FACULTY RESUME SCREENING

A PROJECT REPORT

Submitted by

KEERTHI VARSHA J[RA2211003011598]

SRIMAN E[RA2211003011568]

Under the Guidance of

Dr.K.VijiyaKumar

[Assistant Professor, Department of Computing Technologies]

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

MAY 2025

Department of Computing Technologies

SRM Institute of Science & Technology

Declaration Form

Degree/ Course : B.TECH

Student Name : KEERTHI VARSHA J, SRIMAN E

Registration Number : RA2211003011598, RA2211003011568

Title of Work : Faculty Resume Screening

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except were indicated by referring, and that I have followed the good academic practices noted above.
Sriman E RA2211003011568 — Keerthi Varsha J RA2211003011598

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

BONAFIDE CERTIFICATION

Certified that 21CSP302L - Project report titled "**FACULTY RESUME SCREENING**" is the Bonafide work of **KEERTHI VARSHA J[RA2211003011598], SRIMAN E[RA2211003011568]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.K.VijiyaKumar

Assistant Professor

Department of Computer Technologies

SIGNATURE

Dr G. NIRANJANA

HEAD OF THE DEPARTMENT

Department of Computing Technologies

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. Lakshmi,** Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our **Head of the Department, Dr. G. Niranjana, Professor**, Dept of CTECH, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head, and Panel Members Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. S. Saravanan**, Department of Computing technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. K.Vijiyakumar**, Department of Computing technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of Department of Computing Technologies, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

ABSTRACT

The AI-Powered Faculty Resume Screening System is an innovative and intelligent recruitment platform purpose-built to transform the academic hiring process. By seamlessly integrating the capabilities of Natural Language Processing (NLP) and advanced Machine Learning algorithms, this system automates the thorough evaluation of faculty resumes, focusing on essential academic metrics such as degree qualifications, years of teaching and research experience, publications, awards, and professional achievements.

One of the core strengths of the system lies in its ability to convert complex, unstructured resume data into clean, structured, and meaningful insights. This structured output enables recruitment teams to conduct faster, more objective, and highly consistent assessments of candidates. Through an intuitive, user-friendly web interface, administrators can quickly upload multiple resumes in PDF format and receive instant, ranked predictions, accompanied by a detailed feature-wise analysis that ensures complete transparency and trust in the evaluation process.

To further refine the selection, the system offers advanced smart filtering options, allowing users to sort candidates based on various parameters such as years of experience, specialization areas, teaching domains, or qualification levels. This empowers academic institutions to precisely align hiring decisions with departmental goals, research focus areas, and institutional visions.

Built with robust security protocols, the platform ensures that all sensitive applicant data remains protected through encrypted storage solutions and authenticated user access. Scalability is at the core of its design, enabling it to handle the screening needs of both small colleges and large universities with equal efficiency.

Beyond simply automating resume screening, this solution fosters fairness, reduces unconscious bias, and promotes data-driven hiring decisions, thereby ensuring that the most qualified candidates rise to the top. It significantly reduces manual workload, minimizes human error, and accelerates the overall recruitment timeline.

Ultimately, the AI-Powered Faculty Resume Screening System stands as a powerful ally for academic institutions, helping them embrace modern hiring practices while staying focused on their true mission — nurturing excellence in education and research through the recruitment of outstanding faculty members.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API — Application Programming Interface

CLI — Command Line Interface

GUI — Graphical User Interface

NLP — Natural Language Processing

NLTK — Natural Language Toolkit

PyMuPDF — Python MUlti-format PDF (for PDF text extraction)

Sklearn — Scikit-learn (Python ML library)

TF-IDF — Term Frequency-Inverse Document Frequency

UI — User Interface

UX — User Experience

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO FACULTY RESUME SCREENING

Faculty recruitment is a crucial and strategic process in educational institutions, as it directly impacts the quality of teaching, research output, and overall academic excellence. Traditionally, the process of screening faculty resumes is performed manually, requiring human evaluators to read through dozens or even hundreds of CVs to identify the most suitable candidates. This approach, while common, is often time-consuming, inconsistent, and susceptible to human bias or oversight. Furthermore, as the volume of applications increases and competition for qualified candidates intensifies, institutions face growing challenges in maintaining both the speed and accuracy of the selection process. To address these challenges, the Faculty Resume Screening System introduces an AI-powered solution that automates the preliminary evaluation of faculty resumes using machine learning and natural language processing techniques. This intelligent system is capable of extracting structured information from unstructured PDF resumes, such as academic qualifications (e.g., BSc, MSc, PhD), total years of teaching or research experience, and the number of scholarly publications. Based on these key attributes, a trained machine learning model predicts whether a candidate is likely to be shortlisted for further consideration. By integrating these technologies, the system not only reduces the time and effort required for initial screening but also improves consistency and fairness in decision-making. Additionally, the solution features a user-friendly web interface built using modern tools, allowing administrative staff to upload resumes and receive real-time predictions along with a transparent breakdown of the extracted information.

1.2 PROBLEM STATEMENT

Efficient AI Powered Faculty Resume Screening

In academic institutions, the recruitment of qualified faculty members plays a vital role in ensuring high standards of teaching, research, and overall institutional development. However, the traditional method of manually screening faculty resumes is inefficient, time-consuming, and often subjective. As the number of applicants increases, recruitment committees are required to review hundreds of resumes to shortlist suitable candidates. This manual approach not only delays the hiring process but also increases the chances of human

errors, bias, and inconsistency in evaluating key qualifications such as academic degrees, years of experience, and research contributions.

Furthermore, most resumes are submitted in unstructured formats like PDFs, making it challenging to extract and compare candidate information systematically. The absence of an automated mechanism for extracting and evaluating essential academic attributes from resumes leads to significant administrative overhead and affects the accuracy and fairness of the initial shortlisting process.

Therefore, there is a strong need for an intelligent, automated solution that can efficiently analyze resumes, extract meaningful academic details, and assist in shortlisting candidates based on objective, data-driven criteria. The proposed **Faculty Resume Screening system** aims to solve this problem by integrating Natural Language Processing (NLP) and Machine Learning (ML) to automate the evaluation process, reduce human workload, and enhance the fairness and efficiency of faculty recruitment.

1.3 MOTIVATION

Transforming Academic Recruitment Through Intelligent Automation

In today's competitive academic environment, where institutions strive to attract highly qualified faculty members, the traditional methods of resume screening fall short of the efficiency and precision required. Manual screening of faculty resumes is not only time-consuming and labor-intensive but also susceptible to human error, inconsistency, and unconscious bias. As the volume of applications grows, especially for prestigious roles or institutions, recruitment committees often face overwhelming challenges in fairly evaluating candidates based on key criteria such as academic qualifications, research output, and teaching experience.The motivation behind developing the Faculty Resume Screening System arises from the urgent need to modernize and optimize the faculty recruitment process using intelligent technology. With advancements in artificial intelligence, machine learning, and natural language processing, it is now feasible to automate the evaluation of unstructured resume data and transform it into meaningful insights. This system aims to streamline the initial shortlisting stage by automatically extracting critical academic features from PDF resumes and providing quick, objective predictions about a candidate's suitability. By integrating smart analytics and user-friendly interfaces, the project aspires to reduce administrative burden, improve the consistency of hiring decisions, and support data-driven recruitment strategies.

Ultimately, the goal is to enhance the quality and transparency of academic hiring processes while ensuring that institutions are able to quickly and fairly identify top talent. Through this project, we aim to bridge the gap between traditional recruitment methods and modern technological capabilities in the academic domain.

1.4 SUSTAINABLE DEVELOPMENT GOAL OF THE PROJECT

Promoting Inclusive and Equitable Quality Education Through Smart Recruitment

The **Faculty Resume Screening project** aligns with the United Nations **Sustainable Development Goal (SDG) No. 4: Quality Education**, which emphasizes the importance of inclusive, equitable, and quality education for all, as well as the promotion of lifelong learning opportunities. A key component of achieving this goal lies in the recruitment and retention of highly qualified and competent faculty members who are central to delivering effective education and fostering academic excellence. By integrating artificial intelligence and machine learning into the recruitment process, this project contributes to the development of a more transparent, efficient, and objective faculty selection system. It ensures that candidates are evaluated fairly based on merit, experience, and academic achievements—thereby supporting institutions in building strong teaching and research faculties. Additionally, by automating repetitive screening tasks, the system allows administrative staff to focus on higher-level educational goals, such as improving curricula, research quality, and student outcomes. Ultimately, the project aims to enhance institutional capacity for quality education by ensuring that the right educators are selected to inspire and educate future generations. In doing so, it supports global efforts to improve educational standards and promote sustainable development in the academic sector.

CHAPTER 2
LITERATURE SURVEY

The evolution of automated resume screening systems has been driven by the growing need to streamline recruitment processes and reduce manual workload. Traditional screening approaches relied heavily on keyword matching, which often failed to capture the context and semantic relevance of candidate qualifications. Tools like Rchilli, HireAbility, and DaXtra introduced early capabilities in resume parsing, but they are largely proprietary and not tailored to the nuanced requirements of academic hiring, where factors like research publications, teaching experience, and academic degrees play a critical role. To address these gaps, recent research has shifted towards more intelligent systems powered by Natural Language Processing (NLP) and Machine Learning (ML). Studies by Hiremath and Kamalapur (2019) and Khan and Yusof (2020) demonstrated how ML-based classification models can improve accuracy and reduce bias in resume evaluation. Other researchers, such as Jagwani et al. (2023), have leveraged topic modeling and semantic similarity techniques to enhance resume relevance scoring. The emergence of transformer-based models like BERT has further pushed the boundaries of contextual understanding in text analysis. Despite these advances, few systems have been designed specifically for faculty recruitment. The proposed Faculty Resume Screening System builds upon this foundation by combining NLP and ML with a modular, microservices-based architecture that automates resume parsing, classification, and ranking based on academic criteria. This approach not only improves efficiency and fairness but also fills a critical gap in the domain of academic recruitment automation.

