HALDIA INSTITUTE OF TECHNOLOGY

Department of Information Technology

Haldia, Purba Medinipur, West Bengal – 721657

Session: 2026-2027

Project Report

On

Title of the Project:

AI RESUME SHORTLISTING SYSTEM

Submitted by

1. **GOURAV KUMAR**

University Roll No: 10300224219

2. **KOUSHIK BARMAN**

University Roll No: 10300224221

3. **MD. ALFAJ HUSSAIN**

University Roll No: 10300224224

Date: 13.02.2026

Acknowledgement

We would like to express our sincere gratitude to everyone who supported us in the successful completion of our project titled **“AI Resume Shortlisting System.”**

First and foremost, we are deeply thankful to our Training In-Charge, **Mr. Debasish Sahoo**, for his constant guidance, valuable suggestions, and technical support throughout the development of this project. His mentorship helped us understand the practical implementation of Machine Learning and Natural Language Processing concepts in real-world applications.

We also extend our heartfelt thanks to our faculty members and department for providing us with the opportunity, resources, and academic environment necessary to carry out this project successfully.

As a team of three members, we sincerely appreciate the collaborative effort, dedication, and coordination among us, which played a crucial role in completing this project efficiently. Working together not only enhanced our technical knowledge but also strengthened our teamwork and problem-solving skills.

We would also like to acknowledge the open-source community and the developers of Python, Flask, Scikit-learn, NLTK, GitHub, and Render for providing powerful tools and platforms that made the development, version control, and deployment of this project possible.

Finally, we are grateful to our families and friends for their continuous encouragement and support throughout this journey.

Abstract

The **AI Resume Shortlisting System** is a Machine Learning-based web application designed to automate the process of resume evaluation by comparing a candidate's resume with a given job description. The system uses Natural Language Processing (NLP) techniques such as text preprocessing, TF-IDF vectorization, and Cosine Similarity to compute a matching percentage score.

The application enables recruiters to upload a resume (PDF format) and paste a job description to instantly receive a similarity score indicating how well the resume matches the job requirements. The project is implemented using Python, Flask, Scikit-learn, and NLTK, version-controlled using GitHub, and deployed on Render for real-time access.

Contents

Introduction

Recruitment processes often involve manually screening hundreds of resumes for a single job role. This process is time-consuming, inconsistent, and prone to human bias.

The AI Resume Shortlisting System aims to:

- Automate resume screening
- Provide objective matching scores
- Reduce hiring time
- Improve recruitment efficiency

The system compares resume content with job descriptions using NLP and machine learning techniques to calculate a percentage match score.

Problem Statement

Manual resume screening presents several challenges:

- Large number of applicants per job posting
- Time-consuming manual comparison
- Human bias and inconsistency
- Difficulty in identifying the best match quickly

There is a need for an automated and intelligent system that evaluates resumes against job descriptions efficiently and accurately.

Objectives of the project

The main objectives of the project titled **“AI Resume Shortlisting System”** are as follows:

1. **To develop an automated resume shortlisting system** that compares a candidate’s resume with a given job description.
2. **To implement Natural Language Processing (NLP) techniques** for cleaning and preprocessing resume and job description text.
3. **To apply TF-IDF (Term Frequency–Inverse Document Frequency) vectorization** to convert textual data into numerical feature vectors.
4. **To compute similarity using Cosine Similarity** in order to generate an accurate resume–job matching percentage score.
5. **To design and develop a user-friendly web interface** using Flask and HTML/CSS for uploading resumes and entering job descriptions.
6. **To deploy the application on a cloud platform (Render)** for real-time accessibility and practical usability.
7. **To use GitHub for version control and project management**, ensuring proper code tracking and collaboration among team members.
8. **To reduce manual effort in recruitment** by providing a fast, objective, and efficient screening mechanism.
9. **To enhance understanding of Machine Learning implementation in real-world applications**, particularly in recruitment and HR systems.

System Overview

The system consists of:

1. **Frontend (HTML/CSS)**
2. **Backend (Flask - Python)**
3. **Machine Learning Model**
4. **Cloud Deployment (Render)**

Methodology

1. Resume Upload & Text Extraction

From `app.py`

- Resume is uploaded in PDF format
- PyPDF2 extracts text from the PDF
- Extracted text is passed to the ML model

```
resume_text = extract_text_from_pdf(resume_file)

score = match_resume(resume_text, job_desc)
```

2. Text Preprocessing (NLP)

From `model.py`

Steps performed:

1. Convert text to lowercase
2. Remove punctuation
3. Remove stopwords (NLTK)

4. Clean text for analysis

```
def clean_text(text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    words = text.split()
    words = [w for w in words if w not in stopwords.words("english")]
```

