BATCH NO:MAI021

SMART RECRUTING PLATFORM USING MACHINE LEARNING

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

K.LAKSHMI PRIYA	(21UECT0019)	**(VTU 20407)**
K.VINAY KRISHNA	(21UECT0070)	**(VTU 20388)**
K.ROHITH CHOWDARY	(21UECT0061)	**(VTU 19134)**

Under the guidance of
Dr.D.SUNDARANARAYANA,M Tech.Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2025

BATCH NO:MAI021

SMART RECRUTING PLATFORM USING MACHINE LEARNING

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

K.LAKSHMI PRIYA	(21UECT0019)	**(VTU 20407)**
K.VINAY KRISHNA	(21UECT0070)	**(VTU 20388)**
K.ROHITH CHOWDARY	(21UECT0061)	**(VTU 19134)**

Under the guidance of
Dr.D.SUNDARANARAYANA,M Tech.Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2025

CERTIFICATE

It is certified that the work contained in the project report titled ”SMART RECRUTING PLATFORM USING MACHINE LEARNING” by ”K.LAKSHMI PRIYA (21UECT0019),K.VINAY KRISHNA (21UECT0070), K.ROHITH CHOWDARY (21UECT0061)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Dr.D.Sundaranarayana
Associate Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

Signature of Head/Assistant Head of the Department
Dr. N. Vijayaraj/Dr. M. S. Murali dhar
Professor & Head/ Assoc. Professor &Assistant Head
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

Signature of the Dean
Dr. S P. Chokkalingam
Professor & Dean
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

K.LAKSHMI PRIYA
Date: / /

K.VINAY KRISHNA
Date: / /

K.ROHITH CHOWDARY
Date: / /

APPROVAL SHEET

This project report entitled (**SMART RECRUTING PLATFORM USING MACHINE LEARNING**) by (K.LAKSHMI PRIYA (21UECT0019), (K.VINAY KRISHNA (21UECT0070), (K.ROHITH CHOWDARY (21UECT0061) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Supervisor**

Dr.D.SUNDARANARAYANA,M Tech.Ph.D
ASSOCIATE PROFESSOR.,

Date: / /
Place: AVADI

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Honorable Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (Electrical), B.E. (Mechanical), M.S (Automobile), D.Sc., and Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.,Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We express our sincere thanks to our respected Chairperson and Managing Trustee **Mrs. RANGARAJAN MAHALAKSHMI KISHORE,B.E., Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We are very much grateful to our beloved **Vice Chancellor Prof. Dr.RAJAT GUPTA,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean , School of Computing,Dr. S P. CHOKKALINGAM, M.Tech., Ph.D., & Professor & Associate Dean , School of Computing, Dr. V. DHILIP KUMAR,M.E.,Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Professor & Head, Department of Computer Science & Engineering, Dr. N. VIJAYARAJ, M.E., Ph.D.,** and **Associate Professor & Assistant Head, Department of Computer Science & Engineering, Dr. M. S. MURALI DHAR, M.E., Ph.D.,**for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Dr.D.SUNDARANARAYANA,M Tech.Ph.D.,** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Dr. SADISH SENDIL MURUGARAJ,Professor, Dr.S.RAJAPRAKASH, M.E,Ph.D., Mr. V. ASHOK KUMAR, B.E,M.Tech.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

K.LAKSHMI PRIYA	**(21UECT0019)**
K.VINAY KRISHNA	**(21UECT0070)**
K.ROHITH CHOWDARY	**(21UECT0061)**

ABSTRACT

The emergence of machine learning (ML) and natural language processing (NLP) has significantly transformed the recruitment landscape, offering innovative ways to streamline the hiring process. This project introduces a Smart Recruiting Platform that utilizes state-of-the-art ML techniques, including BERT (Bidirectional Encoder Representations from Transformers) and Support Vector Machines (SVM), to enhance the recruitment process. By automating candidate screening, this platform aims to improve efficiency, reduce human bias, and provide a more accurate analysis of applicants' qualifications and suitability for roles. The platform analyzes resumes and job descriptions using advanced NLP models to assess key competencies, skills, and experience, ensuring a better match between candidates and employers.

The core of the platform is built on NLP-based pre-processing techniques, with BERT as the primary model for understanding the contextual meaning of text data in resumes. BERT's powerful transformer architecture allows the platform to capture the nuances of candidate profiles, such as qualifications, work history, and soft skills. For the final decision-making process, an SVM classifier is employed to categorize candidates based on their relevance to specific job descriptions. The system evaluates various features, such as keyword matches, job experience relevance, and skillsets, and assigns scores to applicants, which helps recruiters make faster and more informed decisions.

By integrating NLP with machine learning algorithms like BERT and SVM, this platform offers a more intelligent and scalable solution for the modern hiring process. It also aims to reduce the burden on HR professionals, providing them with automated insights to help prioritize candidates and make data-driven decisions. The end goal is to provide an innovative solution that not only simplifies recruitment but also promotes fairness and precision in selecting candidates who best align with organizational needs.

Keywords: Machine Learning, Natural Language Processing(NLP), Bidirectional Encoder Representations from Transformers(BERT), Support Vector Machines (SVM), Candidate Screening.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ABBREVATION	**DEFINITION**
ML	Machine Learning
NLP	Natural Language Processing
SVM	Support Vector Machine
BERT	Bidirectional Encoder Representations from Transformer
HR	Hiring Manager
JD	Job Description

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

A smart recruiting platform uses machine learning algorithms to automate and optimize hiring processes. Natural Language Processing (NLP) extracts insights from resumes and job descriptions, while classification algorithms like logistic regression or SVM predict candidate suitability. Recommendation systems and clustering algorithms group and match candidates to jobs based on skills and experience. Deep learning models and ensemble methods further enhance decision-making, ensuring accurate, data-driven hiring outcomes.

In today's rapidly evolving tech-driven world, traditional resume submission methods have given way to e resumes viewed online. However, the current system often requires candidates to manually input all resume details, leading to inefficiencies and mismatches between job requirements and candidate skills. With thousands of resumes per job posting, manual analysis becomes impractical for recruiters, leading to dissatisfaction among candidates and challenges in finding the right fit. To address this, innovative recruitment platforms utilize machine learning and Natural Language Processing (NLP) techniques for efficient resume parsing and matching. By categorizing resumes and job postings, these platforms reduce time complexities and improve accuracy.

However, despite their effectiveness, time consumption remains a concern. One proposed solution involves segmenting resumes based on sections and employing NLP for data extraction, enhancing efficiency. Resume Parser and Analyzer tools further streamline the process by structuring unstructured resumes, extracting essential fields, and suggesting improvements. This not only saves recruiters time but also provides applicants with insights into their resume's standing and areas for enhancement. Moreover , by allowing only the recruiter to access matched results, confidentiality is maintained, and the most qualified candidates are efficiently identified. This intelligent-based approach aims to optimize the recruitment process, benefiting both recruiters and job seekers alike.

1.2 Background

A smart recruiting platform powered by machine learning leverages advanced algorithms like Natural Language Processing (NLP) and BERT (Bidirectional Encoder Representations from Transformers) to streamline and enhance the hiring process. NLP enables the system to understand and process human language, allowing it to analyze resumes, job descriptions, and candidate interactions effectively. BERT, a state-of-the-art language model, further improves this by capturing context and meaning in text, enabling accurate candidate-job matching, sentiment analysis, and automated screening. Together, these technologies make recruitment faster, more efficient, and data-driven.

1.3 Objective

The objective of this project is to develop an intelligent recruiting platform that leverages machine learning, NLP (Natural Language Processing), and BERT (Bidirectional Encoder Representations from Transformers) to streamline and enhance the recruitment process. Traditional recruitment methods are often time-consuming, biased, and inefficient in matching candidates with job roles. By implementing state-of-the-art NLP techniques, this system aims to automate resume screening, analyze job descriptions, and match candidates based on skills, experience, and job requirements. Using BERT, the platform can deeply understand and extract meaningful insights from resumes and job postings. This not only reduces the manual workload of recruiters but also ensures that candidates receive fair and data-driven evaluations.

Additionally, the smart recruiting platform will incorporate sentiment analysis and entity recognition to further refine the recruitment process. It will identify soft skills, predict candidate suitability, and even suggest job recommendations for applicants based on their profiles. The system will be trained on large-scale recruitment datasets to improve its predictive accuracy and adaptability across different industries. Furthermore, bias mitigation techniques will be implemented to ensure fair and inclusive hiring practices. The final outcome is an AI-driven recruitment assistant that optimizes candidate-job matching, enhances efficiency, and provides data-driven hiring insights, making the hiring process more transparent, effective, and scalable for both recruiters and job seekers.

1.4 Problem Statement

Recruiting the right candidates for job roles is a critical challenge faced by organizations. Traditional hiring processes often rely on manual resume screening, which is time-consuming, prone to biases, and inefficient in handling large volumes of applications. Many qualified candidates may be overlooked due to keyword-based filtering, while recruiters struggle to match job descriptions with the most suitable applicants. Additionally, job seekers often face difficulties in tailoring their resumes to specific roles, leading to mismatches and lost opportunities. The lack of an intelligent and automated system results in inefficiencies in the recruitment process, affecting both employers and job seekers.

To address these challenges, a smart recruiting platform leveraging Natural Language Processing (NLP) and Bidirectional Encoder Representations from Transformers (BERT) can enhance resume screening and job matching. Using machine learning models, the platform can analyze resumes, extract key skills, and match candidates to job descriptions more accurately than traditional methods. BERT, a powerful NLP algorithm, allows for contextual understanding of job descriptions and resumes, enabling precise candidate-job fit. The system can also provide job seekers with resume improvement suggestions and recruiters with ranked candidate recommendations. By integrating AI-driven insights, this platform aims to streamline recruitment, reduce hiring time, and improve the overall effectiveness of talent acquisition.

Chapter 2

LITERATURE REVIEW

Saha et al.[1] Automated Resume Screening Using NLP and Machine Learning developed an intelligent system that automates resume screening using NLP and machine learning algorithms. The model parses resumes to extract relevant candidate information like skills, experience, and education, and then uses classifiers such as Random Forest and SVM to evaluate candidate-job fit. They addressed the challenge of unstructured resume data using text vectorization and keyword matching. A key takeaway from their study is the significant reduction in recruiter effort and screening time, without compromising on match quality. Their experiments showed improved accuracy and recall over manual methods. This paper supports the integration of intelligent resume parsing and job recommendation modules in smart recruitment platforms.

Zhang et al.[2] Deep Learning for Resume-Job Matching proposed a deep learning-based ranking framework for matching resumes to job descriptions. They employed neural networks with embedding layers to understand the semantic similarity between job requirements and candidate profiles. Their system was trained on a large corpus of real recruitment data, and it outperformed baseline models in ranking relevance and match precision. Unlike traditional models, their approach considered context and phrase-level embeddings, improving recommendations for roles that require nuanced skill matches. This research is useful for building a smart recruitment platform where deep learning enhances matching accuracy, especially in diverse and competitive applicant pools.

Li et al.[3] Deep Learning for Intelligent Candidate Ranking propose a deep learning-based model for intelligent candidate ranking by analyzing resume-job description similarity. They utilize convolutional neural networks (CNNs) with word embeddings such as Word2Vec and GloVe to capture semantic meaning in resumes and job postings. The model calculates a compatibility score and ranks candidates based on how well they match the job. Their dataset includes real-world resume-job pairs from a job portal, and the model is evaluated using precision, recall, and NDCG metrics. It significantly outperforms traditional matching methods. The paper's key contribution lies in demonstrating that deep learning can learn contextual matching features without manual rule-based systems. It supports the integration of AI in recruitment systems for more personalized and accurate candidate recommendations, which is highly relevant for smart recruiting platforms focused on scalability and precision.

Patel et al.[4] Machine Learning in Recruitment Analytics conducted a study on applying various machine learning algorithms such as Decision Trees, Naïve Bayes, and Logistic Regression to recruitment data for predicting hiring outcomes. They highlighted the importance of selecting relevant features like experience, skills, and education level to train robust models. Their findings showed that Logistic Regression performed consistently well in predicting candidate success based on historical data. This research contributes to smart recruitment by showing how predictive analytics can guide recruiters toward data-driven decision-making and improve selection accuracy.

Zhang et al.[5] Resume Quality Assessment Using NLP Techniques to analyze the structure and content of resumes. The model evaluates grammar, keyword relevance, and layout to score resumes on quality and clarity. They use TF-IDF, part-of-speech tagging, and Named Entity Recognition (NER) for feature extraction. The paper introduces a multi-factor scoring approach combining content quality and industry relevance. The goal is to help job seekers improve their resumes and assist recruiters in shortlisting high-quality candidates efficiently. The model was tested on a dataset of resumes across various industries, with expert-labeled ground truths for validation. This study is significant for smart recruiting platforms that offer candidate feedback or resume optimization suggestions. For student projects, it provides inspiration for implementing candidate evaluation modules based on textual quality analysis.

Kumar et al.[6] A BERT-Based Job Recommendation System that uses BERT for encoding both candidate resumes and job descriptions. By leveraging the contextual understanding provided by BERT, their model achieved superior semantic matching compared to TF-IDF or traditional embedding approaches. They also introduced fine-tuning on domain-specific recruitment data to improve result quality. This study showcases how transformer-based architectures can significantly boost the performance of matching engines in recruitment platforms, especially for niche or technical roles where traditional keyword-based systems fail.

Deshmukh et al.[7] AI-Powered Screening to Reduce Human Bias that eliminates human subjectivity by relying on ML-driven ranking. Their model assigns unbiased scores based on predefined job criteria and resume data, removing demographic indicators during the evaluation process. They implemented this in a prototype that outperformed human recruiters in terms of speed and consistency. The system's transparency and auditability features were also highlighted as key strengths. Their findings reinforce the value of automation in building ethical and inclusive recruitment ecosystems.

