# My 4 Blocks: Voice & Chat Architecture

> *A guide to how typed chat and voice mode work—including system prompts, RAG retrieval, and information flow.*

---

## Overview

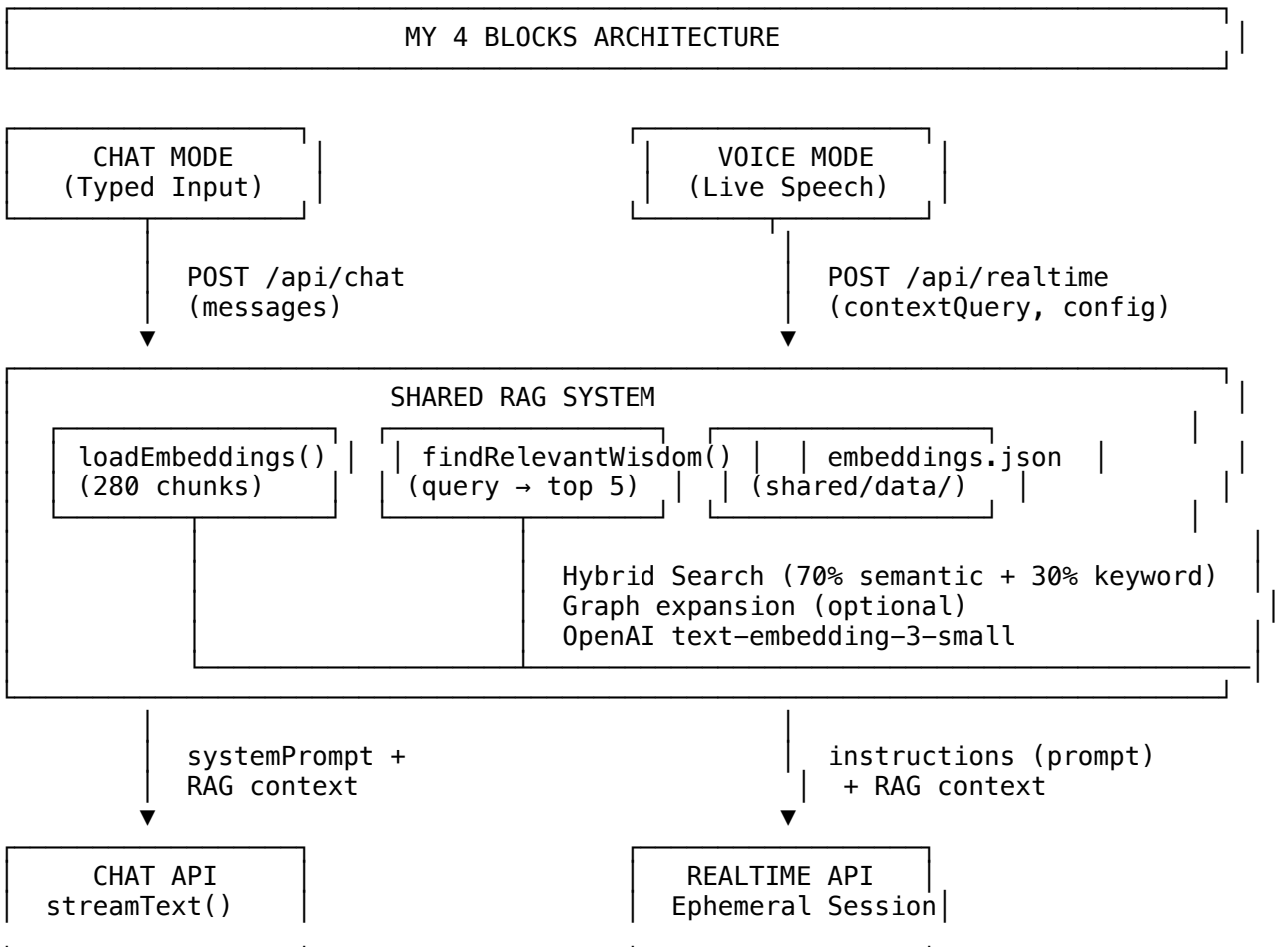
My 4 Blocks offers two ways to interact with the AI guide:

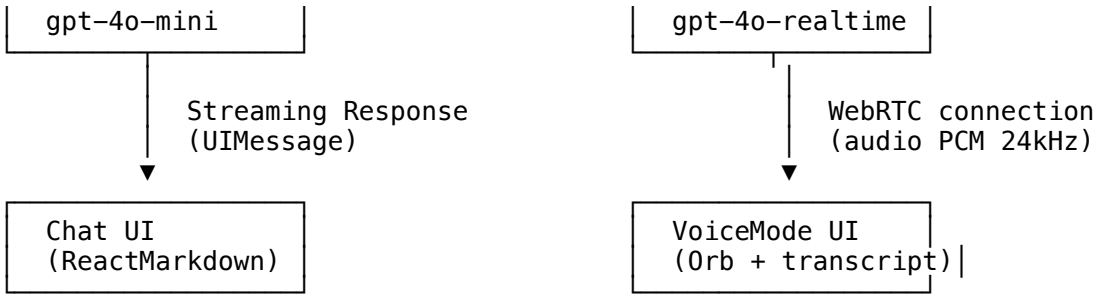
| Mode | Input | Output | API |
|---|---|---|---|
| **Chat** | Typed text | Streamed text (markdown) | Vercel AI SDK → OpenAI Chat |
| **Voice** | Live speech | Real-time speech | OpenAI Realtime API (WebRTC) |

Both modes share the same **RAG retrieval system** and **core knowledge base**, but differ in how they deliver that knowledge to the user.

---

## Architecture Diagram






---

# Chat Mode

## Flow

1. **User types** → `sendMessage()` from `useChat()`
2. **Client** sends `POST /api/chat` with full message history
3. **Server** (`handleChatRequest`):
   - Loads RAG embeddings (lazy init)
   - Extracts query from last user message
   - Calls `findRelevantWisdom(query, 5)` → hybrid search
   - Builds `systemPrompt = SYSTEM_PROMPT + "\n\n## Relevant Book Context\n" + ragContext`
   - Streams via `streamText()` with `system`, `messages`, `temperature`, `maxTokens`
4. **Response** streamed back as `UIMessage` stream
5. **Client** renders markdown with `ReactMarkdown`

## System Prompt (Chat)

The chat system prompt is **richer** and includes:

- **Book structure** – Chapter outline
- **Four Blocks** – Nuanced definitions (Anger, Anxiety, Depression, Guilt)
- **Depression vs Guilt** – Exact distinction
- **ABC Model** – A–E (Activating Event, Belief, Consequence, Disputing, Effective new belief)
- **Seven Irrational Beliefs** – Full list
- **Three Insights** – Core principles
- **Narrator vs Observer** – Mindfulness framing
- **Communication style** – Warm, compassionate, non-judgmental
- **Key quotes** – Dr. Parr, Dōgen

RAG adds **top 5 relevant chunks** from the book under `## Relevant Book Context`.

## Configuration

| Option | Default | Description |
|---|---|---|
| `model` | `gpt-4o-mini` | Chat model |
| `temperature` | 0.7 | Response randomness |
| `maxTokens` | 2000 | Max tokens per response |

| | | |
|---|---|---|
| `ragEnabled` | true | RAG retrieval |
| `ragTopK` | 5 | Chunks retrieved |

---

# Voice Mode

## Flow

1. **User** starts voice session → `VoiceMode` component calls `POST /api/realtime`
2. **Request** body: `{ contextQuery?, config? }` (voice, style, ragEnabled, etc.)
3. **Server** (`handleRealtimeRequest`):
   - Loads RAG embeddings (lazy init)
   - Calls `buildVoiceInstructions(contextQuery, config)`:
     - Builds voice system prompt with style
     - If `ragEnabled` and `contextQuery`, adds RAG context
   - Creates ephemeral session via `POST https://api.openai.com/v1/realtime/sessions`
   - Returns `{ client_secret, id, ... }` for WebRTC
4. **Client**:
   - Uses `client_secret` to establish WebRTC connection
   - Sends audio (PCM 24kHz) via WebRTC
   - Receives audio stream, plays via `AudioContext`
   - Shows transcript from `data` channel

