TeethJob - Technical Documentation

Architecture Overview

TeethJob is a vertical SaaS marketplace designed for the dental industry, connecting practices with professionals. The application utilizes a modern **Serverless Architecture**.

- **Frontend**: A Single Page Application (SPA) built with React and Vite. It handles all UI rendering and client-side routing.
- **Backend & Database**: Powered by **Convex**, a backend-as-a-service platform. Convex handles the real-time database, server-side functions (Queries/Mutations), and file storage.
- **Authentication**: Managed by **Clerk**, integrated directly with Convex for secure data access policies.
- **Payments**: Integrated with **Stripe** for subscription management and Stripe Connect for provider onboarding.

There is no traditional Node.js/Express server to maintain. The backend logic resides entirely within the `convex/` directory and runs on the Convex cloud infrastructure.

Tech Stack

Core Framework

- **Runtime**: Node.js
- **Build Tool**: Vite (`^6.2.0`)
- **Frontend Library**: React 19 (`^19.2.3`)
- **Language**: TypeScript

Backend Services

- **BaaS**: Convex (`^1.31.6`) - Handles Database, Cron Jobs, File Storage, and API endpoints.

- **Auth**: Clerk (`@clerk/clerk-react`)
- **Payments**: Stripe SDK (`^20.2.0`)

UI & Styling

- **Styling**: Tailwind CSS (via CDN script in `index.html`)
- **Animations**: Framer Motion (`^12.29.0`)
- **Icons**: Lucide React (`^0.562.0`)
- **Routing**: React Router DOM (`^7.12.0`)

Project Structure

```
FLIPPA-TeethJob
├── components/          # Reusable UI components
│   ├── layout/          # Structural components (Navbar, Sidebar, Footer)
│   ├── shared/          # Logic wrappers (ErrorBoundary, ScrollToTop)
│   └── ui/              # Atomic elements (Buttons, Inputs, Cards, Badges)
├── context/             # React Context (MarketplaceContext) for global state
├── convex/              # Backend Logic (Serverless Functions)
│   ├── _generated/      # Convex auto-generated types (do not edit)
│   ├── schema.ts        # Database schema definition
│   ├── auth.config.ts   # Clerk integration config
│   ├── seed.ts          # Database seeding script
│   └── *.ts             # API modules (jobs, users, practices, etc.)
├── pages/               # Application Views/Routes
│   ├── admin/           # Admin Panel
│   ├── auth/            # Login/Signup pages
│   ├── dashboard/       # Protected dashboards for Practices & Professionals
│   ├── inner/           # Static pages (About, Privacy, Terms)
│   └── public/          # Publicly accessible pages (Job Board, Storefront)
├── dist/                # Production build output (generated after build)
├── App.tsx              # Main routing and provider setup
└── vite.config.ts       # Vite configuration
```

Database Schema

The database is schema-enforced via Convex (`convex/schema.ts`).

Table Name	Description	Key Relationships

users	`tokenIdentifier` `clerkId`
practices	`users`
professionals	`users`
jobs	`practices`
applications	`jobs` `practices` `professionals`
interviewRequests	`practices` `professionals`
notifications	`users`
platformSettings	
testimonials	

Setup & Installation

1. Prerequisites

- Node.js (v18 or higher)
- NPM

2. Installation

Extract the project files and install dependencies:

```
npm install
```

3. Environment Variables

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

```
VITE_CLERK_PUBLISHABLE_KEY=pk_test_c3RyaWtpbmctc2FpbGZpc2gtMjYuY2xlcmsuYWNjb3VudHM
VITE_CONVEX_URL=https://basic-jellyfish-174.convex.cloud
```

Note: You may need to replace these with your own keys if you initialize a new

Convex project.

4. Convex Initialization

Initialize the backend. This will prompt you to log in to Convex and create a project.

```
npx convex dev
```

5. Configure Convex Variables

In your Convex Dashboard (Settings ->Environment Variables), set the following:

- `CLERK_SIGNING_KEY`: `https://[your-clerk-domain].clerk.accounts.dev` (Found in Clerk Dashboard ->JWT Templates).
- `STRIPE_API_KEY`: Secret key from Stripe.
- `HOST_URL`: `http://localhost:5173` (for local) or your production domain.

6. Seed Database

Populate the database with dummy data (practices, professionals, jobs):

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

7. Run Locally

```
npm run dev
```

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

Admin Credentials

To access the global settings panel at `/admin/settings`:

- **Username:** codeblix
- **Password:** codeblix

Deployment Guide

Deployment Guide

This application is optimized for standard Shared Hosting (Apache/cPanel). A Vercel/Netlify account is **not** required, though possible.

1. Build the Application

Run the build script to generate static files:

```
npm run build
```

This creates a `dist` folder containing HTML, CSS, and JavaScript.

2. Prepare for Shared Hosting

The application uses client-side routing. On shared hosting, you must redirect all requests to `index.html` so React Router can handle the paths.

1. Locate the file named `htaccess.txt` in the root directory.
2. Copy this file into the `dist` folder.
3. Rename it from `htaccess.txt` to `.htaccess` (note the dot at the beginning).

Content of `.htaccess`:

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

3. Deploy Backend

Ensure your Convex backend is deployed to production:

```
npx convex deploy
```

Note: In the Convex dashboard, update the `HOST_URL` environment variable to your actual domain name (e.g., `https://your-dental-site.com`).

4. Upload Files

1. Access your hosting File Manager or FTP.
2. Upload the **contents** of the `dist` folder to your public directory (usually

2. Upload the **contents** of the `dist` folder to your public directory (usually `public_html`).

5. Post-Deployment (Stripe)

1. Go to your Stripe Dashboard.
2. Add a Webhook endpoint pointing to: `https://[your-convex-deployment-url].convex.site/stripe-webhook`.
3. Select events to listen for: `checkout.session.completed`.