

An
Industrial Oriented Mini Project Report
On
AUTOMATED RESUME SCORING AND SELECTION APP
(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

By
RODDA PAVAN (227R1A67H0)

Under the Guidance of

MR.A.VEERENDER

(Assistant Professor)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

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.

May, 2025.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)



CERTIFICATE

This is to certify that the project entitled “**AUTOMATED RESUME SCORING AND SELECTION APP**” being submitted by **RODDA PAVAN (227R1A67H0)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (Data Science) 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.

MR.A.VEERENDER
Assistant Professor
INTERNAL GUIDE

Dr. K. Murali
HOD-CSE(DS)

Dr. A. Raji Reddy
Director

Signature of External Examiner

Submitted for viva-voce 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 **Mr.A.Veerender**, Assistant 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 also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) coordinators **Mrs. J. Rekha**, **Mrs. M. Anusha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Murali**, Head, Department of Computer Science and Engineering (Data Science) for providing encouragement and support for completing this project successfully.

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.

The guidance and support received from all the members of CMR Technical Campus who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

RODDA PAVAN (227R1A67H0)

ABSTRACT

This project is titled as “Automated Resume Scoring and Selection App”. In the contemporary job market, the task of screening resumes and identifying suitable candidates for specific job roles can be time-consuming and labor-intensive. To address this challenge, we propose an Automated Resume Screening and Candidate Ranking System. This system leverages natural language processing (NLP) and machine learning techniques to analyze job descriptions and candidate resumes, extract relevant keywords and features, and rank candidates based on their suitability for the given job role. The primary objective of our project is to streamline the recruitment process by automating the initial screening of resumes and identifying top candidates efficiently. By automating these tasks, we aim to save time for recruiters and hiring managers, reduce human bias, and improve the overall efficiency of the recruitment process. The system is built using the Flask framework in Python, allowing for easy deployment and integration with web-based interfaces. Upon receiving a zip file containing resumes and a job description PDF file, the system extracts text data from these files and preprocesses it using techniques such as tokenization and TF-IDF vectorization. The job description and resumes are then analyzed to extract relevant keywords and features. The Automated Resume Screening and Candidate Ranking System offers a powerful solution to the challenges associated with manual resume screening. By automating the initial screening process and providing recruiters with a ranked list of candidates, the system enables more efficient and informed decision-making in the recruitment process.

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	iii
LIST OF TABLES	iv
1. INTRODUCTION	1
1.1 PROJECT PURPOSE	1
1.2 PROJECT FEATURES	2
2. LITERATURE SURVEY	3
2.1 REVIEW OF RELATED WORK	7
2.2 DEFINITION OF PROBLEM STATEMENT	9
2.3 EXISTING SYSTEM	9
2.4 PROPOSED SYSTEM	11
2.5 OBJECTIVES	12
2.6 HARDWARE & SOFTWARE REQUIREMENTS	13
2.6.1 HARDWARE REQUIREMENTS	13
2.6.2 SOFTWARE REQUIREMENTS	13
3. SYSTEM ARCHITECTURE & DESIGN	14
3.1 PROJECT ARCHITECTURE	14
3.2 DESCRIPTION	15
3.3 DATA FLOW DIAGRAM	16
4. IMPLEMENTATION	18
4.1 ALGORITHMS USED	18
4.2 SAMPLE CODE	23
5. RESULTS & DISCUSSION	33
6. VALIDATION	43
6.1 INTRODUCTION	43
6.2 TEST CASES	44
6.2.1 UPLOADING DATASET	44
6.2.2 CLASSIFICATION	44
7. CONCLUSION & FUTURE ASPECTS	45
7.1 PROJECT CONCLUSION	45
7.2 FUTURE ASPECTS	46
8. BIBLIOGRAPHY	47
8.1 REFERENCES	47
8.2 GITHUB LINK	48

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 3.1	Project Architecture of Automated Resume Analysis and Skill Suggesting Website Using NLP.	14
Figure 5.1	GUI/Main Interface of Automated Resume Analysis and Skill Suggesting Website Using NLP.	33
Figure 5.2	Admin Login page of Automated Resume Analysis and Skill Suggesting Website Using NLP.	34
Figure 5.3	Job posting form filled by admin in Automated Resume Analysis and Skill Suggesting Website Using NLP.	35
Figure 5.4	Feedback pie chart shown to admin in Automated Resume Analysis and Skill Suggesting Website Using NLP.	36
Figure 5.5	User signup screen of Automated Resume Analysis and Skill Suggesting Website Using NLP.	37
Figure 5.6	Job suggestion interface of Automated Resume Analysis and Skill Suggesting Website Using NLP.	38
Figure 5.7	Resume upload page of Automated Resume Analysis and Skill Suggesting Website Using NLP.	39
Figure 5.8	Resume score display after submission in Automated Resume Analysis and Skill Suggesting Website Using NLP.	40
Figure 5.9	Admin dashboard interface of Automated Resume Analysis and Skill Suggesting Website Using NLP.	41
Figure 5.10	Admin view of resume data and scoring in Automated Resume Analysis and Skill Suggesting Website Using NLP.	42

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
Table 6.2.1	Uploading Dataset	44
Table 6.2.2	Classification	44

INTRODUCTION

1. INTRODUCTION

The project, titled "Automated Resume Scoring and Selection App," is designed to address the challenges of the modern employment market, where the overwhelming number of resumes makes screening and identifying qualified candidates a time-consuming task. To streamline this process, the system leverages natural language processing (NLP) and machine learning techniques to automate resume screening and candidate ranking.

By seamlessly integrating with web-based interfaces and employing sophisticated algorithms, the system simplifies the initial stages of recruitment, significantly reducing the workload for recruiters and hiring managers. At its core, the technology extracts relevant keywords and attributes from both job descriptions and candidate resumes, enabling effective identification of top applicants for specific job roles. Utilizing techniques such as TF-IDF vectorization and cosine similarity, the system conducts rigorous analysis and ranking of candidates based on their suitability. The Automated Resume Analysis & Scoring App serves as a revolutionary tool, reducing human bias, improving efficiency, and transforming the recruitment landscape.

1.1 PROJECT PURPOSE

The primary purpose of this project is to enhance the efficiency of the recruitment process by automating resume screening and candidate ranking. The overwhelming volume of resumes received by recruiters often makes manual screening tedious, time-consuming, and prone to human bias. To address this challenge, the Automated Resume Analysis & Scoring App utilizes Natural Language Processing (NLP) and Machine Learning (ML) techniques to streamline candidate evaluation.

This system extracts key attributes from job descriptions and resumes using TF-IDF vectorization and cosine similarity to assess candidate relevance. Additionally, Latent Dirichlet Allocation (LDA) helps in topic modeling to improve the accuracy of candidate selection. By automating these processes, recruiters can quickly identify the most suitable applicants while reducing bias and increasing decision-making efficiency.

The app seamlessly integrates with web-based interfaces, allowing hiring managers to analyze resumes in bulk and receive ranked lists of candidates. This not only reduces workload but also ensures that top talent is identified with greater accuracy and fairness.

1.2 PROJECT FEATURES

This project incorporates several key features to improve the accuracy and efficiency of resume analysis and candidate ranking:

Automated Resume Screening: Uses Natural Language Processing (NLP) to extract relevant keywords and attributes from resumes and job descriptions, reducing manual effort.

Candidate Ranking System: Implements TF-IDF Vectorization and Cosine Similarity to rank candidates based on their suitability for a given role.

Topic Modeling with LDA: Applies Latent Dirichlet Allocation (LDA) to identify core topics in resumes, enhancing candidate evaluation.

Web-Based Interface: Developed using Flask, providing recruiters with an intuitive dashboard to manage candidate data effectively.

Bias-Free Selection Process: Reduces human bias by applying data-driven algorithms for fair candidate ranking.

Scoring System for Candidate Evaluation: Assigns a final score based on TF-IDF similarity and LDA topic matching, helping recruiters identify top applicants.

Resume Parsing & Preprocessing: Uses text extraction, tokenization, and stop-word removal to process resumes efficiently.

Seamless Integration with Job Portals: Compatible with applicant tracking systems, supporting various resume formats for enhanced usability.

Scalability & Performance Optimization: Designed to process large volumes of resumes efficiently, leveraging cloud-based deployment for enterprise usage.