3. Feature Extraction – TF-IDF

TF-IDF (Term Frequency – Inverse Document Frequency) converts text into numerical vectors.

```
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([resume_clean, jd_clean])
```

TF-IDF ensures:

- Important words get higher weight
- Common words get lower weight

4. Similarity Calculation – Cosine Similarity

Cosine Similarity measures the angle between two text vectors.

```
similarity = cosine_similarity(vectors[0:1], vectors[1:2])
score = round(similarity[0][0] * 100, 2)
```

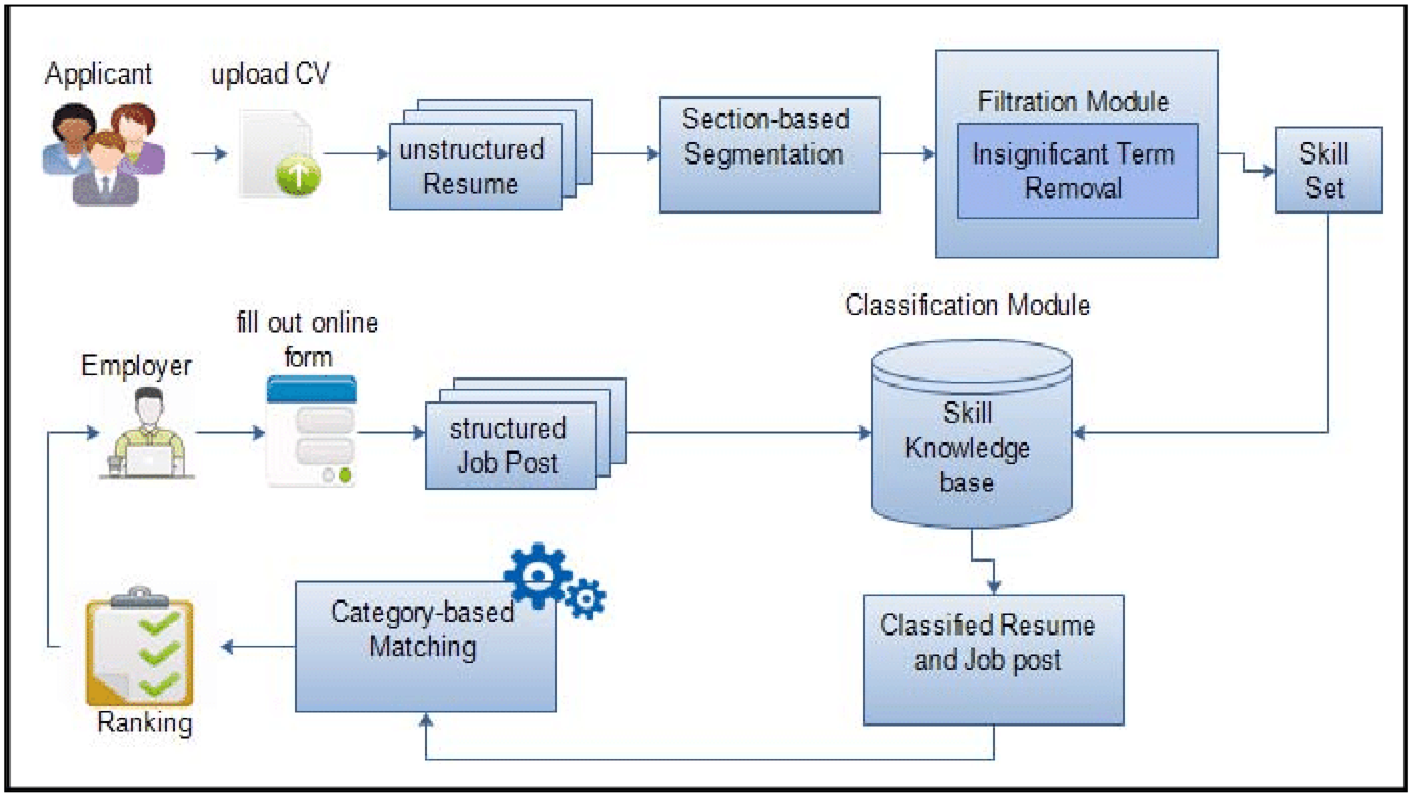
Output:

- Score between 0% to 100%
- Higher score → Better resume-job match

System Architecture

- User uploads Resume (PDF)
- User pastes Job Description
- PDF text extraction
- Text preprocessing (NLP)
- TF-IDF vectorization
- Cosine Similarity computation
- Display matching score

System Architecture Diagram:



Architectural Components:

