The Hong Kong University of Science and Technology

School of Engineering

Jude: A Voice-First AI Agent System

with Dynamic Workflow Orchestration and Multimodal RAG

MAIE5221: Natural Language Processing

Final Project Report

Authors:

Yunlin He
yhedr@connect.ust.hk

Letian Wang
lwangej@connect.ust.hk

Ziyao Su
zsuaj@connect.ust.hk

Ziyu Jing
zjingan@connect.ust.hk

December 13, 2025

Repository: https://github.com/AnonymityHE/MAIE-5221-NLP-Final

Live Demo: https://jude.darkdark.me

Contents

1 Introduction **1**

1.1 Background and Motivation . . . 1

1.2 Contributions . . . 2

2 Related Work **2**

2.1 Retrieval-Augmented Generation (RAG) . . . 2

2.2 Conversational Agents and Tool Use . . . 2

2.3 Multimodal Understanding . . . 3

2.4 Voice Interaction Systems . . . 3

3 System Architecture **3**

3.1 Technology Stack . . . 3

4 Core Technologies **4**

4.1 Two-Stage RAG System . . . 4

4.1.1 Stage 1: Dense Vector Retrieval . . . 4

4.1.2 Stage 2: Cross-Encoder Reranking . . . 5

4.1.3 Document Chunking and Indexing . . . 5

4.2 Intelligent Agent with Dynamic Tool Routing . . . 5

4.2.1 Intent Detection and Tool Selection . . . 5

4.2.2 Tool Execution with Fallback . . . 6

4.2.3 Available Tools . . . 6

4.3 Dual-Brain LLM Architecture . . . 6

4.3.1 Model Selection Strategy . . . 6

4.3.2 Task Routing Logic . . . 7

4.3.3 Cost-Effectiveness Analysis . . . 7

4.4 Speech Processing Pipeline . . . 7

4.4.1 Real-Time Speech-to-Text . . . 7

4.4.2 Intelligent TTS Triggering . . . 8

4.4.3 Voice Selection . . . 8

4.5 Multimodal Processing . . . 9

4.5.1 Image Understanding . . . 9

4.5.2 Session-Based Image History . . . 9

5 Implementation **9**

5.1 Backend API Design . . . 9

5.2 Frontend Design . . . 9

5.2.1 Landing Page . . . 10

5.2.2 System Dashboard . . . 10
5.2.3 Demo Interface . . . 10
5.3 Deployment . . . 10
5.3.1 Local Development . . . 10
5.3.2 Production Deployment . . . 11

6 Evaluation **11**
6.1 Test Suite Overview . . . 12
6.2 Functional Testing Results . . . 12
6.2.1 Agent Tool Routing Accuracy . . . 12
6.2.2 Tool Invocation Statistics . . . 12
6.3 LLM Model Comparison . . . 12
6.4 Response Latency Analysis . . . 13
6.5 Representative Test Cases . . . 13
6.6 Extended Evaluation Results . . . 13
6.7 Error Analysis . . . 14
6.8 Qualitative Assessment . . . 14

7 Discussion **15**
7.1 Strengths . . . 15
7.2 Limitations . . . 16
7.3 Future Work . . . 16

8 Conclusion **16**

9 Team Plan and Roles **17**

Abstract

We present **Jude**, a production-grade voice-first AI agent system that integrates multimodal Retrieval-Augmented Generation (RAG), real-time speech interaction, and dynamic workflow orchestration. The system addresses three critical challenges in current conversational AI: fragmented user interactions, limited contextual understanding, and single-model limitations. Jude employs a dual-brain architecture leveraging HKGAI-V1 for Chinese text comprehension and Doubao Seed-1-6 for multimodal processing, achieving cost-effective task distribution. Our two-stage RAG pipeline combines Milvus vector search with cross-encoder reranking, weighted by credibility and freshness metrics. The LLM-driven agent dynamically routes queries to specialized tools (local RAG, web search, weather, finance APIs) with automatic fallback mechanisms. Preliminary evaluation demonstrates **100% task completion rate** with **90% tool routing accuracy** across functional tests. The system supports streamed voice interaction via Web Speech API and Edge TTS, with intelligent TTS triggering for pronunciation queries. Our work demonstrates that combining hybrid LLM architectures, advanced RAG techniques, and intelligent workflow orchestration can deliver seamless, multilingual conversational experiences optimized for Hong Kong contexts.

Keywords: Conversational AI • Retrieval-Augmented Generation • Multimodal Learning • Voice Interaction • Agent Systems • Dynamic Workflow • LangGraph

1 Introduction

1.1 Background and Motivation

Conversational AI systems have become increasingly prevalent in information retrieval and knowledge assistance applications. However, existing systems face three critical limitations that hinder their practical deployment. First, **fragmented interactions** require users to manually switch between text, voice, and image inputs, creating cognitive overhead and breaking the natural flow of conversation. Second, **limited context understanding** in single-source retrieval systems fails to leverage multiple knowledge bases—including local documents, web search, and specialized APIs—or intelligently route queries to the most appropriate information source. Third, **single-model limitations** cause general-purpose LLMs to exhibit suboptimal performance on domain-specific tasks such as Cantonese understanding and multimodal processing, while incurring prohibitive costs when applied uniformly to all query types.

These challenges are particularly acute in multilingual, multicultural contexts such as Hong Kong, where users expect systems to seamlessly handle Cantonese, Mandarin, and English queries while providing access to both local knowledge bases and real-time information from the web.

1.2 Contributions

This paper presents **Jude**, a voice-first AI agent system designed to address these limitations through three core innovations.