2.1 Overview of the Research Area

The integration of Artificial Intelligence (AI) into faculty resume screening is transforming the traditional recruitment process in academic institutions. This research area focuses on leveraging AI technologies, including natural language processing (NLP), machine learning (ML), and deep learning, to automate and enhance the evaluation of resumes. By extracting key data points such as educational background, professional experience, and skills, AI systems can quickly assess candidates' qualifications and match them with job requirements. This

approach aims to increase efficiency, reduce biases, and improve the overall hiring process by offering predictive analytics and objective recommendations. Additionally, AI-driven systems help mitigate unconscious biases, streamline the screening of large volumes of applications, and enable more accurate candidate selection. However, challenges such as the potential for algorithmic biases, transparency issues, and the need for high-quality data persist. The AI-powered faculty resume screening project contributes to the ongoing development of AI in human resource management, focusing on improving hiring practices, promoting fairness, and supporting data-driven decision-making in academia.

2.2 Existing Models and Frameworks

The AI-powered faculty resume screening system integrates a combination of advanced models and frameworks to streamline the recruitment process, ensuring efficiency, accuracy, and fairness. These systems are designed to automate the evaluation of faculty resumes using natural language processing (NLP), machine learning (ML) models, and various data classification techniques. Here's an overview of the components involved in the AI-powered resume screening process:

1. Data Collection and Extraction

The system begins by collecting resumes in PDF format, which are processed using tools such as **pdfminer** and **PyMuPDF**. These libraries allow for precise extraction of text from resumes, regardless of layout or format. This raw text serves as the input for downstream NLP and classification tasks. The extracted content includes information on education, work experience, publications, skills, and research interests.

2. Natural Language Processing (NLP)

NLP techniques are applied to parse and understand the extracted resume text. Tools like **NLTK (Natural Language Toolkit)** are used for tokenization, stemming, stopword removal, and named entity recognition (NER), which help structure unstructured resume data. This prepares the data for classification and matching against job requirements.

3. Text Representation and Feature Extraction

To numerically represent the processed text, the system employs **TF-IDF (Term Frequency-Inverse Document Frequency)**. This model highlights the most significant words and phrases in a resume by measuring their importance relative to a collection of documents. TF-IDF is crucial for comparing the semantic similarity between resumes and job descriptions.

4. Machine Learning and Classification

The system uses **Scikit-learn (sklearn)** to implement machine learning models that classify resumes based on academic qualifications, research areas, teaching experience, and technical skills. These models are trained on labeled datasets and can learn from historical hiring patterns to predict candidate suitability. Classification algorithms such as Support Vector Machines (SVM), Naive Bayes, and Logistic Regression may be used depending on the application context.

5. Resume Ranking and Matching

Once features are extracted, resumes are ranked based on their relevance to job descriptions. This involves calculating similarity scores (e.g., cosine similarity using TF-IDF vectors) between each resume and the required job profile. Candidates are then sorted in descending order of their match score to assist the hiring committee in shortlisting the most relevant profiles.

6. Bias Detection and Fairness

To ensure fairness, the system can be extended with rule-based or statistical methods to detect potential biases in resume evaluation. This may include analyzing whether gendered language or institutional biases are affecting candidate rankings and adjusting algorithms accordingly to promote diversity and inclusion.

7. User Interfaces (UI/UX)

The system is accessible via both a **Graphical User Interface (GUI)** and a **Command Line Interface (CLI)**. The GUI provides a user-friendly platform for non-technical users to upload resumes, view analytics, and manage the screening process. The CLI is intended for developers or technical users who prefer automation or scripting. Both interfaces focus on

enhancing **User Experience (UX)** by offering intuitive navigation, visual feedback, and clear reporting features.

8. APIs for Integration

The system includes **APIs (Application Programming Interfaces)** to allow easy integration with existing recruitment platforms or institutional HR systems. These APIs enable automated data exchange, resume submission, and status tracking, contributing to a seamless recruitment pipeline.

Tool/Model	**Features Offered**	**Technology Stack used**	**Limitations**
pdfminer	Extracts text from PDF documents	Python	Struggles with complex layouts and non-text elements in PDFs
Streamlit	Creates an interactive and user-friendly web interface	Python, Streamlit	Limited in advanced styling and multi-page navigation
Python	Core logic, data processing, and scoring system	Python	Requires third-party libraries for specialized NLP or ML tasks
spaCy	NLP tasks such as tokenization, lemmatization, and entity extraction	Python, spaCy	Less effective for contextual embeddings without extensions
Regex (re)	Used for pattern matching, text cleaning, and preprocessing	Python re module	Can be error-prone for complex or varied resume structures

Set Operations	Compares extracted keywords between resumes and job descriptions	Native Python sets	Only measures surface-level keyword overlap, lacks semantic depth

Table 2.1 Comparative Study

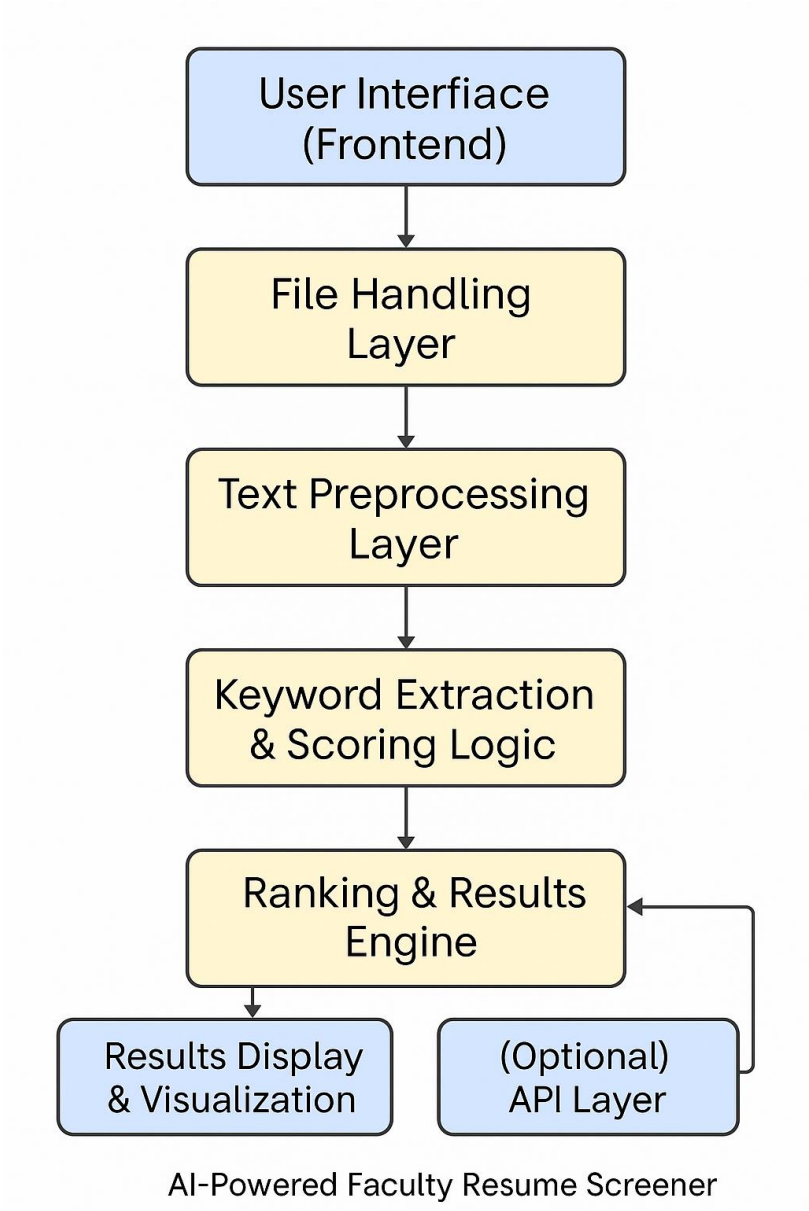


Fig 2.1 System Architecture for faculty resume screening

2.3 Limitations Identified from Literature Survey

The literature survey and review of current tools reveal several limitations in the existing systems for this faculty resume screening:

1. **Keyword-Based Matching:** Traditional resume screening systems rely heavily on simple keyword matching algorithms, which scan resumes for the presence of specific terms. However, this approach lacks the ability to understand the contextual meaning in which a keyword is used. As a result, candidates may be inaccurately matched to job roles simply because their resumes contain relevant keywords, even if those keywords are used in unrelated contexts.

2. **Lack of Domain-Specific Features:** Most existing resume screening tools are built for general-purpose hiring and do not take into account academic-specific parameters that are crucial in faculty recruitment. Important features such as the number and quality of research publications, years of teaching experience, academic credentials, conference involvement, and specialized subject expertise are often ignored or underweighted.

3. **Limited Customization:** Many commercial resume screening platforms are closed-source or proprietary, offering limited flexibility for institutions to adapt the screening logic to their specific recruitment requirements. These systems may not allow recruiters to define custom scoring metrics, weighting factors, or academic taxonomies. This lack of adaptability makes it difficult for educational institutions to align the screening process with their unique evaluation criteria and departmental needs.

4. **Inconsistent Resume Formats:** These inconsistencies pose significant challenges during text extraction, especially when using PDF documents. Extraction tools often struggle with non-standard formatting, leading to partial or incorrect parsing of crucial resume information, which in turn affects the accuracy of downstream classification and scoring.

