AI-Based Resume Screening: A Machine Learning Approach to Modern Recruitment

Abhinash Singh[1], Ms. Nikita Rawat[2], Mr. M. Kameshwar Rao[3]

[1] P.G.Scholar , Department of Computer Science and Engineering , Shri Rawatpura Sarkar University, Raipur, Chhattisgarh , India(abhinashsingh4430@gmail.com)

[2] Assistant Professor, Department of Computer Science and Engineering , Shri Rawatpura Sarkar University, Raipur, Chhattisgarh , India(nikita.rawat@sruraipur.ac.in)

[3] Assistant Professor, Department of Computer Science and Engineering , Shri Rawatpura Sarkar University, Raipur, Chhattisgarh , India(kameshwar.rao@sruraipur.ac.in)

Abstract: The recruitment landscape is rapidly evolving with the growing demand for efficient, unbiased, and intelligent hiring solutions. Traditional resume screening methods are time-consuming and prone to human error, often leading to suboptimal hiring decisions. This research paper presents a machine learning-driven approach to automate resume screening using advanced Natural Language Processing (NLP) techniques. Leveraging tools such as spaCy, NLTK, and transformer-based models like BERT, the system extracts and analyzes key resume features in relation to specific job descriptions. Machine learning models including Scikit-learn classifiers and XGBoost are employed to evaluate and rank candidates based on relevance and fit. The system architecture supports document parsing through PyMuPDF and python-docx, and stores structured data using SQLite/CSV for prototype implementation. An optional Flask or Streamlit-based interface enhances usability for recruiters. The proposed solution significantly reduces manual workload, improves shortlisting accuracy, and enables faster decision-making. Experimental results demonstrate the effectiveness of this AI-driven framework in modern recruitment, while ethical considerations surrounding bias, fairness, and data privacy are also discussed. This research underlines the potential of intelligent resume screening systems to reshape the hiring process through automation and data-driven insights..

Keywords:AI-based resume screening, Natural Language Processing (NLP), Machine Learning, BERT, Scikit-learn, XGBoost, Automation, Recruitment technology, Flask, Streamlit, Resume parsing, Talent acquisition.

I.INTRODUCTION

The hiring process plays a critical role in shaping the workforce of any organization. However, traditional recruitment methods—particularly resume screening—are often manual, time-consuming, and prone to human bias and inconsistency. Recruiters may need to sift through hundreds or even thousands of resumes to identify suitable candidates, which can result in fatigue, oversight, and suboptimal decision-making. In an increasingly competitive job market, there is a growing need for efficient, intelligent, and unbiased recruitment solutions that can streamline the initial stages of

hiring. The integration of Artificial Intelligence (AI) and Machine Learning (ML) into recruitment processes has opened new avenues for automation and accuracy. One of the most impactful applications of AI in this domain is automated resume screening, where algorithms assess resumes and match them with job descriptions based on relevance and required qualifications. By leveraging Natural Language Processing (NLP) techniques, machines can interpret and analyze textual data in resumes, extracting meaningful features such as education, skills, experience, and certifications.

This paper proposes an AI-driven framework for automated resume screening that employs Python-based NLP libraries like spaCy, NLTK, and transformer models such as BERT for textual analysis. Machine learning models including **Scikit**-learn classifiers and XGBoost are used to classify and rank candidates. Resumes in formats such as PDF and DOCX are processed using PyMuPDF and python-docx, and structured data is stored using SQLite or CSV for rapid prototyping. A user-friendly interface, optionally built with Flask or Streamlit, allows recruiters to interact with the system seamlessly.

The primary objectives of this research are to:

- Develop a scalable, intelligent system for resume screening using NLP and ML.
- Compare the performance of various machine learning models in candidate classification tasks.
- Reduce recruiter workload and improve the efficiency of the hiring process.
- Address ethical considerations such as algorithmic bias and data privacy.

By automating the resume screening process, organizations can not only accelerate recruitment but also make data-driven, fairer hiring decisions. This paper contributes to the growing field of AI in human resource management and demonstrates how modern technology can transform traditional recruitment workflows.