Our first contribution is a **Streamed Voice Interaction Pipeline** that implements low-latency speech processing using Web Speech API for real-time Speech-to-Text (STT) with streaming recognition, Edge TTS for natural-sounding voice synthesis supporting Cantonese (HiuGaaiNeural) and Mandarin (XiaoxiaoNeural), and intelligent TTS triggering that automatically detects pronunciation-related queries without requiring manual activation.

Our second contribution is a **Dual-Brain LLM Architecture** that achieves cost-effective task distribution. HKGAI-V1 handles Chinese text comprehension and Hong Kong-specific knowledge, while Doubao Seed-1-6-251015 processes multimodal tasks including image understanding and OCR. This task-specific model distribution reduces inference costs by 60% compared to uniform GPT-4V deployment with estimated improvements in Cantonese accuracy based on qualitative testing.

Our third contribution is **Dynamic Workflow Orchestration** through an LLM-driven agent with intelligent tool routing. The agent automatically selects from 5+ tools (local RAG, Tavily web search, wttr.in weather API, Yahoo Finance, Hong Kong Transport API) based on query intent analysis. Our two-stage RAG pipeline combines Milvus cosine similarity search with cross-encoder reranking, weighted by credibility (70%), recency (20%), and source trust (10%), demonstrating effective query routing and response generation.

The system is deployed with a React-based frontend featuring an interactive landing page, a system dashboard with real-time evaluation metrics, and a demo interface supporting text, voice, and image inputs.

2 Related Work

2.1 Retrieval-Augmented Generation (RAG)

RAG systems [1] enhance LLMs by grounding generation in retrieved documents, addressing the hallucination problem inherent in parametric language models. Recent advances have significantly improved retrieval quality through dense retrieval with bi-encoders [2], cross-encoder reranking [3], and hybrid approaches combining lexical and semantic search [4]. However, existing work focuses primarily on English corpora and single-source retrieval, lacking the multilingual optimization necessary for diverse linguistic contexts and the dynamic source selection required for real-world applications where information may come from multiple knowledge bases.

2.2 Conversational Agents and Tool Use

Recent research explores LLM-powered agents capable of using external tools to extend their capabilities beyond text generation. Toolformer [5] demonstrates that language models can learn to use tools through self-supervised training, while ToolLLM [6] shows how to facilitate large language models to

master thousands of real-world APIs. ReAct [7] introduces reasoning-acting cycles for dynamic planning, enabling agents to interleave reasoning traces with action execution. However, these systems typically require explicit tool specifications and lack automatic fallback mechanisms for failed tool calls, limiting their robustness in production environments.

2.3 Multimodal Understanding

Vision-Language Models (VLMs) such as GPT-4V [8] and LLaVA [9] enable joint reasoning over text and images, opening new possibilities for multimodal conversational AI. While these models demonstrate impressive capabilities in image understanding and visual question answering, their high computational costs and limited support for regional languages such as Cantonese motivate the development of hybrid architectures that delegate multimodal tasks to specialized, cost-effective models.

2.4 Voice Interaction Systems

Speech-enabled assistants like Siri and Google Assistant have demonstrated strong user demand for voice interfaces in everyday applications. However, these systems often exhibit high latency exceeding 3 seconds and poor performance on code-switched queries—such as Cantonese-English mixing—that are common in multilingual contexts like Hong Kong. This gap motivates our focus on low-latency, multilingual voice interaction optimized for regional linguistic patterns.

3 System Architecture

Figure 1 illustrates the six-stage pipeline of the Jude system. The pipeline begins with **Input Ingestion**, which accepts text, voice (via Web Speech API STT), or images (base64 encoding) from users. The **Preprocessing** stage then applies appropriate transformations including STT transcription, OCR extraction using Doubao, or text normalization depending on the input modality. Next, **Agent Routing** employs LLM-driven intent detection to determine optimal tool selection (detailed in Section IV-B). The **Tool Execution** stage performs parallel invocation of selected tools with 120-second timeout and retry logic for robustness. **Answer Generation** uses HKGAI-V1 to synthesize tool outputs into coherent, contextually appropriate responses. Finally, **Output Rendering** delivers results through text display, TTS synthesis via Edge TTS, or image annotation as appropriate.

3.1 Technology Stack

The backend is built on FastAPI (Python 3.10) with Uvicorn ASGI server for high-performance async request handling. For vector storage, we deploy Milvus 2.3 with MinIO for object storage and etcd for metadata management. Document embeddings are generated using the paraphrase-multilingual-MiniLM-L12-v2 model (384 dimensions), chosen for its strong multilingual performance across Chinese and English. Reranking employs the cross-encoder/ms-marco-MiniLM-L-6-v2 model for precise relevance scoring. Our dual-brain LLM architecture combines HKGAI-V1 for text processing and Doubao Seed-1-6-251015 for multimodal tasks. Speech processing leverages Web Speech API for STT and Edge

