

# HerJungle Technical Handover Guide

## Product and system overview

HerJungle is a React single-page marketplace for rare indoor plants. It has public discovery pages, buyer and seller identity flows, grower storefronts, inventory management, marketplace checkout, order tracking, reviews, seller subscriptions, and a separate owner area.

The browser application uses Convex for application data, server functions, realtime updates, and file storage. Clerk provides buyer and seller identity. Stripe provides Checkout, Billing, Connect Express onboarding, destination payments, and signed webhook events. Optional order email delivery uses Resend. The frontend builds to static files that can run on an Apache-compatible shared host when single-page routing is configured.

## Main application areas and roles

- Public visitor: home, catalogue, product details, grower storefronts, pricing, about, contact, privacy, and terms.
- Buyer: account access, shopping bag, shipping details, payment, order history, and delivered-purchase reviews.
- Seller: onboarding, plan selection, storefront branding, payout connection, inventory, incoming orders, and fulfilment status.
- Platform owner: separate protected login, site settings, plan management, user controls, support messages, and credential rotation.

## Architecture summary

### Browser layer

React 19 and React Router provide the page tree and protected navigation. TypeScript defines shared application types. Tailwind CSS, application CSS, Framer Motion, and Lucide icons provide the visual system. The cart is kept in browser storage until Stripe confirms payment and the created order appears.

### Application layer

Convex queries read public and role-scoped data. Mutations change profiles, inventory, orders, configuration, and administrative records. Actions call Stripe and other external services. Convex storage holds seller branding and plant images.

### Identity layer

Clerk signs in public buyers and sellers. Protected Convex functions use the server identity subject to scope account data. The platform owner uses a different server-managed session with an expiring token. The owner credential must be changed immediately after transfer.

### Payment layer

Stripe Billing handles paid seller plans. Stripe Connect Express handles grower onboarding and payout readiness. Marketplace orders use Stripe Checkout and destination payments to the selected grower's connected account. The server reads current prices, stock, ownership, and quantities before it creates a payment session.

Signed webhook processing is the source of truth for completed payments. Each Stripe event is recorded once so a repeated event does not create a duplicate order or reduce stock twice.

## Technology stack

- React 19 and React DOM.
- TypeScript 5.8.
- Vite 7 project tooling with a Vite 6-compatible development dependency range in the current package metadata.
- React Router 7.
- Tailwind CSS 4, Framer Motion, and Lucide React.
- Convex 1.31 for data, server functions, realtime subscriptions, and storage.
- Clerk React for buyer and seller identity.
- Stripe for Checkout, Billing, Connect, and webhooks.
- Browser image compression for seller uploads.
- Vitest and Testing Library for unit and component checks.
- Playwright for browser flow checks.
- pnpm 10 with a committed lockfile.

## Project structure

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| App.tsx            | Route tree, providers, and route guards                       |
| components/        | Public, seller dashboard, and owner interface components      |
| pages/             | Public, buyer, seller, identity, policy, and owner pages      |
| lib/               | Application context, cart state, and shared data helpers      |
| convex/            | Schema, queries, mutations, actions, webhooks, and seed logic |
| public/            | Brand and catalogue assets                                    |
| docs/              | Product completion evidence                                   |
| tests/             | Browser flow checks                                           |
| .github/workflows/ | Build and deployment automation                               |
| package.json       | Scripts and dependency declarations                           |
| pnpm-lock.yaml     | Locked pnpm dependency graph                                  |

## Data model

### User records

User records connect a Clerk identity to a buyer or seller role. Seller fields include the store name, unique store slug, biography, branding storage references, selected plan, Stripe customer and subscription references, Connect account reference, and payout readiness.

### Plant records

Plant records store seller ownership, plant name, scientific name, price, image addresses and storage references, description, optional care information, optional lineage, stock, and category. Seller and category indexes support inventory and discovery queries.

### Order records

Orders store buyer and seller identity, immutable item snapshots, total, shipping address, payment session and event references, creation time, and one of three statuses: pending, shipped, or delivered.

