Technical Documentation - Nail Salon Marketplace Platform

1. Architecture Overview

This application is a modern, full-stack marketplace designed for Nail Salons and Beauty Service providers. It utilizes a **Serverless Architecture** to ensure high scalability and low maintenance costs.

- **Frontend (Client):** A Single Page Application (SPA) built with React and TypeScript. It interfaces directly with the backend via API hooks.
- **Backend (Serverless):** Powered by **Convex**, a Backend-as-a-Service (BaaS). Convex handles the database, server-side functions, real-time subscriptions, and file storage.
- **Authentication:** Managed via an external identity provider integrated with Convex (referenced in `convex/auth.config.ts`).
- **Payments:** Integrated with **Stripe** for handling booking transactions and payouts.

2. Tech Stack

Core Frameworks

- **Language:** TypeScript (Strict typing for both frontend and backend).
- **Frontend Library:** React (Functional components, Hooks).
- **Build Tool:** Vite (Fast HMR and optimized bundling).
- **State Management:** React Context API (`AuthContext`, `PlatformContext`).

Backend & Database

- **Platform:** Convex (Real-time database and backend functions).
- **Database Type:** Document-oriented (JSON-like documents).
- **File Storage:** Convex File Storage (for salon images and user avatars).

Integrations

- **Payments:** Stripe API (Custom Checkout Forms).

- **Icons:** Lucide React / React Icons (Inferred).

3. Project Structure

The codebase is organized to separate UI concerns from business logic and database interactions.

```
FLIPPA-NailSalon/
├── components/          # Reusable UI components (Hero, Navbar, Checkout)
├── context/             # Global State (Auth, Provider data)
├── convex/              # BACKEND: Database schema and API functions
│   ├── _generated/      # Auto-generated Convex types
│   ├── bookings.ts      # Booking logic
│   ├── payments.ts      # Stripe integration logic
│   ├── schema.ts        # Database definitions
│   └── users.ts         # User management
├── layouts/             # Page wrappers (Admin, Dashboard, Main)
├── pages/               # Application Routes/Views
│   ├── admin/           # Super Admin control panels
│   ├── provider/        # Salon Owner dashboards
│   └── ...              # Public pages (Home, Explore, Auth)
├── public/              # Static assets and .htaccess for deployment
├── App.tsx              # Main entry point and routing setup
└── package.json         # Dependencies and scripts
```

4. Database Schema

The database is defined in `convex/schema.ts`. Based on the backend file structure, the application utilizes the following tables (collections):

- **Users (`users`):** Stores customer and provider profiles, authentication IDs, and roles (admin/provider/customer).
- **Salons (`salons`):** Stores business details, descriptions, location data, and owner references.
- **Services:** (Likely embedded in Salons or separate collection linked by ID) Stores menu items, pricing, and duration.
- **Bookings (`bookings`):** Links a User, a Salon, and a Service. Tracks status (pending, confirmed, completed, cancelled) and timestamps.
- **Reviews (`reviews`):** Stores ratings and text feedback linked to specific bookings or salons.
- **Payments (`payments`):** Stores Stripe transaction IDs, payment status, and amounts.

- **Files (`files`):** Metadata for images stored in Convex storage.

5. Setup & Installation

Prerequisites

- Node.js (v18 or higher)
- npm or yarn
- A Convex account (for backend)
- A Stripe account (for payments)

Step 1: Clone and Install

```
git clone <repository_url>
cd FLIPPA-NailSalon
npm install
```

Step 2: Environment Configuration

Create a `.env.local` file in the root directory. You will need keys from your Convex dashboard and Stripe dashboard.

```
# Example .env.local structure
VITE_CONVEX_URL=https://your-convex-instance.convex.cloud
VITE_PUBLISHABLE_KEY=pk_test_... # Stripe Public Key
```

Step 3: Backend Initialization

Initialize the Convex backend. This will sync the `convex/` folder with your cloud instance.

```
npx convex dev
```

Login to your Convex account when prompted.

Step 4: Seed Database (Optional)

If a seed script is provided to populate initial data:

```
npx convex run seed
```

Step 5: Start Development Server

In a new terminal window (keep `npx convex dev` running):

```
npm run dev
```

The application will launch at `http://localhost:5173`.

6. Admin Panel Access

The application includes a hardcoded Super Admin entry point for platform management.

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

7. Deployment Guide

Since this is a client-side rendered (CSR) React application, it does **not** require specialized Node.js hosting (like Vercel or AWS Lambda) for the frontend. The backend runs entirely on Convex's cloud.

Shared Hosting (Apache/Nginx/cPanel) is sufficient.

Build Process

1. Run the build command to generate static assets:

   ```
   npm run build
   ```

2. This creates a `dist/` folder containing `index.html`, JavaScript, and CSS files.

Deploying to Shared Hosting

1. Access your hosting File Manager or FTP.
2. Upload the **contents** of the `dist/` folder to your `public_html` directory.
3. **Important:** Ensure the `.htaccess` file (located in `public/.htaccess`) is uploaded. This file handles URL rewriting to support React Router on page refresh.

Content of `.htaccess` included in build:

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

Backend Deployment

The backend is deployed separately to Convex cloud.

```
npx convex deploy
```

This pushes your schema and functions to the production environment of your Convex project. Update your environment variables in your production build to point to the production Convex URL.