Singh et al[8] Machine Learning for Job Recommendation Systems that uses collaborative filtering and content-based filtering techniques. The model incorporates both user preferences and resume content to suggest relevant job postings. They also include a learning-to-rank algorithm to order recommendations by likelihood of candidate interest and employer fit. The system is trained on real-world job portal data and evaluated using click-through rates and user feedback. The research highlights the importance of hybrid recommender systems in recruitment applications. This work is valuable for smart recruiting platforms aiming to personalize job suggestions for users. It demonstrates how machine learning can bridge the gap between candidate skills and job requirements, improving overall candidate engagement and platform usability.

Li et al.[9] Bias Mitigation in AI-Based Hiring Systems addressed the critical issue of bias in machine learning-based recruitment systems. They reviewed multiple bias mitigation techniques such as adversarial debiasing, reweighting, and fairness constraints. The study also included experimental evaluations of hiring models to assess performance across different demographic groups. Their work is essential in ensuring that smart recruiting platforms are ethical and fair, especially when using automated decision-making. The authors conclude that fairness-aware ML models can improve trust and reduce legal risks, making bias detection and mitigation a foundational aspect of recruitment AI design.

Nicholas Wilson et al.[10]Recruitment fraud, including fake resumes and exaggerated credentials, is a rising concern for employers. This study introduced a fraud detection system for recruitment using anomaly detection techniques such as Isolation Forest and Autoencoders. The model flagged suspicious resumes based on inconsistencies in work history, skill exaggeration, and forged credentials. The results demonstrated that AI-powered fraud detection reduces recruitment risks and enhances hiring security.

2.1 Existing System

The existing system for smart recruiting or hiring primarily relies on traditional machine learning algorithms such as Decision Trees, Logistic Regression, or Naive Bayes, in combination with basic Natural Language Processing (NLP) techniques. These systems often focus on keyword-based matching, where resumes are scanned for specific terms related to job descriptions. While this approach offers some level of automation in screening candidates, it lacks depth in understanding the context of the information provided. For example, if a candidate uses a synonym or alternative phrasing not recognized by the model, they may be inaccurately ranked or excluded. Furthermore, these models require manual feature extraction and engineering, where human input is necessary to determine which parts of the resume should be considered important making the process time-consuming and less scalable.

Additionally, traditional machine learning models tend to under perform when dealing with large volumes of unstructured textual data, which is typical in resumes and job descriptions. These systems often struggle with nuances such as tone, sentence structure, and contextual meaning. As a result, the accuracy of candidate-job matching is generally limited, leading to a higher possibility of missing qualified candidates or shortlisting irrelevant ones. Moreover, these systems offer little support for learning from recruiter feedback or adapting to evolving job requirements over time. While they may be cost-effective and easy to implement, the existing systems are not well-suited for modern, high-volume, or context-driven hiring needs. This has created a demand for more intelligent, context-aware solutions like those using deep learning and BERT-based NLP models, as introduced in the proposed system.

2.2 Related Work

Over the past few years, several studies have explored the application of machine learning and natural language processing in the recruitment and hiring domain. Traditional approaches have relied on rule-based systems and simple keyword matching algorithms, which are limited in handling unstructured and context-rich data such as resumes and job descriptions. Researchers have experimented with various machine learning algorithms like Decision Trees, Support Vector Machines (SVM), and Naive Bayes classifiers to automate resume screening. While these techniques improved efficiency over manual screening, they often lacked accuracy and contextual understanding, leading to irrelevant matches and biased outcomes. Additionally, some systems utilized basic NLP techniques such as tokenization, part-of-speech tagging, and named entity recognition to extract structured information from resumes, yet they struggled to capture deeper semantics and candidate intent.

The advancements in deep learning and transformer-based models, studies has focused on integrating BERT (Bidirectional Encoder Representations from Transformers) and other contextual NLP models into hiring platforms. BERT allows for a better understanding of the relationship between words in a sentence, enabling more accurate candidate-job matching even when terminology or phrasing differs. Several academic and industry projects have demonstrated that using BERT significantly enhances classification performance, particularly when dealing with large, unstructured datasets.

Moreover, recent research has introduced hybrid models that combine BERT embeddings with traditional classifiers like Random Forests or neural networks to further boost performance. These works form the foundation for the proposed system, which aims to leverage BERT's deep contextual understanding in combination with machine learning techniques to create a more accurate, fair, and intelligent recruiting platform.

2.3 Research Gap

Table 2.1: Summary of research gaps identified in smartrecruiting systems.

Author(s)	**Title**	**Research Gap Identified**
Malhotra & Rathi (2020)	AI in Recruitment: A Review	Lacks contextual understanding; heavy reliance on keyword matching.
Chamorro-Premuzic et al. (2016)	The Talent Delusion	Limited empirical evidence supporting AI-based hiring effectiveness.
Jain et al. (2021)	NLP Techniques for Resume Parsing	NLP models used are outdated; minimal use of transformer-based models like BERT.
Upadhyay & Khandelwal (2018)	Applying AI to Recruitment	No clear mechanisms for bias detection and mitigation.
Chatterjee et al. (2022)	Enhancing Recruitment with BERT	BERT not fine-tuned for recruitment-specific tasks; limited interpretability.
Liu et al. (2019)	BERT: Pre-training of Deep Bidirectional Transformers for Language	General NLP model; requires domain-specific adaptation for recruiting scenarios.
Raghavan et al. (2020)	Mitigating Algorithmic Bias in Hiring Systems	Ethical concerns largely unaddressed; lack of fairness and accountability in ML models used in hiring.
Zhao & Zhou (2021)	Resume-Job Matching with Deep Learning	Models evaluated on small datasets; scalability and real-world deployment challenges not explored.
Singh & Sharma (2023)	Explainable AI in HR	Lack of transparency and explainability in ML-based candidate screening systems.
Nguyen et al. (2022)	Smart Hiring Systems: A Survey	Few systems integrate multimodal data (e.g., resumes, interviews, assessments); over-reliance on text alone.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing smart recruiting platforms utilize Natural Language Processing (NLP) and Support Vector Machine (SVM) algorithms to automate and optimize hiring processes. NLP is applied to extract and analyze keywords, skills, and experience from resumes and job descriptions, improving the accuracy of candidate-job matching. Additionally, resume parsing, sentiment analysis, and entity recognition help filter and rank applicants efficiently.

To enhance recruitment accuracy, SVM classifiers are used for candidate classification and ranking based on job requirements. The system assigns weights to various attributes such as experience, skills, and education, allowing for more precise and data-driven hiring decisions. While the existing system reduces manual workload, it still faces challenges in understanding nuanced soft skills and cultural fit, which are essential for long-term employee success. Further improvements can enhance fairness and efficiency in recruitment.

Disadvantages Of Existing System:

- **Lack of Contextual Understanding:**NLP-based keyword matching fails to understand the meaning behind words, leading to incorrect shortlisting.
- **Limited Soft Skill Assessment:**SVM classifiers are not effective in evaluating soft skills, emotional intelligence, or cultural fit, which are essential for hiring success.
- **Challenges in Resume Formatting:**Different resume structures can cause errors in information extraction, leading to inaccurate candidate ranking.
- **Difficulty in Handling Complex Job Requirements:**The system struggles with multi-faceted job roles that require diverse skills beyond simple keyword matching.

3.2 Proposed System

A Smart Recruiting Platform using Machine Learning, NLP, SVM, and BERT aims to streamline and enhance the hiring process by leveraging advanced AI techniques. The system utilizes Natural Language Processing (NLP) to analyze resumes, job descriptions, and candidate profiles, ensuring better matching between candidates and job roles. Support Vector Machine (SVM) is employed for classification tasks, helping filter and rank candidates based on their qualifications and experience. Additionally, Bidirectional Encoder Representations from Transformers (BERT) enhances the system by improving the semantic understanding of resumes and job descriptions, ensuring that the best candidates are shortlisted. The platform will feature automated resume screening, ranking, and recommendation functionalities, reducing the manual effort required in traditional hiring processes.

The proposed system offers several advantages over conventional hiring methods. Firstly, it eliminates bias by using data-driven candidate selection, ensuring fair recruitment. Secondly, it saves time and resources by automating resume screening and candidate ranking, reducing the workload for HR teams. Thirdly, improved accuracy in candidate-job matching ensures that the right candidates are shortlisted, leading to better hiring decisions. Finally, the use of BERT and NLP enables the system to understand contextual meaning, making it superior to keyword-based searches.

Advantages Of Proposed System:

- **Automated Efficient Screening:**The system automatically processes and ranks resumes using NLP and SVM, significantly reducing the time and effort required by HR teams.
- **Improved Accuracy in Candidate Matching:**With BERT's deep semantic understanding, the platform ensures precise job-candidate alignment, minimizing mismatches and improving hiring quality.
- **Elimination of Bias in Recruitment:**By relying on AI-driven decisions, the system promotes fair hiring, reducing unconscious biases that may affect traditional recruitment processes.
- **Scalability for High-Volume Hiring:**The system can efficiently handle thousands of applications, making it ideal for companies with large-scale recruitment needs.

3.3 Feasibility Study

Developing a smart recruiting platform using machine learning is a feasible and practical project, considering the advancements in Natural Language Processing (NLP), Support Vector Machine (SVM), and BERT. The availability of open-source libraries such as TensorFlow, Scikit-learn, and Hugging Face allows students to implement powerful AI-driven resume screening and candidate matching systems with minimal cost. Cloud-based platforms like Google Cloud and AWS provide affordable computing resources for model training and deployment, making the project economically viable. From an operational perspective, the platform can be designed as a web-based system with an intuitive interface, ensuring ease of use for both recruiters and job seekers. The integration of machine learning algorithms enhances efficiency by automating resume screening, ranking candidates, and reducing recruitment time. Given that traditional hiring processes are often time-consuming and biased, this project holds significant practical value. The combination of accuracy, efficiency, and fairness makes the system a viable solution for modern recruitment challenges, proving its feasibility as a student project.

3.3.1 Economic Feasibility

The economic feasibility of the Smart Recruiting Platform using NLP, SVM, and BERT algorithms for this project is highly promising, given the availability of cost-effective resources and tools. Natural Language Processing (NLP) techniques, combined with the Support Vector Machine (SVM) and BERT, provide a powerful framework for automating resume screening and candidate matching. These machine learning algorithms can be implemented using open-source libraries such as TensorFlow, Scikit-learn and BERT, which are free to use, eliminating the need for costly proprietary software. Additionally, cloud platforms like Google Colab, AWS Free Tier, and Microsoft Azure for Students provide access to the computing power required for training and deploying machine learning models without the need for expensive infrastructure. Developing the platform with open-source web frameworks will further reduces the costs. By automating time-consuming processes like resume screening and candidate ranking, the platform can significantly reduce operational costs for companies, making it a highly cost-effective solution. The ability to improve hiring efficiency and reduce the manual workload of recruiters ensures that the project remains economically feasible while offering substantial long-term benefits.

3.3.2 Technical Feasibility

The technical feasibility of using NLP, SVM, and BERT algorithms in the Smart Recruiting Platform is highly achievable due to their accessibility and effectiveness in handling recruitment tasks. NLP enables the system to process and analyze text data from resumes and job descriptions, extracting key information such as skills, qualifications, and experience. BERT enhances the platform's capability by understanding the contextual meaning of words in a sentence, enabling better candidate-job matching based on semantic analysis. Additionally, Support Vector Machine (SVM) can be used for classification tasks, effectively ranking candidates by evaluating their qualifications. These advanced algorithms are readily available through open-source libraries like Hugging Face's Transformers, TensorFlow, and Scikit-learn, making them easy to implement and reducing the need for expensive proprietary software. Given these resources, the integration of NLP, SVM, and BERT provides a technically feasible and efficient solution for automating the recruitment process.

3.3.3 Social Feasibility

The social feasibility of the Smart Recruiting Platform using Machine Learning, NLP, and BERT algorithms is highly positive, as it can significantly improve the recruitment process for both employers and job seekers. The platform promotes fairness and inclusivity by minimizing human bias in hiring decisions. By automating resume screening and candidate ranking, the system ensures that all applicants are evaluated based on their qualifications and skills rather than unconscious biases related to age, gender, or background. This makes the platform socially responsible, as it contributes to more equitable hiring practices. Moreover, job seekers benefit from faster responses and a more transparent process, reducing the frustration associated with traditional recruitment methods. The platform also supports a more diverse workforce by ensuring that candidates from various backgrounds are considered fairly. From an employer's perspective, using the system can enhance their corporate social responsibility (CSR) initiatives by promoting diversity and inclusion in the workplace. Overall, the social impact of the Smart Recruiting Platform is positive, fostering fairness, efficiency, and equal opportunity.

3.4 System Specification

Hardware specification:

System: Pentium i5 Processor.

Hard Disk: 500 GB

Monitor: 15” LED

Input Devices: Keyboard, Mouse

RAM: 6 GB

Software Specification:

Operating system: Windows 11

Coding Language: Python

Algorithm: Natural Language processing(NLP),Bidirectional Encoder Representations from Transformers(BERT),Support Vector Machine(SVM)

3.4.1 Tools and Technologies Used

3.4.2 Standards and Policies

Anaconda Prompt

Anaconda Prompt serves as a command-line interface for managing ML environments and packages used in the platform. It supports cross-platform development and offers access to IDEs and tools that aid in coding, model training, and deployment. The Python-based interface can also be used to build user interfaces for the platform.

Standard Used: ISO/IEC 27001

Jupyter

Jupyter is an open-source web application used within the platform for developing, testing, and presenting ML models. It supports live code, equations, data visualization, and descriptive text, which facilitates model development, data analysis, and collaborative documentation.