### Review records

Reviews connect a buyer and plant with a rating, comment, display details, and date. Review eligibility requires a delivered order that contains the plant. A buyer cannot review the same plant twice.

## Platform configuration

The platform configuration stores the public site name, optional logo address, buyer onboarding price, and up to six seller plans. Each plan has an ID, name, monthly price, feature list, and optional Stripe price reference.

## Operational records

Separate records support owner credentials and expiring sessions, processed Stripe event IDs, contact messages, and optional email delivery results. These records are private and must never be returned through public projections.

## Required buyer-owned services

The buyer should create fresh accounts and values for:

- Convex production deployment for data, functions, and storage.
- Clerk application for buyer and seller authentication.
- Stripe account with Billing and Connect enabled.
- Static web hosting with HTTPS and single-page routing support.
- Optional Resend account and verified sender for order email.
- Domain registrar and DNS control for HerJungle.com.

Do not reuse seller secrets after handover. Public browser values may use public keys. Private Stripe, webhook, deployment, seed, and email values must remain server-only.

## Configuration placeholders

Create `.env.local` in the project folder for public frontend values:

```
CONVEX_DEPLOYMENT=prod:YOUR_CONVEX_DEPLOYMENT
VITE_CONVEX_URL=https://YOUR_CONVEX_DEPLOYMENT.convex.cloud
VITE_CONVEX_SITE_URL=https://YOUR_CONVEX_DEPLOYMENT.convex.site
VITE_CLERK_PUBLISHABLE_KEY=YOUR_PUBLIC_CLERK_KEY
VITE_STRIPE_PUBLISHABLE_KEY=YOUR_PUBLIC_STRIPE_KEY
```

Set private values in the Convex production environment:

```
CLERK_JWT_ISSUER_DOMAIN=https://YOUR_CLERK_ISSUER
STRIPE_SECRET_KEY=YOUR_SERVER_STRIPE_SECRET
STRIPE_WEBHOOK_SECRET=YOUR_WEBHOOK_SIGNING_SECRET
SITE_ID=YOUR_SITE_IDENTIFIER
SITE_URL=https://HerJungle.com
SEED_SECRET=YOUR_PRIVATE_SEED_SECRET
RESEND_API_KEY=YOUR_SERVER_EMAIL_KEY
EMAIL_FROM=HerJungle <orders@YOUR_VERIFIED_DOMAIN>
```

RESEND\_API\_KEY and EMAIL\_FROM are optional until email delivery is enabled. Never put server secrets in a VITE\_ variable.

## Setup and local verification

Requirements are Node.js 22 or later and pnpm 10 or later.

```
pnpm install
pnpm exec convex dev
pnpm dev
```

Use the existing lockfile. Do not switch package managers. Configure the buyer-owned development services before testing account or payment flows.

Targeted checks:

```
pnpm test
pnpm build
pnpm test:e2e
```

The build includes TypeScript checking. Browser checks require a configured Convex deployment and suitable demonstration data. Run payment checks in Stripe test mode before using live keys.

## Deployment sequence

- 1 Create the buyer-owned Convex production deployment.
- 2 Set the private server environment values in Convex.
- 3 Create the buyer-owned Clerk application and configure the issuer used by Convex.
- 4 Configure Stripe Billing, Connect, and the signed webhook endpoint.
- 5 Set the public frontend values for the production build.
- 6 Run the production build once with `pnpm build`.
- 7 Upload the generated `dist/` folder to an HTTPS static host.
- 8 Configure the host so unknown application paths return `index.html`.
- 9 Point HerJungle.com DNS to the buyer-owned host.
- 10 Review the seed content and run the idempotent seed only when demonstration data is wanted.
- 11 Sign in to the owner area, change the credential, review the brand, and confirm seller plans.
- 12 Complete the functional verification checklist before accepting real users or payments.

Shared hosting is enough for the frontend when it supports HTTPS, static files, and Apache-style single-page routing. Convex continues to run the backend. A separate frontend platform is not required.

## Payment setup

### Seller subscriptions

The pricing page reads seller plans from platform configuration. A free plan activates without Stripe. A paid plan creates or reuses a Stripe product and monthly price, starts Checkout in subscription mode, and saves the subscription state after signed webhook confirmation.

