

A
Industrial-Oriented Mini Project
On
**AUTOMATED RESUME ANALYSIS & SKILL MATCHING
WEBSITE USING NLP**

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING
By

V Venkata Sai Anand	(227R1A0559)
A Ambika	(227R1A0501)
K Sirivalli	(227R1A0531)

Under the Guidance of
Dr. J. NARASIMHARAO
(Associate Professor)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR TECHNICAL CAMPUS
UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)
Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,
Kandlakoya (V), Medchal Road, Hyderabad-501401.

June, 2025.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project entitled “**AUTOMATED RESUME ANALYSIS & SKILL MATCHING WEBSITE USING NLP**” being submitted by **V. Venkata Sai Anand (227R1A0559), A. Ambika (227R1A0501) & K. Sirivalli (227R1A0531)** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, during the year 2024-25.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. J. Narasimharao
Associate Professor
INTERNAL GUIDE

Dr. Nuthanakanti Bhaskar
HoD

Dr. A. Raji Reddy
DIRECTOR

Signature of External Examiner

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project, we take this opportunity to express our profound gratitude and deep regard to our guide **Dr. J. Narasimharao**, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We take this opportunity to extend our heartfelt appreciation to the Project Review Committee (PRC) Coordinators—**Y Varalaxmi, B Sekhar, D Nageswar Rao, and SVSV Prasad Sanaboina**—for their unwavering support, insightful guidance, and valuable inputs, which played a crucial role in steering this project through its various stages.

Our sincere appreciation also goes to **Dr. Nuthanakanti Bhaskar**, Head, for his encouragement and continuous support in ensuring the successful completion of our project.

We are deeply grateful to **Dr. A. Raji Reddy**, Director, for his cooperation throughout the course of this project. Additionally, we extend our profound gratitude to Sri. **Ch. Gopal Reddy**, Chairman, Smt. **C. Vasantha Latha**, Secretary and Sri. **C. Abhinav Reddy**, Vice-Chairman, for fostering an excellent infrastructure and a conducive learning environment that greatly contributed to our progress.

We also acknowledge and appreciate the guidance and assistance provided by the faculty and staff of **CMR Technical Campus**, whose contributions have been invaluable in bringing this project to fruition.

Lastly, we sincerely thank our families for their unwavering support and encouragement. We also extend our gratitude to the teaching and non-teaching staff of CMR Technical Campus for their guidance and assistance. Their contributions, along with the support of everyone who helped directly or indirectly, have been invaluable in the successful completion of this project.

V Venkata Sai Anand (227R1A0559)

A Ambika (227R1A0501)

K Sirivalli (227R1A0531)

VISION AND MISSION

INSTITUTE VISION:

To Impart quality education in serene atmosphere thus strive for excellence in Technology and Research.

INSTITUTE MISSION:

1. To create state of art facilities for effective Teaching- Learning Process.
2. Pursue and Disseminate Knowledge based research to meet the needs of Industry & Society.
3. Infuse Professional, Ethical and Societal values among Learning Community.

DEPARTMENT VISION:

To provide quality education and a conducive learning environment in computer engineering that foster critical thinking, creativity, and practical problem-solving skills.

DEPARTMENT MISSION:

1. To educate the students in fundamental principles of computing and induce the skills needed to solve practical problems.
2. To provide State-of-the-art computing laboratory facilities to promote industry institute interaction to enhance student's practical knowledge.
3. To inculcate self-learning abilities, team spirit, and professional ethics among the students to serve society.

ABSTRACT

This project is titled as “Automated Resume Analysis & Skill Matching Website using NLP”. Online job postings attract a massive number of applications in a short time, making manual resume screening inefficient, costly, and prone to bias. Many highly qualified candidates are overlooked, leading to hiring mismatches. To address these challenges, we propose an intelligent automated system that leverages Natural Language Processing (NLP) with SpaCy and Machine Learning (ML) to streamline the resume evaluation process. Our system utilizes SpaCy's NLP capabilities to extract key details such as skills, education, and experience from unstructured resumes and generates concise summaries by eliminating irrelevant information. This significantly reduces the workload for recruiters, making the screening process more effective. The system employs a vectorization model with cosine similarity to compare resumes against job descriptions, ranking candidates based on relevance. Employers can input job requirements and constraints, and the system automatically evaluates and ranks resumes accordingly. This ensures that only the most suitable candidates are shortlisted, improving hiring accuracy and efficiency. Furthermore, we have implemented a selection and rejection feature that updates job applicants on their status in real time. Once an administrator selects or rejects an application, the candidate’s job card is updated immediately, enhancing transparency in the recruitment process. By automating resume screening with SpaCy, integrating ML-based ranking, and providing real-time applicant status updates, our system optimizes hiring workflows, minimizes biases, and enables recruiters to identify the best-fit candidates quickly and accurately.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 3.1	Project Architecture of Automated Resume Analysis & Skill Matching Website Using NLP	14
Figure 3.2	Use Case Diagram of Automated Resume Analysis & Skill Matching Website Using NLP	16
Figure 3.3	Class Diagram of Automated Resume Analysis & Skill Matching Website Using NLP	17
Figure 3.4	Sequence Diagram of Automated Resume Analysis & Skill Matching Website Using NLP	18
Figure 3.5	Collaboration Diagram of Automated Resume Analysis & Skill Matching Website Using NLP	19
Figure 4.1	File Structure of Automated Resume Analysis & Skill Matching Website Using NLP	22
Figure 4.2	Resumes Folder Showing Sample Resume Files of Automated Resume Analysis & Skill Matching Website Using NLP	23

Figure 4.3	Resumes Stored After Being Uploaded To The Website of Automated Resume Analysis & Skill Matching Website Using NLP	23
Figure 5.1	Terminal Output Indicating The Application Is Running of Automated Resume Analysis & Skill Matching Website Using NLP	53
Figure 5.2	Home Page of Automated Resume Analysis And Skill Matching With NLP	54
Figure 5.3	Admin Login Page of Automated Resume Analysis And Skill Matching With NLP	55
Figure 5.4	User Login Page of Automated Resume Analysis And Skill Matching With NLP	55
Figure 5.5	User Registration Page of Automated Resume Analysis And Skill Matching With NLP	56
Figure 5.6	Feedback Page of Automated Resume Analysis And Skill Matching With NLP	57

Figure 5.7	Admin Dashboard Screen of Automated Resume Analysis And Skill Matching With NLP	58
Figure 5.8	Post New Job Screen of Automated Resume Analysis And Skill Matching With NLP	59
Figure 5.9	Post New Job Successful Pop-Up Screen of Automated Resume Analysis And Skill Matching With NLP	59
Figure 5.10	All Job's Resumes View Screen of Automated Resume Analysis And Skill Matching With NLP	60
Figure 5.11	Selected Job Detail Card of Automated Resume Analysis And Skill Matching With NLP	61
Figure 5.12	Rejected Job Detail Card of Automated Resume Analysis And Skill Matching With NLP	61
Figure 5.13	View User Feedback Screen of Automated Resume Analysis And Skill Matching With NLP	62

Figure 5.14	User Dashboard Screen of Automated Resume Analysis And Skill Matching With NLP	63
Figure 5.15	View All Available Job's Screen of Automated Resume Analysis And Skill Matching With NLP	64
Figure 5.16	Upload Resume Page of Automated Resume Analysis And Skill Matching With NLP	65
Figure 5.17	Selected Job Card of Automated Resume Analysis And Skill Matching With NLP	66
Figure 5.18	Rejected Job Card of Automated Resume Analysis And Skill Matching With NLP	66
Figure 5.19	User View Job Card of Automated Resume Analysis And Skill Matching With NLP	66
Figure 5.20	Admin View Job Card of Automated Resume Analysis And Skill Matching With NLP	66

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
Table 6.1	Uploading Data To Database	70
Table 6.2	Updated database upload_resume Table	71

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	ii
LIST OF TABLES	v
1. INTRODUCTION	1
1.1 PROJECT PURPOSE	1
1.2 PROJECT FEATURES	2
2. LITERATURE SURVEY	4
2.1 REVIEW OF RELATED WORK	5
2.2 DEFINITION OF PROBLEM STATEMENT	7
2.3 EXISTING SYSTEM	8
2.4 PROPOSED SYSTEM	10
2.5 OBJECTIVES	11
2.6 HARDWARE & SOFTWARE REQUIREMENTS	12
2.6.1 HARDWARE REQUIREMENTS	12
2.6.2 SOFTWARE REQUIREMENTS	12
3. SYSTEM ARCHITECTURE & DESIGN	14
3.1 PROJECT ARCHITECTURE	14
3.2 DESCRIPTION	15
3.3 UML DIAGRAMS	16
4. IMPLEMENTATION	21
4.1 ALGORITHMS USED	21
4.2 SAMPLE CODE	24
5. RESULTS & DISCUSSION	53
6. VALIDATION	68
6.1 INTRODUCTION	69
6.2 TEST CASES	70
6.2.1 UPLOADING DATA TO DATABASE	70
6.2.2 UPDATED DATABASE TABLE	71
7. CONCLUSION & FUTURE ASPECTS	73
7.1 PROJECT CONCLUSION	73
7.2 FUTURE ASPECTS	74
8. BIBLIOGRAPHY	76
8.1 REFERENCES	76
8.2 GITHUB LINK	76

1. INTRODUCTION

1. INTRODUCTION

The project titled "Automated Resume Analysis & Skill Matching Website using NLP" is designed to address the inefficiencies and biases present in traditional recruitment processes. With companies receiving thousands of resumes for a single job posting, manual screening becomes time-consuming, error-prone, and subjective. This often leads to missed opportunities where qualified candidates are overlooked and hiring mismatches occur.

To solve these problems, our project introduces an intelligent web-based platform that uses Natural Language Processing (NLP) with SpaCy and Machine Learning (ML) algorithms to automate resume analysis and skill matching. The system parses unstructured resume data to extract critical details such as skills, qualifications, education, and experience. It then compares these attributes against the job requirements provided by employers using vectorization and cosine similarity, effectively ranking candidates based on their relevance.

In addition to improving accuracy in candidate-job matching, the platform includes real-time status updates, allowing applicants to receive instant feedback on their application status. This enhances transparency and engagement in the recruitment process. By automating resume screening and ranking, this system not only saves time and cost but also ensures unbiased and efficient hiring decisions.

1.1 PROJECT PURPOSE

The primary purpose of this project is to modernize and optimize the resume screening process using NLP and ML-based ranking algorithms. Traditional recruitment systems either rely on manual efforts or keyword-based filters, both of which have limitations such as subjectivity, inefficiency, and vulnerability to keyword stuffing.

Our system overcomes these limitations by offering a smarter solution that deeply understands the context of resumes using NLP and compares them intelligently against job descriptions. It simplifies candidate evaluation, ensures the best fit for a role, and provides both recruiters and job seekers with a seamless experience.

By integrating automated parsing and intelligent shortlisting, the system enhances the effectiveness of recruitment, eliminates biases, and matches candidates to jobs that truly reflect their skills and aspirations.

1.2 PROJECT FEATURES

This project introduces several key features that make the recruitment process smarter, faster, and more reliable:

Resume Parsing with NLP: Utilizes SpaCy to extract structured information (skills, education, experience) from unstructured resume files (PDF, DOCX, etc.).

Skill-Based Matching Algorithm: Compares candidate profiles to job requirements using vectorization and cosine similarity, ranking them by relevance.

Real-Time Status Updates: Applicants are notified instantly when their application is selected or rejected by the recruiter.

Bias Reduction: Removes human subjectivity by relying on machine-based evaluation of candidate skills and relevance.

Scalability & Flexibility: Designed to handle large volumes of resumes and job postings, making it suitable for enterprise-level hiring and bulk recruitment drives.

By integrating these features, this deep learning-based system provides a powerful, scalable, and automated solution for inappropriate content detection, ensuring better moderation, improved accuracy, and a safer viewing experience for users.

2. LITERATURE SURVEY

2.LITERATURE SURVEY

The recruitment process has undergone significant changes in recent years with the growing use of automated systems. Early research into automated resume analysis and skill matching focused primarily on keyword-based systems, which involved the extraction of specific terms from resumes and job descriptions to match candidates. However, this approach was often limited, as it could not account for the nuances in experience, soft skills, or broader contextual factors. As the demand for more accurate and scalable recruitment solutions grew, the field turned to natural language processing (NLP) and machine learning (ML) techniques to improve resume analysis. These technologies enabled the development of systems that could not only extract relevant data but also rank candidates based on how well their qualifications aligned with a given job description.

Recent advancements in NLP and ML have significantly enhanced the accuracy and effectiveness of automated resume screening systems. Researchers have explored various machine learning algorithms such as decision trees, random forests, and support vector machines (SVM) for automating the extraction of relevant features from resumes. These approaches help in identifying patterns and ranking candidates according to how well their skills match the job requirements. However, challenges remain in ensuring that these models are not biased and can generalize well to diverse industries and job roles. The integration of deep learning models, particularly recurrent neural networks (RNNs) and transformers, has improved the system's ability to handle more complex data, such as unstructured text and contextual information, leading to better candidate-job matches.

In addition to machine learning, research has also emphasized the importance of analysing external data sources, such as candidates' social media profiles and professional networks, to enhance the matching process. Social profiles often provide valuable insights into a candidate's skill set, career trajectory, and professional reputation, which are not always captured in a traditional resume. The integration of these data points with the resume analysis can provide a more comprehensive view of a candidate's suitability for a job, allowing companies to make more informed decisions. However, challenges regarding data privacy, ethical considerations, and the accuracy of external data sources remain key concerns in the development of such systems. Despite these challenges, integrating social data has the potential

to significantly improve the recruitment process, offering a more holistic and personalized approach to job matching.

2.1 REVIEW OF RELATED WORK

The automated screening of resumes has become a key focus in the recruitment domain, with an increasing reliance on Natural Language Processing (NLP) and Machine Learning (ML) to enhance the accuracy, efficiency, and scalability of the process. Early research in this area concentrated on improving keyword-based filtering systems, which form the backbone of most Applicant Tracking Systems (ATS). However, these systems have major limitations, particularly in the ability to understand the context and meaning of the keywords, leading to inaccurate matches and missed opportunities.

Keyword-Based Filtering and ATS: Applicant Tracking Systems (ATS) are among the most widely adopted tools in the recruitment process. They typically rely on keyword-based filtering, scanning resumes for specific terms that match the job description. Research by Parry & Bissell (2007) highlights how ATS could drastically reduce the time spent on initial resume reviews. However, their study also points to a high false-negative rate, where qualified candidates are overlooked because their resumes don't exactly match the keywords in the job description. This method often fails to identify contextual relationships between the job description and resume content, making it unsuitable for dynamic job roles that require nuanced understanding.

NLP Approaches for Resume Parsing and Matching: To address the limitations of keyword-based systems, several researchers have turned to Natural Language Processing (NLP) to improve the parsing and understanding of resumes. Early work focused on Named Entity Recognition (NER), a process of identifying key entities such as skills, degrees, and job titles from resumes. According to Li et al. (2015), using support vector machines (SVMs) in combination with NER models led to better resume categorization and parsing performance. However, the system still struggled with resumes that had complex formatting, such as PDFs and scanned documents, and lacked the ability to capture more nuanced details like job responsibilities or the context in which skills were applied.

Deep Learning for Semantic Matching: Advancements in deep learning, particularly with transformer-based models like BERT (Bidirectional Encoder Representations from Transformers), have significantly improved the semantic understanding of resumes and job descriptions. Devlin et al. (2019) demonstrated that BERT, trained on vast corpora of text, could be fine-tuned to understand the relationships between job descriptions and resumes more accurately. This model is particularly effective at capturing contextual relationships, allowing for better semantic matching between resumes and job requirements, compared to traditional keyword matching systems. Moreover, research by Joulin et al. (2017) on Doc2Vec embeddings demonstrated the potential of transforming entire documents (such as resumes) into dense vector representations, which can then be used to compute the similarity between candidate resumes and job descriptions. Such approaches allow for semantic similarity evaluation, moving beyond mere keyword matching.