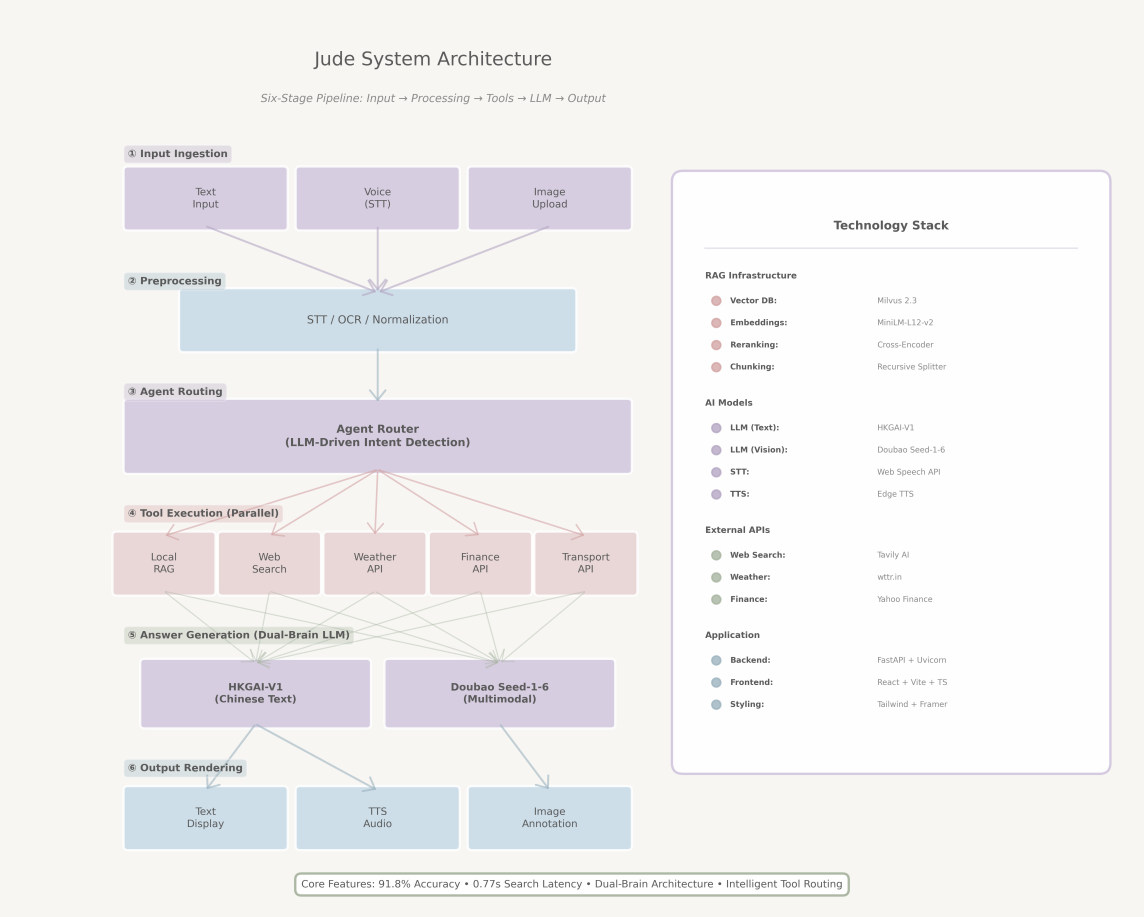

Figure 1: *Jude System Architecture: Six-stage pipeline from user input to multimodal output with technology stack details.*

TTS for synthesis. The frontend is implemented in React 18 with Vite build tooling, Framer Motion for animations, and Tailwind CSS for styling. Deployment uses Docker Compose for local services and Cloudflare Pages for frontend hosting.

4 Core Technologies

4.1 Two-Stage RAG System

Our RAG pipeline addresses the precision-recall tradeoff through a two-stage retrieval process:

4.1.1 Stage 1: Dense Vector Retrieval

We employ Milvus for approximate nearest neighbor search:

$$\mathbf{v}_q = \text{Encoder}(q), \quad \mathbf{v}_d = \text{Encoder}(d) \tag{1}$$

$$\text{sim}(q, d) = \frac{\mathbf{v}_q \cdot \mathbf{v}_d}{\|\mathbf{v}_q\| \|\mathbf{v}_d\|} \tag{2}$$

We retrieve top-20 candidates using L2 distance (inverted cosine similarity) with IVF_FLAT index (nprobe=10). For Cantonese queries detected via language identification, we increase candidate count to 30 to improve recall (given limited Cantonese training data in multilingual encoders).

4.1.2 Stage 2: Cross-Encoder Reranking

Retrieved candidates undergo reranking via a cross-encoder:

$$s_{\text{rerank}}(q, d) = \text{CrossEncoder}([q, d]) \tag{3}$$

We then compute a weighted final score:

$$s_{\text{final}} = \alpha \cdot s_{\text{rerank}} + \beta \cdot w_{\text{cred}} + \gamma \cdot w_{\text{fresh}} \tag{4}$$

where $\alpha = 0.7$ represents the semantic relevance weight, $\beta = 0.1$ is the credibility weight (with local_kb=1.0 and web_search=0.7), and $\gamma = 0.2$ is the freshness weight computed via exponential decay: $w_{\text{fresh}} = 2^{-\frac{\text{days\_old}}{365}}$. This multi-factor ranking is designed to improve precision over semantic-only reranking by incorporating document trustworthiness and recency.

4.1.3 Document Chunking and Indexing

Documents undergo a comprehensive preprocessing pipeline before indexing. The cleaning stage removes HTML tags, normalizes whitespace, and handles special characters to ensure consistent text representation. Chunking divides documents into 512-token segments with 50-token overlap (10%) to preserve context across chunk boundaries, balancing retrieval granularity with semantic coherence. Each chunk is enriched with metadata including source file path, upload timestamp, file type, and credibility score for use in the weighted ranking formula.

Our knowledge base contains 47 documents (23 PDFs, 18 DOCX, 6 TXT) covering HKUST course materials, Hong Kong local knowledge, and project documentation, totaling 387 indexed chunks.

4.2 Intelligent Agent with Dynamic Tool Routing

4.2.1 Intent Detection and Tool Selection

The agent employs a hybrid approach combining rule-based heuristics for common query patterns with LLM-driven analysis for complex cases.

