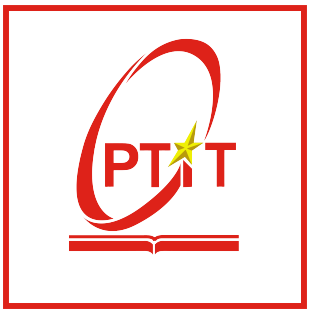
POSTS AND TELECOMMUNICATIONS
INSTITUE OF TECHNOLOGY
INFORMATION TECHNOLOGY



INTERNSHIP

REPORT

WoxionChat

Name: NGUYEN PHAM TRUNG HIEU
Student ID: B23DCCE031
Class ID: B23CQCE04-B
Instructor: KIM NGOC BACH

Hanoi

Table of contents

1 PROJECT OVERVIEW **2**
1.1 Background and Rationale . . . 2
1.2 Research Objectives . . . 2
1.3 Practical Significance . . . 2

2 THEORETICAL FRAMEWORK AND METHODOLOGY **3**
2.1 Retrieval-Augmented Generation (RAG) Architecture . . . 3
2.2 GPT-OSS-20B Fine-tuning Strategy . . . 3
2.3 Markdown Standardization and Voice Technology . . . 4

3 SYSTEM ANALYSIS AND DESIGN **4**
3.1 Data Flow Diagram . . . 4
3.2 Backend Infrastructure and Specialized Database Management . . . 5
3.2.1 Hybrid Backend and Agentic Memory . . . 5
3.2.2 MongoDB as Unified Data Foundation . . . 5
3.3 Security-by-Design Module . . . 5

4 IMPLEMENTATION PLAN AND EVALUATION **6**
4.1 Detailed Roadmap . . . 6
4.2 Success Evaluation Criteria . . . 6

5 CONCLUSION AND FUTURE WORK **6**

Table of contents

1 PROJECT OVERVIEW

1.1 Background and Rationale

In the era of Industrial Revolution 4.0, unstructured data accounts for over 80% of an enterprise's total data volume. These documents include economic contracts, standard operating procedures (SOPs), financial reports, and technical dossiers. Traditional information retrieval based on keyword search faces significant barriers:

- **Information Fragmentation:** Employees lose an average of 20% of their weekly working hours simply searching for necessary information.
- **Contextual Limitations:** Legacy systems cannot understand the relationships between different concepts within the text.

The rise of Large Language Models (LLMs) has provided superior language processing capabilities. However, applying these solutions in the Vietnamese corporate environment presents a critical dilemma regarding **Data Privacy**. When utilizing public cloud APIs:

- **Loss of Data Control:** Sensitive corporate information is transmitted to overseas servers.
- **Leakage Risks:** Data may be unauthorizedly used to retrain third-party models, resulting in a loss of competitive advantage.
- **Cost and Dependency:** API usage costs scale with traffic, and enterprises are completely passive regarding policy changes from providers.

Consequently, the demand for building an **Autonomous AI (Sovereign AI)** system running on private infrastructure (On-premise) is extremely urgent. The **WoxionChat** project was initiated to fundamentally resolve these issues.

1.2 Research Objectives

- To research and implement the RAG (Retrieval-Augmented Generation) architecture to optimize the retrieval of internal data.
- To fine-tune a large open-source model for the Vietnamese language to replace closed-source proprietary solutions.
- To integrate a multimodal document processing workflow, ensuring knowledge integrity across various file formats.

1.3 Practical Significance

The system is not merely a chat tool but a "Centralized Knowledge Brain" for the organization. It helps standardize training processes, provides rapid legal support, and enables hands-free interaction through voice, thereby enhancing overall labor productivity.