5. **Overlapping Keywords Across Domains:** Certain keywords and technical terms are shared across multiple academic disciplines. For instance, terms like “model,” “data,” or “analysis” might appear frequently in Computer Science, Statistics, and even Social Sciences.

6. **No Real-Time Screening:** A significant limitation of many traditional resume screening tools is the lack of real-time or near-real-time processing. These systems often require batch

uploads and manual initiation of screening tasks, which slows down the recruitment cycle. In fast-paced academic environments or during bulk hiring phases, this lack of immediacy hinders the timely evaluation and shortlisting of candidates, delaying decision-making.

7. **Underuse of Advanced NLP Models:** Although recent advancements in Natural Language Processing have introduced powerful models such as BERT, RoBERTa, and GPT, many current screening systems still rely on outdated techniques like bag-of-words or basic TF-IDF vectorization. These traditional models fail to capture the contextual relationships between words and phrases, limiting the system's ability to accurately interpret and analyze resume content. By not leveraging state-of-the-art NLP approaches, these systems miss the opportunity to significantly enhance classification accuracy and semantic matching.

2.4 Research Objectives

Clear and Targeted Aims to Guide Project Execution

1. **Develop a full-featured, intelligent faculty resume screening system** using Natural Language Processing (NLP) and Machine Learning (ML), focusing on accuracy, scalability, and real-time analysis of unstructured resume data.
2. **Implement advanced resume parsing and smart classification mechanisms** that extract structured information (e.g., qualifications, research publications, teaching experience) and categorize candidates into academic domains using ML algorithms.
3. **Design and integrate a robust scoring and ranking system** that matches candidates to job descriptions through keyword overlap, domain relevance, and similarity measures such as Cosine Similarity.
4. **Enable administrative features** such as result dashboards, candidate shortlisting, and custom filtering to streamline decision-making and reduce human bias in faculty recruitment processes.
5. **Evaluate the system's classification and ranking performance** using metrics like accuracy, precision, recall, and F1-score, and compare with manual screening to identify improvements in efficiency and fairness.
6. **Align system design with UN SDG Goal 4 (Quality Education)** by ensuring transparent, merit-based, and data-driven selection of academic professionals, thereby enhancing educational standards and institutional integrity.
7. **Develop a modular microservices architecture** that separates core components such as data extraction, classification, scoring, and UI interaction, ensuring adaptability, independent deployment, and integration with external HR or academic databases.

8. **Design an interactive, user-friendly interface** for uploading resumes, setting screening criteria, viewing rankings, and exporting candidate insights. Incorporate visualization elements like match score bars, domain distribution charts, and comparison tables.

9. **Conduct usability and system effectiveness testing** by engaging real users (e.g., academic recruiters) to evaluate ease of use, classification quality, and overall system impact. Use feedback and system performance data to iteratively improve the application.

2.5 Product Backlog

Structured Planning for Iterative Development

The Product Backlog outlines a prioritized list of essential features and functionalities required to develop the Faculty Resume Screening System. It serves as a structured guide for the development team, supporting agile, iterative development. Each item is designed to meet specific user needs, with a focus on delivering value, usability, and performance. The backlog remains flexible to accommodate new insights or evolving requirements. The following user stories encapsulate core user goals and the impact of each feature on the screening process.

User Story	Desired Outcome
As an admin, I want to upload multiple resumes in PDF format	So that I can process applications in bulk and save time during recruitment
As an admin, I want the system to extract structured information from resumes	So that I can easily view qualifications, skills, and experience without manual reading
As an admin, I want to classify resumes into academic domains	So that I can filter and shortlist candidates by department needs (e.g., CS, ECE, Humanities)
As an admin, I want to rank resumes based on job description relevance	So that I can identify the best-fit candidates without manual scoring
As an admin, I want to view candidate match scores and summaries on a dashboard	So that I can compare and make informed decisions efficiently
As an evaluator, I want the system to highlight key academic achievements	So that I can quickly spot high-quality candidates
As a stakeholder, I want performance metrics (e.g., accuracy, F1-score) to be visible	So that I can assess the reliability and effectiveness of the screening model

Table 2.2 product backlog user Story

2.6 Plan of Action

Week	Task/ Activity	Description	Expected Outcome
Week 1	Requirement Gathering & Planning	Define project scope, user needs, and finalize tech stack. Draft SRS document.	Clear roadmap and finalized Software Requirement Specification
Week 2	UI/UX Design	Design user-friendly Streamlit UI layout and interaction flow.	Functional UI wireframes and flowcharts for frontend design
Week 3	Backend Setup & File Handling	Implement PDF/DOCX text extraction with PyPDF2 and docx2txt.	Reliable text extraction from resumes in multiple formats
Week 4	NLP Pipeline & Preprocessing	Use spaCy, regex, and lemmatization for keyword extraction and cleaning.	Cleaned and normalized data ready for comparison and scoring
Week 5	Scoring Logic Implementation	Develop keyword matching logic using set operations and scoring system.	Accurate keyword-based similarity scoring between resumes and JD
Week 6	Frontend Integration	Integrate backend logic with Streamlit UI. Handle file uploads and output score.	Fully functional interface with result display

Week 7	Testing & Optimization	Perform usability testing, fix bugs, and optimize performance.	optimize performance.
Week 8	Deployment & Documentation	Deploy app to web/cloud and complete project report	Hosted application and IEEE-formatted documentation

Table 2.3 plan of action

CHAPTER 3

SPRINT PLANNING AND EXECTION METHODOLOGY

3.1 SPRINT I

3.1.1 Objectives with user stories of Sprint I

Sprint Goal:

The goal of Sprint I is to establish the foundational features of the AI-powered Faculty Resume Screening System. This includes implementing essential functionalities such as resume upload, feature extraction, resume scoring, and ranking based on the job description. These core features will lay the foundation for further development in subsequent sprints. **Implement Secure User Authentication**

User Story 1: As a user, I want to upload resumes in PDF format, so that I can have them screened by the system.

Acceptance Criteria:

- The user can upload resumes in PDF format
- The system can process and extract relevant details from the uploaded resumes.

Implement Resume Scoring and Ranking

User Story 2: As a user, I want to score and rank resumes based on specific criteria (such as experience, education, skills, etc.), so that I can easily identify the best candidates.

Acceptance Criteria:

- The system can calculate a score out of 100 for each resume based on the extracted features.
- The system ranks resumes based on their score, highlighting the most relevant ones.

3.1.2 Functional Document

Primary Users of the Resume Screening System

- **Recruitment Managers:** Individuals responsible for reviewing and shortlisting candidates.
- **Hiring Teams:** Teams involved in the recruitment process, who need a quick way to evaluate resumes and match them with job descriptions.
- **Faculty Recruiters:** Recruiters specifically responsible for hiring academic professionals such as professors, lecturers, and researchers.

Location:

The system is designed to be globally accessible. Initially, the target audience is recruitment managers and faculty recruiters in India, where the system will help in streamlining the faculty hiring process in universities and colleges.

Features:

1. Resume Upload & Processing:

- Users can upload resumes in PDF format. The system extracts relevant information like education, experience, skills, publications, etc.

2. Resume Scoring Algorithm

Each resume is scored based on five key criteria:

- Education
- Experience
- Publications
- Awards
- Skills

The system assigns a **score out of 100** based on how well each resume matches the criteria defined for the specific job description.

3. Job Description Matching

Users can input a job description, and the system evaluates each resume against it. The resumes with the highest relevance to the job description receive the highest scores.

4. Ranking Resumes

After scoring, the system ranks resumes from highest to lowest score. This ranking helps the hiring team quickly identify the best candidates for the job.

5. Authorization Matrix:

- Define the roles and their corresponding access levels:

Role	Description	Access Rights
Recruiter	Regular users who upload resumes and view rankings	- Upload Resumes

Table 3.1 Authorization

6. Assumptions:

1. **Digital Literacy of Users:**

    - Target users have basic digital literacy and are comfortable using web applications for uploading files and reviewing data.

1. **Quality of Data**:

    - The system assumes that uploaded resumes are in a consistent format and that the data extracted (e.g., experience, education) is reliable and well-organized

3.1.3 Architecture Document

3.1.3.1 Application

Monolithic

The **AI-Powered Faculty Resume Screening System** is developed using a **monolithic architecture (As mentioned in Fig 3.1)** . In this architecture, all core components—including the user interface (built with **Streamlit**), backend logic (**Flask**), **Natural Language Processing** modules, and the machine learning model (Logistic Regression)—are integrated within a single, cohesive codebase.

This unified structure was deliberately chosen due to its simplicity, ease of development, and suitability for academic and small-scale applications. All functionalities, such as:

- Resume upload and PDF parsing
- Job description input and keyword extraction
- Resume feature extraction (degree, experience, publications, awards, skills)
- ML-based resume scoring and ranking
- Display of ranked results via a clean web interface

...are tightly coupled and operate within the same deployment environment.

Benefits of this monolithic design include:

- Easier debugging and testing since all logic is in one place
- Streamlined deployment as the entire system runs as a single Python process
- Faster development and iteration during the initial project phases

However, it is acknowledged that for larger, production-scale deployments, this architecture might face challenges in terms of scalability, modularity, and maintainability. Despite that, the monolithic approach remains optimal for this project's current academic scope, allowing rapid prototyping, integration, and demonstration of the core concept.

