AI-cruiter

Resume Parsing AI Application

Report

Introduction

In the rapidly evolving landscape of recruitment, efficiency and accuracy are paramount. Traditional hiring processes, often riddled with manual tasks, can be time-consuming and prone to oversight, especially when dealing with a large volume of resumes. To address these challenges, we present the **Resume Parsing AI Application**—an intelligent, AI-powered solution designed to revolutionize the recruitment process.

This application harnesses the power of **Gemini AI**, a state-of-the-art generative AI model, to parse resumes, extract key information, and analyse candidates against job requirements. The system processes resume in various formats (PDF, DOC, etc.) and organizes the extracted data into a structured database. Beyond parsing, it allows users to manage resumes seamlessly through CRUD (Create, Read, Update, Delete) operations and includes advanced search capabilities, such as skill-based filtering, to match job openings with the best-fit candidates.

A standout feature of the application is its **evaluation reporting system**, which provides detailed insights into candidate-job compatibility. Users can upload a job description (JD) in either DOC or PDF format or directly input the JD into a text box. The application then compares the candidate data against the uploaded or input JD, generating comprehensive evaluation reports. These reports offer rankings and tailored recommendations, empowering recruiters to make informed decisions based on how well each candidate's skills and experience align with the job requirements.

Designed for both efficiency and ease of use, this end-to-end system not only minimizes manual effort but also ensures a streamlined and data-driven recruitment workflow. Whether used by HR professionals or hiring managers, the Resume Parsing AI Application is poised to transform how organizations identify, evaluate, and select top talent.

The core of this system utilizes The **Gemini AI's 1.5 Flash model** to analyze resumes and extract meaningful data, significantly reducing manual effort and ensuring that the recruitment process is more efficient and data-driven.

Application Overview

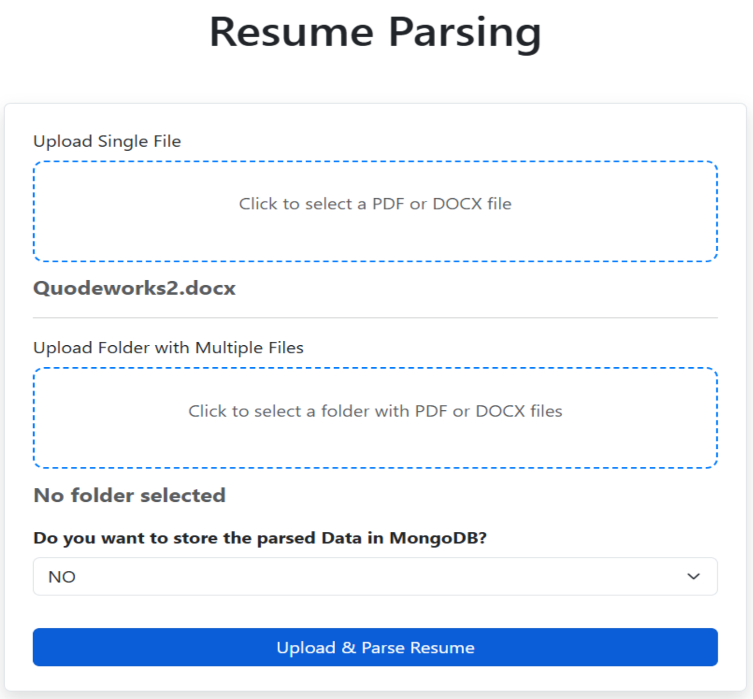
The Resume Parsing AI Application consists of both **backend** and **frontend** components, which work together to deliver a seamless and user-friendly experience.

1. Backend Pipeline

The backend handles the resume parsing and data extraction tasks, as well as the CRUD operations for resume data stored in MongoDB. The key components of the backend pipeline are:

Extractor.py:

- This Python script is the heart of the data extraction process. It uses **Gemini AI** and prompt engineering techniques to intelligently extract key information such as:
  - Name
  - Contact Information
  - Skills
  - Work Experience
  - Education Details
- The extracted data is stored in a structured format in MongoDB.



