Technical Documentation: 1StopSalon (Echo)

1. Architecture Overview

1StopSalon is a full-stack SaaS marketplace application designed for beauty and wellness professionals. It utilizes a **Backend-as-a-Service (BaaS)** architecture, leveraging **Convex** for the database, backend logic, and file storage, and **Clerk** for authentication.

The application follows a reactive, real-time data flow:

1. **Frontend:** A Single Page Application (SPA) built with React and Vite. It connects directly to the Convex cloud backend via WebSockets/HTTP.
2. **Backend:** Serverless functions (Queries, Mutations, Actions) run on the Convex platform. There is no traditional Node.js server to manage.

3. **Authentication:** Clerk handles identity management. Tokens are passed to Convex to secure database access via `auth.config.ts`.
4. **Payments:** Stripe Connect (Express) is used for marketplace payouts. The platform acts as the orchestrator, taking application fees (optional) and routing funds to Providers.

2. Tech Stack

Frontend Core

- **Framework:** React 18
- **Build Tool:** Vite
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Animations:** Framer Motion
- **Icons:** Lucide React
- **Routing:** React Router DOM 6

Backend & Infrastructure

- **Database & Backend:** Convex (Real-time document database)
- **File Storage:** Convex File Storage
- **Authentication:** Clerk (integrated with Convex)
- **Payments:** Stripe (React Stripe.js + Node SDK)
- **Seed Data:** Faker.js

3. Project Structure

```
FLIPPA-1StopSalon
├── convex/                # Backend Logic (Serverless)
│   ├── _generated/        # Auto-generated Convex types
│   ├── auth.config.ts     # Clerk/Convex integration config
│   ├── schema.ts          # Database Schema definition
│   ├── users.ts           # User management functions
│   ├── providers.ts       # Business logic for Storefronts
│   ├── stripe.ts          # Stripe Connect & Payment Actions
│   ├── http.ts            # Webhook routing (Stripe)
│   └── seed.ts            # Database seeding logic
├── public/                # Static assets & .htaccess
├── src/                   # Frontend Application
│   ├── components/        # Reusable UI components
```

```
│   ├── context/            # Global State (AuthContext)
│   ├── lib/                # Utilities & Constants
│   ├── pages/              # Application Routes
│   │   ├── admin/          # Admin Dashboard
│   │   ├── auth/           # Login/Signup/Onboarding
│   │   ├── dashboard/      # Provider Dashboard (CMS)
│   │   ├── storefront/     # Public facing pages
│   │   └── consumer/       # Customer Booking history
│   └── layout/             # Dashboard wrappers
└── index.html              # Entry point
```

4. Database Schema

The database is defined in `convex/schema.ts`. It is a document-oriented database with relational links via IDs.

Table Name	Description	Key Relationships
`users`	`tokenIdentifier`	`providers`
`providers`		`userId`
`services`		`providerId`
`products`		`providerId`
`bookings`		`providerId` `consumerId` `serviceId`
`orders`		`providerId` `consumerId`
`platformConfig`		
`testimonials`		

5. Setup & Installation

Prerequisites

- Node.js (v18+)
- npm

1. Installation

```
# Install dependencies
npm install
```

2. Environment Variables

You need to configure variables in two places: local `.env` file and the Convex Dashboard.

A. Local `.env.local` Create this file in the root directory:

```
# Clerk Public Key (From Clerk Dashboard)
VITE_CLERK_PUBLISHABLE_KEY=pk_test_...

# Stripe Public Key (From Stripe Dashboard)
VITE_STRIPE_PUBLISHABLE_KEY=pk_test_...

# Convex URL (Auto-added by npx convex dev, but good to know)
VITE_CONVEX_URL=https://...
```

B. Convex Dashboard Variables Run `npx convex dev` to open the dashboard, then go to **Settings > Environment Variables**:

- `CLERK_SIGNING_KEY`: The Issuer URL from Clerk (JWT Templates > Convex).
- `STRIPE_SECRET_KEY`: Your Stripe Secret Key (`sk_test_...`).
- `HOST_URL`: The URL of your app (e.g., `http://localhost:5173` or your production domain).
- `STRIPE_WEBHOOK_SECRET` (Optional but recommended): For verifying Stripe webhooks.

3. Initialize Backend

```
npx convex dev
```

This command syncs your schema and functions to the cloud.

4. Seed Data

To populate the app with realistic demo data (Salons, Services, Products):

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

5. Run Frontend

```
npm run dev
```

6. Admin Panel Access

The application includes a hidden Admin Dashboard for platform configuration (Name, Logo, Subscription Pricing).

- **URL:** `/admin`
- **Username:** codeblix
- **Password:** codeblix

7. Stripe Configuration

This project uses **Stripe Connect (Express Accounts)**.

1. **Enable Connect:** In Stripe Dashboard, go to Connect > Settings and ensure "Express" accounts are enabled.
2. **Redirect URI:** Add your domain to Stripe Redirect URIs:
   - Local: `http://localhost:5173/stripe-onboarding`
   - Prod: `https://yourdomain.com/stripe-onboarding`
3. **Webhook:** Point Stripe Webhooks to your Convex HTTP Action:
   - URL: `https://<your-convex-deployment>.convex.site/stripe_webhook`
   - Events: `payment_intent.succeeded`.

8. Deployment Guide

This application is designed to be deployed easily on standard shared hosting. You do **not** need Vercel or a Node.js server for the frontend, as the build output is static HTML/JS/CSS.

Build Process

1. Run the build command:

   ```
   npm run build
   ```

2. This creates a `dist/` directory containing the compiled assets.

2. This creates a `dist/` directory containing the compiled assets.