# Design Document: Resume and Job Description Analysis API

Overview

This API processes job descriptions and candidate resumes to evaluate the suitability of a candidate for a given job role. It extracts text from uploaded files (PDF, DOCX, TXT), analyzes the content using OpenAI's GPT-4o-mini model, and returns a structured JSON response with the evaluation.

Technology Stack

- **Backend:** Next.js API Routes
- **AI Service:** OpenAI GPT-4o-mini
- **File Parsing:** Mammoth (DOCX), PdfReader (PDF)
- **Environment Management:** dotenv

Endpoints

POST /api/analyze

Processes job descriptions and resumes, returning a structured JSON evaluation.

Request

- **Headers:** `Content-Type: multipart/form-data`
- **Body:**
    - `jobDescription`: (String) The job description (Minimum 50 characters).
    - `file`: (File) Candidate’s resume (DOCX, PDF, or TXT format).

Response

Success (200)

```
{
  "message": "Processing complete",
  "parsedJson": {
    "candidate": {
      "about": "Candidate Name",
      "skills": ["Skill 1", "Skill 2"],
      "short_description": ["Brief summary", "Key highlights"],
      "other_summary": {
        "experience": "X Years",
        "preferred_location": "Location",
        "current_company": "Company Name"
      },
      "evaluation": {
        "title": "Evaluation based on JD and Resume",
        "scores": {
          "Overall": 0-100,
          "Skill": 0-100,
          "Experience": 0-100,
          "Others": 0-100
        },
        "evaluation_reason": ["Reason 1", "Reason 2"]
      },
      "can_we_take_this_candidate": "yes" or "no"
    }
  }
}
```

-
- **Client Error (400):**
  - `{"error": "Job description must be at least 50 characters long"}`
  - `{"error": "No file uploaded"}`
- **Server Error (500):**
  - `{"error": "Failed to extract text from PDF"}`
- **Rate Limit Error (429):**
  - `{"error": "Too Many Request Token Limit Exit!"}`

Core Functionalities

1. Input Validation

- Ensures `jobDescription` has at least 50 characters.
- Ensures a file is uploaded.

2. File Processing

- **DOCX:** Extracts raw text using Mammoth.
- **TXT:** Reads the content directly.
- **PDF:** Uses PdfReader to extract text from each page.

3. AI Processing

- Sends the job description and extracted resume text to OpenAI's GPT-4o-mini.
- Follows a strict JSON response format for evaluation.

4. JSON Parsing & Response Handling

- Ensures a valid JSON response is received.
- If an error occurs in parsing, retries the AI request.

Error Handling

- Logs errors for debugging.
- Returns appropriate HTTP status codes and messages.

Security Considerations

- Uses environment variables (`dotenv`) for API keys.
- Validates file type before processing.
- Limits AI token usage to prevent excessive API costs.

Future Enhancements

- Support for additional file formats (e.g., ODT, RTF).
- Enhanced AI model for better candidate-job matching.
- Web-based UI for file uploads and visualization of results.

Data Flow Description

1. **User Uploads Data**
   - The user submits a **job description** and a **resume file** via a form.
2. **Input Validation**
   - The system checks if the job description is at least 50 characters long.
   - The system verifies the file type (DOCX, PDF, TXT).
3. **Text Extraction**
   - If the file is DOCX → Extract text using **Mammoth**.
   - If the file is PDF → Extract text using **PdfReader**.
   - If the file is TXT → Read as UTF-8 text.
4. **AI Processing (GPT-4o-mini)**
   - The system sends the **job description** and **extracted resume text** to OpenAI.
   - The AI generates a **structured JSON response** with evaluation scores.

5. **JSON Parsing & Response Handling**
    - The system parses the JSON response and sends it back to the user.
6. **Error Handling** (if any issue occurs)
    - Invalid input → Returns `400 Bad Request`.
    - File parsing failure → Returns `500 Internal Server Error`.
    - OpenAI token limit exceeded → Returns `429 Too Many Requests`.