

PROJECT DESIGN DOCUMENT: "SoulAlign" Spiritual Coach

Date: October 26, 2023

Role: Senior Software Architect

Status: Draft / Phase 1 Planning

1. Executive Summary

SoulAlign is an AI-powered spiritual coaching platform designed to guide users through personal transformation, trauma healing, and goal achievement.

Unlike standard chatbots, this system utilizes a **Dual-Agent Architecture**:

1. **The Guide:** An interactive persona that adapts its tone and advice based on specific spiritual masters (e.g., Buddhism, Stoicism, Jungian Psychology) selected by the user.
2. **The Observer:** A background process that analyzes conversations to identify subconscious blockages, self-sabotage patterns, and trauma triggers, storing them for long-term coaching context.

2. Technology Stack Selection

We will utilize a **T3-Adjacent Stack** (TypeScript, Next.js, Database) optimized for rapid iteration, type safety, and high-performance AI streaming.

Frontend

- **Framework:** **Next.js 14+ (App Router)** - *Server-side rendering for performance and SEO.*
- **Language:** **TypeScript** - *Strict typing to prevent runtime errors.*
- **Styling:** **Tailwind CSS** - *Utility-first styling for speed.*
- **Animation:** **Framer Motion** - *Essential for creating a "calm" UX (breathing animations, smooth transitions).*
- **State Management:** **Zustand** - *Lightweight global state for chat history.*

Backend & Database

- **Platform:** **Supabase** (BaaS).
- **Database:** **PostgreSQL**.
- **Vector Search:** **pgvector** (inside Supabase) - *Used to store embeddings of spiritual texts for Retrieval Augmented Generation (RAG).*
- **Auth:** **Supabase Auth** (Email/Social Login).

AI & Orchestration

- **LLM Provider: Anthropic Claude 3.5 Sonnet** (Primary Recommendation via API) - *Selected for superior emotional intelligence and nuance compared to GPT-4.*
- **Orchestration: LangChain.js or Vercel AI SDK** - *To manage prompt chaining and streaming.*

3. Database Schema Design

The database requires a hybrid approach: Relational data for user settings and Vector data for the "Knowledge Base."

A. Relational Tables (User Data)

users

- **id** (UUID, Primary Key)
- **email** (String)
- **full_name** (String)
- **goals** (JSONB) - *Array of active goals (e.g., ["Reduce anxiety", "Find purpose"]).*
- **spiritual_config** (JSONB) - *Stores the "Council" configuration.*
 - *Example:* { "masters": ["Rumi", "Marcus Aurelius"], "tone": "Direct", "faith_filter": "Secular" }

sessions

- **id** (UUID, PK)
- **user_id** (FK)
- **title** (String) - *Auto-generated summary of the chat topic.*
- **created_at** (Timestamp)

messages

- **id** (UUID, PK)
- **session_id** (FK)
- **role** (Enum: 'user', 'assistant', 'system')
- **content** (Text)
- **metadata** (JSONB) - *Stores token usage and sentiment score.*

insights (The Awareness Engine)

- **id** (UUID, PK)
- **user_id** (FK)
- **category** (Enum: 'blockage', 'pattern', 'progress', 'trauma_trigger')

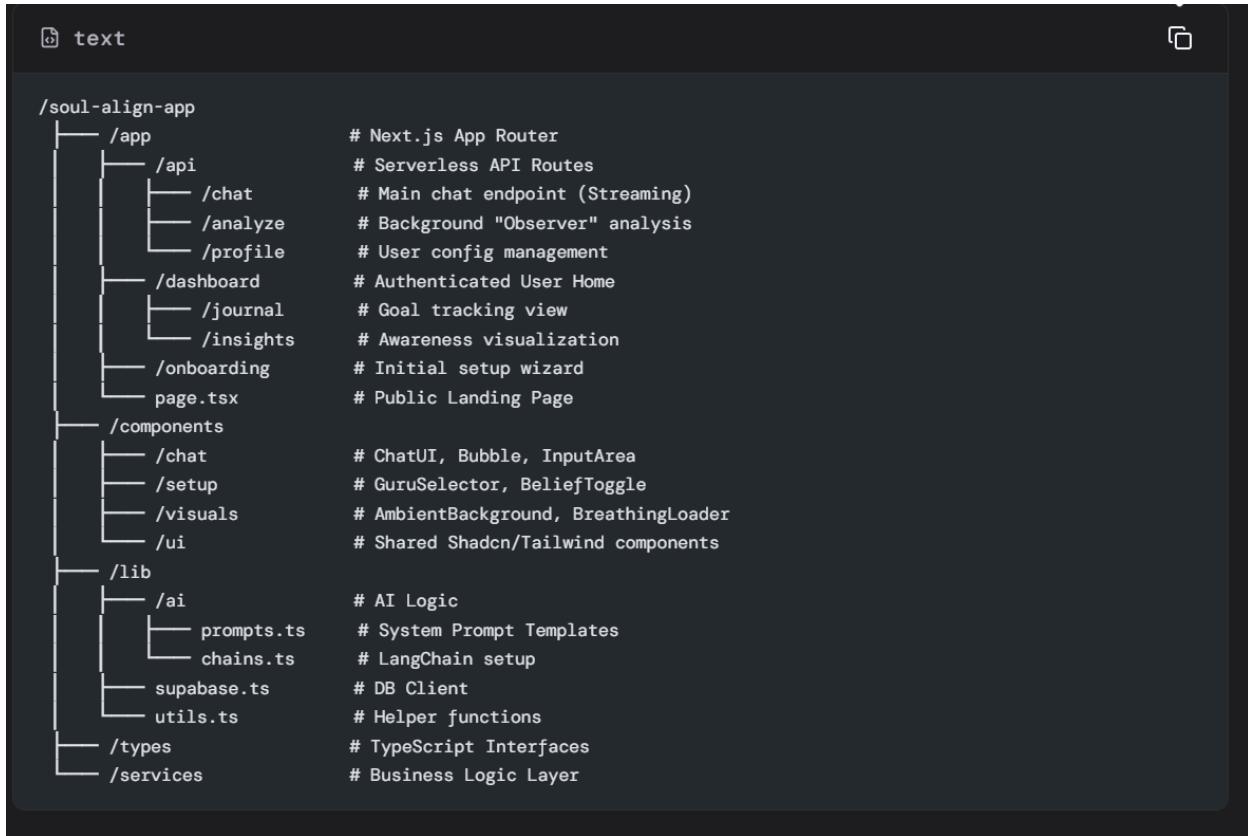
- **observation** (Text) - e.g., "User deflects when asked about father figures."
- **detected_at** (Timestamp)