2 THEORETICAL FRAMEWORK AND METHODOLOGY

2.1 Retrieval-Augmented Generation (RAG) Architecture

RAG is a sophisticated technique that connects language models with reliable external data sources. The process operates based on three main steps:

1. **Indexing:** Text is converted into vector embeddings and stored in a Vector Database.
2. **Retrieval:** Upon a query, the system searches for text segments with the highest mathematical similarity in a multi-dimensional space.
3. **Generation:** The LLM utilizes the context from retrieved segments to generate accurate responses, mitigating the "hallucination" phenomenon common in LLMs [1].

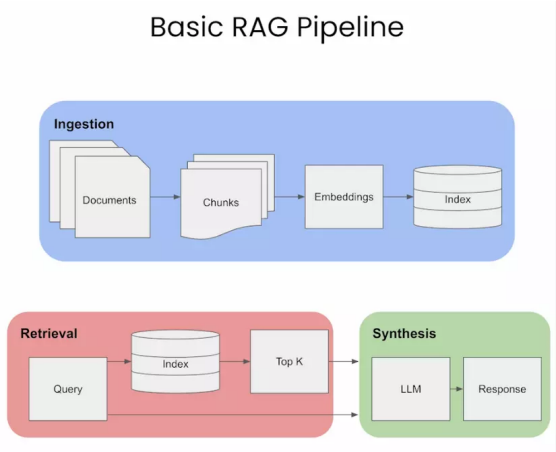


Figure 1: RAG architecture

2.2 GPT-OSS-20B Fine-tuning Strategy

The **gpt-oss-20b** model is one of the most powerful open-source models available. This project applies the **LoRA (Low-Rank Adaptation)** technique to refine this model:

- **Training Data:** Focused on general Vietnamese datasets combined with specialized administrative and corporate data.
- **Benefits:** Fine-tuning allows the model to deeply understand specific Vietnamese sentence structures and technical terminology.
- **Performance:** Model weights are optimized to run smoothly on enterprise-grade GPUs (such as NVIDIA A100 or H100).

Figure 2: gpt-oss

2.3 Markdown Standardization and Voice Technology

- **Microsoft MarkItDown:** A breakthrough in data pre-processing. Instead of extracting raw plain text, this tool preserves the structure of tables and headers in Markdown format.
- **ElevenLabs:** Provides a high-end audio communication layer. ElevenLabs' TTS technology uses neural networks to reproduce Vietnamese speech with full emotional nuances.

Figure 3: MarkItDown

3 SYSTEM ANALYSIS AND DESIGN

3.1 Data Flow Diagram

The WoxionChat processing workflow is designed as a closed-loop pipeline:

1. **Input:** User uploads files (PDF, Excel, Images).
2. **Processing:** MarkItDown converts files to Markdown; Tesseract OCR processes text regions on images.
3. **Storage:** Text is partitioned using Semantic Chunking and pushed to the MongoDB.
4. **Interaction:** User sends queries via WebSocket; Agentic RAG performs retrieval and sends context to the gpt-oss-20b core; results are sent to ElevenLabs for audio generation.

3.2 Backend Infrastructure and Specialized Database Management

The architecture of WoxionChat is optimized for high-performance AI operations, utilizing a hybrid backend approach that combines the strengths of various modern frameworks [7, 12] and specialized database engines [8, 9].

3.2.1 Hybrid Backend and Agentic Memory

- **Main Application Backend (Django):** Built on **Django 5.2** [7], the primary backend manages user authentication, role-based access control (RBAC), and the administrative dashboard. It serves as the secure gateway for the web interface and overall business logic.
- **AI Inference and Agentic RAG Engine (FastAPI):** The core AI intelligence, including the Agentic RAG pipeline, is powered by **FastAPI** [12]. Chosen for its superior asynchronous performance and low overhead, FastAPI handles high-concurrency LLM requests and coordinates the RAG retrieval process.
- **Agentic Memory Management with Redis: Redis** [9] is utilized as a high-speed, in-memory data store to manage the **Long-term and Short-term Memory** of the Agentic RAG. This enables the AI agent to maintain multi-turn conversation context across the Django and FastAPI microservices using frameworks like LangChain [10].
- **Real-time Communication:** Redis also serves as the message broker for **Django Channels**, facilitating seamless WebSocket-based interactions between the client and the backend services.