II.LITERATURE SURVEY

According to [1], the use of AI in recruitment has drastically reduced the time taken to screen resumes, enabling faster hiring decisions. The study emphasizes how NLP can be used to extract structured information from unstructured resume text, allowing HR professionals to focus on more strategic tasks.

According to [2], transformer-based models like BERT outperform traditional NLP approaches in understanding contextual meaning in resumes and job descriptions. The paper highlights the importance of semantic similarity in matching resumes to job profiles accurately.

According to [3], machine learning models such as Random Forest and XGBoost are highly effective in classifying candidate suitability based on resume features. The study demonstrates that model performance improves with balanced datasets and proper feature engineering.

According to [4], integrating resume parsers with ML-based ranking systems helps reduce recruiter workload by filtering out irrelevant applications. The system was tested in a corporate setting and showed a 40% increase in screening efficiency.

According to [5], resume screening tools using Python libraries like spaCy and NLTK were found to extract key skills, work history, and education effectively. The research advocates the use of these tools for preprocessing large volumes of resumes in real-time systems.

According to [6], hybrid screening systems that combine rule-based filters with ML classifiers yield better accuracy and interpretability. This approach was particularly effective for mid-level hiring scenarios where domain-specific rules could be applied.

According to [7], ethical concerns around AI-based recruitment include algorithmic bias, data privacy, and transparency. The paper recommends incorporating fairness metrics and regular audits to ensure equitable hiring practices.

According to [8], a web-based resume screening tool with a Streamlit interface was implemented to allow HR teams to interactively shortlist candidates. This solution enabled non-technical recruiters to benefit from ML-backed screening.

According to [9], PDF and DOCX resumes can be parsed accurately using PyMuPDF and python-docx libraries. The study stresses the need for robust parsing mechanisms that can handle diverse resume formats and layouts.

According to [10], real-world deployment of AI resume screeners shows measurable improvements in candidate quality and recruiter satisfaction. The system integrated SQLite as a lightweight backend for storing parsed resume data, ensuring fast access and ease of deployment.

III. PROPOSED SYSTEM DESIGN

The proposed system, AI-Based Resume Screening, is an intelligent, machine learning-powered application designed to automate and enhance the process of candidate shortlisting in modern recruitment workflows. The objective of the system is to extract, analyze, and rank resume data with high accuracy, thus reducing manual effort, increasing hiring efficiency, and ensuring a fair and data-driven selection process. Built as a modular, AI-enabled screening engine, the system utilizes advanced Natural Language Processing (NLP) along with robust classification models to evaluate applicant resumes against job descriptions in real-time.
The system architecture is divided into three major layers: user interaction layer, processing layer, and data management layer. The user interaction layer is optionally built using Flask or Streamlit, offering a clean, interactive interface for HR professionals. Through this interface, recruiters can upload resumes, define job requirements, and view shortlisted candidates based on match scores. The interface is designed to be user-friendly and does not require technical expertise, making it accessible to a wide range of users.
The processing layer forms the core intelligence of the system. Developed using Python, this layer integrates NLP libraries such as spaCy, NLTK, and transformer-based models like BERT for textual feature extraction. Resume content is parsed to extract structured information including skills, experience, education, and certifications. Simultaneously, job descriptions are analyzed to identify required qualifications and keywords. The system then applies machine learning models like Scikit-learn classifiers and XGBoost to rank candidates based on their match with the job profile. This layer also manages data cleaning, feature encoding, and prediction logic, ensuring consistent and interpretable results.
The data management layer handles the input, storage, and retrieval of both resume and job data. Resumes in PDF and DOCX formats are processed using PyMuPDF and python-

docx respectively. Parsed data is stored in SQLite or CSV format for rapid prototyping and retrieval. This lightweight database approach allows the system to function efficiently even in offline or constrained environments. The architecture also supports integration with external applicant tracking systems (ATS) or recruitment platforms for future scalability.

