**1. Logic Behind Your Resume Recommendation System**

Your system’s purpose: **Recommend the top N resumes for a given job posting.**

It follows this pipeline:

**🔹 Input**

* job\_id (to identify the job in MongoDB)
* top\_n (number of resumes to recommend, default 5)

**🔹 Data Sources**

* **MongoDB (jobs collection)** → Contains job details like title, description, skills, required experience, and education level.
* **MongoDB (resumes collection)** → Contains metadata about resumes (student ID, name, etc.).
* **Cloudinary (PDF Resumes)** → Actual resumes are stored as PDFs.
* **Test Scores (MongoDB, if available)** → Candidate’s test scores.

**🔹 Feature Extraction**

For each resume:

1. **Skill Matching**
   * Extract text from the resume.
   * Compare required job skills with candidate’s skills using **semantic similarity** (SentenceTransformer embeddings).
   * Extra skills don’t reduce score, only required skills are checked.
2. **Experience Matching**
   * Check if candidate’s years of experience ≥ required.
   * Full score if matched/exceeded, partial score if less.
3. **Education Matching**
   * Map education levels to numeric ranks (10th < 12th < Bachelor < Master < PhD).
   * Full score if candidate meets/exceeds requirement.
4. **Test Score (if available)**
   * Normalized to contribute to final ranking.

**🔹 Scoring**

Each resume gets a **final weighted score**:

* **Test Score** (if available)
* **Skill Score** (semantic similarity)
* **Experience Score**
* **Education Score**

**🔹 Output**

* Ranked list of resumes (top N) with details:
  + Applicant name
  + Test score
  + Skill score
  + Experience score
  + Education score
  + Final score
  + Resume URL

**⚙️ 2. What Your Code Does Step by Step**

1. **FastAPI App Setup**
   * Creates an API using FastAPI (main.py).
   * Defines endpoint:
   * /recommend?job\_id=job123&top\_n=5
   * This calls recommend\_resumes(job\_id, top\_n).
2. **MongoDB Connection**
   * Connects to kgamify\_db.
   * Reads job details from jobs collection.
   * Reads applicant details from resumes collection.
3. **Cloudinary Config**
   * Configures Cloudinary with api\_key, cloud\_name, api\_secret.
   * Downloads resume PDFs.
4. **Resume Text Extraction**
   * Uses PyPDF2 to extract text from each PDF.
5. **Skill Matching**
   * Loads a **SentenceTransformer** model.
   * Embeds job skills + resume text.
   * Computes cosine similarity score.
6. **Experience & Education Matching**
   * Extracts candidate’s years of experience & degree from text.
   * Compares with job requirements.
   * Assigns normalized scores.
7. **Test Scores**
   * If test score exists in MongoDB, normalize and add to final score.
8. **Scoring & Ranking**
   * Combines skill, experience, education, and test scores.
   * Sorts resumes by final score.
9. **API Response**  
   Returns JSON like:
10. {
11. "job\_id": "job123",
12. "recommendations": [
13. {
14. "applicant\_id": "app1",
15. "name": "Rahul Sharma",
16. "test\_score": 82,
17. "skill\_score": 0.76,
18. "exp\_score": 1,
19. "edu\_score": 1,
20. "final\_score": 0.892,
21. "resume\_url": "https://cloudinary.com/.../rahul.pdf"
22. },
23. ...
24. ]
25. }

✅ **In short:**  
Your code builds an **AI-driven resume recommendation API**. It fetches job requirements, extracts resumes, applies semantic skill matching, checks experience & education, includes test scores, calculates a weighted ranking, and returns the top N resumes.