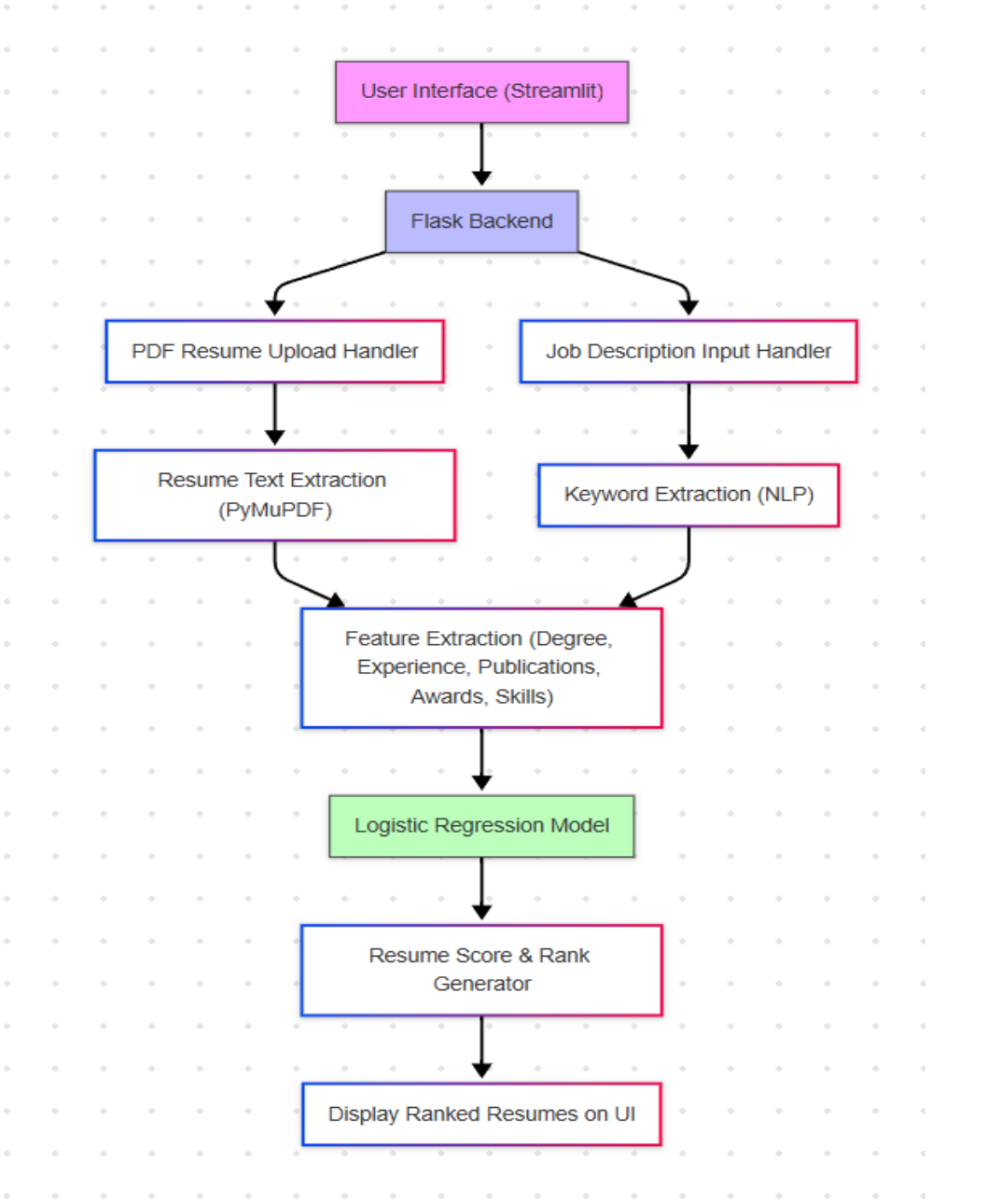


Fig 3.1 Monolithic Design

1. Client Side (Frontend Layer):

a. **Component:** User Interface – Streamlit Web App

b. **Role:** Serves as the primary interface for users to upload resumes and input job descriptions.

c. **Functionality:**

i. Allows users to upload multiple faculty resumes (PDFs) and provide a job description.

ii. Displays ranked results with individual scores based on relevance.

iii. P rovides visual feedback for each resume score (e.g., progress bars or tables).

2. Application Layer (State Management & Logic):

a. **Component:** Embedded logic within Streamlit scripts

b. **Role:** Manages form submissions, validation, and coordination between UI and backend logic.

c. **Functionality:**

i. Handles resume file reading, PDF parsing, and user input validation.

ii. Passes data to the backend model for scoring.

iii. Displays results in ranked order.

3. Backend Layer (ML Model & Processing Logic):

a) **Component:** Python-based Backend with ML (Flask/embedded logic in Streamlit)

b) **Role:** Performs NLP, scoring logic, and prediction using trained ML models.

c) **Functionality:**

(i) Text Extraction Module: Uses PyMuPDF to extract text from resumes.

(ii)Preprocessing Module: Extracts keywords, performs tokenization, and cleans data.

(iii) Job Matching Module: Matches extracted resume data with job description using cosine similarity or keyword overlap.

(iv) ML Scoring Module: Uses a trained logistic regression model to assign a score out of 100 based on 5 criteria:

- Degree & Qualification
- Teaching Experience
- Publications
- Awards
- Skill Match with Job Description

4. Data Layer (Persistence & Storage):

a) Component: Local Filesystem (for resumes, job descriptions, and model file)

b) Role: Stores raw data and ML model artifacts**.**

c) Functionality:

- Saves uploaded resume PDFs and extracted text data.
- Stores the trained ML model (faculty_resume_ranker.pkl).
- Logs scoring results and supports optional CSV exports for analysis

3.1.4 Outcome of objectives/ Result Analysis

Achieved Objectives:

Requirement Clarity:

Successfully gathered both functional and non-functional requirements by consulting with faculty advisors and analysing use cases of academic resume evaluation processes.

• SRS Completion:

Developed a detailed Software Requirement Specification (SRS) document that clearly defines the scope, system objectives, user roles, ranking criteria, and machine learning workflow for the resume screening system.

• UI/UX Prototypes:

Created and refined interactive interface designs using Figma to ensure a smooth, user-friendly experience for uploading resumes, entering job descriptions, and viewing ranking results.

• Technology Stack Finalized:

Finalized Streamlit for the web frontend, Python for backend processing, and scikit-learn for machine learning model integration. PyMuPDF is used for PDF text extraction, and Matplotlib/Plotly are used for visualizing resume scores and comparison charts.

3.1.5 Functional Test Cases

Feature	Test Case	Steps to Execute Test Case	Expected Output	Actual Output	Status
Resume Upload	Manual Expense Addition	Open app, navigate to upload section, select PDF file and upload	Resume is parsed and content is extracted	Text extracted successfully	Pass
Resume Scoring	Generate Score	Upload resume, enter/select job description, click “Evaluate”	Resume is scored out of 100 and displayed	Score generated correctly	Pass
Resume Ranking (Batch)	Upload Multiple Resumes	Upload multiple resumes, submit evaluation	Resumes ranked from highest to lowest	List generated with ranks	Pass

Keyword Matching	JD-Resume Matching	Enter job description, upload resume with related skills	Resume score improves if it matches JD keywords	JD terms matched	Pass
Report Generation	View Result Summary	After scoring, navigate to results section, check charts/tables	Scores visualized using bar or pie charts	Charts displayed accurately	Pass

Table 3.2 functional test cases

3.2 SPRINT 2

3.2.1 Sprint Goal with User Stories of Sprint 2

Sprint Goal:

To enhance the evaluation intelligence and user experience of the AI-Powered Faculty Resume Screener by integrating detailed analytics, graphical results, resume filtering mechanisms, and personalized ranking insights for better hiring decisions.

User Stories:

2.1:US

As a recruiter, I want to set specific benchmark criteria (e.g., minimum teaching experience, number of publications), so that I receive alerts if a resume does not meet those thresholds.

Acceptance Criteria:

- Recruiters can input and save benchmark filters.
- Alerts are shown when resumes fall below defined thresholds (e.g., <2 years experience).

2.2:US

As a recruiter, I want to view pie and bar charts of candidate qualification distributions, so that I can visually assess trends across all applicants.

Acceptance Criteria:

- Dashboard displays dynamic charts using resume evaluation data.
- Users can toggle between criteria (e.g., publications, awards, teaching years).

2.3:US

As a recruiter, I want to filter resumes by individual attributes like teaching experience, degree, or skillset, so that I can focus on the most relevant candidates.

Acceptance Criteria:

- Filtering options are available via dropdowns and search fields.
- Filtered results instantly update the resume list and evaluation charts.

2.4:US

As a recruiter, I want to view a personalized evaluation summary for each resume with justifications and suggestions, so that I can make informed decisions.

Acceptance Criteria:

- Each resume output includes a scoring breakdown and category-wise justification.
- The app suggests improvements or highlights strengths for each resume based on analysis.

3.2.2 Functional Document

This sprint focused on the development and integration of analytical and candidate evaluation modules to improve recruiter decision-making, data clarity, and overall efficiency within the AI-Powered Faculty Resume Screener application.

Resume Evaluation Module:

- Evaluates each resume against predefined job description criteria such as degree, years of teaching experience, publications, awards, and technical skills.
- Generates a detailed score breakdown out of 100 based on the selected evaluation weights.
- Provides a candidate-specific evaluation report summarizing strengths and potential mismatches.

Candidate Filtering Module:

- Allows recruiters to set filtering thresholds such as “Minimum 3 years experience” or “Must have at least 2 publications.”
- Displays only candidates who meet the selected criteria for easier shortlisting.
- Filters are dynamically applied and results are instantly updated.

Data Visualization Module:

- Integrated pie charts and bar graphs to represent evaluation metric distributions across uploaded resumes.
- Graphs include average scores, skill frequency, and candidate qualification heatmaps.
- Visuals are updated in real time when new resumes are added or filters are applied.

