Expert-Influencer Marketplace - Technical Documentation

Architecture Overview

Expert-Influencer is a modern, full-stack marketplace application designed to connect brands with influencers. The application utilizes a **Serverless Architecture** to ensure scalability and low maintenance.

Core Data Flow:

1. **Frontend:** A Single Page Application (SPA) built with React and Vite. It handles all user interactions and UI rendering.
2. **Authentication:** Managed by **Clerk** for user management (Sign up, Sign in) and synchronized with the database.
3. **Backend & Database (Primary): Convex** serves as the reactive backend-as-a-service. It hosts the database, server-side functions (queries/mutations), and handles real-time data synchronization with the frontend.
4. **Blog Module:** A specialized module powered by **Supabase** (PostgreSQL) for SEO-optimized content management.
5. **External Integrations:**
    - **Stripe:** Handles payments and payouts.
    - **Google Gemini AI:** Powers the "AI Finder" to match brands with influencers based on natural language descriptions.

- **Apify:** Scrapes external TikTok/Instagram data for influencer discovery.

Tech Stack

Frontend

- **Framework:** React 19
- **Build Tool:** Vite 6
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Routing:** React Router DOM 7
- **State Management:** React Context API + Convex React Hooks

Backend & Services

- **Primary Backend:** Convex (BaaS) - Handles data, file storage, and API functions.
- **Secondary Backend (Blog):** Supabase (PostgreSQL) - Specifically for the SSG Blog package.
- **Authentication:** Clerk
- **AI Engine:** Google Gemini (via `@google/genai`)
- **Scraping:** Apify
- **Payments:** Stripe

Project Structure

```
InfuencerMarketplace
├── components/          # Reusable UI components (Buttons, Cards, Modals)
├── constants/           # Configuration constants and static data
├── contexts/            # Global state (AuthContext)
├── convex/              # BACKEND LOGIC (Convex)
│   ├── _generated/      # Auto-generated types by Convex
│   ├── ai.ts            # AI matching logic
│   ├── auth.config.ts   # Clerk integration config
│   ├── campaigns.ts     # Campaign CRUD & hiring logic
│   ├── schema.ts        # Database schema definition
│   ├── scrapers.ts      # Apify integration
│   ├── users.ts         # User management & syncing
```

```
│   ├── users.ts           # User management & syncing
│   └── ...
├── docs/                  # Supplementary documentation
├── packages/              # Sub-modules
│   └── react-supabase-ssg-blog # Independent blog module
├── pages/                 # Route components (Dashboard, Home, Login)
├── public/                # Static assets
└── services/              # Frontend service adapters (AI, Mock APIs)
```

Database Schema

The primary application data is stored in **Convex**. Below is the schema definition derived from `convex/schema.ts`.

1. Users (`users`)

Stores user profiles for both Brands and Influencers.

- `tokenIdentifier`: String (Clerk ID, Indexed)
- `role`: String ('Brand' | 'Influencer' | 'Admin')
- `name`, `email`, `avatar`, `bio`: String
- `niche`: String (Influencer only)
- `socials`: JSON Object (e.g., `{ tiktok: "url" }`)
- `stripe_account_id`: String (For payouts)
- `stats`: Object (Followers, views, engagement rate)

2. Campaigns (`campaigns`)

Marketing campaigns created by brands.

- `brand_id`: ID (Reference to `users`)
- `title`, `description`, `image`: String
- `budget`: Number
- `requirements`: Array
- `deliverables`: Array
- `status`: String ('Active' | 'In Progress' | 'Completed')

3. Applications (`applications`)

Influencer applications to campaigns.

Influencer applications to campaigns.

- `campaign_id`: ID (Reference to `campaigns`)
- `influencer_id`: ID (Reference to `users`)
- `proposal`: String
- `status`: String ('Pending' | 'Accepted' | 'Rejected')

4. Financial Records

- `billing_records`: Records of payments made by brands.
- `payout_records`: Records of earnings pending/paid to influencers (90% of budget).

5. Scraped Data (`scraped_influencers`)

External influencer data fetched via Apify for discovery purposes.

Setup & Installation

Prerequisites