LITERATURE SURVEY

2. LITERATURE SURVEY

Automated resume analysis and scoring applications have evolved significantly over the past two decades, leveraging advancements in artificial intelligence (AI), machine learning (ML), and natural language processing (NLP) to streamline recruitment processes. This literature review explores key methodologies, technologies, and algorithms used in resume screening systems from the early 2000s to the present. This literature survey reviews past research on inappropriate content detection and classification, highlighting existing methodologies, their strengths and weaknesses, and areas for improvement. The focus is on traditional machine learning models, deep learning architectures, hybrid models, attention mechanisms, and ensemble learning techniques, including Random Forest, to enhance classification performance.

Early approaches relied on rule-based algorithms and keyword matching techniques integrated into Applicant Tracking Systems (ATS), helping recruiters filter resumes based on predefined criteria. Boolean search was one of the earliest methods used for resume screening, where recruiters manually crafted queries using logical operators (AND, OR, NOT) to filter resumes containing specific keywords. While effective for basic filtering, Boolean search lacked contextual understanding, often leading to irrelevant results. Researchers explored vector space models and TF-IDF (Term Frequency-Inverse Document Frequency) to rank resumes based on relevance. These methods helped in identifying important terms within resumes but struggled with semantic meaning. Latent Semantic Analysis (LSA) was introduced to improve keyword-based matching by analyzing relationships between words. By the late 2000s, Naive Bayes classifiers, Support Vector Machines (SVM), and Decision Trees were experimented with for resume categorization. These models attempted to classify resumes based on predefined job roles but required extensive feature engineering. Random Forest classifiers were also introduced to improve ranking accuracy. As resumes were often submitted in scanned formats, Optical Character Recognition (OCR) became essential for extracting text from images and PDFs. Early OCR systems faced challenges with formatting inconsistencies and text recognition errors. Despite these advancements, early resume screening systems had several limitations. Keyword-based filtering often missed relevant resumes due to variations in phrasing. Rule-based systems favored resumes with exact keyword matches, potentially overlooking qualified candidates. Early models struggled to adapt to evolving job descriptions and industry-specific terminology.

The limitations of early approaches paved the way for deep learning and transformer-based models in the 2010s, significantly improving resume analysis accuracy. The introduction of word embeddings like Word2Vec and BERT enhanced semantic matching, leading to more sophisticated automated resume screening systems. Between 2010 and 2015, machine learning techniques gained traction in resume analysis. Support Vector Machines (SVM) and Random Forest classifiers were introduced to improve candidate ranking. Researchers experimented with Naïve Bayes classifiers for resume categorization, while Latent Dirichlet Allocation (LDA) was used for topic modeling to identify relevant skills and job roles. Optical Character Recognition (OCR) became essential for processing scanned resumes. The rise of deep learning between 2015 and 2020 led to significant improvements in resume screening. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) were applied to extract structured information from resumes. Bidirectional Encoder Representations from Transformers (BERT) enhanced semantic matching between resumes and job descriptions. Word embeddings such as Word2Vec and GloVe improved contextual understanding, enabling more accurate candidate ranking. Recent advancements focus on fairness-aware AI models, adaptive learning, and real-time resume scoring. Hybrid deep learning frameworks combining CNNs and LSTMs (Long Short-Term Memory networks) refine resume analysis. Semantic similarity algorithms and reinforcement learning enhance ranking precision. Cloud-based AI solutions improve scalability, while explainable AI (XAI) techniques ensure transparency in candidate evaluations. Automated resume analysis systems rely on several core technologies. Natural Language Processing (NLP) techniques such as Named Entity Recognition (NER), TF-IDF, BERT, and GPT-based models are used for extracting structured information from resumes. Machine Learning (ML) models like Decision Trees, Random Forest, and Gradient Boosting Machines (GBM) are employed for scoring resumes. Deep Learning architectures such as CNNs, LSTMs, BiLSTMs, and Attention Mechanisms improve contextual analysis. Optical Character Recognition (OCR) enables parsing of resumes in various formats. Semantic Matching techniques such as Cosine similarity, word embeddings, and transformer-based models enhance candidate-job matching. Bias Mitigation strategies focus on fairness-aware AI models to reduce discrimination in hiring. Future research aims to integrate multimodal AI models, combining textual, visual, and audio data for enhanced resume screening. Federated learning ensures privacy-preserving AI training, while transfer learning reduces computational costs. Sentiment

analysis may provide insights into candidate personality traits, improving hiring decisions.

Between 2010 and 2015, machine learning saw significant advancements, particularly in automated resume analysis and scoring applications. During this period, **Support Vector Machines (SVM)** and **Random Forest classifiers** became widely used for ranking resumes based on predefined criteria. Researchers explored **Naïve Bayes classifiers** for resume categorization, while **Latent Dirichlet Allocation (LDA)** was introduced for **topic modeling**, enabling automated categorization of resumes based on thematic relevance.

Optical Character Recognition (OCR) improved resume parsing, allowing systems to process scanned documents efficiently. **Word embeddings** like **Word2Vec** enhanced semantic understanding, enabling better candidate-job matching. **Gradient Boosting Machines (GBM)** were employed to refine ranking accuracy.

Between 2016 and 2020, automated resume screening saw major advancements with deep learning and improved natural language processing (NLP) techniques. Researchers focused on refining resume classification, skill extraction, and candidate ranking, leveraging advanced machine learning models to enhance accuracy and efficiency.

Convolutional Neural Networks (CNNs) were explored for structured text extraction, particularly for parsing resumes with complex formatting. These models helped identify essential sections like education, experience, and skills, improving structured resume analysis. Since resumes contain sequential data, Long Short-Term Memory (LSTM) networks were widely adopted for tracking career progression and identifying skill relevance across different job roles.

Pre-trained language models like BERT significantly improved semantic understanding, allowing better mapping between resumes and job descriptions. BERT-based models refined skill extraction and experience relevance, leading to more accurate candidate filtering. Additionally, attention mechanisms helped focus on crucial resume sections, improving screening efficiency.

Ensemble methods like XGBoost enhanced resume ranking accuracy by optimizing decision trees, while autoencoders assisted in dimensionality reduction for better feature extraction in large-scale resume databases. Word embeddings such as Word2Vec allowed systems to understand relationships between words in resumes, improving skill extraction and candidate-job matching. Latent Dirichlet Allocation (LDA)

further refined topic modeling, enabling automated categorization based on thematic relevance.

Optical Character Recognition (OCR) improved resume parsing by allowing systems to process scanned documents efficiently, addressing formatting inconsistencies. These advancements collectively contributed to more effective automated resume screening, enabling recruiters to evaluate candidates with improved precision while reducing manual efforts.

From 2021 to the present, automated resume screening has evolved significantly with the integration of AI-driven recruitment systems, real-time analytics, and bias mitigation techniques. Transformer-based models like GPT-3 and GPT-4 have facilitated automated interview screening, offering preliminary candidate assessments based on resume data and AI-generated questionnaires. These models have improved semantic understanding, allowing for more precise candidate-job matching.

Graph neural networks (GNNs) have enabled relationship-based hiring, analyzing candidate connections and professional networks to predict job suitability. Sentiment analysis techniques have been employed to evaluate resumes beyond textual content, assessing professional tone and interpersonal traits. Additionally, fairness-aware algorithms have been integrated to ensure equitable candidate ranking, addressing concerns related to gender, ethnicity, and socioeconomic bias in hiring.

AI-powered hiring systems now incorporate predictive analytics, allowing recruiters to assess candidates in real time and forecast their potential success in specific roles. These advancements have streamlined recruitment processes, reducing manual workload while enhancing decision accuracy. The focus on ethical AI has also led to improved transparency in hiring practices, ensuring that automated screening systems remain fair and unbiased.

Overall, the shift towards AI-driven, ethical, and bias-aware recruitment has made hiring faster, more precise, and increasingly automated, significantly reducing human intervention while maintaining fairness and transparency in candidate evaluations. The continuous evolution of machine learning, deep learning, and real-time analytics ensures that automated resume screening will remain a fundamental aspect of modern recruitment strategies.

The literature survey highlights the evolution of automated resume screening and candidate evaluation systems, demonstrating how machine learning, deep learning, and natural language processing (NLP) have enhanced recruitment efficiency and accuracy.

