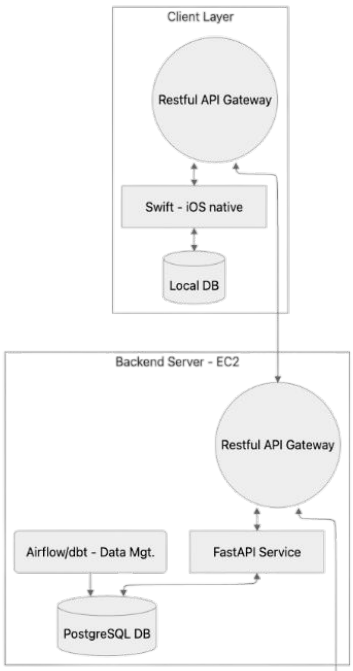
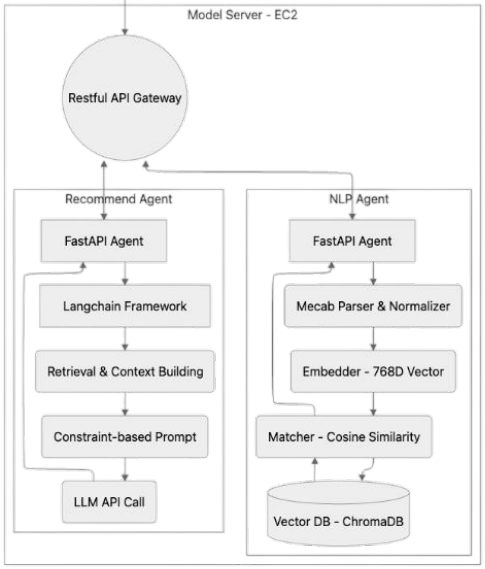
Seungwan, Hong

www.hosugator.com

Dotodo

AI-Powered To-Do Recommendation Service Development





System Architecture

An AWS EC2-based Multi-Instance MSA structure was designed and implemented to decouple the project's core business logic (CRUD) from high-performance computation logic (NLP/ML).
The architecture features server multi-layering into a Backend Server and a Model Server, which achieves Fault Isolation and manages traffic load.
All service communication is enforced to use the standardized JSON/HTTP protocol via a FastAPI-based Restful API Gateway, maximizing client-server communication consistency.

Core Technology

Langchain-based LLM RAG System Implementation and Optimization

A RAG system using the user to-do history to provide personalized recommendation.
- Personalized Context Generation: user data dynamically inserted into LLM prompt.
- Zero-shot Recommendation Strategy: For Cold Start Problem, the system was designed to explicitly inject general patterns, ensuring proper initial recommendations.
- Consistency Control: by tuning Langchain parameters, guaranteeing consistent recommendation results that are contextually relevant to the user's history.

VectorDB and NLP Pipeline for High-Performance Recommendation

A Cosine Similarity-based recommendation system was implemented in parallel with the LLM to secure both response speed and accuracy.

Problem Solving

Improving Recommendation Quality and Utility

- Exclusion Rule for Identical Items: Modified logic to exclude past items that exactly matched the current user's input `simplified_text` from the recommendation list, ensuring Recommendation Diversity.
- Top-K Limiting: Limit the final recommendations to the Top-3 items to prevent overwhelming the user with excessive information, thereby improving the UX.

Optimizing Langchain RAG System Response Speed

Latency occurred due to VectorDB search time and LLM inference time inherent in the Langchain RAG system.

- Asynchronous Processing: Utilized FastAPI's asynchronous features to handle LLM API calls and VectorDB I/O operations in parallel, reducing overall response time.
- Context Size Adjustment: Minimized the size of the user history inserted into the prompt (e.g., using only 3 months of data) to balance LLM inference cost and response time.

Retrospective

Successfully integrated and implemented core industry-demanded technology stacks such as AWS, Docker, MSA decoupling, and NLP Agent, effectively demonstrating complex system construction capability.
Future improvement goals are as follows:
- LLM Evaluation System: Introduce the concept of 'LLM as a judge' by establishing a mechanism where one LLM generates recommendations and a separate LLM evaluates and autonomously improves recommendations.
- Advanced Multi-Recommendation Logic: Beyond the current Cosine Similarity/Frequency-based recommendation, the plan is to refine multi-recommendation modules based on "Pattern/Context/Relevance", significantly enhancing situational recommendation accuracy.

PostgreSQL FastAPI Jira LangChain Chroma aws docker Mecab dbt