

RecipesMondial Technical Handover Guide

Operational guidance for the buyer and future operator

1. Product and system overview

RecipesMondial is a creator-commerce web application for recipe content. It has four main operating areas:

- A public marketplace for discovery, creators, storefronts, recipes, products, pricing, licensing, help, legal information, and contact.
- A customer area for purchases, memberships, licences, favourites, follows, notifications, billing, and a private library.
- A creator area for recipes, products, storefronts, orders, customers, memberships, licences, coupons, promotions, reviews, analytics, billing, and payout onboarding.
- A separate owner area for users, creators, content, transactions, subscriptions, reviews, reports, plans, settings, and credential rotation.

The public site is designed for RecipesMondial.com. Live operation requires buyer-owned accounts, rotated credentials, payment-provider approval, and end-to-end verification before real transactions are accepted.

2. Architecture overview

Browser application

The frontend is a React 19 single-page application written in TypeScript and built with Vite. React Router maps public, authenticated, creator-only, and owner routes. Tailwind CSS and the project styles provide the interface. Browser environment values are public by design and must never contain a secret.

Application backend

Convex stores application data and runs trusted server functions. Server functions enforce user and creator ownership, control premium content access, create payment sessions, process signed payment events, manage uploads, and supply dashboard data.

Authentication

Clerk handles customer and creator sign-in. Convex resolves the current identity and links it to an application user. Creator operations perform an additional creator-profile check.

The owner area uses a separate server-backed session. Owner credentials are stored as a salted hash, sessions expire, and protected owner operations check the session on the server. Change the transferred owner credentials immediately and invalidate old sessions.

Payments and payouts

Stripe Billing supports creator SaaS subscriptions. Stripe Connect supports creator product checkout, platform fees, connected accounts, and payouts. The payment webhook verifies signatures and records processed event identifiers so retries do not fulfil the same purchase twice.

One-time creator products use an application fee. Creator memberships use an application-fee percentage. Paid content access is created after a valid payment event. The application hides protected recipe details when the viewer lacks the required entitlement.

Hosting

Vite produces a static frontend in `dist/`. Shared hosting is sufficient when it provides HTTPS and redirects unknown frontend routes to `index.html`. Convex continues to host the backend functions, data, and payment endpoint.

3. Technology stack

- React 19 and React DOM for the browser interface.
- TypeScript for application and server code.
- Vite for local development and production builds.
- React Router for browser routing.
- Tailwind CSS and project styles for the visual system.
- Convex for data, server functions, file storage metadata, and HTTP routes.
- Clerk for customer and creator authentication.
- Stripe Billing and Stripe Connect for plans, checkout, fees, and payouts.
- pnpm for dependency management and the locked install workflow.

4. Project structure

```
src/  
components/ Shared layouts, cards, states, branding, and uploads  
pages/public/ Marketplace, creators, storefronts, recipes, and products  
pages/dashboard/ Customer and creator workspaces  
pages/admin/ Separate owner interface  
lib/ Small frontend helpers  
convex/  
  schema.ts Data model and indexes  
  auth.config.ts Clerk identity integration  
  http.ts Payment webhook and health route  
  billing*.ts Plans, checkout, portal, Connect, and fulfilment logic  
  admin*.ts Owner authentication and administration  
  recipes.ts Recipe operations  
  products.ts Product operations  
  marketplace.ts Public discovery queries  
  library.ts Purchased access and engagement  
  analytics.ts Creator and marketplace reporting  
  files.ts Upload authorization and metadata  
  seed.ts Idempotent configuration and demonstration data  
  public/ Static assets and shared-hosting fallback rules  
  tests/ Focused contract tests  
  scripts/ Requirements verification  
  docs/ Installation and acceptance guidance
```

5. Data model

Identity and public profiles

- Users link external identity to role, plan, subscription state, onboarding state, and account status.
- Creator profiles hold public identity, specialty, country, verification, feature status, and payout readiness.
- Business profiles hold business-buyer information.

- Storefronts and storefront sections hold creator branding and public page layout.

Content and products

- Recipes hold structured ingredients, instructions, timing, visibility, status, taxonomy, and creator notes.
- Products represent premium recipes, collections, cookbooks, memberships, and commercial licences.
- Product items connect recipes to a sellable product.
- Media and product assets connect authorized uploads to creator content.
- Categories, cuisines, and countries support marketplace discovery.

Commerce and access

- Orders and order items record purchases, prices, platform fees, seller shares, and licence snapshots.
- Entitlements control access to one-time purchases.
- Memberships and membership entitlements control recurring content access.
- Licences preserve the commercial terms accepted at purchase.
- Creator-customer records summarize customer activity without exposing card details.