## System Prompt (Voice)

Voice uses the same knowledge base but **style-specific** instructions:

- **Style** – `direct` (default), `warm`, `casual`, `professional`
- **Book structure** – Same chapter outline
- **Four Blocks** – Same definitions
- **Depression vs Guilt** – Same distinction
- **ABC Model** – A → B → C
- **Seven Irrational Beliefs** – Short list
- **Three Insights** – Same core principles

RAG adds **top 5 relevant chunks** from the book under `## Relevant Book Context`.

## Voice Style Presets

| Style | Description |
|---|---|
| **direct** | Get to the point; no fluff; normal pace |
| **warm** | Friendly, supportive; natural rhythm |
| **casual** | Casual, conversational; everyday language |
| **professional** | Clear, structured; evidence-based |

## Voice Options

9 voices: `ash`, `alloy`, `ballad`, `coral`, `echo`, `marin`, `sage`, `shimmer`, `verse`.

### Configuration

| Option | Default | Description |
|---|---|---|
| `voice` | `ash` | TTS voice |
| `style` | `direct` | Conversation style |
| `model` | `gpt-4o-realtime-preview-2024-12-17` | Realtime model |
| `ragEnabled` | true | RAG retrieval |
| `ragTopK` | 5 | Chunks retrieved |
| `turnDetection` | `semantic_vad`, `low` | When to treat user as finished speaking |