Fairness and Bias in Resume Screening: A growing body of research is focused on ensuring that automated recruitment tools do not perpetuate biases. A study by Binns (2018) outlined how machine learning models used in ATS could inadvertently amplify biases, such as gender or racial bias, based on historical hiring data. To address this, Zhang et al. (2019) proposed fairness-aware learning models, which incorporate fairness constraints into the model training process, ensuring that the system produces fair outcomes for all candidates, regardless of gender, race, or other potentially biased features. Further research by Dastin (2018) found that AI systems used in hiring could reinforce unconscious bias if not properly calibrated. Techniques such as debiasing algorithms and counterfactual fairness have been explored as methods to mitigate such bias and ensure more equitable candidate selection.

Challenges and Future Directions: Despite the progress made, several challenges remain in the development of automated resume analysis systems. First, handling unstructured data remains a significant hurdle, as resumes vary greatly in format and structure, even across similar job roles. Moreover, the ability of models to interpret resumes across different industries and job domains is still under development, as models tend to overfit to the specific datasets they were trained on. Another major challenge lies in ensuring that these systems are capable of handling the scale of real-time recruitment processes, especially for large organizations. Scalability is critical in systems where thousands of applicants may submit resumes for a single job opening. Transfer learning and reinforcement learning have emerged

as promising techniques to address these challenges by allowing models to improve through continuous learning and adaptation to new, unseen resumes.

Recent Advances and Innovations: Recent studies have explored the integration of reinforcement learning into recruitment systems. Vargas et al. (2021) proposed an RL-based system where the model learns to rank resumes by receiving feedback from recruiters. This adaptive system dynamically adjusts its ranking algorithm over time, continuously improving the precision of candidate-job matching. Another innovation is the use of federated learning, which allows the training of machine learning models on decentralized data without compromising privacy. Smith et al. (2020) explored how this technique could be used to develop hiring tools that comply with privacy regulations, offering more secure and compliant recruitment solutions.

2.2 DEFINITION OF PROBLEM STATEMENT

The primary goal of this project is to develop a robust and efficient automated system for resume analysis and skill matching. Current recruitment methods are often time-consuming, inefficient, and lack the flexibility to accurately assess candidates' qualifications. Traditional recruitment processes require candidates to fill out forms online, but this approach may not always provide genuine or comprehensive insights into the candidate's true skills and experience.

Our proposed system overcomes these limitations by allowing candidates to upload their resumes in any format of their choice. Beyond extracting information from the resume, the system also analyzes data from the candidate's social profile to gain a more holistic understanding of their abilities, experiences, and professional background. This comprehensive approach saves time for both candidates and hiring companies while providing more accurate and reliable information for job matching.

By incorporating both resume details and social media insights, our system ensures that candidates are matched with jobs that truly align with their skills and expertise, leading to higher job satisfaction. Similarly, it offers companies a flexible, scalable, and data-driven way to find the best-fit candidates, streamlining the hiring process and making it more efficient.

2.3 EXISTING SYSTEM

The current resume screening and candidate selection systems in use range from traditional manual processes to more sophisticated, AI-powered tools. Despite their advancements, each method still has inherent limitations, which impact efficiency, accuracy, and fairness in recruitment. Below is a discussion on the major existing systems, their features, and their associated drawbacks.

Manual Screening: Manual screening continues to be one of the most widely used methods for recruitment, where recruiters carefully review each resume to assess whether candidates meet the job's requirements. However, this approach has several limitations. It is inherently subjective, as recruiters often base their evaluations on personal judgment, which can lead to inconsistencies and unconscious biases in the selection process. Furthermore, manual screening is a time-consuming process, particularly when reviewing large volumes of resumes. This causes significant delays in the hiring timeline. Additionally, manual screening can be inefficient, as resumes may need to go through several rounds of evaluation before a decision is made, which can slow down the overall process.

Keyword-Based Filtering (ATS): Applicant Tracking Systems (ATS) were introduced to automate the resume screening process by focusing on matching specific keywords in resumes with job descriptions. While ATS systems are more efficient than manual screening, they come with their own set of challenges. One major limitation is their inability to understand the context behind keywords, which can result in mismatched rankings or overlooking qualified candidates. Additionally, ATS may struggle with complex resume formats, such as PDFs or scanned documents, which makes parsing difficult. Another issue is the tendency for candidates to engage in "keyword stuffing" by adding excessive keywords to their resumes in order to pass the ATS filter, which does not necessarily reflect their true suitability for the position.

Machine Learning-Based Resume Ranking: To overcome some of the limitations of ATS, many modern recruitment systems have incorporated Machine Learning (ML) and Natural Language Processing (NLP) techniques. These systems analyze unstructured resume

data and match it to job descriptions based on various features such as skills, experience, and education. While ML-based systems offer more accurate matching, they are not without their drawbacks. These systems require large, labelled datasets for training, which may not always be available, especially in niche industries or specific roles. Moreover, models trained in one domain may not generalize well to others, limiting their effectiveness across different job types. Additionally, ML-based systems typically do not integrate social profile data, such as LinkedIn profiles, which could provide valuable insights into a candidate's skills, network, and professional history.

Data Envelopment Analysis (DEA): Data Envelopment Analysis (DEA) is an analytical method that has been explored for candidate selection in research-driven recruitment models. It evaluates candidates based on multiple criteria, such as skills, experience, education, and certifications, and ranks them according to their efficiency in meeting the job requirements. While DEA offers a more objective and analytical approach to evaluating candidates, it also has certain limitations. First, DEA relies on structured data, which makes it difficult to apply to unstructured resume data or non-standard formats. Additionally, the complexity of DEA algorithms requires a high level of expertise to implement and maintain, making it less accessible for organizations without specialized knowledge. Finally, because DEA models are often tailored to specific industries or roles, they may not be easily adaptable across different sectors or job functions, which limits their versatility in broader recruitment scenarios.

Limitations of Existing System

While these existing systems provide valuable improvements over manual screening, they still exhibit significant limitations:

- **Time and Bias Issues:** Manual screening and ATS systems remain time-consuming and prone to bias.
- **Keyword Matching Limitations:** Both ATS and ML-based systems struggle with understanding the context of keywords and the nuances of candidate qualifications.

- **Data Structure Issues:** Systems like DEA and ATS often face challenges when dealing with unstructured data such as complex resume formats and social media profiles.

2.4 PROPOSED SYSTEM

The proposed system aims to address the challenges associated with existing recruitment systems. It integrates Resume Parsing, Natural Language Processing (NLP), and Machine Learning (ML) to automate the entire resume screening and ranking process. Key features of the proposed system include:

Resume Parsing & NLP API: The system uses a Resume Parser to extract relevant information from unstructured resumes, such as skills, experience, and educational qualifications. SpaCy, a leading NLP library, will be used for tokenization, part-of-speech tagging, and dependency parsing.

Skill Matching & Ranking: Once resumes are parsed, the extracted data is matched against the job description. A similarity scoring mechanism is used to rank candidates based on how well their skills align with the job's requirements. Techniques such as cosine similarity and TF-IDF (Term Frequency-Inverse Document Frequency) are used for this purpose.

Real-Time Status Updates: The system includes a feature that provides real-time updates to candidates about their application status. When recruiters take action on an application, the applicant's status is updated instantly, ensuring transparency and improving the user experience.

Advantages of the Proposed System:

The proposed system offers several advantages over existing systems, making the recruitment process more efficient, accurate, and fair.

- **Automated Resume Screening:** Eliminates the need for manual screening, reducing human error and bias.

- Efficient Candidate Shortlisting: Quickly narrows down the pool of applicants by ranking candidates based on skill relevance.
- Accurate Skill Matching: Uses advanced NLP and ML techniques to match candidates' skills with job descriptions accurately.
- Bias Reduction: Objective algorithms help minimize human biases in recruitment decisions.
- Time and Cost Efficiency: Speeds up the recruitment process and reduces the cost associated with manual screening.
- Real-time Application Status Updates: Keeps candidates informed about their status throughout the recruitment process.

2.5 OBJECTIVES

The primary objective of this project is to streamline and enhance the recruitment process through the development of an intelligent, automated resume analysis and skill-matching platform. This system will leverage Natural Language Processing (NLP), Machine Learning (ML), and Resume Parsing technologies to bridge the gap between job descriptions and candidate capabilities. The platform aims to minimize manual intervention, reduce recruitment time, and improve the fairness and accuracy of candidate selection.

Key objectives of the proposed system are as follows:

- **Automate Resume Screening and Analysis**: Develop an end-to-end automated system that parses and analyzes unstructured resumes to extract key candidate information such as skills, work experience, education, and certifications using NLP techniques
- **Implement Skill-Based Matching and Ranking**: Design a robust skill-matching algorithm that compares candidate profiles with job descriptions to evaluate their suitability. The system will rank candidates based on their relevance to the job role using similarity scoring techniques like TF-IDF and cosine similarity.

- **Enhance Recruitment Efficiency and Accuracy:** Eliminate repetitive manual screening tasks by automating the ranking process, thereby reducing human effort, saving time, and ensuring consistent, unbiased evaluation across all applications.
- **Improve Candidate and Recruiter Experience:** Provide real-time application status updates and transparency to candidates, ensuring they are well-informed throughout the process. Simultaneously, empower recruiters with a user-friendly dashboard to view, filter, and select top candidates quickly and confidently.

2.6 HARDWARE & SOFTWARE REQUIREMENTS

2.6.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements,

- Processor : Intel Core i3
- Hard disk : 20GB
- RAM : 4GB

2.6.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 10
- Language : Python
- Back-End : Django-ORM
- Designing : HTML, CSS, JavaScript
- Database : MYSQL
- Dependencies : Spacy, Pyresparser, Requests and PDFMiner

3. SYSTEM ARCHITECTURE & DESIGN

3. SYSTEM ARCHITECTURE & DESIGN

Project architecture refers to the structural framework and design of a project, encompassing its components, interactions, and overall organization. It provides a clear blueprint for development, ensuring efficiency, scalability, and alignment with project goals. Effective architecture guides the project's lifecycle, from planning to execution, enhancing collaboration and reducing complexity.

3.1 PROJECT ARCHITECTURE

This system architecture illustrates the complete workflow from job requirement input to candidate feedback. Job data and resumes undergo preprocessing and entity extraction using NLP tools like SpaCy and Pyresparser. Resumes are ranked based on relevance to job requirements and presented in a UI for recruiter decisions. Final selections trigger real-time feedback to candidates for improved communication.

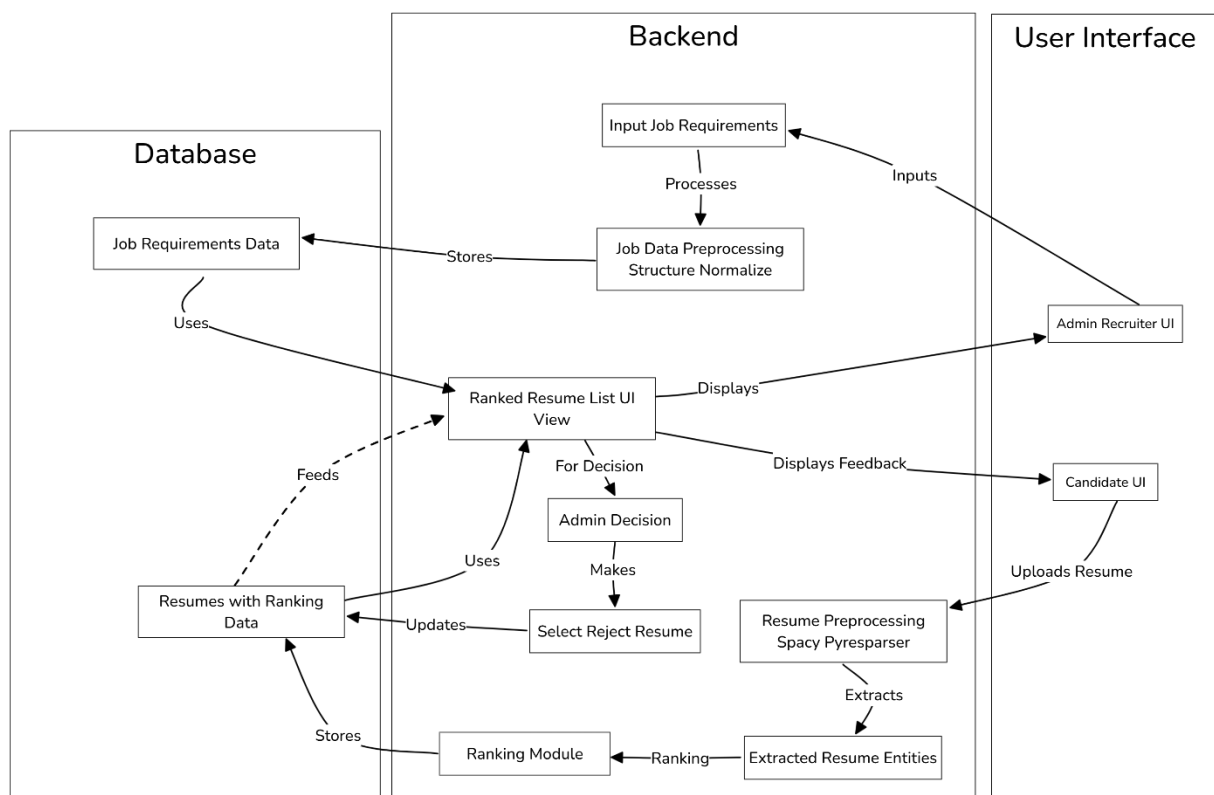


Figure 3.1: Project Architecture of Automated Resume Analysis & Skill Matching Website Using NLP

3.2 DESCRIPTION

The system architecture is designed to streamline the recruitment process by automating resume analysis and candidate ranking using Natural Language Processing (NLP) and Machine Learning (ML) techniques. The process begins with the input of job data from the recruiter, which includes essential job-related information such as title, required skills, experience, educational qualifications, and job description. This job data is then processed into a structured and machine-readable format, ensuring it can be efficiently used for matching with candidate profiles.

On the other end, resume entities are collected. Candidates upload their resumes in any format (PDF, DOCX, etc.), and the system uses a resume parser powered by NLP libraries (like SpaCy or similar tools) to extract relevant fields such as name, contact information, educational background, professional experience, certifications, technical skills, and soft skills. These extracted entities are then vectorized and normalized, making them suitable for comparison with job data.

Once both the job description and the resume data are structured, the system proceeds to the score calculation phase. Here, the system uses algorithms such as cosine similarity, TF-IDF, or word embeddings (like Word2Vec/GloVe) to determine how well each candidate's profile matches the job requirements. This scoring process ensures that candidates are not just matched on exact keyword presence, but also on the semantic relevance and contextual fit between their qualifications and the job role.

After the scores are calculated, the system automatically sorts and ranks all resumes. Candidates with the highest scores are placed at the top of the list, allowing recruiters to quickly focus on the most relevant profiles. This not only reduces hiring time but also enhances accuracy and fairness by minimizing human bias. Moreover, the system supports real-time status updates, ensuring both recruiters and applicants stay informed throughout the process. By combining automation with intelligent ranking mechanisms, the proposed system delivers a scalable, efficient, and data-driven solution for modern recruitment challenge.

3.3 UML DIAGRAM

A Unified Modelling Language (UML) diagrams are essential tools for visualizing the structure and behaviour of a software system. They provide a standardized way to document the architecture, relationships, and interactions within a system, aiding both in design and communication across stakeholders. Below are the UML diagrams used for this project:

3.3.1 USE CASE DIAGRAM

A Use Case Diagram provides a high-level overview of the system's functionality from the perspective of users (actors). It identifies what actions users can perform and how they interact with the system. This diagram helps define the scope of the system and the main features it offers.