Payments

- Connected accounts track creator charge and payout readiness.
- Payment-customer links connect users to payment-provider customers.
- SaaS subscriptions track creator plan status and renewal information.
- Processed-event records enforce webhook idempotency.

Engagement and owner operations

- Favourites, follows, reviews, coupons, promotions, notifications, and analytics support marketplace activity.
- Reports, owner notes, and audit logs support moderation.
- Plans, plan features, site settings, and feature flags control the commercial offer.
- Owner credentials and sessions support separate owner access.
- Contact requests store public enquiries.

6. Required buyer-owned services

Create or transfer these services into accounts controlled by the buyer:

- Domain registrar and DNS for RecipesMondial.com.
- Static web hosting with HTTPS and single-page application fallback support.
- Convex production deployment for data and server functions.
- Clerk application for customer and creator authentication.
- Stripe platform account with Billing and Connect access.

Do not continue using seller-owned secrets after handover. Remove previous team access and rotate all credentials.

7. Configuration

Browser configuration

Copy the included environment example to an ignored local environment file. Use public values only.

```
VITE_CONVEX_URL=https://YOUR_DEPLOYMENT.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=YOUR_CLERK_PUBLISHABLE_KEY
VITE_STRIPE_PUBLISHABLE_KEY=YOUR_STRIPE_PUBLISHABLE_KEY
VITE_SITE_URL=https://recipesmondial.com
```

Every VITE_ value can be delivered to the browser. Never place a password, Stripe secret key, webhook secret, private signing value, or reusable token in a VITE_ variable.

Server configuration

Set the following in the buyer-owned Convex deployment environment:

```
CLERK_SIGNING_KEY=YOUR_CLERK_ISSUER_URL
STRIPE_API_KEY=YOUR_STRIPE_SECRET_KEY
STRIPE_WEBHOOK_SECRET=YOUR_STRIPE_WEBHOOK_SECRET
SITE_ID=YOUR_PRODUCTION_DEPLOYMENT_NAME
APP_URL=https://recipesmondial.com
```

Use separate test and live values. Keep server secrets out of source files and browser configuration.

8. Setup and local verification

Requirements:

- Node.js 20 or newer.
- pnpm, using the included lockfile.
- Buyer-owned Convex, Clerk, and Stripe test accounts.

Install dependencies only when they are missing or the lockfile changed:

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

Link the intended Convex development deployment, add buyer-owned test configuration, and use the included commands:

```
pnpm seed
pnpm typecheck
pnpm test
pnpm verify:srs
pnpm build
```

The seed process is idempotent, but it must only run against the intended environment. Seeded creators, recipes, products, orders, subscriptions, and analytics are demonstration records and must never be counted as real business results.

9. Authentication setup

- 1 Create a buyer-owned Clerk application.
- 2 Add the development and production origins and redirect URLs.
- 3 Create the Convex-compatible JWT integration required by the application.
- 4 Set the Clerk issuer value in the Convex environment.
- 5 Add the Clerk publishable key to the browser environment.

- 6 Test sign-up, sign-in, sign-out, onboarding, protected customer pages, creator-only pages, and account deletion.
- 7 Confirm that a standard member cannot access the separate owner area.
- 8 Change the transferred owner credentials and confirm that earlier sessions no longer work.

10. Plans, checkout, and payouts

Creator SaaS plans

- 1 Review Free, Creator, Pro, and Studio plan records.
- 2 Confirm monthly prices, feature limits, product limits, and platform fees.
- 3 Make the public pricing page match the stored plan configuration.
- 4 Test plan checkout, subscription updates, cancellation, and customer-portal access.

Paid plan products and prices can be created when authenticated checkout first runs. Use the intended Stripe mode and review created products before public launch.

Creator payout accounts

- 1 Enable Stripe Connect on the buyer-owned platform account.
- 2 Review supported creator countries and required business information.
- 3 Complete one creator onboarding flow.
- 4 Wait until Stripe reports that charges and payouts are enabled.
- 5 Confirm that product checkout fails safely when a creator is not ready.

Webhook

Register the buyer deployment endpoint through the payment routing setup used for the transferred application:

```
https://YOUR_DEPLOYMENT.convex.site/stripe/webhook
```

Send the event types used by checkout completion, subscription updates, connected-account status, and refunds. Store the signing secret only on the server.

Test duplicate delivery. Sending the same signed event twice must not create two orders, two subscriptions, or duplicate access.