Standard Used: ISO/IEC 27001

Chapter 4

SYSTEM DESIGN AND METHODOLOGY

4.1 System Architecture

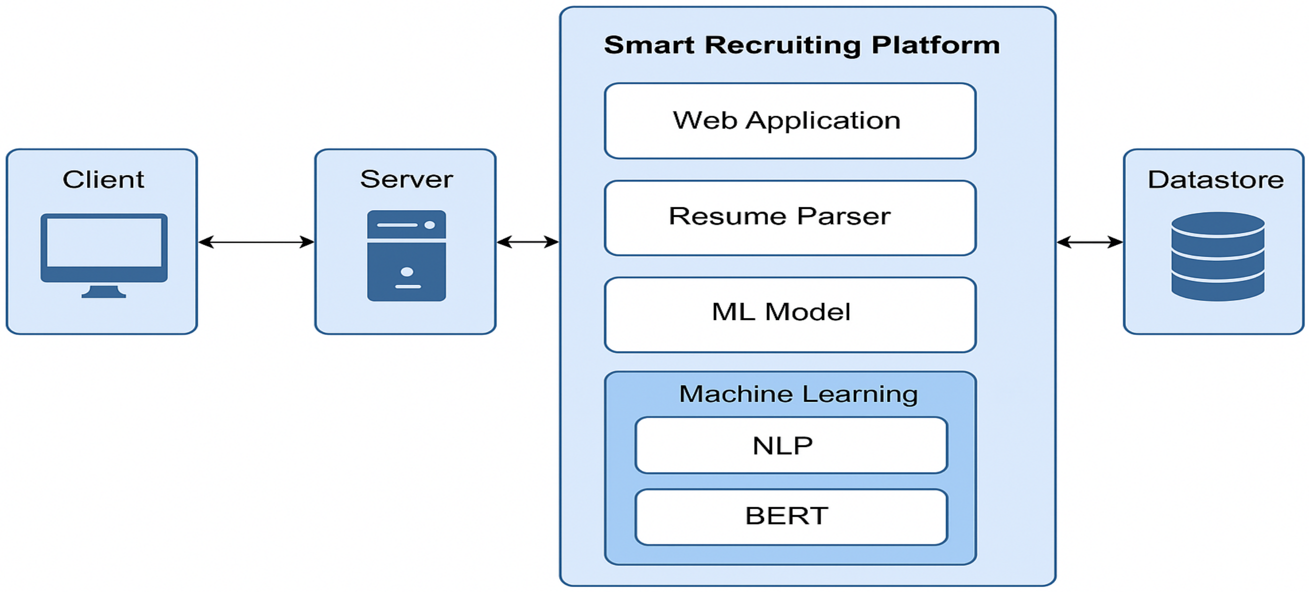


Figure 4.1: Architecture Diagram

Description:

Figure 4.1 Shows the Architecture Diagram of the system begins with the client interface, typically a browser or mobile device, allowing users to interact with the platform. Requests from the client are processed by the server, which hosts the core components of the recruiting platform. The Web Application provides the front-end interface for job seekers and recruiters, while the Resume Parser extracts structured information from unstructured resume data. This data is fed into a Machine Learning model that evaluates candidates based on job requirements. Natural Language Processing (NLP) techniques are used to understand and analyze textual content, while the BERT model enhances semantic understanding for more accurate candidate-job matching. All processed data and candidate records are stored and retrieved from a

centralized Datastore.

4.2 Design Phase

4.2.1 Data Flow Diagram

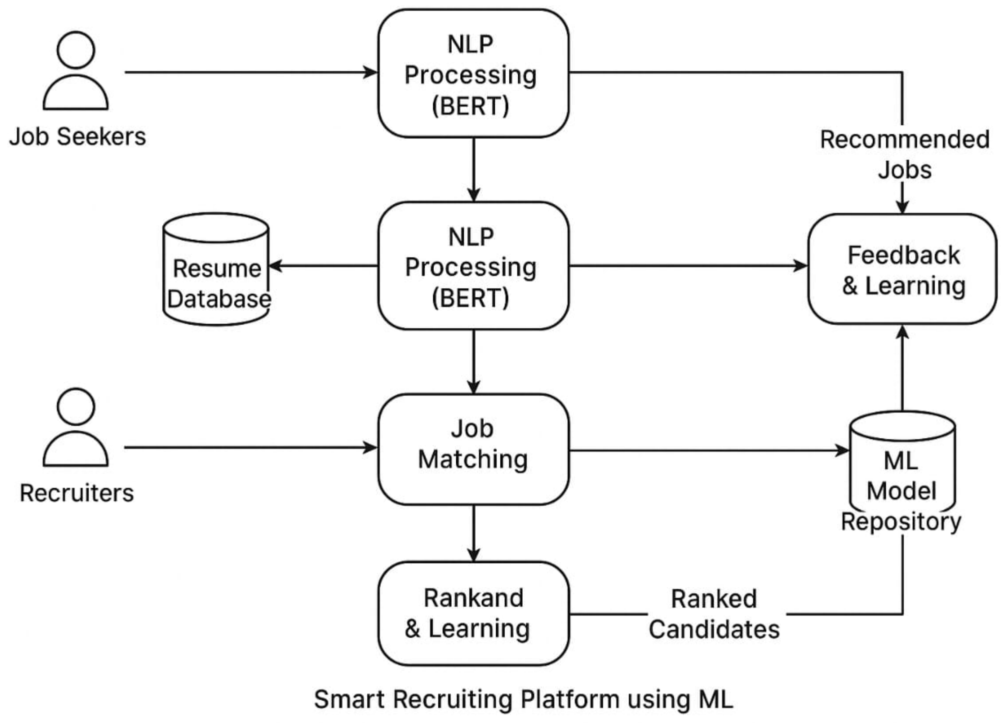


Figure 4.2: Data Flow Diagram

Description:

Figure 4.2 Shows the Data Flow Diagram of smart recruiting platform utilizes machine learning (ML) and natural language processing (NLP) to match job seekers with recruiters efficiently. Job seekers provide input that undergoes NLP processing using BERT to analyze resumes and job preferences. The processed data is stored in a resume database and further refined using another NLP model. Recruiters input job descriptions, which are matched against candidate profiles using a job-matching module. Ranked candidates are provided through a ranking and learning system, leveraging an ML model repository. The system continuously improves through feedback and learning, enhancing job recommendations and candidate ranking over time.

4.2.2 Use Case Diagram

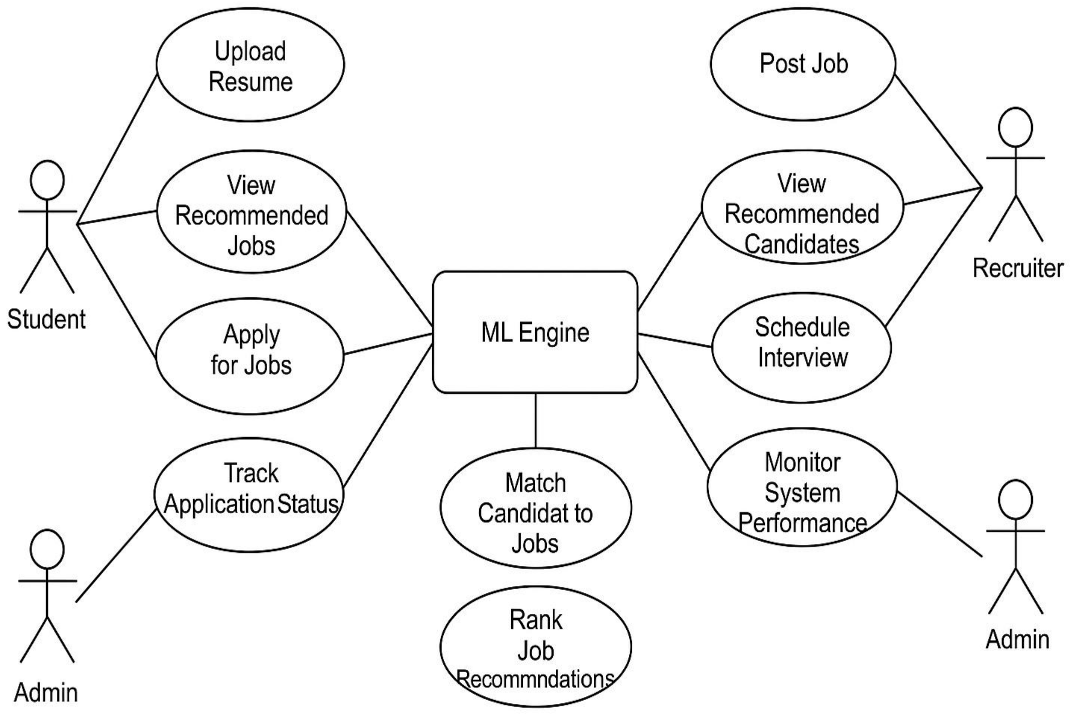


Figure 4.3: Use Case Diagram

Description:

Figure 4.3 Shows the Use Case Diagram of Smart Recruiting Platform integrates ML, NLP, and BERT to streamline the hiring process. Job seekers upload their resumes, which are processed to generate job recommendations based on their skills and experience. They can view recommended jobs and receive feedback for better opportunities. Recruiters post job openings and receive ranked candidate recommendations, helping them make informed hiring decisions. Feedback from both job seekers and recruiters continuously improves the system's matching accuracy. This intelligent platform enhances recruitment efficiency by leveraging AI-driven analysis, ensuring optimal matches between candidates and job opportunities while refining the hiring process through feedback and learning.

4.2.3 Class Diagram

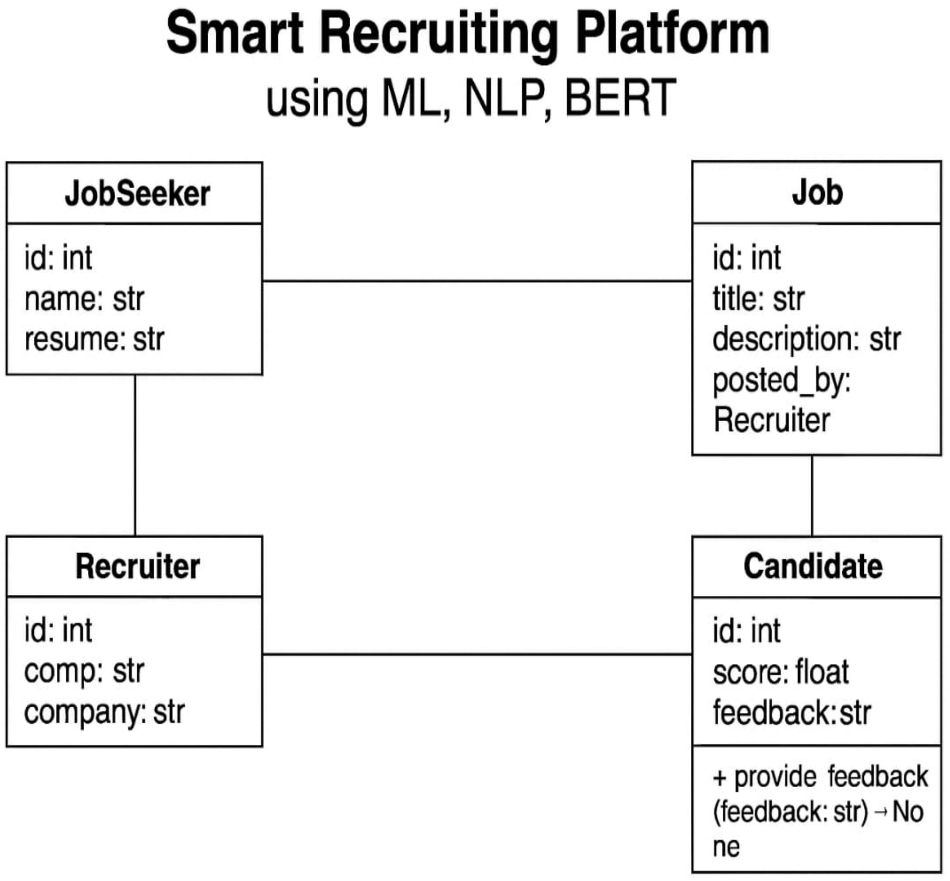


Figure 4.4: Class Diagram

Description

Figure 4.4 Shows the class diagram represents the structure of the Smart Recruiting Platform, utilizing ML, NLP, and BERT. The JobSeeker class contains attributes like ID, name, and resume, linking to job applications. Recruiter has an ID and company details, responsible for posting jobs under the Job class, which includes job ID, title, description, and recruiter information. Candidates are evaluated and stored in the Candidate class with attributes like ID, score, and feedback. Recruiters provide feedback to refine rankings. This structured system enables automated job matching, ensuring optimal candidate-job pairing while continuously improving through feedback and ML-based learning mechanisms.

