HR Resource Query Chatbot

🎯 Objective

Build an intelligent HR assistant chatbot that can answer resource allocation queries using natural language processing and retrieval techniques.

📋 Problem Statement

Create an AI-powered chatbot that helps HR teams find employees by answering queries like:

- "Find Python developers with 3+ years experience"
- "Who has worked on healthcare projects?"
- "Suggest people for a React Native project"
- "Find developers who know both AWS and Docker"

🏗️ Core Requirements

1. Data Layer

- Create sample employee dataset (15+ employees)
- Include: name, skills, experience_years, past_projects, availability

2. AI/ML Component (RAG System)

- **Retrieval**: Parse queries and find relevant employees using embeddings
- **Augmentation**: Combine retrieved employee data with query context
- **Generation**: Use LLM to create natural language responses with recommendations

Implementation Options:

- **Full RAG**: OpenAI/Llama + embeddings + vector search
- **Hybrid**: Semantic search + template-based responses
- **Advanced**: HuggingFace transformers + FAISS + local LLM (Ollama)
- **Simple**: Embedding similarity + smart formatting (no LLM required)

3. Backend API (Python/FastAPI)

- REST endpoint for chat queries: `POST /chat`
- Employee search endpoint: `GET /employees/search`

- Basic error handling and validation
- **Preferred**: FastAPI for automatic docs and async support

4. Frontend Interface

- Simple chat interface
- Display employee results clearly
- Can use Streamlit, React, or simple HTML

📊 Sample Data Structure


```
{
  "employees": [
    {
      "id": 1,
      "name": "Alice Johnson",
      "skills": ["Python", "React", "AWS"],
      "experience_years": 5,
      "projects": ["E-commerce Platform", "Healthcare Dashboard"],
      "availability": "available"
    }
  ]
}
```


🔍 Expected Interaction (RAG System)

User: "I need someone experienced with machine learning for a healthcare project"

RAG Process:
1. Retrieval: Find employees with ML + healthcare experience
2. Augmentation: Combine their profiles with query context
3. Generation: Create natural response

Response:
"Based on your requirements for ML expertise in healthcare, I found 2 excellent candidates:

Dr. Sarah Chen would be perfect for this role. She has 6 years of ML experience and specifically worked on the 'Medical Diagnosis Platform' project where she implemented computer vision for X-ray analysis. Her skills include TensorFlow, PyTorch, and medical data processing. She's currently available and has published 3 papers on healthcare AI.

Michael Rodriguez is another strong candidate with 4 years of ML experience. He built the 'Patient Risk Prediction System' using ensemble methods and has experience with HIPAA

compliance for healthcare data. He knows scikit-learn, pandas, and has worked with electronic health records.

Both have the technical depth and domain expertise you need. Would you like me to provide more details about their specific healthcare projects or check their availability for meetings?"

🎯 Evaluation Criteria

Technical Implementation (70%)

- **RAG Pipeline**: Retrieval + Augmentation + Generation quality
- **ML/AI Integration**: Embeddings, vector search, LLM usage
- **Code Architecture**: Clean separation of RAG components
- **Performance**: Query response time and relevance

User Experience (20%)

- Chat interface usability
- Result presentation quality
- Error handling

Innovation & Problem-Solving (10%)

- Creative solutions and features
- Handling of edge cases
- Technical decision justification

📋 Deliverables

Required:

1. **GitHub Repository** with complete source code
2. **Working Demo** (local setup or deployed)
3. **README.md** with:
    - Setup and installation instructions
    - API documentation
    - Architecture overview
    - **AI Development Process** section (see below)
4. **Sample dataset** with realistic employee data

Bonus Features:

- Vector similarity search with embeddings
- Deployed version (Streamlit Cloud, Vercel, etc.)

📄 README.md Requirements

Your README.md should include these sections:

HR Resource Query Chatbot

Overview
Brief description of the project and approach

Features
List of implemented features

Architecture
System design and component overview

Setup & Installation
Step-by-step instructions to run locally

API Documentation
Endpoint descriptions and examples

AI Development Process
Document how you used AI tools in your development:
- Which AI coding assistants did you use? (Cursor, GitHub Copilot, ChatGPT, etc.)
- How did AI help in different phases? (code generation, debugging, architecture decisions)
- What percentage of code was AI-assisted vs hand-written?
- Any interesting AI-generated solutions or optimizations?
- Challenges where AI couldn't help and you solved manually?

Technical Decisions
Explain your choice of technologies and trade-offs:
- Why did you choose OpenAI vs open-source models?
- Local LLM (Ollama) vs cloud API considerations?
- Performance vs cost vs privacy trade-offs?

Future Improvements
What would you add with more time?

Demo
Link to live demo or screenshots

💡 Development Tips

AI-Assisted Development

Feel free to use modern AI development tools:

- **Cursor AI, GitHub Copilot** for code generation
- **ChatGPT, Claude** for architecture planning
- **AI debuggers** for troubleshooting
- Document your AI usage process in README

Quick Start Options

- **FastAPI + Streamlit**: Recommended Python stack
- **FastAPI + HuggingFace + React**: Full-stack option
- **FastAPI + Ollama**: Complete offline solution (Llama, Mistral, Qwen)
- **Flask alternative**: If more familiar, but FastAPI preferred

Success Strategy

1. Start with basic keyword search, then enhance with AI
2. Use JSON files for quick data setup
3. Focus on working functionality over perfect code
4. Test with realistic HR queries
5. Document your AI development journey

🚀 Submission

1. **GitHub Repository**: Public repo with all code
2. **Demo Link**: Working application (Streamlit Cloud, Vercel, etc.) or clear local setup
3. **Email**: Send repo link + demo link
4. **Timeline**: Complete at your own pace

Note: This is a practical assessment of your ability to rapidly build AI-powered applications using modern development tools and practices. We're interested in both your technical skills and how you leverage AI assistance in development.