Acceptance tests

Run controlled tests for:

- One-time premium product checkout.
- Recurring creator membership checkout.
- Creator SaaS subscription checkout.
- Coupon discounts.
- Platform fee and creator-share calculation.
- Customer entitlement creation.
- Commercial licence snapshots.
- Refund and access state.

- Membership cancellation and access state.
- Customer billing portal access.
- Creator charge and payout readiness.

11. File uploads and protected content

Upload requests must be authorized on the server. Check the current creator, permitted file type, and file size before accepting upload metadata. Paid content must not become available merely because a visitor knows an identifier or file address.

Verify these boundaries:

- A creator can add allowed files to their own recipes and products.
- A creator cannot change another creator's content or files.
- A visitor without access cannot receive protected recipe details.
- A customer with a valid entitlement can open purchased content.
- Refunded and expired access follows the chosen business policy.

12. Deployment guide

- 1 Create buyer-owned Convex, Clerk, and Stripe accounts.
- 2 Configure public browser values locally.
- 3 Configure secret server values in the intended Convex deployment.
- 4 Deploy the Convex functions to production.
- 5 Seed configuration and clearly labelled demonstration content only when needed.
- 6 Run type checks, focused tests, requirements verification, and one production build.
- 7 Upload the contents of `dist/` to the shared-hosting document root.
- 8 Preserve the included single-page application fallback rule.
- 9 Point RecipesMondial.com to the hosting account and enable HTTPS.
- 10 Register the Stripe webhook endpoint and verify signed delivery.
- 11 Test public pages, sign-in, creator onboarding, owner access, uploads, checkout, access, refunds, cancellations, and payout readiness.

Shared hosting is enough for the static frontend when it supports HTTPS and fallback routing. Vercel is not required. Convex hosts the backend functions and data.

13. Security and credential rotation

- Change owner credentials immediately after transfer.
- Replace seller-owned access for the domain, DNS, hosting, Convex, Clerk, Stripe, and automation services.
- Use separate test and live payment values.
- Keep secrets in provider environment settings, never in browser variables or committed files.
- Remove old collaborators and review account recovery methods.
- Verify webhook signatures before processing events.
- Keep owner sessions short and invalidate them after a credential change.

- Enforce ownership in server functions, not only in the interface.
- Validate file type and size on the server.
- Back up important configuration and transaction records.

14. Functional verification checklist

Public area

- Homepage, discovery, creator directory, storefront, recipe, product, pricing, licensing, help, legal, and contact pages load over HTTPS.
- Search and filters return the expected creators, recipes, and products.
- Unknown browser routes use the application fallback and show the correct not-found experience.

Customer area

- Sign-up, sign-in, onboarding, sign-out, and account deletion work.
- Purchases, memberships, licences, favourites, follows, notifications, billing, and library require authentication.
- Successful payment creates the correct order and access.
- Protected content stays hidden without access.
- Only a verified purchaser can leave a product review.

Creator area

- Onboarding creates the creator profile and storefront.
- Recipe and product create, edit, publish, archive, and visibility controls work.
- Orders, customers, memberships, licences, coupons, promotions, reviews, analytics, billing, and payouts show only the current creator's data.
- Storefront changes appear on the public page.

Owner area

- Owner login works after credential rotation.
- Users, creators, recipes, transactions, subscriptions, reviews, reports, plans, settings, and credential rotation require a valid owner session.
- Verification, featuring, suspension, moderation, and report changes work.
- Demonstration figures remain clearly separate from real activity.

15. Backup, monitoring, and maintenance

- Schedule regular Convex data exports or backups according to the buyer's plan and risk level.
- Keep recovery methods current for the domain, DNS, hosting, Clerk, Convex, and Stripe.
- Monitor frontend availability, the health route, webhook delivery, payment failures, and provider status.
- Review failed webhooks and refunds frequently during launch.
- Review dependency updates and security advisories on a planned schedule.
- Test restoration steps before relying on a backup.
- Keep terms, privacy, refunds, licensing, and support content current.

- Remove or clearly label demonstration data before reporting business figures.

16. Product and support boundaries

The application cannot complete platform approval, creator identity verification, domain transfer, tax registration, legal review, or marketing for the buyer. Payment and payout availability varies by country and provider policy.

Codeblix provides up to 30 days of reasonable technical handover support. This includes transfer questions, source orientation, and explanation of existing workflows. It does not include unlimited development, new features, ongoing hosting management, third-party fees, provider approval, marketing, creator recruitment, or legal services unless separately agreed.