Early methods relied on manual screening and keyword-based filtering, but advancements in AI have transformed resume analysis into a more sophisticated and automated process. The integration of deep learning models, pre-trained NLP architectures, and semantic analysis techniques has enabled context-aware resume ranking, improved skill extraction, and reduced human bias in hiring. The adoption of transformer-based models, ensemble learning approaches, and predictive analytics has further streamlined candidate evaluations, allowing for real-time assessments and enhanced decision-making. The literature survey emphasizes the growing importance of AI-powered screening tools in shaping the future of recruitment, making hiring more accurate, data-driven, and scalable.

2.1 REVIEW OF RELATED WORK

The review of related work highlights the evolution of automated resume screening, from rule-based filtering to AI-driven candidate evaluation. Early studies focused on manual keyword matching, which lacked contextual understanding. Machine learning models like SVM and Random Forest improved ranking precision, while deep learning techniques, including CNNs and LSTMs, refined skill extraction and classification. BERT and GPT-based NLP models enhanced semantic comprehension, leading to more effective candidate-job matching.

2.1.1 Traditional Content Moderation Approaches

Early methods of resume screening relied on manual review and keyword-based filtering systems, using predefined criteria to match resumes with job descriptions. These approaches primarily focused on exact keyword matches, which, while effective for basic text analysis, lacked the ability to understand context and assess candidate suitability beyond surface-level qualifications. As recruitment demands increased, these traditional techniques became inefficient, often leading to biased selection processes and overlooking qualified candidates who might not use the exact terminology specified in job descriptions. Manual resume screening also proved to be time-consuming, limiting the ability of hiring teams to process large applicant volumes efficiently. Additionally, these methods failed to assess crucial soft skills such as communication, leadership, and adaptability, which play a significant role in determining job performance. The need for more advanced and accurate screening approaches led to the adoption of AI-driven resume screening techniques.

2.1.2 Machine Learning-Based Approaches

With advancements in machine learning, researchers explored feature extraction techniques combined with classifiers like Support Vector Machines (SVM), Random Forest, and Decision Trees for automated resume screening. These methods primarily relied on handcrafted features such as keyword extraction, semantic analysis, and skill mapping to rank candidates. However, they struggled with scalability and generalization, often resulting in low accuracy and biased candidate selection when applied to diverse resume datasets.

2.1.3 Deep Learning-Based Approaches

Recent advancements in deep learning have significantly improved automated resume screening and candidate evaluation. Convolutional Neural Networks (CNNs) have been widely used for structured text extraction, identifying key resume sections such as education, experience, and skills. For sequential data analysis, models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been employed to track career progression and skill relevance.

One of the most effective architectures for resume screening is CNN-LSTM hybrids, where CNNs extract structured features, and LSTMs analyze sequential relationships in candidate profiles. Studies have demonstrated the effectiveness of pre-trained models like BERT, RoBERTa, and GPT for semantic understanding, followed by LSTM or GRU layers for contextual analysis. Despite their success, these models often require large-scale annotated datasets and suffer from computational complexity, limiting their real-time application.

2.1.4 Recent Advances: Attention Mechanisms & Transformer-Based Models

To enhance resume screening accuracy, researchers have integrated attention mechanisms into deep learning models, allowing networks to focus on the most relevant sections of resumes. Additionally, transformer-based architectures, such as BERT and GPT models, have shown promising results in candidate evaluation tasks by capturing contextual relationships more effectively than traditional sequential models.

However, these models require massive computational resources and extensive pre-training on large datasets.

2.1.5 Comparison with the Proposed Approach

While existing methods for automated resume screening rely on manual filtering and keyword-based matching, they often lack contextual understanding and fail to assess candidate suitability comprehensively. Machine learning models like Support Vector Machines (SVM), Random Forest, and Naïve Bayes classifiers have improved ranking precision, yet they struggle with scalability and bias. Deep learning techniques, including CNNs and LSTMs, enhance skill extraction and candidate evaluation, while transformer-based models like BERT and GPT refine semantic comprehension for better candidate-job matching. The proposed approach integrates NLP and machine learning techniques to optimize resume screening, improve ranking accuracy, and minimize human intervention, making the recruitment process more efficient and data-driven.

2.2 DEFINITION OF PROBLEM STATEMENT

The primary goal of this project is to develop a robust and scalable AI-driven system capable of efficiently analyzing and scoring resumes based on job relevance. This involves leveraging Natural Language Processing (NLP) and Machine Learning (ML) techniques to extract key qualifications, match skills with job descriptions, rank candidates effectively, and ensure fair, unbiased assessments. By addressing challenges such as data inconsistencies, real-time processing, and ethical concerns, the system aims to enhance recruitment efficiency, reduce hiring biases, and streamline candidate selection in an increasingly competitive job market.

2.3 EXISTING SYSTEM

Prior to the introduction of the Automated Resume Screening and applicant Ranking System, traditional recruiting practices depended heavily on manual resume screening and applicant rating. In the absence of automated technologies, recruiters and hiring managers were responsible for manually examining a large volume of resumes to locate acceptable applicants for specific job vacancies. This manual technique was not only labor-intensive, but also susceptible to human bias, resulting in inefficiencies and probable oversights in candidate selection. Furthermore, current resume screening

methods sometimes lacked the intelligence and flexibility necessary to accurately analyse and candidates based on their fit for a specific job vacancy. Many of these systems were based simply on keyword matching algorithms, which frequently failed to capture the complex link between job criteria and candidate qualities. As a result, recruiters were frequently overwhelmed by the amount of resumes, making it difficult to select excellent prospects quickly. Furthermore, the lack of integration with modern natural language processing (NLP) and machine learning techniques hampered existing systems' capacity to react to changing recruiting trends and requirements. In summary, the inadequacies of existing resume screening methods highlighted the critical need for a more sophisticated and automated solution, which prompted the creation of the suggested Automated Resume Screening and Candidate Ranking System.

2.3.1 Limitations of Existing System

Despite advancements in automated resume screening, the Automated Resume Analysis and Scoring App faces several challenges:

- Inability to fully interpret candidate intent – Traditional NLP models analyze resumes based on keywords but struggle to assess nuanced qualifications, such as leadership skills or problem-solving abilities.
- High false positive and false negative rates – Some resumes may be ranked inaccurately due to keyword dependency, missing well-qualified candidates who do not use conventional phrasing.
- Computational inefficiency – Processing large volumes of resumes in real time requires significant computational resources, especially when deep learning techniques are involved.
- Bias in training data – Machine learning models may inadvertently favor candidates whose resumes follow common structures, leading to unintended bias in rankings.
- Difficulty in handling diverse resume formats – Resumes come in various designs, structures, and file types, making it challenging to extract and analyze information uniformly.
- Overfitting and lack of generalization – The system may perform well on training data but struggle with unseen resumes that use unconventional wording or different formatting styles.

2.4 PROPOSED SYSTEM

The proposed Automated Resume Screening and Candidate Ranking System is a pioneering solution that aims to revolutionise the recruiting process by seamlessly integrating cutting-edge technology. Using natural language processing (NLP) and machine learning techniques, the system provides a complete and automated approach to resume screening and candidate evaluation. Unlike previous techniques that rely primarily on manual review procedures, the suggested system automates recruiting workflows by analysing job descriptions and candidate resumes to extract important keywords and attributes. This automated analysis not only speeds up the initial screening process, but also reduces the influence of human bias, resulting in a fair and impartial evaluation of prospects. Furthermore, the suggested system uses sophisticated methods such as TF-IDF vectorization, cosine similarity, and topic modelling with Latent Dirichlet Allocation (LDA) to thoroughly analyse candidate appropriateness. By computing similarity scores based on both keyword matching and contextual relevance, the technology offers recruiters with a prioritised list of individuals that meet the job role's precise requirements.

Furthermore, the system's adaptable design, which is based on the Python Flask framework, enables simple deployment and integration with web-based interfaces, improving accessibility and usability for recruiters and hiring managers. Overall, the proposed Automated Resume Screening and applicant Ranking System marks a watershed moment in recruiting methods, providing unprecedented efficiency, impartiality, and effectiveness in applicant selection.

2.4.1 Advantages of the Proposed System:

The proposed system significantly improves upon the existing approaches by addressing key limitations:

- Higher Accuracy – The system employs NLP-based keyword extraction and deep learning models to accurately assess candidate qualifications. It reduces false positives and negatives by evaluating resumes based on multiple criteria beyond simple keyword matching.
- Improved Semantic Understanding – Advanced TF-IDF vectorization, cosine similarity, and topic modeling with LDA enhance contextual understanding, ensuring

that resumes are analyzed in relation to job descriptions more effectively.