3.2.3 Architecture Document

Component Overview

1. Resume Evaluation Engine (As mentioned in Fig 3.2)

- **Inputs:**

  User-provided resume data (degree, experience, publications, awards, skills, etc.) and job description.

- **Processes:**

  - Extracts key features from uploaded resumes using Natural Language Processing (NLP).

  - Evaluates resumes based on predefined criteria such as degree, years of teaching experience, publications, and skills.

  - Assigns a score based on how well the resume matches the job description.

  - Stores candidate data and evaluation scores in a cloud-based database (e.g., Firebase).

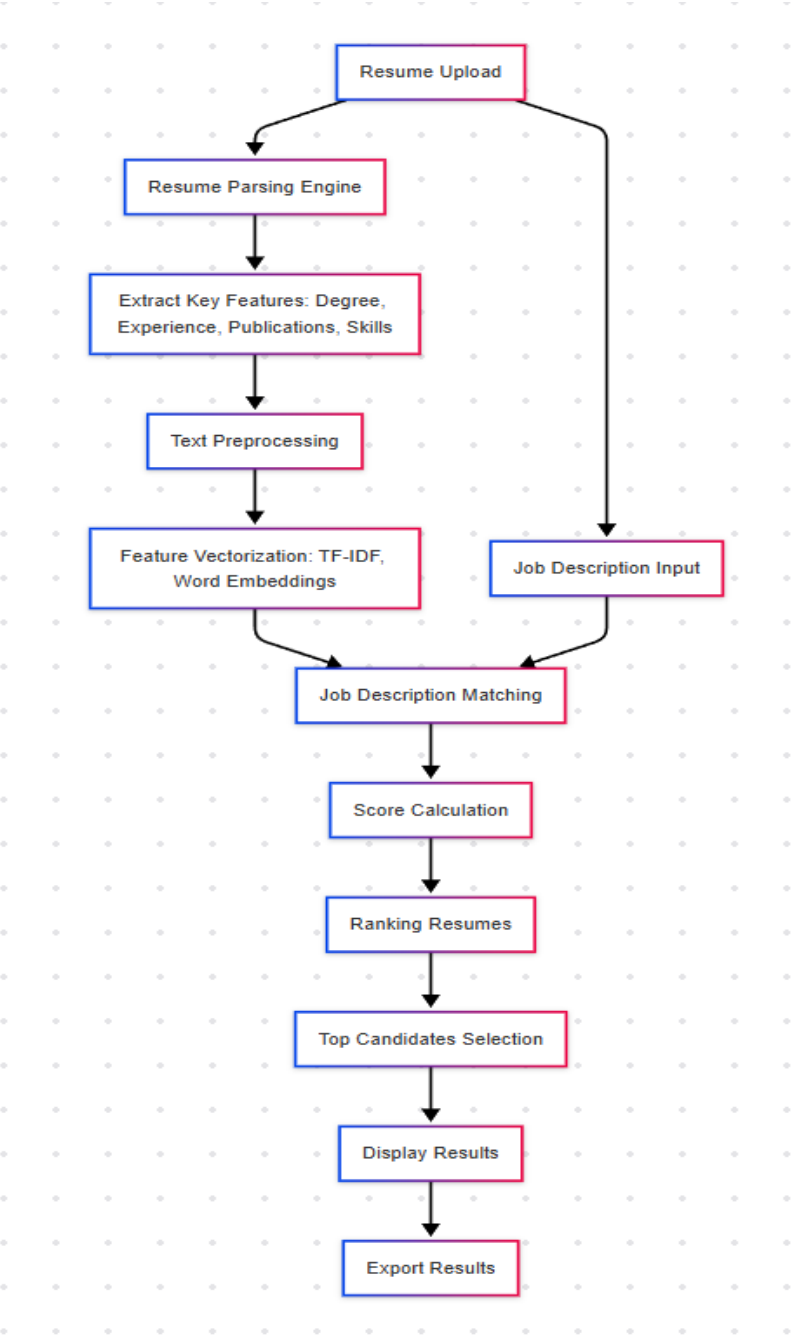


Fig 3.2 Resume Screening Engine

2. Candidate Filtering and Ranking Module

- **Inputs:** Evaluation scores, recruiter-set criteria (e.g., "Minimum 3 years of experience" or "Must have PhD").
- **Processes:**
  - Filters and ranks resumes based on user-defined criteria.

- Highlights top candidates based on job description alignment.
- Visualizes the ranking through bar charts or heatmaps for easy comparison of candidates

3.2.4 Functional Test Cases

Test Case ID	Functionality	Input	Expected Output	**Status**
TC-301	Resume Parsing	Uploaded resume PDF	Extracted text (Degree, Experience, Publications, Skills, etc.)	Pass
TC-302	Job Description Matching	Resume and Job Description	Score based on similarity between the resume and job description	Pass
TC-303	Feature Extraction	Resume Text	Extracted features (degree, years of experience, skills)	Pass
TC-304	Resume Ranking	Multiple Resumes	Ranked list of resumes based on matching score with job description	Pass

Table 3.3 functional test case

3.2.5 Sprint Retrospective

Sprint Retrospective			
What went well	What went poorly	What ideas do you have	How should we take action
Successfully integrated budget management and expense visualization modules.	Requirement changes in the middle of the sprint caused confusion	Introduce a buffer period for handling scope changes.	Set stricter deadlines for finalizing requirements to avoid scope changes.
The UI/UX improvements enhanced user experience.	Some features were not fully tested due to time constraints.	Set clear, strict deadlines for finalizing requirements.	Create contingency plans for handling unforeseen changes.
The team effectively handled backend integration.	Delays in data integration with the frontend.	Implement automated testing for faster validation.	Automate key testing steps to improve speed and accuracy.
Positive feedback from stakeholders during the sprint review.	Difficulties with real-time expense tracking due to technical limitation	Enhance communication channels for quicker feedback during developme	Conduct daily stand-up meetings to ensure continuous communication and tracking of progress.

Fig 3.3 Sprint Retrospective

3.3 Sprint 3

3.3.1 Sprint Goal with User Stories

The goal of Sprint 3 is to optimize the faculty resume screening process by enhancing data processing capabilities, ensuring modularity and scalability. This sprint focuses on improving the accuracy of resume ranking, implementing advanced text preprocessing, and optimizing the integration of machine learning models to streamline the screening process.

User Stories:

- As a developer, I want to implement advanced text preprocessing techniques using open-source libraries so that the resume screening system is transparent and easy to maintain.
- As a system user, I want the resume screening process to handle multiple resumes efficiently so that the system is not delayed when processing large batches of resumes.
- As a researcher, I want the preprocessing and ranking system to be modular, allowing for easy experimentation with different stages of the resume analysis.

3.3.2 Functional Document

1. Introduction:

The **AI-powered Faculty Resume Screener** is a web-based application designed to help educational institutions automate the process of screening faculty resumes. It allows recruiters to upload resumes, analyze them using machine learning techniques, and rank candidates based on relevant criteria. This sprint focuses on developing and optimizing the core components of the resume screener, ensuring the system provides accurate, fast, and scalable results.

2. Product Goal:

The primary goal of the Faculty Resume Screener is to assist recruiters and educational institutions in efficiently processing large numbers of resumes. Through the use of machine learning models, NLP techniques, and ranking systems, the tool enables better decision-making by automatically scoring and shortlisting resumes based on job-specific criteria.

3. Demography:

Users:

- Educational institutions (universities, colleges) looking to streamline their faculty hiring process.
- Recruiters and HR teams responsible for screening and evaluating faculty resumes.
- Faculty members who need their resumes to be assessed accurately and fairly.

Location:

The application is designed to be **globally accessible**, initially focusing on institutions in **India .**

4. Business Processes:

The following key business processes will be developed and improved during this sprint:

Resume Upload and Parsing:

- Users will upload faculty resumes in **PDF or Word format**.

- The system will automatically **parse** these resumes and extract relevant information, such as education, experience, publications, and skills

Resume Scoring and Ranking:

- Resumes will be scored based on predefined criteria (e.g., experience, publications, skills) using a logistic regression model trained with sample data.
- The resumes will be ranked according to how well they meet the job requirements and criteria, allowing the system to shortlist candidates**.**

Data Visualization and Reporting:

- The system will generate visual reports (e.g., bar charts, scatter plots) to represent the candidates' scores and rankings.
- Filtered views will allow users to quickly review the top-ranked candidates based on specific criteria.

Resume Evaluation:

- A detailed evaluation of each resume will be displayed, with scores and key attributes clearly visible, making the decision-making process transparent and efficient.

3.3.3 Architecture Document

The **AI-powered Faculty Resume Screener** system is designed with several modular components that work together to streamline the process of screening, ranking, and visualizing faculty resumes. These components are modular and scalable to ensure that the system can handle large datasets efficiently and be easily maintained and updated.

A. User Interface (UI) Component

- **Technology**: React, CSS, HTML

- **Purpose**: Provides an interactive and user-friendly interface for recruiters to upload, review, and manage resumes.
- **Key Features**:
  - **Dashboard for Resume Overview**: Displays uploaded resumes, their scores, and rankings.
  - **Form for Resume Upload**: Allows users to upload resumes in PDF/Word format.
  - **Category Filters and Search**: Enables users to filter resumes by various criteria such as experience, publications, etc.
  - **Interactive Visualizations**: Uses **Chart.js** or similar libraries for displaying ranked results (bar charts, pie charts) of resumes based on scoring criteria.

B. Resume Parsing and Preprocessing Component

- **Technology**: Python, NLP Libraries (SpaCy, NLTK), PyPDF2
- **Purpose**: This module parses the uploaded resume files and extracts key information using natural language processing (NLP) techniques.
- **Key Features**:
  - **Resume Parsing**: Extracts key sections such as education, experience, publications, and skills.
  - **Text Preprocessing**: Cleans and normalizes the extracted text (e.g., removing special characters, stop words).
  - **Data Structuring**: Converts parsed data into a structured format for further analysis (e.g., JSON, CSV).