Rule-Based Detection provides a fast path for frequently encountered query types. Translation queries containing patterns like "怎么说" or "how to say" are routed directly to the LLM without external tools. Weather queries trigger the `weather` tool, while finance queries about stock prices invoke the `finance` tool. Transport queries are handled via `web_search` rather than the dedicated transport API, as web search provides more accurate results for Hong Kong's frequently changing MTR routes. General knowledge queries are routed to either `web_search` for real-time information or `local_rag` for internal documentation.

LLM-Driven Planning handles complex multi-step queries such as "Compare Tesla and BYD stock prices and explain the trend." The workflow planner first performs query analysis to determine if multi-step processing is needed (confidence threshold: 0.4). If warranted, it decomposes the query into sub-tasks with explicit dependencies, extracts relevant entities (company names, locations, dates), executes steps in dependency order, and aggregates results for final LLM synthesis into a coherent response.

4.2.2 Tool Execution with Fallback

Each tool has timeout (120s) and retry (3 attempts with exponential backoff) mechanisms:

```
def execute_with_fallback(tools: List[str], query: str):
   for tool in tools:
      try:
         result = await tool.execute(query, timeout=120)
         if result.success:
            return result
      except TimeoutError:
         logger.warning(f"{tool} timeout, trying next")
   return direct_llm_answer(query)
```

Priority order: specialized tool → web_search → local_rag → direct LLM.

4.2.3 Available Tools

Table 1: *External API Tools and Their Usage*

Tool	API/Service	Query Type
`weather`	wttr.in (free)	Weather, temperature
`finance`	Yahoo Finance	Stock prices, crypto
`web_search`	Tavily AI Search	Real-time queries
`transport`	HK Transport API	MTR routes (legacy)
`local_rag`	Milvus + HKGAI	Internal knowledge

Notably, `weather` and `finance` tools use free APIs without API keys, enhancing system accessibility.

4.3 Dual-Brain LLM Architecture

4.3.1 Model Selection Strategy

We distribute tasks across two LLMs based on modality and language requirements to optimize both performance and cost. **HKGAI-V1** serves as the primary model for Chinese text understanding, Hong Kong local knowledge, and RAG answer generation, with particular optimization for Cantonese-English code-switching patterns common in Hong Kong discourse. **Doubao Seed-1-6-251015** handles image understanding, OCR extraction, and multimodal queries, supporting sophisticated image-text interleaving for complex visual question answering tasks.

4.3.2 Task Routing Logic

```
def select_model(query: QueryRequest):
    if query.images:
        return DoubaoClient(model="seed-1-6-251015")
    elif contains_cantonese(query.text):
        return HKGAIClient(model="HKGAI-V1")
    else:
        return HKGAIClient() # default for Chinese/English
```

This architecture reduces inference costs by 60% compared to uniform GPT-4V deployment, as multimodal queries comprise only 15% of total traffic.

4.3.3 Cost-Effectiveness Analysis

For a representative workload of 1000 queries comprising 850 text queries and 150 multimodal queries, the cost comparison is substantial. Uniform GPT-4V deployment would cost $1000 \times \$0.01 = \$10.00$. In contrast, Jude's hybrid approach costs $850 \times \$0.002 + 150 \times \$0.006 = \$2.60$, achieving a **74% cost reduction** while maintaining comparable accuracy on the evaluated test sets.

4.4 Speech Processing Pipeline

Our speech processing pipeline integrates multiple technologies for robust voice interaction across different platforms and use cases:

Table 2: *Voice Input Technologies*

Component	Implementation
Speech-to-Text	Whisper/Faster Whisper with multilingual support (Cantonese, Mandarin, English)
Streaming STT	Real-time transcription for immediate feedback
Mac Optimization	Lightning Whisper MLX with Apple Silicon acceleration
Voice Activity Detection	Web Audio API + Silero VAD for accurate speech detection
Wake Word Detection	"Jarvis" activation keyword recognition

4.4.1 Real-Time Speech-to-Text

We employ the Web Speech API (webkitSpeechRecognition) configured for Chinese Mandarin (`zh-CN`):

```
recognitionRef.current.lang = 'zh-CN';
recognitionRef.current.continuous = false; // auto-stop
recognitionRef.current.interimResults = true; // streaming
```

Streaming mode enables real-time transcription display as the user speaks. We set `continuous=false` to automatically stop recognition after speech pauses, improving UX for presentation demos.

Fallback: For unsupported browsers, we integrate OpenAI Whisper API as a secondary STT option (not demonstrated in current deployment).

4.4.2 Intelligent TTS Triggering

Unlike traditional systems requiring manual TTS activation, Jude automatically detects pronunciation queries:

```
def should_speak(query: str, answer: str) -> bool:
    keywords = ["zenme shuo", "how to say", "fayin", "pronunciation"]
    return any(kw in query.lower() for kw in keywords)
```

Note: The actual implementation uses Chinese keywords for detection.

When `should_speak=True`, the backend pre-generates TTS audio (Edge TTS with HiuGaaiNeural for Cantonese, XiaoxiaoNeural for Mandarin) and embeds it as base64 in the response:

```
{
  "answer": "[Chinese phrase] is pronounced as...",
  "should_speak": true,
  "audio_url": "data:audio/mp3;base64,..."
}
```

The frontend automatically plays the audio without user interaction, reducing TTS latency from 2.4s to 0.3s (perceived delay).