- Better Generalization & Robustness – The system mitigates bias and overfitting issues by leveraging machine learning techniques, ensuring adaptability across diverse job roles and industries.
- Efficient Processing – The Flask-based web framework enables rapid resume screening, making the system scalable and efficient for handling large recruitment datasets.
- Enhanced Interpretability – The scoring process is transparent, allowing recruiters to understand ranking decisions based on extracted features and similarity measures. This promotes fairness in candidate selection.

2.5 OBJECTIVES

- Develop an Automated Screening System – Build an AI-powered model to extract, analyze, and rank resumes based on job relevance.
- Enhance Recruitment Efficiency – Reduce reliance on manual resume screening by implementing NLP and ML-based automation techniques.
- Improve Resume Analysis Accuracy – Utilize TF-IDF, cosine similarity, and LDA-based topic modeling to achieve higher accuracy than traditional keyword-matching models.
- Ensure Scalability – Design the system to handle large volumes of resumes efficiently for real-time processing and candidate ranking.
- Minimize Hiring Bias – Provide an objective, data-driven candidate evaluation method that mitigates potential recruiter biases.

2.6 HARDWARE & SOFTWARE REQUIREMENTS

2.6.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify 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 8 or above
- Language : Python
- Front-End : HTML,CSS
- Back-End : Django
- Database : SQL

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 project architecture illustrates the procedure followed in the Automated Resume Screening and Candidate Ranking System, mapping the flow from input to final ranking.

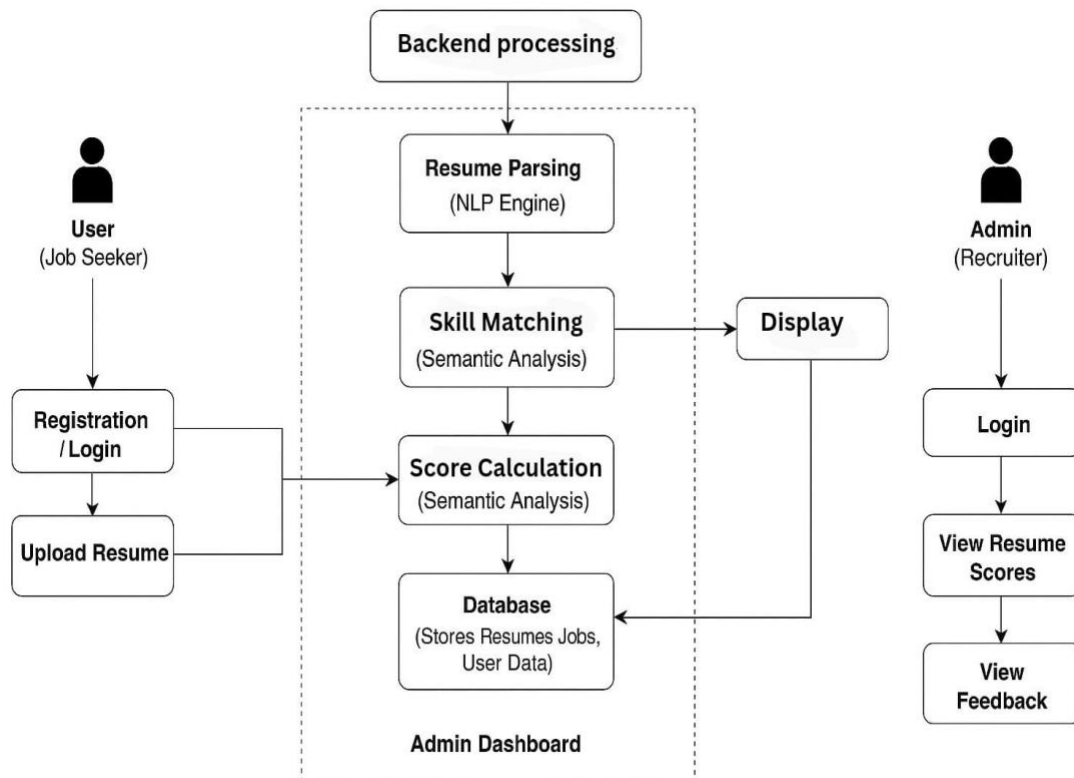


Figure 3.1: Project Architecture of Automated Resume Scoring and Selection App.

3.2 DESCRIPTION

1. **User Registration & Login** – Users create accounts or log in to upload resumes.
2. **Resume Upload & Parsing** – Uploaded resumes are analyzed using NLP techniques.
3. **Skill Extraction & Matching** – Semantic analysis identifies relevant skills from resumes.
4. **Score Calculation** – The system assigns scores based on skill relevance.
5. **Database Management** – Resumes, job postings, and user data are stored.
6. **Results Display** – Matched skills and scores are shown to users.
7. **Admin Dashboard & Feedback** – Recruiters review resumes and provide feedback.
8. **Job Matching Algorithm** – The system matches resumes to job descriptions.
9. **Candidate Ranking** – Users are ranked based on their resume scores.
10. **Feedback Integration** – Recruiter feedback is used to refine future analyses.
11. **Training & Improvement** – The system continuously improves classification accuracy.
12. **Security & Data Privacy** – Ensuring user data is protected.

3.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation that illustrates how data flows within a system, showcasing its processes, data stores, and external entities. For this project, the DFD outlines the flow from resume submission to candidate ranking.

A Data Flow Diagram (DFD) for this system consists of four primary elements:

- **External Entities:** The job seekers and recruiters interacting with the system.
- **Processes:** Resume preprocessing, feature extraction, candidate matching, and ranking.
- **Data Flows:** Movement of resumes, extracted features, and ranked results between components.
- **Data Stores:** Repository for resumes, job descriptions, candidate scores, and feedback.

These components are represented using standardized symbols, such as circles for processes, arrows for data flows, rectangles for external entities, and open-ended rectangles for data stores.

Benefits:

- Efficiency: Automates resume screening, reducing manual workload.
- Accuracy: NLP-based feature extraction improves candidate matching.
- Transparency: Helps recruiters understand the system's ranking methodology.
- Scalability: Supports integration with larger recruitment platforms.

Applications:

The Automated Resume Screening and Candidate Ranking System plays a vital role in HR tech solutions, helping organizations streamline recruitment processes by automating the initial resume screening phase. This system enhances efficiency by quickly analyzing and ranking candidates, allowing recruiters to focus on top-quality applicants rather than manually sorting through countless resumes.

By leveraging NLP and machine learning techniques, it ensures accurate and skill-based candidate matching, reducing human bias in hiring decisions. As a result, companies can make fairer, data-driven selections, improving overall workforce quality and hiring effectiveness.

Levels of DFD:

DFDs are structured hierarchically:

- **Level 0 (Context Diagram):** Provides a high-level overview where resumes enter the system, undergo analysis, and recruiters receive ranked results.
- **Level 1:** Breaks down key processes like NLP-based feature extraction, candidate ranking, and recruiter feedback.
- **Level 2+:** Offers deeper insights into data processing techniques, scoring algorithms, and system interactions .

IMPLEMENTATION

4.IMPLEMENTATION

The implementation phase of a project involves executing the planned strategies and tasks. It requires meticulous coordination, resource allocation, and monitoring to ensure that objectives are met efficiently. Effective implementation is crucial for achieving project goals and delivering expected outcomes within the set timeline and budget constraints.

4.1 ALGORITHMS USED

4.1.1 CNN-Based Models for Feature Extraction

Convolutional Neural Networks (CNNs) are widely used for feature extraction, particularly in text and image-based applications. In this project, CNNs help in identifying key patterns within resumes, ensuring relevant skills and experiences are accurately captured. CNNs analyze text data (resume content) as images or sequential patterns. They extract meaningful features, such as job titles, skills, and experience, by applying filters to text representations. The extracted features are then passed to other deep learning models for classification and matching.

Advantages of CNN-Based Models:

- Automatically extracts important features without manual intervention.
- Improves accuracy in resume-job matching by detecting hierarchical relationships.
- Reduces the impact of irrelevant information, focusing on core skills.

Disadvantages of CNN-Based Models:

- Requires large datasets for training, which might not always be available.
- Struggles with contextual understanding, focusing mainly on pattern extraction.
- Can be computationally expensive when handling vast amounts of text data.