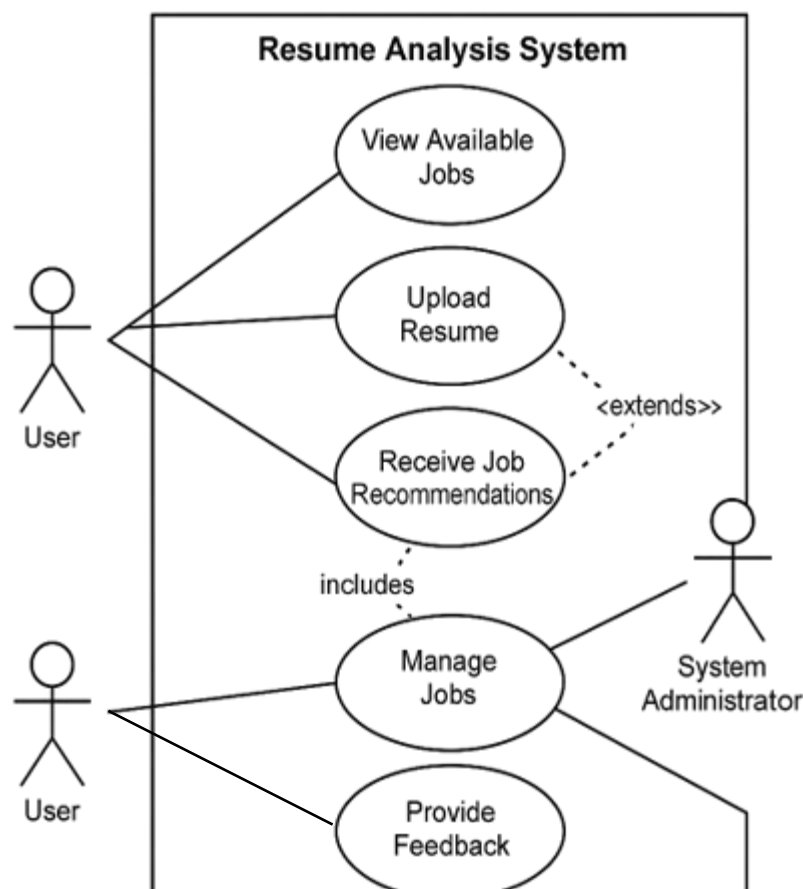


Figure 3.2: Use Case Diagram of Automated Resume Analysis & Skill Matching Website Using NLP

3.3.2 CLASS DIAGRAM

The Class Diagram outlines the static structure of the system. It defines each class, including its attributes and methods, and illustrates how classes relate to one another through associations, inheritance, or dependencies. This diagram helps in understanding the data model and the object-oriented design of the system.

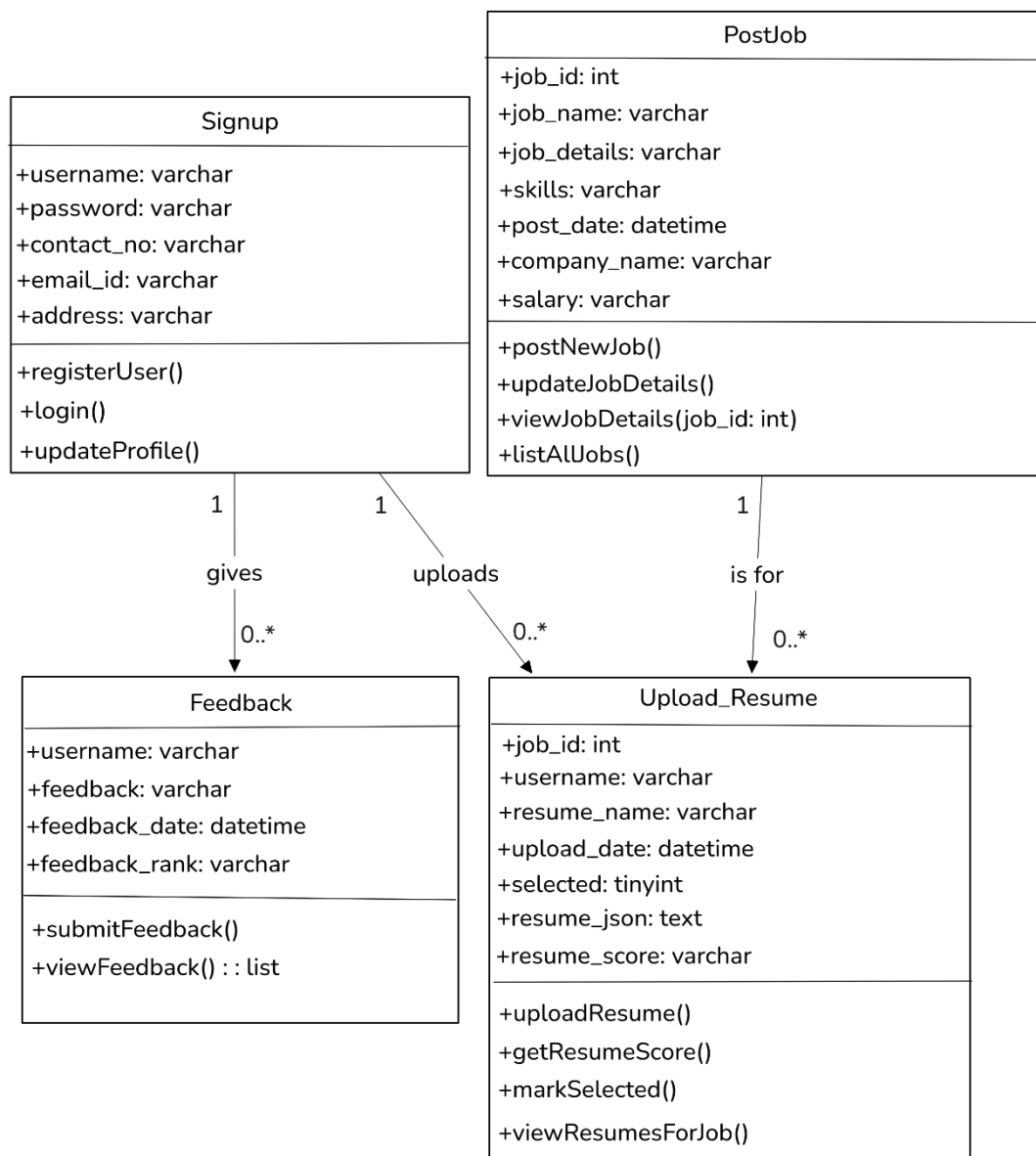


Figure 3.3: Class Diagram of Automated Resume Analysis & Skill Matching Website Using NLP

3.3.3 SEQUENCE DIAGRAM

A Sequence Diagram shows how objects in the system interact with each other over time. It captures the order of messages exchanged between actors and system components for a particular use case. This diagram is useful for modeling the dynamic behavior of the system and the sequence of operations.

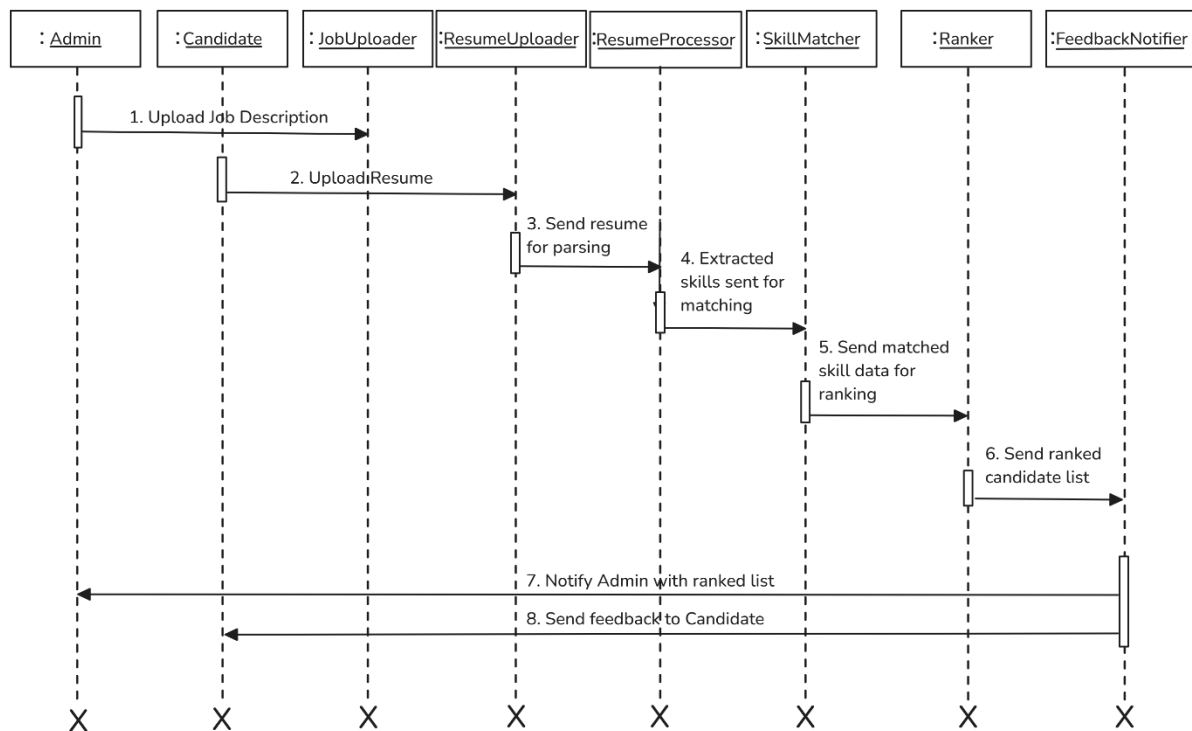


Figure 3.4: Sequence Diagram of Automated Resume Analysis & Skill Matching Website Using NLP

3.3.4 COLLABORATION DIAGRAM

A The collaboration diagram illustrates the interaction between the core components of the job portal system, namely the admin, new user, and server. It highlights the sequence of messages exchanged among these entities to perform various tasks. The admin communicates with the server to post jobs, view resume scores, submit new job postings, view feedback, and logout. Meanwhile, the new user interacts with the server to view available jobs, receive job suggestions, and logout. Each interaction is marked with a numbered message, indicating the order of operations and the direction of communication. This diagram effectively captures how different parts of the system collaborate to achieve functional objectives, making it easier to understand the system's structure and behavior during execution.

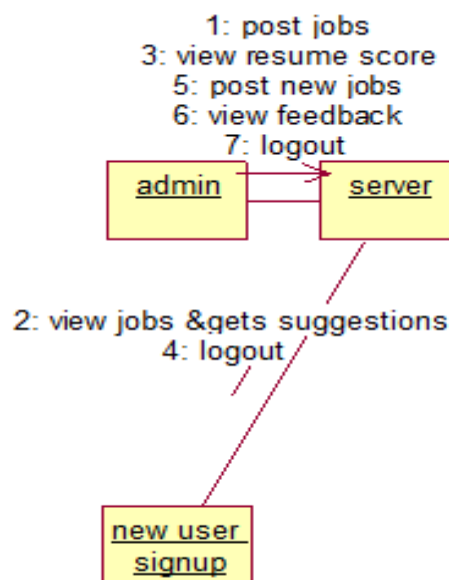


Figure 3.5: Collaboration Diagram of Automated Resume Analysis & Skill Matching Website Using NLP

4. IMPLEMENTATION

4. IMPLEMENTATION

This section outlines the practical realization of the proposed system, including the integration of core functionalities such as resume parsing, skill extraction, and candidate ranking. The system has been implemented using modern web technologies and powerful NLP tools to enable intelligent and automated recruitment processes.

The backend is responsible for parsing and analyzing resumes, while the frontend provides a clean and interactive user interface for both applicants and recruiters. The system allows real-time resume uploads and processes them to extract relevant data and present ranked results to the recruiter.

4.1 ALGORITHMS USED

The success of this resume screening system is driven by the application of key algorithms and techniques designed to process and analyze unstructured text data. These include:

Natural Language Processing (NLP) – SpaCy

The SpaCy is an industrial-strength NLP library that was used extensively for resume analysis. The core techniques applied include:

- **Tokenization:** Breaking down the resume text into individual tokens (words, punctuation, etc.).
- **Named Entity Recognition (NER):** Identifying named entities such as skills, organizations, designations, and educational institutions.
- **Part-of-Speech Tagging:** Classifying each word by its grammatical role (e.g., noun, verb, adjective).
- **Dependency Parsing:** Analyzing sentence structure and understanding relationships between words.

These NLP features allow the system to intelligently extract structured information from diverse and unstructured resume formats

Resume Ranking Algorithm

After extracting relevant features from the resume, the system ranks candidates based on a weighted evaluation of critical attributes such as:

- Key Skills Match
- Years of Experience
- Educational Qualifications
- Certifications (if available)

The ranking algorithm assigns weights to each of these categories and generates a composite score for every candidate. Candidates are sorted based on this score, ensuring that the most suitable applicants appear at the top of the list.

The **Figure 4.1** shows the overall structure of the project directory, highlighting the organization of various files and folders. It includes backend scripts, frontend resources, configuration files, and data directories, which are structured for efficient development and maintenance.

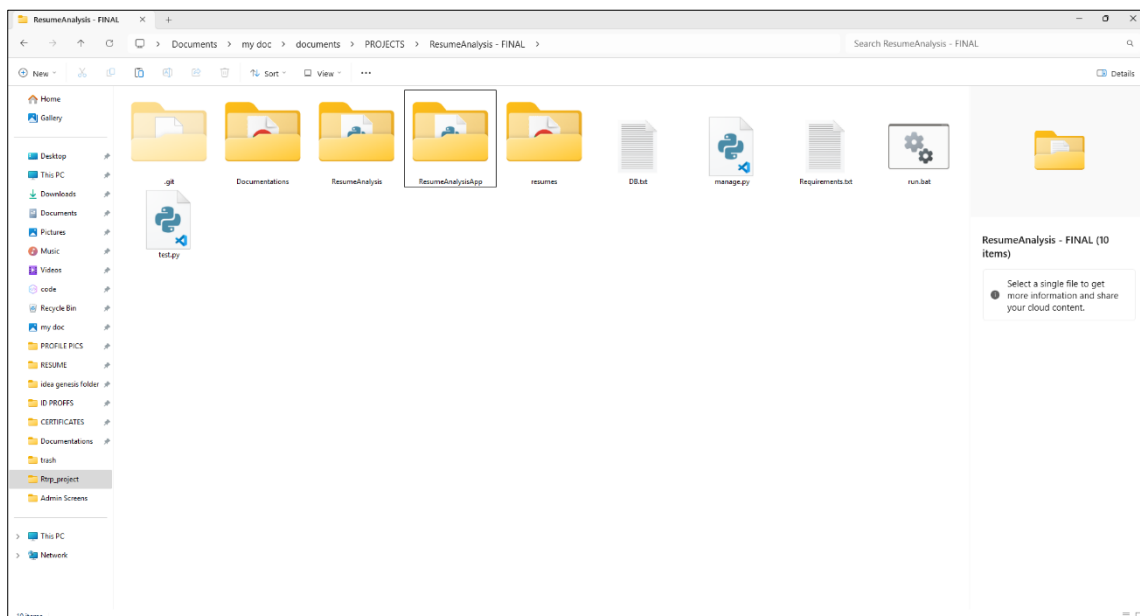


Figure 4.1: File Structure of Automated Resume Analysis & Skill Matching Website Using NLP

The **Figure 4.2** displays the contents of the Resumes folder, containing sample resume files in formats like PDF and DOCX. These files are used to test and validate the resume upload and processing functionality of the system.