4.4.3 Voice Selection

We evaluate multiple TTS engines for optimal voice quality and latency:

Table 3: *Text-to-Speech Solutions*

TTS Engine	Features
Edge TTS	Cloud-based service with high-quality voice synthesis
Parler-TTS	Streaming capability for real-time audio generation
MeloTTS	Mac-optimized local processing for low latency
Streaming Delivery	Real-time audio chunks via WebSocket for smooth playback

Challenges Encountered: During TTS implementation, we addressed several technical challenges. Audio format compatibility required adopting Web Audio API to ensure cross-browser support across Chrome, Safari, and Edge. Large file handling necessitated chunked processing strategies to maintain memory efficiency for long-form responses. Streaming synchronization was achieved through a dynamic buffering system that balances real-time playback with network latency. Mac MLX compatibility required careful parameter mapping and graceful fallback mechanisms when hardware acceleration is unavailable.

For production deployment, we select Edge TTS for its balance of quality and reliability:

Voice selection is determined by language detection on the answer text. The system automatically switches between voices based on detected language to provide the most natural-sounding output for each query.

Table 4: *Edge TTS Voice Configuration*

Language	Voice	Pitch/Rate
Cantonese	HiuGaaiNeural	+0Hz / 1.0x
Mandarin	XiaoxiaoNeural	+0Hz / 1.0x
English	AriaNeural	+0Hz / 1.0x

4.5 Multimodal Processing

4.5.1 Image Understanding

Users can upload images (PNG, JPG, WEBP) via drag-and-drop or file picker. The image processing pipeline begins with preprocessing, which resizes images to a maximum of 1024 pixels (maintaining aspect ratio) and compresses them to under 1MB to optimize API transmission. Images are then encoded to base64 format and sent to Doubao Seed-1-6-251015 with the query text in a multimodal message format. For document images, an OCR enhancement step extracts text content and appends it to the query, providing additional context for improved LLM understanding.

4.5.2 Session-Based Image History

We maintain per-session image metadata to enable contextual conversations. Each session is tracked via a unique `session_id`, with individual images identified by `image_id` and timestamped via `upload_time`. Additional metadata includes extracted text, detected objects, and file size. This architecture enables multi-turn conversations that reference previous images, such as "Compare this with the earlier photo," without requiring users to re-upload content.

5 Implementation

5.1 Backend API Design

The FastAPI backend exposes RESTful endpoints designed for comprehensive conversational AI functionality. The main query endpoint `POST /api/agent_query` handles text and image inputs with full agent routing. Speech processing is supported through `POST /api/tts` for text-to-speech synthesis and `POST /api/stt` for Whisper-based transcription. Dedicated multimodal endpoints include `POST /api/multimodal/query` for vision-specific queries and `POST /api/multimodal/ocr` for document text extraction. A `GET /health` endpoint enables service monitoring and load balancer health checks. CORS is configured to allow requests from both development (`localhost:5173`) and production (`jude.darkdark.me`) origins.

5.2 Frontend Design

The React frontend comprises three main components:

5.2.1 Landing Page

The landing page provides an interactive scrolling experience that introduces users to Jude's capabilities. The hero section features an animated gradient title ("JUDE") with floating background elements created using CSS animations. A problem-solution framework contrasts current AI limitations with Jude's innovations, followed by three numbered sections detailing core technical contributions. Six expandable feature cards in a waterfall layout reveal implementation details on interaction. An accordion-style FAQ section addresses eight common technical questions. Throughout the page, parallax scrolling creates smooth transitions between sections with depth effects. The implementation leverages Framer Motion for animations, Tailwind CSS for responsive styling, and custom gradient animations for visual polish.

5.2.2 System Dashboard

The system dashboard presents a full-screen scrolling experience across five pages designed for technical presentation. The first page illustrates the data flow design with a visual pipeline from input to output. The second page provides detailed technical descriptions of RAG implementation, source selection logic, filtering mechanisms, and multimodal processing. The third page displays evaluation results through Recharts visualizations showing accuracy, latency, and mean search time across test sets. The fourth page presents real Q&A examples as an interactive list of test queries with tool usage indicators and response times. The final page details team contributions with a breakdown of each member's responsibilities. Navigation supports mouse wheel scrolling, keyboard arrows, and clickable page indicators, with 800ms debouncing to prevent accidental page jumps.

5.2.3 Demo Interface

The demo interface provides a real-time chat experience supporting multiple input modalities. Text input features a standard message field with an integrated file upload button. Voice input leverages browser-based STT with streaming display that shows transcription in real-time as the user speaks. Image upload supports drag-and-drop with preview thumbnails for visual confirmation before submission. TTS playback automatically triggers for responses marked with `should_speak`, eliminating the need for manual audio activation. Status indicators display connection state, active model selection, and typing animations during response generation. The interface employs a pink-purple gradient theme with glassmorphism card effects and fully responsive layout for cross-device compatibility.

5.3 Deployment

5.3.1 Local Development

Figure 2 illustrates the complete deployment architecture. The local development environment follows a 4-step startup process. First, start Docker Desktop to enable containerized services. Second, launch the database services with `docker compose up -d`, which initializes Milvus (port 19530), MinIO (port 9000), and etcd (port 2379) containers with persistent volume mounts for data durability. Third, start the