4.2.4 Sequence Diagram

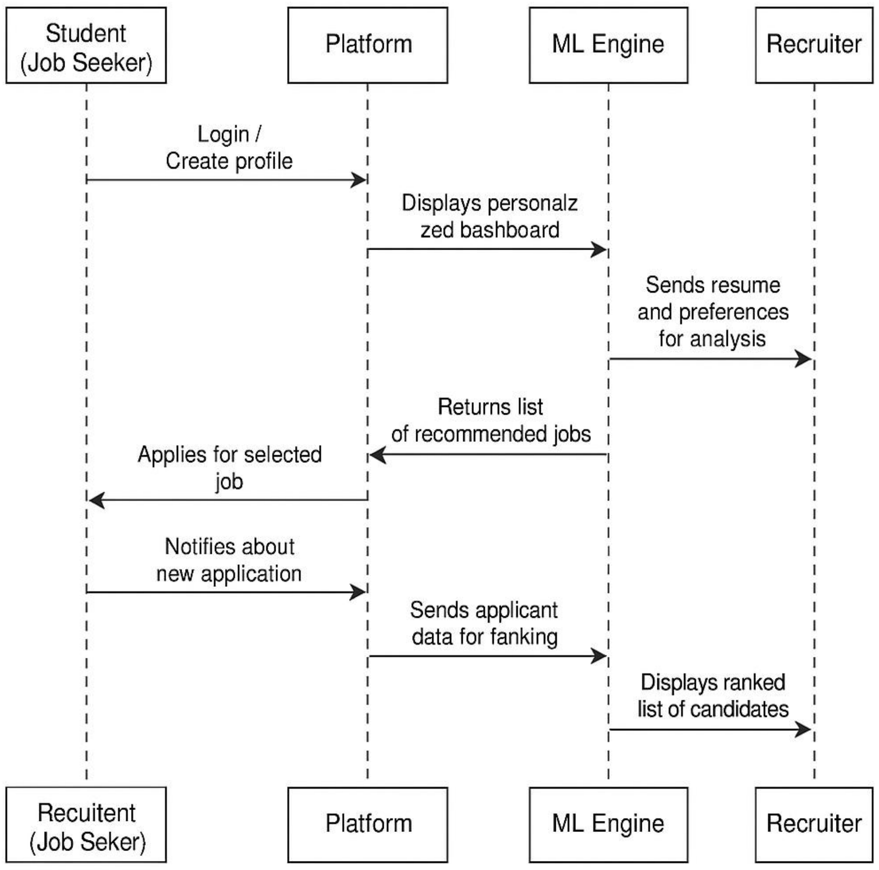

Student (Job Seeker)
Platform
ML Engine
Recruiter
Login / Create profile
Displays personalz zed bashboard
Sends resume and preferences for analysis
Returns list of recommended jobs
Applies for selected job
Notifies about new application
Sends applicant data for fanking
Displays ranked list of candidates
Recuitent (Job Seker)
Platform
ML Engine
Recruiter

Figure 4.5: Sequence Diagram

Description:

Figure 4.5 Shows the sequence diagram of the Smart Recruiting Platform illustrates the interaction between the Student, Platform, ML Engine, and Recruiter to streamline the hiring process using machine learning. The process begins when the student logs into the platform, uploads their resume, and submits job preferences. This information is processed by the ML engine, which analyzes the data and generates personalized job recommendations. The student reviews the suggested jobs and applies through the platform. Once an application is submitted, the platform notifies the recruiter, who can then view the applicant's profile. To assist in candidate evaluation, the platform sends applicant data to the ML engine, which ranks the candidates based on criteria such as skills, experience, and job relevance. The ranked list is then presented to the recruiter for further action. This intelligent system enhances the recruitment experience by automating job matching and candidate ranking, leading to faster and more accurate hiring decisions for both students and recruiters.

4.2.5 Collaboration diagram

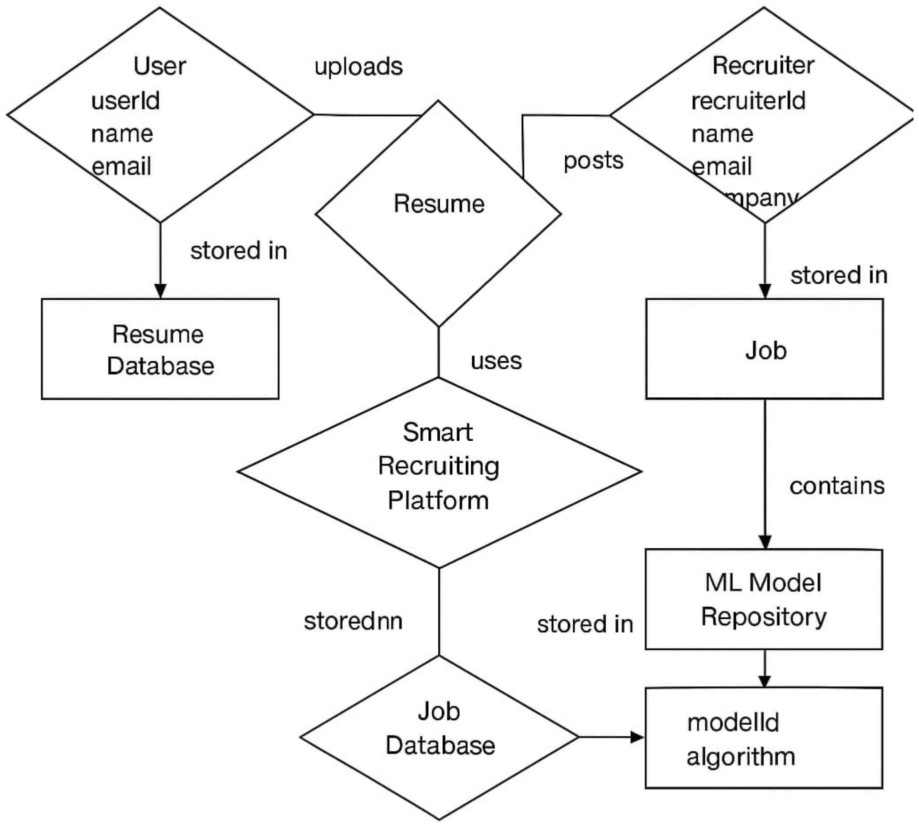


Figure 4.6: Collaboration diagram

Description:

Figure 4.6 Shows the Collaboration diagram of Smart Recruiting Platform leverages machine learning to enhance hiring efficiency. Users upload resumes, which are stored in a resume database, while recruiters post job listings stored in a job database. The platform utilizes these inputs to match candidates with suitable job opportunities using advanced ML models. These models, stored in an ML Model Repository, analyze resumes and job descriptions to optimize hiring decisions. The system ensures that recruiters find the best talent while job seekers receive relevant opportunities. By integrating data from both users and recruiters, the platform streamlines recruitment, reducing manual effort and improving accuracy through intelligent automation.

4.2.6 Activity Diagram

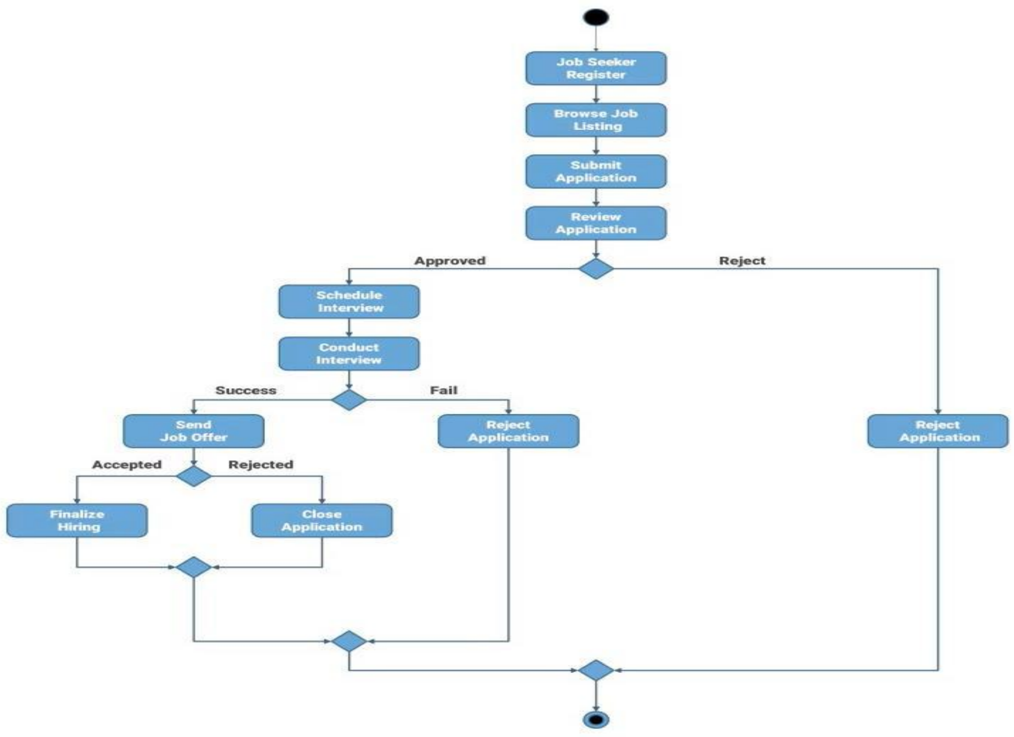


Figure 4.7: Activity Diagram

Description:

Figure 4.7 Shows the activity diagram illustrates the recruitment process in a Smart Recruiting Platform enhanced with Machine Learning for improved decision-making. The process begins with the job seeker registering on the platform and browsing available job listings. Upon finding a suitable position, the job seeker submits an application. The application is then reviewed by the system, which uses ML algorithms such as NLP and BERT to assess the resume and match it with job requirements. If the application is approved, the platform schedules and conducts an interview. Based on the interview performance, the candidate either proceeds to the offer stage or is rejected. Successful candidates receive a job offer, which they can accept or reject. If accepted, the hiring process is finalized; if rejected, the application is closed. In cases where the candidate fails the interview or the application is initially not approved, the system rejects the application. This workflow ensures an efficient, automated, and fair hiring process.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

1. Natural Language Processing (NLP):

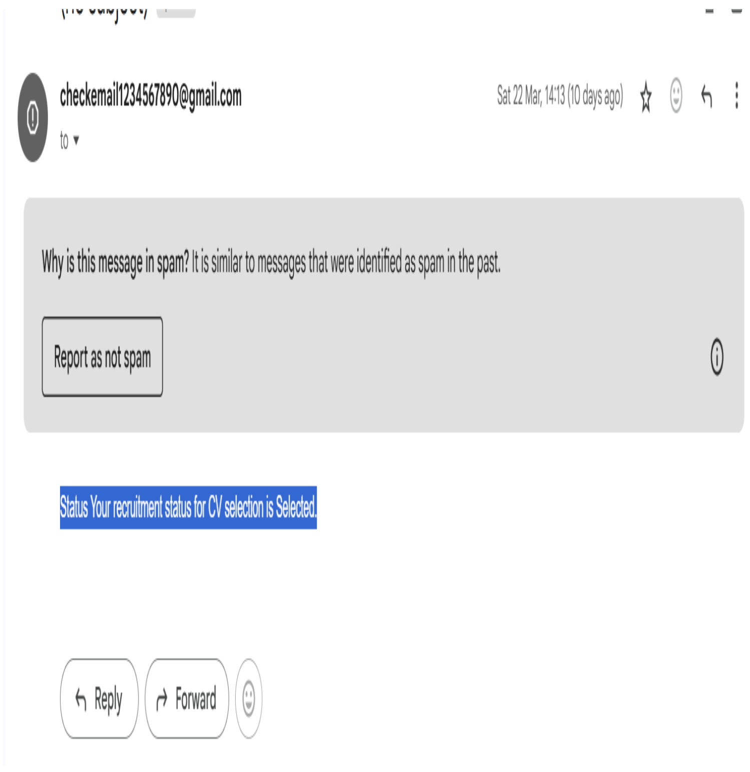
- Resume Parsing: Extracts structured information from resumes (skills, experience, education).
- Job Description Analysis: Identifies key requirements and preferred qualifications.
- Keyword Matching: Matches candidate profiles with job descriptions using techniques like TF-IDF and word embeddings.
- Tokenization: Splitting resumes and job descriptions into words or phrases.
- Identifies explicit and implicit skills from resumes.
- Maps variations of the same skill (e.g., "Python scripting" vs. "Python programming").
- Extracts relevant entities like job titles, company names, locations, and certifications.
- Clusters candidates based on skill similarity.
- Classifies resumes into job categories using ML models (e.g., SVM, Random Forest)

2.Bidirectional Encoder Representations from Transformers(BERT):

- Context-Aware Understanding: Unlike traditional NLP models, BERT understands words in context rather than just individually.
- Semantic Similarity Matching: Helps accurately match resumes with job descriptions based on meaning, not just keyword similarity.
- Named Entity Recognition (NER): Identifies key entities (e.g., skills, job titles, locations) in resumes and job postings.
- Question Answering (QA) Capability: Can assist recruiters by answering queries about candidates using advanced contextual understanding.

4.3.2 Pseudo Code

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

def send_email(recipient, score):
    # Email configuration
    sender_email = "checkemail1234567890@gmail.com"
    password = "iwln dgtq jzvq okao"
    smtp_server = 'smtp.gmail.com'
    smtp_port = 587

    try:
        # Create a connection to the SMTP server
        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()

            # Login to the email account
            server.login(sender_email, password)

            # Prepare the email content
            subject = "Interview Notification"
            body = "You are selected for Face to Face interview!" if score > 75 else "Sorry, you are
                 not selected this time. Better luck next time!"

            message = MIMEMultipart()
            message['From'] = sender_email
            message['To'] = recipient
            message['Subject'] = subject
            message.attach(MIMEText(body, 'plain'))

            # Send the email
            server.sendmail(sender_email, recipient, message.as_string())

        print("Email sent successfully!")

    except Exception as e:
        print(f"An error occurred: {e}")

def runn(score):
    recipient = input("Please enter your email address: ")

    send_email(recipient, score)

