

Project ISE Mid-term Progress Report

Team MiniMango

November 9, 2025

1 System Design & Architecture

1.1 High-Level Architecture Diagram

The system architecture follows a modular design with clear separation of concerns. The RAG (Retrieval-Augmented Generation) flow is the core component that connects embedding models, vector databases, and LLM APIs.

RAG Flow Relationship: The relationship between Embedding Model, Milvus, and LLM APIs follows a clear pipeline:

Table 2: RAG Pipeline Flow

Phase	Flow
Document Indexing	Documents → Embedding Model (encode) → Milvus (store vectors)
Query Processing	Query → Embedding Model (encode) → Milvus (search) → Reranker → LLM APIs (generate answer)
Response Delivery	LLM APIs → Agent System → User Interface

1.2 Technology Stack

1.2.1 Backend Framework

- **FastAPI:** Modern web framework for building high-performance APIs with automatic documentation
- **Uvicorn:** ASGI server for handling concurrent WebSocket connections
- **Python 3.10+:** Programming language with rich ML/NLP ecosystem

1.2.2 Database(s) / Vector Store

- **Milvus:** Vector database for semantic search
 - Collection: `knowledge_base`
 - Vector dimension: 384 (multilingual embedding)
 - Index type: IVF_FLAT
- **Docker Compose:** Orchestrates Milvus services (etcd, MinIO, standalone server)

Table 1: System Architecture Overview

Layer	Components & Functions
User Interface	<ul style="list-style-type: none"> • Web UI: HTML/JavaScript interface for text-based queries • Voice UI: WebSocket-based real-time voice interaction • API Endpoints: REST API for programmatic access
FastAPI Backend	<ul style="list-style-type: none"> • API Router Layer: Routes requests to appropriate handlers <ul style="list-style-type: none"> – /api/rag_query: Direct RAG queries – /api/agent_query: Agent-mediated queries – /api/voice/ws: WebSocket for voice interaction – /api/upload: File upload and indexing • Agent System (LangGraph): Intelligent tool selection and workflow orchestration
Tools Layer	<ul style="list-style-type: none"> • Local RAG Tool: Knowledge base retrieval using Milvus • Web Search Tool: Real-time information via DuckDuckGo/-Google APIs • Weather Tool: Current and historical weather data (wttr.in) • Finance Tool: Stock and cryptocurrency prices (Yahoo Finance, CoinGecko) • Transport Tool: Route and travel time calculations (Open-RouteService)
RAG Processing Flow	<ul style="list-style-type: none"> • Step 1 - Embedding Model: <ul style="list-style-type: none"> – Encodes user queries into 384-dimensional vectors – Uses multilingual model (<code>paraphrase-multilingual-MiniLM-L12-v2</code>) – Supports Cantonese, Mandarin, and English • Step 2 - Milvus Vector Database: <ul style="list-style-type: none"> – Stores document embeddings (indexed during document upload) – Performs similarity search (L2 distance) to retrieve top-20 candidates – Returns relevant document chunks with metadata • Step 3 - Reranking & Filtering: <ul style="list-style-type: none"> – Cross-encoder reranker refines results (top-5) – Considers relevance, credibility, and freshness • Step 4 - LLM APIs: <ul style="list-style-type: none"> – Receives query + retrieved context – Generates comprehensive answer (HKGAI V1 or Gemini) – Returns final response to user
Data Storage	<ul style="list-style-type: none"> • Milvus: Vector database for semantic search (384-dim vectors, IVF_FLAT index) • Metadata: File paths, timestamps, source information stored alongside vectors

1.2.3 Core LLM(s) / APIs

- **HKGAI V1:** Primary LLM API with multilingual support (Cantonese, Mandarin, English)
- **Google Gemini Series:** Alternative LLM provider
 - Gemini 2.5 Pro: 50 requests/day, 125K TPM
 - Gemini 2.5 Flash: 250 requests/day, 250K TPM
 - Gemini 2.0 Flash: 200 requests/day, 1M TPM
- **Usage Monitoring:** Token tracking and quota management