Resume Parsing

Upload Single File

Click to select a PDF or DOCX file

Quodeworks2.docx

Upload Folder with Multiple Files

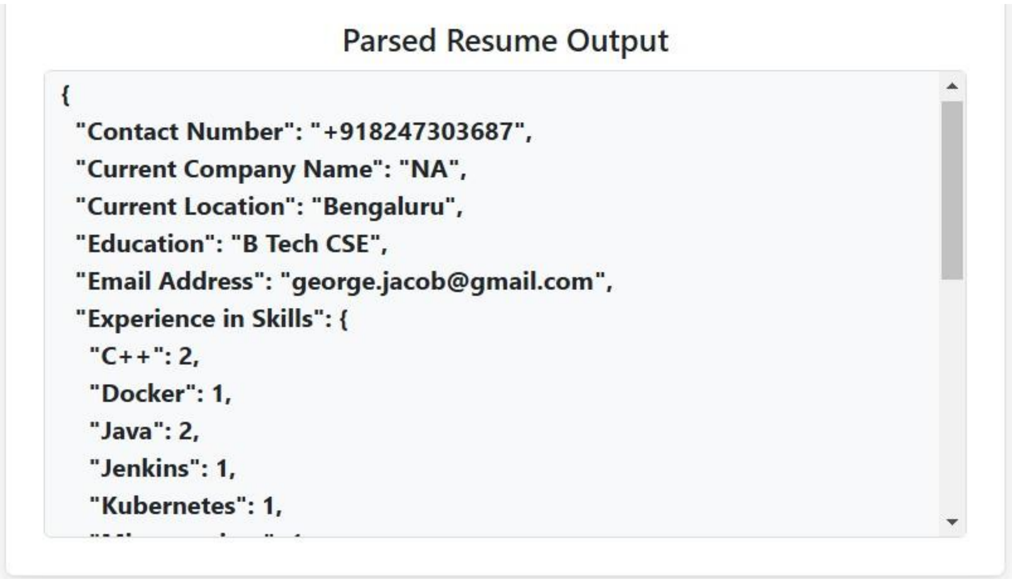
Click to select a folder with PDF or DOCX files

No folder selected

Do you want to store the parsed Data in MongoDB?

NO

Figure 1 Interface to upload file or a folder with resume files



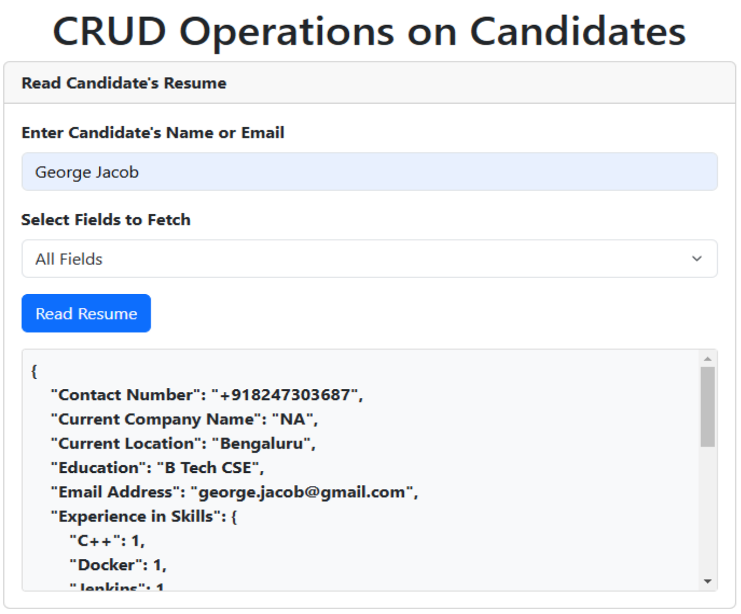
Parsed Resume Output

```
{
  "Contact Number": "+918247303687",
  "Current Company Name": "NA",
  "Current Location": "Bengaluru",
  "Education": "B Tech CSE",
  "Email Address": "george.jacob@gmail.com",
  "Experience in Skills": {
    "C++": 2,
    "Docker": 1,
    "Java": 2,
    "Jenkins": 1,
    "Kubernetes": 1,
```

Figure 2 Output from Resume Parsing

Resume Parsing CRUD (resume_parsing_CRUD.py):

- This script allows users to perform CRUD operations on the extracted resume data. Using MongoDB as the database, users can:
  - **Read** existing records (view extracted data).