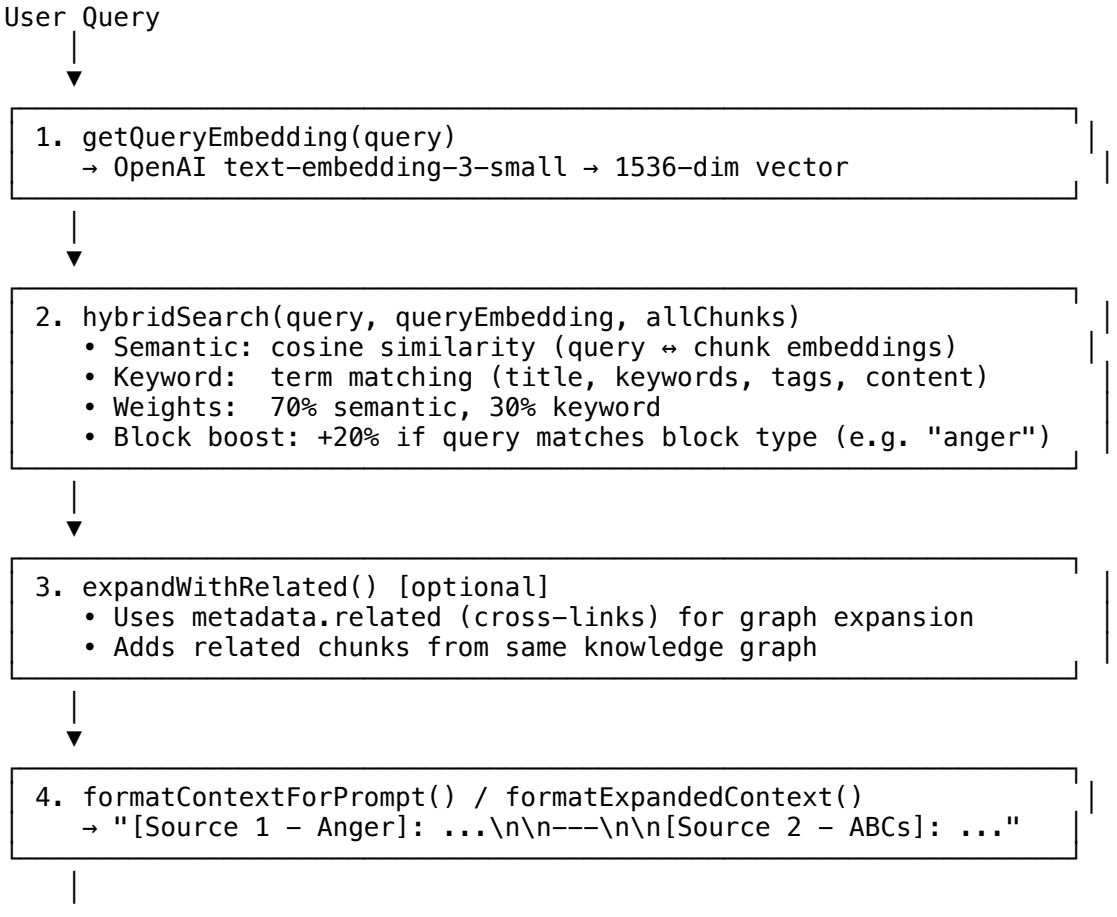
---

# RAG: Retrieval-Augmented Generation

## Data Source

- **Source**: full PDF of *You Only Have Four Problems by Dr. Vincent E. Parr*
- **Chunking**: Chonkie `TokenChunker` (500 tokens, 100 overlap)
- **Embeddings**: OpenAI `text-embedding-3-small` (1536 dims)
- **Storage**: `shared/data/embeddings.json` (~280 chunks)

## Retrieval Pipeline



▼

```
Appended to system prompt as "## Relevant Book Context"
```

### Fallback

If the embedding API fails, the system falls back to **keyword-only** search.

---

# Information Management Summary

| Aspect | Chat | Voice |
|---|---|---|
| **Context window** | Full message history sent each request | Ephemeral session; instructions fixed at session start |
| **RAG trigger** | Query = last user message | Query = optional `contextQuery` (e.g. suggested prompt) |
| **Streaming** | Text stream (UIMessage) | Audio stream (WebRTC) |
| **Transcription** | N/A | Whisper-1 on server |
| **Turn detection** | N/A | Semantic VAD (server decides when user finished) |
| **Session persistence** | Client maintains conversation | Session ends when WebRTC disconnected |

---

# File Reference

| File | Purpose |
|---|---|
| `shared/api/chat.ts` | Chat API handler, `SYSTEM_PROMPT`, `handleChatRequest` |
| `shared/api/realtime.ts` | Voice API handler, `buildSystemPrompt`, `createRealtimeSession` |
| `shared/lib/rag.ts` | `getRAGContext`, `findRelevantWisdom`, `loadEmbeddings` |
| `shared/lib/hybridSearch.ts` | Hybrid semantic + keyword search |
| `shared/lib/embeddings.ts` | Query embedding (OpenAI or local) |
| `shared/lib/vectorSearch.ts` | Cosine similarity, `formatContextForPrompt` |
| `shared/lib/keywordSearch.ts` | Keyword search, emotion keyword expansion |
| `shared/lib/graphExpansion.ts` | Cross-link expansion via `metadata.related` |
| `shared/data/embeddings.json` | Pre-computed embeddings (280 chunks) |
| `shared/components/VoiceMode.tsx` | Voice UI, WebRTC, orb, transcript |

---

# Suggested Prompts (Chat Page)

When the user starts a chat, they can choose from suggested prompts:

- **Managing Anger** – "I keep getting angry at things I can't control. How can I stop?"
- **Understanding Anxiety** – "I'm worried about what might happen. How can I stop worrying?"

- **Overcoming Depression** – "I feel hopeless and unmotivated. What can I do?"
- **Releasing Guilt** – "I feel guilty about something I did. How can I let it go?"
- **The ABCs Model** – "How do my thoughts create my emotions?"
- **Core Beliefs** – "What are the irrational beliefs that cause suffering?"

These prompts are used both as **chat starters** and as **contextQuery** for voice sessions when the user starts voice mode from a suggested prompt.

---

*Document generated for My 4 Blocks—the emotional wellness chat powered by the Four Blocks framework.*