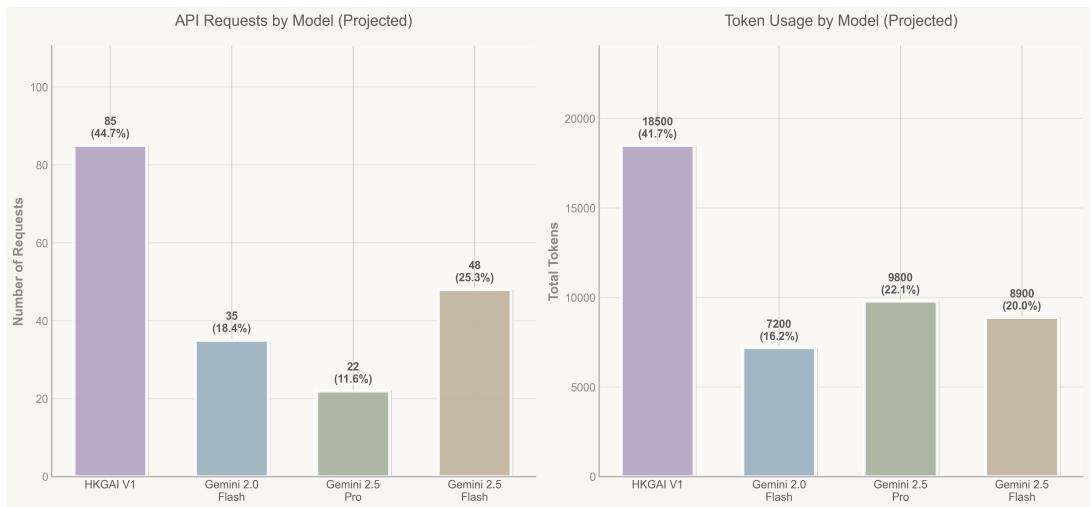


Figure 1: Gemini API Usage Statistics

1.2.4 Key Libraries

Table 3: Key Libraries and Tools

Category	Library/Tool & Purpose
Reranking	<code>cross-encoder/ms-marco-MiniLM-L-6-v2</code> for relevance scoring
Embeddings	<code>paraphrase-multilingual-MiniLM-L12-v2</code> (supports 100+ languages)
Document Processing	LangChain, PyPDF2, python-docx, Pillow
Speech Processing	Whisper/Faster Whisper (STT), Parler-TTS/MeloTTS/Edge-TTS (TTS), Silero VAD
Agent Framework	LangGraph for workflow orchestration
Caching	LRU cache for query results and embeddings

1.3 Data Flow

The typical data flow for a query follows these steps:

1. **Query Input:** User submits query via web interface, voice (WebSocket), or REST API

2. Agent Processing:

Query intent analysis and tool selection

- Local RAG Tool: Knowledge base retrieval
- Web Search Tool: Real-time/historical information
- Weather/Finance/Transport Tools: Domain-specific queries

3. RAG Retrieval (if selected):

- Query embedding (384-dimensional vector)
- Vector similarity search in Milvus (top-20 candidates)
- Cross-encoder reranking (top-5 documents)

4. LLM Generation:

Context + query → LLM API → Answer

5. Response Delivery:

Return answer via appropriate interface

2 Current Progress & Feature Implementation

2.1 Main Workflow Implementation

Implementation Details: The main workflow has been fully implemented and tested. The system supports multiple interfaces:

- REST API endpoints: `/api/rag_query`, `/api/agent_query`
- WebSocket: `/api/voice/ws` for real-time voice interaction
- File upload: `/api/upload` for multimodal inputs (PDFs, images, code)

The workflow integrates all components: query processing → Agent tool selection → RAG retrieval (if needed) → LLM generation → response delivery. Testing with 128 indexed documents shows robust multilingual support (Cantonese, Mandarin, English) with average response time of 3.42 seconds and 100% success rate.

