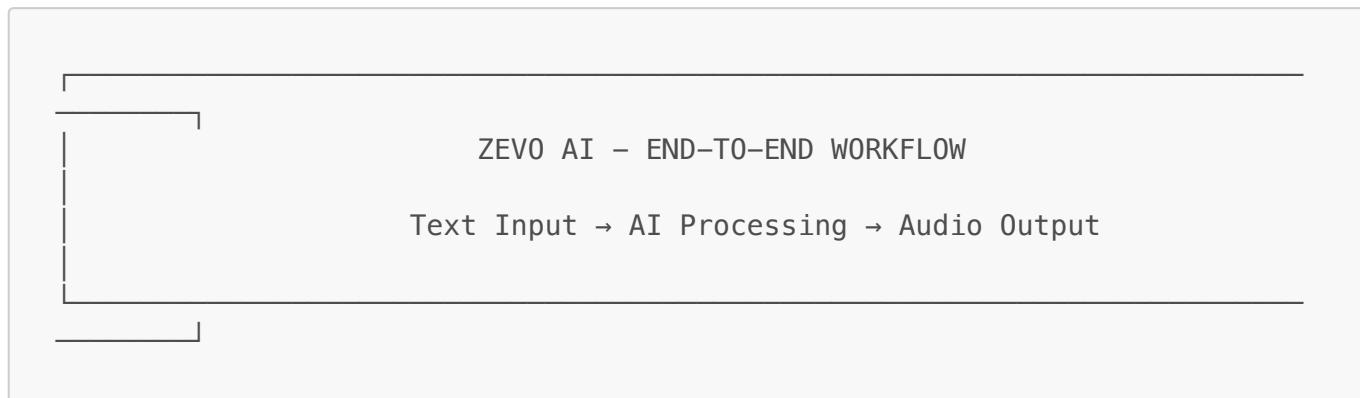
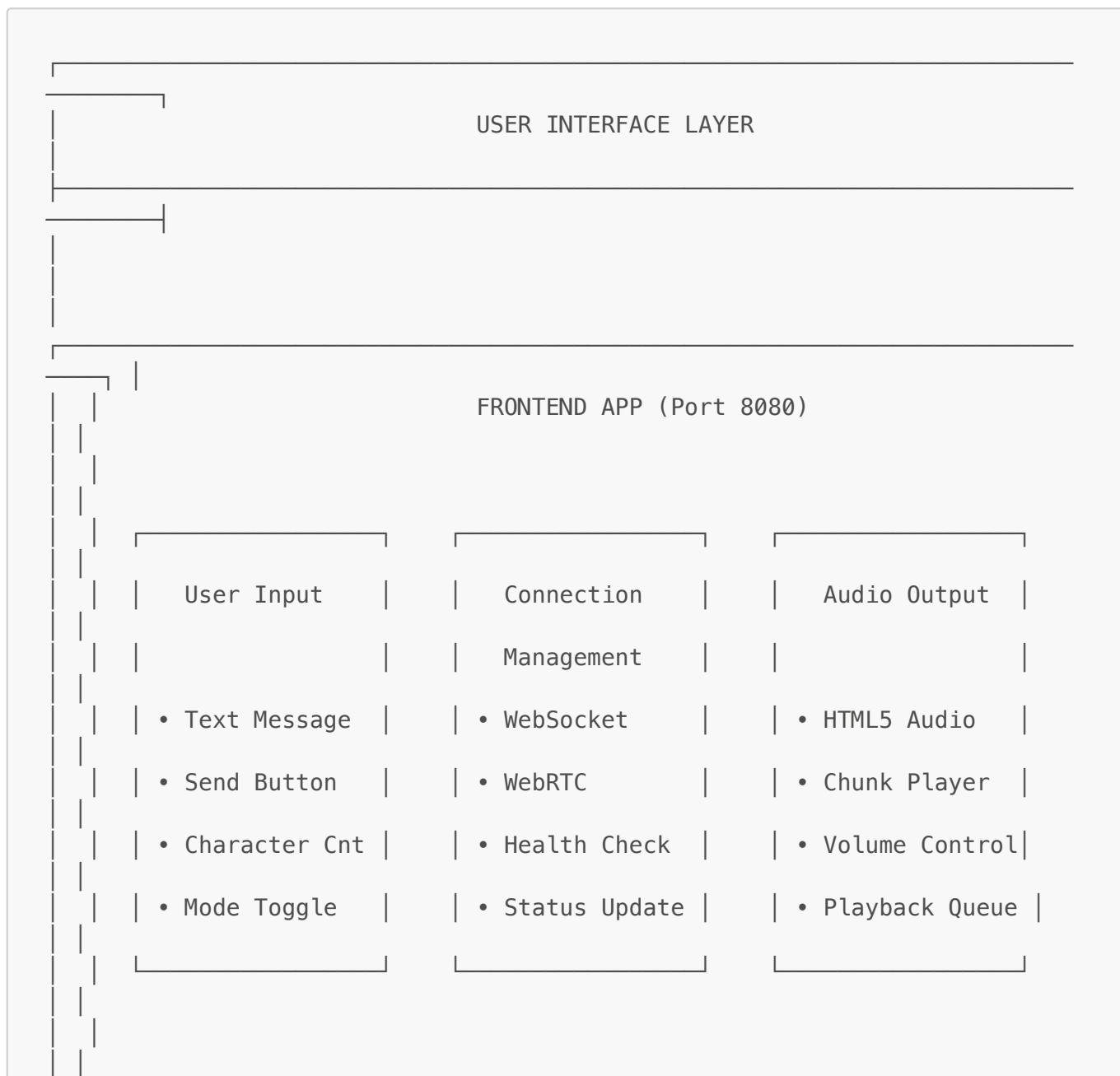