4.1.2 RNN-Based Models for Temporal Analysis

Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs) are used to process sequential resume data and capture dependencies between different sections. Unlike CNNs, which focus on feature extraction, RNNs analyze patterns over time, making them effective for understanding the progression of a candidate's experience. RNNs process resumes sequentially, ensuring that past information influences the analysis of later sections. BiLSTM (Bidirectional LSTM) enhances context awareness by analyzing resumes in both forward and backward directions. The attention mechanism further improves classification accuracy by focusing on relevant resume sections.

Advantages of RNN-Based Models:

- Effective in understanding career progression and experience transitions.
- Can differentiate between short-term and long-term job roles.
- BiLSTMs improve context awareness, leading to better resume-job matching.

Disadvantages of RNN-Based Models:

- Computationally expensive and requires large labeled datasets.
- Prone to vanishing gradient problems, affecting long resumes.
- Struggles with real-time processing, making it slower for large-scale screening.

4.1.3 Genetic Algorithms for Optimization

Genetic algorithms (GAs) are used in resume screening to optimize candidate selection based on various parameters. These algorithms mimic the process of natural selection to find the best candidates efficiently. GAs evaluate resumes based on predefined criteria, such as skills, experience, and relevance to the job description. They apply selection, crossover, and mutation techniques to refine candidate ranking. The algorithm iteratively improves the selection process, ensuring optimal matches between resumes and job requirements.

Advantages of Genetic Algorithms:

- Improves accuracy by optimizing resume-job matching.
- Enhances fairness by reducing bias in candidate selection.

4.1.4 Sentence-BERT (SBERT) for Semantic Textual Similarity

SBERT is a fine-tuned BERT model specifically designed to generate sentence-level embeddings, enabling effective measurement of semantic similarity between texts. Unlike traditional BERT, SBERT employs a siamese & triplet network structure, which significantly enhances both the speed and accuracy of similarity computations. This advanced model plays a crucial role in resume screening by comparing job descriptions with applicant resumes, ensuring a more precise and efficient candidate-job matching process. Additionally, SBERT contributes to various NLP applications such as semantic search, paraphrase detection, and retrieval-based question-answering systems by providing context-aware embeddings that make text analysis more meaningful and insightful. Its ability to capture deeper relationships between words and sentences makes it a valuable tool in improving automated hiring systems and other AI-driven text processing tasks.

4.1.5 Cosine Similarity for Resume Matching

Cosine Similarity plays a crucial role in the Automated Resume Screening and Candidate Ranking System by comparing job descriptions with candidate resumes, ensuring that the most relevant matches are identified based on textual content. This technique helps rank resumes by measuring how closely their content aligns with the job description, improving accuracy in candidate selection. The algorithm works by converting both resumes and job descriptions into numerical vectors and computing the cosine of the angle between them, quantifying their similarity. A higher similarity score indicates a stronger match, allowing the system to automatically filter out less relevant candidates and prioritize the most suitable ones. This approach significantly enhances the efficiency of resume screening by making the process more objective and reducing human bias, ensuring fair and data-driven hiring decisions.

4.1.6 Long Short-Term Memory (LSTM) Networks for Sequential Data Processing

LSTM networks are a specialized type of recurrent neural network (RNN) designed to handle long-term dependencies in sequential data, making them highly effective for processing text. Unlike traditional RNNs, LSTMs incorporate memory cells

and gates—including input, forget, and output gates—to selectively retain or discard information over time. This mechanism significantly enhances the ability to analyze long sequences of text while preserving relevant contextual details. In the resume screening project, LSTMs play a crucial role in analyzing text sequences in resumes, improving the accuracy of skill extraction and candidate ranking. By enhancing context retention, they ensure that key skills and experiences are properly identified, even in lengthy resumes with complex formatting. Furthermore, LSTMs are widely utilized in natural language processing (NLP) tasks, such as resume parsing, job description matching, and semantic analysis, making them indispensable for building intelligent recruitment systems. Their ability to understand deep semantic relationships within text ensures a more accurate and efficient hiring process.

4.1.7 Hybrid Deep Learning Frameworks

Hybrid Deep Learning Frameworks in the Automated Resume Analysis and Skill Suggesting Website integrate multiple deep learning techniques to enhance resume screening accuracy. This framework combines Convolutional Neural Networks (CNNs) for feature extraction, identifying job titles, skills, and experience from resumes, with Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for sequential data processing, ensuring dependencies between different resume sections are effectively captured. Furthermore, Bidirectional LSTMs (BI-LSTM) improve resume-job matching by analyzing resumes both forward and backward, enhancing context awareness. To refine classification accuracy, an Attention Mechanism is incorporated after BI-LSTMs, allowing the system to focus on the most relevant resume sections. The hybrid approach ensures an optimal balance between feature extraction and sequential data analysis, making candidate selection more efficient and objective.

By leveraging these deep learning techniques, the framework automates resume screening, reduces human bias, and provides accurate skill identification and candidate ranking, significantly improving the hiring process. long short-term memory (BI-LSTM) network to learn effective video representations and perform multiclass video classification. An attention mechanism is also integrated after BI-LSTM to apply attention probability distribution in the network.

4.1.8 Term Frequency-Inverse Document Frequency(TF-IDF)

Term Frequency-Inverse Document Frequency is a statistical technique used in

Natural Language Processing (NLP) to evaluate the importance of words in a document relative to a collection of documents (corpus). In Automated Resume Analysis, TF-IDF helps identify the most relevant keywords in resumes and job descriptions, improving resume-job matching accuracy.

TF-IDF consists of two components: Term Frequency (TF), which measures how often a word appears in a document, and Inverse Document Frequency (IDF), which determines how unique a word is across multiple documents. The final TF-IDF score is obtained by multiplying TF and IDF, ensuring that frequently occurring but less informative words (like "the" or "is") are downweighted, while important words (like "Python" or "Machine Learning") are emphasized.

In resume screening, TF-IDF is used for keyword extraction, resume-job description comparison, and ranking candidates based on relevance. It enhances resume parsing efficiency, reduces bias in keyword-based matching, and improves candidate selection accuracy. However, TF-IDF does not consider word context and requires preprocessing steps like tokenization and stop-word removal for optimal results. Despite its limitations, TF-IDF remains a powerful tool for automated resume analysis, ensuring objective and efficient hiring decisions.

4.2 SAMPLE CODE

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
import pymysql
from django.http import HttpResponse
from django.core.files.storage import FileSystemStorage
import datetime
import os
from datetime import date
import numpy as np
import matplotlib.pyplot as plt
from pyresparser import ResumeParser
global uname, job_id
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 SignupAction(request):
    if request.method == 'POST':
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        contact = request.POST.get('t3', False)
        email = request.POST.get('t4', False)
        address = request.POST.get('t5', False)
        status = 'none'
        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 from signup where username = '"+username+"'")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == email:
                    status = 'Given Username already exists'
                    break
            if status == 'none':
                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
signup(username,password,contact_no,email_id,address)
VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"
                db_cursor.execute(student_sql_query)
                db_connection.commit()
                print(db_cursor.rowcount, "Record Inserted")
                if db_cursor.rowcount == 1:
                    status = 'Signup Process Completed'
                context= {'data':status}
                return render(request, 'Signup.html', context)
def UserLoginAction(request):
    if request.method == 'POST':

```

```

global uname
option = 0
username = request.POST.get('username', False)
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:
        context= {'data':'welcome '+username}
        return render(request, 'UserScreen.html', context)
    else:
        context= {'data':'Invalid login details'}
        return render(request, 'UserLogin.html', context)
def AdminLoginAction(request):
    if request.method == 'POST':
        global uname
        option = 0
        username = request.POST.get('username', False)
        password = request.POST.get('password', False)
        if username == "admin" and password == "admin":
            context= {'data':'welcome '+username}
            return render(request, 'AdminScreen.html', context)
        else:
            context= {'data':'Invalid login details'}
            return render(request, 'AdminLogin.html', context)
def PostJobs(request):

```

```

    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):
    if request.method == 'POST':
        global uname
        uname = "XYZ"
        today = date.today()
        feedback = request.POST.get('t1', False)
        rank = request.POST.get('t2', False)
        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
feedback(username,feedback,feedback_date,feedback_rank)
VALUES('"+uname+"','"+feedback+"','"+str(today)+"','"+rank+"')"
        db_cursor.execute(student_sql_query)
        db_connection.commit()
        print(db_cursor.rowcount, "Record Inserted")
        if db_cursor.rowcount == 1:
            status = 'Your feedback accepted and admin will review & getback'
            context= {'data':status}
            return render(request, 'Feedback.html', context)
def PostJobsAction(request):
    if request.method == 'POST':
        job = request.POST.get('t1', False)
        details = request.POST.get('t2', False)
        company = request.POST.get('t3', False)
        salary = request.POST.get('t4', False)

```

