

WowGyms Technical Handover Guide

OPERATING AND HANDOVER REFERENCE

1. Product and system overview

WowGyms is a web marketplace for gym discovery and flexible pass booking. It has four main application areas:

- Public pages for discovery, gym details, content, and contact requests
- Customer pages for account access, bookings, QR passes, travel planning, and concierge history
- Provider pages for storefronts, passes, bookings, payouts, pricing tools, campaigns, and forecasts
- A separate platform administrator area for branding, provider plans, users, messages, and security

The frontend is a React single-page application. Convex provides application data, server functions, storage references, real-time updates, and HTTP endpoints. Clerk provides customer and provider identity. Stripe provides hosted checkout, provider payout onboarding, provider-plan billing, refunds, and webhooks. Gemini provides the AI-assisted itinerary, concierge, campaign, and forecast functions.

The transferred package is a starter asset. It does not by itself prove production traffic, revenue, customers, provider agreements, or completed third-party onboarding.

2. Technology stack

- Node.js 22
- pnpm package manager
- React 19 and React Router
- Vite and TypeScript
- Tailwind CSS
- Convex backend, database, actions, file storage, and HTTP routes
- Clerk authentication
- Stripe Checkout, Connect Express, Billing, and refunds
- Google Gemini for AI-assisted features
- Static frontend hosting with SPA routing support

Use the lockfile supplied with the application package. Do not replace pnpm with another package manager.

3. Main roles and access boundaries

Customer and provider identity

Clerk authenticates customer and provider accounts. On first use, an account selects either customer or provider. The role is stored with the application user and treated as immutable. Protected routes and backend functions check the authenticated identity and expected role.

Platform administrator

The administrator area uses a separate database-backed session rather than customer or provider authentication. Sessions expire after 12 hours and sign-in attempts are rate limited. Initial access details are supplied privately during handover. Change them immediately and do not place them in source files or public documentation.

4. Buyer-safe architecture

```

Browser
-> Static React application
  -> Clerk for user identity
  -> Convex for queries, mutations, actions, and stored data
    -> Stripe for checkout, billing, refunds, and connected payouts
    -> Gemini for AI-assisted responses
    -> Convex HTTP endpoint for signed Stripe webhooks

```

The browser receives only public configuration values. Secret keys remain in the Convex deployment environment. Payment cards and provider bank details are handled on Stripe-hosted pages and should never be stored by the application.

5. Project structure

```

src/
  App.tsx           Route definitions and access gates
  main.tsx          Application providers and startup
  components/       Shared navigation, marketing, auth, and UI components
  pages/            Public, customer, provider, and admin screens
  index.css         Shared visual system

convex/
  schema.ts         Data model and indexes
  users.ts          Identity synchronization and role logic
  providers.ts       Storefront and provider settings
  passes.ts         Pass inventory and price references
  bookings.ts       Booking states and QR validation
  payments.ts       Checkout and refund actions
  connect.ts        Provider payout onboarding
  billing.ts        Provider subscription plan synchronization
  ai.ts             AI-assisted customer and provider features
  http.ts           Public health and webhook endpoints
  admin*.ts         Administrator sessions and operations
  seed.ts           Optional demonstration data

public/assets/      Public brand and demonstration images
.env.example        Public configuration template
package.json        Commands and dependency declarations
pnpm-lock.yaml      Locked dependency versions

```

6. Data model

users

Stores the external identity reference, name, email, selected role, account status, optional payment customer reference, and timestamps.

providers

Stores the owner link, public slug, storefront content, address and coordinates, images, equipment, amenities, publish state, payout onboarding state, plan code, peak-pricing settings, ratings, and neighborhood score.

passes

Stores provider-owned daily or weekly passes, descriptions, integer prices in minor currency units, currency, active state, and payment product and price references.

bookings

Stores customer, provider, and pass links, visit date, booking status, base and charged amounts, currency, checkout and payment references, QR value, and timestamps. Supported states are pending, confirmed, used, cancelled, and failed.

platformSettings and providerPlans

Platform settings store public brand values and the support contact. Provider plans store the subscription name, description, amount, currency, interval, features, payment references, and active state.

AI and operations data

Itinerary history, concierge messages, provider campaign drafts, contact requests, notifications, webhook events, audit logs, administrator credentials, administrator sessions, and rate-limit records support the remaining workflows.