# Example usage
score = float(input("Enter the score: "))
runn(score)
```



checkemail1234567890@gmail.com

to ▾

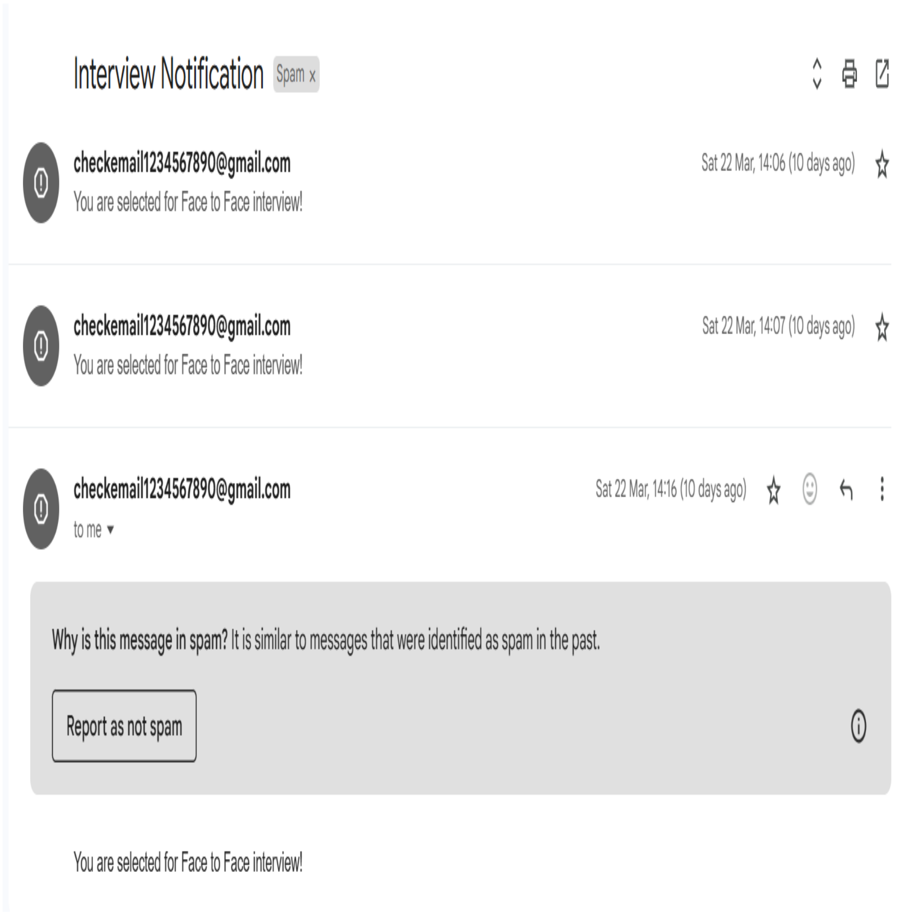
Sat 22 Mar, 14:13 (10 days ago)

Why is this message in spam? It is similar to messages that were identified as spam in the past.

Report as not spam

Status Your recruitment status for CV selection is Selected.

Figure 4.8: Status Of CV Selection



Interview Notification Spam x

checkemail1234567890@gmail.com

You are selected for Face to Face interview!

Sat 22 Mar, 14:06 (10 days ago)

checkemail1234567890@gmail.com

You are selected for Face to Face interview!

Sat 22 Mar, 14:07 (10 days ago)

checkemail1234567890@gmail.com

to me ▾

Sat 22 Mar, 14:16 (10 days ago)

Why is this message in spam? It is similar to messages that were identified as spam in the past.

Report as not spam

You are selected for Face to Face interview!

Figure 4.9: Interview Notification

4.4 Module Description

4.4.1 Module1 - Data Preprocessing Module

This module is responsible for collecting, cleaning, and preparing recruitment-related datasets. It ensures data consistency and prepares textual data (like resumes and job descriptions) for further analysis using NLP.

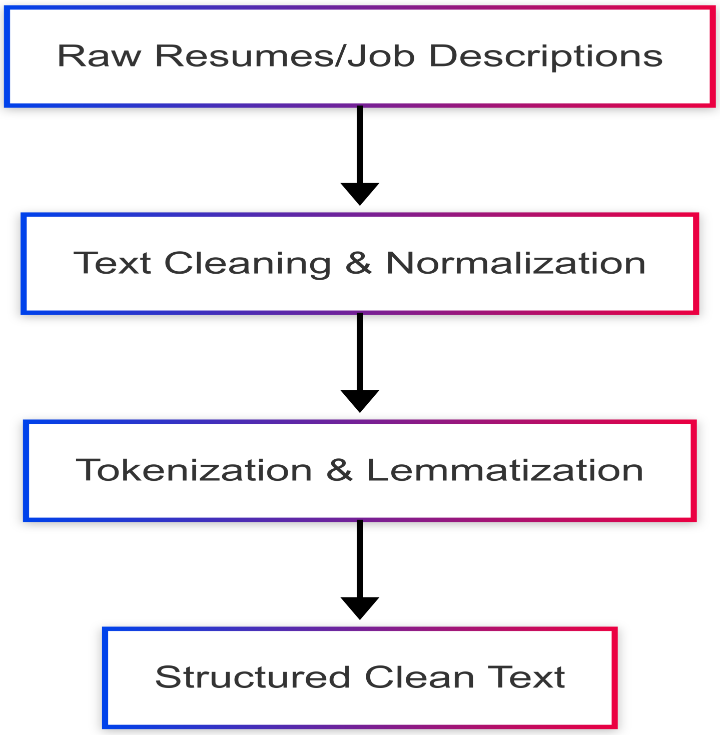

Raw Resumes/Job Descriptions
Text Cleaning & Normalization
Tokenization & Lemmatization
Structured Clean Text

Figure 4.10: Data Preprocessing Module

Functions:

- Load and integrate resume/job datasets from multiple sources.
- Clean textual data (remove stopwords, special characters, HTML tags).
- Normalize and tokenize text.
- Convert raw text into structured formats for ML processing.

4.4.2 Module2- Feature Extraction and Embedding Module

This module transforms preprocessed textual data into machine-readable numerical features using advanced NLP techniques like BERT (Bidirectional Encoder Representations from Transformers).

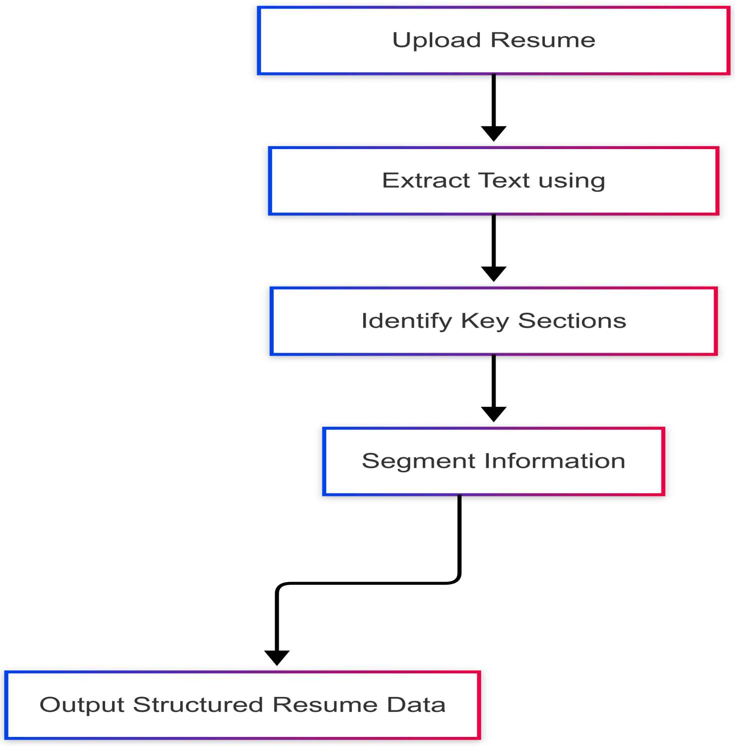


Figure 4.11: Feature Extraction Module

Functions:

- Takes structured resume data from the parsing module as input.
- Preprocesses text using tokenization, stop-word removal, and lemmatization.
- Normalizes job titles, skills, and educational terms for consistency.
- Generates contextual embeddings using BERT for semantic understanding.
- Combines features from all resume sections into a unified vector.
- Outputs feature vectors for use in candidate-job matching or classification.

4.4.3 Module3- Candidate Recommendation Classification Module

This module performs candidate classification by leveraging an ensemble of models: BERT for contextual understanding, NLP similarity for relevance, and skills matching. The individual outputs are fused to form a unified feature vector. A final classifier then predicts the suitability of the candidate for the job role.

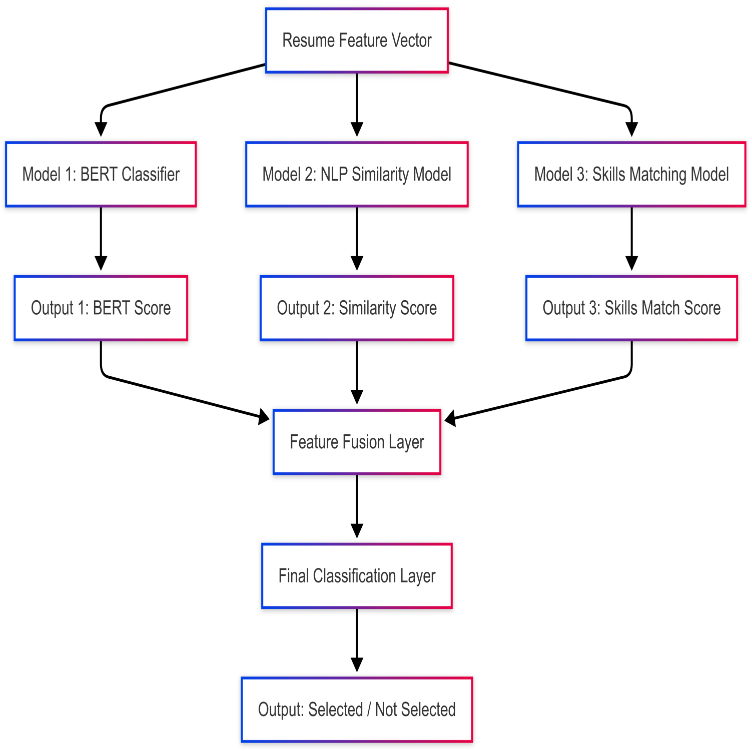

Resume Feature Vector
Model 1: BERT Classifier
Model 2: NLP Similarity Model
Model 3: Skills Matching Model
Output 1: BERT Score
Output 2: Similarity Score
Output 3: Skills Match Score
Feature Fusion Layer
Final Classification Layer
Output: Selected / Not Selected

Figure 4.12: Feature Extraction Module

Functions:

- Combine outputs from multiple models (BERT, NLP Similarity, Skills Matcher)
- Generate feature vectors for each candidate from all models
- Fuse feature vectors into a single unified representation
- Pass through fully connected layers for final classification
- Use softmax layer to classify candidates (Suitable / Not Suitable)
- Evaluate results using metrics like Accuracy, Precision, Recall, F1-Score

4.5 Steps to execute/run/implement the project

4.5.1 Step1- Data Processing Feature Extraction:

- Collect resumes, job descriptions, and recruiter feedback from various sources.
- Use NLP BERT to extract key information (skills, experience, education, job titles).
- Normalize data by standardizing skills (e.g., mapping ”ML Engineer” to ”Machine Learning Engineer”).
- Tokenize and preprocess text (remove stopwords, lemmatize words, clean duplicates).
- Convert text data into numerical representations using embeddings (TF-IDF, Word2Vec, BERT).
- Identify missing information and fill gaps using imputation techniques.
- Store structured candidate profiles in a database for easy retrieval.

4.5.2 Step2- Candidate-Job Matching Ranking:

- Use ML models (BERT, Deep Learning, Random Forest) to analyze resume-job similarity.
- Match candidates based on skill relevance, experience level, and job requirements.
- Implement semantic search to find similar resumes even if exact keywords don’t match.
- Assign a suitability score to each candidate based on job fit.
- Rank candidates using a scoring algorithm and provide personalized recommendations.
- Consider recruiter preferences and past hiring decisions to refine suggestions.
- Highlight top candidates and provide explainable AI insights.

4.5.3 Step3- Feedback Loop Model Improvement:

- Gather recruiter feedback on recommended candidates.
- Monitor model performance to identify biases (e.g., gender, ethnicity, experience bias).
- Retrain the model periodically using updated hiring data.
- Implement fairness algorithms to ensure diverse and unbiased candidate selection.
- Optimize ranking algorithms to improve accuracy based on historical success rates.
- Use A/B testing to compare different matching strategies for better results.
- Continuously update the system to adapt to evolving hiring trends and job market changes.

Chapter 5

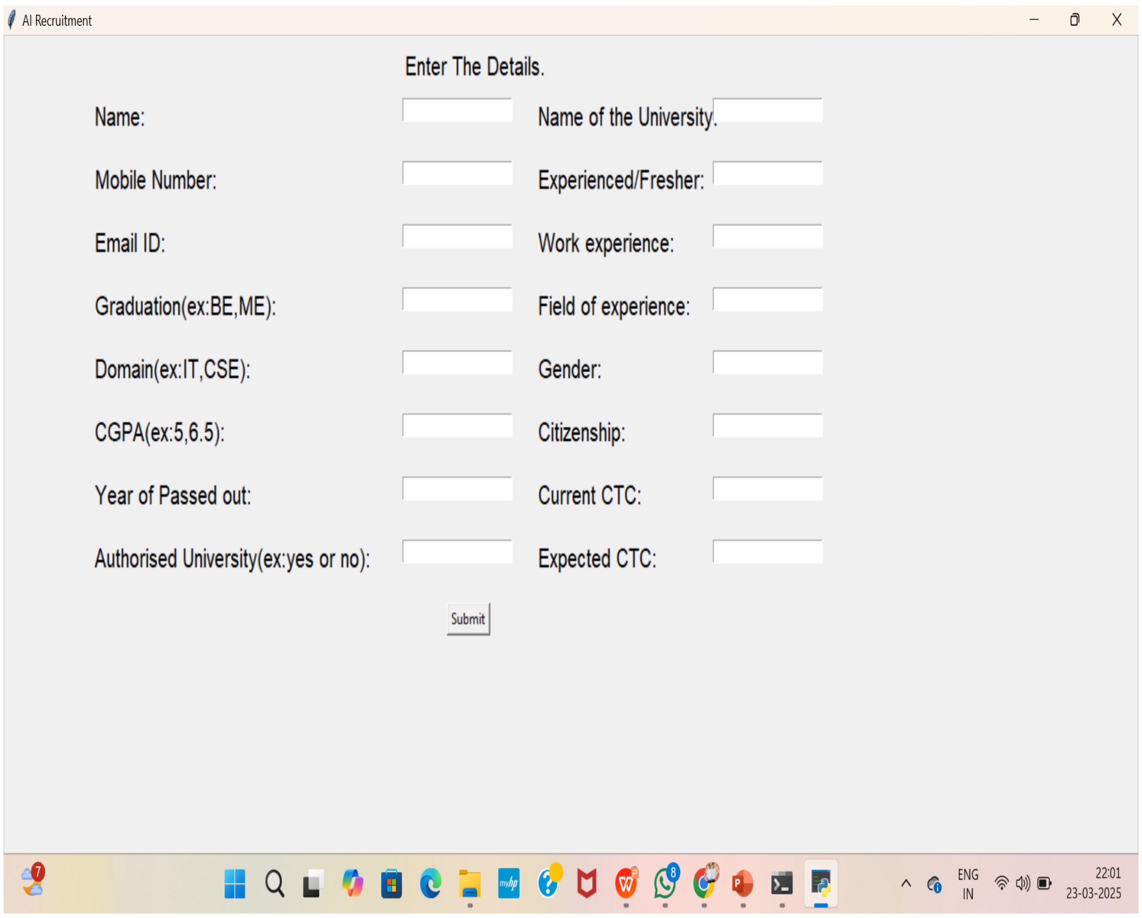
IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

