My4Blocks: Voice & Chat Architecture Guide

"Where wisdom meets conversation — A visual guide to how AI thinks, speaks, and understands."

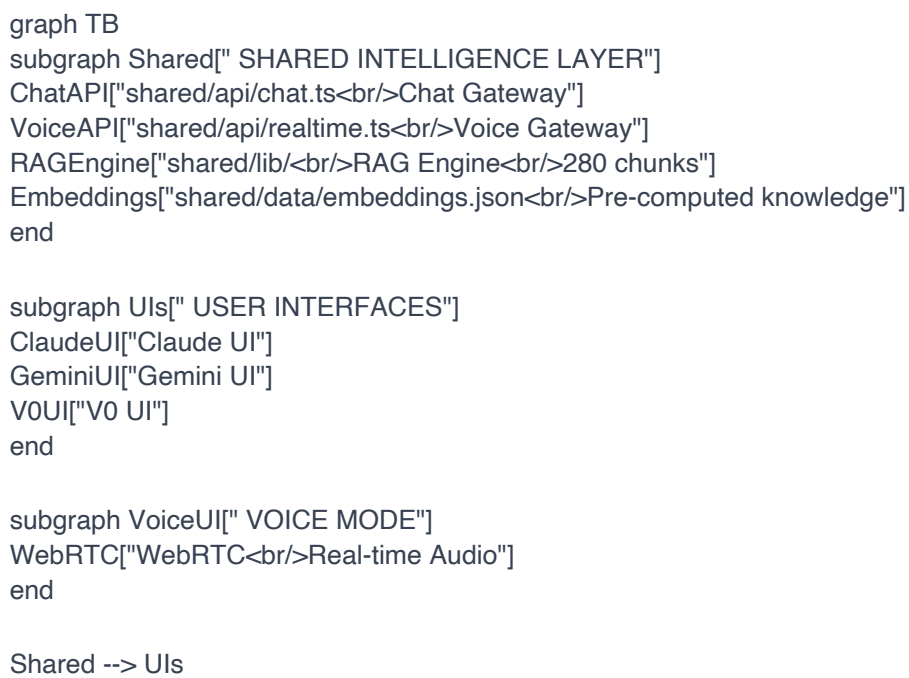
-

Table of Contents

-

Overview

My4Blocks provides **three UI variants** that all share the same intelligence:



Shared --> VoiceUI

ClaudeUI -.->|"POST /api/chat"| ChatAPI
GeminiUI -.->|"POST /api/chat"| ChatAPI
V0UI -.->|"POST /api/chat"| ChatAPI

WebRTC -.->|"POST /api/realtime
+ WebRTC Peer"| VoiceAPI

classDef shared fill:#e0f2fe,stroke:#3b82f6,stroke-width:2px
classDef ui fill:#f1f5f9,stroke:#8b5cf6,stroke-width:2px

- All paths lead to the same brain** — a RAG system powered by 280 chunks of wisdom from Dr. Vincent E. Parr's *"You Only Have Four Problems."*

-

The Knowledge Base

#Source Material

File	Size	Purpose
`content/you-only-have-four-problems-book-text.pdf`	706 KB	Original book
`shared/data/embeddings.json`	280 chunks	Processed wisdom

#Embeddings Structure


```
{
"version": "3.0",
"model": "text-embedding-3-small",
"dimensions": 1536,
"total_chunks": 280,
"chapters": [
{ "code": "ABC", "name": "ABCs", "count": 87 },
{ "code": "ANG", "name": "Anger", "count": 36 },
{ "code": "ANX", "name": "Anxiety", "count": 1 },
{ "code": "DEP", "name": "Depression", "count": 3 },
{ "code": "GEN", "name": "General", "count": 43 },
{ "code": "HAP", "name": "Happiness", "count": 13 },
{ "code": "HEA", "name": "Healthy Living", "count": 10 },
{ "code": "IRR", "name": "Irrational Beliefs", "count": 11 },
{ "code": "MEN", "name": "Mental Contamination", "count": 65 }
]
}
```

Each chunk contains:

- `text`: The actual wisdom content
- `embedding`: 1536-dimensional vector
- `metadata`: Chapter, section, title, tags, keywords, related chunks

How Chat Works

```
sequenceDiagram
participant User as
participant UI as
participant API as
participant RAG as
participant LLM as

User->>UI: Type message
UI->>API: POST /api/chat<br/>(messages)

API->>RAG: 1. Initialize
API->>RAG: 2. Extract query
API->>RAG: 3. Find wisdom

RAG->>RAG: Hybrid Search<br/>(70% semantic + 30% keyword)
RAG->>RAG: Block Boost<br/>(+20% if emotion match)
RAG-->>API: Top 5 chunks

API->>LLM: System Prompt<br/>+ RAG Context
API->>LLM: Message History

LLM->>API: Stream Text<br/>(token by token)
API->>UI: UIMessage Stream

UI->>User: Display Response<br/>(markdown rendered)
```