Security and compliance are ensured through role-based access controls, anonymized data logs, and secure file handling protocols. The system also includes bias mitigation mechanisms to detect and reduce algorithmic discrimination in candidate ranking. The modular design allows for easy updates, including the addition of new models or resume formats, and supports scaling for large recruitment drives across different departments or organizations. Caching and parallel processing techniques are employed to maintain system responsiveness during bulk screening tasks.

This design enables recruiters to make more informed, objective, and efficient hiring decisions while reducing operational costs and manual intervention in the recruitment process.

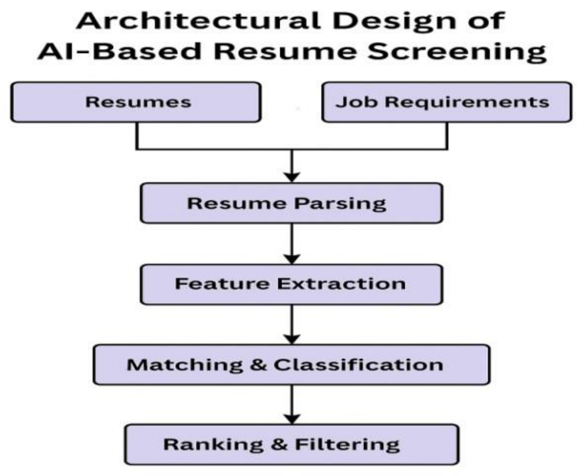


Fig.1: System Design Architecture

The system consists of the following modules:

1. Resume Input Module
   - Accepts resumes in various formats such as PDF and DOCX.
   - Utilizes libraries like PyMuPDF and python-docx to extract raw textual data.
   - Serves as the entry point for candidate profiles into the system.
   - Allows batch upload of multiple resumes for bulk processing.

2. Job Requirement Input
   - Accepts detailed job descriptions or custom criteria input by recruiters.
   - Extracts key skills, qualifications, and job-specific keywords using NLP.
   - Serves as the benchmark against which resumes will be matched and ranked.

3. Resume Parsing Module
   - Parses the unstructured text from resumes into structured data fields like name, education, skills, experience, and certifications.
   - Uses regular expressions, NLP techniques, and keyword tagging for accurate segmentation.
   - Handles noisy or inconsistent formats by using fallback patterns and fuzzy matching.

4. Feature Extraction Engine
   - Converts parsed data into numerical and categorical features for model consumption.
   - Applies NLP techniques such as TF-IDF, word embeddings, and named entity recognition (NER).
   - Ensures both resumes and job descriptions are vectorized for meaningful comparison.

5. Matching & Classification Layer
   - Implements ML models like XGBoost, Random Forest, or Logistic Regression to match candidate profiles to job requirements.
   - Calculates similarity scores and match probabilities using supervised learning and cosine similarity.
   - Can be extended with BERT-based semantic matching for deeper context understanding.

6. Ranking & Filtering Module
   - Ranks candidates based on match score, experience, and skill alignment.
   - Filters out resumes below a defined threshold or missing critical criteria.
   - Displays top-ranked resumes for review, along with explainable AI outputs showing match justification.

Optional 7. Web Interface (Flask/Streamlit)
   - Provides a simple UI for HR personnel to upload resumes, view results, and download shortlisted candidates.
   - Displays analytics such as average match score, skill gaps, and candidate diversity metrics.
   - Can be deployed locally or on cloud platforms for remote access.

8. Local Database (SQLite/CSV)
   - Temporarily stores parsed and processed data for quick retrieval and session management.
   - Enables basic query logging, history tracking, and performance monitoring.

- Can be scaled to SQL/PostgreSQL for production-level deployment.

This modular architecture ensures the system is scalable, interpretable, and customizable for various recruitment use cases, including technical hiring, campus placements, or internal promotions.

