HallPros - Technical Documentation

1. Architecture Overview

HallPros is a vertical SaaS marketplace designed for the event venue industry. It utilizes a **Modern React architecture** coupled with a **Backend-as-a-Service (BaaS)** model to deliver real-time data synchronization, secure authentication, and complex payment flows without managing traditional server infrastructure.

Key Workflows

1. **Consumer Flow:** Users search for venues, view availability, and book instantly or request quotes.
2. **Provider Flow (SaaS):** Venue owners subscribe to the platform (via Stripe), manage venues, define availability (blackouts), and receive payouts via Stripe Connect.
3. **Admin Flow:** Platform owners oversee users, manage subscription plans, and configure global whitelabel settings.

The application is built as a **Single Page Application (SPA)** using Vite, relying on **Convex** for database/backend logic and **Clerk** for authentication.

---

2. Tech Stack

Frontend

- **Framework:** React 19
- **Build Tool:** Vite
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Animations:** Framer Motion
- **Icons:** Lucide React
- **Routing:** React Router v7

Backend & Database

Backend & Database

- **Platform:** Convex (Serverless functions, Real-time Database, File Storage)
- **Database Type:** Document-oriented (JSON-like)
- **Scheduling:** Convex Cron Jobs (implied capability)

Infrastructure & Services

- **Authentication:** Clerk (integrated with Convex via JWT)
- **Payments:** Stripe (Checkout, Billing/Subscriptions, Connect Express)
- **Hosting:** Compatible with any static hosting (Shared Hosting, Apache, Nginx)

---

3. Project Structure

```
FLIPPA-HallPros/
├── components/          # Reusable UI components
│   ├── admin/           # Admin-specific layout and components
│   ├── dashboard/       # Provider Dashboard layout and sidebar
│   └── ui/              # Generic UI elements (Buttons, Inputs, etc.)
├── context/             # React Context for global state (HallProsContext)
├── convex/              # Backend Logic
│   ├── _generated/      # Auto-generated Convex types
│   ├── schema.ts        # Database schema definition
│   ├── auth.config.ts   # Clerk integration config
│   ├── http.ts          # Webhook endpoint definitions (Stripe)
│   └── *.ts             # Backend functions (bookings, venues, users, etc.)
├── pages/               # Application Routes/Views
│   ├── admin/           # Admin pages (Overview, Plans, Users)
│   ├── dashboard/       # Provider pages (Venues, Payouts, Calendar)
│   └── *.tsx            # Public pages (Home, Marketplace, Auth)
├── public/              # Static assets and .htaccess for routing
├── scripts/             # Utility scripts (Stripe, Webhook router)
├── App.tsx              # Main Router configuration and Auth protection
└── package.json         # Dependencies and scripts
```

---

4. Database Schema

The database is managed by Convex. Below is the schema structure defined in `convex/schema.ts`.

Table Name	Description	Key Relationships
`users`	Stores user profiles for all roles.	`clerkId` (Auth), `stripeAccountId` (Connect)
`venues`	Venue details, capacity, and pricing.	`providerId` (Links to User)
`bookings`	Transaction records for venue reservations.	`venueId`, `providerId`, `consumerId`
`platformConfig`	Singleton configuration for whitelabel settings.	None
`blackouts`	Date-specific unavailability rules.	`venueId`, `providerId`
`externalCalendars`	Integration tokens for Google/Outlook sync.	`providerId`
`testimonials`	Marketing testimonials for the landing page.	None
`contactMessages`	Inbound contact form submissions.	None

5. Setup & Installation

Prerequisites

1. **Node.js** (v18 or higher)
2. **Convex Account** (convex.dev)
3. **Clerk Account** (clerk.com)
4. **Stripe Account** (stripe.com)

Local Development Steps

1. **Clone the repository:**

```
git clone <repository-url>
cd FLIPPA-HallPros
```

2. **Install dependencies:**

```
npm install
```

3. **Environment Setup:** Create a `.env.local` file in the root directory:

```
VITE_CLERK_PUBLISHABLE_KEY=pk_test_...  # From Clerk Dashboard
# VITE_CONVEX_URL is added automatically by the CLI in the next step
```

4. **Initialize Convex:**

   ```
   npx convex dev
   ```

   Log in when prompted. This will create your Convex deployment.

5. **Configure Clerk & Stripe in Convex:** In your Convex Dashboard (Settings -> Environment Variables), add:
   - `CLERK_SIGNING_KEY`: (From Clerk JWT Templates)
   - `STRIPE_SECRET_KEY`: `sk_test_...`
   - `STRIPE_WEBHOOK_SECRET`: `whsec_...`
   - `HOST_URL`: `http://localhost:5173` (for local)

6. **Seed Database:** Populate the database with initial venues, users, and configuration:

   ```
   npx convex run seed:seedData
   ```

7. **Run the Application:**

   ```
   npm run dev
   ```

Admin Credentials

To access the admin panel at `/admin/login`, use the following credentials:

- **User:** codeblix
- **Password:** codeblix

---

6. Deployment Guide

This application is designed to run efficiently on standard **Shared Hosting** (cPanel/Apache). Expensive cloud platforms like Vercel or Netlify are **not required**.

1. Deploying the Backend (Convex)

1. Run the deployment command to push your backend functions to production:

   ```
   npx convex deploy
   ```

2. Update your **Production Environment Variables** in the Convex Dashboard:
   - Set `HOST_URL` to your actual domain (e.g., `https://hallpros.com`).
   - Ensure Stripe keys are set to **Live** keys.

2. Building the Frontend

1. Open `.env.local` or your environment configuration.
2. Ensure `VITE_CONVEX_URL` points to your **Production** Convex URL.
3. Run the build command:

   ```
   npm run build
   ```

   This creates a `dist` folder containing optimized HTML, CSS, and JS files.

3. Hosting on Shared Hosting (cPanel)

1. Access your hosting File Manager or FTP.
2. Upload the **contents** of the `dist` folder to your `public_html` directory (or subdomain folder).
3. **Important:** Ensure the `.htaccess` file (located in `public/`) is uploaded. This handles routing for the Single Page Application, directing all requests to `index.html`.

`.htaccess` content:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteBase /
  RewriteRule ^index\.html$ - [L]
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteRule . /index.html [L]
</IfModule>
```

4. Stripe Webhook Configuration

1. Go to the Stripe Dashboard -> Developers -> Webhooks.
2. Add an endpoint pointing to your Convex HTTP action:

```
https://<YOUR-CONVEX-PROD-URL>.convex.site/stripe/webhook
```

3. Select events: `checkout.session.completed`, `payment_intent.succeeded`, `account.updated`, `customer.subscription.deleted`, `invoice.paid`.

The application is now live and fully functional on shared hosting.