Layer	Technology Used	Purpose
Frontend	HTML, CSS	User interaction
Backend	Flask (Python)	Request handling
NLP Module	NLTK	Text cleaning
ML Module	Scikit-learn	TF-IDF & similarity
PDF Processing	PyPDF2	Text extraction
Deployment	Render + Gunicorn	Cloud hosting
Version Control	GitHub	Code management

Deployment Architecture:

1. Code pushed to GitHub
2. Repository connected to Render
3. Render installs dependencies from requirements.txt

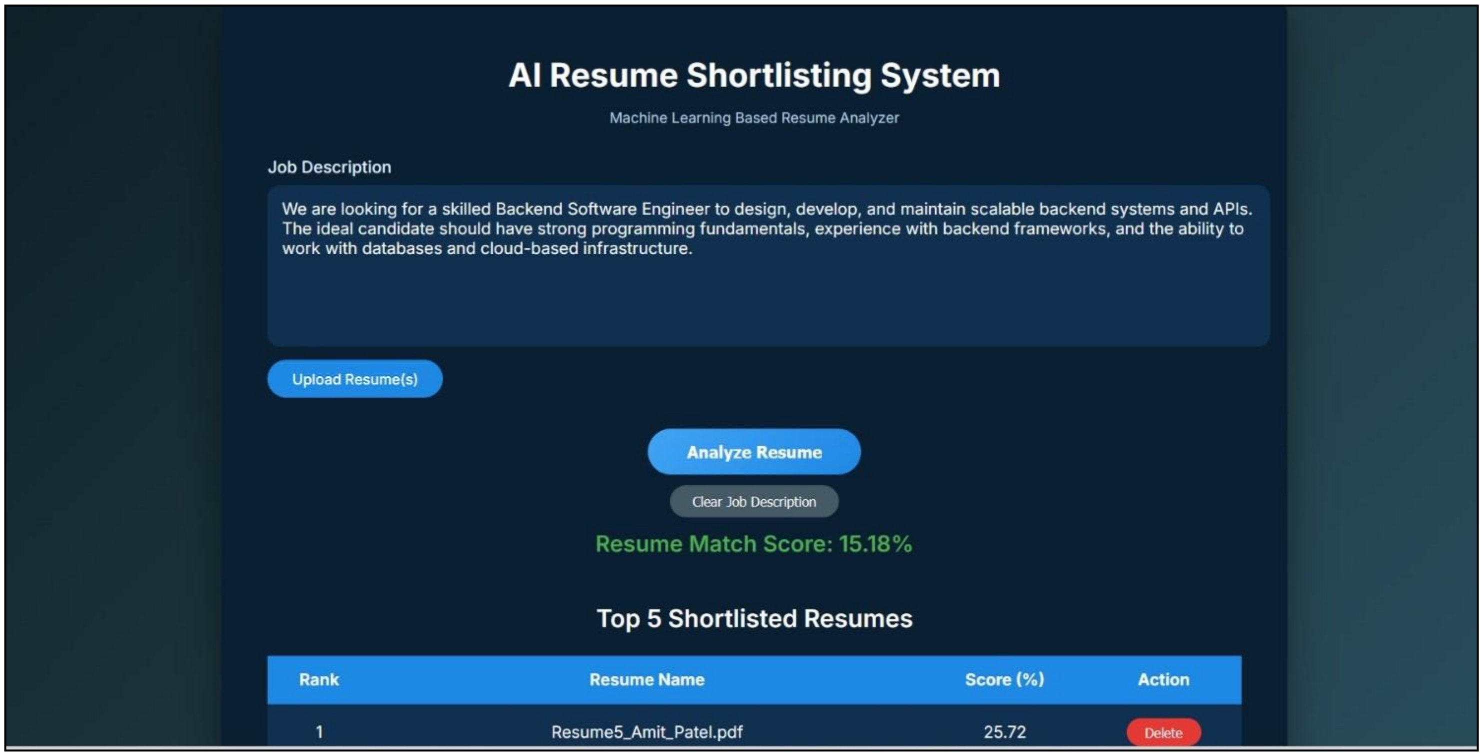
From `requirements.txt`:

- flask
- scikit-learn
- nltk
- PyPDF2
- gunicorn

4. Gunicorn runs Flask app
5. Application accessible via public URL

User Interface

The **User Interface (UI)** of the AI Resume Shortlisting System is designed to be modern, professional, and user-friendly. It allows recruiters or HR professionals to easily upload resumes, enter job descriptions, and view matching results in an organized format.