Zeko AI - End-to-End Workflow Diagram

⌚ Complete End-to-End Workflow for Text Mode



📱 Step 1: User Interface Layer



Technologies: HTML5, JavaScript, WebSocket API, WebRTC API

🔗 Step 2: Communication Layer

COMMUNICATION LAYER

PROTOCOL ROUTING

Text Mode: WebSocket (`wss://agent.zevo360.in/ws/chat/{session_id}`)

Voice Mode: WebRTC Data Channel + WebSocket fallback

Health Check: HTTP REST (`http://agent.zevo360.in/health`)

Session ID Format: `session_{timestamp}_{random_string}`

Example: `session_1758281032749_sle20ndep`

⟳ Step 3: Orchestration Service

ORCHESTRATION SERVICE (Port 8000)

PIPELINE COORDINATOR

Input: Text Message + Session ID + Conversation History

Output: Audio Chunks + Text Response + Status Updates

| Session | Context | Latency |
|-----------------|-----------------|----------------|
| Management | Preparation | Tracking |
| • Store History | • System Prompt | • Step Timing |
| • Track State | • Recent Chat | • Performance |
| • Error Handle | • Context Merge | • Optimization |

Technologies: FastAPI, Python, AsyncIO, httpx, WebSocket

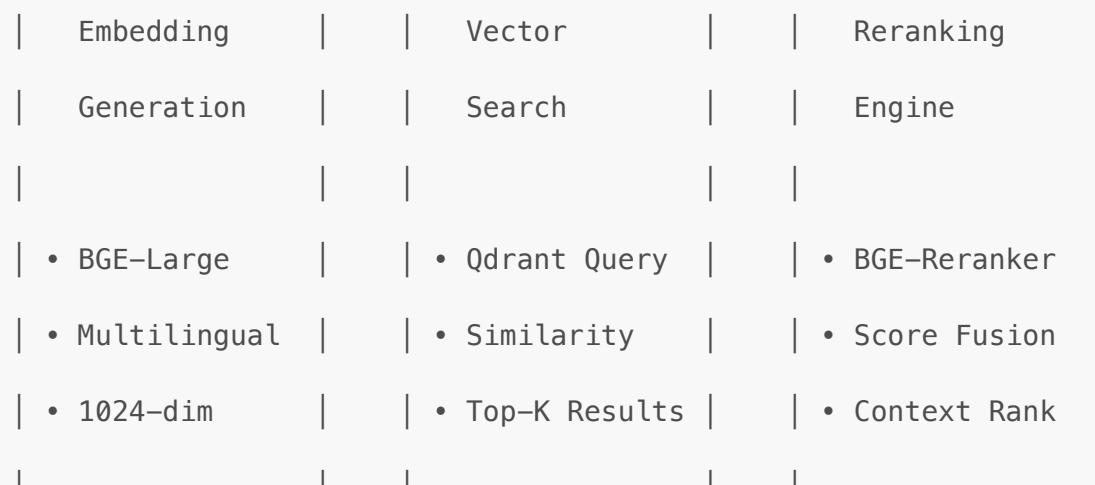
🧠 Step 4: RAG Service (Context Retrieval)