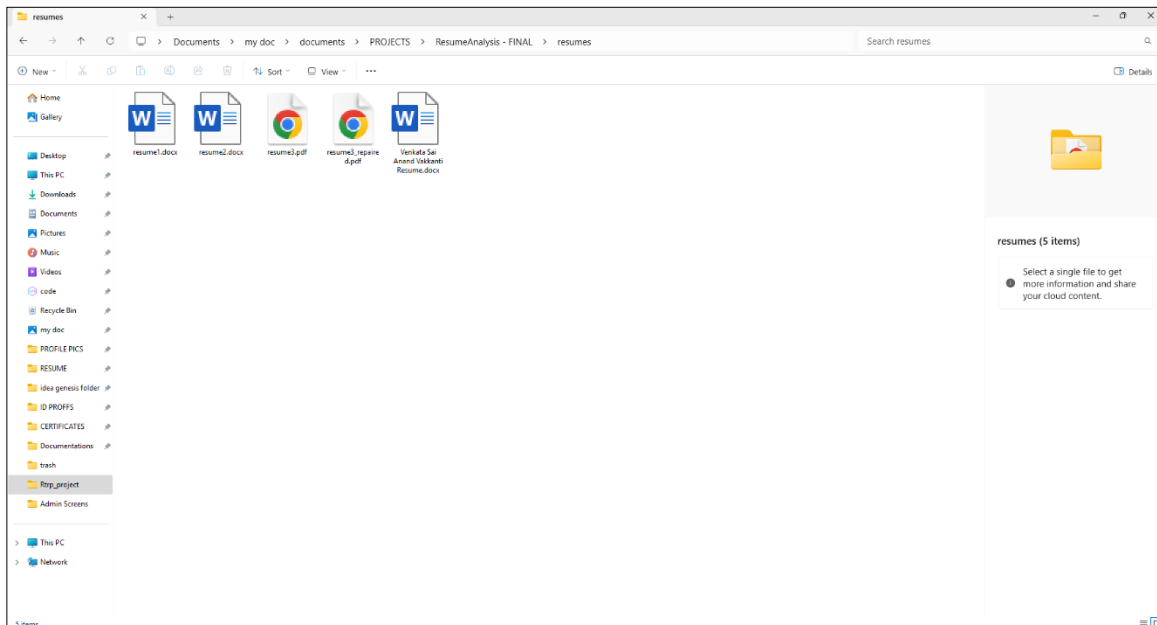


Figure 4.2: Resumes Folder Showing Sample Resume Files of Automated Resume Analysis & Skill Matching Website Using NLP

The **Figure 4.3** presents the resumes as they appear in the system after being uploaded through the website. This screenshot confirms that the uploaded files are successfully stored and accessible within the application interface.

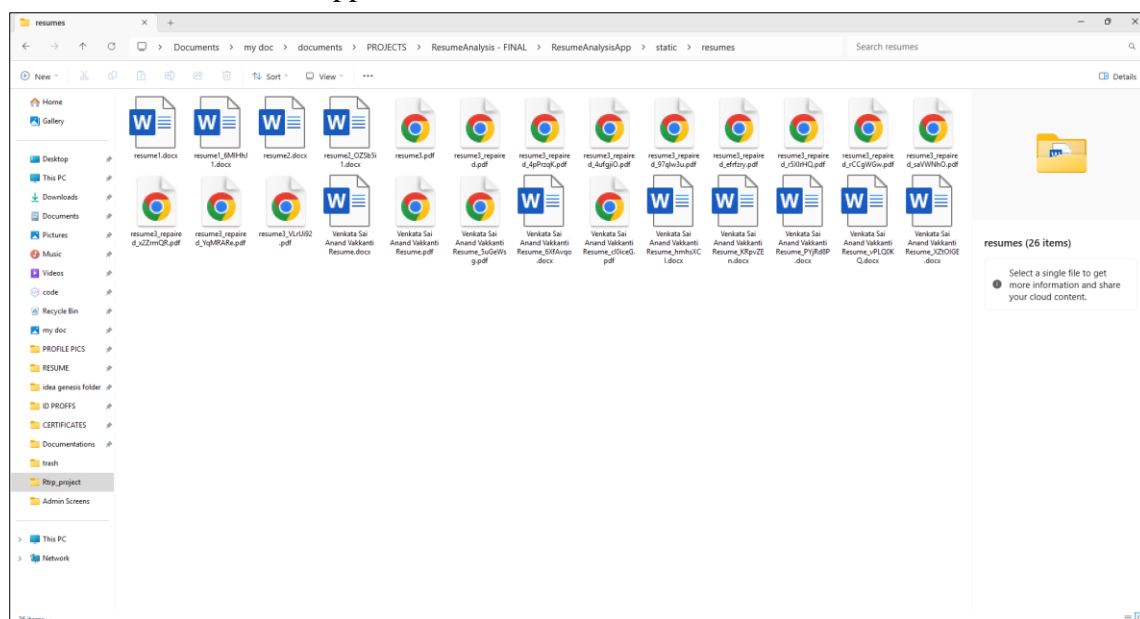


Figure 4.3: Resumes Stored After Being Uploaded To The Website of Automated Resume Analysis & Skill Matching Website Using NLP

4.2 SAMPLE CODE

```

from django.shortcuts import render, redirect
import pymysql
from django.http import HttpResponse, JsonResponse
from django.core.files.storage import FileSystemStorage
import datetime
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt
from pyresparser import ResumeParser
import io, json, base64, re

global uname, job_id
global admin_login_status
user_job_list = []
all_jobs_list = []

def checkUserLogin():
    global uname
    try:
        if not uname:
            print("Not logged in - Please login first")
            return False
    except NameError:
        return False
    return True

def checkadminloginstatus():
    global admin_login_status
    try:
        if not admin_login_status:
            print("Not logged in - Please login first")

```

```

        return False
    except NameError:
        return False
    return True

def AdminLogin(request):
    if request.method == 'GET':
        return render(request, 'AdminLogin.html', {})

def UserLogin(request):
    if request.method == 'GET':
        return render(request, 'UserLogin.html', {})

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def Signup(request):
    if request.method == 'GET':
        return render(request, 'Signup.html', {})

def Aboutus(request):
    if request.method == 'GET':
        return render(request, 'Aboutus.html', {})

def Feedback(request):
    if request.method == 'GET':
        return render(request, 'Feedback.html', {})

def EditJobs(request, job_id):
    global all_jobs_list
    all_jobs_list = []
    update_all_jobs_list()

```

```

job_to_edit = None
for job in all_jobs_list:
    if job['job_id'] == job_id:
        job_to_edit = job
        break
if job_to_edit is None:
    return render(request, 'AdminScreen.html', {
        'Admin_name': "Administrator",
        'edit_job_status': False,
        'edit_job_status_message': "Job not found" + ' - Invalid request method',
        'Admin_jobs_List': all_jobs_list
    })
all_skills = ["Access", "C", "C++", "Cloud", "CSS", "Data
analysis", "Database", "HTML", "Java", "JavaScript", "Microsoft
Word", "OpenCV", "Oracle", "PHP", "Python", "Scrum", "Shell", "SQL", "Technical"]
return render(request, 'EditJobs.html', {
    'jobid': job_id,
    'job_title': job_to_edit['title'],
    'job_details': job_to_edit['description'],
    'company': job_to_edit['company'],
    'salary': job_to_edit['salary'],
    'skills': job_to_edit['skills_required'],
    'all_skills': all_skills
})

def SignupAction(request):
    Signup_status = False
    Signup_status_message = "
    if request.method == 'POST':
        username = request.POST.get('t1', "").strip()
        password = request.POST.get('t2', "").strip()
        contact = request.POST.get('t3', "").strip()
        email = request.POST.get('t4', "").strip()
        address = request.POST.get('t5', "").strip()

```

```

try:
    with pymysql.connect(
        host='127.0.0.1', port=3306, user='root', password='root',
        database='resumeanalysis', charset='utf8'
    ) as con:
        with con.cursor() as cur:
            cur.execute("SELECT username FROM signup WHERE username =
%s", (username,))
            existing_user = cur.fetchone()
            if existing_user:
                Signup_status_message = "Given Username already exists."
            else:
                signup_query = """
                INSERT INTO signup (username, password, contact_no,
email_id, address)
                VALUES (%s, %s, %s, %s, %s)
                """
                cur.execute(signup_query, (username, password, contact, email,
address))
                con.commit()

                if cur.rowcount == 1:
                    Signup_status = True
                    Signup_status_message = "Your account has been created
successfully."
                except pymysql.MySQLError as e:
                    Signup_status_message = "There was an error creating your account. Please
try again later." + f"Database Error: {e}"
                except Exception as e:
                    Signup_status_message = "An unexpected error occurred. Please try again
later." + f"Unexpected Error: {e}"
            context = {
                'Signup_status': Signup_status,
                'Signup_status_message': Signup_status_message

```

```

    }
    return render(request, 'Signup.html', context)

def UserLoginAction(request):
    if request.method == 'POST':
        global uname
        option = 0
        username = request.POST.get('username', False).strip()
        password = request.POST.get('password', False)
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password =
'root', database = 'resumeanalysis',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM signup")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == username and row[1] == password:
                    uname = username
                    option = 1
                    break
            if option == 1:
                return redirect('UserDashboard')
            else:
                context= {'data':'Invalid login details'}
                return render(request, 'UserLogin.html', context)

def AdminLoginAction(request):
    if request.method == 'POST':
        global uname , admin_login_status
        username = request.POST.get('username', False).strip()
        password = request.POST.get('password', False).strip()
        if username == "admin" and password == "admin":
            admin_login_status = True
            return redirect('AdminDashboard')

```



```
else:
```

```
    context= {'data':'Invalid login details'}
```

```
    return render(request, 'AdminLogin.html', context)
```

```
def PostJobs(request):
```

```
    if not checkadminloginstatus():
```

```
        return redirect('AdminLogin')
```

```
    if request.method == 'GET':
```

```
        return render(request, 'PostJobs.html', {})
```

```
def Feedback(request):
```

```
    if request.method == 'GET':
```

```
        return render(request, 'Feedback.html', {})
```

```
def Aboutus(request):
```

```
    if request.method == 'GET':
```

```
        return render(request, 'Aboutus.html', {})
```

```
def FeedbackAction(request):
```

```
    Feedback_status = False
```

```
    Feedback_status_message = "
```

```
    global uname
```

```
    if request.method == 'POST':
```

```
        try:
```

```
            if not checkUserLogin():
```

```
                uname = "Anonymous"
```

```
            Feedback_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
```

```
            feedback = request.POST.get('t1', "").strip()
```

```
            rank = request.POST.get('t2', "").strip()
```

```
            with pymysql.connect(
```

```
                host='127.0.0.1', port=3306, user='root', password='root',
```

```
                database='resumeanalysis', charset='utf8'
```

```
            ) as db_connection:
```

```

        with db_connection.cursor() as db_cursor:
            student_sql_query = f"INSERT INTO feedback (username, feedback,
feedback_date, feedback_rank) VALUES ({'uname}', {'feedback}',
'Feedback_date', {rank})"
            db_cursor.execute(student_sql_query)
            db_connection.commit()
            if db_cursor.rowcount == 1:
                Feedback_status = True
                Feedback_status_message = 'Your feedback has been accepted. The
admin will review and get back to you.'
            except pymysql.MySQLError as e:
                print(f"Database Error: {e}")
                Feedback_status_message = 'There was an error posting your feedback.
Please try again later.'
            except Exception as e:
                print(f"Unexpected Error: {e}")
                Feedback_status_message = 'An unexpected error occurred. Please try again
later.'

        context = {
            'Feedback_status': Feedback_status,
            'Feedback_status_message': Feedback_status_message
        }
        return render(request, 'Feedback.html', context)

def PostJobsAction(request):
    error_message = "There was an error posting your job. Please try again later."
    if not checkadminloginstatus():
        return redirect('AdminLogin')
    PostJobs_status = False
    PostJobs_status_message = error_message
    if request.method == 'POST':
        try:
            job = request.POST.get('t1', "").strip()
            details = request.POST.get('t2', "").strip()

```

```

company = request.POST.get('t3', "").strip()
salary = request.POST.get('t4', "").strip()
skills = request.POST.getlist('t5')
skills = ','.join(skills)
today = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
job_id = 0
con = pymysql.connect(
    host='127.0.0.1', port=3306, user='root', password='root',
    database='resumeanalysis', charset='utf8'
)
with con.cursor() as cur:
    cur.execute("SELECT COUNT(job_id) FROM postjob")
    job_id = cur.fetchone()[0] + 1
with pymysql.connect(
    host='127.0.0.1', port=3306, user='root', password='root',
    database='resumeanalysis', charset='utf8'
) as db_connection:
    with db_connection.cursor() as db_cursor:
        student_sql_query = """
            INSERT INTO postjob (job_id, job_name, job_details, skills,
post_date, company_name, salary)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
        """
        db_cursor.execute(student_sql_query, (job_id, job, details, skills,
today, company, salary))
        db_connection.commit()
        if db_cursor.rowcount == 1:
            PostJobs_status = True
            PostJobs_status_message = f'Your job was posted successfully... You
can see your {job} job in the dashboard.'
        except pymysql.MySQLError as e:
            PostJobs_status_message = error_message + f'Database Error: {e}'
        except Exception as e:
            PostJobs_status_message = error_message + f'Unexpected Error: {e}'

```

```

context = {
    'PostJobs_status': PostJobs_status,
    'PostJobs_status_message': PostJobs_status_message
}

return render(request, 'PostJobs.html', context)

def EditJobsAction(request, job_id):
    global all_jobs_list
    if not checkadminloginstatus():
        return redirect('AdminLogin')
    EditJobs_status = False
    EditJobs_status_message = ""
    job_data = None
    if request.method == 'POST':
        try:
            job = request.POST.get('t1', "").strip()
            details = request.POST.get('t2', "").strip()
            company = request.POST.get('t3', "").strip()
            salary = request.POST.get('t4', "").strip()
            skills = request.POST.getlist('t5')
            skills = ','.join(skills)
            today = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            with pymysql.connect(
                host='127.0.0.1', port=3306, user='root', password='root',
                database='resumeanalysis', charset='utf8'
            ) as db_connection:
                with db_connection.cursor() as db_cursor:
                    student_sql_query = """
                        UPDATE postjob
                        SET job_name = %s, job_details = %s, skills = %s, post_date = %s,
company_name = %s, salary = %s
                        WHERE job_id = %s
                    """
                    db_cursor.execute(student_sql_query, (job, details, skills, today,

```

```

company, salary, job_id))
        db_connection.commit()

        if db_cursor.rowcount == 1:
            EditJobs_status = True
            EditJobs_status_message = f'Your job was updated successfully...
You can see the updated {job} job in the dashboard.'
        elif db_cursor.rowcount == 0:
            EditJobs_status_message = "No job was updated. Please check the
job ID."
        else:
            EditJobs_status_message = "Multiple jobs were updated. This should
not happen."

    except pymysql.MySQLError as e:
        EditJobs_status_message = f"Database Error: {e}"
    except Exception as e:
        EditJobs_status_message = f"Unexpected Error: {e}"

    if EditJobs_status:
        all_jobs_list = []
        update_all_jobs_list()
        return render(request, 'AdminScreen.html', {'Admin_name':
"Administrator", 'Admin_jobs_List': all_jobs_list, 'EditJobs_status': True,
'EditJobs_status_message': EditJobs_status_message})

    try:
        with pymysql.connect(
            host='127.0.0.1', port=3306, user='root', password='root',
            database='resumeanalysis', charset='utf8'
        ) as db_connection:
            with db_connection.cursor(pymysql.cursors.DictCursor) as db_cursor:
                db_cursor.execute("SELECT * FROM postjob WHERE job_id = %s",
(job_id,))
                job_data = db_cursor.fetchone()
                if not job_data:
                    EditJobs_status_message = "Job not found."