AI Resume Shortlisting System

Machine Learning Based Resume Analyzer

Job Description

We are looking for a skilled Backend Software Engineer to design, develop, and maintain scalable backend systems and APIs. The ideal candidate should have strong programming fundamentals, experience with backend frameworks, and the ability to work with databases and cloud-based infrastructure.

Resume Match Score: 15.18%

Top 5 Shortlisted Resumes

Rank	Resume Name	Score (%)	Action
1	Resume5_Amit_Patel.pdf	25.72	Delete

Interface Components Explanation:

1. Header Section

- Project Title: **AI Resume Shortlisting System**
- Subtitle: *Machine Learning Based Resume Analyzer*
- Professional dark-themed layout

This clearly identifies the system purpose.

2. Job Description Input Section

- Large text box for entering job description
- Allows recruiters to paste detailed role requirements

- Supports backend, frontend, or any domain-specific job role

Purpose:

To provide job requirement text that will be compared with uploaded resumes.

3. Resume Upload Section

- “Upload Resume(s)” button
- Supports PDF file format
- Multiple resume upload capability (as shown in interface)

Purpose:

Allows HR to upload candidate resumes for evaluation.

4. Analyze Resume Button

- “Analyze Resume” button triggers backend processing
- Sends resume and job description to Flask server
- Calls ML model for matching score calculation

5. Matching Score Display

- Displays: **Resume Match Score (e.g., 15.18%)**
- Highlighted in green for visibility
- Indicates similarity percentage between resume and job description

Purpose:

Shows how well the resume fits the job requirements.

6. Top 5 Shortlisted Resumes Table

The interface displays:

Rank	Resume Name	Score (%)	Action

Features:

- Ranks resumes based on highest similarity score
- Shows score percentage clearly
- Delete option available
- Helps HR quickly identify best candidates

UI Design Characteristics

- Dark professional theme
- Clean typography
- Button-based actions
- Responsive layout
- Clear result visualization
- Organized ranking system

UI Workflow

- Enter Job Description
- Upload Resume(s)
- Click Analyze Resume
- System processes data
- Matching Score displayed
- Top resumes ranked automatically
-

Advantages of the UI

- Easy to use (no technical knowledge required)
- Minimal input steps
- Instant result display
- Clear ranking structure
- HR-friendly dashboard design

GitHub Version Control

The project is uploaded to GitHub for:

- Source code management
- Version tracking
- Collaboration
- Backup
- CI/CD deployment support

Repository contains:

- templates

  -index.html

- app.py
- model.py
- requirements.txt
- README.md

Deployment on Render

The project is deployed on Render using:

- Gunicorn as WSGI server
- Environment-based port configuration
- Public URL for access

Deployment process:

1. Push code to GitHub
2. Connect GitHub repo to Render

3. Configure build command
4. Deploy automatically

Live system runs as a production web application.

Advantages

- Fast resume evaluation
- Objective comparison
- Reduces manual screening time
- Cloud accessible
- Lightweight and scalable
- Easy to maintain

Limitations

- Only supports PDF resumes
- Works best with text-based PDFs
- Does not understand deep semantic meaning
- No deep learning implementation
- Accuracy depends on job description clarity

Future Enhancements

- Add deep learning (BERT-based matching)
- Multi-resume bulk upload
- Ranking system for multiple candidates
- Resume keyword suggestion
- ATS (Applicant Tracking System) integration
- Database storage
- Role-based admin panel

Learning Outcomes

Through this project, the following concepts were learned:

- Natural Language Processing
- Text Cleaning & Preprocessing
- TF-IDF Vectorization
- Cosine Similarity
- Flask Web Development
- RESTful Application Flow
- Cloud Deployment (Render)
- GitHub Version Control
- Production Deployment using Gunicorn

Conclusion

The **AI Resume Shortlisting System** successfully demonstrates the practical implementation of Machine Learning and Natural Language Processing in the recruitment domain. The system automates the resume screening process by comparing uploaded resumes with job descriptions and generating a similarity-based matching score using TF-IDF vectorization and Cosine Similarity.

By reducing manual resume evaluation, the system improves efficiency, minimizes bias, and speeds up the shortlisting process. The web-based interface built using Flask provides a simple and user-friendly platform for recruiters to upload resumes, analyze candidates, and view ranked results instantly.

The project also highlights the integration of multiple technologies, including Python, Scikit-learn, NLTK, PyPDF2, GitHub for version control, and Render for cloud deployment. This end-to-end implementation reflects real-world software development practices, from model development to production deployment.

Overall, this project enhances recruitment efficiency and demonstrates how AI-driven systems can support decision-making in HR processes. With further improvements such as deep learning-based semantic analysis, database integration, and bulk processing features, the system can be extended into a fully scalable recruitment solution.