RAG SERVICE (Port 8004)

CONTEXT RETRIEVAL ENGINE

Input: User Query + Session Context

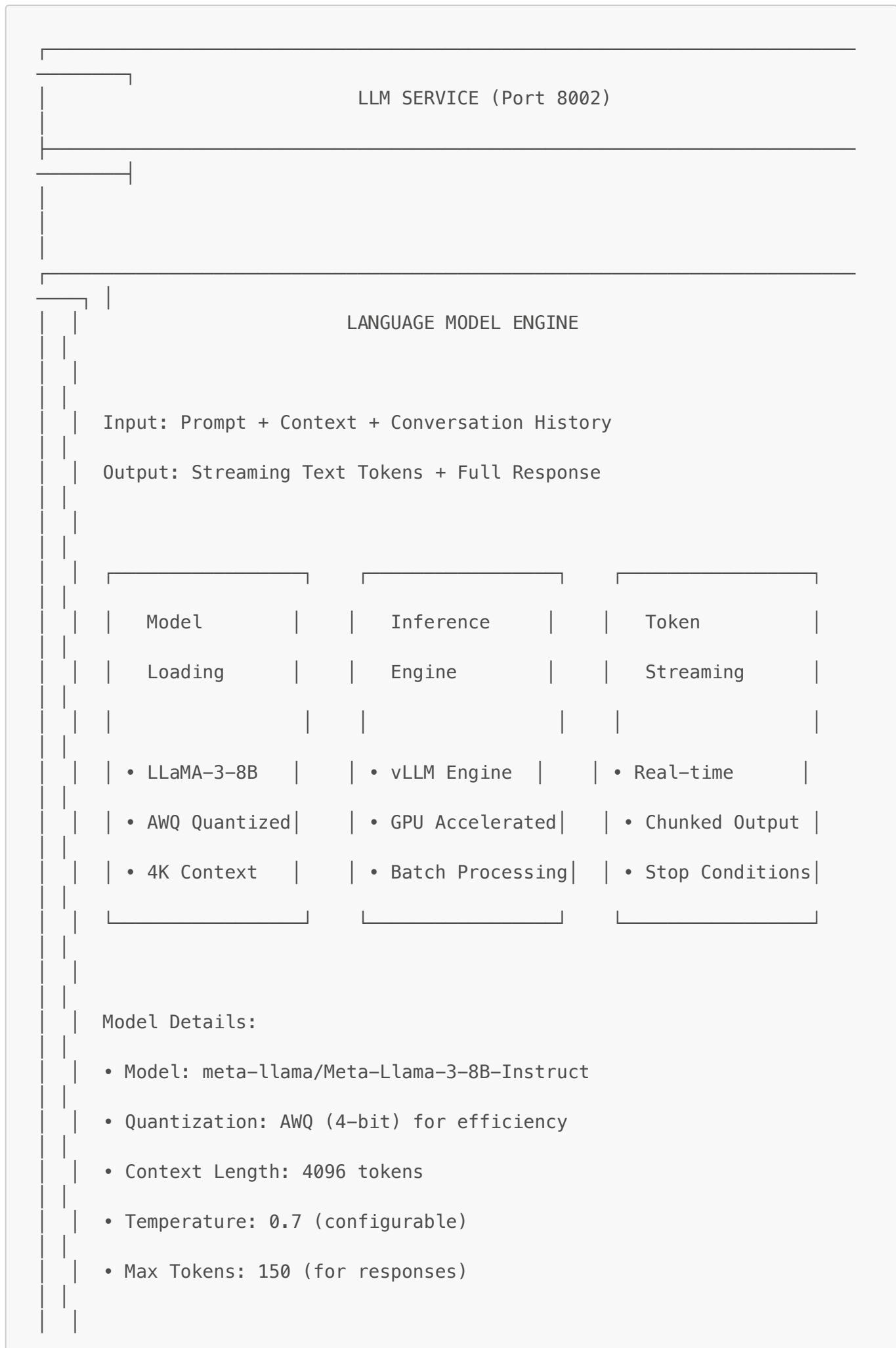
Output: Relevant Context + Confidence Scores