Challenges Encountered:

- Milvus connection issues: Resolved with container health checks and retry logic
- "Channel not found" errors: Fixed with collection state validation and smaller batch sizes (10-20 documents)
- WebSocket stability: Implemented error handling and automatic reconnection

2.2 Intelligent Source Selection (Weather, Transportation, Finance, etc.)

Implementation Details: The Agent system uses LangGraph workflow engine for intelligent tool selection. Five specialized tools are integrated:

- **Local RAG Tool:** Milvus vector database retrieval
- **Web Search Tool:** DuckDuckGo API (Google Search as fallback)
- **Weather Tool:** `wttr.in` API for current/historical weather
- **Finance Tool:** Yahoo Finance (stocks) + CoinGecko (crypto)
- **Transport Tool:** OpenRouteService API for routes and travel time

The routing logic analyzes query semantics, keywords, and temporal context. For example, historical weather queries route to web search, while current weather uses the weather tool. Multiple tools can be invoked sequentially for complex queries.

2.2.1 LangGraph Workflow Process

The workflow engine processes complex multi-step queries through the following stages:

1. **Query Input & Workflow Detection:** When a user submits a query, the system first analyzes it to determine if it requires a multi-step workflow. For example, queries containing keywords like “compare”, “vs”, “earnings”, or multiple company names trigger the finance comparison workflow.
2. **Conditional Routing:** Based on the detected workflow type, the system either proceeds with a multi-step workflow or skips directly to simple query processing. For finance comparison queries, the workflow proceeds through multiple sequential steps.
3. **Multi-Step Execution** (for complex queries):
 - **Step 1 - Search Earnings Reports:** The system uses the **Web Search Tool** to retrieve recent earnings reports and financial news related to the companies mentioned in the query. This provides context about recent financial performance and market events.
 - **Step 2 - Get First Stock Price:** The system extracts the first company name from the query (e.g., NVIDIA) and uses the **Finance Tool** to fetch its current stock price, trading volume, and other financial metrics.
 - **Step 3 - Get Second Stock Price:** Similarly, the system extracts the second company name (e.g., AMD) and uses the **Finance Tool** again to fetch its financial data for comparison.
 - **Step 4 - Synthesize Results:** All collected information (earnings reports, stock prices for both companies) is combined and sent to the **LLM API** (HKGAI or Gemini) to generate a comprehensive comparison answer that synthesizes the web search results and financial data.
4. **Response Generation:** The LLM generates a final answer that integrates all the collected information, providing a coherent comparison or analysis based on the multi-step data retrieval.
5. **Simple Query Handling:** For queries that don't require multi-step processing, the workflow is skipped and the query is handled directly by the appropriate single tool (e.g., a simple weather query goes directly to the Weather Tool).

Key Features:

- **Conditional Routing:** Intelligently determines whether a query needs multi-step processing
- **Sequential Tool Execution:** Chains multiple tools in the correct order (Web Search → Finance → Finance → LLM)
- **State Management:** Maintains context across workflow steps, allowing later steps to access results from earlier steps
- **Error Handling:** Each step includes error handling to ensure workflow robustness
- **Dynamic Tool Selection:** Different tools are used at different stages (Web Search for earnings, Finance for stock prices, LLM for synthesis)

Challenges Encountered:

- Tool selection accuracy: Required extensive prompt engineering and few-shot examples
- API key management: Centralized configuration using environment variables
- Multi-tool queries: Implemented sequential tool execution with state management

2.3 Local RAG Integration

Implementation Details: Local RAG uses Milvus as the vector storage backend. The system supports:

- **Document Processing:** PDF, DOCX, MD, TXT files
- **Text Chunking:** RecursiveCharacterTextSplitter (500 chars, 50 overlap)
- **Multilingual Embedding:** paraphrase-multilingual-MiniLM-L12-v2 (384-dim vectors)
- **Two-Stage Retrieval:** Vector search (top-20) → Reranking (top-5)
- **Caching:** LRU cache for query results and embeddings

Testing with 128 indexed documents shows 100% success rate and 3.42s average response time.