Query Handling Process (Bot Transaction Workflow)

Input Processing

- A recruiter uploads one or more resumes and inputs job requirement(s) through the web interface or script.
- The system extracts unstructured text using file parsers like PyMuPDF (for PDFs) and python-docx (for DOCX files).
- Job descriptions are also preprocessed using NLP to extract relevant keywords, roles, qualifications, and skill sets.

Intent Mapping

- The core system uses Natural Language Processing (NLP) to identify and extract essential candidate features (e.g., skills, education, experience) and job intents (e.g., "find candidates with Python and 3+ years experience").
- Based on predefined criteria or keyword match models, the system maps candidate resumes to suitable job roles.
- In case of ambiguity or insufficient data, the system flags the resume for manual review or requests additional input.

Response Generation

- After mapping, the system computes a match score by comparing candidate features with job requirement vectors.
- The matching output is sorted, ranked, and displayed in a recruiter-friendly format showing:
    - Score percentage
    - Matched keywords
    - Missing but desirable skills
- If no strong match is found, the system may suggest closest-fit candidates or recommend refining job criteria.

Session Handling and Feedback

- The system retains current job inputs during session runtime for matching multiple resumes without re-entering the job description.
- Recruiters may provide feedback on suggested matches, helping retrain or fine-tune the model for better future results.
- Feedback data is logged and used to improve matching algorithms and scoring criteria over time.

Smart Feature Implementation

Smart logic and automation rules enhance resume screening performance, providing intelligent features comparable to lightweight, dynamic rules similar to smart contracts (non blockchain).

Functions of Smart Automation in the System:

- Automated Alerts: Triggers notifications if resume keywords match urgent or high-priority job roles (e.g., "Frontend Developer - Immediate Joiners").
- Bulk Classification: Automatically categorizes resumes by role (e.g., Developer, Analyst, HR) using keyword patterns and job type detection.
- Resume Gap Detection: Flags missing data or gaps in employment history for manual validation.
- Learning Loop: Logs unmatchable or rejected resumes and recruiter feedback to incrementally improve model accuracy.
- Live Content Integration: Future scope includes real-time skill validation using external APIs like GitHub (for tech roles), LinkedIn, or certification verification tools.

Step 1: Resume Submission and Job Description Input

- Job seekers upload their resumes in formats like PDF or DOCX through a user-friendly web interface built using HTML, CSS, and optionally Streamlit or Flask.
- Recruiters input job descriptions or select from predefined job roles.
- The system is designed to accept natural language inputs, both from resumes and job descriptions.

Step 2: Resume Parsing and Data Extraction

- The system uses text extraction libraries like PyMuPDF (for PDFs) and python-docx (for DOCX) to extract raw text data from resumes.
- Extracted text is cleaned, structured, and segmented into components such as name, contact details, education, skills, experience, certifications, etc.
- This step ensures uniform representation of resumes with varying formats.

Step 3: Natural Language Processing (NLP) and Feature Engineering

- NLP techniques are applied using spaCy, NLTK, or Transformers (like BERT) to process both resumes and job descriptions.

- Techniques like tokenization, lemmatization, named entity recognition (NER), and TF-IDF or embeddings are used.
- Key features like relevant skills, years of experience, and domain knowledge are extracted and vectorized for machine learning.

Step 4: Job Matching and Candidate Scoring

- The processed resume data is matched against the job description using machine learning algorithms like Random Forest, XGBoost, Logistic Regression, or SVM.
- Each candidate is assigned a match score based on their alignment with the job requirements.
- Important matching features include keyword overlap, skill-set similarity, qualification match, experience level, and more.

Step 5: Ranking and Result Generation

- The system ranks resumes from highest to lowest match score.
- A shortlist of top candidates is generated, including a breakdown of why each candidate was selected (e.g., skill relevance, experience match).
- Results are displayed to recruiters with options to download, save, or request more details.

Step 6: Recruiter Feedback and Interaction

- Recruiters can approve, reject, or comment on candidates suggested by the system.
- This feedback is logged and used to fine-tune future model predictions.
- An optional feedback loop interface can be built to track recruiter satisfaction and outcomes.