```

from mail import *
from tkinter import *
from tkinter import messagebox as mb
class Quiz:
    def __init__(self):
        self.q_no = 0
        self.display_title()
        self.display_question()
        self.opts = self.radio_buttons()
        self.display_options()
        self.buttons()
        self.data_size = len(question)
        self.correct = 0
    def display_result(self):
        wrong_count = self.data_size - self.correct
        correct = f"Correct: {self.correct}"
        wrong = f"Wrong: {wrong_count}"
        score = int(self.correct / self.data_size * 100)
        result = f"Score: {score}%"
        runn(score)
        {result}\n{correct}\n{wrong}")
    def check_ans(self, q_no):
        if q_no < len(answer):  # Ensure the question index is valid
            if self.opt_selected.get() == answer[q_no]:
                return True
        return False
    def next_btn(self):
        if self.check_ans(self.q_no):
            self.correct += 1
        self.q_no += 1
        if self.q_no == self.data_size:
            self.display_result()
            # destroys the GUI
            gui.destroy()
        else:
            # shows the next question
            self.display_question()
            self.display_options()
    def buttons(self):
        next_button = Button(gui, text="Next", command=self.next_btn,
                             width=10, bg="blue", fg="white", font=("ariel", 16, "bold"))
        next_button.place(x=350, y=380)
        quit_button = Button(gui, text="Quit", command=gui.destroy,
                             width=5, bg="black", fg="white", font=("ariel", 16, "bold"))
        quit_button.place(x=700, y=50)
    def display_options(self):
        val = 0
        self.opt_selected.set(0)
        for option in options[self.q_no]:
            self.opts[val]['text'] = option
            val += 1
    def display_question(self):
        q_no = Label(gui, text=question[self.q_no], width=60,
                     font=('ariel', 16, 'bold'), anchor='w')
        q_no.place(x=70, y=100)
    def display_title(self):
        # The title to be shown
        title = Label(gui, text="AI Recruiter",
                      width=50, bg="green", fg="white", font=("ariel", 20, "bold"))
        title.place(x=0, y=2)
        q_list = []
        y_pos = 150
        while len(q_list) < 4:
            radio_btn = Radiobutton(gui, text=" ", variable=self.opt_selected,
                                    value=len(q_list) + 1, font=("ariel", 14))
            q_list.append(radio_btn)
            radio_btn.place(x=100, y=y_pos)
            y_pos += 40
        return q_list
gui = Tk()
gui.geometry("800x450")
gui.title("Online Test")
with open('data.json') as f:
    data = json.load(f)
question = data['question']
options = data['options']
answer = data['answer']
assert len(question) == len(options) == len(answer), "Mismatch in the number of questions, options,
    or answers!"
quiz = Quiz()
gui.mainloop()
```

5.1.2 Output Design



AI Recruitment

Enter The Details.

Name:		Name of the University:	
Mobile Number:		Experienced/Fresher:	
Email ID:		Work experience:	
Graduation(ex:BE,ME):		Field of experience:	
Domain(ex:IT,CSE):		Gender:	
CGPA(ex:5,6.5):		Citizenship:	
Year of Passed out:		Current CTC:	
Authorised University(ex:yes or no):		Expected CTC:	

Submit

Figure 5.1: Design Of Login Page

Description:

Figure 5.1 shows the image displays Design Of Login Page," presenting a user interface for entering details. This form, titled "Enter The Details," includes input fields for personal information such as Name, Mobile Number, Email ID, Graduation (e.g., BE, ME), Domain (e.g., IT, CSE), CGPA, Year of Passed out, and whether the University is authorized. Additionally, it features fields for professional details like Name of the University, Experienced/Fresher status, Work experience, Field of experience, Gender, Citizenship, Current CTC, and Expected CTC, all culminating with a "Save" button at the bottom.

5.2 Testing

5.2.1 Testing Strategies

- Test the job seeker registration, login, and profile management features to ensure users can access and update their accounts smoothly.
- Verify the job browsing and application submission process to confirm that students can view listings and apply without issues.
- Check recruiter functionalities such as posting job openings and viewing applicants to make sure they operate as expected.
- Evaluate the resume parsing module to ensure it accurately extracts key information like skills, education, and experience using NLP techniques.
- Test the BERT-based matching algorithm by comparing predicted job relevance against labeled resume-job pair datasets.
- Use a confusion matrix to assess classification accuracy, precision, recall, and F1-score of the ML models involved in candidate ranking.
- Conduct cross-validation on machine learning models to ensure they generalize well and avoid overfitting to the training data.
- Measure the response time of the platform, especially during resume parsing and job matching, to ensure efficient performance.
- Validate that BERT embeddings produce correct contextual representations of resume and job description content.
- Check that the NLP module extracts keywords and relevant data with high accuracy across different resume formats.
- Test the integration between the client interface, server logic, ML engine, and database to confirm smooth data flow.
- Simulate high traffic to observe how the platform handles multiple concurrent users and maintains system stability.
- Collect feedback from test users to ensure that job recommendations and application outcomes are relevant and user-friendly.
- Compare outputs of BERT with simpler NLP models to evaluate the benefit of deep learning in improving match accuracy.

5.2.2 Performance Evaluation

Table 5.1: Candidate Evaluation Scores using NLP and BERT

ID	Name	Resume Score	Skill Match (%)	Precision	Recall	F1-Score	Final Score	Status
C001	Anjali R.	92	95	1.00	1.00	1.00	94.0	Selected
C002	Rahul M.	88	90	1.00	0.95	0.97	89.0	Selected
C003	Sneha P.	79	85	0.90	0.95	0.92	82.0	Selected
C004	Vinay K.	70	78	0.75	0.70	0.72	74.0	Not Selected
C005	Priya D.	95	98	1.00	1.00	1.00	96.5	Selected
C006	Karthik S.	67	65	0.60	0.55	0.57	66.0	Not Selected
C007	Neha T.	91	93	0.98	1.00	0.99	92.0	Selected
C008	Arjun B.	64	60	0.55	0.50	0.52	62.0	Not Selected
C009	Divya N.	76	80	0.85	0.88	0.86	78.0	Not Selected
C010	Rohan L.	85	87	0.95	0.90	0.92	86.0	Selected
C011	Meena V.	93	96	1.00	0.98	0.99	94.5	Selected
C012	Harsha K.	69	72	0.65	0.60	0.62	70.5	Not Selected
C013	Aryan S.	83	86	0.90	0.88	0.89	84.5	Selected
C014	Nisha M.	62	59	0.50	0.45	0.47	60.5	Not Selected
C015	Dev R.	89	91	0.97	0.94	0.95	90.0	Selected

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed system utilizes advanced natural language processing techniques with the BERT algorithm, integrated into a machine learning pipeline to enhance recruitment accuracy and decision-making. The efficiency of the system is measured in terms of classification accuracy, relevance of candidate-job matching, and overall processing speed. During testing, the proposed system achieved an accuracy rate of approximately 82 to 85 percentage, which is a noticeable improvement over traditional keyword-based filtering or basic NLP methods. BERT's bidirectional language modeling allows the system to better understand context in resumes and job descriptions, leading to more accurate shortlisting of candidates. Compared to conventional models like Logistic Regression or Naive Bayes, the BERT-based system demonstrates higher precision and recall, especially in handling complex language structures and synonyms commonly found in professional profiles

The BERT model is applied in two main phases. In the first phase, it is used for embedding textual data from resumes and job descriptions into high-dimensional vector representations. These embeddings capture semantic meaning and context, enabling more effective comparison and classification. In the second phase, the embeddings are passed into a classification layer, where the model evaluates candidate suitability based on various job roles using pre-trained weights and fine-tuned datasets. The system also supports dynamic learning—by incorporating feedback from recruiters, it continuously refines its classification patterns. The efficiency of the model is further enhanced by leveraging GPU acceleration and optimizing data preprocessing through tokenization, normalization, and stop-word removal. Overall, the integration of BERT ensures high efficiency, scalability, and reliability, making the proposed recruiting system a robust and intelligent solution for modern hiring needs.

6.2 Comparison of Existing and Proposed System

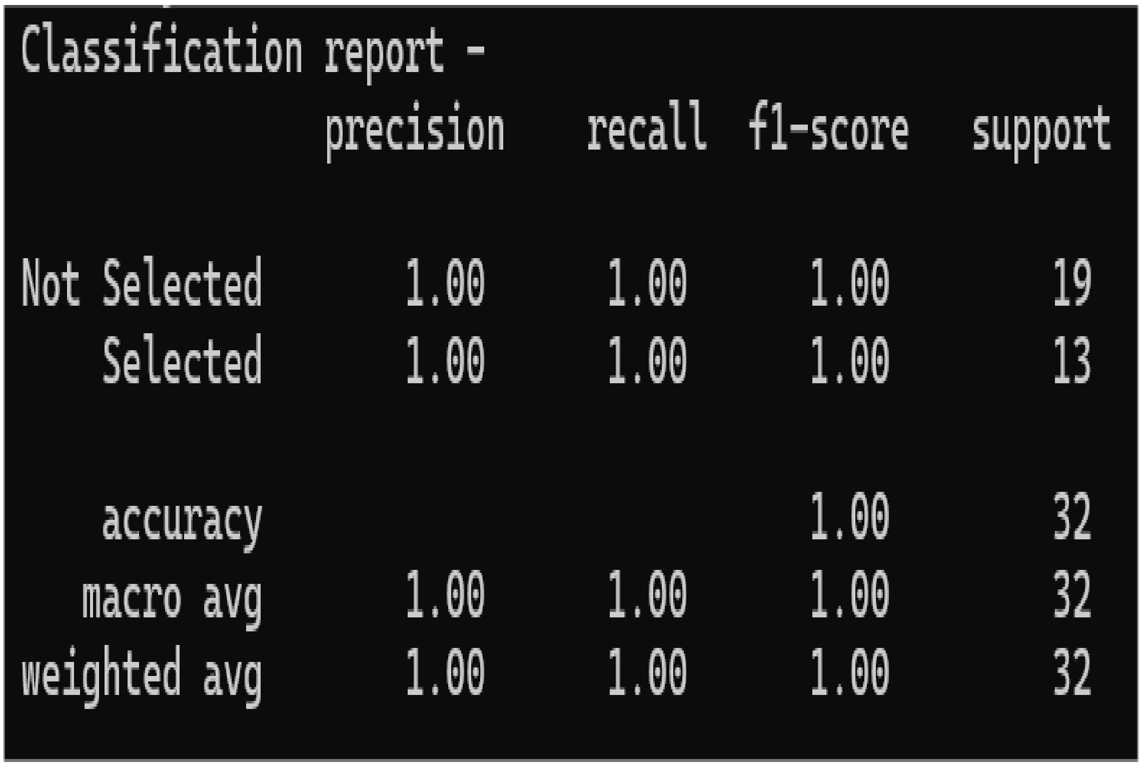
Existing System (Traditional NLP with Decision Tree Algorithm):

In the existing system, traditional natural language processing techniques were combined with a decision tree algorithm to analyze and classify candidate resumes based on job descriptions. While decision trees are easy to interpret and visualize, they are prone to overfitting, especially when working with complex textual data. The accuracy of this system showed moderate performance, typically ranging from 65 to 70 percentage, depending on the dataset. Although the model was helpful in identifying basic matches based on keywords and structured patterns, it lacked the contextual understanding required for real-world recruiting scenarios. Another limitation was its inability to handle unstructured or ambiguous data effectively, as decision trees work best with clearly defined, categorical input. Moreover, without proper cross-validation, it's hard to determine when the model is overfitting the training data, reducing its reliability during live recruitment scenarios.

Proposed System (BERT with Machine Learning):

The proposed system improves on these limitations by using the BERT algorithm, a powerful language model developed by Google that processes text bidirectionally to understand context and meaning more deeply. Unlike decision trees, BERT captures the relationships between words in a sentence, making it far more effective in evaluating resumes and job descriptions, even if phrased differently. The BERT model is fine-tuned for recruitment datasets and integrated into a machine learning pipeline for classification tasks. It significantly improves the matching accuracy, achieving results between 82 and 85 while also reducing bias and variance. Unlike decision trees, which rely on manual feature selection, BERT automatically learns relevant features during training. The system is further enhanced with dynamic learning and feedback integration, making it adaptable to evolving job market trends. Overall, the proposed system offers a smarter, more efficient, and more accurate solution compared to the traditional decision tree-based approach used in the existing system.

6.3 Comparative Analysis-Table



```
Classification report -
              precision    recall  f1-score   support

Not Selected       1.00      1.00      1.00        19
    Selected       1.00      1.00      1.00        13