```

```

        return render(request, 'AdminScreen.html', {'Admin_name':
"Administrator", 'Admin_jobs_List': all_jobs_list, 'EditJobs_status': False,
'EditJobs_status_message': EditJobs_status_message})
        if job_data['skills']:
            job_data['skills'] = job_data['skills'].split(',')
        else:
            job_data['skills'] = []
    except pymysql.MySQLError as e:
        EditJobs_status_message = f"Database Error: {e}"
        job_data = None
    except Exception as e:
        EditJobs_status_message = f"Unexpected Error: {e}"
        job_data = None
    context = {
        'Admin_name': "Administrator",
        'Admin_jobs_List': all_jobs_list,
        'EditJobs_status': EditJobs_status,
        'EditJobs_status_message': EditJobs_status_message,
        'job_data': job_data,
    }
    return render(request, 'AdminScreen.html', context)

def ViewFeedback(request):
    if request.method == 'GET':
        Feedback_list = []
        rank = []
        con = pymysql.connect(host='127.0.0.1', port=3306, user='root',
password='root', database='resumeanalysis', charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select username,feedback,feedback_rank,feedback_date
FROM feedback ORDER BY feedback_rank DESC")
            rows = cur.fetchall()
            for row in rows:

```

```

        Feedback_list.append({
            'username': row[0],
            'feedback_text': row[1],
            'feedback_rank': row[2],
            'feedback_date' : row[3]
        })
        rank.append(row[2])
    unique, count = np.unique(np.asarray(rank), return_counts=True)
    plt.figure(figsize=(6, 6)) # Create a new figure
    plt.pie(count, labels=unique, autopct='%1.1f%%')
    plt.title('Feedback Ranking Graph')
    plt.axis('equal')
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
    image_png = buffer.getvalue()
    buffer.close()
    graphic = base64.b64encode(image_png).decode('utf-8')
    img_data = f'data:image/png;base64,{graphic}'
    context = {'Feedback_list': Feedback_list, 'chart': img_data}
    return render(request, 'ViewFeedback.html', context)

def UserDashboard(request):
    if not checkUserLogin():
        return redirect('UserLogin')
    global uname, user_job_list
    update_all_jobs_list()
    update_user_job_list()
    return render(request, 'UserScreen.html', {'C_username': uname, 'User_jobs':
user_job_list})

def AdminDashboard(request):
    if not checkadminloginstatus():
        return redirect('AdminLogin')

```

```

global all_jobs_list
update_all_jobs_list()
return render(request, 'AdminScreen.html', {'Admin_name': "Administrator",
'Admin_jobs_List': all_jobs_list})

```

```

def update_user_job_list():
    """Function to refresh the global user_job_list. Called whenever UserDashboard
    is accessed"""
    global user_job_list, uname
    user_job_list = []
    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password =
'root', database = 'resumeanalysis',charset='utf8')
    with con:
        cur = con.cursor()
        job_query = f"SELECT p.job_name, p.job_details, p.company_name, p.skills ,
u.resume_score , p.job_id , p.salary , u.selected , u.upload_date FROM postjob p
JOIN upload_resume u ON p.job_id = u.job_id WHERE u.username = \"{uname}\"
ORDER BY u.upload_date DESC"
        cur.execute(job_query)
        applied_jobs = cur.fetchall()
        for job in applied_jobs:
            skills_string = job[3]
            skills_list = [skill.strip() for skill in skills_string.split(',')] if skills_string else
            []
            job_logo_name = skills_list[0] if skills_list else "default"
            job_logo_name = job_logo_name + ".png"
            user_job_list.append({
                'title': job[0],
                'description': job[1],
                'company': job[2],
                'skills_required': job[3],
                'resume_score' : job[4],
                'job_id': job[5],
                'salary' : job[6],

```



```

        'job_logo_name': job_logo_name,
        'selected': job[7] if job[7] is not None else -1,
        'applied_date': job[8]
    })
    print(user_job_list)

def update_all_jobs_list():
    """Function to refresh the global all_jobs_list posted """
    global all_jobs_list
    all_jobs_list = []
    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password =
'root', database = 'resumeanalysis',charset='utf8')
    with con:
        cur = con.cursor()
        query = "SELECT p.job_id, p.job_name, p.job_details, p.company_name,
p.skills, p.post_date, p.salary, COALESCE(u.resume_count, 0) AS resume_count
FROM postjob p LEFT JOIN ( SELECT job_id, COUNT(job_id) AS resume_count
FROM upload_resume GROUP BY job_id ) u ON p.job_id = u.job_id ORDER BY
resume_count DESC, p.post_date DESC"
        cur.execute(query)
        applied_jobs = cur.fetchall()
        for job in applied_jobs:
            skills_string = job[4]
            skills_list = [skill.strip() for skill in skills_string.split(',')] if skills_string else
            []
            cur.execute(f"SELECT COUNT(*) FROM upload_resume WHERE job_id
= {job[0]} AND selected = 1")
            NumberOfSelected = cur.fetchone()[0]
            cur.execute(f"SELECT COUNT(*) FROM upload_resume WHERE job_id
= {job[0]} AND selected = 0")
            NumberOfRejected = cur.fetchone()[0]
            job_logo_name = skills_list[0] if skills_list else "default"
            job_logo_name = job_logo_name + ".png"
            all_jobs_list.append({

```

```

        'job_id': job[0],
        'title': job[1],
        'description': job[2],
        'company' : job[3],
        'skills_required': job[4],
        'post_date' : job[5],
        'salary' : job[6],
        'NumberOfApplied': job[7],
        'NumberOfNew': job[7]-NumberOfSelected-NumberOfRejected,
        'NumberOfSelected': NumberOfSelected,
        'NumberOfRejected': NumberOfRejected,
        'job_logo_name': job_logo_name
    })

```

```

def getScore(job_id, skills):
    require_skills = None
    score = 0

    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password =
'root', database = 'resumeanalysis',charset='utf8')

    with con:
        cur = con.cursor()
        cur.execute("select skills FROM postjob where job_id='"+job_id+"'")
        rows = cur.fetchall()
        for row in rows:
            require_skills = row[0]
        require_skills = require_skills.strip().split(",")
        for i in range(len(skills)):
            skills[i] = skills[i].lower().strip()
        for i in range(len(require_skills)):
            require_skills[i] = require_skills[i].lower()
        print(require_skills)
        found_skills = [x for x in skills if x in require_skills]
        if len(found_skills) > 0:
            if len(found_skills) >= len(require_skills):

```

```

        score = 100
    else:
        score = len(found_skills) / len(require_skills)
        score = score * 100
    return round(score)

def UploadResumeAction(request):
    error_message = "There was an error uploading your resume. Please try again later."
    if request.method != 'POST':
        return render(request, 'UploadResume.html', {'resume_upload_status': False, 'resume_upload_status_message': error_message + ' - Invalid request method'})
    if not checkUserLogin():
        return redirect('UserLogin')
    global uname
    resume_upload_status = False
    try:
        job_id = request.POST.get('t1', False)
        myfile = request.FILES['t2']
        if not job_id:
            return render(request, 'UploadResume.html', {'resume_upload_status': False, 'resume_upload_status_message': error_message + ' - Job ID is missing'})
        if not myfile:
            return render(request, 'UploadResume.html', {'resume_upload_status': False, 'resume_upload_status_message': error_message + ' - Resume file is missing'})
        fname = request.FILES['t2'].name
        upload_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        fs = FileSystemStorage()
        filename = fs.save('ResumeAnalysisApp/static/resumes/'+fname, myfile)
        try:
            data = ResumeParser('ResumeAnalysisApp/static/resumes/'+filename).get_extracted_data()
            if not data or 'skills' not in data:
                return render(request, 'UploadResume.html', {'resume_upload_status':

```

```

False, 'resume_upload_status_message': error_message + ' - Failed to extract data
from resume'})

except Exception as e:
    return render(request, 'UploadResume.html', {'resume_upload_status': False,
'resume_upload_status_message': error_message + f' - Resume parsing error:
{str(e)}'})

    skills = data['skills']
    score = getScore(job_id, skills)
    try:
        json_data = json.dumps(data)
        json_data = re.sub(r'[\x00-\x1F\x7F]', '', json_data)
        json_data = json_data.replace("\u2022", "- ")
        json_data = json_data.replace("\u2019", "")
        json_data = json_data.replace("\n", " ")
    except Exception as e:
        print(f"Error in json_data processing: {e}")
        json_data = "{}"
    try:
        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root',
password = 'root', database = 'resumeanalysis',charset='utf8')
        db_cursor = db_connection.cursor()
        student_sql_query = "INSERT INTO
upload_resume(job_id,username,resume_name,upload_date,resume_json,resume_s
core)
VALUES('"+str(job_id)+"','"+uname+"','"+fname+"','"+str(upload_date)+"','"+str(j
son_data)+"','"+str(score)+"')
        db_cursor.execute(student_sql_query)
        db_connection.commit()
    except pymysql.Error as db_err:
        return render(request, 'UploadResume.html', {'resume_upload_status': False,
'resume_upload_status_message': error_message + f' - Database error:
{str(db_err)}'})

    print(db_cursor.rowcount, "Record Inserted")
    if db_cursor.rowcount == 1:

```

```

        status_message = "Your resume has been uploaded, and we will get in touch
with you."

```

```

        resume_upload_status = True

```

```

    else:

```

```

        return render(request, 'UploadResume.html', {'resume_upload_status': False,
'resume_upload_status_message': error_message + ' - Database insertion failed'})

```

```

    except Exception as e:

```

```

        return render(request, 'UploadResume.html', {'resume_upload_status': False,
'resume_upload_status_message': error_message + f' - Unexpected error: {str(e)}'})

```

```

    context = {'user_resume_score': score, 'resume_upload_status':
resume_upload_status, 'resume_upload_status_message': status_message}

```

```

    return render(request, 'UploadResume.html', context)

```

```

def DeleteJobAction(request):

```

```

    error_message = "There was an error deleting the job. Please try again later."

```

```

    if not checkadminloginstatus():

```

```

        return redirect('AdminLogin')

```

```

    global all_jobs_list

```

```

    if request.method != 'GET':

```

```

        return render(request, 'AdminScreen.html', {
            'Admin_name': "Administrator",
            'delete_job_status': False,
            'delete_job_status_message': error_message + ' - Invalid request method',
            'Admin_jobs_List': all_jobs_list
        })

```

```

    delete_job_status = False

```

```

    try:

```

```

        job_id = request.GET.get('jobid')

```

```

        job_name = request.GET.get('jobTitle')

```

```

        if not job_id:

```

```

            return render(request, 'AdminScreen.html', {
                'Admin_name': "Administrator",
                'delete_job_status': False,
                'delete_job_status_message': error_message + ' - Job ID is missing',

```

```

        'Admin_jobs_List': all_jobs_list
    })
    try:

        con = pymysql.connect(host='127.0.0.1', port=3306, user='root',
password='root', database='resumeanalysis', charset='utf8')
        cur = con.cursor()
        cur.execute(f"DELETE FROM postjob WHERE job_id = {job_id}")
        con.commit()
        if cur.rowcount == 1:
            delete_job_status = True
            status_message = f"the job {job_name} has been deleted successfully."
        else:
            all_jobs_list = []
            update_all_jobs_list()
            return render(request, 'AdminScreen.html', {
                'Admin_name': "Administrator",
                'delete_job_status': False,
                'delete_job_status_message': error_message + ' - Job not found or
already deleted',
                'Admin_jobs_List': all_jobs_list
            })
    except pymysql.Error as db_err:
        return render(request, 'AdminScreen.html', {
            'Admin_name': "Administrator",
            'delete_job_status': False,
            'delete_job_status_message': error_message + f' - Database error:
{str(db_err)}',
            'Admin_jobs_List': all_jobs_list
        })
    finally:
        con.close()
        all_jobs_list = []
        update_all_jobs_list()

```

```

except Exception as e:
    return render(request, 'AdminScreen.html', {
        'Admin_name': "Administrator",
        'delete_job_status': False,
        'delete_job_status_message': error_message + f' - Unexpected error: {str(e)}',
        'Admin_jobs_List': all_jobs_list
    })
return render(request, 'AdminScreen.html', {
    'Admin_name': "Administrator",
    'delete_job_status': delete_job_status,
    'delete_job_status_message': status_message,
    'Admin_jobs_List': all_jobs_list
})

```

```

def UploadResume(request):
    global all_jobs_list
    all_jobs_list = []
    update_all_jobs_list()
    if request.method == 'GET':
        job_id = request.GET.get('t1', False)
        print(job_id)
        job_Name = request.GET.get('title', False)
        job_description = next(
            (job['description'] for job in all_jobs_list if str(job['job_id']) == job_id),
            "Description not available"
        )
        output = '<tr><td><font size=""
color="black">Job&nbsp;ID</b></td><td><input type="text" name="t1"
style="font-family: Comic Sans MS" size="30" value="'+job_id+'
readonly/></td></tr>'
        print("hit: "+job_description)
        context= {'data':output, 'job_title':job_Name, 'job_description': job_description
    }

```

```
return render(request, 'UploadResume.html', context)
```

```
def ViewJobs(request):
```

```
    if not checkUserLogin():
```

```
        return redirect('UserLogin')
```

```
    global all_jobs_list, user_job_list
```

```
    all_jobs_list = []
```

```
    user_job_list = []
```

```
    update_all_jobs_list()
```

```
    update_user_job_list()
```

```
    applied_job_ids = {job['job_id'] for job in user_job_list} # Set for faster lookup
```

```
    user_available_jobs = [job for job in all_jobs_list if job['job_id'] not in  
applied_job_ids]
```

```
    if request.method == 'GET':
```

```
        context= { 'Available_jobs':user_available_jobs}
```

```
        return render(request, 'UserScreen.html', context)
```

```
def ViewScore(request):
```

```
    if not checkadminloginstatus():
```

```
        return redirect('AdminLogin')
```

```
    applicants = []
```

```
    message = ""
```

```
    success_message = ""
```

```
    text = ""
```

```
    selectednumber = None
```

```
    count_selected_null = 0
```

```
    count_selected_0 = 0
```

```
    count_selected_1 = 0
```

```
    try:
```

```
        job_id = request.GET.get('jobid', None)
```

```
        job_title = request.GET.get('jobTitle', None)
```

```
        selected = request.GET.get('selected', None)
```

```
        con = pymysql.connect(host='127.0.0.1', port=3306, user='root',  
password='root',
```



```

        database='resumeanalysis', charset='utf8')

with con.cursor() as cur:
    query = """
        SELECT r.job_id, r.username, s.contact_no, s.email_id, r.resume_name,
               r.upload_date, r.selected, r.resume_json, r.resume_score
        FROM upload_resume r
        LEFT JOIN signup s ON r.username = s.username
    """

    params = []
    if job_id is not None:
        query += " WHERE r.job_id = %s"
        params.append(job_id)

    if selected is not None:
        if selected == 'None':
            text = "New"
            selectednumber = None
            query += " AND r.selected IS NULL"
        elif selected in ['0', '1']:
            selectednumber = int(selected)
            print("hit: ",selectednumber)
            text = "Selected" if selected == '1' else "Rejected"
            query += " AND r.selected = %s"
            params.append(selected)
        query += " ORDER BY r.upload_date ASC"
    cur.execute(query, params)
    rows = cur.fetchall()
    if rows:
        for row in rows:
            selected_status = "Under Review"
            selectednumber_res = -1;
            resume_json = json.loads(row[7])
            if row[6] is not None:
                selectednumber_res = row[6]

```

```

        selected_status = "Selected" if row[6] == 1 else "Rejected"
    applicants.append({
        "job_id": row[0],
        "job_name": job_title,
        "username": row[1],
        "contact_no": row[2] if row[2] else "N/A",
        "email_id": row[3] if row[3] else "N/A",
        "resume_name": row[4],
        "upload_date": row[5],
        "selected": selected_status,
        "selected_number": selectednumber_res,
        "resume_json": resume_json,
        "resume_score": row[8]
    })
    success_message = f"Successfully retrieved {len(applicants)}
applications for {job_title} job."
    else:
        message = f"No {text} applications found for {job_title} job."
    if job_id:
        count_query = """
        SELECT
            SUM(CASE WHEN selected IS NULL THEN 1 ELSE 0 END) AS
count_null,
            SUM(CASE WHEN selected = 0 THEN 1 ELSE 0 END) AS
count_0,
            SUM(CASE WHEN selected = 1 THEN 1 ELSE 0 END) AS count_1
        FROM upload_resume WHERE job_id = %s
        """
        cur.execute(count_query, (job_id,))
        result = cur.fetchone()
        count_selected_null, count_selected_0, count_selected_1 = result
    except Exception as e:
        message = f"An error occurred: {str(e)}"
    finally:

```

```

    if con:
        con.close()
context = {
    'job_id': job_id,
    'job_title': job_title,
    'applicants': applicants,
    'message': message,
    'success_message': success_message,
    'selected': selectednumber,
    'count_selected_null': count_selected_null,
    'count_selected_0': count_selected_0,
    'count_selected_1': count_selected_1
}
return render(request, 'ViewScore.html', context)

def AllViewScore(request):
    if not checkadminloginstatus():
        return redirect('AdminLogin')
    applicants = []
    message = ""
    success_message = ""
    selectednumber = None
    try:
        selected = request.GET.get('selected', None)
        con = pymysql.connect(host='127.0.0.1', port=3306, user='root',
password='root', database='resumeanalysis', charset='utf8')
        with con.cursor() as cur:
            query = """
                SELECT r.job_id, r.username, s.contact_no, s.email_id, r.resume_name,
                    r.upload_date, r.selected, r.resume_json, r.resume_score, p.job_name
                FROM upload_resume r
                LEFT JOIN signup s ON r.username = s.username
                LEFT JOIN postjob p ON r.job_id = p.job_id
            """