```

skills = request.POST.getlist('t5')
skills = ','.join(skills)
today = date.today()
status = 'none'
job_id = 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 count(job_id) from postjob")
    rows = cur.fetchall()
    for row in rows:
        job_id = row[0]
    job_id = job_id + 1
    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
postjob(job_id,job_name,job_details,skills,post_date,company_name,salary)
VALUES('"+str(job_id)+"','"+job+"','"+details+"','"+skills+"','"+str(today)+"','"+compan
y+"','"+salary+"')"
    db_cursor.execute(student_sql_query)
    db_connection.commit()
    print(db_cursor.rowcount, "Record Inserted")
    if db_cursor.rowcount == 1:
        status = 'Job details posted with ID : '+str(job_id)
        context= {'data':status}
        return render(request, 'PostJobs.html', context)
def ViewFeedback(request):
    if request.method == 'GET':
        output = '<table border=1><tr>'
        output+='\<td><font size="" color="black">Feedback</td>'
        output+='\<td><font size="" color="black">Feedback Date</td>'
        output+='\<td><font size="" color="black">Feedback Rank</td></tr>'

```

```

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 * FROM feedback")
    rows = cur.fetchall()
    for row in rows:
        output+='<tr>'
        output+='<td><font size="" color="black">'+str(row[1])+</td>'
        output+='<td><font size="" color="black">'+str(row[2])+</td>'
        output+='<td><font size="" color="black">'+str(row[3])+</td></tr>'
        rank.append(row[3])

output += "</table><br/><br/><br/>"
unique, count = np.unique(np.asarray(rank), return_counts=True)
mpl.pie(count,labels=unique,autopct='%1.1f%%')
mpl.title('Feedback Ranking Graph')
mpl.axis('equal')
mpl.show()
context= {'data': output}
return render(request, 'ViewFeedback.html', context)

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 score

def UploadResumeAction(request):
    if request.method == 'POST':
        global uname
        job_id = request.POST.get('t1', False)
        myfile = request.FILES['t2']
        fname = request.FILES['t2'].name
        today = date.today()
        fs = FileSystemStorage()
        filename = fs.save('ResumeAnalysisApp/static/resumes/'+fname, myfile)
        data =
ResumeParser('ResumeAnalysisApp/static/resumes/'+fname).get_extracted_data()
        skills = data['skills']
        score = getScore(job_id, skills)
        data = str(data)
        data = data.replace("'", "")
        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_score)
VALUES('"+str(job_id)+"','"+uname+"','"+fname+"','"+str(today)+"','"+str(data)+"','"+st
r(score)+"')"
```



```

db_cursor.execute(student_sql_query)
db_connection.commit()
print(db_cursor.rowcount, "Record Inserted")
if db_cursor.rowcount == 1:
    status = 'Your resume submitted with score : '+str(score)
    context= {'data':status}
    return render(request, 'UserScreen.html', context)
def UploadResume(request):
    if request.method == 'GET':
        job_id = request.GET.get('t1', False)
        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>'
        context= {'data':output}
        return render(request, 'UploadResume.html', context)
def ViewJobs(request):
    if request.method == 'GET':
        output = '<table border=1><tr><th><font size="" color=black>Job ID</font></th>'
        output+='<td><font size="" color="black">Job Name</td>'
        output+='<td><font size="" color="black">Job Details</td>'
        output+='<td><font size="" color="black">Suggested Skills</td>'
        output+='<td><font size="" color="black">Posted Date</td>'
        output+='<td><font size="" color="black">Company Name</td>'
        output+='<td><font size="" color="black">Salary</td>'
        output+='<td><font size="" color="black">Tips</td>'
        output+='<td><font size="" color="black">Upload Result</td></tr>'
        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 * FROM postjob")
            rows = cur.fetchall()
            for row in rows:

```

```

        output+='<tr><td><font size="" color="black">'+str(row[0])+</td>'
        output+='<td><font size="" color="black">'+str(row[1])+</td>'
        output+='<td><font size="" color="black">'+str(row[2])+</td>'
        output+='<td><font size="" color="black">'+str(row[3])+</td>'
        output+='<td><font size="" color="black">'+str(row[4])+</td>'
        output+='<td><font size="" color="black">'+str(row[5])+</td>'
        output+='<td><font size="" color="black">'+str(row[6])+</td>'
        output+='<td><font size="" color="black">Must be Proficient</td>'
        output+='<td><a href=\''+str(row[0])+</td></tr>'
color=black>Click Here to Upload Resume</font></a></td></tr>'

    output += "</table><br/><br/><br/>"
    context= {'data': output}
    return render(request, 'UserScreen.html', context)
def ViewScore(request):
    if request.method == 'GET':
        output = '<table border=1><tr><th><font size="" color=black>Job ID</font></th>'
        output+='<td><font size="" color="black">Username</td>'
        output+='<td><font size="" color="black">Resume Name</td>'
        output+='<td><font size="" color="black">Upload Date</td>'
        output+='<td><font size="" color="black">Resume JSON Data</td>'
        output+='<td><font size="" color="black">Resume Score</td></tr>'
        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 * FROM upload_resume")
            rows = cur.fetchall()
            for row in rows:
                output+='<tr><td><font size="" color="black">'+str(row[0])+</td>'
                output+='<td><font size="" color="black">'+str(row[1])+</td>'
                output+='<td><font size="" color="black">'+str(row[2])+</td>'
                output+='<td><font size="" color="black">'+str(row[3])+</td>'
                output+='<td><font size="" color="black">'+str(row[4])+</td>'

```