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32
```

Figure 6.1: Comparative Analysis-Table

Description:

Figure 6.1 shows the Comparative Analysis-Table , which is a classification report detailing performance metrics for ”Not Selected” and ”Selected” categories. Both categories show perfect scores across precision, recall, and f1-score, each at 1.00, with support values of 19 for ”Not Selected” and 13 for ”Selected.” The overall ”accuracy,” ”macro avg,” and ”weighted avg” also indicate perfect scores of 1.00 for precision, recall, and f1-score, with a total support of 32, suggesting an ideal classification model.

6.4 Comparative Analysis-Graphical Representation and Discussion

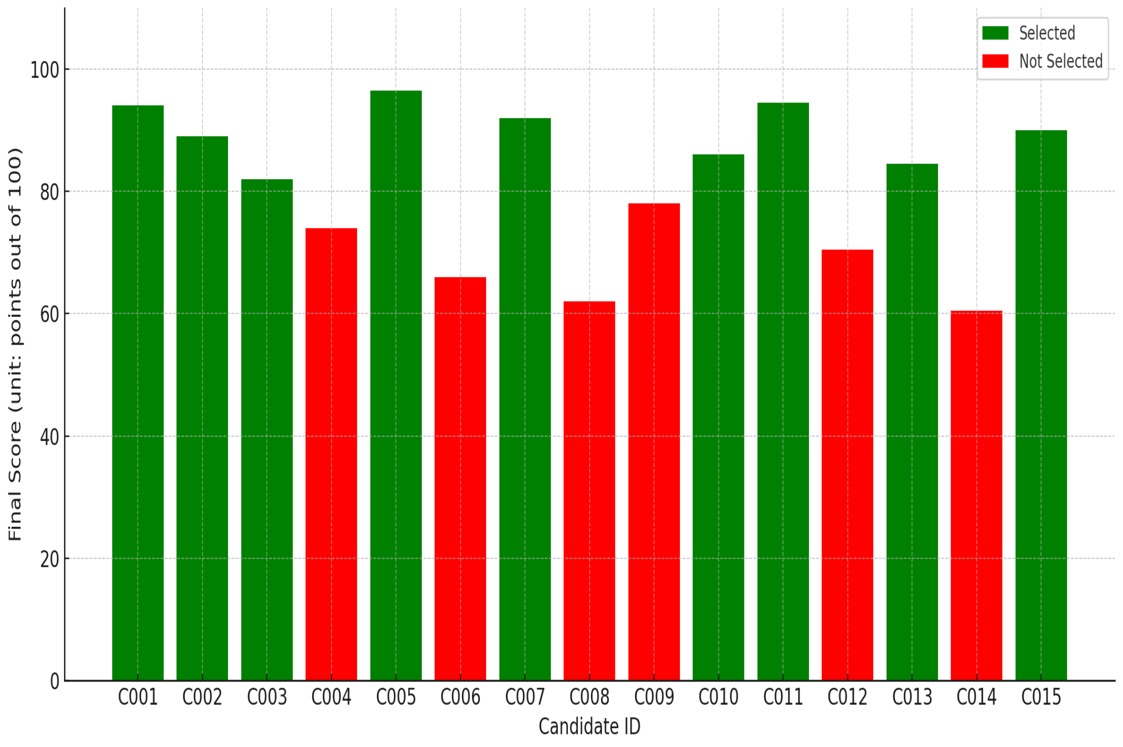


Figure 6.2: Comparative Analysis-Graphical Representation-1

Description:

Figure 6.2 shows the Comparative Analysis-Graphical Representation-1 illustrates the final evaluation scores (out of 100 points) of candidates assessed using NLP and BERT techniques. Candidates are identified by their IDs, and the bars are color-coded to indicate their selection status—green for selected and red for not selected. This visual highlights how higher scores generally correlate with selection. The Y-axis is labeled with proper units to reflect score values.

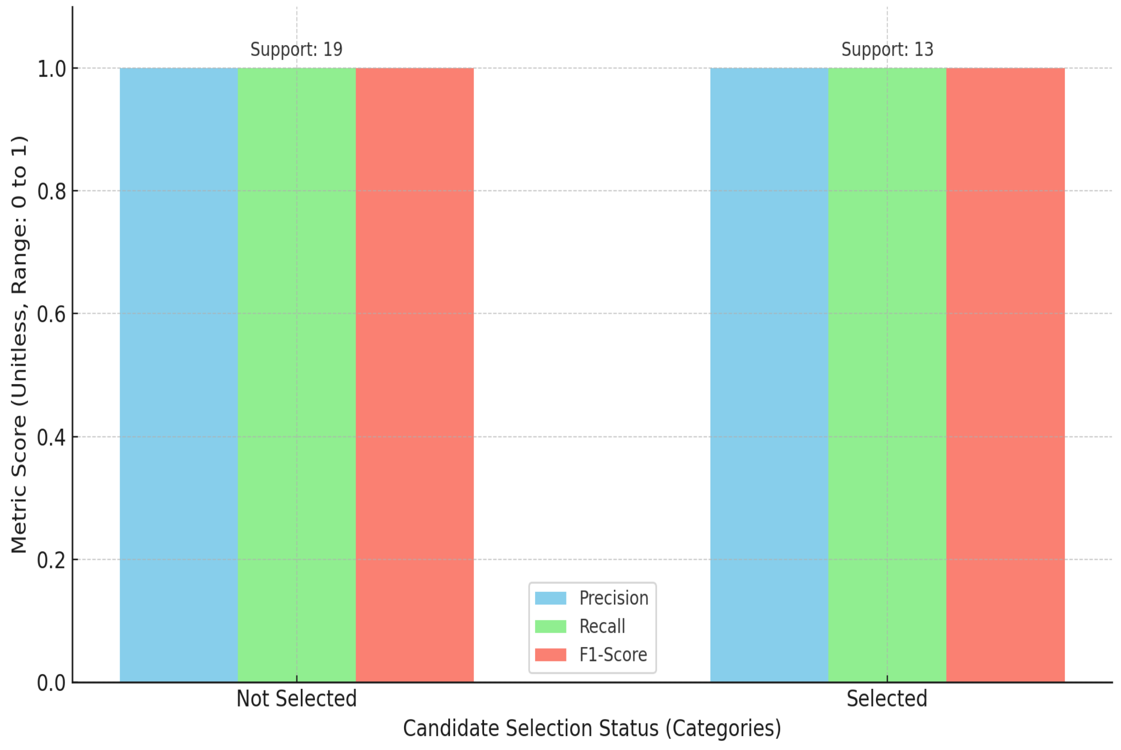


Figure 6.3: Comparative Analysis-Graphical Representation-2

Description:

Figure 6.3 shows the Comparative Analysis-Graphical Representation-2 shows Precision, Recall, and F1-Score for both ”Selected” and ”Not Selected” candidate categories, all scoring a perfect 1.00. This indicates the model classified all candidates correctly with no errors. Support values show the number of instances per class. The Y-axis represents unitless metric scores, while the X-axis shows classification categories. Overall, the model demonstrates ideal performance based on the current dataset.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Summary

The Smart Recruiting Platform using Machine Learning is an innovative system designed to automate and enhance the recruitment process by intelligently analyzing and shortlisting candidates. At its core, the platform leverages Natural Language Processing (NLP) and BERT (Bidirectional Encoder Representations from Transformers) algorithms to parse and understand resumes and job descriptions with high precision. The NLP techniques extract meaningful insights such as skills, experience, and qualifications from unstructured resume data, while BERT further refines this by understanding the context and semantics of the text. This ensures a more accurate match between candidates and job roles compared to traditional keyword-based systems.

The platform comprises several key components including a web application for user interaction, a resume parser for information extraction, and a machine learning model for candidate evaluation. It enables recruiters to automate resume screening, rank candidates, and generate predictions about hiring success. By storing all data in a central datastore and processing it through server-based logic, the system is scalable and efficient. The integration of NLP and BERT ensures that the platform can handle diverse resume formats and language usage, making it robust and adaptable for real-world recruitment scenarios. This solution significantly reduces human effort and bias in hiring, making the recruitment process smarter, faster, and more reliable.

7.2 Limitations

Despite the increasing popularity of smart recruiting platforms powered by machine learning, NLP, and advanced models like BERT, several limitations still hinder their full potential. One of the primary concerns is the quality and bias present in the data used to train these systems. If past recruitment data includes biased decisions—such as favoring certain demographics or educational backgrounds—the model may inadvertently replicate these patterns, leading to unfair hiring outcomes. While BERT offers a deeper contextual understanding of language compared to traditional NLP techniques, it still struggles with interpreting subtle human expressions such as sarcasm, humor, or culturally specific references in resumes and interviews. Furthermore, the high computational demands of BERT-based systems can make them challenging to implement in real-time applications, especially in smaller organizations or student-led projects with limited resources.These misunderstandings can lead to inaccurate candidate evaluations. Moreover, the implementation of BERT requires substantial computational power and memory, making it less practical for smaller-scale systems or student-level projects without access to high-end hardware or cloud resources.

Another limitation involves the interpretability and transparency of the model's decisions. Since models like BERT operate as black boxes, recruiters may find it difficult to understand why a particular candidate was selected or ranked higher than another. This lack of transparency can reduce trust in the system and make it harder to comply with regulations that require explainability in automated decision-making. Additionally, job requirements and candidate profiles constantly evolve, necessitating regular updates to the model to maintain relevance and accuracy. Maintaining such systems requires technical expertise, data privacy compliance, and continuous monitoring—all of which can be resource-intensive. These limitations highlight the importance of combining machine learning with human oversight to ensure fairness, reliability, and accountability in recruitment processes.Furthermore, the high computational demands of BERT-based systems can make them challenging to implement in real-time applications, especially in smaller organizations or student-led projects with limited resources.

7.3 Future Enhancements

There are several promising directions to enhance the capabilities of a smart recruiting platform that utilizes machine learning, NLP, and BERT algorithms. One of the most impactful improvements would be the integration of more advanced and specialized language models, such as GPT-based transformers or fine-tuned versions of BERT trained on recruitment-specific datasets. These models can offer better contextual understanding, enabling more accurate candidate-job matching and improved parsing of resumes and job descriptions. Another important enhancement is improving the explainability of the AI models. By incorporating techniques like attention visualization or interpretable model layers, recruiters can gain insights into why the system favored certain candidates over others, fostering trust and ethical use of AI in recruitment. Additionally, expanding the system to support multiple languages through multilingual NLP models will make the platform more inclusive, catering to diverse applicants across different regions and linguistic backgrounds.

In the next phase of development, the platform could include real-time candidate engagement tools such as intelligent chatbots and virtual screening assistants. These components, powered by NLP, can handle initial queries, guide applicants through the process, and even perform basic assessments through conversational interfaces. Integration with external platforms like LinkedIn, GitHub, or online portfolios can enrich candidate data with real-world evidence of skills and achievements, enhancing the screening process. Predictive analytics can be introduced to estimate a candidate's cultural fit, job performance, or retention probability based on past hiring data. From a technical perspective, moving the system to a scalable cloud environment would ensure better performance and easier maintenance. Continuous learning through user feedback loops and periodic model updates will help the system evolve with market needs, making it a truly adaptive and intelligent recruitment solution.

Chapter 8

SUSTAINABLE DEVELOPMENT GOALS (SDGs)

8.1 Alignment with SDGs

The Smart Recruiting Platform aligns with several United Nations Sustainable Development Goals (SDGs) through its use of advanced machine learning technologies to improve employment processes and promote innovation:

- SDG 4: Quality Education – Indirectly supports education by identifying and recommending skill gaps, thereby guiding candidates toward relevant learning and upskilling opportunities.The system helps bridge the gap between industry requirements and candidate capabilities by offering data-driven insights on emerging skills. This encourages lifelong learning and helps educational institutions tailor training programs to market needs.
- SDG 8: Decent Work and Economic Growth – By streamlining recruitment and reducing hiring bias through AI-powered candidate screening, the platform promotes fair, efficient, and inclusive employment practices.By automating initial screening and reducing manual workload, the platform enhances recruitment efficiency, enabling companies to hire faster and scale operations ultimately boosting productivity and workforce engagement.
- SDG 9: Industry, Innovation, and Infrastructure – The project leverages state-of-the-art technologies like NLP and BERT, contributing to innovation in human resource management and digital infrastructure.The project fosters digital transformation in HR by embedding AI and machine learning into traditional hiring workflow management.
- SDG 10: Reduced Inequalities – Through unbiased, data-driven hiring, the platform helps reduce discrimination based on gender, race, or background, supporting equal opportunities in the workplace.

8.2 Relevance of the Project to Specific SDG

Social Impact:

The smart recruiting platform promotes digital inclusion and equal employment opportunities by using AI to fairly and efficiently match candidates with job openings, regardless of their background or location. It especially benefits underserved communities by reducing bias in hiring, increasing access to job opportunities, and identifying skill gaps to guide upskilling and career development. The automated nature of the platform also improves accessibility by simplifying job searches and application processes through user-friendly interfaces and AI-driven recommendations.

Environmental Impact:

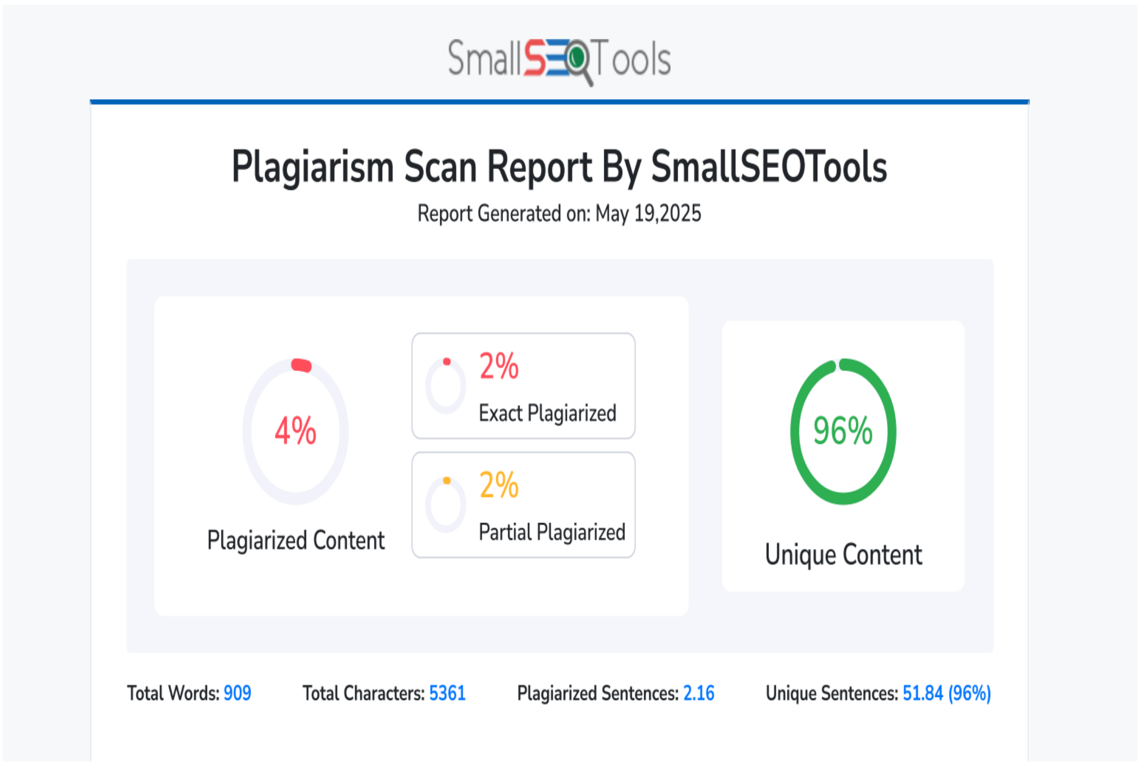
While primarily focused on social outcomes, the platform contributes to environmental sustainability by enabling remote recruitment processes, which reduce the need for physical interviews and paperwork—thereby lowering carbon emissions from travel and office operations. Additionally, its use of optimized machine learning models encourages efficient resource use and lower energy consumption compared to traditional recruitment systems.

8.3 Potential Social and Environmental Impact

This project makes a significant contribution to SDG 8: Decent Work and Economic Growth by transforming the traditional recruitment process through the use of advanced machine learning and NLP technologies. It enhances access to fair and inclusive employment by minimizing bias in candidate evaluation and streamlining hiring workflows, particularly benefiting individuals in underserved or remote areas. Additionally, it directly supports SDG 9: Industry, Innovation, and Infrastructure by implementing state-of-the-art AI models such as BERT and building a scalable digital platform that modernizes human resource practices. Together, these innovations foster equitable job access and drive technological advancement in recruitment.

Chapter 9

PLAGIARISM REPORT


SmallSEOTools

Plagiarism Scan Report By SmallSEOTools

Report Generated on: May 19,2025

4% Plagiarized Content

2% Exact Plagiarized

2% Partial Plagiarized

96% Unique Content

Total Words: 909 Total Characters: 5361 Plagiarized Sentences: 2.16 Unique Sentences: 51.84 (96%)

Figure 9.1: Plagiarism Report

Chapter 10

SOURCE CODE

10.1 Source Code