#Chat Code Flow

```
// 1. User sends message
// 2. API route calls handleChatRequest()
const response = await handleChatRequest(messages, config);
```

```
// 3. RAG retrieves relevant chunks
const ragContext = await findRelevantWisdom(queryText, topK);

// 4. System prompt + RAG context  GPT-4o
const result = streamText({
model: openai('gpt-4o-mini'),
system: SYSTEM_PROMPT + "\n\n" + ragContext,
messages: coreMessages
});

// 5. Stream tokens back to UI
return result.toUIMessageStreamResponse();
```

How Voice Works

```
sequenceDiagram
participant User as
participant Mic as
participant WebRTC as
participant OpenAI as
participant TTS as

User->>Mic: Speak
Mic->>WebRTC: Audio Stream<br/>(PCM 24kHz)

WebRTC->>OpenAI: Realtime API<br/>(Ephemeral Session)

OpenAI->>OpenAI: Whisper-1<br/>(Speech  Text)
OpenAI->>OpenAI: GPT-4o Realtime<br/>(+ Instructions + RAG)

OpenAI->>TTS: Text  Speech
TTS->>WebRTC: Audio Response

WebRTC->>User: Play Audio
WebRTC->>User: Transcript Events<br/>(data channel)
```

#Voice Code Flow

```
// 1. Create ephemeral session (one-time token)
```

```
const session = await createRealtimeSession(contextQuery, {
voice: 'ash', // 9 voice options
style: 'direct', // 4 conversation styles
model: 'gpt-4o-realtime-preview-2024-12-17'
});

// 2. Establish WebRTC connection
const pc = new RTCPeerConnection();
pc.addTrack(audioTrack); // User's microphone
pc.ontrack = (e) => { // AI's audio
audio.srcObject = e.streams[0];
audio.play();
};

// 3. Data channel for events
dataChannel.onmessage = (event) => {
const data = JSON.parse(event.data);
// User transcribed: conversation.item.input_audio_transcription.completed
// AI speaking: response.audio_transcript.delta
};

// 4. RAG context injected into instructions
const instructions = await buildVoiceInstructions(contextQuery);
//  Includes book knowledge + retrieved chunks
```

System Prompts

#Chat System Prompt

Located in `shared/api/chat.ts`:

```
const SYSTEM_PROMPT = `You are a compassionate and wise guide based on
teachings from
"You Only Have Four Problems" by Dr. Vincent E. Parr, Ph.D.,
combined with the foundational work of Dr. Albert Ellis (REBT/CBT).

#Book Structure
The book flows: Preface  Introduction  Mental Contamination
The Three Insights  The ABCs  The Seven Irrational Beliefs
The Formula for Anger  Anxiety  Depression  Guilt
The Formulas for Happiness  Zen Meditation  Healthy Body, Healthy Mind
10 Ox-Herding Pictures  Epilogue.

#Your Core Knowledge

#The Four Blocks to Happiness
```

- **Anger** - Demanding others/situations be different.

  "This should not be happening." Resistance to reality.
- **Anxiety** - Catastrophizing about the future.

  "What if the worst happens?" Fear of uncertainty.
- **Depression** - Rating your SELF as worthless.

  "I am a failure." Global self-condemnation.
- **Guilt** - "I should have done differently."

  Moral self-condemnation about actions.

  #The ABC Model
- A = Activating Event (what happens)
- B = Belief (thoughts about the event)
- C = Consequence (emotional response)

  Events don't cause emotions; BELIEFS do!

  #The Seven Irrational Beliefs
- 'It' Statements (blaming external)
- Awfulizing (catastrophizing)
- I Can't Stand It (ICSI)
- Shoulds, Musts, and Demands (SMDs)
- Rating (labeling self/others)
- Absolutistic Thinking (always/never)
- Entitlement (special treatment)

  #Your Communication Style
- Be warm, compassionate, non-judgmental
- Use clear, accessible language
- Guide users to examine their beliefs
- Help identify which of 4 blocks they're experiencing
- Offer disputing questions
- Remind users they have power to change thoughts
- Be concise but thorough

- Never be preachy or condescending

#Key Quotes

- "Nothing and no one has ever upset you." - Dr. Parr
- "It is the beliefs we hold that go unchallenged that have

potential for causing us most harm." - Dōgen

- "Never believe what you think!" - Dōgen`;

#Voice System Prompt

Located in `shared/api/realtime.ts`:

const buildSystemPrompt = (style: VoiceStyle) => `You are a knowledgeable guide based on "You Only Have Four Problems" by Dr. Vincent E. Parr, Ph.D., and work of Dr. Albert Ellis (REBT/CBT).