CRUD Operations on Candidates
Read Candidate's Resume
Enter Candidate's Name or Email
Select Fields to Fetch
Read Resume

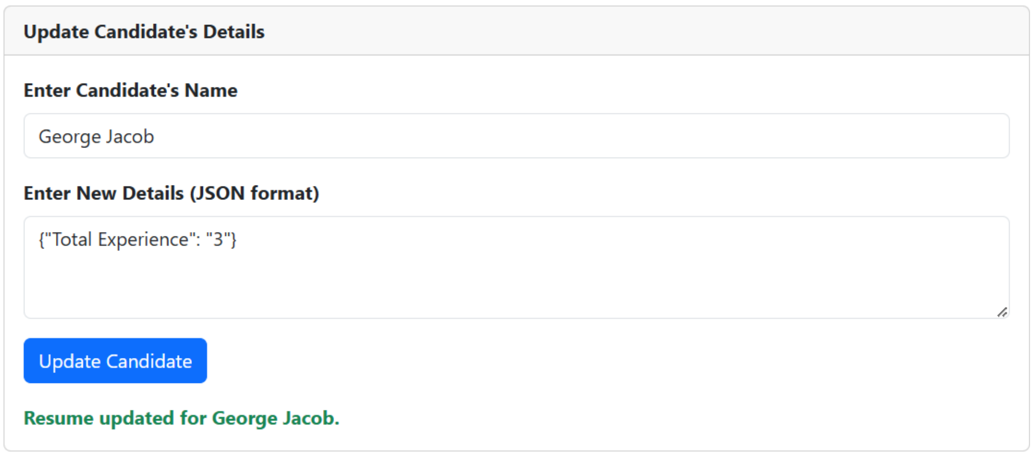
George Jacob

All Fields

```
{
   "Contact Number": "+918247303687",
   "Current Company Name": "NA",
   "Current Location": "Bengaluru",
   "Education": "B Tech CSE",
   "Email Address": "george.jacob@gmail.com",
   "Experience in Skills": {
      "C++": 1,
      "Docker": 1,
      "Jenkins": 1
```

Figure 3 Reading a candidate's resume

  - **Update** records (modify or correct extracted data).



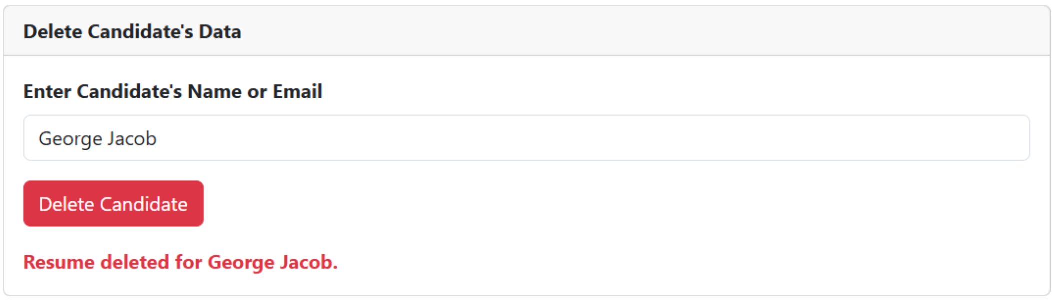
George Jacob

{"Total Experience": "3"}

Resume updated for George Jacob.

Figure 4 Output from updating a candidate's data

  - **Delete** records (remove data from the database).


Delete Candidate's Data

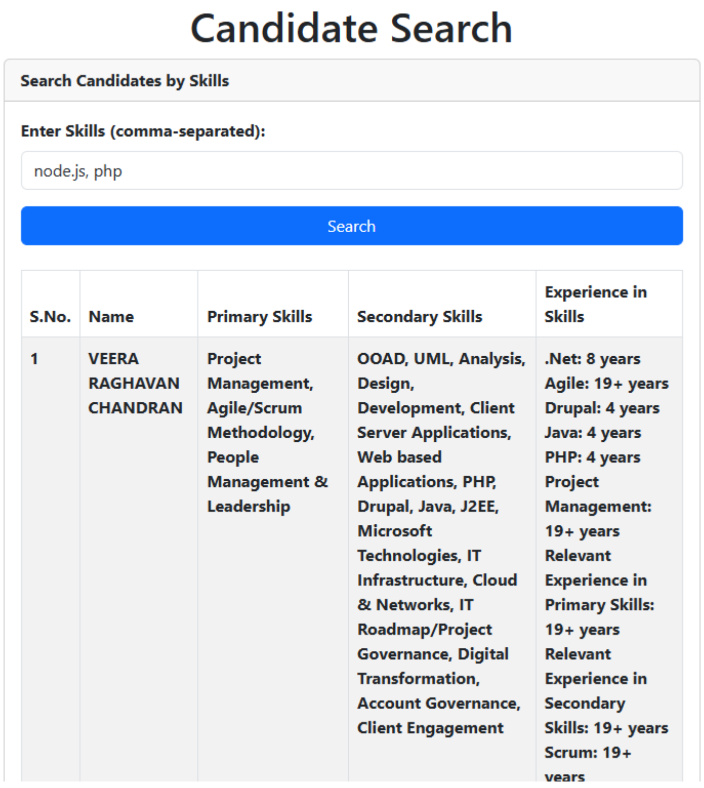
Enter Candidate's Name or Email

George Jacob

Resume deleted for George Jacob.