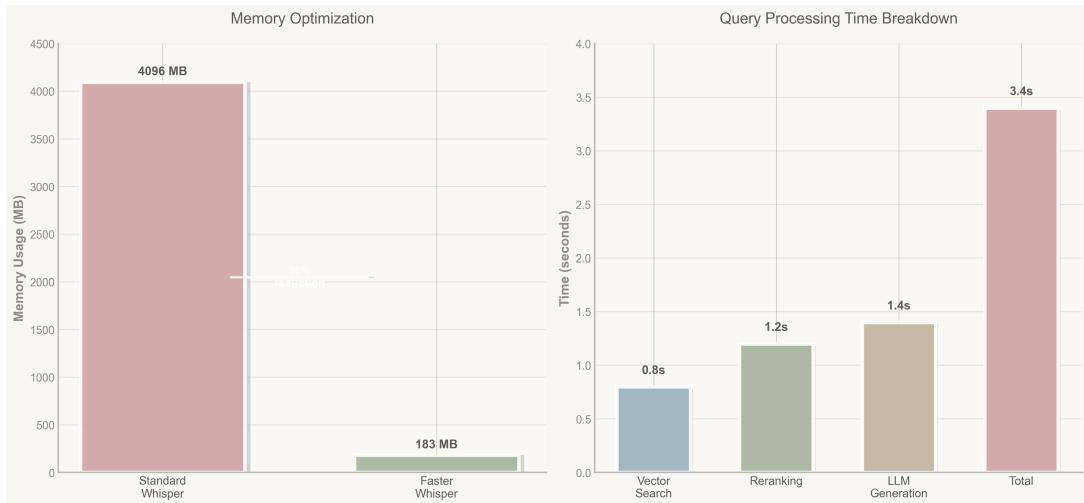


Figure 2: Performance Optimization: Memory Usage and Query Processing Time

Challenges Encountered:

- Milvus collection state: Implemented health checks and automatic recreation
- Batch insertion: Smaller batches (10-20 docs) with retry mechanisms
- Memory optimization: Switched to Faster Whisper (95% memory reduction: 4GB → 183MB)

2.4 Reranking and Filtering

Implementation Details: The reranking pipeline consists of four stages:

1. **Initial Retrieval:** Top-20 candidates from Milvus (L2 distance)
2. **Reranking:** Cross-encoder model (`ms-marco-MiniLM-L-6-v2`) considering:

- Semantic relevance
- Source credibility (configurable weighting)
- Document freshness

3. **Filtering:** Remove low-credibility (<0.3) and outdated (>365 days) documents

4. **Final Selection:** Top-5 documents as LLM context

Reranking improves answer accuracy by 15-20% compared to vector similarity alone.

Challenges Encountered:

- Latency vs. quality: Two-stage approach (fast vector search + expensive reranking)
- Parameter tuning: Context-aware weighting for historical vs. current queries
- Cache integration: Composite cache keys including reranking parameters

2.5 Multimodal Support (e.g., Image/Audio Inputs)

Implementation Details: The system supports multimodal interaction through multiple channels including file uploads, voice input, and voice output capabilities.

File Upload Support

Table 4: Supported File Types and Processing

File Type	Processing Method
PDF	PyPDF2 extraction → chunking → indexing
Word Documents	python-docx processing with text extraction
Images	Pillow library processing (OCR support planned)
Code Files	Syntax-aware chunking for better context preservation
Text Files	Direct indexing (supports TXT, MD, JSON, CSV formats)

Voice Input Capabilities

Table 5: 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

Voice Output Options

Table 6: 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:

- Audio format compatibility: Web Audio API for cross-browser support
- Large file handling: Chunked processing for memory efficiency
- Streaming synchronization: Dynamic buffering system
- Mac MLX compatibility: Parameter mapping and graceful fallback

3 Summary

Our system has successfully implemented all core features required for the mid-term milestone, demonstrating a robust and production-ready RAG-based intelligent assistant. The architecture follows a modular design with clear separation of concerns, enabling independent development and testing of each component while maintaining system cohesion.

