HR Assistant System - Project Summary

■ Technical Source Code Documentation

■ Project Overview

HR Assistant System is an AI-powered HR chatbot with CV evaluation using:
• **RAG (Retrieval-Augmented Generation)** for intelligent Q&A;
• **Azure OpenAI** for natural language processing
• **FAISS** for vector similarity search
• **Multi-language Support** (English/Vietnamese)

■■ Architecture Overview

■ Source Code Structure & Purpose

1. Backend Core Files*

app.py (Main API Server)
Purpose: FastAPI application with REST endpoints

Key Functions:

Technical Details:
• **Framework**: FastAPI (async Python web framework)
• **Validation**: Pydantic models for type safety
• **CORS**: Handles cross-origin requests from frontend
• **Logging**: Request/response logging for debugging
• **Error Handling**: Try-catch for fallback mechanisms

---

chain_setup.py (RAG System)
Purpose: Initialize and manage retrieval-augmented generation pipeline

Key Components:

Technical Flow:
1. **Input**: User question + language
2. **Embedding**: Convert text → vector (768-dim)
3. **Search**: FAISS finds top-5 similar FAQs (similarity search)
4. **Context**: Combines found FAQs into prompt
5. **Generation**: Azure OpenAI generates response using context
6. **Fallback**: If Azure fails → SimpleFallbackLLM (rule-based)

Key Dependencies:
• `langchain`: RAG pipeline orchestration
• `faiss-cpu`: Vector similarity search
• `azure-openai`: LLM + embeddings

---

cv_extractor.py (CV Parser)
Purpose: Extract information from CV documents

Key Functions:

Supported Formats:
• PDF (`.pdf`) PyPDF2
• Word (`.docx`) python-docx

• Plain Text (`.txt`) String parsing

---

function_tools.py (Utility Functions)
Purpose: Helper functions for data processing

Key Functions:

---

company_data.py (Business Logic)
Purpose: Company information and job positions

Data Includes:

Used By:

• CV evaluation (skill matching)
• HR FAQ responses (policy questions)

---

2. Frontend Components (React/Next.js)*

pages/index.jsx (Main Page)
Purpose: Main application entry point

---

components/ChatBox.jsx (Chat Display)
Purpose: Display chat messages and responses

Key Features:

---

components/InputBar.jsx (Message Input)
Purpose: Handle user input and message submission

---

components/CVUpload.jsx (CV Upload & Evaluation)
Purpose: Upload CV and trigger evaluation

---

components/Sidebar.jsx (Navigation)
Purpose: Language toggle and settings

---

3. Data Files

backend/data/hr_faq.csv
Purpose: HR knowledge base for RAG system
Format:

Usage:
• Loaded into FAISS vector index
• Used for semantic search in chat
• Provides context for LLM responses

■ Technology Deep Dive

Backend Technologies

1. FastAPI
What: Modern Python web framework
Why:
• Fast (ASGI)
• Type hints (automatic validation)
• Auto-generates API docs
• Great for AI/ML

Usage in Project:

---

2. LangChain
What: Framework for building LLM applications
Why:
• Abstracts LLM complexity
• Built-in RAG support
• Memory management
• Tool integration

Usage in Project:

---

3. FAISS (Facebook AI Similarity Search)
What: Vector similarity search library
Why:
• Fast nearest neighbor search
• Supports millions of vectors
• In-memory or disk storage
• Optimized for high dimensions

Usage in Project:

---

4. Azure OpenAI
What: Cloud-hosted OpenAI models

Why:
• Managed service (no infrastructure)
• Enterprise security
• Multiple models available
• Pay-as-you-go

Models Used:

---

5. PyPDF2
What: PDF text extraction
Why: Parse CV PDFs

---

6. python-docx
What: Word document parsing
Why: Parse CV Word files

Frontend Technologies

1. Next.js
What: React framework with SSR/SSG
Why:
• File-based routing
• API routes
• Built-in optimization
• Vercel deployment

Usage in Project:

---

2. React Hooks
What: Function-based component state management
Why: Simpler than class components

Usage in Project:

---

3. Tailwind CSS
What: Utility-first CSS framework
Why: Fast styling without custom CSS

---

4. Axios
What: HTTP client library
Why: Simple API calls

■ Data Flow Examples

Example 1: User Asks Question

Example 2: User Uploads CV*

■ Key Technical Features

1. Multi-Language Support*

Implementation:
• Language detection: Keyword matching
• Vietnamese keywords: "tôi", "c■a", "gì"
• English keywords: "I", "me", "what"
• Responses: Stored for each language
• Database: FAQ can have multiple language entries

---

2. Fallback Mechanisms (3-Level)*

---

3. FAISS Vector Search*

***4. Skill Matching Algorithm**

■ Dependencies Summary

***Backend (requirements.txt)**

***Frontend (package.json)**

■ Security Considerations

1. API Keys*

• **Storage**: `.env` file (never commit)
• **Loading**: `python-dotenv`
• **Environment Variables**: Set in Vercel/Render dashboard

2. CORS (Cross-Origin)*

3. Input Validation*

■ Performance Characteristics

Operation	Time	Notes
Chat response	1-3s	Azure + FAISS search
CV evaluation	2-5s	PDF parsing + skill matching
FAISS search	<100ms	1000 FAQs
Embedding	200-500ms	Azure OpenAI API
Fallback response	<50ms	In-memory rules

■ Deployment Architecture

■ Learning Resources

***Key Concepts**:*

1. **RAG**: How to augment LLMs with retrieval
2. **Vector Databases**: Similarity search mechanics
3. **FastAPI**: Building production APIs
4. **React Hooks**: State management patterns
5. **Deployment**: Cloud infrastructure

***Further Reading**:*

• LangChain Docs: https://python.langchain.com
• Azure OpenAI: https://learn.microsoft.com/en-us/azure/ai-services/openai/
• FAISS: https://github.com/facebookresearch/faiss
• Next.js: https://nextjs.org/docs

■ Conclusion

This HR Assistant System demonstrates:

- ■ Modern full-stack development (Next.js + FastAPI)
- ■ AI/ML integration (Azure OpenAI + FAISS)
- ■ RAG implementation (question answering)
- ■ Multi-language support (EN/VI)
- ■ Robust error handling (3-level fallbacks)
- ■ Production-ready architecture (Vercel + Render)

All components work together seamlessly to provide an intelligent HR assistant that can answer questions and evaluate CVs!