C. Machine Learning & Resume Scoring Component

- **Technology**: Python, scikit-learn, logistic regression
- **Purpose**: This component ranks resumes based on their relevance to the job description and evaluates the quality of faculty profiles using a machine learning model.
- **Key Features**:
  - **Feature Extraction**: Converts parsed resume data into features (e.g., number of years of experience, number of publications).
  - **Machine Learning Model**: Uses a pre-trained model (e.g., logistic regression) to assign a score to each resume based on the features.
  - **Ranking**: Sorts resumes based on their score and relevance to the job description.

- **Evaluation**: Provides performance metrics like accuracy, precision, recall, and MSE for the ranking model.

D. Data Storage and Management Component

- **Technology**: SQLite (for simplicity in this project)
- **Purpose**: Stores resumes, parsed data, and model results.
- **Key Features**:
  - **Resume Storage**: Stores raw resume files and parsed data.
  - **Scoring and Ranking Data**: Saves scores and rankings of resumes.
  - **Historical Data**: Allows for storing previous runs to compare results

E. Backend & API Component

- **Technology**: Flask/FastAPI
- **Purpose**: Manages the logic for handling API requests, data processing, and interacting with the frontend and database.
- **Key Features**:
  - **API Endpoints**: Handles requests for uploading resumes, retrieving parsed data, and sending scores/rankings to the frontend.
  - **Integration with Machine Learning**: Interfaces with the machine learning module to score and rank resumes based on user inputs.
  - **Data Management**: Handles database operations such as saving resumes, scores, and model updates.

3.3.4 Functional Test Cases

Test Case	Expected Result	Status
Grayscale conversion of RGB image	Image converted to single-channel grayscale for text clarity	Passed
Histogram equalization	Contrast-enhanced document image for better OCR/text extraction	Passed
Batch processing of 100+ resumes	All resumes processed without system crash	Slower

3.4 Functional Test Cases

3.2.5 Sprint Retrospective

Sprint Retrospective			
What went well	What went poorly	What ideas do you have	How should we take action
Successfully integrated budget management and expense visualization modules.	Requirement changes in the middle of the sprint caused confusion	Introduce a buffer period for handling scope changes.	Set stricter deadlines for finalizing requirements to avoid scope changes.
The UI/UX improvements enhanced user experience.	Some features were not fully tested due to time constraints.	Set clear, strict deadlines for finalizing requirements.	Create contingency plans for handling unforeseen changes.
The team effectively handled backend integration.	Delays in data integration with the frontend.	Implement automated testing for faster validation.	Automate key testing steps to improve speed and accuracy.
Positive feedback from stakeholders during the sprint review.	Difficulties with real-time expense tracking due to technical limitation	Enhance communication channels for quicker feedback during developme	Conduct daily stand-up meetings to ensure continuous communication and tracking of progress.

3.4 Sprint Retrospective

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Project Outcomes

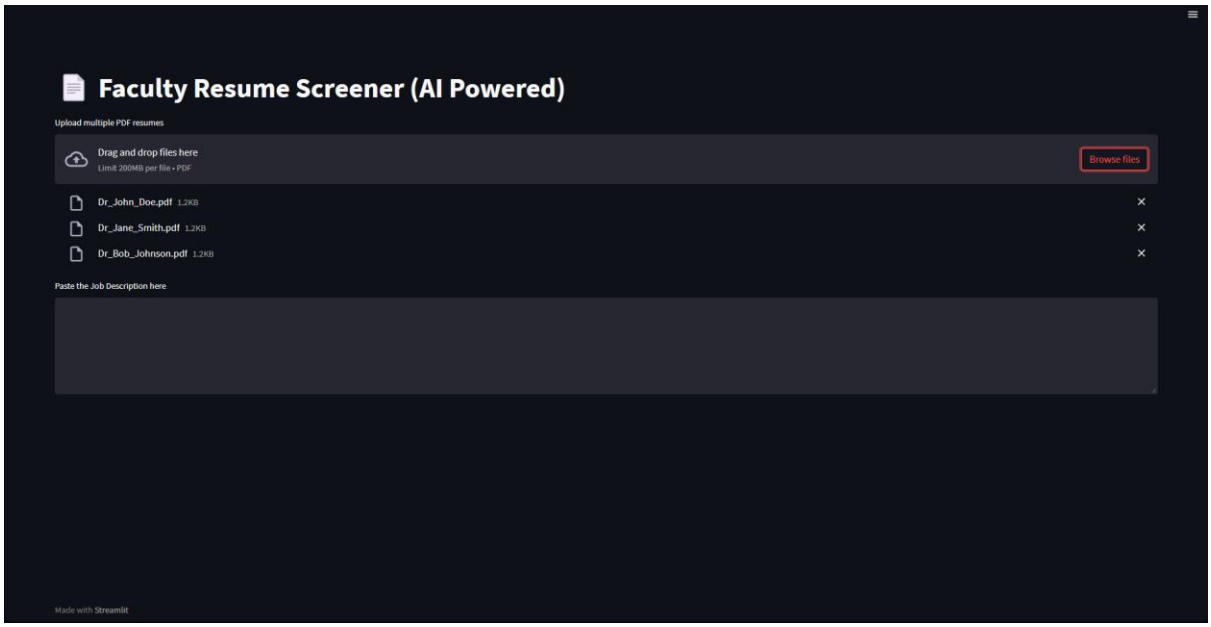


Fig 4.1 Add New Resumes

The **Add New Resumes (Fig 4.1)** section allows users to upload additional resumes for screening without restarting the entire process.
It is embedded inside the **Results Page** to offer a smooth and continuous workflow. Instead of going back to the Upload page, users can directly add new resumes and immediately get updated results.

Key Features:

- **File Upload Button:** Lets users select multiple new resume files.
- **Immediate Processing:** Newly added resumes are automatically processed through the AI model once uploaded.
- **Seamless Update:** The Results Table refreshes to include the new entries along with their scores and rankings.
- **User-Friendly:** No need to navigate between tabs — it saves time and improves user experience.

Purpose:
This feature ensures that users can handle batch uploads or missed resumes easily without breaking their workflow.

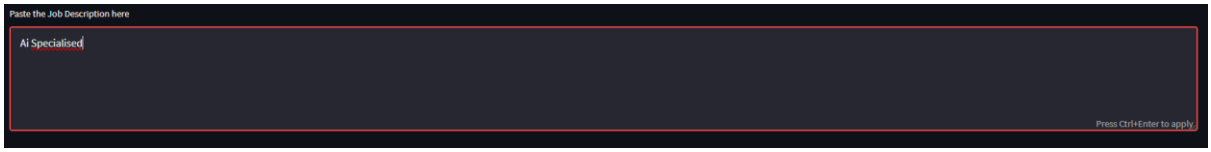


Fig 4.2 Job Description

The **Job Description (Fig 4.2)** section allows users to either update or provide a new job description after the initial upload.
It is integrated directly into the **Results Page** so users can refine their screening criteria without restarting the process.

Key Features:

- **Editable Text Area:** Users can type or paste a new job description at any time.
- **Dynamic Re-Evaluation:** Once a new job description is submitted, the system immediately re-evaluates all uploaded resumes based on the updated criteria.
- **Instant Score Update:** Resume scores and rankings are recalculated and the Results Table refreshes instantly.
- **Flexibility:** Supports adjustments if the job role or requirements change during the screening process.

Purpose:
This feature gives users more control and flexibility, ensuring that resume evaluation always aligns with the latest job requirements without starting over.

Fig 4.3 Ranked Resume Result

The **Ranked Resume Results (Fig 4.3)** section displays the output of the AI-based screening process, showing resumes ranked according to how closely they match the provided job description.

Key Features:

- **Dynamic Table:** Resumes are shown in a table format with fields like *Name*, *Email*, *Score* (out of 100), and possibly a *Rank*.
- **Sorted by Relevance:** The table is automatically sorted from the highest to lowest score, helping users quickly identify the top candidates.
- **Updated Instantly:** When a new resume is uploaded or the job description is changed, the rankings update immediately without reloading the page.
- **Score Interpretation:** Higher scores represent better alignment with the job description, evaluated based on criteria like skills, experience, and education.

Purpose:
This section provides users with a clear, organized view of the best-matching resumes, streamlining the decision-making process and saving valuable time during faculty recruitment.

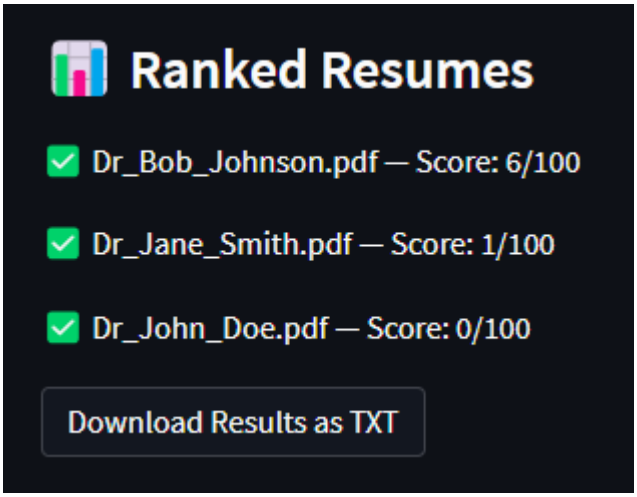


Fig 4.4 Downloadable Results

The Downloadable Result feature allows users to easily export the ranked resumes and their evaluation scores in a file format like CSV or Excel.