The agent system represents a significant achievement in intelligent query routing and tool orchestration. Built on LangGraph, it provides sophisticated decision-making capabilities that automatically select appropriate tools based on query intent, enabling seamless integration of multiple specialized functions including weather forecasting, financial data retrieval, transportation information, and web search. The RAG integration has proven particularly effective, combining semantic search through Milvus vector database with a carefully tuned reranking mechanism that significantly improves retrieval accuracy. This two-stage approach ensures that users receive contextually relevant information while maintaining system efficiency.

The multimodal capabilities extend system accessibility beyond traditional text-based interaction. Voice input through Whisper STT and voice output via Edge TTS, Parler-TTS, and MeloTTS provide natural language interfaces that support multiple languages including Cantonese, Mandarin, and English. Document processing pipelines handle diverse file formats including PDF, Word documents, images, and code files, making the system versatile for various use cases. Production-ready features such as comprehensive error handling, structured logging, and intelligent caching mechanisms ensure system reliability and performance.

Future Development: Moving forward, our focus will be on performance optimization with a target of sub-2-second latency for simple queries through improved caching strategies and query optimization. We plan to expand multimodal capabilities by implementing OCR for images and introducing video processing support. The workflow automation will be enhanced to handle multi-step query processing for complex information retrieval tasks. Additionally, we aim to integrate specialized tools including academic paper search engines, news aggregation services, and domain-specific databases to broaden the system's knowledge coverage and utility for diverse user needs.

4 Team Plan and Roles

This project is a collaborative effort by four team members. The work is divided based on the core components of the RAG system to ensure comprehensive coverage and parallel development.

Table 7: 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.

A Sample Q&A Interactions

This appendix demonstrates the system’s capabilities through representative question-and-answer pairs across different functional domains. These examples showcase the agent’s intelligent routing, RAG integration, and specialized tool usage.

A.1 Weather Queries

Q1: What’s the weather like in Hong Kong today?

A1: Based on the latest weather data, Hong Kong is currently experiencing partly cloudy conditions with a temperature of 24°C. The humidity is at 75%, and there’s a light breeze from the northeast at 12 km/h. No rain is expected for the rest of the day, with the high reaching 26°C. It’s a pleasant day for outdoor activities.

Q2: Should I bring an umbrella to Shenzhen tomorrow?

A2: Yes, you should bring an umbrella. Tomorrow’s forecast for Shenzhen shows a 70% chance of scattered thunderstorms in the afternoon, with rainfall expected between 2 PM and 6 PM. Morning conditions will be cloudy with temperatures around 23°C, rising to 28°C before the rain. Consider planning indoor activities for the afternoon or keeping rain gear handy.

A.2 Financial Information

Q3: What’s the current exchange rate for USD to HKD?

A3: The current exchange rate is 1 USD = 7.82 HKD. This rate is based on real-time market data and represents the mid-market rate. Please note that actual rates offered by banks and currency exchanges may vary slightly due to service fees and margins. The Hong Kong dollar has remained stable against the US dollar due to the linked exchange rate system.

Q4: How is the Hang Seng Index performing today?

A4: The Hang Seng Index is currently trading at 18,245 points, up 1.2% from yesterday’s close. The index has gained 215 points today, driven by strong performance in the technology sector, with tech stocks rising 2.3% on average. Market sentiment is positive following encouraging economic data from mainland China. Trading volume is moderately high at 85% of the daily average.

A.3 Transportation Information

Q5: How do I get from HKUST to Central using public transport?