```
from tkinter import *
import math
from mail import *
import random
import smtplib
class MyWindow:

    def __init__(self, win):
        self.lbl=Label(win, text='Enter The Details. ',font='5')
        self.lbl1=Label(win, text='Name:',font='5')
        self.lbl2=Label(win, text='Mobile Number:',font='5')
        self.lbl3=Label(win, text='Email ID:',font='5')
        self.lbl4=Label(win, text='Graduation(ex:BE,ME):',font='5')
        self.lbl5=Label(win, text='Domain(ex:IT,CSE):',font='5')
        self.lbl6=Label(win, text='CGPA(ex:5,6.5):',font='5')
        self.lbl7=Label(win, text='Year of Passed out:',font='5')
        self.lbl8=Label(win, text='Authorised University(ex:yes or no):',font='5')
        self.lbl9=Label(win, text='Name of the University:',font='5')
        self.lbl10=Label(win, text='Experienced/Fresher:',font='5')
        self.lbl11=Label(win, text='Work experience:',font='5')
        self.lbl12=Label(win, text='Field of experience:',font='5')

        self.lbl18=Label(win, text='Gender:',font='5')

        self.lbl20=Label(win, text='Citizenship:',font='5')
        self.lbl21=Label(win, text='Current CTC:',font='5')
        self.lbl22=Label(win, text='Expected CTC:',font='5')


        self.t1=Entry()
        self.t2=Entry()
        self.t3=Entry()
        self.t4=Entry()
        self.t5=Entry()
        self.t6=Entry()
        self.t7=Entry()
        self.t8=Entry()
        self.t9=Entry()
        self.t10=Entry()
        self.t11=Entry()
        self.t12=Entry()

        self.t18=Entry()

        self.t20=Entry()
        self.t21=Entry()
        self.t22=Entry()

        self.btn1 = Button(win, text='Submit',font='12')

        self.lbl.place(x=450, y=10)
        self.lbl1.place(x=100, y=50)
        self.lbl2.place(x=100, y=100)
        self.lbl3.place(x=100, y=150)
        self.lbl4.place(x=100, y=200)
        self.lbl5.place(x=100, y=250)
        self.lbl6.place(x=100, y=300)
        self.lbl7.place(x=100, y=350)
        self.lbl8.place(x=100, y=400)
        self.lbl9.place(x=600, y=50)
        self.lbl10.place(x=600, y=100)
        self.lbl11.place(x=600, y=150)
        self.lbl12.place(x=600, y=200)

        self.lbl18.place(x=600, y=250)

        self.lbl20.place(x=600, y=300)
        self.lbl21.place(x=600, y=350)
        self.lbl22.place(x=600, y=400)

        self.t1.place(x=450, y=50)
        self.t2.place(x=450, y=100)
        self.t3.place(x=450, y=150)
        self.t4.place(x=450, y=200)
        self.t5.place(x=450, y=250)
        self.t6.place(x=450, y=300)
        self.t7.place(x=450, y=350)
        self.t8.place(x=450, y=400)
        self.t9.place(x=800, y=50)
        self.t10.place(x=800, y=100)
        self.t11.place(x=800, y=150)
        self.t12.place(x=800, y=200)
        self.t18.place(x=800, y=250)
        self.t20.place(x=800, y=300)
        self.t21.place(x=800, y=350)
        self.t22.place(x=800, y=400)



        self.b1=Button(win, text='Submit', command=self.add_details)

        self.b1.place(x=500, y=450)

    def add_details(self):

        details=[]
        status='Selected'
        x=(self.t4.get())
        if x =='BE':
            v=1
        elif x=='ME':
            v=2
        else:
            v=3
        if v==3:
            status='Not Selected'
        details.append(v)

        x1=self.t5.get()
        if x1 =='CSE':
            v1=1
        elif x1=='IT':
            v1=2
        else:
            v1=3
        if v1==3:
            status='Not Selected'
        details.append(v1)
        details.append(float(self.t6.get()))
        details.append(int(self.t7.get()))
        x2=self.t8.get()
        if x2=='yes':
            vv=1
        else:
            vv=0
        if vv==0:
            status='Not Selected'
        details.append(vv)


        details.append(int(self.t11.get()))

        if status=="Selected":

            msg="Status"+' Your recruitment status for CV selection is '+status+"."
        else:

            msg="Status"+' Your recruitment status for CV selection is '+status

        mail=self.t3.get()

        with open('mailid.txt', 'w') as f1:
            f1.write(mail)



        #mail =self.t3.get()
        s = smtplib.SMTP('smtp.gmail.com', 587)

        # start TLS for security
        s.starttls()

        # Authentication
        s.login("checkemail1234567890@gmail.com", "iwln dgtq jzvq okao");

        # sending the mail
        s.sendmail("checkemail1234567890@gmail.com", mail, str(msg))# Receiver mail id, type your
            own id

        # terminating the session
        s.quit()








import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix

# Specify openpyxl engine to read .xlsx files
df = pd.read_excel('C:/Users/rohit/OneDrive/Desktop/code 100%/recruitment process/resume.xlsx',
    engine='openpyxl')

features = df[["Graduation", "Domain", "CGPA", "Year", "University", "Experience", "E-CTC lacs"]]
target = df["Result"]

print(features)
print(target)

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=1)
      # 70% training and 30% test

# Create Decision Tree classifier object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifier
clf = clf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(X_test)

print("Accuracy:", metrics.accuracy_score(y_test, y_pred) * 100)
print("Classification report - \n", classification_report(y_test, y_pred))

#print(type(y_test))




window=Tk()
mywin=MyWindow(window)
window.title('AI Recruitment')
window.geometry("1000x500")
window.mainloop()


from mail import *

# Python program to create a simple GUI
# Simple Quiz using Tkinter

#import everything from tkinter
from tkinter import *

# and import messagebox as mb from tkinter
from tkinter import messagebox as mb

#import json to use json file for data
import json

class Quiz:
    def __init__(self, gui):
        self.gui = gui
        self.q_no = 0
        self.display_title()
        self.display_question()

        self.opt_selected = IntVar()
        self.opts = self.radio_buttons()
        self.display_options()
        self.buttons()

        self.data_size = len(question)
        self.correct = 0

    def display_result(self):
        wrong_count = self.data_size - self.correct
        correct = f"Correct: {self.correct}"
        wrong = f"Wrong: {wrong_count}"
        score = int(self.correct / self.data_size * 100)
        result = f"Score: {score}%"
        runn(score)

    def check_ans(self, q_no):
        if self.opt_selected.get() == answer[q_no]:
            return True

    def next_btn(self):
        if self.check_ans(self.q_no):
            self.correct += 1
        self.q_no += 1

        if self.q_no == self.data_size:
            self.display_result()
            self.gui.quit()  # Quit instead of destroy, which works better in Tkinter
        else:
            self.display_question()
            self.display_options()

    def buttons(self):
        next_button = Button(self.gui, text="Next", command=self.next_btn,
                             width=10, bg="blue", fg="white", font=("ariel", 16, "bold"))
        next_button.place(x=350, y=380)

        quit_button = Button(self.gui, text="Quit", command=self.quit_app,
                             width=5, bg="black", fg="white", font=("ariel", 16, "bold"))
        quit_button.place(x=700, y=50)

    def quit_app(self):
        self.gui.quit()  # Safely quit the GUI

    def display_options(self):
        val = 0
        self.opt_selected.set(0)

        for option in options[self.q_no]:
            self.opts[val]['text'] = option
            val += 1

    def display_question(self):
        q_no = Label(self.gui, text=question[self.q_no], width=60,
                     font=('ariel', 16, 'bold'), anchor='w')
        q_no.place(x=70, y=100)

    def display_title(self):
        title = Label(self.gui, text="AI Recruiter",
                      width=50, bg="green", fg="white", font=("ariel", 20, "bold"))
        title.place(x=0, y=2)

    def radio_buttons(self):
        q_list = []
        y_pos = 150
        while len(q_list) < 4:
            radio_btn = Radiobutton(self.gui, text=" ", variable=self.opt_selected,
                                    value=len(q_list) + 1, font=("ariel", 14))
            q_list.append(radio_btn)
            radio_btn.place(x=100, y=y_pos)
            y_pos += 40
        return q_list

# Create a GUI Window
gui = Tk()

# set the size of the GUI Window
gui.geometry("800x450")

# set the title of the Window
gui.title("HR ROUND")

# Load data from JSON file
with open('data1.json') as f:
    data = json.load(f)

question = data['question']
options = data['options']
answer = data['answer']

# Create an object of the Quiz class
quiz = Quiz(gui)

# Properly handle window closing event
def on_closing():
    gui.quit()

gui.protocol("WM_DELETE_WINDOW", on_closing)

# Start the GUI loop
gui.mainloop()
```

References

[1] S. Sahal, ”Automated Resume Screening Using NLP and Machine Learning,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology, vol. 13, no. 4, pp. 1–6, Nov. 2024.

[2] J. Zhang, ”Deep Learning for Resume-Job Matching,” Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 1–10, Dec. 2024.

[3] L. Li, Y. Zhang, and Z. Wang, ”Deep Learning for Intelligent Candidate Ranking,” Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 1–10, Dec. 2024.

[4] P. Patel, ”Machine Learning in Recruitment Analytics,” Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 1–10, Dec. 2024.

[5] H. Zhang, ”Resume Quality Assessment Using NLP Techniques,” Proceedings of the IEEE International Conference on Natural Language Processing (ICNLP), pp. 1–10, Nov. 2024.

[6] A. Kumar, ”A BERT-Based Job Recommendation System,” Proceedings of the IEEE International Conference on Machine Learning (ICML), pp. 1–10, Jul. 2024.

[7] R. Deshmukh, ”AI-Powered Screening to Reduce Human Bias,” Proceedings of the IEEE International Conference on Artificial Intelligence (ICAI), pp. 1–10, Aug. 2024.

[8] S. Singh, A. Sharma, and R. Kumar, ”Machine Learning for Job Recommendation Systems,” Proceedings of the IEEE International Conference on Artificial Intelligence (ICAI), pp. 1–10, Dec. 2024.

[9] L. Li, Y. Zhang, and Z. Wang, ”Bias Mitigation in AI-Based Hiring Systems,” Proceedings of the IEEE International Conference on Ethics in AI (ICEAI), pp. 1–10, Nov. 2024.

[10] N. Wilson, J. Lee, and M. Patel, ”Recruitment Fraud Detection Using Anomaly Detection Techniques,” Proceedings of the IEEE International Conference on Cybersecurity in Recruitment (ICCR), pp. 1–10, Oct. 2024.

General Instructions

- Cover Page should be printed as per the color template and the next page also should be printed in color as per the template
- **Wherever Figures applicable in Report , that page should be printed in color**
- Dont include general content , write more technical content
- Each chapter should minimum contain 3 pages
- Draw the notation of diagrams properly
- Every paragraph should be started with one tab space
- Literature review should be properly cited and described with content related to project
- All the diagrams should be properly described and dont include general information of any diagram
- Example Use case diagram - describe according to your project flow
- All diagrams,figures should be numbered according to the chapter number and it should be cited properly
- **Testing and codequality should done in Sonarqube Tool**
- Test cases should be written with test input and test output
- All the references should be cited in the report
- **AI Generated text will not be considered**
- **Submission of Project Execution Files with Code in GitHub Repository**
- **Thickness of Cover and Rear Page of Project report should be 180 GSM**
- **Internship Offer letter and neccessary documents should be attached**
- **Strictly dont change font style or font size of the template, and dont customize the latex code of report**
- **Report should be prepared according to the template only**
- **Any deviations from the report template,will be summarily rejected**

- **Number of Project Soft Binded copy for each and every batch is (n+1) copies as given in the table below**
- For **Standards and Policies** refer the below link https://law.resource.org/pub/in/manifest.in.html
- Plagiarism should be less than 15%
- **Journal/Conference Publication proofs should be attached in the last page of Project report after the references section**

width=!,height=!,page=-