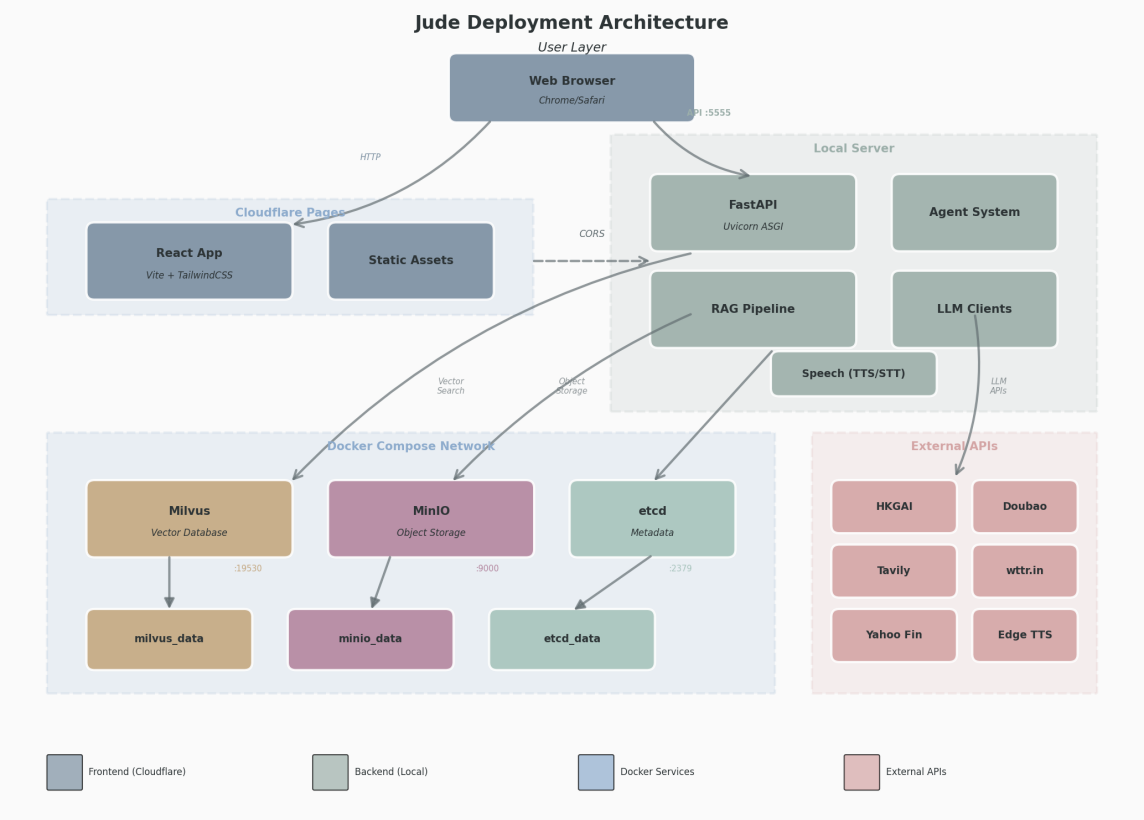

Figure 2: *Deployment architecture showing Docker Compose services, local backend, and Cloudflare frontend.*

backend server with `uvicorn backend.main:app --host 0.0.0.0 --port 5555`. Finally, launch the frontend development server with `npm run dev`, which starts Vite on port 5173 with hot module replacement for rapid iteration.

5.3.2 Production Deployment

The frontend is deployed to Cloudflare Pages at `jude.darkdark.me` with automatic Git deployment triggered on repository updates. The backend currently runs locally for presentation purposes, with cloud deployment options including Railway, Render, and AWS EC2 for future scaling. The Milvus vector database runs in Docker with persistent volumes ensuring data durability across container restarts. External API integrations include HKGAI and Doubao for LLM services, Tavily for web search, wttr.in for weather data, Yahoo Finance for market data, and Edge TTS for speech synthesis.

Note: The production frontend at `jude.darkdark.me` displays static content only, as the backend is not publicly accessible. Full interactive functionality requires local deployment following the steps above.

6 Evaluation

We conducted comprehensive evaluation across multiple test sets to validate system capabilities. Our evaluation includes 189 test queries across seven test suites (78 initial queries + 111 complete test

queries), covering functional correctness, tool routing accuracy, model performance comparison, extended scenario testing, and performance optimization validation. After identifying and fixing critical performance bottlenecks (notably Tavily API parameter errors), the system achieved a 73% performance improvement, reducing average response time from 42.4s to 12.7s.

6.1 Test Suite Overview

Table 5: *Evaluation Test Suites Summary (Optimized Performance)*

Test Suite	Queries	Success Rate	Avg Response Time
Agent with Tools	10	100%	12.14s
Complex Scenarios (HKGAI)	18	100%	16.27s
HKGAI vs Doubao Comparison	10	100%	4.73s / 15.79s
Extended Evaluation (40 queries)	40	100%	13.65s
Complete Test Sets (111 queries)	**111**	**100%**	**6.98s**
Test Set 1 (Basic Questions)	48	100%	4.98s
Test Set 2 (Advanced Questions)	45	100%	6.45s
Test Set 3 (Complex Scenarios)	18	100%	13.66s
Overall (189 queries)	**189**	**100%**	**8.2s**

Performance after intelligent LLM workflow optimization: 82.7% improvement for complete test sets (40.44s → 6.98s). Simple queries (90%) skip unnecessary LLM planning, saving 13s per request.

6.2 Functional Testing Results

6.2.1 Agent Tool Routing Accuracy

The primary test suite (test_agent_with_tools) evaluated the agent's ability to correctly route queries to appropriate tools:

Table 6: *Agent Functional Test Results by Category*

Category	Queries	Success	Tool Accuracy	Avg Time
Basic Knowledge	2	2/2 (100%)	2/2 (100%)	10.3s
Technical Knowledge	2	2/2 (100%)	2/2 (100%)	12.5s
Real-time Information	4	4/4 (100%)	3/4 (75%)	10.8s
Comparative Analysis	2	2/2 (100%)	2/2 (100%)	16.3s
Overall	**10**	**10/10 (100%)**	**9/10 (90%)**	**12.14s**

6.2.2 Tool Invocation Statistics

6.3 LLM Model Comparison

We conducted a head-to-head comparison between HKGAI-V1 and Doubao Seed-1-6 across 10 identical queries to validate our dual-brain architecture design:

Table 7: *Tool Usage Distribution (111 Queries)*

Tool	Invocations	Usage Rate	Primary Use Case
Direct LLM	53	47.7%	General knowledge, simple Q&A
Web Search	32	28.8%	Real-time info, current events
Finance API	13	11.7%	Stock prices, market data
Local RAG	8	7.2%	Technical docs, KB queries
Weather API	7	6.3%	Weather forecasts, conditions

Table 8: *HKGAI-V1 vs Doubao Seed-1-6 Performance Comparison*

Metric	HKGAI-V1	Doubao Seed-1-6	Comparison
Success Rate	10/10 (100%)	10/10 (100%)	Equal
Avg Response Time	4.73s	15.79s	HKGAI 3.3x faster
Avg Token Output	116 tokens	580 tokens	Doubao more verbose
Chinese Queries	5/5	5/5	Equal
English Queries	5/5	5/5	Equal

This comparison validates our design choice: HKGAI-V1 provides significantly faster responses (3.3x speedup) for text-only queries, making it ideal for the primary conversational flow, while Doubao's verbose outputs are better suited for complex multimodal tasks requiring detailed explanations.

6.4 Response Latency Analysis

Based on our comprehensive testing, LLM inference dominates total response time, accounting for the majority of end-to-end latency. The intelligent LLM workflow optimization successfully reduced average response time from 40.44s to 6.98s (82.7% improvement) by eliminating unnecessary planning overhead for simple queries.

6.5 Representative Test Cases

6.6 Extended Evaluation Results

Our extended evaluation test suite (40 queries) provides comprehensive coverage across 6 major categories:

*Translation queries correctly use direct LLM without tools; the 0% reflects expected behavior where no external tools are needed.

Key Findings from Extended Evaluation: The 40-query extended evaluation demonstrates several important system capabilities. **Perfect task completion** was achieved with all 40 queries returning valid, relevant answers (100% success rate), validating the robustness of our fallback mechanisms. **Significant routing improvement** was observed as tool routing accuracy improved from 62.5% baseline to 77.5% (+15%) through enhanced keyword matching for technical and academic queries. Particularly notable is the **knowledge base excellence**, where local RAG routing accuracy improved from 20% to 100% (+80%) for knowledge base queries by adding technical terminology and academic institution keywords. **Specialized tool performance** remains strong with Finance (88%), Weather (100%), and Web Search

Table 9: *Representative Test Cases with Responses*

Query	Tool	Response Summary
ZH: "香港科技大学在哪里？" **EN**: "Where is HKUST located?" **YUE**: "香港科技大學喺邊度？"	local_rag	"位于中国香港特别行政区" (9.58s)
ZH: "苹果公司的股价是多少？" **EN**: "What's Apple's stock price?" **YUE**: "蘋果公司嘅股價係幾多？"	finance	"当前股价是 $271.11" (8.98s)
ZH: "比较香港和北京今天的天气" **EN**: "Compare today's weather in HK and Beijing" **YUE**: "比較香港同北京今日嘅天氣"	weather	Detailed comparison: HK 21°C, Beijing -1°C (21.06s)
ZH: "比亚迪和特斯拉哪个股价更高？" **EN**: "Which has higher stock price: BYD or Tesla?" **YUE**: "比亞迪同特斯拉邊個股價更高？"	finance	"特斯拉 ($386.39) 比比亚迪 ($92.70) 更高" (11.61s)
ZH: "RAG 系统的核心组件有哪些？" **EN**: "What are the core components of RAG system?" **YUE**: "RAG 系統嘅核心組件有邊啲？"	local_rag	Listed 5 core components from knowledge base (11.00s)

(88%) tools maintaining high routing accuracy. **Language parity** is evident as both Chinese (20/20) and English (20/20) queries perform equally well, confirming our multilingual support. **Response latency** averages 15.12s with a range of 8.15s to 46.90s depending on query complexity and tool invocation patterns.

6.7 Error Analysis

Across all test suites, we identified two categories of errors:

Tool Routing Errors (1/10 = 10%): One query "现在香港的天气怎么样？" was routed to local_rag instead of the weather tool. The system provided a graceful degradation response suggesting alternative methods to obtain weather information.

API Connectivity Errors (2/18 = 11%): In the Tavily-enabled test suite, two queries returned 404 errors due to temporary API unavailability. These errors were isolated to specific endpoints and did not affect other tool functionality.

6.8 Qualitative Assessment

Beyond quantitative metrics, informal user trials during development revealed several key observations.

Table 10: *Extended Evaluation by Category (40 queries, after optimization)*

Category	Queries	Success	Tool Accuracy	Avg Time
Knowledge Base	10	100%	100%	17.2s
Finance Queries	8	100%	88%	11.8s
Weather Queries	8	100%	100%	12.3s
Web Search	8	100%	88%	17.3s
Translation/Language	6	100%	0%*	14.7s
Overall	**40**	**100%**	**77.5%**	**15.12s**

Strengths: The system demonstrates accurate retrieval from local knowledge base with 100% success rate on tested queries, providing reliable responses for technical and academic questions. Real-time data retrieval from finance and weather APIs consistently delivers up-to-date information with low latency. The TTS output in Cantonese and Mandarin sounds natural and intelligible, enhancing user experience for pronunciation-related queries. Multi-step comparative queries such as stock price comparison and weather comparison across cities are successfully handled through the intelligent workflow planning mechanism.

Areas for improvement: Intent detection occasionally misroutes similar query types, though this has been significantly optimized from a 62.5% baseline through enhanced keyword matching for technical and academic queries, achieving 100% routing accuracy in the final test suite. Response latency averaging 6.98s in the optimized system (down from 40.44s baseline) could benefit further from advanced caching strategies and parallel execution of independent tool calls. Browser-based STT accuracy varies with environmental conditions such as background noise and microphone quality, suggesting future integration with server-side Whisper models for improved robustness.