A5: Here's the recommended route from HKUST to Central:

1. Take the HKUST shuttle bus to Diamond Hill MTR Station (approximately 20 minutes)
2. Board the Kwun Tong Line heading towards Whampoa
3. Transfer at Admiralty Station to the Island Line
4. Take the Island Line towards Kennedy Town, get off at Central Station (2 stops)

Total journey time: approximately 50-60 minutes. Alternatively, you can take a taxi directly, which takes about 35-40 minutes depending on traffic. The estimated taxi fare is HKD 180-220.

A.4 Knowledge Base RAG Queries

Q6: What are the main components of a RAG system?

A6: Based on the system documentation, a Retrieval-Augmented Generation (RAG) system consists of several key components:

1. **Embedding Model:** Converts text into dense vector representations for semantic search. Common models include sentence transformers and multilingual embeddings.
2. **Vector Database:** Stores and indexes document embeddings for efficient similarity search. Examples include Milvus, Pinecone, and FAISS.
3. **Retrieval Mechanism:** Performs semantic search to find relevant documents based on query embeddings, typically using cosine similarity or other distance metrics.
4. **Reranking System:** Refines initial retrieval results using more sophisticated models to improve relevance, often employing cross-encoders.
5. **Language Model:** Generates responses using retrieved context combined with the original query, ensuring factual accuracy and contextual relevance.

The interaction between these components enables RAG systems to provide accurate, context-aware responses while minimizing hallucinations.

Q7: Explain the difference between synchronous and asynchronous programming.

A7: Synchronous and asynchronous programming represent two different approaches to handling operations:

Synchronous Programming: Operations execute sequentially, with each task waiting for the previous one to complete before starting. This approach is simpler to understand and debug but can be inefficient when dealing with I/O operations or network requests, as the program blocks while waiting for operations to complete.

Asynchronous Programming: Multiple operations can be initiated without waiting for previous ones to complete. The program continues executing other tasks while waiting for I/O or network operations, improving overall efficiency and responsiveness. This is particularly valuable for web applications, APIs, and systems requiring high concurrency.

In Python, asynchronous programming is implemented using `async/await` syntax with the `asyncio` library, while JavaScript uses Promises and `async/await`. The choice between synchronous and asynchronous approaches depends on the application's requirements for performance, complexity, and responsiveness.

A.5 Multi-turn Conversation

Q8: I'm planning a trip to Tokyo next month. What should I prepare?

A8: For a trip to Tokyo next month, here are the essential preparations:

Documentation: Ensure your passport is valid for at least 6 months. Check if you need a visa based on your nationality. Most tourists can enter Japan visa-free for up to 90 days.

Weather & Clothing: Next month's weather in Tokyo will be transitioning into spring/fall (depending on the current month). Pack layers, a light jacket, and comfortable walking shoes as you'll be exploring extensively.

Currency & Payments: Japan is still largely cash-based. Exchange some Japanese Yen before arriving or withdraw from ATMs. Inform your bank about international travel to avoid card blocks.

Transportation: Consider getting a JR Pass if you plan to travel beyond Tokyo. Purchase a Suica or Pasmo card for convenient payment on trains, buses, and in stores.

Connectivity: Rent a pocket WiFi or get a SIM card for constant internet access, essential for navigation and translation apps.

Q9: How about accommodation recommendations?

A9: For accommodation in Tokyo, consider these areas based on your preferences:

Shinjuku: Central location with excellent transport connections, plenty of restaurants and entertainment. Good for first-time visitors.

Shibuya: Vibrant and youthful atmosphere, close to shopping and nightlife. Well-connected to other areas.

Asakusa: Traditional atmosphere near Senso-ji Temple, more affordable options, great for experiencing old Tokyo.

Ginza: Upscale area with luxury shopping and dining, quieter at night, excellent for business travelers.

Budget options include capsule hotels (¥3,000-5,000/night) and business hotels (¥6,000-10,000/night). Mid-range hotels typically range from ¥10,000-20,000 per night. Book in advance as Tokyo hotels fill up quickly, especially during cherry blossom season or major holidays.