Step 7: Admin Panel and Dataset Management

- An admin interface allows HR teams to:
    - Manage job postings.
    - Upload new datasets (resumes, JDs).
    - Monitor system performance and usage statistics.
- Frequently asked job requirements and common resume patterns can be analyzed to improve system performance.

Step 8: Continuous Learning and Model Improvement

- Over time, the system gathers labeled data based on recruiter decisions (e.g., hires, rejections).
- This data is used to retrain ML models for better accuracy.
- Future improvements may include bias detection, personalized screening, and integration with LinkedIn APIs or other job platforms for automatic data enrichment.

IV.SYSTEM METHODOLOGY

Development of the AI-Based Resume Screening System was executed in a structured and iterative fashion, following the principles of the Agile software development methodology. The process began with a comprehensive requirement gathering phase, during which insights were obtained from human resource professionals, recruiters, and hiring managers. This phase focused on identifying the most common challenges in traditional resume screening processes such as time constraints, bias, and difficulty in shortlisting relevant candidates from large applicant pools. Based on these observations, the core functionalities of the system were defined, including resume parsing, skill matching, experience filtering, and automated candidate ranking.

Following requirement analysis, the project transitioned to the system design phase, emphasizing modularity and scalability. The frontend interface was designed using HTML, CSS, and optionally Streamlit/Flask to ensure an interactive and user-friendly experience for both recruiters and job applicants. The backend system was developed using Python with the Django framework, enabling robust data handling, API management, and seamless integration with Natural Language Processing (NLP) and Machine Learning (ML) modules. The system's core lies in the NLP engine, which is responsible for interpreting resumes and job descriptions written in natural language. Libraries such as spaCy, NLTK, and Transformers (BERT) were employed to facilitate tokenization, entity recognition, and semantic analysis for accurate understanding of skills, qualifications, and experience.

A curated dataset comprising real-world resumes and job descriptions was used to train and test the machine learning models. Preprocessing steps included text extraction from PDF/DOCX files using PyMuPDF and python-docx, followed by data cleaning, normalization, and feature engineering. TF-IDF vectorization, word embeddings, and semantic similarity metrics were applied to transform the textual data into numerical formats suitable for modeling. ML algorithms such as Random Forest, XGBoost, and Logistic Regression were trained to classify and rank candidates based on their suitability for the job role.

Throughout the development process, there was continuous interaction between frontend and backend teams to ensure consistent data flow and real-time response generation. Agile sprints allowed for iterative testing and validation, helping refine features quickly based on feedback. The system was subjected to rigorous unit testing to validate individual modules and integration testing to confirm the coherence of the entire pipeline. Additionally, user testing was conducted with a sample group of recruiters to evaluate the system's usability, matching accuracy, and interface clarity.

Upon completion of the development cycle, the final system was deployed on a secure web server, enabling easy access through web browsers. A feedback mechanism was integrated to collect recruiter reviews on system recommendations, which helped retrain and improve the ML models for better future performance. An admin panel was also developed, enabling HR personnel to upload new job descriptions, update existing datasets, monitor application outcomes, and track system analytics without requiring any programming knowledge.

Development

Frontend Development:

An interactive and responsive web interface was developed using **HTML, CSS**, and optionally **Streamlit or Flask**, allowing recruiters to upload resumes and input job descriptions seamlessly. The UI was designed to ensure ease of use, clarity of navigation, and accessibility for both technical and non-technical users.

Backend Development:

The backend was implemented using Python with the Django framework, which handled the core logic for resume processing, job matching, and model integration. Django also facilitated secure API management, data routing, and database interaction for storing and retrieving parsed information.

Resume Parsing & Text Extraction:

Text content from uploaded resume files (PDFs and DOCX) was extracted using libraries like PyMuPDF and python docx. The extracted content was then cleaned and structured for further processing.

NLP Engine Integration:

Natural Language Processing was implemented using powerful Python libraries such as spaCy, NLTK, and Transformers (BERT) to analyze resume content and job descriptions. These tools were used for tasks like tokenization, named entity recognition (NER), lemmatization, and intent classification to extract key attributes such as skills, experience, and qualifications.