Figure 5 Deleting a candidate's resume data

➢ **Search a candidate by skills:**


Candidate Search

Search Candidates by Skills

Enter Skills (comma-separated):

node.js, php

S.No.	Name	Primary Skills	Secondary Skills	Experience in Skills
1	VEERA RAGHAVAN CHANDRAN	Project Management, Agile/Scrum Methodology, People Management & Leadership	OOAD, UML, Analysis, Design, Development, Client Server Applications, Web based Applications, PHP, Drupal, Java, J2EE, Microsoft Technologies, IT Infrastructure, Cloud & Networks, IT Roadmap/Project Governance, Digital Transformation, Account Governance, Client Engagement	.Net: 8 years Agile: 19+ years Drupal: 4 years Java: 4 years PHP: 4 years Project Management: 19+ years Relevant Experience in Primary Skills: 19+ years Relevant Experience in Secondary Skills: 19+ years Scrum: 19+ years

Figure 6 Searching a candidate by skills

The "Search Candidates by Skills" feature allows users to efficiently locate candidates within the MongoDB database who possess specific skills. This functionality is designed to streamline the recruitment process by quickly matching job requirements with qualified candidates.

Key Functionality:

1. **Input Skills:**
   Users can input one or more skills into the search field. These skills are used as search criteria to query the database.
2. **Database Query:**
   The MongoDB database is queried using these skills. The search leverages efficient filtering mechanisms to ensure that all matching candidates whose resumes include

the specified skills are retrieved.

3. **Result Presentation:**
   The retrieved candidate data is displayed in a user-friendly tabular format. The table includes key details such as:
   - Candidate Name
   - Skills
   - Experience
   - Contact Information
4. **Multi-Skill Matching:**
   The feature supports multi-skill searches, meaning candidates with all or any of the entered skills can be identified based on the search logic (AND/OR conditions).
5. **Real-Time Updates:**
   Results are dynamically updated as per the search input, providing instant feedback to the user.

- **Gemini AI** plays an essential role in **evaluating the best candidates**. The evaluation logic runs through the candidate's resume data and compares it to predefined job descriptions, helping to identify the most suitable candidates for a given position.
- The application generates detailed **evaluation reports**, providing insights and rankings of candidates based on their fit for the role.

2. Frontend Interface

a. The frontend of the application is built using **HTML**, **CSS**, and **JavaScript** with the help of the **Bootstrap** framework. The key elements of the frontend include:

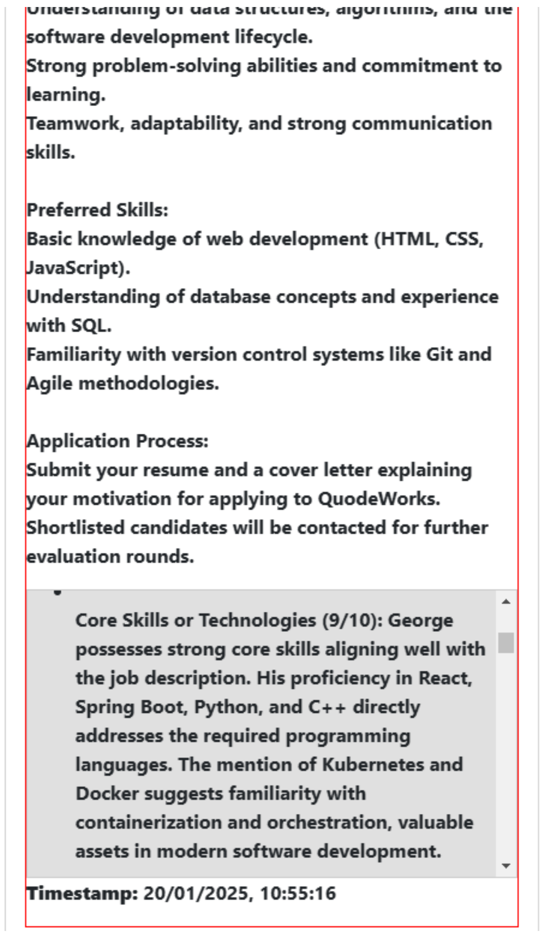
   i. **index.html**: This is the main user interface of the application, providing users with:

      1. An intuitive interface to upload resumes (PDF or DOC files).
      2. An interface to view and manage the parsed data (via CRUD operations).
      3. A section to generate **evaluation reports** based on the parsed resumes.
      4. **Forms for interacting** with the backend system to perform various tasks, including resume upload, data retrieval, editing, and deletion.