```

```

params = []
where_clause = []
if selected is not None:
    if selected == 'None':
        text = "New"
        selectednumber = None
        where_clause.append("r.selected IS NULL")
    elif selected in ['0', '1']:
        selectednumber = int(selected)
        text = "Selected" if selected == '1' else "Rejected"
        where_clause.append("r.selected = %s")
        params.append(selected)
if where_clause:
    query += " WHERE " + " AND ".join(where_clause)
query += " ORDER BY r.upload_date ASC"
cur.execute(query, params)
rows = cur.fetchall()
if rows:
    for row in rows:
        selected_status = "Under Review"
        selectednumber_res = -1
        resume_json = json.loads(row[7])
        if row[6] is not None:
            selectednumber_res = row[6]
            selected_status = "Selected" if row[6] == 1 else "Rejected"
        applicants.append({
            "job_id": row[0],
            "job_name": row[9],
            "username": row[1],
            "contact_no": row[2] if row[2] else "N/A",
            "email_id": row[3] if row[3] else "N/A",
            "resume_name": row[4],
            "upload_date": row[5],
            "selected": selected_status,

```

```

        "selected_number": selectednumber_res,
        "resume_json": resume_json,
        "resume_score": row[8]
    })
    if selected is not None:
        if selected == 'None':
            success_message = f"Successfully retrieved {len(applicants)} new
applications."
        elif selected == '0':
            success_message = f"Successfully retrieved {len(applicants)}
rejected applications."
        elif selected == '1':
            success_message = f"Successfully retrieved {len(applicants)}
selected applications."
        else:
            success_message = f"Successfully retrieved All {len(applicants)}
applications."
    else:
        if selected is not None:
            if selected == 'None':
                message = f"No new applications found."
            elif selected == '0':
                message = f"No rejected applications found."
            elif selected == '1':
                message = f"No selected applications found."
        else:
            message = f"No applications found."
except Exception as e:
    message = f"An error occurred: {str(e)}"
finally:
    if con:
        con.close()
context = {
    'type': 'All',

```

```

    'applicants': applicants,
    'message': message,
    'success_message': success_message,
    'selected': selectednumber,
}
return render(request, 'ViewScore.html', context)

def MakeSelected(request):
    if not checkadminloginstatus():
        return redirect('AdminLogin')
    username = request.GET.get('username', None)
    job_id = request.GET.get('job_id', None)
    if username and job_id:
        try:
            con = pymysql.connect(host='127.0.0.1', port=3306, user='root',
password='root',
                                database='resumeanalysis', charset='utf8')
            with con.cursor() as cur:
                cur.execute("UPDATE upload_resume SET selected = 1 WHERE
username = %s AND job_id = %s", (username, job_id))
                con.commit()
                makeSelected_statues = True
                message = f"Successfully selected {username} for {job_id} job."
            except Exception as e:
                message = f"An error occurred: {str(e)}"
            finally:
                if con:
                    con.close()
            else:
                message = "Invalid username or job_id provided."
            return JsonResponse({'action': 'Selected', 'makeSelected_statues':
makeSelected_statues, 'message': message})

def MakeRejected(request):

```

```

if not checkadminloginstatus():
    return redirect('AdminLogin')
username = request.GET.get('username', None)
job_id = request.GET.get('job_id', None)
MakeRejected_statues = False
if username and job_id:
    try:
        con = pymysql.connect(host='127.0.0.1', port=3306, user='root',
password='root',database='resumeanalysis', charset='utf8')
        with con.cursor() as cur:
            cur.execute("UPDATE upload_resume SET selected = 0 WHERE
username = %s AND job_id = %s", (username, job_id))
            con.commit()
            MakeRejected_statues = True
            message = f"Successfully rejected {username} for {job_id} job."
    except Exception as e:
        message = f"An error occurred: {str(e)}"
    finally:
        if con:
            con.close()
    else:
        message = "Invalid username or job_id provided."
    return JsonResponse({'action': 'Rejected', 'MakeRejected_statues':
MakeRejected_statues, 'message': message})

```

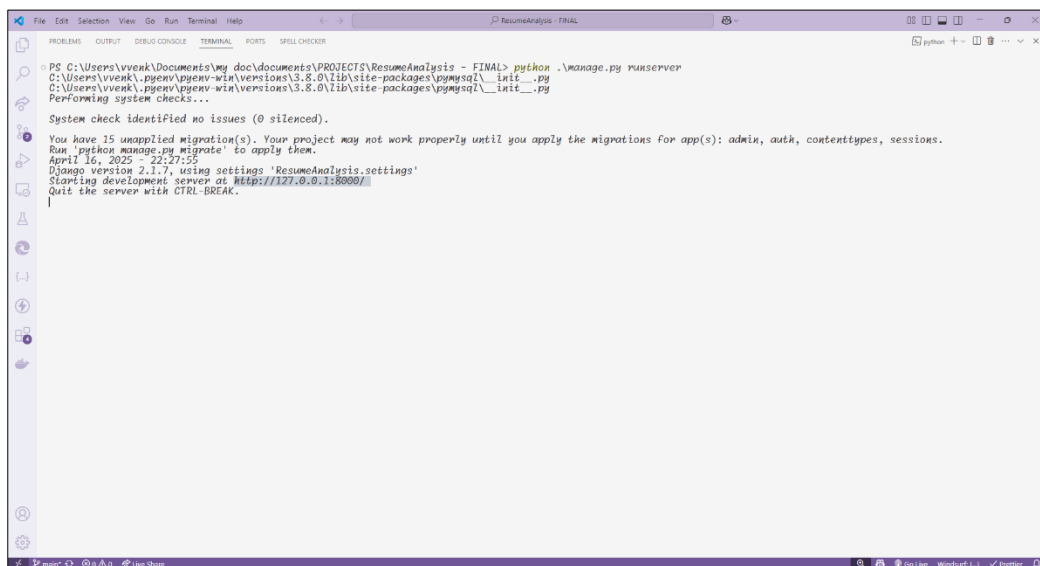
5. RESULTS & DISCUSSION

5. RESULTS & DISCUSSION

The following screenshots showcase the results of our project, highlighting key features and functionalities. These visual representations provide a clear overview of how the system performs under various conditions, demonstrating its effectiveness and user interface. The screenshots serve as a visual aid to support the project's technical and operational achievements.

5.1 Running Application:

In the screen shown below, the Django development server has been successfully started, indicating that the backend is running without any errors. To view the application, open a web browser and enter the following URL in the address bar: `http://127.0.0.1:8080`. This will load the Home page of the Django project, allowing you to interact with the application through its user interface. The URL points to the local server running on your machine at port 8080, which is the default development environment used during testing and development.



```

PS C:\Users\vvenk\Documents\My doc\documents\PROJECTS\ResumeAnalysis - FINAL> python .\manage.py runserver
C:\Users\vvenk\pgenv\pgenv-win\versions\3.8.0\lib\site-packages\pygments\__init__.py
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 16, 2025 - 22:27:55
Django version 2.1.7, using settings 'ResumeAnalysis.settings'
Starting development server at http://127.0.0.1:8080/
Quit the server with CTRL-BREAK.
  
```

Figure 5.1 : Terminal Output Indicating The Application Is Running of Automated Resume Analysis & Skill Matching Website Using NLP

5.2 Home Page:

The below screen shows the homepage of the "Automated Resume Analysis And Skill Matching With NLP" web application, featuring login options for admins and users, an "About Us" section, and a "Contact" link. The central focus of the application, as indicated by the title, is the automated analysis of resumes and skill matching using Natural Language Processing.

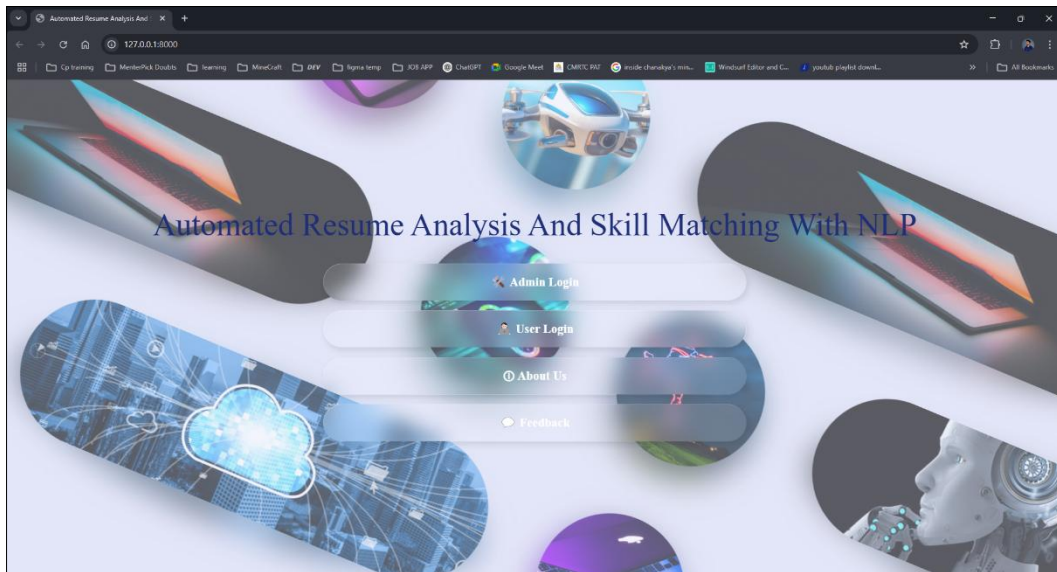


Figure 5.2 : Home Page of Automated Resume Analysis And Skill Matching With NLP

5.3 Login and Registration Screens:

In below screens, the application features distinct login interfaces for administrators and general users, along with a user registration page, all maintaining a consistent clean and modern design with a light gray background and subtle abstract dark blue circular elements.

Figure 5.3 displays the Admin Login screen designed specifically for administrators to access the system. It features a simple and clean interface with input fields for "USERNAME" and "PASSWORD", along with a clearly visible "SUBMIT" button. The background showcases a light gray theme with subtle, abstract dark blue circular elements, giving the interface a modern and minimalistic appearance.

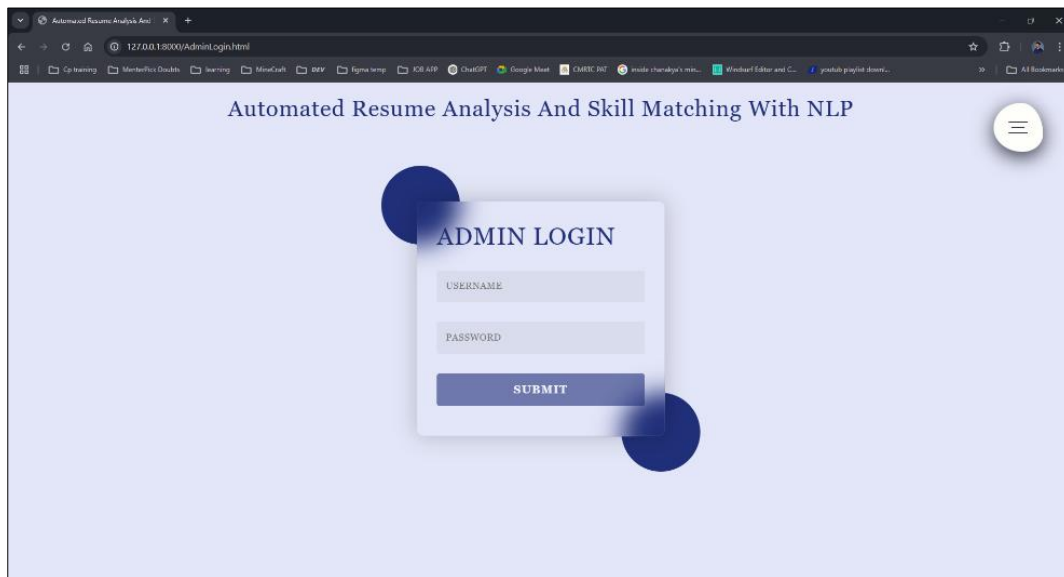


Figure 5.3 : Admin Login Page of Automated Resume Analysis And Skill Matching With NLP

Figure 5.4 displays the User Login screen, which allows general users to sign into the application. It includes fields for "USERNAME" and "PASSWORD", a "SUBMIT" button, and an additional "REGISTER" link for new users who need to sign up. The interface also incorporates a stylized illustration of a person interacting with digital elements, enhancing the visual appeal while maintaining design consistency.

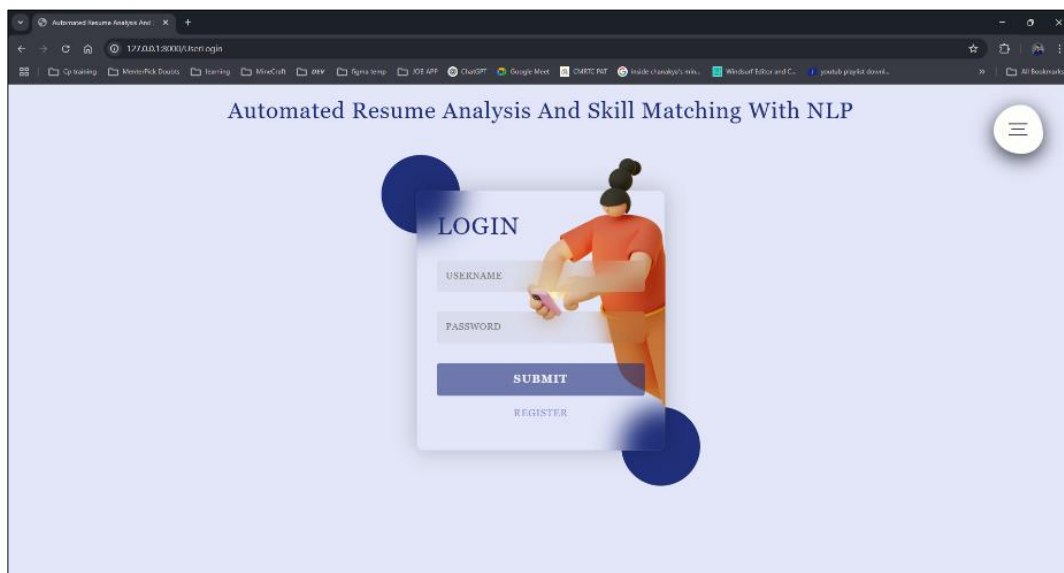


Figure 5.4 : User Login Page of Automated Resume Analysis And Skill Matching With NLP

Figure 5.5 displays the User Registration (Signup) screen, enabling new users to create an account. This screen includes input fields for "USERNAME," "PASSWORD," "CONTACT NO," "EMAIL ID," and "ADDRESS". It also provides a "SUBMIT" button to complete the registration process and a "LOGIN" link for users who already have an account. Like the user login screen, it features the same stylized illustration, reinforcing the unified design theme.

