned data, cart access, order access, session validation, and webhook deduplication.

6. Required buyer-owned services

Create buyer-owned accounts and production projects for:

- Convex for the backend deployment, database, server functions, and file storage.
- Clerk for shopper and provider identity.
- Stripe for customer payments, subscriptions, connected accounts, payouts, and webhooks.
- Google Gemini for assisted tools, if those tools will be enabled.
- A source-control host for the transferred source package.
- Static web hosting with HTTPS and a dedicated site directory.
- Domain registrar and DNS management for AfroZeal.com.

The static browser build can run on ordinary shared Apache hosting. Vercel is not required. Convex remains the hosted application backend.

7. Configuration contract

Browser-safe variables

Create a local browser environment file from the supplied example and set:

```
VITE_CONVEX_URL=https://YOUR_CONVEX_DEPLOYMENT.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=YOUR_CLERK_PUBLISHABLE_KEY
VITE_STRIPE_PUBLISHABLE_KEY=YOUR_STRIPE_PUBLISHABLE_KEY
```

Only publishable values belong in browser variables.

Server-only variables

Configure the following in the buyer's Convex deployment or secret manager:

```
CLERK_JWT_ISSUER_DOMAIN=https://YOUR_CLERK_INSTANCE.clerk.accounts.dev
STRIPE_SECRET_KEY=YOUR_STRIPE_SECRET_KEY
STRIPE_WEBHOOK_SECRET=YOUR_STRIPE_WEBHOOK_SECRET
GEMINI_API_KEY=YOUR_GEMINI_API_KEY
SITE_URL=https://AfroZeal.com
SITE_ID=YOUR_CONVEX_SITE_IDENTIFIER
SETUP_SECRET=YOUR_RANDOM_ONE_TIME_SETUP_SECRET
INITIAL_ADMIN_USERNAME=YOUR_INITIAL_OWNER_USERNAME
INITIAL_ADMIN_PASSWORD=YOUR_INITIAL_OWNER_PASSWORD
```

Mark Stripe, Gemini, setup, and owner values as server-only. Never place them in a `VITE_` variable, browser bundle, public document, screenshot, issue, or chat message.

Use a high-entropy one-time setup secret. Rotate or remove it after the seed has completed successfully.

8. Installation sequence

Requirements

- Node.js 22 or newer.
- pnpm 10.
- Buyer-owned Convex, Clerk, Stripe, and hosting access.
- Gemini access if assisted tools are enabled.

Install and configure

From the project folder:

```
pnpm install --frozen-lockfile
```

Do not use npm. Keep the committed lockfile unchanged unless dependencies are intentionally updated.

Create the local browser configuration from `.env.example`. Set the browser-safe values, then configure server-only values in Convex.

For development, use two terminals:

```
pnpm exec convex dev
```

```
pnpm dev
```

Run the protected seed

The seed is a manual one-time action. It is designed to avoid duplicate platform settings, plan data, demo providers, demo products, and payment-plan linkage when run correctly.

The seed requires buyer-owned setup values and the required pre-uploaded file identifiers. It installs:

- One whitelabel platform-settings record.
- One monthly Artisan Pro plan with a \$0 starting amount.
- Four demo provider storefronts.
- Twelve demo catalogue products.
- One salted and hashed independent owner credential.

Do not run the protected seed on an established production dataset without first reviewing its inputs and taking a backup.

9. Authentication setup

- 1 Create the buyer's Clerk application.
- 2 Configure permitted origins and redirect URLs for local and production domains.
- 3 Add the publishable key to the browser environment.
- 4 Set the Clerk issuer domain in Convex server variables.
- 5 Configure the Convex authentication integration.

- 6 Test sign-up, sign-in, sign-out, and one permanent role selection for each role.
- 7 Confirm that a provider cannot open shopper-only operations and a shopper cannot open provider operations.

Clerk controls customer and provider passwords. The application stores identity references and role data, not those account passwords.

10. Payment, subscription, and payout setup

Customer checkout

- 1 Add the Stripe publishable key to the browser environment.
- 2 Add the Stripe secret key to Convex server variables.
- 3 Confirm that cart totals are calculated on the server from current product data.
- 4 Use Stripe test mode to create a checkout.
- 5 Confirm that the browser receives only a PaymentIntent client secret.

Payment webhook

The public backend route is `/stripe/webhook`. Configure the buyer's webhook endpoint or forwarding layer to send the original request body and the `Stripe-Signature` header to that route.

Subscribe to the events required by the current payment flow. Set the resulting signing secret as `STRIPE_WEBHOOK_SECRET`. Never parse and rebuild the request body before signature verification.

The successful-payment handler records event identifiers for deduplication. It creates grouped provider orders only once and attempts transfers only for eligible connected accounts.

Provider plan

The starting Artisan Pro plan is \$0 per month. The platform owner can change the amount. When the amount changes, the prior Stripe mapping is cleared so the next activation can create a matching recurring Product and Price.

Test plan activation and the billing portal in Stripe test mode before charging providers.