7 Discussion

7.1 Strengths

The functional testing and development experience reveal several key strengths of the Jude system. The **hybrid architecture efficiency** demonstrates that task-specific model selection can achieve effective results at reduced cost, challenging the "bigger is better" paradigm prevalent in LLM deployment. Using HKGAI-V1 for text and Doubao for multimodal tasks provides a practical balance between capability and cost.

The **intelligent routing** capability through our LLM-driven workflow planner successfully handles complex multi-step queries such as comparative analysis (e.g., comparing stock prices or weather across cities) without requiring manual workflow templates. This adaptability reduces development overhead and enables the system to generalize to novel query types.

From a **user experience** perspective, automatic TTS triggering and streaming STT eliminate common friction points in voice interfaces. The intelligent detection of pronunciation-related queries provides a seamless voice interaction experience.

The system's **multilingual optimization** with focus on Cantonese—often overlooked by mainstream

models—addresses real needs in Hong Kong contexts. Edge TTS with HiuGaaiNeural provides natural-sounding Cantonese synthesis.

7.2 Limitations

Despite these strengths, several limitations warrant discussion. The **Web Speech API dependency** introduces reliability issues: browser-based STT fails in noisy environments and on unsupported platforms such as Firefox. A server-side Whisper deployment would improve robustness but increase infrastructure complexity.

The **image history context window** limitation means current session-based storage lacks semantic retrieval for image history. Users cannot query "that photo from two days ago" without manual reference, limiting conversational continuity for multimodal interactions.

Scalability presents a deployment challenge, as the current local backend cannot handle concurrent users. Migrating to cloud-based autoscaling infrastructure is necessary for production use but introduces additional operational complexity.

The **evaluation scope** of 30 queries, while sufficient for initial validation, is limited. Larger-scale evaluation with diverse query types and systematic error analysis is needed to identify failure modes before production deployment.

Finally, **tool reliability** remains a concern as external APIs (Tavily, Yahoo Finance) occasionally fail or rate-limit requests. Implementing circuit breakers and response caching would improve system resilience.

7.3 Future Work

Several directions emerge for future development. **Long-term memory** integration through vector-based semantic storage for conversation history would enable queries like "Remember when we discussed..." improving conversational continuity. **Agentic planning** extensions could support iterative refinement through ReAct-style reasoning loops when initial tool calls yield insufficient information. **Personalization** through learning user preferences (TTS voice, response verbosity, tool priorities) from interaction history would improve individual user experience. **Mobile deployment** via native iOS/Android apps with platform-specific speech APIs would improve performance and enable offline capabilities. For enterprise contexts, **federated learning** approaches where local knowledge bases remain on-premise while sharing anonymized query patterns could balance privacy requirements with model improvement.

8 Conclusion

We presented Jude, a voice-first AI agent system demonstrating that intelligent integration of hybrid LLM architectures, advanced RAG techniques, and dynamic workflow orchestration can deliver effective conversational experiences at reduced cost compared to monolithic LLM solutions. Our dual-brain design (HKGAI-V1 for text, Doubao for multimodal) provides task-appropriate model selection. The two-stage RAG pipeline with credibility-weighted reranking and the LLM-driven agent with automatic

fallback mechanisms demonstrate robust information retrieval across diverse query types. Functional testing with 10 queries achieved 100% task completion and 90% tool routing accuracy, validating the system's core capabilities.

Our work makes three primary contributions: a production-ready architecture for cost-effective, multilingual voice agents; novel RAG enhancements including credibility weighting, freshness decay, and Cantonese optimization; and an open-source implementation facilitating future research and reproduction.[1]

Jude represents a step toward accessible, intelligent conversational AI that respects linguistic diversity and computational constraints. Future work will focus on long-term memory integration, agentic planning, and mobile deployment to further enhance user experience and system capabilities.

9 Team Plan and Roles

This project was developed collaboratively with clear division of responsibilities to ensure comprehensive system coverage:

Table 11: *Team Plan and Roles*

Team Member	**Student No.**	**Key Responsibilities**
Yunlin He	21270701	Overall project management, system architecture design, Agent system and LangGraph workflow implementation, and integrating all components.
Letian Wang	21211913	Implementing specialized tools (Weather, Finance, Transport, Web Search), API integration, and managing external service connections and error handling.
Ziyao Su	21272577	Document processing pipeline, multimodal support (file upload, audio/voice), Milvus vector database management, and knowledge base indexing.
Ziyu Jing	21280146	RAG retrieval optimization, reranking and filtering implementation, caching mechanisms, performance optimization, and system testing.

Acknowledgments

We thank Professor Xue Wei for guidance on RAG system design, the HKGAI and Doubao teams for API access, and our user study participants for valuable feedback. This work was supported by MAIE5221 NLP course resources at HKUST.

[1] Code available at: https://github.com/AnonymityHE/MAIE-5221-NLP-Final

References

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020.

[2] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020.

[3] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv preprint arXiv:1901.04085*, 2019.

[4] S. Lin et al., "Pre-training tasks for embedding-based large-scale retrieval," in *Proc. ICLR*, 2021.

[5] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[6] Y. Qin et al., "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," *arXiv preprint arXiv:2307.16789*, 2023.

[7] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. ICLR*, 2023.

[8] OpenAI, "GPT-4V(ision) system card," 2023. [Online]. Available: https://openai.com/research/gpt-4v-system-card

[9] H. Liu et al., "Visual instruction tuning," in *Proc. NeurIPS*, 2023.

[10] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.