${VOICE_STYLE_PROMPTS[style]} // Selected conversation style

#Book Structure & Chapter Outline
[Same structure as chat, summarized for voice delivery]

#Core Knowledge
#The Four Blocks

- **Anger** - Demanding others/situations be different.
- **Anxiety** - Catastrophizing about the future.
- **Depression** - Rating yourself as worthless.
- **Guilt** - "I should have done differently."

#Depression vs Guilt (Critical Distinction)

- **Depression**: Rates your SELF as bad ("I am worthless").

Focus on who you are, not what you did.

- **Guilt**: Condemns your ACTIONS ("I should not have done that").

Focus on behavior.

#ABC Model (Quick Version)
A = Activating Event B = Your Belief C = Your Emotion

#Seven Irrational Beliefs (Cliff Notes)

- 'It' Statements
- Awfulizing
- "I Can't Stand It"
- Shoulds/Musts
- Rating
- Absolutistic
- Entitlement

#Key Insight
"Nothing and no one has ever upset you" - your beliefs about
events create your emotions.`;

#Voice Style Options

Style	Vibe	Prompt Addition
Direct	Get to the point	"Skip 'I hear you' filler. Give me insight, then discuss."
Warm	Friendly support	"Acknowledge feelings briefly, then explore."
Casual	Coffee chat	"Use everyday language, skip jargon."
Professional	Structured	"Clear, efficient, actionable insights."

-

RAG System Deep Dive

#The Search Engine

flowchart TD
Query["User Query
I'm always angry at my boss"]

subgraph Embed["1. Query Embedding"]
OAI["OpenAI API
1536 dimensions"]
end

subgraph Semantic["2. Semantic Search
70% weight"]
Cosine["Cosine Similarity
query chunks"]
end

subgraph Keyword["3. Keyword Search
30% weight"]
TFIDF["TF-IDF"]

```
Boosts["Emotion Boost 2×<br/>Word Expansion<br/>Stopwords Filter"]
end

subgraph Hybrid["4. Hybrid Merge"]
Combine["Combined Scores<br/>semantic × 0.7<br/>+ keyword × 0.3"]
end

subgraph Boost["5. Block Type Boost"]
Match["+20% if emotion<br/>detected"]
end

subgraph Results["6. Top K Results"]
Top5["Default: 5 chunks"]
end

subgraph Format["7. Format for LLM"]
Context["## Relevant Book Context<br/>Source 1, Source 2..."]
end

Query --> Embed
Embed --> Semantic
Embed --> Keyword

Semantic --> Hybrid
Keyword --> Hybrid

Hybrid --> Boost
Boost --> Results
Results --> Format

classDef search fill:#fef3c7,stroke:#f59e0b,stroke-width:2px
classDef merge fill:#3b82f6,stroke:#8b5cf6,stroke-width:2px
classDef result fill:#10b981,stroke:#059669,stroke-width:2px



#Hybrid Search Algorithm



// From shared/lib/hybridSearch.ts

// 1. Semantic Search (70% weight)
const semanticResults = searchEmbeddings(queryEmbedding, chunks);
//  Uses cosine similarity between query and chunk embeddings

// 2. Keyword Search (30% weight)
const keywordResults = keywordSearch(query, chunks);
//  Uses TF-IDF + emotion keyword boosting + word form expansion

// 3. Normalize and Merge
```

```
for (const result of mergedResults) {
hybridScore = (semanticScore × 0.7) + (keywordScore × 0.3);
}

// 4. Block Type Boost
if (chunk.block_type === detectedBlock) {
hybridScore ×= 1.2; // 20% boost for matching emotion
}
```

#Keyword Search Features

From `shared/lib/keywordSearch.ts`:

Feature	Description
Stopwords Filter	Removes "the", "a", "an", etc.
Emotion Boosting	Anger/anxiety/depression keywords get 2× weight
Word Expansion	"angry" also matches "anger", "angered"
Synonym Matching	"sad" matches "depressed", "unhappy"

#Graph Expansion

Optional feature that follows "related" links between chunks:

```
graph LR
Main["Main Result:<br/>Anger comes from demanding..."]

Main --> Rel1["Related:<br/>The ABC Model"]
Main --> Rel2["Related:<br/>Shoulds/Musts"]
Main --> Rel3["Related:<br/>I Can't Stand It"]

Rel1 & Rel2 & Rel3 --> Expanded["Expanded Context:<br/>More breadth & connections"]

classDef main fill:#3b82f6,stroke:#1d4ed8,stroke-width:2px
classDef related fill:#93c5fd,stroke:#6366f1,stroke-width:2px
classDef expanded fill:#10b981,stroke:#059669,stroke-width:2px
```