Models Used:

- BGE-Large-EN-v1.5 (English embeddings)
- multilingual-E5-Large (Multilingual embeddings)
- BGE-Reranker-Large (Context reranking)

Technologies: sentence-transformers, Qdrant, Python, FastAPI



Technologies: vLLM, PyTorch, CUDA, Transformers

Step 6: TTS Service (Audio Synthesis)

TTS SERVICE (Port 8003)

TEXT–TO–SPEECH ENGINE

Input: Generated Text + Voice Parameters

Output: High-Quality Audio Chunks (Streaming)

Text

Audio

Chunk

Processing

Synthesis

Streaming

• Text Analysis

• MeloTTS

• 100ms Chunks

• Emotion Detect

• Voice Cloning

• Base64 Encoded

• Speed Control

• Quality Opt

• Real-time

Audio Configuration:

- Sample Rate: 22,050 Hz (CD quality)
- Bitrate: 64 kbps (high quality)
- Format: WAV (uncompressed)
- Chunk Duration: 100ms (smooth playback)
- Voice: Default (configurable)
- Speed: 1.0x (natural speech)

Models Used:

- MeloTTS (Primary) – High-quality neural TTS
- gTTS (Fallback) – Google Text-to-Speech

Technologies: MeloTTS, PyTorch, torchaudio, pydub, webrtcvad

Step 7: Vector Database

QDRANT DATABASE (Port 6333)

VECTOR STORE

Purpose: Store and retrieve contextual information for RAG

| | | |
|-------------|---------------|-----------------|
| Collections | Vector | Similarity |
| Management | Operations | Search |
| | | |
| • Document | • Embedding | • Cosine |
| Storage | Indexing | Similarity |
| • Metadata | • Vector CRUD | • Top-K Results |
| Management | • Batch Ops | • Filtering |