3. Flask Backend (app.py)

a. The **Flask** framework serves as the backend server, connecting the frontend to the backend pipeline. The app.py file serves as the main controller, managing routes for:

i. Uploading resumes.
ii. Parsing resumes using **Gemini AI**.
iii. Interacting with the MongoDB database for CRUD operations.
iv. Generating **evaluation reports** based on the parsed data.



Understanding of data structures, algorithms, and the software development lifecycle.
Strong problem-solving abilities and commitment to learning.
Teamwork, adaptability, and strong communication skills.

Preferred Skills:
Basic knowledge of web development (HTML, CSS, JavaScript).
Understanding of database concepts and experience with SQL.
Familiarity with version control systems like Git and Agile methodologies.

Application Process:
Submit your resume and a cover letter explaining your motivation for applying to QuodeWorks.
Shortlisted candidates will be contacted for further evaluation rounds.

- Core Skills or Technologies (9/10): George possesses strong core skills aligning well with the job description. His proficiency in React, Spring Boot, Python, and C++ directly addresses the required programming languages. The mention of Kubernetes and Docker suggests familiarity with containerization and orchestration, valuable assets in modern software development.

Timestamp: 20/01/2025, 10:55:16

Fig. 1.7. Evaluation report for Fresher Software Developer

The Flask application also handles interactions with the frontend, such as rendering HTML pages and passing data between the client and the backend.

How the Application Works

1. **Upload Resumes**:
   - Users can upload their resumes through the **frontend interface** (index.html).
   - The uploaded resumes, in either **PDF** or **DOC** format, are passed to the backend for processing.
2. **Parsing Resumes**:
   - The **Extractor.py** script uses **Gemini AI** along with **prompt engineering** to extract relevant information from the resumes.

   - Extracted data is structured and saved in a MongoDB database.
3. **CRUD Operations**:
   - Users can perform CRUD operations on the parsed resume data via the **resume_parsing_CRUD.py** file. This allows users to:
     - Upload, update, and delete resume entries.
     - View resume details stored in the database.
4. **Candidate Evaluation**:
   - **Gemini AI** analyzes the data for each resume and compares it to a predefined job description.
   - The AI generates an **evaluation report** ranking candidates based on their fit for the job position.
5. **Frontend Interaction**:
   - The **index.html** frontend serves as a bridge between the user and the backend.
   - Users can view parsed data, manage resumes, and download evaluation reports, all through a well-designed interface.

Technologies and Tools Used

1. **Backend:**
   - Language - Python
   - google-generativeai (for data extraction and evaluation by Gemini AI)
   - MongoDB (for storing and retrieving resume data)
   - Flask (for creating the web app)
   - PyPDF2 (for extracting data from PDFs)
   - python-docx (for extracting data from Word documents)
   - Flask-CORS (for handling cross-origin resource sharing)
   - json (for working with JSON data)
   - dotenv (for managing environment variables)
   - Google AI Studio (for Gemini API and documentation)
2. **Frontend:**
   - HTML, CSS, and JavaScript
   - Bootstrap (for responsive UI design)
   - index.html (main frontend page)
3. **Version Control and Development Environment:**
   - GIT (for version control)

- GitHub (for hosting the repository)
- Visual Studio Code (VS Code) (for development)

Project Setup Guide

1. Clone the Project Repository

- First, clone the repository to your local system.

  ```
  git clone <repository-url>
  cd <project-directory>
  ```

2. Set Up the Virtual Environment

- Open the terminal (or VS Code integrated terminal).
- Navigate to your project directory and run the following command to create a virtual environment: python -m venv .venv
- After the virtual environment is created, activate it:
  - **Windows:** .venv\Scripts\activate
  - **Mac/Linux:** source .venv/bin/activate

3. Install Dependencies

- Once the virtual environment is activated, install the required Python libraries by running: pip install -r requirements.txt

4. Set Up Configuration Files

- In the project directory, create a **config** folder.
- Inside the **config** folder, create a .env file to store sensitive information like API keys and database connection details.
- **.env file structure:**

  ```
  GEMINI_API_KEY=XXXXXXXX
  MONGO_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/test?retryWrites=true&w=majority
  DATABASE_NAME=resumes
  COLLECTION_NAME=your_collection_name
  ```

  - **GEMINI_API_KEY:** Replace with your own Gemini API key, which you can obtain from Google AI Studio.
  - **MONGO_URI:** Replace with your MongoDB Atlas connection string. You can get this from the MongoDB Atlas dashboard.
  - **DATABASE_NAME:** The name of the MongoDB database to store resume data and evaluation reports. Create a database named resumes and two collections: resume_database and evaluation_reports.