Connected provider payouts

- 1 Start provider onboarding from a provider account.
- 2 Complete Stripe Connect Express onboarding in test mode.
- 3 Return to AfroZeal and run the verification step.
- 4 Confirm that publication remains blocked until payout and plan requirements are met.
- 5 Complete a test purchase and confirm the expected provider grouping, processing-fee allocation, transfer status, and order state.

Stripe controls country availability, identity checks, capability status, settlement timing, and bank payouts.

11. Assisted-feature setup

Set GEMINI_API_KEY only in Convex server variables. The current action uses Gemini 2.5 Flash and stores each completed result with the requesting user.

Verify each workflow with neutral test data:

- Pricing benchmark.
- Product description translation.
- Cultural evidence review.
- Global trend localisation.
- Size and fit guidance.
- Personal styling support.

Treat all output as assistive. Do not present cultural evidence output as certification, price output as a guaranteed market value, or fit output as a guarantee.

If Gemini is not configured, disable or clearly label the assisted features before launch.

12. Build and deployment

Run targeted verification before the final production build:

```
pnpm verify:srs
pnpm typecheck
```

Then run one final build:

```
pnpm build
```

Upload the generated dist/ contents to the dedicated shared-hosting document root. Keep the included .htaccess file. It provides single-page route fallback, redirects to HTTPS when the host supports it, and adds defensive response headers.

Deployment checks:

- 1 Confirm the homepage loads over HTTPS.
- 2 Open a nested public route directly in a new browser session.
- 3 Test authentication redirects on the production domain.
- 4 Confirm the browser connects to the buyer's Convex deployment.
- 5 Confirm that public products and storefronts load.
- 6 Confirm that file uploads use the buyer's storage.
- 7 Deliver a signed Stripe test webhook.
- 8 Complete one test checkout and inspect the resulting order.

13. Security and credential rotation

- Transfer no reusable seller secret into production.

- Rotate domain, hosting, source-control, Convex, Clerk, Stripe, Gemini, and owner credentials during handover.
- Store server secrets only in buyer-owned deployment variables or a secret manager.
- Change the temporary owner credential at first sign-in and confirm that older sessions are revoked.
- Remove or rotate the one-time setup secret after seeding.
- Keep webhook signature verification and event deduplication enabled.
- Never expose secret keys in browser variables, logs, public error messages, screenshots, or documentation.
- Review account suspension and owner access regularly.
- Apply dependency and security updates in a controlled branch and test before production release.

14. Backup, monitoring, and maintenance

Backups

- Schedule database exports or backups using the buyer's Convex plan and operating policy.
- Keep separate records of domain ownership, DNS settings, service accounts, and environment variable names.
- Back up original production media when licensing and retention rules require it.
- Test restoration steps before relying on the backup process.

Monitoring

- Monitor frontend availability and HTTPS expiry.
- Review Convex function errors and usage.
- Review Clerk authentication failures and permitted origins.
- Review Stripe webhook failures, payment status, transfer failures, disputes, and connected-account capability changes.
- Review assisted-feature errors and usage limits without logging sensitive user input unnecessarily.

Maintenance

- Apply dependency updates in planned batches.
- Run type checking and SRS verification before the final build.
- Review public policies when payment, provider, fulfilment, tax, or privacy practices change.
- Remove abandoned demo accounts and content before commercial operation.

15. Functional verification checklist

Public and shopper

- Homepage, navigation, footer, and mobile menu render correctly.
- Marketplace search, category filter, and price filter update results.
- Product and storefront pages open directly.

- Contact, newsletter, personal styling, and custom-order forms behave as expected.
- Shopper sign-up, permanent role choice, cart, checkout, order confirmation, order history, profile, and security settings work.

Provider

- Provider sign-up routes to onboarding.
- Connected payout onboarding and return verification work.
- Provider plan activation works in test mode.
- Storefront fields and media save correctly.
- Product create, edit, image upload, stock, status, and featured controls work.
- Storefront publication remains gated until required conditions are met.
- Orders load and fulfilment status changes persist.
- Each assisted workflow returns and saves a result when enabled.

Platform owner

- Owner login rejects incorrect credentials.
- Overview counts load.
- Users can be suspended and restored.
- Custom requests and contact statuses persist.
- Branding changes appear publicly.
- Provider plan amount and availability save.
- Owner credential changes revoke older sessions.

Payments and external services

- A signed successful-payment event creates orders only once.
- Provider grouping and payout status are correct.
- Test settlement and connected-account capability checks pass.
- Domain, DNS, HTTPS, email delivery, identity checks, and production payment behavior are verified with buyer-owned accounts.

16. Product and support boundaries

The transferred application includes the source package and implemented workflows described in this guide. It does not prove production uptime, live payment settlement, active provider relationships, customer traffic, revenue, legal compliance, tax treatment, content licensing, or third-party account approval.

Codeblix provides 30 days of hands-on technical support for the current product. Support covers transfer guidance, buyer-owned service configuration, deployment assistance, credential rotation, and questions about existing features. It does not include unlimited development, new features, ongoing marketplace operations, marketing, provider recruitment, or legal and tax advice.