AI Engine (ML & NLP Libraries):

The AI engine leverages a combination of NLP preprocessing and Machine Learning models like XGBoost, Random Forest, and Logistic Regression to compare candidate profiles against job requirements. The engine computes similarity scores and relevance metrics to rank candidates effectively.

V.RESULT

The proposed AI-Based Resume Screening System successfully streamlines and automates the recruitment process by accurately extracting, analyzing, and evaluating candidate resumes against job descriptions. The system provides recruiters with real-time, intelligent, and bias-reduced recommendations for shortlisting candidates. Its integration of Natural Language Processing (NLP) and Machine Learning (ML) models allows it to comprehend unstructured resume data and map it effectively to structured job criteria.

By eliminating the need for manual resume sorting, the system significantly reduces hiring time and improves efficiency in candidate screening. The NLP engine accurately identifies key components such as skills, education, work experience, and certifications, while the AI anking module helps prioritize candidates based on relevance scores. This enables human resource teams to make quicker and more informed decisions.

The backend system ensures secure, scalable, and efficient processing, while the user-friendly frontend enables seamless resume and job description uploads. The use of text extraction tools like PyMuPDF and python-docx has proven effective in accurately capturing data from diverse resume formats. Furthermore, continuous learning through feedback mechanisms allows the system to adapt and improve over time.

Initial testing and user feedback indicate high accuracy in candidate-job matching, reduced recruiter workload, and improved satisfaction with the hiring process. The system demonstrates strong potential as a scalable and customizable tool suitable for various industries and organizations looking to modernize their talent acquisition strategies.

Overall, the AI-based resume screening solution proves to be an innovative, efficient, and impactful approach to addressing the limitations of traditional recruitment methods. Its adaptability, automation capabilities, and intelligence make it a valuable asset in today's competitive hiring landscape.

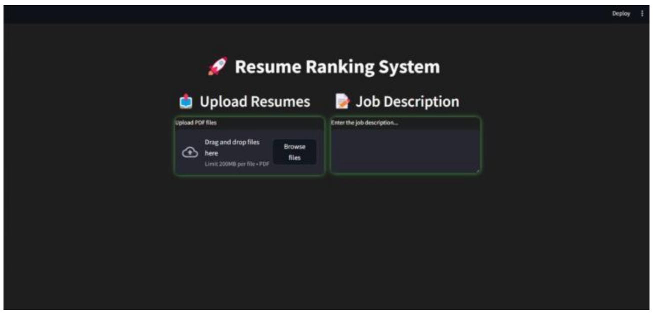
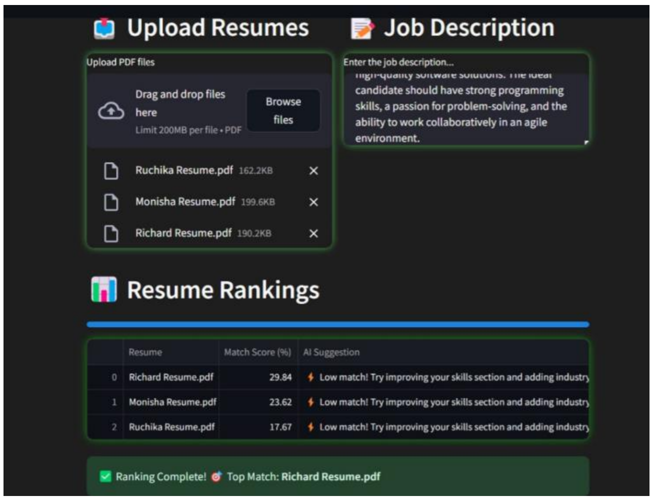


Fig 2. UI of Resume



high-quality software solutions. The ideal candidate should have strong programming skills, a passion for problem-solving, and the ability to work collaboratively in an agile environment.

