# Technical Documentation: GolfWeGo

## Architecture Overview

GolfWeGo is a modern Vertical SaaS application designed for the golf industry, functioning as a marketplace and management platform for golf courses (Providers) and golfers (Buyers).

The application utilizes a Serverless architecture:

- **Frontend**: A Single Page Application (SPA) built with React and Vite. It communicates directly with the backend via WebSocket/HTTP.
- **Backend & Database**: Handled entirely by **Convex**, which provides a

real-time reactive database, serverless functions (queries/mutations/actions), and file storage.

- **Authentication**: Managed by **Clerk**, providing secure JWT-based authentication integrated directly with Convex.
- **Payments**: Powered by **Stripe Connect** for split payments, subscription billing, and secure checkout.
- **AI Integration**: Utilizes the **Google Gemini API** (via `@google/genai`) for generating personalized course recommendations and AI Caddie insights.

## Tech Stack

### Frontend

- **Framework**: React 19
- **Routing**: React Router v7
- **Build Tool**: Vite
- **Styling**: Tailwind CSS
- **Animations**: Framer Motion
- **Icons**: Lucide React
- **Markdown Parsing**: React Markdown

### Backend & Services

- **BaaS**: Convex (`convex` package)
- **Auth**: Clerk (`@clerk/clerk-react`)
- **Payments**: Stripe (`stripe` Node SDK, `@stripe/react-stripe-js`)
- **AI**: Google Gen AI SDK (`@google/genai`)
- **Mock Data**: Faker.js (for database seeding)

## Project Structure

```
├── components/          # Reusable UI components (Navbar, Buttons, List
```

```
├── convex/              # Backend logic and database configuration
│   ├── _generated/      # Auto-generated Convex types and API routes
│   ├── ai.ts            # Google Gemini AI actions (recommendations, ar
│   ├── bookings.ts      # Reservation logic and endpoints
│   ├── schema.ts        # Database schema definition
│   ├── seed.ts          # Mock data generator for testing
│   ├── stripe.ts        # Stripe webhooks, Connect onboarding, and chec
│   └── users.ts         # User management and Clerk synchronization
├── lib/                 # Utility hooks and helpers
├── pages/               # React Router page components (Auth, Dashboard
├── public/              # Static assets and .htaccess for server routir
├── App.tsx              # Main application layout and route protection
├── index.tsx            # React DOM entry point and context providers
└── vite.config.ts       # Vite build configuration
```

## Database Schema

The database is built on Convex. Below are the primary tables and their relationships inferred from `convex/schema.ts`:

### Core Entities

- `users`: Stores all user accounts (Golfers, Providers, Admins). Links to Clerk via `clerkId` and Stripe via `stripeCustomerId` / `stripeAccountId`.
  - *Fields*: `name`, `email`, `clerkId`, `role`, `onboarded`, `subscriptionPlan`, `stripeConnected`, etc.
- `services`: The listings created by providers (tee times, rentals, packages).
  - *Fields*: `name`, `description`, `price`, `type` (TEE_TIME/PACKAGE/RENTAL), `providerId`, `rating`, `location`, `difficulty`.
- `bookings`: Records of reservations made by golfers.
  - *Fields*: `serviceId`, `golferId`, `providerId`, `date`, `status`, `amount`, `stripePaymentIntentId`.
- `transactions`: Financial records tracking payouts and payments to

- `transactions`: Financial records tracking payouts and payments to providers.
  - *Fields*: `providerId`, `amount`, `date`, `status`, `description`.
- `reviews`: Ratings and comments left by golfers after completing a booking.
  - *Fields*: `serviceId`, `golferId`, `rating`, `comment`, `date`.

### Platform & Settings

- `appSettings`: Global platform configurations.
  - *Fields*: `platformName`, `platformLogoId`, `proPrice`, `stripePriceId`, `platformCommission`.
- `providerSettings`: White-label storefront configurations for individual golf courses.
  - *Fields*: `userId`, `name`, `logoId`, `description`, `subdomain` (for storefront URLs).

### Ancillary Data

- `events` / `eventRegistrations` / `eventInquiries`: Manages platform-hosted tournaments and events.
- `testimonials`: Stores reviews used on the landing page.
- `newsletter` / `jobApplications` / `contactInquiries`: Marketing and HR data.

## Setup & Installation

### 1. Prerequisites

- Node.js (v18 or higher recommended)
- Convex Account
- Clerk Account
- Stripe Account
- Google Gemini API Key

## 2. Environment Variables

Create a `.env.local` file in the root directory and add the following keys:

```
# Frontend Variables (Vite)
VITE_CLERK_PUBLISHABLE_KEY=your_clerk_publishable_key
VITE_CONVEX_URL=your_convex_deployment_url
VITE_STRIPE_PUBLISHABLE_KEY=your_stripe_publishable_key

# Backend Variables (Add these in the Convex Dashboard -> Settings -> Er
CLERK_SIGNING_KEY=your_clerk_issuer_url
STRIPE_SECRET_KEY=your_stripe_secret_key
STRIPE_WEBHOOK_SECRET=your_stripe_webhook_secret
HOST_URL=http://localhost:5173
GEMINI_API_KEY=your_gemini_api_key
```

## 3. Installation

Install the required NPM packages:

```
npm install
```

## 4. Initialize Backend

Start the Convex development server. This will prompt you to log in and configure your Convex project, automatically setting the `VITE_CONVEX_URL` in your `.env.local` file.

```
npx convex dev
```

## 5. Seed the Database

To populate the application with mock providers, users, and golf services, run the seed script:

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

### 6. Start the Development Server

```
npm run dev
```

### Admin Access

To access the `/admin` portal, user is codeblix and pass is codeblix.

## Deployment Guide

Shared hosting is enough, no vercel required.

To deploy the application to a production environment:

1. **Build the Project**: Run the Vite build command to generate the static production files.

   ```
   npm run build
   ```

2. **Upload to Server**: Upload the contents of the generated `dist` folder to the public directory of your shared hosting environment (e.g., `public_html` or `www`).

3. **Routing Configuration**: Because this is a Single Page Application using