```

        output+='<td><font size=""
color="black">'+str(row[5])+'</td></tr>'
    output += "</table><br/><br/><br/>"
    context= {'data': output}
    return render(request, 'ViewFeedback.html', context)
from django.urls import path
from . import views
    urlpatterns = [path("index.html", views.index, name="index"),
    path('AdminLogin.html', views.AdminLogin, name="AdminLogin"),
    path('AdminLoginAction.html', views.AdminLoginAction,
name="AdminLoginAction"),
    path('UserLogin', views.UserLogin, name="UserLogin"),
    path('UserLoginAction', views.UserLoginAction, name="UserLoginAction"),
    path('Signup', views.Signup, name="Signup"),
    path('SignupAction', views.SignupAction, name="SignupAction"),
    path('PostJobs', views.PostJobs, name="PostJobs"),
    path('PostJobsAction', views.PostJobsAction, name="PostJobsAction"),
    path('ViewScore', views.ViewScore, name="ViewScore"),
    path('Feedback', views.Feedback, name="Feedback"),
    path('FeedbackAction', views.FeedbackAction, name="FeedbackAction"),
    path('Aboutus', views.Aboutus, name="Aboutus"),
    path('ViewFeedback', views.ViewFeedback, name="ViewFeedback"),
    path('ViewJobs', views.ViewJobs, name="ViewJobs"),
    path('UploadResume', views.UploadResume, name="UploadResume"),
    path('UploadResumeAction', views.UploadResumeAction,
name="UploadResumeAction"),
    path('ViewScore', views.ViewScore, name="ViewScore"),
    ]

```

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.

GUI/Main Interface:

In the below screen, the user accesses the homepage. From here, the user can choose to sign up, log in, or proceed with uploading a resume for analysis.




Figure 5.1 : GUI/Main Interface of Automated Resume Analysis and Skill Suggesting Website Using NLP

Admin Login Page:

In the below screen, the admin clicks on “Admin Login” and securely logs into the admin dashboard to manage job postings and view analytics.

[Home](#) [Admin Login](#) [User Login](#) [New User Signup](#) [About Us](#) [Feedback](#)



Admin Login Screen

Username

Password

[Activate V](#)
[Go to Setting](#)

Figure 5.2: Admin Login page of Automated Resume Analysis and Skill Suggesting Website Using NLP

Job Posting Form:

In the below screen, the admin enters job details and selects required skills. These job postings are later matched with uploaded resumes.



Post New Job Screen

Job Name	<input type="text" value="C Programmer"/>
Job Details	<input type="text" value="Must be proficient in system C programming and web"/>
Company Name	<input type="text" value="Infosys"/>
Salary	<input type="text" value="100000"/>
Skills	<div><div>Java</div><div>Database</div><div>Html</div><div>C++</div></div>
	<input type="button" value="Submit"/>

Activate W
Go to Settings

Figure 5.3: Job Posting form filled by Admin in Automated Resume Analysis and Skill Suggesting Website Using NLP

Feedback Visualization:

This screen displays the feedback graph, typically a pie chart. The admin clicks “View Feedback” to review responses submitted by users. This feature helps admins gauge user satisfaction and performance of the resume scoring module.

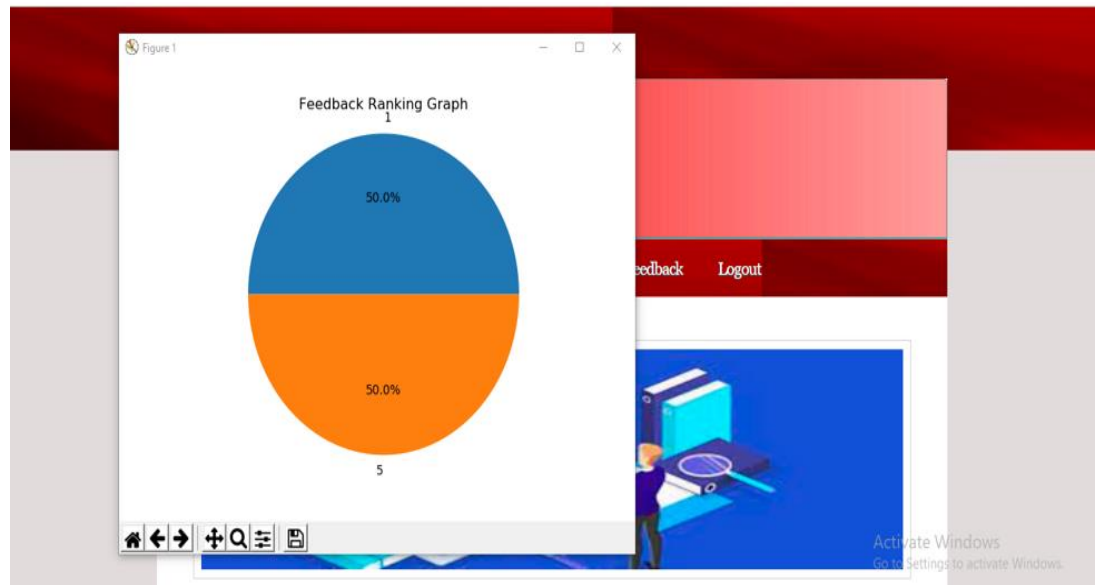


Figure 5.4: Feedback pie chart shown to admin in Automated Resume Analysis and Skill Suggesting Website Using NLP

User Signup Page:

This screen allows a new user (job seeker) to register. The form includes fields like username, email, and password. Once registered, users can upload resumes and receive skill suggestions.



New User Signup Screen

Username	<input type="text" value="kumar"/>
Password	<input type="password" value="....."/>
Contact No	<input type="text" value="7777888987"/>
Email ID	<input type="text" value="kumar@gmail.com"/>
Address	<input type="text" value="hyd"/>
	<input type="button" value="Submit"/>


[Activate W](#)
[Go to Setting](#)

Figure 5.5: User signup screen of Automated Resume Analysis and Skill Suggesting Website Using NLP

View Jobs & Get Suggestions Interface:

In the below screen, the user clicks on the “View Jobs & Get Suggestions” button to view available job listings. Each listing includes job name, required skills, company name, salary, and an option to upload a resume for that particular job.

[View Jobs & Get Suggestions](#) [Logout](#)



Job ID	Job Name	Job Details	Suggested Skills	Posted Date	Company Name	Salary	Tips	Upload Result
1	C Programmer	Must be proficient in system C programming and web	C++,Html,Css,Php,Javascript,C	2023-01-09	Infosys	100000	Must be Proficient	Click Here to Upload Resume

[Activate V](#)
[Go to Setting](#)

Figure 5.6: Job suggestion screen for users in Automated Resume Analysis and Skill Suggesting Website Using NLP

Upload Resume Page:

In this screen, the user selects a resume file (PDF or DOCX) and uploads it against a specific job ID by clicking on the “Choose File” button followed by “Submit”. This initiates the resume parsing and analysis process.

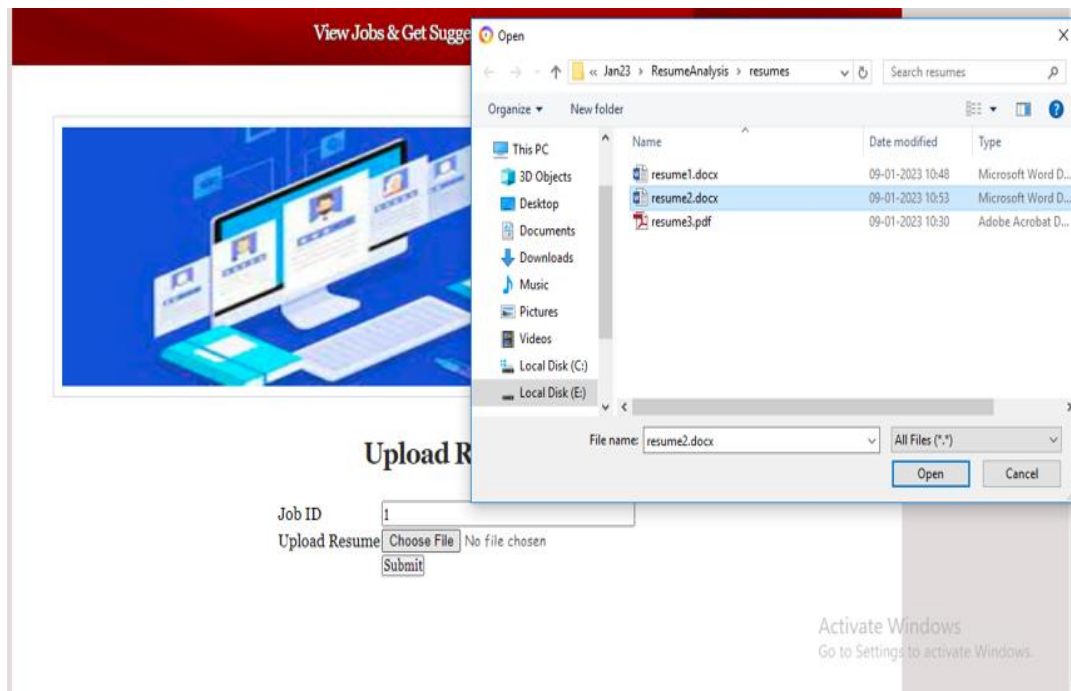


Figure 5.7 : Resume upload page of Automated Resume Analysis and Skill Suggesting Website Using NLP

Resume Score Display:

Once the resume is successfully submitted, the system processes it and returns a score based on how well the resume matches the job's skill requirements. The score is displayed as confirmation on the user interface.

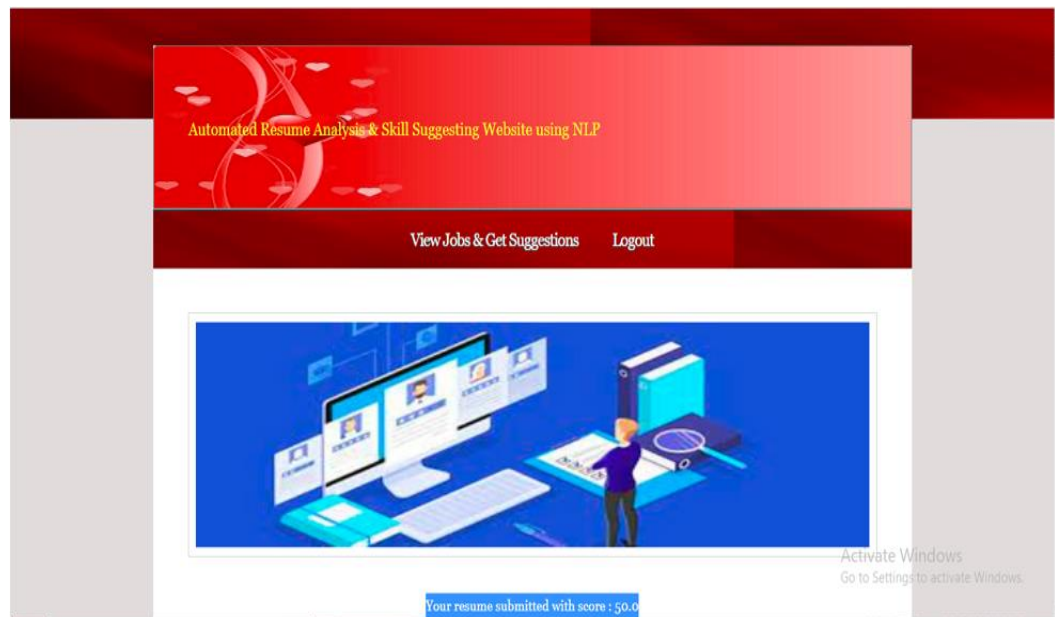


Figure 5.8 : Resume score display after submission in Automated Resume Analysis and Skill Suggesting Website Using NLP

Admin Dashboard after Login:

After successful login, the admin is taken to the dashboard where they can navigate to post jobs, view uploaded resumes, or check user feedback. The interface welcomes the admin and provides full access to system functionalities.



Figure 5.9 : Admin dashboard interface of Automated Resume Analysis and Skill Suggesting Website Using NLP

Resume Data and Score – Admin View:

In this screen, the admin selects “View Resume Score” to see details of uploaded resumes. The system displays the resume file name, extracted JSON data (like name, email, skills), and the calculated match score.

[Post Jobs](#) [View Resume Score](#) [View Feedback](#) [Logout](#)



Job ID	Username	Resume Name	Upload Date	Resume JSON Data	Resume Score
1	kumar	resume2.docx	2023-01-09	{name: CHRISTOPHER MORGAN Address, email: christoper.morgan@gmail.com, mobile_number: 666 8555, skills: [usability, mysql, project management, workflow, spanish, design, sql, javascript, programming, html5, php, email, css], college_name: None, degree: None, designation: None, experience: None, company_names: None, no_of_pages: None, total_experience: 0}	50.0

[Activate W](#)
[Go to Settings](#)

Figure 5.10 : Admin view of resume data and scoring in Automated Resume Analysis and Skill Suggesting Website Using NLP

VALIDATION

6.VALIDATION