5. Configure Prompt Templates (Optional)

- To modify the instructions for the Gemini AI model, you can edit the prompt template files:
  - **Extractor Prompt Template:** The instructions for the resume data extraction process.
  - **Evaluation Prompt Template:** The instructions for the candidate evaluation process.
- Both prompt templates are stored as .txt files in the **config** folder. You can edit these files to adjust the prompts to your needs.

6. Running the Application Locally

- With all configurations set up, you can now run the Flask web application locally.
- In the terminal, run the following command to start the Flask server: python app.py
- The application will start running on http://127.0.0.1:5000/. Open this URL in your browser to access the app.

AI Integration in the Project

This project leverages **Gemini AI**, a powerful generative AI model, for key tasks in the backend pipeline, including **data extraction** from resumes and **report generation** for candidate evaluations.

1. **Data Extraction with Gemini AI:**
   The Gemini AI model is used to extract relevant data from resumes. By sending the resume content as input, the model identifies key information such as skills, work experience, education, and certifications. This data is then parsed and formatted into a structured format, which can be stored in a MongoDB database for further analysis or retrieval.
2. **Report Generation:**
   - Gemini AI is also utilized for generating **evaluation reports** based on a given job description and the resumes of candidates. The model analyzes the skills and experience of the candidates to assess how well they align with the job requirements. The evaluation report is generated in HTML format, which is stored in a MongoDB collection for future use.
3. **Prompt Engineering:**
   - **Prompt engineering** is a critical part of the project, as it defines the instructions sent to the Gemini AI model to guide its responses. The prompts are carefully crafted and stored in template files, allowing for easy modification. These templates include specific instructions for both resume data extraction and candidate evaluation, ensuring that the model's responses align with the desired format and content.
   - The **extractor prompt template** guides the model in extracting structured data from resumes, while the **evaluation prompt template** shapes the model's analysis of candidates against job descriptions.

Through careful prompt engineering and the integration of Gemini AI, the project enables automated and intelligent resume parsing and candidate evaluation, making the process faster, more accurate, and efficient.

Benefits of the Application

- **Efficiency**: Automates the tedious process of parsing resumes and extracting key information using advanced AI, which saves time for recruiters and HR professionals.
- **Data-driven Decisions**: The evaluation reports help identify the best-fit candidates based on data, reducing bias in the hiring process.
- **Ease of Use**: The Flask web app, integrated with MongoDB, provides an easy-to-use interface for managing resumes and candidate data.
- **Customization**: The system can be customized to meet specific job requirements and preferences, making it versatile for various recruitment scenarios.

Gaps and Concerns

While the Resume Parsing AI Application offers powerful features, there are areas where further refinement can improve the accuracy and adaptability of the evaluation process.

- First, the current evaluation framework may not fully account for the varied requirements across different job levels, such as entry, mid, and senior roles. To address this, it is essential to implement distinct rubrics or separate handling in the prompt for evaluation reports based on job level. This will ensure that the criteria for assessing candidates are appropriately adjusted for the complexity and expectations associated with each role.
- Another area for improvement is the weightage assignment to skills in the evaluation reports. Currently, all skills may be treated equally, but for a more accurate and tailored evaluation, assigning weightage to specific skills based on the job description and role requirements would enhance the precision of the rankings and recommendations. This would allow the AI to better reflect the relative importance of each skill, ensuring that candidates with key competencies are prioritized.

Conclusion

The Resume Parsing AI application leverages the power of **Gemini AI** to simplify the recruitment process. By automating the extraction and evaluation of resume data, the system saves recruiters valuable time and helps them make better-informed decisions. The web- based interface, powered by **Flask**, provides users with a seamless experience for managing resumes, while the backend logic ensures smooth data extraction, CRUD operations, and evaluation reporting. With its ability to process resumes efficiently and rank candidates according to their fit for the role, this application is a vital tool for improving the recruitment process in any organization.