7. Required buyer-owned services

Create or transfer buyer-owned accounts for:

- Convex application backend
- Clerk authentication
- Stripe payments and connected payouts
- Google Gemini API access
- Static website hosting with HTTPS and SPA fallback support
- Domain registrar and DNS management

Do not reuse seller credentials in a long-term deployment. Rotate every transferred secret before commercial launch.

8. Configuration

Copy the public environment template to the local environment file and use buyer-owned values.

Frontend variables:

```
VITE_CONVEX_URL=https://YOUR_CONVEX_DEPLOYMENT.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=YOUR_PUBLIC_CLERK_KEY
VITE_STRIPE_PUBLISHABLE_KEY=YOUR_PUBLIC_STRIPE_KEY
VITE_SITE_URL=https://YOUR_SITE_URL
```

Server-only Convex variables:

```
CLERK_JWT_ISSUER_DOMAIN=https://YOUR_CLERK_ISSUER
STRIPE_SECRET_KEY=YOUR_SERVER_ONLY_STRIPE_SECRET
STRIPE_WEBHOOK_SECRET=YOUR_SERVER_ONLY_WEBHOOK_SECRET
GEMINI_API_KEY=YOUR_SERVER_ONLY_GEMINI_KEY
```

```
GEMINI_MODEL=YOUR_APPROVED_GEMINI_MODEL
SITE_URL=https://YOUR_SITE_URL
SITE_ID=YOUR_UNIQUE_SITE_ID
```

Confirm the final variable names against the supplied environment template and server functions. Never place server-only values in variables beginning with `VITE_`, because those values are included in the browser build.

9. Local setup

Install Node.js 22 and pnpm, then open the application package directory.

```
pnpm install --frozen-lockfile
pnpm exec convex dev
pnpm dev
```

Use separate terminal sessions for the Convex development process and frontend development server. Stop both when testing is complete.

Useful checks:

```
pnpm lint
pnpm exec convex dev --once --typecheck enable
pnpm build
```

Run the build once after configuration and targeted checks pass. The production output is written to `dist/`.

10. Authentication setup

- 1 Create or transfer the Clerk application.
- 2 Add the production domain and allowed redirect URLs.
- 3 Create the Clerk JWT template required by Convex.
- 4 Set the public Clerk key in the frontend environment.
- 5 Set the Clerk issuer domain in the Convex environment.
- 6 Test new registration, sign-in, sign-out, customer role selection, and provider role selection with separate accounts.
- 7 Confirm a provider cannot access customer-only pages and a customer cannot access provider tools.

11. Convex setup

- 1 Create a buyer-owned Convex project and production deployment.
- 2 Link the transferred application package to that project.
- 3 Configure all server-only environment values in the Convex dashboard.
- 4 Deploy schema and functions using the Convex deployment command defined by the existing workflow.
- 5 Confirm the public health endpoint returns a successful response.
- 6 Use the seed command only for staging or a controlled demonstration environment.
- 7 Remove or clearly label all demonstration users, providers, bookings, ratings, campaign data, and charts before commercial launch.

12. Stripe payment setup

Customer checkout

The booking action validates the signed-in customer, active pass, published provider, and future visit date. It creates a pending booking, prepares the price, and redirects to Stripe Checkout. Signed webhook events move the booking to confirmed or failed.

Provider payouts

Providers complete Stripe Connect Express onboarding on a Stripe-hosted page. A storefront cannot be published until Stripe reports that charges are enabled. Eligible checkout payments use destination transfer details for the connected provider.

Provider subscriptions

The administrator can set the provider plan name, monthly amount, currency, and feature list. The plan synchronization action creates or updates the related Stripe product and price. The supplied demonstration plan is free, so the buyer must choose and configure any paid pricing.

Refunds

A confirmed booking can request a Stripe refund until 24 hours before the booked date when a refundable payment reference exists. The application updates the booking to cancelled after a successful refund event.

Webhook sequence

- 1 Deploy the Convex HTTP endpoint.
- 2 Add its Stripe webhook URL in the buyer-owned Stripe account.
- 3 Subscribe to the checkout, payment, and refund events handled by the application.
- 4 Copy the new signing secret into the Convex environment.
- 5 Send Stripe test events and confirm they are recorded once.
- 6 Verify a successful checkout creates one confirmed booking and one customer notification.