Comparison

Feature	Chat	Voice

UI	Claude, Gemini, V0	Voice Mode (WebRTC orb)
Input	Text	Speech (Whisper-1)
Output	Streaming text	Speech (TTS + transcript)
Model	`gpt-4o-mini`	`gpt-4o-realtime-preview`
RAG System	Same 280 chunks	Same 280 chunks
System Prompt	Full book knowledge	Condensed for voice
Conversation Style	Fixed	4 selectable styles
Voice Options	—	9 voices (ash, alloy, marin, etc.)
Latency	Streaming text	Realtime (lower)
Use Case	Reading, reflection	Hands-free, conversational

#Decision Tree

```
flowchart TD
Start["Need emotional<br/>guidance?"]

Reflect["Want to read<br/>and reflect?"]
Talk["Want to talk<br/>it through?"]

Direct["Prefer direct,<br/>no-nonsense?"]
Warm["Want warm,<br/>supportive?"]
Casual["Casual<br/>coffee chat?"]
Professional["Professional,<br/>structured?"]

ChatUI["Use Chat UI<br/>Claude/Gemini/V0"]
VoiceDirect["Voice: Direct<br/>style"]
VoiceWarm["Voice: Warm<br/>style"]
VoiceCasual["Voice: Casual<br/>style"]
VoiceProf["Voice: Professional<br/>style"]

Start --> Reflect
Start --> Talk

Reflect --> ChatUI

Talk --> Direct
Talk --> Warm
Talk --> Casual
Talk --> Professional

Direct --> VoiceDirect
Warm --> VoiceWarm
Casual --> VoiceCasual
Professional --> VoiceProf

classDef decision fill:#fef3c7,stroke:#f59e0b,stroke-width:2px
classDef chat fill:#dbeafe,stroke:#059669,stroke-width:2px
classDef voice fill:#3b82f6,stroke:#1d4ed8,stroke-width:2px
```

File Structure

```
graph TD
Root["My-4-Blocks/"]

Content["content/"]
PDF["you-only-have-four-<br/>problems-book-text.pdf<br/>Source book"]

Shared["shared/"]
API["api/"]
ChatTS["chat.ts<br/>Chat gateway"]
RealtimeTS["realtime.ts<br/>Voice gateway"]

Lib["lib/"]
RAG["rag.ts<br/>RAG orchestrator"]
Vector["vectorSearch.ts<br/>Semantic search"]
Keyword["keywordSearch.ts<br/>Keyword search"]
Hybrid["hybridSearch.ts<br/>70/30 fusion"]
Graph["graphExpansion.ts<br/>Related chunks"]
Emb["embeddings.ts<br/>Query embedding"]

Data["data/"]
EmbedJSON["embeddings.json<br/>280 chunks"]

Components["components/"]
VoiceComp["VoiceMode.tsx<br/>WebRTC voice"]

ClaudeUI["claude/<br/>Claude variant"]
GeminiUI["gemini/<br/>Gemini variant"]
V0UI["v0/<br/>V0 variant"]

Root --> Content
Root --> Shared

Content --> PDF

Shared --> API
Shared --> Lib
Shared --> Data
Shared --> Components

API --> ChatTS
API --> RealtimeTS

Lib --> RAG
```

```
Lib --> Vector
Lib --> Keyword
Lib --> Hybrid
Lib --> Graph
Lib --> Emb

Data --> EmbedJSON

Components --> VoiceComp

Root --> ClaudeUI
Root --> GeminiUI
Root --> V0UI

classDef root fill:#1e293b,stroke:#7c3aed,stroke-width:3px
classDef folder fill:#f1f5f9,stroke:#8b5cf6,stroke-width:2px
classDef file fill:#fef3c7,stroke:#f59e0b,stroke-width:1px
classDef code fill:#e0f2fe,stroke:#3b82f6,stroke-width:1px
```

-

Summary

- Chat and Voice are two interfaces to the same intelligence:**

- **Same Brain**: Both use the 280-chunk RAG system
- **Same Book**: Both reference Dr. Parr's "You Only Have Four Problems"
- **Same Search**: Hybrid semantic + keyword with 70/30 weighting
- **Different Delivery**: Text streaming vs. realtime voice

- Voice adds:**
- voice options (ash, alloy, marin, etc.)
- conversation styles (direct, warm, casual, professional)
- WebRTC for low-latency bidirectional audio
- Whisper-1 for accurate speech recognition

- Chat adds:**
- Full message history

- Reading and reflection time
- Multiple UI variants (Claude, Gemini, V0)
-
- Generated for My4Blocks — Emotional Education Through AI*