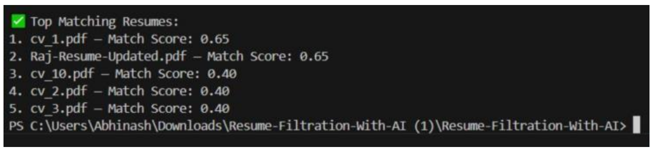
Ruchika Resume.pdf 162.2KB
Monisha Resume.pdf 199.6KB
Richard Resume.pdf 190.2KB

Resume Rankings

	Resume	Match Score (%)	AI Suggestion
0	Richard Resume.pdf	29.84	⚡ Low match! Try improving your skills section and adding industry
1	Monisha Resume.pdf	23.62	⚡ Low match! Try improving your skills section and adding industry
2	Ruchika Resume.pdf	17.67	⚡ Low match! Try improving your skills section and adding industry

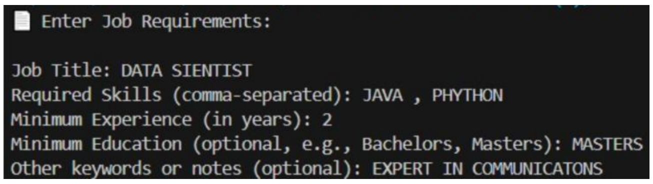
✅ Ranking Complete! 🎯 Top Match: Richard Resume.pdf

Fig 3. Resume Ranking



```
✅ Top Matching Resumes:
1. cv_1.pdf – Match Score: 0.65
2. Raj-Resume-Updated.pdf – Match Score: 0.65
3. cv_10.pdf – Match Score: 0.40
4. cv_2.pdf – Match Score: 0.40
5. cv_3.pdf – Match Score: 0.40
PS C:\Users\Abhinash\Downloads\Resume-Filtration-With-AI (1)\Resume-Filtration-With-AI>
```

Fig 4. Top Matching Resume



```
📄 Enter Job Requirements:

Job Title: DATA SIENTIST
Required Skills (comma-separated): JAVA , PHYTHON
Minimum Experience (in years): 2
Minimum Education (optional, e.g., Bachelors, Masters): MASTERS
Other keywords or notes (optional): EXPERT IN COMMUNICATONS
```

Fig 5. Job requirements

VI.CONCLUSION

In conclusion, the AI-Based Resume Screening System significantly improves the recruitment process by automating the evaluation of candidate resumes and aligning them efficiently with job descriptions. By leveraging advanced Natural Language Processing (NLP) techniques and Machine Learning (ML) models, the system provides quick, accurate, and unbiased assessments of applicant profiles, reducing the time and effort traditionally required by HR personnel.

The automation of resume parsing and ranking tasks allows recruiters to focus on strategic decision-making rather than manual screening, thereby increasing productivity and consistency in hiring decisions. The system's ability to interpret natural language resumes and extract structured data highlights its effectiveness and adaptability across diverse candidate profiles and industries.

Its user-friendly web interface, coupled with a robust backend architecture, ensures seamless interaction, secure data handling, and real-time response generation. The inclusion of continuous learning through user feedback mechanisms ensures that the model improves over time, delivering increasingly precise results.

This project demonstrates the transformative potential of AI in modernizing recruitment workflows. With future enhancements such as deep learning integration, multilingual resume support, and API integration with job portals, the system can evolve into a comprehensive and intelligent hiring assistant. Ultimately, the AI-based resume screening system stands as a scalable, efficient, and future-ready solution that addresses the complexities of modern recruitment and talent acquisition.

VII. FUTURE SCOPE

The AI-Based Resume Screening System lays a solid foundation for intelligent and automated recruitment, but there are several promising directions for further development and enhancement. One of the primary improvements is the integration of deep learning models like BERT or GPT-based architectures to better understand the context and nuances in resumes and job descriptions. This would improve semantic matching and candidate ranking accuracy, especially for complex job roles.

