**1. Problem Explanation**

The problem you aimed to solve is helping students find jobs or internships that match their skills, experience, and preferences more accurately and efficiently.

* Traditional job search platforms rely heavily on keyword matching, which often produces irrelevant results.
* Students may have limited experience in framing their skills in the right keywords for job portals.
* There’s a gap in personalization, where recommendations don’t adapt to a user's profile and changing market data

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

**RAG (Retrieval-Augmented Generation)** improves LLM performance by combining:

* **Retriever**: Finds relevant documents from a knowledge base (in your case, job postings and skills database).
* **Generator**: Uses LLM (Gemini) to answer questions based on retrieved documents.

**3. Challenges faced**

* **Challenge:** Integrating Gemini API with LangChain for RAG pipeline.  
  **Solution:** Followed API documentation closely, implemented proper authentication, and debugged API calls using test prompts.
* **Challenge:** Designing an intuitive Streamlit UI.  
  **Solution:** Iterated through multiple UI layouts, implemented interactive filters, and ensured a responsive design.
* **Challenge:** Ensuring recommendations match students’ skills and preferences.  
  **Solution:** Fine-tuned prompt templates and implemented skill-weighted scoring to improve accuracy.

**4. Summary**

* **LangChain RAG workflow** is extremely effective in combining real-time data with LLM reasoning.
* **Gemini API** can produce high-quality, conversational recommendations when grounded with factual context.
* Designing for **scalability** early (batch embedding, auto-updates) prevents future performance bottlenecks.
* A **clear folder/code structure** helps manage complexity in multi-component AI projects.