FoodAsFun Technical Documentation

1. Architecture Overview

FoodAsFun is a Vertical SaaS platform designed for culinary professionals to create digital storefronts, manage experiences, and accept bookings. The application is built on a **Serverless Architecture**, leveraging **React** for the frontend and **Convex** as a real-time Backend-as-a-Service (BaaS).

High-Level Data Flow

1. **Client (React):** Handles UI rendering, routing, and user interaction. It communicates with the backend via strongly-typed Remote Procedure Calls (RPCs).
2. **Authentication (Clerk):** Manages user identity. Tokens are passed to Convex to secure backend functions.
3. **Backend (Convex):** Hosts the database, file storage, and server-side functions (Queries, Mutations, Actions). It handles business logic, data validation, and real-time subscriptions.
4. **Payments (Stripe):**

- **Stripe Billing:** Handles subscription payments for Providers (SaaS revenue).
- **Stripe Connect (Express):** Handles marketplace transactions (Consumers booking Experiences), routing funds directly to Providers while managing platform fees.

---

2. Tech Stack

Frontend

- **Framework:** React 18
- **Build Tool:** Vite
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Animations:** Framer Motion
- **Routing:** React Router DOM (v6/v7)
- **Icons:** Lucide React

Backend & Database

- **Platform:** Convex (Real-time database, Schema validation, Serverless functions)
- **Language:** TypeScript

Integrations

- **Authentication:** Clerk (`@clerk/clerk-react`)
- **Payments:** Stripe (`stripe`, `@stripe/react-stripe-js`)
- **Image Sources:** Pexels API (used in seeding script)

---

3. Project Structure

The codebase follows a standard Vite + Convex structure:

```
├── convex/                 # Backend Logic & Database
│   ├── _generated/         # Convex auto-generated types
│   ├── auth.config.ts      # Clerk integration config
│   ├── schema.ts           # Database Schema definition
```

```
│   ├── seed.ts              # Data seeding script
│   ├── stripe.ts            # Stripe payment logic (Actions)
│   └── *.ts                 # Domain-specific logic (users, bookings, etc.)
├── public/                  # Static assets & server config (.htaccess)
├── src/                     # Frontend Application
│   ├── components/          # Reusable UI components
│   │   └── ui/              # Atom components (Buttons, Inputs)
│   ├── layouts/             # Page layouts (Auth, Dashboard, Main)
│   ├── lib/                 # Utilities and helpers
│   ├── pages/               # Route components
│   │   ├── admin/           # Admin-specific views
│   │   ├── consumer/        # Consumer dashboard
│   │   ├── dashboard/       # Provider dashboard
│   │   ├── onboarding/      # Stripe/Role setup flow
│   │   └── public/          # Public-facing storefronts
│   ├── App.tsx              # Main Router configuration
│   └── main.tsx             # Entry point
└── vite.config.ts           # Vite configuration
```

---

4. Database Schema

The database is a document-oriented relational store hosted by Convex. The schema is defined in `convex/schema.ts`.

Core Tables

Table	Description	Key Relationships
users		`clerkId` `stripeCustomerId`
providers		`userId` `stripeAccountId`
experiences		`providerId`
bookings		`experienceId` `providerId` `consumerId`
settings		
testimonials		

Indexes

- **Users:** Indexed by `clerkId` for fast auth lookup

- **Users:** Indexed by `clerkId` for fast auth lookup.
- **Providers:** Indexed by `userId` (dashboard lookup) and `slug` (public storefront URL).
- **Experiences:** Indexed by `providerId`.

---

5. Setup & Installation

Prerequisites

- Node.js (v18+)
- Stripe Account (Test mode)
- Clerk Account (Development mode)

1. Environment Configuration

Create a `.env.local` file in the root directory:

```
# Frontend Keys
VITE_CONVEX_URL=https://your-convex-url.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=pk_test_...
VITE_STRIPE_PUBLISHABLE_KEY=pk_test_...

# Backend Keys (Set these in Convex Dashboard or via CLI)
# CLERK_ISSUER_URL=https://your-clerk-issuer.clerk.accounts.dev
# STRIPE_SECRET_KEY=sk_test_...
```

2. Install Dependencies

```
npm install
```

3. Initialize Convex

This will set up your backend project and prompt you to log in.

```
npx convex dev
```

4. Configure Stripe

1. Enable **Stripe Connect** in your Stripe Dashboard.
2. Select **Express** account type.

3. Add `http://localhost:5173/onboarding/connect` (or your dev port) to Stripe Connect Redirects.

5. Seed the Database

Populate the app with realistic data (Providers, Experiences, Images):

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

6. Run the Application

```
npm run dev
```

Admin Access

To access the platform admin panel at `/admin`:

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

---

6. Deployment Guide

This application is designed to be deployment-agnostic. While specialized frontend hosts are common, they are not required.

Backend Deployment

Deploy your Convex functions and schema to production:

```
npx convex deploy
```

Ensure you set your Production Environment Variables (`STRIPE_SECRET_KEY`, `CLERK_ISSUER_URL`) in the Convex Dashboard.

Frontend Deployment (Shared Hosting)

You do **not** need Vercel or Netlify. This application can be hosted on any standard Shared Hosting (cPanel, Apache, Nginx).

1. **Build the Project:**

```
npm run build
```