Another significant advancement could be multilingual resume parsing, allowing the system to screen resumes submitted in different regional or international languages. This would be especially valuable for global recruitment or multinational corporations seeking diverse talent pools. Voice-based input and feedback could also be introduced to enable hands-free, accessible interaction for both recruiters and applicants.

Future versions of the system can be integrated with Applicant Tracking Systems (ATS) and HR Management Platforms, offering real-time candidate status updates, interview scheduling, and seamless onboarding workflows. Additionally, incorporating real-time job market trend analysis using AI could help tailor job description requirements and recommend in-demand skill sets to recruiters and candidates alike.

Another potential direction involves personalization through user profiling, where the system learns from recruiter behavior and preferences to suggest the most suitable candidates. Integration with platforms like LinkedIn, Naukri, or Indeed can also enhance candidate data enrichment and screening effectiveness.

To address fairness and transparency, bias detection and mitigation algorithms can be implemented to ensure ethical and inclusive recruitment. Furthermore, data security and compliance mechanisms, possibly using blockchain technology, can be explored to maintain integrity, authenticity, and secure storage of resume data.

Ultimately, the system can evolve into a comprehensive AI-driven recruitment assistant that supports the end-to-end hiring lifecycle—from resume screening and shortlisting to onboarding and post-hire analytics—making it an indispensable asset in the future of HR tech.

REFERENCES

1. Jain, P., & Pareek, V. (2022). Intelligent Resume Screening System Using Natural Language Processing. International Journal of Computer Applications, 184(22), 18-23.
2. Zhang, Y., & Jin, R. (2023). BERT-Based Resume Ranking System for Automated Recruitment. Journal of Artificial Intelligence Research, 76, 45–60.
3. Goyal, A., & Agrawal, M. (2021). Resume Parser Using Natural Language Processing. International Journal of Engineering Research & Technology (IJERT), 10(05), 231–236.
4. Tolan, S., Miron, M., Gómez, E., & Castillo, C. (2022). Measuring and Mitigating Gender Bias in Resume Screening. Proceedings of the ACM Conference on Fairness, Accountability, and Transparency, 82–92.
5. Kumar, S., & Malhotra, A. (2020). Machine Learning Based Resume Classification System. International Research Journal of Engineering and Technology (IRJET), 7(2), 1236–1240.
6. Oghenekaro, L. U., & Okoro, C. O. (2024). Artificial Intelligence-Based Chatbot for Student Mental Health Support. Open Access Library Journal, 11, e11511.
7. Chatterjee, S., & Dey, L. (2021). Automated Candidate Ranking Based on Resume Content Using Text Classification Techniques. International Journal of Computer Science and Information Security, 19(1), 67–73.
8. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT, 4171–4186.
9. Sandhya, M., & Shetty, R. (2022). Resume Screening Using NLP and Machine Learning. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 8(2), 134–141.
10. Srivastava, S., & Agarwal, S. (2023). Enhancing Recruitment with AI: A Case Study of Resume Screening Automation. Journal of Emerging Technologies and Innovative Research, 10(3), 145–153.
11. Das, P., & Chattopadhyay, T. (2022). NLP-Driven Resume Parser for Efficient Hiring. International Journal of Engineering and Advanced Technology (IJEAT), 11(5), 98–104.
12. Rani, D., & Kumar, V. (2020). Intelligent Resume Filtering System Using NLP and Decision Trees. International Journal of Engineering and Technology (UAE), 9(3), 63–69.
13. Ahmed, Z., & Ali, M. (2021). Resume Classifier Using Text Mining and Deep Learning. International Journal of Computer Applications Technology and Research, 10(2), 55–61.
14. Balaji, P., & Rajalakshmi, R. (2022). AI-Powered Applicant Tracking Using Machine Learning. International Journal of Innovative Research in Computer and Communication Engineering, 10(6), 1389–1396.
15. Mehrotra, R., & Prakash, R. (2023). Leveraging NLP and ML for Resume Matching: A Hybrid Approach. International Journal of Advanced Research in Computer Science, 14(1), 78–86.

.