B. Vector Table (The Brain)

`knowledge_base`

- **id** (UUID)
- **content** (Text) - A chunk of text from a spiritual book or teaching.
- **author** (String) - e.g., "Eckhart Tolle"
- **embedding** (Vector) - 1536-dimensional vector representation.

4. Project File Structure



```

@soul-align-app
├── /app
│   ├── /api
│   │   ├── /chat
│   │   ├── /analyze
│   │   ├── /profile
│   │   ├── /dashboard
│   │   ├── /journal
│   │   └── /insights
│   ├── /onboarding
│   └── page.tsx
├── /components
│   ├── /chat
│   ├── /setup
│   ├── /visuals
│   └── /ui
└── /lib
    ├── /ai
    │   ├── prompts.ts
    │   ├── chains.ts
    │   ├── supabase.ts
    │   └── utils.ts
    ├── /types
    └── /services

```

5. Core Logic & API Endpoints

A. Endpoint: POST /api/chat (The Guide)

This is the real-time interaction layer.

1. **Receive:** `message, history, userId`.

2. **Fetch Context:** Retrieve user's `spiritual_config` (e.g., "Buddhist + Stoic").
3. **RAG Retrieval:** (Optional) If the user asks a specific question, search `knowledge_base` for relevant quotes from the selected masters.
4. **System Prompt Assembly:**

"You are a spiritual guide blending the voices of [Author A] and [Author B]. The user is working on [Goal]. Your tone is [Tone]. Use the following retrieved context to answer..."

5. **Stream:** Return text response to frontend.
6. **Trigger:** Asynchronously call `/api/analyze`.

B. Endpoint: POST `/api/analyze` (The Observer)

This runs in the background after a message is exchanged.

1. **Input:** The recent conversation exchange.
2. **Logic:** Uses a specialized AI prompt to act as a Psychologist/Analyst.

"Analyze the user's last message. Are they displaying cognitive dissonance, fear, or a specific trauma response? If yes, extract a brief 'Insight' summary."

3. **Action:** If an insight is found, write to the `insights` table.

C. Endpoint: POST `/api/profile/config`

Updates the user's spiritual "Council."

6. Frontend Component Features

1. The Guru Selector (Onboarding)

- **UI:** A visual grid of spiritual archetypes/authors (e.g., The Mystic, The Philosopher, The Scientist).
- **Logic:** Allows the user to multiselect. This selection dictates the `vocabulary` the AI uses.

2. The Immersive Chat

- **UI:** Minimalist. No clutter.
- **Feature:** "Centering Moment" - Before the chat activates, the screen pulses for 3 seconds, prompting the user to breathe.
- **Input:** Supports text and voice (Speech-to-Text).

3. The Awareness Map (Dashboard)

- **UI:** A visual list or graph of the `insights` table.
- **Feature:** Displays "Recurring Patterns."
 - *Example:* "The AI has noticed you often feel unworthy when discussing career. Click here to explore this."

7. Implementation Phases

Phase 1: The Foundation (MVP)

- Setup Next.js & Supabase.
- Build Authentication.
- Build the "Guru Selector" (store preferences in JSON).
- Create basic Chat interface connecting to Claude/OpenAI.
- **Goal:** User can chat with a specific "Persona."

Phase 2: The Memory & Observer

- Implement the `insights` table.
- Build the "Observer" background agent to tag user blockages.
- Update the Chat System Prompt to *read* previous insights so the AI remembers the user's trauma history.
- **Goal:** The AI remembers context and patterns across sessions.

Phase 3: The Knowledge Base (RAG)

- Scrape/Import public domain spiritual texts (Tao Te Ching, Meditations, etc.).
- Vectorize them into Supabase.
- Connect chat to Vector search.
- **Goal:** The AI quotes actual texts to support its advice.

Phase 4: Visuals & Polish

- Add Framer Motion animations.
- Implement "Breathing Exercises" features.
- Mobile responsiveness.