SimplyPup - Technical Documentation

1. Architecture Overview

SimplyPup is a modern, full-stack Single Page Application (SPA) designed as a vertical marketplace for ethical dog breeders. The architecture leverages a **serverless** approach using **Convex** as the backend-as-a-service (BaaS), which handles the database, file storage, and API functions..

- **Frontend:** Built with React and Vite, utilizing Tailwind CSS for styling and Framer Motion for animations. It communicates with the backend via the Convex React client.
- **Backend & Database:** Convex provides a realtime, reactive database. All business logic (queries and mutations) resides in the `convex/` directory.
- **Authentication:** Integrated with **Clerk** for secure user management. Clerk handles identity, while Convex handles authorization via JWT verification.
- **Payments: Stripe Connect** is used for breeder payouts (onboarding breeders as merchants) and **Stripe Checkout** is used for consumer deposits and breeder subscriptions.

2. Tech Stack

Core Frameworks

- **Runtime/Build Tool:** Node.js, Vite
- **Frontend Library:** React v19
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Routing:** React Router DOM v7

Backend & Services

- **Database / API:** Convex (Realtime Document Database)
- **Authentication:** Clerk (@clerk/clerk-react)
- **File Storage:** Convex File Storage

- **Payments:** Stripe API (Node.js SDK)
- **Data Seeding:** Faker.js (for generating mock data)

3. Project Structure

```
FLIPPA-SimplyPup
├── convex/                # Backend logic (Serverless functions)
│   ├── _generated/        # Auto-generated Convex types
│   ├── auth.config.ts     # Clerk JWT configuration
│   ├── schema.ts          # Database schema definition
│   ├── seed.ts            # Data seeding script
│   ├── stripe.ts          # Stripe integration logic
│   └── [module].ts        # API endpoints (users, breeders, litters, etc.)
├── public/                # Static assets & .htaccess
├── src/                   # Frontend Application
│   ├── components/        # Reusable UI components
│   ├── layouts/           # Dashboard and Page layouts
│   ├── lib/               # Custom hooks (useSimplyPup) and utilities
│   ├── pages/             # Route views (Home, Auth, Dashboard, etc.)
│   ├── AppConvex.tsx      # Main entry point with Providers
│   └── index.tsx          # React DOM mount point
├── .env.local             # Local environment variables
└── vite.config.ts         # Vite configuration
```

4. Database Schema

The database is defined in `convex/schema.ts`. Below are the primary tables and their relationships:

Table	Description	Key Fields
users		`clerkId` `email` `role`
breeders		`userId` `kennelName` `stripeAccountId` `isPublished` `verified`
consumers		`userId` `fullName` `contactInfo`
litters		`breederId` `sire` `dam` `price` `status` `puppiesCount`
applications		`litterId` `consumerId` `breederId` `status` `stripePaymentIntentId`
documents		

documents	`breederId` `fileStorageId` `status`
reviews	`breederId` `rating` `comment`
notifications	`userId` `type` `message` `isRead`

5. Setup & Installation

Prerequisites

- Node.js (v18+)
- NPM or Yarn
- A Convex account (convex.dev)
- A Clerk account (clerk.com)
- A Stripe account (stripe.com)

Step 1: Install Dependencies

```
npm install
```

Step 2: Environment Configuration

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

```
# Convex Settings (Found in Convex Dashboard)
VITE_CONVEX_URL=https://your-convex-url.convex.cloud

# Clerk Settings (Found in Clerk Dashboard)
VITE_CLERK_PUBLISHABLE_KEY=pk_test_your_clerk_key
```

Step 3: Configure Backend Services

1. **Initialize Convex:**

   ```
   npx convex dev
   ```

   This will prompt you to log in and select a project.

2. **Configure Clerk Issuer:**

   - Go to your **Convex Dashboard** > Settings > Environment Variables.

- Add `CLERK_SIGNING_KEY`.
- Value should be your Clerk Issuer URL (e.g., `https://your-app.clerk.accounts.dev`).

3. **Configure Stripe (Optional for local dev, required for payments):**
   - In Convex Dashboard > Environment Variables, add:
     - `STRIPE_SECRET_KEY` (starts with `sk_test_...`)
     - `STRIPE_WEBHOOK_SECRET` (starts with `whsec_...`)

Step 4: Seed Database

Populate the database with realistic test data (Breeders, Litters, Reviews):

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

Step 5: Run Locally

```
npm run dev
```

Access the app at `http://localhost:5173`.

Admin Access

To access the Admin Dashboard logic or specific admin routes:

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

6. Deployment Guide

This application is a static site (SPA) that communicates with a managed backend (Convex). You do **not** need Vercel or Node.js server hosting for the frontend.

Frontend Deployment

You can deploy the frontend to any **Shared Hosting** (cPanel, Hostinger, GoDaddy, Apache/Nginx).

1. **Build the Project:**

```
npm run build
```

This creates a `dist` folder containing the compiled HTML, CSS, and JS.

2. **Prepare for Shared Hosting (Apache):** The project includes a `.htaccess` file in the `public/` folder. Ensure this file is present in your build output. This file handles the URL rewriting required for React Router to work on Apache servers (redirecting all requests to `index.html`).

3. **Upload:** Upload the contents of the `dist` folder to your hosting's `public_html` directory via FTP or File Manager.

Backend Deployment

The backend is managed by Convex.

1. **Deploy Functions:**

```
npx convex deploy
```

This pushes your schema and functions to the Convex production environment.

2. **Production Environment Variables:** Ensure you add all environment variables (Stripe keys, Clerk issuer) to the **Production** environment in your Convex Dashboard settings, separate from your Development environment.