3.2.2 MongoDB as Unified Data Foundation

To address the complexity of enterprise document intelligence, the system employs **MongoDB** [8] as its primary data backbone:

- **Backend Persistence:** MongoDB stores structured and semi-structured data including user profiles, granular role-based permissions, and comprehensive interaction logs.
- **Agentic RAG Knowledge Base:** MongoDB acts as the document store for the RAG engine. Its document-oriented nature allows for the storage of complex Markdown outputs from MarkItDown alongside rich metadata, facilitating flexible and high-speed retrieval of document chunks.
- **Unified Schema Flexibility:** The schema-less nature of MongoDB enables the system to adapt to various document types without requiring frequent database migrations.

3.3 Security-by-Design Module

To ensure maximum security, the system applies the following principles:

- **Isolation:** The entire inference process takes place on a Local Server.
- **Audit Log:** Records all interactions to facilitate post-event auditing.
- **Encrypted Storage:** Data in the Vector Database and MongoDB are fully encrypted.

4 IMPLEMENTATION PLAN AND EVALUATION

4.1 Detailed Roadmap

Table 1: Detailed Project Implementation Roadmap by Week

Phase	Key Tasks	Duration
Survey & Design	Hardware infrastructure analysis; Database Schema design.	2 weeks
Backend Development	API construction with Django; Redis and Celery integration.	1 weeks
Model Training	Vietnamese data collection; LoRA Fine-tuning implementation.	1 weeks
RAG & OCR Integration	Deployment of MarkItDown and Vector retrieval system.	1 weeks
UI & Voice Integration	Frontend finalization; ElevenLabs integration.	1 weeks
Optimization & Acceptance	Accuracy measurement and latency evaluation.	1 weeks

4.2 Success Evaluation Criteria

- **Response Quality:** Achieve a score above 85% in Groundedness tests.
- **Vietnamese Understanding:** The fine-tuned model must handle complex, ambiguous queries and achieve above 50% at VMLU benchmark [13]
- **Stability:** Handle at least 50 concurrent queries without failure.

5 CONCLUSION AND FUTURE WORK

The **WoxionChat** project has demonstrated that building an internal AI system is not only feasible but also provides a strategic advantage in security. Future work will focus on integrating **Multi-modal Agent** capabilities, enabling the AI to perform business tasks such as automatically updating CRM/ERP systems.

References

[1] Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS.

[2] Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. ACL.

[3] Smith, R. (2007). *An Overview of the Tesseract OCR Engine*. ICDAR.

[4] Hu, E. J., et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv.

[5] Microsoft. (2024). *MarkItDown: Multimodal tool for converting files to Markdown*. GitHub.

[6] ElevenLabs. (2024). *Neural Speech Synthesis and Voice Cloning Technology*. API Documentation.

[7] Django Software Foundation. (2024). *Django Documentation (Version 5.2)*. https://docs.djangoproject.com/

[8] Banker, K., et al. (2016). *MongoDB in Action*. Manning Publications.

[9] Carlson, J. L. (2013). *Redis in Action*. Manning Publications.

[10] Chase, H. (2022). *LangChain: Building applications with LLMs through composability*. GitHub.

[11] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

[12] Ramírez, S. (2024). *FastAPI Documentation*. https://fastapi.tiangolo.com/

[13] Cuc Thi Bui, Nguyen Truong Son, Truong Van Trang, Lam Viet Phung, Pham Nhut Huy, Hoang Anh Le, Quoc Huu Van, Phong Nguyen-Thuan Do, Van Le Tran Truc, Duc Thanh Chau, and Le-Minh Nguyen. (2025). *VMLU Benchmarks: A comprehensive benchmark toolkit for Vietnamese LLMs. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11495–11515, Vienna, Austria. Association for Computational Linguistics.*