13. AI setup

Add a buyer-owned Gemini API key to the Convex environment. The model can be selected with the optional model variable. Test each feature separately:

- Customer itinerary generation based on published directory gyms
- Customer concierge answers based on live listings
- Provider winback campaign drafting
- Provider booking-based forecast guidance

AI output must be treated as assistance, not a guarantee. Add user-facing review language where decisions, pricing, or customer communications could create risk.

The WhatsApp function is a user-initiated share link. A fully inbound WhatsApp or SMS bot is outside the transferred configuration and would require a messaging provider, approved sender, webhook work, and additional credentials.

14. Deployment sequence

Shared hosting is sufficient for the static frontend when it supports HTTPS, uploaded build files, and SPA fallback routing. The Convex backend remains separately hosted by Convex.

- 1 Configure buyer-owned Clerk, Convex, Stripe, Gemini, hosting, registrar, and DNS accounts.
- 2 Add the production domain to authentication and payment settings.
- 3 Deploy the Convex backend and set server-only environment values.
- 4 Run targeted type checks and lint checks.
- 5 Run one production build with `pnpm build`.
- 6 Upload the contents of `dist/` to the shared hosting web root.
- 7 Configure the web server so unknown frontend routes return `index.html`.
- 8 Point DNS to the host and enable HTTPS.
- 9 Add the deployed Stripe webhook URL and signing secret.
- 10 Test public pages, registration, both user roles, provider payout onboarding, storefront publication, pass creation, checkout, webhook confirmation, QR validation, cancellation, admin access, and AI tools.

15. Security and credential rotation

- Change the platform administrator credentials immediately.
- Rotate every transferred API key, webhook secret, deployment token, and hosting password.
- Keep server secrets only in managed environment settings.
- Keep browser variables limited to values designed to be public.
- Require HTTPS for all production pages and callbacks.
- Verify Stripe webhook signatures and preserve idempotent event handling.
- Review administrator audit logs and failed sign-in rate limits.
- Remove demonstration identities and payment references.
- Restrict third-party account access to the minimum required roles.
- Back up data before schema or migration work.

16. Functional verification checklist

Public and customer

- Homepage, search, filters, gym pages, and content pages load correctly
- A new account must choose a role
- Customer checkout rejects past dates and inactive passes
- Successful Stripe test payment confirms one booking
- My Bookings shows the correct pass, amount, date, status, and QR value
- QR validation changes one confirmed booking to used

- Eligible cancellation creates a refund and cancelled state
- Itinerary and concierge responses use published directory data

Provider

- Provider can save a storefront draft
- Photo upload and removal work
- Daily and weekly passes can be created, updated, activated, and deactivated
- Publication is blocked until payout onboarding reports charges enabled
- Provider sees only their own bookings and analytics
- Peak pricing affects only selected days
- Winback drafts and forecast guidance save or display correctly

Administrator

- Administrator session expires correctly
- Repeated failed sign-in attempts are rate limited
- Branding changes appear on public and authentication pages
- Provider plan updates and payment synchronization work
- User suspension and reactivation work
- Contact messages can be reviewed and resolved
- Credential change invalidates the previous administrator session

17. Backup, monitoring, and maintenance

- Schedule regular Convex data exports and document the restore procedure.
- Keep the application package and lockfile under buyer-controlled version management.
- Monitor the public health endpoint, frontend availability, Convex errors, Stripe webhook failures, and authentication errors.
- Review failed bookings and webhook events before changing payment logic.
- Apply dependency updates in a staging environment first.
- Run targeted checks after small changes and a single production build before release.
- Review privacy, terms, refund rules, provider terms, and local payment obligations as the marketplace expands.

18. Product and support boundaries

The transferred application includes the implemented marketplace and operating guides. Buyer-owned service accounts, production data, legal documents, gym agreements, traffic, marketing, and ongoing operations remain the buyer's responsibility.

Codeblix provides 30 days of reasonable technical handover support for setup questions and minor fixes. The support period does not include unlimited development, new integrations, new features, third-party fees, provider recruitment, marketing, or continuing system administration.