Kowshik - SE_1

🚀 Backend Developer Integration Guide

📌 Overview

This document provides a **step-by-step guide** for the **backend developer** to integrate the AI Resume Screening API with the **frontend & database**.

✅ **Supports ZIP uploads & Google Drive links for resumes**
✅ **Processes and ranks 1000+ resumes efficiently**
✅ **Exposes a FastAPI-based REST API for integration**
✅ **Saves processed results in JSON & CSV formats**

📌 1️⃣ Setup Backend on Server

🔹 1. Clone the Repository

```sh
git clone https://github.com/your-repo/resume-screening.git
cd resume-screening
```

🔹 2. Install Dependencies

```sh
pip install -r requirements.txt
```

🔹 3. Start the API Server

```sh
```

```
uvicorn api_service:app --host 0.0.0.0 --port 8000
```

🔹 **Backend API will now be available at:**

`http://localhost:8000` *(Local)*

`http://your-server-ip:8000` *(Production)*

📌 2️⃣ API Endpoints for Integration

The frontend should use the following **REST API endpoints** for integration.

🔹 1. Upload ZIP File & Process Resumes

Endpoint:

```http
POST /upload-resumes/
Content-Type: multipart/form-data
```

Body Parameters:

Parameter	Type	Description
`file`	`ZIP`	ZIP file containing resumes
`job_description`	`str`	Job description for relevance analysis
`weight_experience`	`int`	Weightage for experience (e.g., 3)
`weight_projects`	`int`	Weightage for projects (e.g., 2)
`weight_certifications`	`int`	Weightage for certifications (e.g., 1)

Example CURL Request:

```sh
```

```
curl -X 'POST' \
  'http://localhost:8000/upload-resumes/' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@resumes.zip' \
  -F 'job_description="Looking for a Data Scientist with expertise in Python,
Machine Learning, and NLP."' \
  -F 'weight_experience=3' \
  -F 'weight_projects=2' \
  -F 'weight_certifications=1'
```

📌 **API Response Example:**

```json
{
  "message": "Resumes processed & ranked successfully!",
  "ranked_candidates": "/ranked-candidates"
}
```

🔹 2. Get Ranked Candidates

Endpoint:

```http
GET /ranked-candidates
```

Example Request:

```sh
curl -X 'GET' 'http://localhost:8000/ranked-candidates'
```

📌 **API Response Example:**

```json
[
  {
    "filename": "resume1.pdf",
    "analysis": {
      "Key Skills": ["Python", "Machine Learning", "NLP"],
      "Experience Relevance": 9.1,
      "Projects Relevance": 8.5,
      "Certifications Relevance": 7.8,
      "Relative Ranking Score": 98.5
    }
  },
  {
    "filename": "resume2.docx",
    "analysis": {
      "Key Skills": ["Java", "Spring Boot", "Microservices"],
      "Experience Relevance": 7.2,
      "Projects Relevance": 6.8,
      "Certifications Relevance": 9.1,
      "Relative Ranking Score": 85.3
    }
  }
]
```

📌 3️⃣ Backend Database Integration (Optional)

The backend should store processed resumes in a **PostgreSQL or MongoDB** database.

🔹 1. PostgreSQL Integration

- **Install PostgreSQL:**

```sh
sudo apt update
sudo apt install postgresql postgresql-contrib
```

- **Create Database & Table:**

```sql
CREATE DATABASE resume_screening;
\c resume_screening;

CREATE TABLE candidates (
    id SERIAL PRIMARY KEY,
    filename TEXT,
    key_skills TEXT[],
    experience_relevance FLOAT,
    projects_relevance FLOAT,
    certifications_relevance FLOAT,
    relative_ranking_score FLOAT
);
```

- **Insert Data from API Response:**

```python
import psycopg2
import json

def save_to_db():
    with open("processed_data/ranked_candidates.json", "r") as f:
        data = json.load(f)

    conn = psycopg2.connect("dbname=resume_screening user=postgres
password=yourpassword")
    cur = conn.cursor()

    for candidate in data:
        cur.execute(
            "INSERT INTO candidates (filename, key_skills, experience_relevance,
projects_relevance, certifications_relevance, relative_ranking_score) VALUES (%s,
%s, %s, %s, %s, %s)",
            (candidate["filename"], candidate["analysis"]["Key Skills"],
            candidate["analysis"]["Experience Relevance"],
            candidate["analysis"]["Projects Relevance"],
            candidate["analysis"]["Certifications Relevance"],
            candidate["analysis"]["Relative Ranking Score"])
        )
    conn.commit()
    cur.close()
    conn.close()

save_to_db()
```

📌 4️⃣ Frontend Integration Guide

🔹 1. Upload ZIP from UI

1. **Frontend should provide a file upload button for resumes.**
2. **Once the user uploads a ZIP, send it to `/upload-resumes/` via `POST`.**
3. **Show the "Processing" status while resumes are being analyzed.**
4. **Redirect user to ranking page after processing.**

🔹 2. Display Ranked Candidates

1. **Frontend should fetch `/ranked-candidates/` via `GET`.**
2. **Display rankings in a table format with filters:**
   - Filter by **Skills**
   - Sort by **Ranking Score**
   - Show top candidates dynamically
3. **Provide "Download CSV" option for recruiters.**

📌 5️⃣ Deployment (Docker & Production)

🔹 1. Create `Dockerfile`

```dockerfile
FROM python:3.9
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "api_service:app", "--host", "0.0.0.0", "--port", "8000"]
```

🔹 2. Build & Run Container

```sh
docker build -t resume-screening-api .
docker run -p 8000:8000 resume-screening-api
```

🔹 3. Deploy on AWS/GCP

- Use **EC2 (AWS) or Compute Engine (GCP)**
- Set up **Nginx** as a reverse proxy to forward requests to FastAPI

📌 6️⃣ Error Handling & Debugging

🔹 1. Check FastAPI Logs

```sh
journalctl -u api_service --follow
```

🔹 2. Debug API Errors

If an API fails, check:

1. **Logs (`uvicorn --reload` will show errors).**
2. **Response Codes (should be 200 OK or 422 for validation errors).**
3. **Database connection (`psql resume_screening` to verify entries).**

📌 Final Deliverables

✅ `api_service.py` *(FastAPI backend)*
✅ `process_resumes.py` *(Resume processing)*
✅ `rank_candidates.py` *(Ranking script)*
✅ `requirements.txt` *(Dependencies)*
✅ `Dockerfile` *(For deployment)*
✅ `README.md` *(Documentation)*

🚀 Next Steps

Would you like me to: ✅ **Build a React.js UI for frontend integration?**
✅ **Deploy with Kubernetes for scalability?**
✅ **Optimize for AWS Lambda (Serverless deployment)?**

Let me know how you'd like to proceed! 🚀🔥