💼 **AI-Powered Job Recommendation System using LangChain & Gemini**

### **1. Problem Explanation**

Finding the right job or internship can be challenging for students due to:

* Overwhelming number of postings across platforms
* Difficulty in matching opportunities to personal skills and experiences
* Lack of personalized recommendations  
   This project solves the problem by **automating job discovery** using LLM-based RAG (Retrieval-Augmented Generation) techniques.

### **2. Use of RAG with LangChain & Gemini**

**RAG Process in the App:**

1. **Input Gathering** – User provides skills/experience via manual entry or resume upload.
2. **Retrieval** – Relevant job postings are fetched from Google Jobs API (via SerpAPI).
3. **Augmentation** – The retrieved job postings are added as contextual information.
4. **Generation** – LangChain passes skills + jobs to the Gemini LLM via Groq for generating top 5 personalized recommendations with explanations.

**Tech Flow Diagram** :

User → Resume Parser → Skills Extracted

Skills + Location → Job Scraper (Google Jobs)

Jobs + Skills → LangChain → Gemini LLM → Recommendations → Streamlit UI

### **3. Folder & Code Walkthrough**

job\_recommender\_ai\_1/

│

├── app.py # Streamlit app UI

├── requirements.txt # Dependencies

├── .env # API keys

│

├── src/

│ ├── job\_scraper.py # Fetches Google Jobs postings via SerpAPI

│ ├── resume\_parser.py # Extracts text from PDF/DOCX resumes

│ ├── recommender.py # LangChain + Gemini logic for recommendations

│ └── utils.py # Helper functions (currently empty)

│

├── data/ # Optional saved jobs or datasets

├── docs/ # Screenshots, diagrams

└── notebook/

### **4. Challenges Faced & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Parsing resumes with different formats | Used pdfplumber for PDF and python-docx for DOCX |
| API rate limits | Implemented minimal job fetch calls, cached results during testing |
| Matching jobs to user skills | Used LLM reasoning via LangChain prompt templates |
| Displaying clean UI | Styled Streamlit with emojis and sections for better readability |

### **5. Summary of Learnings**

* How to integrate **LangChain** with external APIs like **Groq Gemini**.
* How **RAG** can be applied in job recommendation systems.
* Working with **SerpAPI** to fetch structured job postings.
* Streamlit app design and deployment to Streamlit Cloud.
* Handling file parsing and preprocessing in Python.