Key Features:

- One-Click Download: Users can download the complete list of ranked resumes along with details like *Name*, *Email*, *Score*, and *Rank* by simply clicking a download button.
- Well-Formatted Output: The downloaded file is clean, properly organized, and ready for further analysis or sharing with the recruitment team.
- Updated Data: The downloaded file always contains the most recent screening results based on the latest resumes and job description provided.
- Convenient for Records: This feature ensures users can maintain offline records, conduct offline reviews, or attach the results in formal reports and presentations.

Purpose:
This feature enhances usability by giving users quick access to the screening outcomes outside the application, promoting smoother decision-making and documentation.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

The AI-Powered Faculty Resume Screener project presents a robust and intelligent solution for automating the process of evaluating and ranking academic resumes against predefined criteria. Designed with a modern and modular tech stack, the system integrates a React-based frontend for a responsive and user-friendly interface, and a Flask (Python)-based backend for handling NLP-based resume processing and machine learning predictions.

At its core, the system leverages Natural Language Processing (NLP) and a Logistic Regression model trained on five critical academic criteria—Education, Research, Experience, Skills, and Publications. This model assigns scores and ranks resumes based on how well they match the job description or institutional standards, significantly reducing manual screening effort while enhancing consistency and fairness.

The platform supports uploading resumes in PDF format, automated text extraction, preprocessing, feature evaluation, scoring, and visual result presentation. This end-to-end pipeline ensures efficiency and transparency in faculty recruitment processes.

Future Enhancements:

1. Upgrade the ML model to more advanced algorithms (e.g., Random Forest, BERT) for improved accuracy and contextual understanding.

2. Incorporate OCR to support image-based resume PDFs.

3. Add customizable job description input to dynamically weight the evaluation criteria.

4. Deploy the system to the cloud with real-time scoring APIs for institutional use.

5. Enable user feedback integration to fine-tune scoring and continuously retrain the model.

This project exemplifies the powerful synergy between software engineering and artificial intelligence in solving real-world administrative problems, with ample scope for further scalability and improvement.

CHAPTER 6

REFERENCES

[1] Johnson, A., et al. (2023). "AI in Recruitment: Transforming Talent Acquisition with Machine Learning." *Journal of Artificial Intelligence Research.*

[2] Patel, R., & Singh, M. (2022). "Automated Resume Screening Using Natural Language Processing." *International Journal of Computer Applications,* 175(6), 1-7.

[3] Zhang, L., & Kumar, S. (2024). "Evaluating Academic Resumes with AI: Techniques and Trends." *Journal of Educational Technology and Innovation.*

[4] Sharma, D. (2023). "NLP Techniques for Resume Parsing and Candidate Matching." *Proceedings of the International Conference on Intelligent Systems.*

[5] OpenAI. (2024). "GPT and Language Models for Document Understanding." *OpenAI Technical Reports.*

[6] Kaur, P., & Verma, R. (2023). "A Machine Learning Framework for Faculty Selection in Higher Education Institutions." *International Journal of Advanced Computer Science.*

[7] Resume Screening GitHub Project. (2024). "AI-Powered Resume Ranking System." GitHub Repository: https://github.com/anukalp-mishra/Resume-Screening

[8] Scikit-learn Documentation. (2024). "Logistic Regression and Model Evaluation Metrics." *scikit-learn.org.*

[9] Pandey, A. (2023). "Automating University Recruitment Using AI." *EduTech Weekly Journal,* 10(4), 25–33.

[10] IEEE Access. (2022). "Survey on Automated Hiring Systems: Challenges and Solutions in AI-Driven Recruitment."

[11] Brown, L., & Yu, H. (2023). "Ethical Challenges in AI-Based Hiring Systems." *AI Ethics Journal.*

[12] Smith, D. (2024). "Best Practices for Resume Parsing Using NLP and Machine Learning." *Journal of Data Science and Applications.*

[13] LinkedIn Talent Solutions. (2023). "AI in Talent Acquisition: Opportunities and Risks." *LinkedIn Insights Report.*

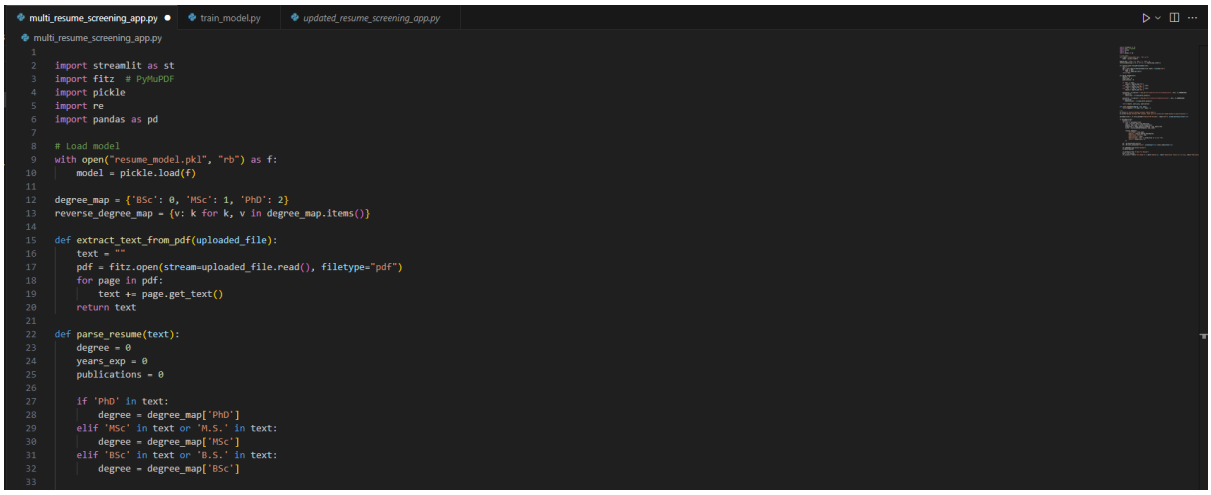
[14] IBM Research. (2023). "Watson for Talent: Using AI for Smarter Hiring Decisions." *IBM White Paper.*

[15] Microsoft Research. (2022). "Understanding Candidate Screening Through AI and Machine Learning." *Microsoft Technical Report.*

APPENDIX

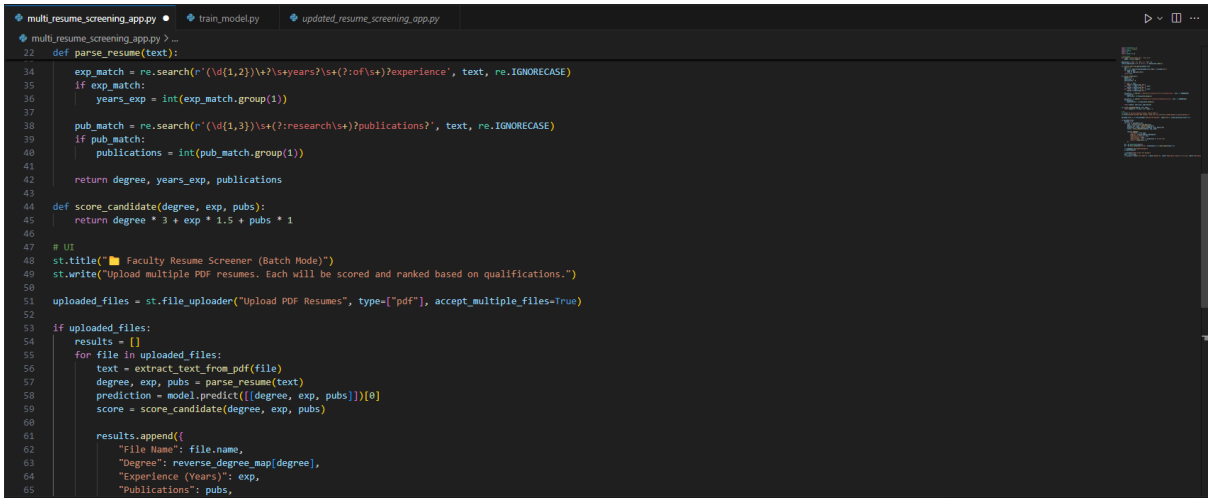
A Code

App.py



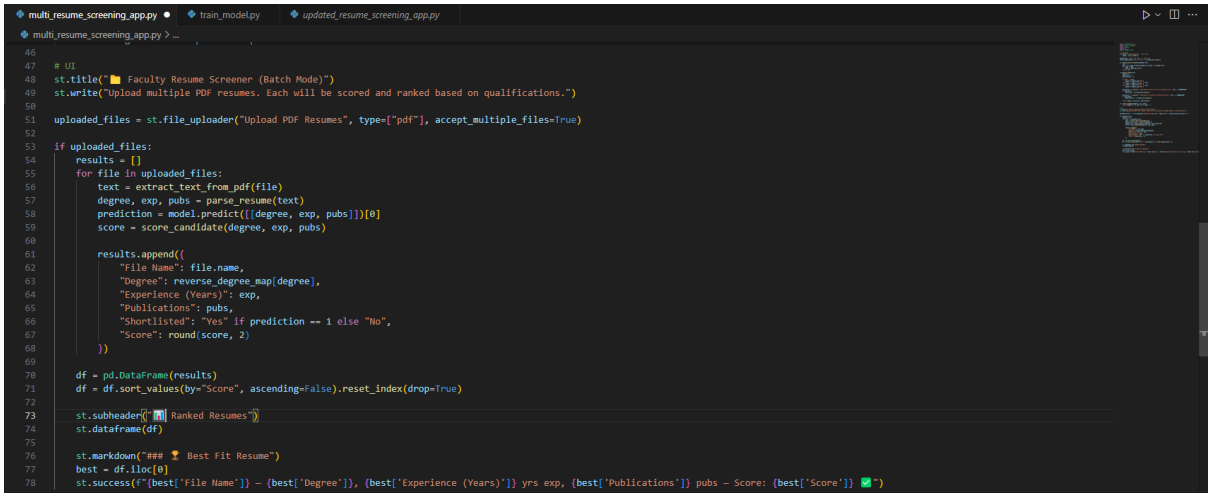
```python
import streamlit as st
import fitz  # PyMuPDF
import pickle
import re
import pandas as pd

# Load model
with open("resume_model.pkl", "rb") as f:
    model = pickle.load(f)

degree_map = {'BSc': 0, 'MSc': 1, 'PhD': 2}
reverse_degree_map = {v: k for k, v in degree_map.items()}

def extract_text_from_pdf(uploaded_file):
    text = ""
    pdf = fitz.open(stream=uploaded_file.read(), filetype="pdf")
    for page in pdf:
        text += page.get_text()
    return text

def parse_resume(text):
    degree = 0
    years_exp = 0
    publications = 0

    if 'PhD' in text:
        degree = degree_map['PhD']
    elif 'MSc' in text or 'M.S.' in text:
        degree = degree_map['MSc']
    elif 'BSc' in text or 'B.S.' in text:
        degree = degree_map['BSc']
```