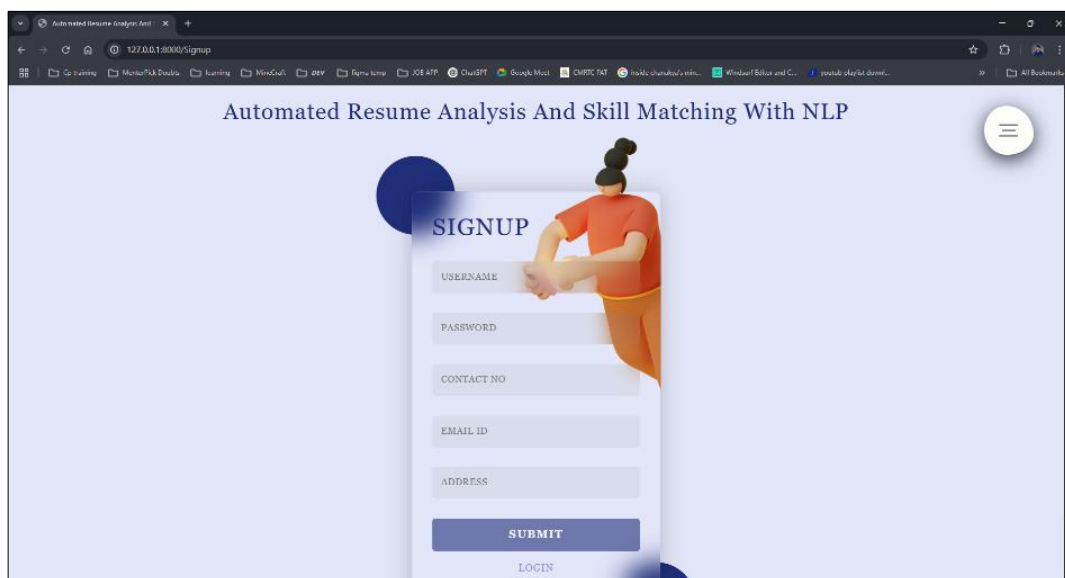


Figure 5.5 : User Registration Page of Automated Resume Analysis And Skill Matching With NLP

5.4 Feedback Page:

The below screen displays the Feedback Page of the "Automated Resume Analysis And Skill Matching With NLP" web application, where users can rate their satisfaction using a slider and provide detailed feedback in a text field before submitting it via the "Send Feedback" button.

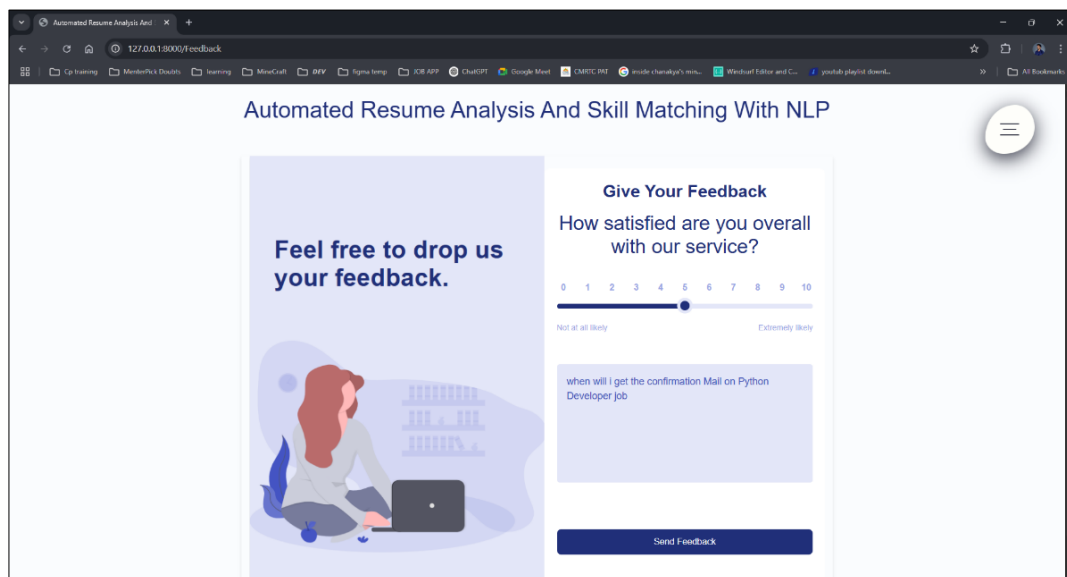


Figure 5.6 : Feedback Page of Automated Resume Analysis And Skill Matching With NLP

5.5 Admin Screens:

The below screen displays the Admin Dashboard of the "Automated Resume Analysis And Skill Matching With NLP" application. It presents a list of "All Posted Jobs" in a card-based layout. Each card represents a job posting and includes an icon related to the job role (e.g., a 'C' for UI/UX Designer, a Python logo for Python Developer), the job title, and potentially some additional interactive elements indicated by icons at the top of each card. The dashboard also features a welcome message for the administrator.

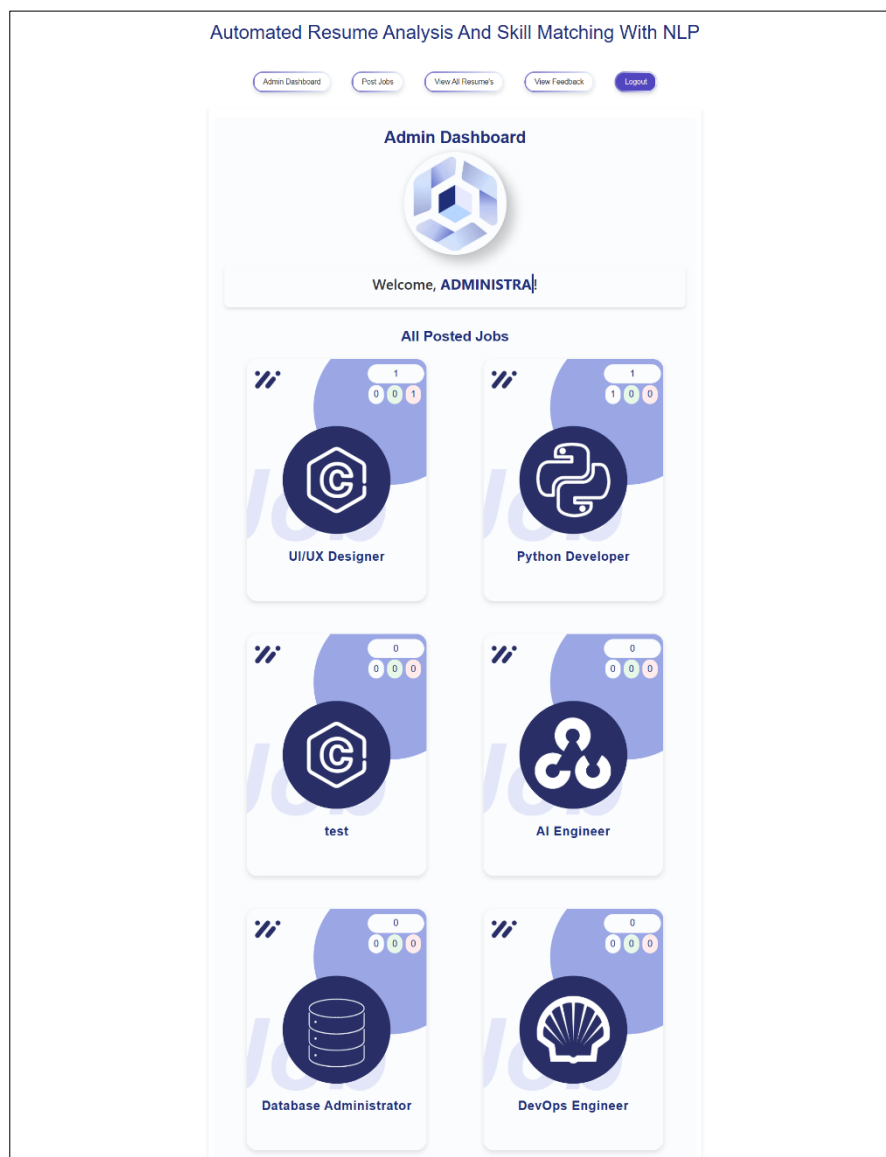


Figure 5.7 : Admin Dashboard Screen of Automated Resume Analysis And Skill Matching With NLP

The below screen displays the Post New Job Screen of the "Automated Resume Analysis And Skill Matching With NLP" application. It features input fields for "Job Title," "Job Details," "Company," "Salary," and a multi-select dropdown for "Skills." A "Post Job" button is located at the bottom to submit the new job posting.

Figure 5.8 : Post New Job Screen of Automated Resume Analysis And Skill Matching With NLP

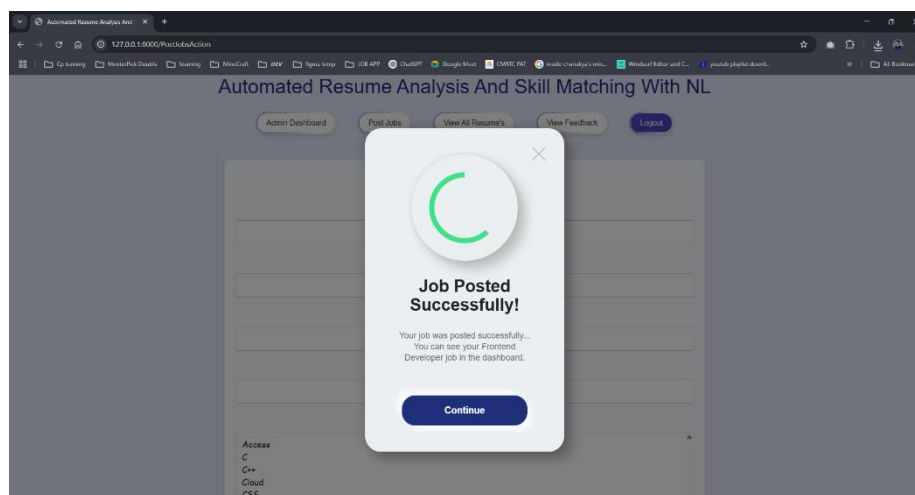


Figure 5.9 : Post New Job Successful Pop-Up Screen of Automated Resume Analysis And Skill Matching With NLP

The below screen displays a section where all job resume details can be viewed. Each resume is presented with key information such as the applied job title, resume filename, upload date, application status, and a list of skills. The interface also provides options to categorize or manage these resumes into different sections, as indicated by the presence of "Selected" and "Under Review" labels and corresponding "Rejected" and "Selected" buttons for each resume entry. Additionally, a navigation bar at the top allows filtering resumes based on categories like "All Resumes," "New Resumes," "Selected Resumes," and "Rejected Resumes."

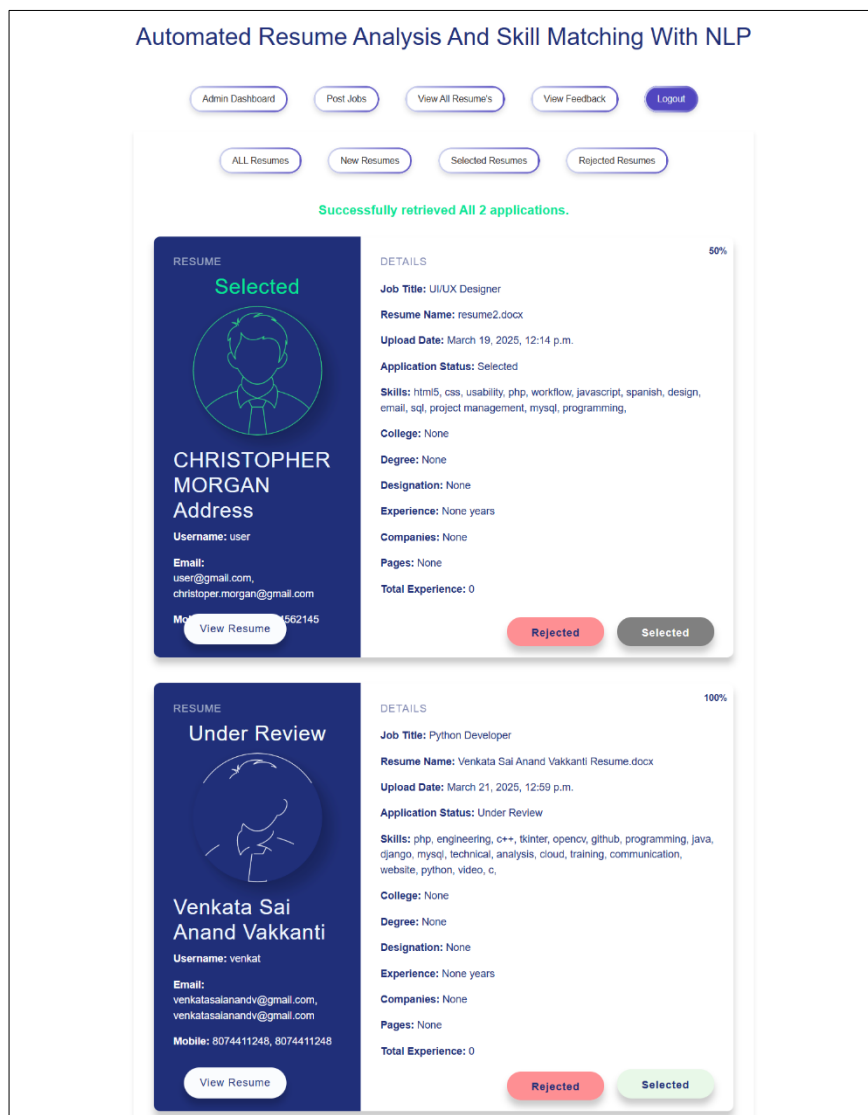


Figure 5.10 : All Job's Resumes View Screen of Automated Resume Analysis And Skill Matching With NLP

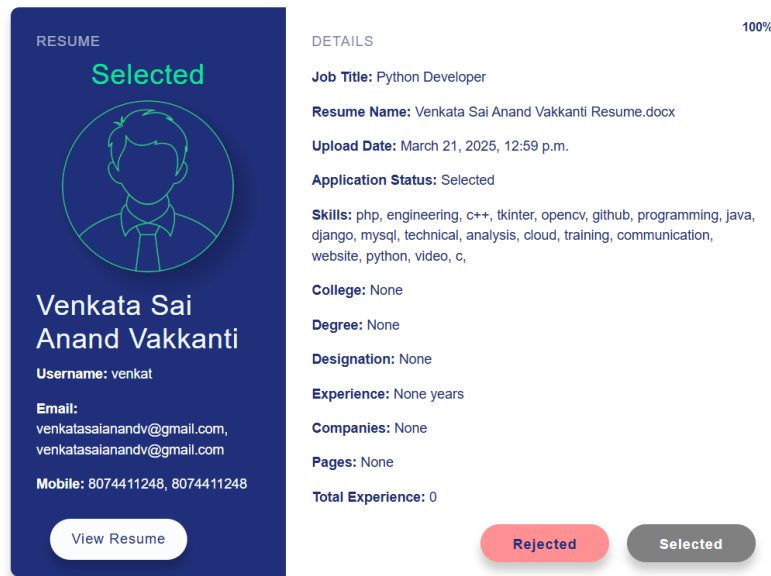


Figure 5.11 : Selected Job Detail Card of Automated Resume Analysis And Skill Matching With NLP

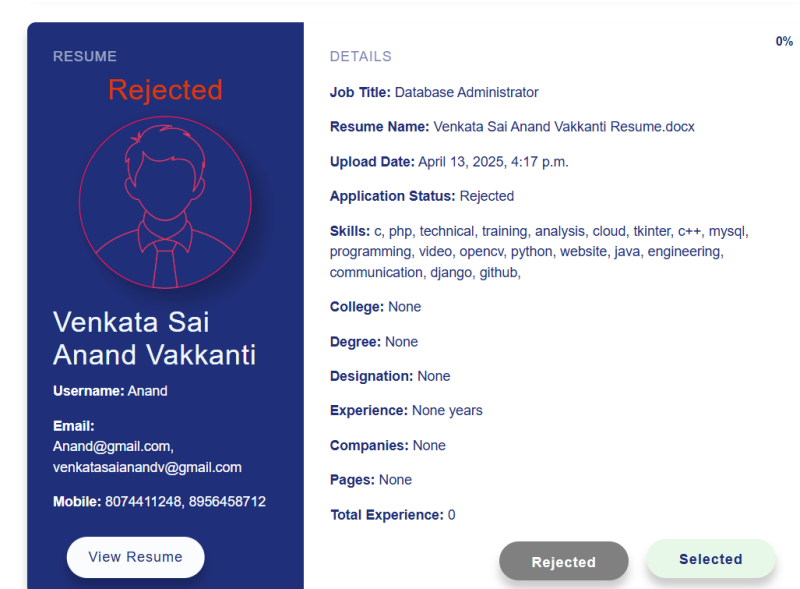


Figure 5.12 : Rejected Job Detail Card of Automated Resume Analysis And Skill Matching With NLP

The below screen displays the User Feedback View section of the "Automated Resume Analysis And Skill Matching With NLP" application. At the top, a pie chart visually represents the distribution of user satisfaction scores. Below the chart, individual User Feedbacks are listed, each attributed to "Anonymous." For each feedback entry, you can see the score provided by the user and their corresponding feedback text. This section allows administrators to get an overview of user satisfaction through the chart and read the specific comments provided by individual users. The top navigation bar provides links to other administrative sections of the application



Figure 5.13 : View User Feedback Screen of Automated Resume Analysis And Skill Matching With NLP

5.6 User Screens:

The below screen displays the User Dashboard of the "Automated Resume Analysis And Skill Matching With NLP" application. It welcomes the logged-in user. The section titled "My Applied Jobs" presents job applications in a card-based layout. Each job card displays the application status (e.g., "REJECTED," "SELECTED") along with the job title (e.g., "Database Administrator," "AI Engineer," "UI/UX Designer") and a relevant icon. The top navigation bar provides links to the "Dashboard," "View Available Jobs," and "Logout."

