ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

TÀI LIỆU KỸ THUẬT

TECH STACK
HỆ THỐNG UIT AI ASSISTANT

NHÓM THỰC HIỆN:
Quách Gia Kiệt - 23520819
Nguyễn Tuấn Kiệt - 23520815

GV HƯỚNG DẪN:
Th.S Nguyễn Công Hoan

TP. HỒ CHÍ MINH, 2025

Mục lục

Chương 1 Giới thiệu

Tài liệu này mô tả chi tiết về tech stack được sử dụng trong hệ thống UIT AI Assistant - một hệ thống AI hỗ trợ sinh viên truy vấn thông tin học vụ. Hệ thống được xây dựng trên kiến trúc microservices, kết hợp các công nghệ hiện đại trong lĩnh vực backend, AI/ML, database, và frontend.

Tech stack được lựa chọn dựa trên các tiêu chí:

- **Hiệu năng:** Khả năng xử lý requests nhanh và ổn định
- **Khả năng mở rộng:** Dễ dàng thêm tính năng mới và scale hệ thống
- **Hệ sinh thái:** Có community lớn, tài liệu phong phú, và libraries hỗ trợ
- **Đặc thù bài toán:** Phù hợp với yêu cầu xử lý tiếng Việt và tích hợp AI

Hệ thống được tổ chức thành 5 thành phần chính: Backend Services (API Gateway, AI Agent, MCP Server), AI/ML Components (RAG, LLM, Embeddings), Data Layer (4 loại databases), Frontend (Web + Extension), và Infrastructure (Docker, Communication Protocols).

Chương 2 Backend Technologies

2.1 Go - API Gateway

2.1.1 Mô tả

API Gateway được phát triển bằng Go với framework Gin. Go là ngôn ngữ compiled, statically-typed, được thiết kế bởi Google, nổi tiếng với hiệu năng cao và concurrency tốt.

2.1.2 Vai trò trong hệ thống

API Gateway đóng vai trò cầu nối giữa frontend và backend services. Các chức năng chính:

- Xử lý HTTP/REST requests từ web frontend
- Quản lý chat sessions và lưu trữ chat history vào MongoDB
- Giao tiếp với AI Agent thông qua gRPC
- Cache credentials và session data trong Redis
- Expose WebSocket endpoint để support real-time streaming

2.1.3 Lý do lựa chọn

- **Hiệu năng cao:** Go compiler tạo native binary, runtime không có GC pauses lớn
- **Concurrency:** Goroutines và channels giúp xử lý nhiều requests đồng thời hiệu quả
- **Gin framework:** Lightweight, nhanh, dễ sử dụng cho REST APIs
- **gRPC support:** Go có support tốt cho gRPC, phù hợp cho communication với Python Agent

2.2 Python - AI Agent và MCP Server

2.2.1 Mô tả

AI Agent và MCP Server được xây dựng bằng Python, sử dụng các frameworks chuyên biệt:

- **LangGraph:** Framework để xây dựng agent với stateful, multi-step reasoning
- **FastMCP:** Framework để triển khai Model Context Protocol server
- **LangChain:** Library cho LLM integration và tool calling

2.2.2 Vai trò trong hệ thống

AI Agent:

- Orchestrate toàn bộ quy trình xử lý query
- Thực hiện ReAct pattern (Reasoning & Acting)
- Gọi MCP tools và native tools để lấy thông tin
- Tổng hợp kết quả và sinh câu trả lời cho sinh viên
- Quản lý state và context thông qua checkpointer

MCP Server:

- Expose các tools theo chuẩn Model Context Protocol
- Cung cấp retrieval tools (retrieve_regulation, retrieve_curriculum)
- Cung cấp DAA integration tools (get_grades, get_schedule)
- Thực thi tool functions và trả về structured JSON results

2.2.3 Lý do lựa chọn

- **Hệ sinh thái AI/ML:** Python là ngôn ngữ dominates trong lĩnh vực AI, có hầu hết các libraries và frameworks
- **LangGraph:** Cho phép xây dựng agent với complex reasoning flow, state persistence, và human-in-the-loop
- **FastMCP:** Triển khai MCP protocol dễ dàng, tự động generate tool schemas từ Python functions
- **Rapid development:** Python có syntax đơn giản, phù hợp cho việc thử nghiệm và iterate nhanh

Chương 3 AI & ML Technologies

3.1 Large Language Models

Hệ thống hỗ trợ hai nhà cung cấp LLM chính thông qua LangChain:

3.1.1 OpenAI GPT

- **Models:** GPT-4, GPT-4-turbo, GPT-3.5-turbo
- **Vai trò:** Thực hiện reasoning, tool calling, và text generation
- **Ưu điểm:** Function calling tốt, hiểu ngữ cảnh sâu, response chất lượng cao

3.1.2 Google Gemini

- **Models:** Gemini Pro, Gemini Flash
- **Vai trò:** Thực hiện reasoning và generation, đặc biệt dùng Gemini Flash cho metadata generation
- **Ưu điểm:** Chi phí thấp hơn GPT, xử lý tiếng Việt tốt, có free tier rộng rãi

3.2 RAG Framework - LlamaIndex

3.2.1 Mô tả

LlamaIndex (trước đây là GPT Index) là framework chuyên biệt cho việc xây dựng RAG (Retrieval-Augmented Generation) systems. Framework cung cấp các công cụ cho indexing, retrieval, và query processing.

3.2.2 Vai trò trong hệ thống

- Xây dựng knowledge base từ documents (PDF, DOCX)
- Chunking strategies cho văn bản tiếng Việt và cấu trúc quy định UIT
- Query processing và semantic search
- Integration với ChromaDB vector store

3.2.3 Lý do lựa chọn

- **Specialized for RAG:** Tập trung vào retrieval, không bloated như LangChain
- **Flexible chunking:** Hỗ trợ custom chunking strategies
- **Multi-database support:** Dễ dàng switch giữa các vector stores
- **Query engine:** Có sẵn query optimization và reranking pipelines

3.3 Agent Orchestration - LangGraph

3.3.1 Mô tả

LangGraph là framework từ LangChain team dành cho việc xây dựng stateful, multi-actor applications với LLMs. Framework cho phép mô hình hóa agent logic dưới dạng directed graph với nodes và edges.

3.3.2 Vai trò trong hệ thống

- Xây dựng AI Agent với ReAct pattern (Reasoning & Acting)
- Quản lý agent state qua nhiều lượt hội thoại
- Orchestrate luồng xử lý: agent node → tools node → agent node
- Persist state vào PostgreSQL qua checkpointer
- Support conditional routing và loops trong agent flow

3.3.3 Kiến trúc Agent Graph

Agent được mô hình hóa bằng StateGraph với các thành phần:

- **AgentState:** Chứa messages (chat history) và user context
- **Agent Node:** LLM thực hiện reasoning và quyết định gọi tools
- **Tools Node:** Thực thi tools mà LLM yêu cầu
- **Conditional Edges:** Route dựa trên LLM output (cần tools hay không)
- **Checkpointer:** Tự động save/load state từ PostgreSQL

Flow: START → Agent Node → [Conditional] → Tools Node → Agent Node → ... → END

3.3.4 Lý do lựa chọn

- **State management:** Built-in checkpointer cho conversation persistence
- **Complex flows:** Hỗ trợ multi-step reasoning, loops, conditional routing
- **Debuggability:** Visualize graph, inspect state tại mỗi node
- **Human-in-the-loop:** Có thể interrupt và resume execution
- **Tool integration:** Tích hợp tốt với LangChain tools và MCP adapters
- **Production-ready:** Thread-safe, có persistence, error handling

3.4 Embeddings - OpenAI text-embedding-3-small

3.4.1 Mô tả

OpenAI text-embedding-3-small là embedding model tạo vector representations cho text. Model có dimensionality 1536, được train trên multilingual data bao gồm tiếng Việt.

3.4.2 Vai trò trong hệ thống

- Convert documents thành vector embeddings trong quá trình indexing
- Convert user queries thành vectors để thực hiện semantic search
- Đảm bảo query và documents nằm trong cùng embedding space

3.4.3 Lý do lựa chọn

- **Quality:** Accuracy cao trên nhiều benchmarks
- **Vietnamese support:** Xử lý tiếng Việt tốt hơn các open-source alternatives
- **Cost-effective:** text-embedding-3-small rẻ hơn đáng kể so với ada-002
- **Dimensionality:** 1536 dimensions cân bằng giữa accuracy và storage cost

3.5 Reranking - ViRanker

3.5.1 Mô tả

ViRanker là cross-encoder model được fine-tune cho tiếng Việt, dùng để rerank các candidates từ semantic search. Model được deploy trên Modal serverless GPU.

3.5.2 Vai trò trong hệ thống

- Nhận top-K candidates từ ChromaDB semantic search
- Tính toán relevance scores chi tiết hơn simple cosine similarity
- Rerank candidates theo scores và filter theo threshold
- Trả về top-3 chunks chính xác nhất cho agent

3.5.3 Lý do lựa chọn

- **Vietnamese-specific:** Model được train riêng cho tiếng Việt
- **Cross-encoder architecture:** Hiểu ngữ cảnh tốt hơn bi-encoder embeddings
- **Modal deployment:** Serverless GPU giúp scale tự động và chỉ trả tiền khi sử dụng
- **Accuracy improvement:** Cải thiện đáng kể so với chỉ dùng cosine similarity

3.6 Document Parsing - LlamaParse

3.6.1 Mô tả

LlamaParse là document parser dựa trên LLM, có khả năng hiểu cấu trúc tài liệu phức tạp. Parser sử dụng vision models để analyze layout và extract content chính xác.

3.6.2 Vai trò trong hệ thống

- Parse PDF và DOCX thành Markdown format
- Xử lý các bảng biểu phức tạp với merged cells
- Nhận diện hierarchy của tài liệu (CHƯƠNG, Điều, Khoản)
- Giữ nguyên formatting quan trọng (lists, numbering)

3.6.3 Lý do lựa chọn

- **LLM-based:** Hiểu context tốt hơn rule-based parsers (PyPDF, PDFMiner)
- **Table handling:** Parse bảng phức tạp chính xác, tạo markdown tables đúng format

- **Layout understanding:** Xử lý multi-column, letterhead, footer tốt
- **Vietnamese support:** Không bị OCR errors với tiếng Việt

Chương 4 Data Layer

Hệ thống sử dụng 4 loại databases khác nhau, mỗi loại phục vụ một mục đích cụ thể trong kiến trúc.

4.1 MongoDB

4.1.1 Mô tả

MongoDB là NoSQL document database, lưu trữ dữ liệu dưới dạng JSON-like documents (BSON).

4.1.2 Vai trò trong hệ thống

- Lưu trữ chat sessions (session_id, user_id, created_at, updated_at)
- Lưu trữ chat messages (role, content, timestamp, session_id)
- Query để hiển thị chat history trên frontend

4.1.3 Lý do lựa chọn

- **Schema flexibility:** Messages có thể có fields khác nhau (text, tool calls, etc.)
- **Document model:** Phù hợp với chat data có cấu trúc nested
- **Query performance:** Fast queries cho listing sessions và messages
- **Easy to scale:** Replica sets và sharding khi cần

4.2 PostgreSQL

4.2.1 Mô tả

PostgreSQL là relational database, ACID-compliant, hỗ trợ JSONB và advanced queries.

4.2.2 Vai trò trong hệ thống

- Làm backend cho LangGraph checkpointer
- Lưu trữ agent state (messages, context) theo thread_id
- Support atomic updates và transactions

4.2.3 Lý do lựa chọn

- **ACID compliance:** Đảm bảo consistency khi update state
- **JSONB support:** Lưu agent state dưới dạng JSON nhưng vẫn có ACID
- **LangGraph compatibility:** Official checkpointer support
- **Reliability:** Mature, stable, có backup/recovery tốt

4.3 Redis

4.3.1 Mô tả

Redis là in-memory key-value store, hỗ trợ các data structures như strings, hashes, lists, sets.

4.3.2 Vai trò trong hệ thống

- Cache DAA cookies sync từ browser extension
- Store session tokens và temporary data
- Fast lookups cho credentials khi agent gọi DAA tools

4.3.3 Lý do lựa chọn

- **Speed:** In-memory, sub-millisecond latency
- **TTL support:** Tự động expire cookies sau một khoảng thời gian
- **Simple:** Key-value model đơn giản, phù hợp cho caching
- **Lightweight:** Không cần overhead của full database

4.4 ChromaDB

4.4.1 Mô tả

ChromaDB là vector database được thiết kế cho AI applications. Database lưu trữ embeddings và metadata, hỗ trợ similarity search.

4.4.2 Vai trò trong hệ thống

- Lưu trữ vector embeddings của knowledge base chunks
- Lưu trữ metadata (document_id, title, category, hierarchy_path)
- Thực hiện semantic search với approximate nearest neighbor
- Tổ chức data thành collections (regulation, curriculum)

4.4.3 Lý do lựa chọn

- **Easy to use:** Python-first API, không cần setup phức tạp
- **Metadata filtering:** Filter theo category trước khi search
- **Lightweight:** Có thể chạy embedded hoặc client-server
- **Open source:** Free, self-hostable, không vendor lock-in

Chương 5 Frontend Technologies

5.1 Web Frontend - React

5.1.1 Mô tả

Web frontend được xây dựng bằng React với Vite làm build tool và Tailwind CSS cho styling.

5.1.2 Tech stack chi tiết

- **React:** Library cho building UI với component-based architecture
- **Vite:** Next-generation build tool, fast HMR, optimized builds
- **Tailwind CSS:** Utility-first CSS framework
- **React Router:** Client-side routing

5.1.3 Vai trò trong hệ thống

- Cung cấp chat interface cho sinh viên
- Hiển thị danh sách sessions và chat history
- Gửi messages tới API Gateway qua HTTP
- User authentication và authorization

5.1.4 Lý do lựa chọn

- **React:** Component reusability, large ecosystem, team familiarity
- **Vite:** Nhanh hơn Create React App đáng kể, DX tốt
- **Tailwind:** Rapid UI development, consistent design, no CSS files bloat
- **Responsive:** Hoạt động tốt trên cả desktop và mobile

5.2 Browser Extension - Svelte

5.2.1 Mô tả

Browser extension được phát triển bằng Svelte, tuân theo Chrome Extension Manifest V3.

5.2.2 Vai trò trong hệ thống

- Tự động detect khi sinh viên truy cập daa.uit.edu.vn
- Extract cookies từ browser
- Upload cookies lên Redis với key format daa_cookie:user_id
- Background service để sync cookies định kỳ

5.2.3 Lý do lựa chọn

- **Bundle size:** Svelte compiles to vanilla JS, nhẹ hơn React
- **Performance:** Không có virtual DOM overhead
- **Simplicity:** Extension logic đơn giản, không cần complexity của React
- **Manifest V3:** Svelte phù hợp với service worker architecture

Chương 6 Infrastructure & Communication

6.1 Docker & Docker Compose

6.1.1 Mô tả

Tất cả các services được containerized bằng Docker và orchestrate bằng Docker Compose.

6.1.2 Services được containerize

- API Gateway (Go)
- AI Agent (Python)
- MCP Server (Python)
- Knowledge Builder CLI (Python)
- MongoDB
- PostgreSQL
- Redis
- ChromaDB

6.1.3 Lý do lựa chọn

- **Environment consistency:** Dev, staging, prod giống nhau
- **Dependency isolation:** Mỗi service có dependencies riêng
- **Easy deployment:** docker-compose up để start toàn bộ stack
- **Resource management:** Limit CPU/memory cho từng container

6.2 Communication Protocols

Hệ thống sử dụng nhiều protocols khác nhau cho communication giữa các components.

6.2.1 HTTP/REST

- **Frontend ↔ API Gateway:** REST APIs cho chat operations
- **Browser Extension ↔ API Gateway:** POST requests để upload cookies
- **Agent ↔ MCP Server:** MCP protocol over HTTP

6.2.2 gRPC

- **API Gateway ↔ AI Agent:** Bidirectional streaming cho chat
- **Ưu điểm:** Binary protocol, nhanh hơn JSON/REST, support streaming

6.2.3 WebSocket

- **Frontend ↔ API Gateway:** Real-time streaming responses (planned)
- **Ưu điểm:** Full-duplex communication, low latency

6.2.4 MCP over HTTP

- **Agent ↔ MCP Server:** Tool discovery và invocation
- **Ưu điểm:** Chuẩn hóa tool integration, không hard-code, dễ add tools mới

Chương 7 Pipelines & System Flow

7.1 Indexing Pipeline

Pipeline xây dựng knowledge base từ raw documents thành vector embeddings.

7.1.1 Luồng xử lý

1. **Document Collection:**
    - Thu thập PDF từ website UIT
    - Crawl Markdown từ DAA portal
    - Lưu vào data/raw/regulation và data/raw/curriculum
2. **Parsing & Cleaning:**
    - LlamaParse convert PDF → Markdown
    - Loại bỏ letterhead, footer, navigation
    - Normalize headings và lists
3. **Metadata Generation:**
    - LLM extract metadata (title, category, year, effective_date)
    - Validate schema với Pydantic
    - Lưu vào data/processed/
4. **Chunking:**
    - Chia documents theo semantic boundaries
    - Nhận diện hierarchy (CHƯƠNG, Điều, Khoản)
    - Add context (metadata + hierarchy_path) vào mỗi chunk
5. **Embedding & Indexing:**
    - OpenAI text-embedding-3-small tạo vectors
    - ChromaDB lưu embeddings + metadata
    - Tổ chức thành collections (regulation, curriculum)

7.1.2 Tools sử dụng

- Knowledge Builder CLI (Python)
- LlamaParse API
- Google Gemini Flash (metadata generation)
- OpenAI Embeddings API
- ChromaDB

7.2 Retrieval Pipeline

Pipeline tìm kiếm và trả về chunks liên quan từ knowledge base.

7.2.1 Luồng xử lý

1. **Query Embedding:**
    - Agent gọi MCP tool retrieve_regulation/retrieve_curriculum
    - MCP Server nhận query string
    - OpenAI embeddings convert query → vector
2. **Semantic Search:**
    - ChromaDB search theo cosine similarity
    - Filter theo collection (regulation hoặc curriculum)
    - Lấy top-20 candidates
3. **Reranking:**
    - Gửi query + candidates tới ViRanker (Modal GPU)
    - ViRanker tính relevance scores
    - Rank lại candidates theo scores
4. **Filtering:**
    - Filter theo threshold score (≥0.25)
    - Program filter (tránh nhầm lẫn giữa các ngành)
    - Lấy top-3 chunks

5. **Return Results:**
    - MCP Server trả JSON: chunks + metadata
    - Agent nhận kết quả, add vào context
    - LLM sử dụng context để sinh câu trả lời

7.3 Agent ReAct Flow

Luồng xử lý query của agent theo ReAct pattern.

7.3.1 Luồng xử lý

1. **User Query:**
    - Sinh viên gửi message từ frontend
    - API Gateway forward tới Agent qua gRPC
    - Agent load state từ PostgreSQL checkpointer
2. **Reasoning:**
    - LLM analyze query + chat history + tool results
    - Quyết định có cần gọi tools không
    - Nếu có: chọn tools và generate arguments
3. **Acting:**
    - Agent gọi MCP tools (retrieve, get_grades, get_schedule)
    - Agent gọi native tools (get_user_credential)
    - Tools thực thi và trả kết quả
4. **Observation:**
    - Agent nhận tool results
    - Add results vào state as tool messages
    - Quay lại bước Reasoning với context mới
5. **Response:**
    - Khi đủ thông tin, LLM sinh câu trả lời
    - Agent save state vào PostgreSQL

- API Gateway save message vào MongoDB
- Response trả về frontend

7.3.2 Tools available

MCP Tools:

- retrieve_regulation(query: str) → chunks
- retrieve_curriculum(query: str) → chunks
- get_grades(cookie: str) → grades JSON
- get_schedule(cookie: str) → schedule JSON

Native Tools:

- get_user_credential(user_id: str) → cookie from Redis

Chương 8 Kết luận

Tech stack của hệ thống UIT AI Assistant được lựa chọn và tổ chức cẩn thận để đáp ứng các yêu cầu về hiệu năng, khả năng mở rộng, và đặc thù xử lý tiếng Việt.

8.1 Điểm mạnh

- **Kiến trúc modular:** Separation of concerns rõ ràng, dễ maintain và extend
- **Best-in-class tools:** Mỗi component dùng công nghệ phù hợp nhất (Go cho gateway, Python cho AI)
- **Chuẩn hóa:** Sử dụng MCP protocol, gRPC, REST - tránh hard-coding
- **Vietnamese-optimized:** LlamaParse, ViRanker, custom chunking cho tiếng Việt
- **Scalable:** Docker, microservices, multiple databases cho different workloads

8.2 Trade-offs

- **Complexity:** Nhiều technologies khác nhau, learning curve cao
- **Operational overhead:** Phải manage 4 databases, multiple services
- **Cost:** OpenAI API, LlamaParse, Modal GPU có chi phí
- **Latency:** Multi-hop requests (Gateway → Agent → MCP → ChromaDB) add latency

8.3 Cải tiến trong tương lai

- Thêm observability (Prometheus metrics, tracing với OpenTelemetry)
- Implement caching layer để giảm LLM API calls
- Migrate ChromaDB sang Qdrant hoặc Weaviate cho production scale
- Add load balancing và horizontal scaling cho Agent và MCP Server
- Implement CI/CD pipelines và automated testing