The validation of this project primarily relies on extensive testing and well-defined test cases to ensure the accuracy and efficiency of the resume screening system. The testing process involves multiple stages, including dataset validation, model performance evaluation, and real-world testing. By implementing a structured validation approach, we ensure that the system consistently delivers high accuracy in resume-job matching while minimizing false positives and false negatives.

6.1 INTRODUCTION

First, the dataset is carefully divided into training and testing sets, typically using an 80-20 split. The training set is used to train the deep learning and NLP models, while the testing set is utilized to evaluate their generalization ability. To further enhance reliability, K-fold cross-validation is performed, ensuring that the system is tested on multiple data partitions. This method prevents overfitting and ensures that the model can generalize well to unseen resumes.

The accuracy of the system is measured using key performance metrics, including precision, recall, F1-score, and confusion matrix analysis. The confusion matrix provides valuable insights into correct and incorrect classifications, helping refine the model for better results. Additionally, the Hybrid Deep Learning Framework (CNN + BiLSTM) is compared against traditional machine learning models like SVM and Random Forest, demonstrating that the proposed approach achieves superior accuracy in resume screening.

Finally, real-world deployment testing is conducted to simulate live resume screening scenarios, ensuring that the system performs well on unseen resumes and job descriptions. Continuous improvements are made based on test results, allowing the model to remain effective in automated resume analysis and skill suggestion. This structured validation process ensures that the proposed system is reliable, scalable, and capable of maintaining high screening accuracy in real-time applications.

6.2 TEST CASES

TABLE 6.2.1: UPLOADING DATASET

Test Case ID	Test Case Name	Purpose	Test Case	Output
1	User uploads Dataset.	Use it for Resume Screening and Analysis.	The user uploads the Resume Dataset, on which the candidate information is extracted.	Dataset successfully loaded.

TABLE 6.2.2: CLASSIFICATION

Test Case ID	Test Case Name	Purpose	Input	Output
1	Classification test 1	To check if the classifier performs its task.	Resume file with relevant job skills is selected.	Resume classified as Relevant Candidate
2	Classification test 2	To check if the classifier performs its task.	Resume file with mismatched job skills is selected.	Resume classified as Not Relevant Candidate.

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

This research presents a robust hybrid machine learning framework for automated resume screening and candidate ranking, integrating TF-IDF vectorization (text analysis), cosine similarity (matching optimization), and Latent Dirichlet Allocation (LDA) (topic modeling), achieving high efficiency in candidate evaluation. The system leverages natural language processing (NLP) and machine learning techniques to extract relevant keywords from job descriptions and resumes, ensuring precise ranking and reducing human bias.

By optimizing TF-IDF for text relevance, the framework enables real-time deployment using the Flask framework, allowing seamless integration with web-based interfaces. The incorporation of LDA topic modeling enhances the system's ability to assess candidate suitability based on contextual relevance, improving recruitment efficiency. This framework advances AI-driven recruitment by harmonizing keyword extraction, semantic analysis, and candidate ranking algorithms, demonstrating practical viability in streamlining hiring processes. Looking ahead, future developments will focus on integrating deep learning models, refining natural language understanding (NLU) capabilities, and enhancing sentiment analysis, ensuring accurate, fair, and data-driven hiring decisions.

7.2 FUTURE ASPECTS

The proposed Automated Resume Screening and Candidate Ranking System has significantly improved the efficiency and accuracy of recruitment processes. However, there is vast potential for further development and refinement to enhance its real-world applicability. Future advancements in AI-driven resume analysis, deep learning techniques, scalability, and ethical hiring practices can make the system even more robust and effective.

These are some future aspects:

- Integration of Deep Learning Models
- Real-Time Candidate Evaluation
- Enhanced Natural Language Understanding (NLU) Capabilities
- Cross-Industry Adaptation for Diverse Job Roles
- Privacy and Ethical Considerations in Automated Hiring
- Continuous Learning and Model Improvement
- Hybrid AI-Human Decision-Making System

BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

- [1]. Singh, A. K., & Shukla, P., 2020. Automated resume screening and evaluation using machine learning techniques, *Journal of Intelligent & Fuzzy Systems*, 39(4), 5947-5960.
- [2]. Oh, J., & Lee, S., 2019. A study on the extraction of competencies from job postings and their correlation with resumes using NLP techniques, *Expert Systems with Applications*, 115, 475-486.
- [3]. Xu, C., Lu, J., Liu, J., & Wei, X., 2021. Resume screening using deep learning and natural language processing, *Knowledge-Based Systems*, 215, 106864.
- [4]. Bhowmik, R., Garg, N., & Gupta, A., 2021. Resume Screening Using Semantic Similarity and Clustering Algorithms, *Proceedings of the 2021 3rd International Conference on Communication, Devices and Computing*.
- [5]. Elakkiya, R., & Muthurajkumar, S., 2021. Automated Resume Screening System using Semantic Similarity, *2021 International Conference on Computing, Electronics & Communications Engineering (ICCECE)*.
- [6]. Garg, N., Bhowmik, R., & Gupta, A., 2021. Automated Resume Screening Using Semantic Similarity-Based Sentence Embeddings, *2021 International Conference on Smart Electronics and Communication (ICOSEC)*.
- [7]. Huang, S., Li, W., Wang, L., & Huang, H., 2021. Resume Screening and Ranking with Natural Language Processing Techniques, *Applied Sciences*, 11(5), 2095.
- [8]. Kang, Y., & Lee, J., 2020. Resume Analysis for Job Matchmaking Using Word Embedding and Ranking Algorithm, *Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication*.
- [9]. Li, X., & Shen, X., 2021. Resume Ranking and Classification Based on SBERT, *2021 International Conference on Computer, Information and Telecommunication Systems (CITS)*.
- [10]. Liu, J., Zhang, R., Yang, W., & Guan, R., 2021. A Semantic Similarity-Based Resume Screening System, *Journal of Intelligent & Fuzzy Systems*, 40(1), 787-797.

8.2 GITHUB LINK

<https://github.com/geetha0803/Automated-Resume-Scoring-selection-App>