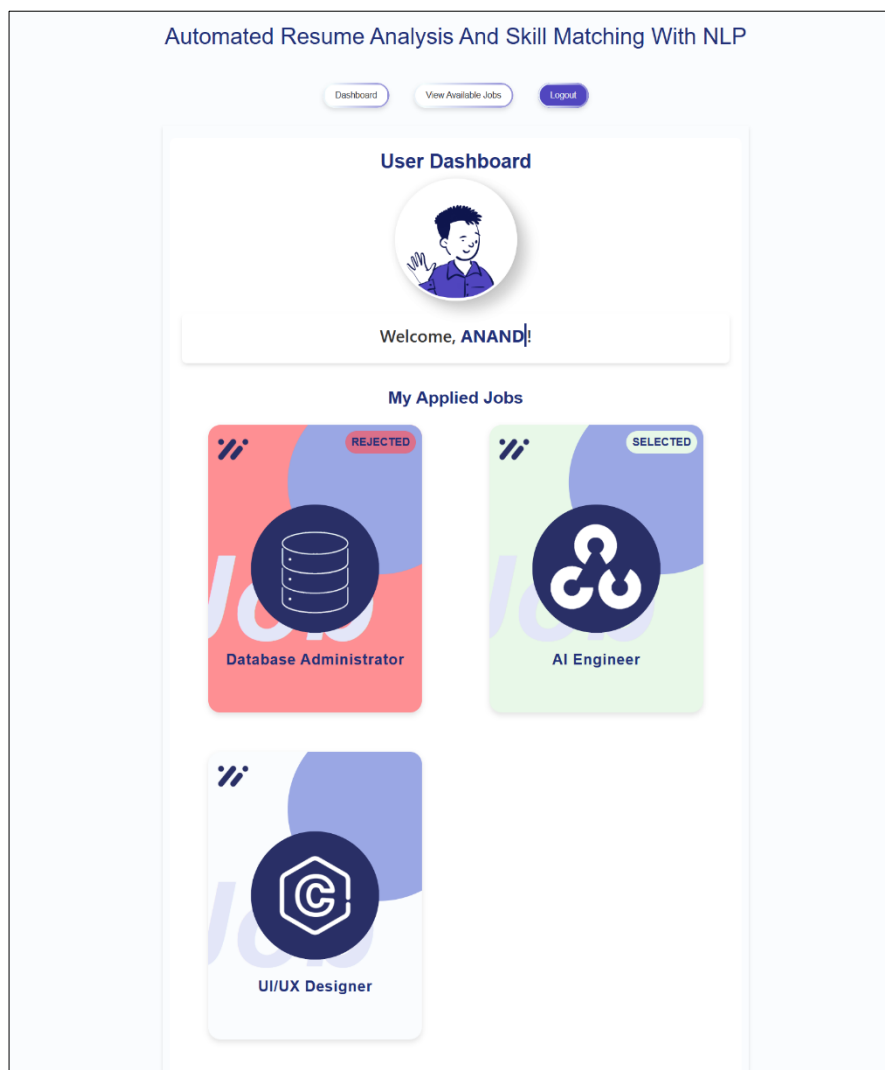


Figure 5.14 : User Dashboard Screen of Automated Resume Analysis And Skill Matching With NLP

The below screen displays the View Available Jobs section of the "Automated Resume Analysis And Skill Matching With NLP" application. It presents a list of job openings in a card-based layout. Each card displays the job title (e.g., "UI/UX Designer," "Python Developer," "AI Engineer," "Database Administrator," "DevOps Engineer," "Cloud Engineer," "Full Stack Developer," "Backend Developer," "Java Developer") along with a relevant icon.



Figure 5.15 : View All Available Job's Screen of Automated Resume Analysis And Skill Matching With NLP

The below screen displays the Upload Resume Page of the "Automated Resume Analysis And Skill Matching With NLP" application. It allows users to upload their resume for a specific job application. The page includes fields for "Job Title" (pre-filled or selectable, e.g., "UI/UX Designer") and "Job Description." There is a designated area to "Attach Your Resume" with instructions to "Drag and drop files here or select a file from your computer." It also indicates the accepted file types (e.g., "File types: doc, pdf"). A section shows the name of the uploaded resume file (e.g., "Uploaded: Venkata Sai Anand Vakkanti Resume.docx"). Finally, a "Submit Resume" button is available to complete the upload process.

Automated Resume Analysis And Skill Matching With NLP

Dashboard View Available Jobs Logout

Upload Resume

Job Title

UI/UX Designer

Job Description

We need a UI/UX Designer to craft intuitive and visually appealing user experiences.

Attach Your Resume

Drag and drop files here
or
select a file from your computer
File type: doc, pdf

Uploaded: Venkata Sai Anand Vakkanti Resume.docx

Submit Resume

Figure 5.16 : Upload Resume Page of Automated Resume Analysis And Skill Matching With NLP

5.7 User Screens:

In The following images illustrate job cards as they appear within the application under different circumstances, reflecting varying application statuses and potentially different options available to administrators versus regular users.



Figure 5.17 : Selected Job Card of Automated Resume Analysis And Skill Matching With NLP



Figure 5.18 : Rejected Job Card of Automated Resume Analysis And Skill Matching With NLP



Figure 5.19 : User View Job Card of Automated Resume Analysis And Skill Matching With NLP



Figure 5.20 : Admin View Job Card of Automated Resume Analysis And Skill Matching With NLP

6. VALIDATION

6. VALIDATION

The validation is a critical step in confirming the effectiveness and reliability of the implemented resume screening and ranking system. The goal is to ensure that the system performs consistently, extracts accurate information from resumes, and ranks candidates in a meaningful way.

To validate the system, a set of test resumes with varying formats, skills, and qualifications were uploaded and processed. The system was observed for correct extraction of details such as name, contact information, skills, education, and work experience. The parsed data was compared against the original content of the resumes to verify accuracy. In most cases, the parser accurately recognized and categorized the relevant information using SpaCy's NLP capabilities.

Furthermore, validation also included comparing the system's ranking output with manual evaluations done by recruiters. For each test case, recruiters were asked to manually rank candidates based on job requirements, and these rankings were then compared to the system-generated rankings. The results showed that the system was largely consistent with human judgment, especially in cases where the resumes were well-structured and clearly outlined skills and experience.

In addition, user feedback from recruiters and test candidates was collected to evaluate usability, system responsiveness, and overall satisfaction. This qualitative data was used to fine-tune the user interface and improve the transparency of the ranking logic, thereby enhancing trust in the system's recommendations.

Through these validation efforts, the system was proven to be effective, user-friendly, and accurate in screening and ranking resumes for the recruitment process. The insights gained during testing contributed to iterative improvements, strengthening the core functionality of the application. The ability of the system to adapt to different resume structures further affirmed its robustness. Overall, the validation process ensured that the solution not only met technical expectations but also aligned with real-world recruitment workflows. This confirms its potential to assist HR professionals in making informed and efficient hiring decisions.

6.1 INTRODUCTION

First, to validate the performance of the resume screening and ranking system, data was first prepared by collecting a variety of resumes in different formats, such as .pdf, .docx, and .txt. These files were then passed through the resume parsing module, and the extracted data was analyzed for accuracy and consistency. A structured dataset containing candidate information like name, email, skills, experience, and education was created to support this process.

The accuracy of information extraction and candidate ranking was assessed using key metrics such as precision, recall, and F1-score. These metrics helped evaluate the system's ability to correctly identify relevant skills and match them with the job description. Additionally, a confusion matrix was generated to observe misclassifications and improve the extraction logic. The proposed system was also tested against a basic keyword-matching system to demonstrate improvements in accuracy and candidate-job relevance.

Finally, real-world deployment tests were performed to evaluate how the system behaves with live user data, including resume uploads and recruiter actions. These tests ensured that the system responds efficiently, updates candidate status in real-time, and maintains accuracy across various industries and job roles. This comprehensive validation process confirms that the proposed system is robust, scalable, and suitable for real-time recruitment applications.

To further assess user satisfaction, feedback was collected from HR professionals and job seekers who interacted with the system during the pilot phase. Their inputs were instrumental in identifying usability bottlenecks and streamlining user navigation. Stress testing was also carried out to determine how the system performs under high-load scenarios, simulating multiple resume uploads and searches simultaneously. Performance metrics such as response time, memory usage, and error rate were monitored to ensure system stability. Security aspects were reviewed to confirm safe handling of sensitive personal data. The ability to integrate with existing HR software solutions was tested through API endpoints. These enhancements and evaluations contributed to shaping a more reliable and user-centric application.

6.2 TEST CASES

The following data is captured and stored in the database when a user uploads their resume through the system. The table includes essential candidate details such as name, email, contact number, and address, all of which are automatically extracted from the uploaded resume. Additionally, the resume file is saved in its original format for reference.

One of the key fields in this table is resume_json. This field stores the structured information extracted from the resume using Natural Language Processing (NLP) techniques powered by the SpaCy library. Once a resume is uploaded, the system processes it through the SpaCy pipeline to extract important elements such as skills, education, work experience, and other relevant attributes. The extracted data is then converted into a structured JSON format, which is stored in the resume_json field for further analysis and matching against job descriptions. This processed and structured data plays a crucial role in the resume ranking and job matching modules of the system, ensuring accurate and efficient recruitment decisions.

TABLE 6.1 UPLOADING DATA TO DATABASE

job_id	username	resume_name	upload_date	selected	resume_json	Resume score	Test Result
2	Venkat	Venkata Sai Anand Vakkanti Resume.docx	2025-03-21 12:59:49	NULL	{ "name": "Venkata Sai Anand Vakkanti", "email": "venkatasaianandv@gmail.com", "mobile_number": "8087459658", "skills": ["php", "engineering", "c++", "tkinter", "opencv", "github", "programming", "java", "django", "mysql", "technical", "analysis", "cloud", "training", "communication", "website", "python", "video", "c"], "total_experience": 0 }	70	YES

TABLE 6.2 UPDATED DATABASE upload_resume TABLE

job_id	username	resume_name	upload_date	selected	resume_json	resume_score
10	Manogna	Manogna.pdf	2025-03-19 12:14:36	1	{ "name": "Manogna", "email": "manogna@gmail.com", "mobile_number": "9848012345", "skills": ["php", "engineering", "c++", "tkinter", "opencv", "github", "programming", "java", "django", "mysql", "technical", "analysis", "cloud", "training", "communication", "website", "python", "video", "c"], "college_name": "CMRTC", "degree": "CSE", "designation": null, "experience": null, "company_names": null, "no_of_pages": null, "total_experience": 0 }	90
2	Venkat	Venkata Sai Anand Vakkanti Resume.docx	2025-03-21 12:59:49	NULL	{ "name": "Venkata Sai Anand Vakkanti", "email": "venkatasaiianandv@gmail.com", "mobile_number": "8087459658", "skills": ["php", "engineering", "c++", "tkinter", "opencv", "github", "programming", "java", "django", "mysql", "technical", "analysis", "cloud", "training", "communication", "website", "python", "video", "c"], "college_name": null, "degree": null, "designation": null, "experience": null, "company_names": null, "no_of_pages": null, "total_experience": 0 }	70

7. CONCLUSION & FUTURE ASPECTS

7. CONCLUSION & FUTURE ASPECTS

In conclusion, the project has successfully achieved its objectives, showcasing significant progress and outcomes. The implementation and execution phases were meticulously planned and executed, leading to substantial improvements and insights. Looking ahead, the future aspects of the project hold immense potential. Future developments will focus on expanding the scope, integrating new technologies, and enhancing sustainability. These advancements will not only strengthen the existing framework but also open new avenues for growth and innovation, ensuring the project remains relevant and impactful in the long term. This strategic approach will drive continuous improvement and success.

7.1 PROJECT CONCLUSION

The primary objective of this project was to design and implement an intelligent resume screening system that automates the candidate shortlisting process using Natural Language Processing (NLP) and Resume Parsing techniques. By integrating spaCy for NLP and applying structured resume extraction, the system effectively overcomes the limitations of traditional manual screening and keyword-based filtering methods.

The proposed system provides a seamless and user-friendly experience for both candidates and recruiters. Candidates are able to upload their resumes in any format, after which the system extracts relevant information such as skills, qualifications, work experience, and personal details. This extracted data is stored in a structured JSON format, which makes it easier to match against job requirements. Additionally, the platform maintains transparency by offering real-time application status updates to candidates.

On the recruiter's side, the system enables efficient data handling and candidate evaluation, reducing the time, effort, and potential biases involved in manual processes. It ensures a more accurate and scalable solution to resume screening, particularly useful in high-volume recruitment scenarios. In conclusion, the system proves to be a practical, scalable, and intelligent solution for modern recruitment challenges. It enhances the overall hiring experience by providing speed, accuracy, and fairness, ultimately increasing the likelihood of finding the best-suited candidates for specific roles.

7.2 FUTURE ASPECTS

Looking ahead, the proposed resume screening system has significant potential for further enhancement and broader application. One promising direction is the integration of APIs from professional platforms such as LinkedIn or popular job portals, allowing for real-time verification of candidate profiles and automated extraction of professional history. Additionally, the system could incorporate an AI-driven recommendation engine that suggests the most suitable job roles to applicants based on their skills, experience, and career preferences. Enhancements in machine learning could also enable predictive analytics, allowing recruiters to assess a candidate's potential success or cultural fit within the organization. Furthermore, the system could be expanded to support multilingual resume parsing, making it more inclusive and applicable in global recruitment scenarios. A recruiter-facing admin panel can be introduced to provide customizable filtering, detailed analytics, and improved communication tools. Finally, offering mobile app support would increase the platform's accessibility and usability for both candidates and recruiters. These future advancements would transform the current system into a comprehensive, intelligent recruitment solution capable of addressing the dynamic needs of modern hiring processes.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

- [1] Pradeep Kumar Roy, Vellore Institute of Technology, 2019. A Machine learning approach for automation of resume recommendation system, ICCIDS 2019. 10.1016/j.procs.2020.03.284.
- [2] Thimma Reddy Kalva, Utah State University, 2013. Skill-Finder: Automated Job-Resume Matching system. 3]Yong Luo, Nanyang Technological University, 2018. A Learning- Based Framework for automatic resume quality assessment, arXiv:1810.02832v1 cs.IR].
- [3] Suhjit Amin, Fr. Conceicao Rodrigues Institute of Technology, 2019. Web Application for Screening resume, IEEE DOI: 10.1109/ICNTE44896.2019.8945869.
- [4] Tejaswini K, Umadevi V, Shashank M Kadiwal, Sanjay Revanna, Design and Development of Machine Learning based Resume Ranking System (2021), DOI: <https://doi.org/10.1016/j.gltp.2021.10.002>.
- [5] Riza tana Fareed, rajah V, and Sharadadevi kaganumath, “Resume Classification and Ranking using KNN and Cosine Similarity” In 2021 International Journal of Engineering Research & Technology (IJERT), ISSN: 2278- 0181, Vol.10.
- [6] Suhas Tangadle Gopalakrishna, Vijayaraghavan Varadharajan, “Automated Tool for Resume Classification Using Semantic Analysis”, International Journal of Artificial Intelligence and Applications (IJAIA), Vol. 10, No.1, January 2019.

8.2 GITHUB LINK



https://github.com/vvenkatasaiand/Automated_Resume_Analysis_And_Skill_Matching_Website_using_NLP