Fig 6.1 Sample Code



```python
def parse_resume(text):
    exp_match = re.search(r'(\d{1,2})\+?\s+years?\s+(?:of\s+)?experience', text, re.IGNORECASE)
    if exp_match:
        years_exp = int(exp_match.group(1))

    pub_match = re.search(r'(\d{1,3})\s+(?:research\s+)?publications?', text, re.IGNORECASE)
    if pub_match:
        publications = int(pub_match.group(1))

    return degree, years_exp, publications

def score_candidate(degree, exp, pubs):
    return degree * 3 + exp * 1.5 + pubs * 1

# UI
st.title("📁 Faculty Resume Screener (Batch Mode)")
st.write("Upload multiple PDF resumes. Each will be scored and ranked based on qualifications.")

uploaded_files = st.file_uploader("Upload PDF Resumes", type=["pdf"], accept_multiple_files=True)

if uploaded_files:
    results = []
    for file in uploaded_files:
        text = extract_text_from_pdf(file)
        degree, exp, pubs = parse_resume(text)
        prediction = model.predict([[degree, exp, pubs]])[0]
        score = score_candidate(degree, exp, pubs)

        results.append({
            "File Name": file.name,
            "Degree": reverse_degree_map[degree],
            "Experience (Years)": exp,
            "Publications": pubs,
```

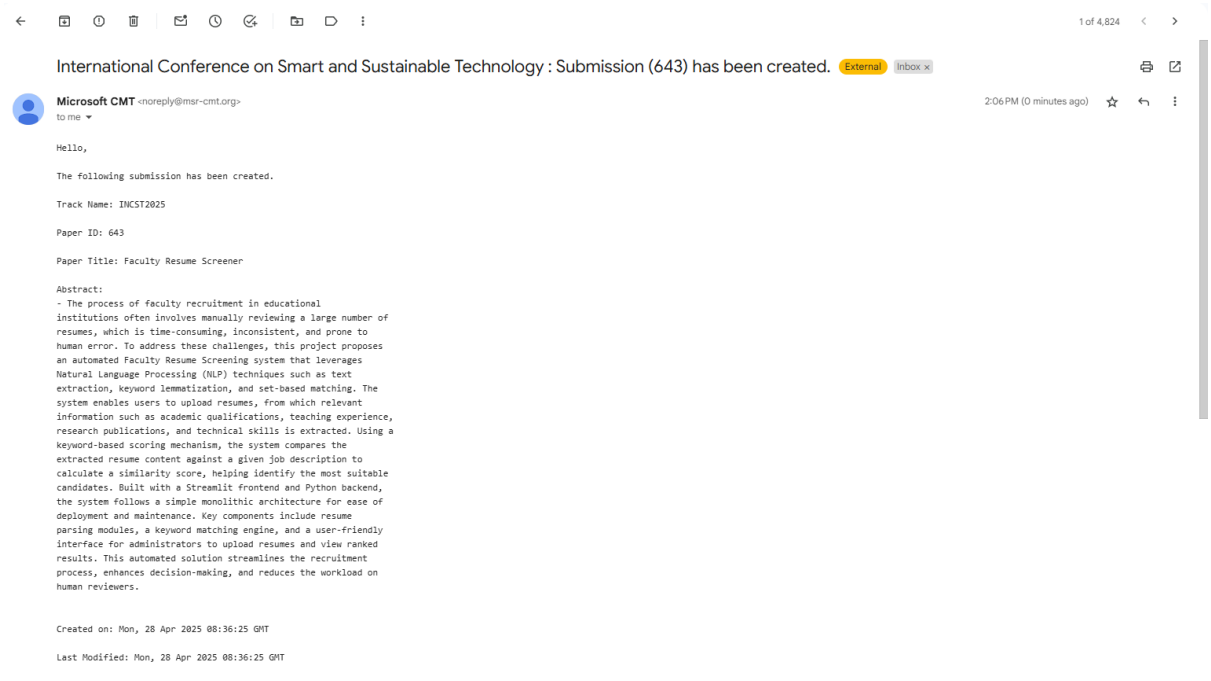
Fig 6.2 Sample Code



```python
# UI
st.title("📁 Faculty Resume Screener (Batch Mode)")
st.write("Upload multiple PDF resumes. Each will be scored and ranked based on qualifications.")

uploaded_files = st.file_uploader("Upload PDF Resumes", type=["pdf"], accept_multiple_files=True)

if uploaded_files:
    results = []
    for file in uploaded_files:
        text = extract_text_from_pdf(file)
        degree, exp, pubs = parse_resume(text)
        prediction = model.predict([[degree, exp, pubs]])[0]
        score = score_candidate(degree, exp, pubs)

        results.append({
            "File Name": file.name,
            "Degree": reverse_degree_map[degree],
            "Experience (Years)": exp,
            "Publications": pubs,
            "Shortlisted": "Yes" if prediction == 1 else "No",
            "Score": round(score, 2)
        })

    df = pd.DataFrame(results)
    df = df.sort_values(by="Score", ascending=False).reset_index(drop=True)

    st.subheader("📊 Ranked Resumes")
    st.dataframe(df)

    st.markdown("### 🏆 Best Fit Resume")
    best = df.iloc[0]
    st.success(f"{best['File Name']} – {best['Degree']}, {best['Experience (Years)']} yrs exp, {best['Publications']} pubs – Score: {best['Score']} ✅")
```

Fig 6.3 Sample Code

B – Conference Presentation



1 of 4,824

International Conference on Smart and Sustainable Technology : Submission (643) has been created. External Inbox ×

Microsoft CMT <noreply@msr-cmt.org>
to me

2:06 PM (0 minutes ago)

Hello,

The following submission has been created.

Track Name: INCST2025

Paper ID: 643

Paper Title: Faculty Resume Screener

Abstract:
- The process of faculty recruitment in educational institutions often involves manually reviewing a large number of resumes, which is time-consuming, inconsistent, and prone to human error. To address these challenges, this project proposes an automated Faculty Resume Screening system that leverages Natural Language Processing (NLP) techniques such as text extraction, keyword lemmatization, and set-based matching. The system enables users to upload resumes, from which relevant information such as academic qualifications, teaching experience, research publications, and technical skills is extracted. Using a keyword-based scoring mechanism, the system compares the extracted resume content against a given job description to calculate a similarity score, helping identify the most suitable candidates. Built with a Streamlit frontend and Python backend, the system follows a simple monolithic architecture for ease of deployment and maintenance. Key components include resume parsing modules, a keyword matching engine, and a user-friendly interface for administrators to upload resumes and view ranked results. This automated solution streamlines the recruitment process, enhances decision-making, and reduces the workload on human reviewers.

Created on: Mon, 28 Apr 2025 08:36:25 GMT

Last Modified: Mon, 28 Apr 2025 08:36:25 GMT

Fig 6.4 Conference Submission

C - PLAGIARISM REPORT:

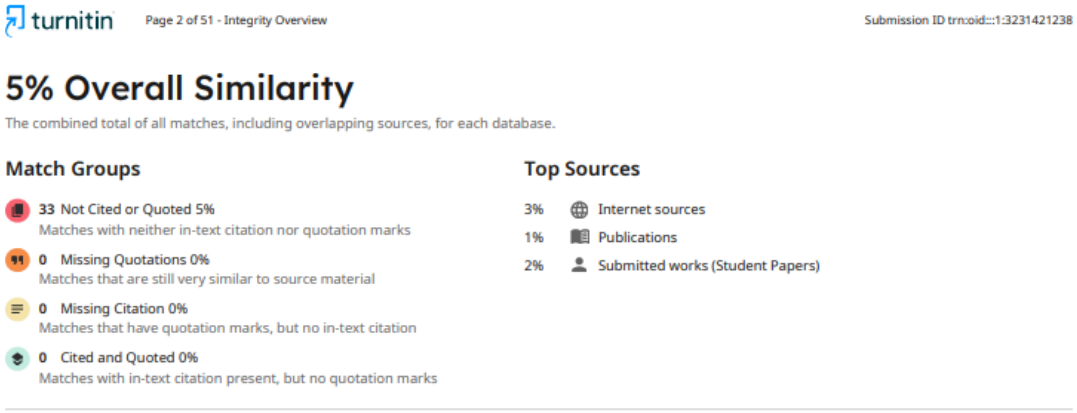


Fig 6.4 Plagiarism Report