EpicFreelancer Technical Documentation

1. Architecture Overview

EpicFreelancer is a modern, scalable multi-marketplace platform built as a Single Page Application (SPA) with specific Static Site Generation (SSG) capabilities for SEO-critical sections (Blog).

The application follows a **Headless Architecture**:

- **Frontend**: A React application built with Vite and TypeScript, utilizing a component-based architecture (Shadcn UI). It handles the user interface, client-side routing, and state management.
- **Backend**: Powered by Supabase (Backend-as-a-Service). It manages authentication, the PostgreSQL database, real-time subscriptions, and file storage.
- **Edge Logic**: Supabase Edge Functions are used for secure server-side logic, specifically for handling Stripe payments and complex image generation tasks.
- **SEO Layer**: A custom local package (`react-supabase-ssg-blog`) utilizes Puppeteer during the build process to pre-render blog pages into static HTML,

allowing for high performance and SEO on standard shared hosting.

2. Tech Stack

Frontend

- **Framework**: React 18
- **Build Tool**: Vite
- **Language**: TypeScript
- **Styling**: Tailwind CSS, PostCSS
- **UI Components**: Shadcn UI (Radix Primitives), Lucide React (Icons)
- **Routing**: React Router DOM 6
- **State Management/Data Fetching**: TanStack Query (React Query)
- **Forms**: React Hook Form + Zod (Validation)

Backend & Infrastructure

- **Database**: PostgreSQL (via Supabase)
- **Auth**: Supabase Auth
- **Storage**: Supabase Storage (S3 compatible)
- **Serverless**: Supabase Edge Functions (Deno)

Integrations

- **Payments**: Stripe (Checkout & Webhooks)
- **AI**: Google Gemini API (Product generation), Cloudflare Flux (Image generation)

3. Project Structure

```
EpicFreelancer.com/
├── public/                # Static assets and .htaccess configurations
├── packages/              # Local workspace packages
│   └── react-supabase-ssg-blog/ # Custom SEO/SSG Blog module
├── src/
│   ├── components/        # Reusable UI components
│   │   ├── admin/         # Admin panel specific components
│   │   ├── dashboard/     # User/Seller dashboard components
│   │   ├── detail/        # Product/Service detail views
│   │   ├── onboarding/    # Seller onboarding flows
│   │   └── ui/            # Shadcn primitive components
│   ├── contexts/          # React Contexts (Auth, Cart, Settings)
```

```
│   ├── hooks/              # Custom React hooks (Supabase data, UI logic)
│   ├── integrations/       # API clients (Supabase, Stripe)
│   ├── lib/                # Utility functions and API helpers
│   ├── pages/              # Application route pages
│   └── App.tsx             # Main application entry point
├── supabase/
│   ├── edge-functions/     # Server-side logic (Stripe, AI)
│   └── migrations/         # SQL database schema definitions
└── index.html              # Entry HTML file
```

4. Database Schema

The database is hosted on PostgreSQL. The schema is defined in the `supabase/migrations` folder. Key tables include:

Core Tables

- `users`: Extends Supabase `auth.users`. Stores basic user profile info.
- `seller_profiles`: Linked 1:1 to users. Stores business data, PayPal info, and approval status.
- `categories`: Lookup table for marketplace categories (Digital Products, Services, Micro SaaS).
- `listings`: The central table for all items.
  - `type`: Enum ('product', 'service', 'tool').
  - `pricing_type`: Enum ('fixed', 'hourly', etc.).
  - `status`: Enum ('active', 'pending', etc.).
- `orders`: transactional records linking buyers to listings.
- `reviews`: Ratings and comments linked to listings and buyers.

Blog & SEO

- `blog_articles`: Stores content, slug, SEO metadata, and publication status.

Security

- **Row Level Security (RLS)** is enabled. Policies are defined to ensure: