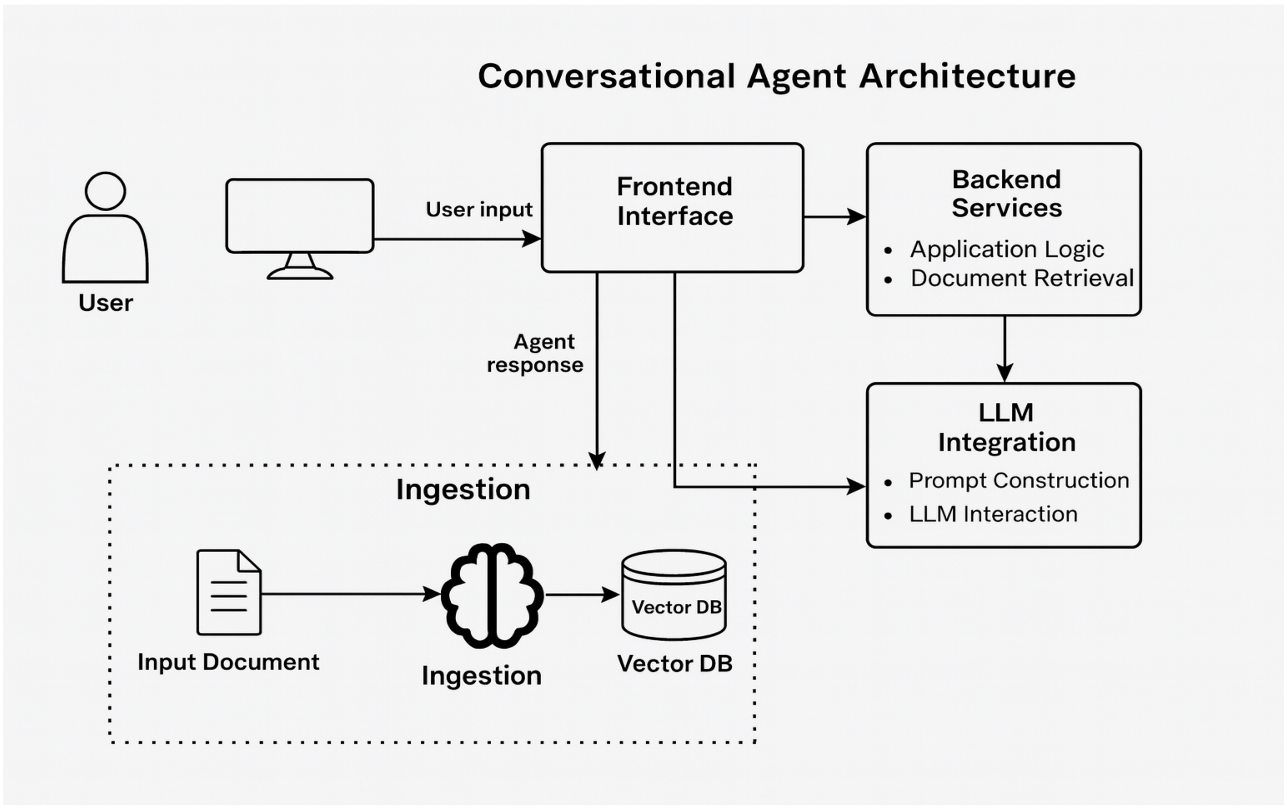
Conversational Agent Architecture Overview

Architecture Diagram



1. Objective

The Conversational Agent system enables intelligent, context-aware dialogue through natural language interaction. It combines a retrieval-augmented backend (using FastAPI, FAISS, and OpenAI APIs) with a lightweight web-based frontend (Streamlit) to deliver contextualized responses using user-ingested documents or knowledge sources. The system design prioritizes persistence, modularity, and scalability while maintaining a simple deployment workflow.

2. System Components

Component	Technology	Purpose
Frontend UI	Streamlit	Provides a web interface for ingestion and Q&A.
Backend API	FastAPI	Handles ingestion, embeddings, and conversation logic.
Vector Database	FAISS	Stores and retrieves document embeddings.
LLM Engine	OpenAI GPT-4o/4o-mini	Generates contextual answers.
Persistence Layer	FAISS Binary + JSON	Preserves index and metadata across restarts.
Environment Management	python-dotenv + PowerShell	Manages secrets and process startup.

3. Process Flow

1. User uploads or pastes text in Streamlit sidebar. 2. Streamlit sends POST /ingest to backend. 3. Backend generates embeddings, stores them in FAISS, and saves metadata. 4. User queries via POST /ask, backend retrieves top documents, and calls OpenAI Chat API. 5. Responses are displayed in conversational format.

4. Data Storage

File	Purpose	Format
faiss_index.bin	FAISS vector index	Binary
metadata.json	Document metadata	JSON
.env	API key and config	Key-value
sessions	Chat history (in-memory)	Python dict

5. Deployment Overview

Local Development: Run run_all.ps1 to start backend (Uvicorn) and frontend (Streamlit). Accessible at http://localhost:8501 for frontend and http://127.0.0.1:5050 for backend. Production Deployment: Backend served behind Nginx with HTTPS, Streamlit or React frontend hosted separately, and persistent FAISS directory mounted on durable storage.

6. Security and Scalability Highlights

Security: Environment secrets via .env, HTTPS recommended, and isolated user sessions. Scalability: Horizontal scaling via distributed FAISS/Redis, multi-worker Uvicorn setup, and embedding batching.

7. Future Enhancements

1. Cross-encoder reranking for improved retrieval. 2. Persistent session memory in Redis or SQLite. 3. Multi-tenancy support by tenant namespacing. 4. Audit logging and monitoring integration. 5. Plugin system for domain-specific tools.

8. Summary

The Conversational Agent architecture provides a lightweight, modular foundation for context-aware dialogue systems using FastAPI for orchestration, FAISS for semantic retrieval, OpenAI LLMs for reasoning, Streamlit for UI, and persistent storage for durability. It’s extensible towards multi-user, enterprise-scale deployments and serves as a foundation for future agentic AI workflows.