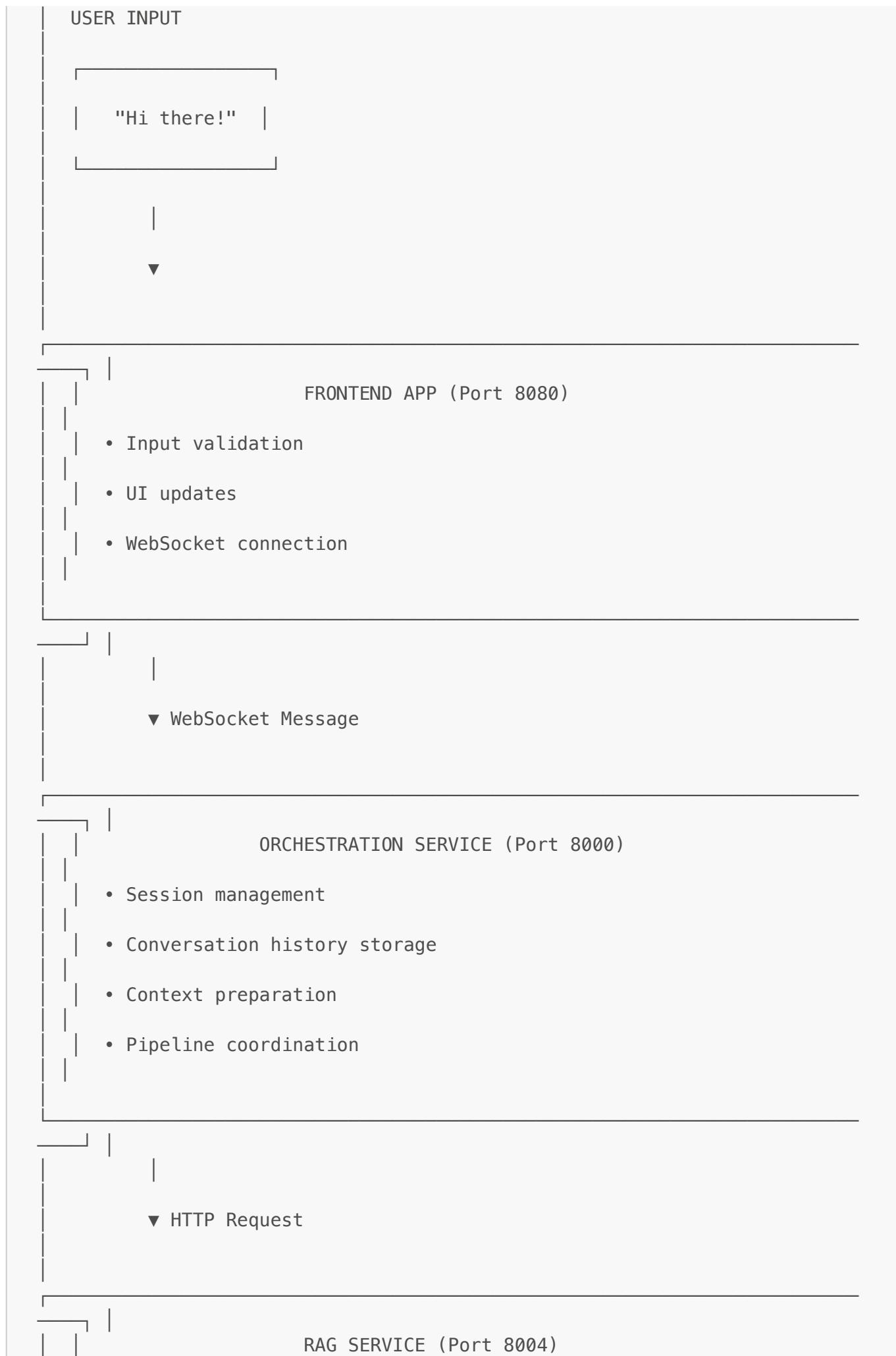
Configuration:

- Vector Size: 1024 (BGE-Large) / 1024 (multilingual-E5)
- Distance Metric: Cosine Similarity
- Index Type: HNSW (Hierarchical Navigable Small World)
- Storage: Persistent (Docker volume)

Technologies: Qdrant, Rust, HTTP API, gRPC API

Complete End-to-End Data Flow

COMPLETE END-TO-END DATA FLOW



- BGE-Large-EN-v1.5 embedding generation
- Qdrant vector search
- BGE-Reranker-Large context reranking
- Relevant context retrieval

▼ HTTP Request

LLM SERVICE (Port 8002)

- LLaMA-3-8B-Instruct (AWQ quantized)
- vLLM high-throughput inference
- Real-time token streaming
- Context-aware response generation