### Grower payouts

Seller onboarding creates a Stripe Express account and sends the seller through Stripe's hosted identity flow. The seller is treated as connected only after Stripe reports completed details, enabled charges, and enabled payouts.

### Marketplace checkout

The checkout action accepts plant IDs and quantities, then reads the seller, stock, and current prices on the server. One order can contain items from only one grower. Stripe Checkout includes the product items and a shipping line. Destination payment data sends product revenue to the connected grower.

### Webhook events

Configure the Convex HTTP endpoint to receive the Stripe events used by seller plans, Connect status, invoices, and completed checkout sessions. Keep the signing secret in the Convex environment. Test event replay to confirm duplicate delivery does not create duplicate orders.

## Authentication and owner access

Clerk must issue tokens accepted by the Convex authentication configuration. Test new buyer and seller accounts, role selection, protected routes, and sign-out. Public projections must not expose payment IDs or private account fields.

The owner area does not use Clerk. Its credential is stored as a salted password hash, and successful login returns an expiring server session. Transfer the initial credential privately, change it immediately, revoke old sessions, and do not place the credential in documentation or frontend files.

## Storage and email

Seller logo, banner, and plant image uploads use protected upload URLs. The browser compresses listing images before upload. Confirm that only the owning seller can update or remove their listings and associated references.

Email delivery is optional. If enabled, use a buyer-owned Resend account and verified sender. The application records sent, skipped, and failed delivery results and avoids sending the same order notice twice.

## Security checklist

- Replace all transferred public and private service values.
- Keep all server secrets outside frontend variables and public files.
- Rotate the owner credential and invalidate old sessions.
- Require Stripe signatures and retain event idempotency checks.
- Keep price, stock, seller ownership, and quantity validation on the server.
- Confirm buyer and seller reads are scoped to the signed-in identity.
- Confirm administrative mutations require a valid owner session.
- Use HTTPS for the website and webhook endpoints.
- Review privacy, terms, shipping, returns, taxes, and plant-health rules.
- Remove or replace demonstration users, plants, images, testimonials, and messages.

## Functional verification checklist

- Public home, catalogue, product, storefront, pricing, contact, privacy, and terms pages load.
- Buyer and seller sign-up, role selection, and protected navigation work.
- Seller store name, slug, logo, banner, and biography save correctly.
- Seller plan selection works for both free and test paid plans.
- Stripe Connect reaches ready status only after charges and payouts are enabled.
- Plant creation, editing, deletion, image upload, price, and stock controls work.
- A one-grower cart creates a server-priced Stripe Checkout session.
- Signed webhook processing creates one order and reduces stock once.
- Sellers can move orders through pending, shipped, and delivered.
- Buyers see their own orders and delivered buyers can leave one review per plant.
- Owner sessions expire, sign-out works, and the credential can be changed.
- Site settings, plans, user controls, and support messages work.
- Nested application routes load directly on the production host.

## Backup, monitoring, and maintenance

- Use Convex export and backup features on a schedule suitable for the buyer's risk level.
- Keep a copy of the application package and the exact lockfile used for each release.
- Monitor Convex function errors, Stripe webhook failures, payment disputes, and email failures.
- Review domain renewal, HTTPS expiry, Stripe account health, and service billing each month.
- Test restore and deployment steps before a major change.
- Apply dependency updates in a separate test branch and run targeted checks before release.
- Keep seed operations idempotent and never use seed content as production customer data.

## Product and support boundaries

HerJungle.com currently serves the branded application shell, but the buyer must verify all production integrations under buyer-owned accounts. No revenue, traffic, customers, transactions, uptime, or search performance are represented.

The buyer is responsible for hosting, service accounts, payment compliance, shipping rules, tax, privacy, refunds, plant-health policies, grower recruitment, customer support, marketing, backups, and ongoing costs. Codeblix provides up to 30 days of reasonable technical handover support. Ongoing operations, unlimited custom development, legal advice, and business results are outside that support boundary.