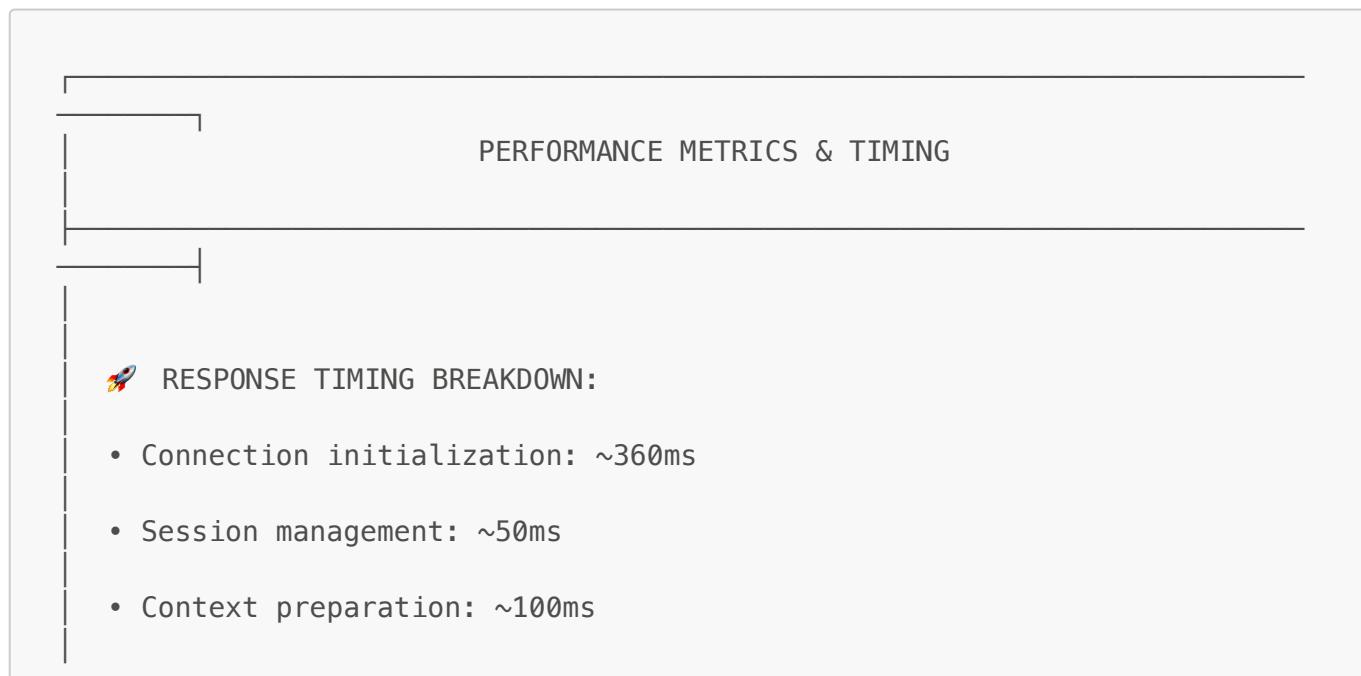
▼ HTTP Request

TTS SERVICE (Port 8003)

- MeloTTS neural synthesis
- 22,050 Hz, 64 kbps, WAV format
- 100ms audio chunks
- Real-time streaming



📊 Performance Metrics & Timing



- RAG retrieval: ~200ms
- LLM generation: ~2,000ms (streaming)
- TTS synthesis: ~1,500ms (streaming)
- Audio playback: ~4,000ms (100+ chunks)
- Total end-to-end: ~4,600ms

✓ THROUGHPUT CAPABILITIES:

- LLM: 50+ tokens/second (vLLM optimized)
- TTS: 100+ chunks/second (MeloTTS streaming)
- RAG: 1000+ queries/second (Qdrant vector search)
- WebSocket: 1000+ messages/second (real-time)

🔧 RESOURCE UTILIZATION:

- GPU Memory: ~8GB (LLM + TTS)
- CPU Usage: ~40% (orchestration + RAG)
- RAM Usage: ~16GB (all services)
- Network: ~1MB/s (audio streaming)

🛠️ Technology Stack Summary

TECHNOLOGY STACK SUMMARY

FRONTEND:

- HTML5, JavaScript, WebSocket API, WebRTC API

- Real-time UI updates, audio playback, connection management

BACKEND SERVICES:

- Python, FastAPI, AsyncIO, httpx
- Docker containerization, health monitoring

AI MODELS:

- LLaMA-3-8B-Instruct (AWQ quantized) – LLM
- BGE-Large-EN-v1.5 – English embeddings
- multilingual-E5-Large – Multilingual embeddings
- BGE-Reranker-Large – Context reranking
- MeloTTS – Neural text-to-speech

INFRASTRUCTURE:

- vLLM – High-throughput LLM inference
- Qdrant – Vector database
- Docker Compose – Service orchestration
- GPU acceleration (CUDA)

COMMUNICATION:

- WebSocket – Real-time bidirectional communication
- HTTP/REST – Service-to-service communication
- WebRTC – Ultra-low latency voice communication

This end-to-end workflow diagram shows the complete journey from user text input to audio output, including all service names, models used, communication protocols, and performance metrics. The system is designed for production-grade conversational AI with real-time streaming and high-quality audio output.

